Test Specification

Rev. — 27 September 2023

Revision History

Revision	Date	Author	Description	Review
0.8.19	20230927	Bala Mannam	Test specification for 0.8.19	
0.8.18	20230823	Dalton Correya	Test specification for 0.8.18	

1 Introduction

This document contains the test procedures used for RFE testing, performed by the development team and by the independent test group.

It defines test specifications and classes, test cases, and pass/fail criteria used for performing testing activities.

The audience consists of Project Manager, Safety Manager, Software Developers, Software Testers, and Quality Assurance.

1.1 Overview

This document describes test specification for NXP SmartTRX Radar Front End software.

The scope of this test spec is to cover the test scenarios for the RFE requirements. The audience consists of Software Architects, Project Manager, Safety Architect, Software Developers, Software Testers and Quality Assurance

1.2 About this Manual

This Technical Reference employs the following typographical conventions:

Boldface type: Bold is used for important terms, notes and warnings.

Italic font: Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note: *This is a note.*

1.3 Acronyms and Definitions

Table 1. Acronyms and Definitions

Term	Definition
ARM64	Binary code run on ARM-64 bit CPU-compiler aarch64-fsl-linux
CPU	Central Processing Unit. In this context, it refers to the PowerPC e200.
FFT	Fast Fourier Transform
IUT	Implementation Under Test
RFE	Radar Front End
LSB	Least Significant Bit
N/A	Not Applicable
SPT	Signal Processing Toolbox
SUT	System Under Test
MCAL	Microcontroller Abstraction layer

1.4 Reference List

Table 2. Reference List

Item No	Title	Version
1		
2		

2 Test Environment and Constraints

2.1 Test Environment

Table 3. Software Items

Software Item Name	Description	Version
GHS	Green Hills Software	
GCC	GNU Compiler Collection	
DIAB	Wind River Diab Compiler	
S32DS	S32 Design Studio	
Cygwin	UNIX emulator for Windows	

Table 4. Hardware/Firmware Items

Hardware Item Name	Description	Remarks
Lauterbach Trace32 debugger	Lauterbach Trace32 debugger	
MPC5675KEVB257	MPC5675K EVB	
S32R45	System Reference board for racerunner IC	IPC testing across ARM cores shall make use of S32R45 boards
STRX system reference board (post silicon stage)	System Reference board for STRX IC, including the external on-board peripherals and the tooling to interface to a host (PC)	IPC testing across all the cores shall make use of STRX boards
Lauterbach Trace32 for Debug, image loading	LTB tracer	
ZeBu Emulation platform for STRX Digital Subsystem (pre-silicon stage)	STRX SoC Digital Subsystem emulation including all the peripherals and tooling to interface to a host (PC) except the RFE Analog IPs	App-M7 to RFE-M7 communication. (API calls, Cmd server and IPC)RFE-M7 Infra IP drivers Interface testing for Timing Engine, PDC, Packer and BIST DMA.
C Simulation environment RFE Simulation environment RFE Analog(AMS models), RFE Digital Access and Control	RFE Analog IPs models will be used which runs on PC.	Analog IP drivers, whole RF API functionality, Radar System cycle (Calibration, Frame and BIST) RFE-M7 Infra IP drivers.Drivers for Timing Engine, PDC, Packer and BIST DMA. RFE-M7 Infra IP drivers.Drivers for Timing Engine, PDC, Packer and BIST DMA.

Table 4. Hardware/Firmware Items*...continued*

Hardware Item Name	Description	Remarks
RTL Simulation environment RFE Simulation environment : RFE Analog (AMS models), RFE Digital Access and Control	RFE Analog IPs models will be used which runs on PC.	Analog IP drivers, whole RF API functionality, Radar System cycle (Calibration, Frame and BIST)RFE-M7 Infra IP drivers.Drivers for Timing Engine, PDC, Packer and BIST DMA.
FPGA Board for digital part of the RFE subsystem (pre-silicon stage)	STRX SoC RFE Digital IPs, RFE Access and RFE control mapped onto an FPGA and tooling to interface to a host (PC).	RFE-M7 Infra IP drivers.Interface testing for Timing Engine, PDC, Packer and BIST DMA.
STRX system reference board (post silicon stage)	System Reference board for STRX IC, including the external on-board peripherals and the tooling to interface to a host (PC)	App-M7 to RFE-M7 communication. (API calls, Cmd server and IPC)Analog IP drivers, whole RF API functionality, Radar System cycle (Calibration, Frame and BIST)RFE-M7 Infra IP drivers.Drivers for Timing Engine, PDC, Packer and BIST DMA.
CSI2 Capturer, GUI Chirp Designer, MATLAB reference model for data visulaization	TBD	

2.2 Dependencies to other modules

N, A

2.3 Known limitations and assumptions

NA

FPGA not real-time.

Analog is simulated.

3 Test Suite

3.1 Test Suite rfe_TS_001

QUALIFICATION Test Suite TS_001 - TS_001.

Table 5. Test Sequence rfe_TS_001

Test Case	Description
rfe_tc_001_validSampleCount Configuration	The RFE Radar Use Case API shall support programming of 8-8192 samples per Chirp. Here different samples values in range are checked.

3.2 Test Suite rfe_TS_002

QUALIFICATION Test Suite TS_002 - TS_002.

Table 6. Test Sequence rfe_TS_002

Test Case	Description
rfe_tc_002_sampleCountLessThanMinRange	The RFE Radar Use Case API shall support programming of 8-8192 samples per Chirp. Here Sample count less than Min range value is checked.

3.3 Test Suite rfe_TS_003

QUALIFICATION Test Suite TS_003 - TS_003.

Table 7. Test Sequence rfe_TS_003

Test Case	Description
rfe_tc_003_sampleCountGreaterThanMaxRange	The RFE Radar Use Case API shall support programming of 8-8192 samples per Chirp. Here Sample count greater than max range value is checked.

3.4 Test Suite rfe_TS_004

QUALIFICATION Test Suite TS_004 - TS_004.

Table 8. Test Sequence rfe_TS_004

Test Case	Description
rfe_tc_004_reConfigureRfeWithDifferentSamplesCount	This test case is used to programming Samples when RFE is in 'CONFIGURED' state.

3.5 Test Suite rfe_TS_005

QUALIFICATION Test Suite TS_005 - TS_005.

Table 9. Test Sequence rfe_TS_005

Test Case	Description
rfe_tc_005_validChirpCountConfiguration	This test case is used to check the RFE API shall support programming maximum of 4096 Chirps. Here a valid chirp count range check is done.

3.6 Test Suite rfe_TS_006

QUALIFICATION Test Suite TS_006 - TS_006.

Table 10. Test Sequence rfe_TS_006

Test Case	Description
rfe_tc_006_chirpCountZero	This test case is used to check the RFE API shall support programming maximum of 4096 Chirps.Here a chirp count '0' check is done.

3.7 Test Suite rfe_TS_007

QUALIFICATION Test Suite TS_007 - TS_007.

Table 11. Test Sequence rfe_TS_007

Test Case	Description
rfe_tc_007_chirpCountGreaterThanMaxRange	This test case is used to check the RFE API shall support programming maximum of 4096 Chirps.Here a count range greater than 4096 check is done.

3.8 Test Suite rfe_TS_008

QUALIFICATION Test Suite TS_008 - TS_008.

Table 12. Test Sequence rfe_TS_008

Test Case	Description
rfe_tc_008_reConfigureRfeWithDifferentChirpCount	This test case is used to programming Chirps when RFE is in 'CONFIGURED' state.

3.9 Test Suite rfe_TS_010

QUALIFICATION Test Suite TS_010 - TS_010.

Table 13. Test Sequence rfe_TS_010

Test Case	Description
rfe_tc_010_readCurrentTempBeforeAfterChirpSequence	The RFE SW shall provide an API to get the current temperatures, the temperatures before and after the last chirp sequence.

3.10 Test Suite rfe_TS_011

QUALIFICATION Test Suite TS_011 - TS_011.

Table 14. Test Sequence rfe_TS_011

Test Case	Description
rfe_tc_011_stopRadarCycle	The RFE SW shall provide functionality to stop an running radar use case, via the RFE Radar Use Case API.

3.11 Test Suite rfe_TS_012

QUALIFICATION Test Suite TS_012 - TS_012.

Table 15. Test Sequence rfe_TS_012

Test Case	Description
rfe_tc_012	This test case is used to check the RFE restrict access to the defined API groups and to individual API calls depending on its state.

3.12 Test Suite rfe_TS_013

QUALIFICATION Test Suite TS_013 - TS_013.

Table 16. Test Sequence rfe_TS_013

Test Case	Description
rfe_tc_013	This test case is used to check the RFE SW shall support definition of multiple radar use cases.

3.13 Test Suite rfe_TS_014

QUALIFICATION Test Suite TS_014 - TS_014.

Table 17. Test Sequence rfe_TS_014

Test Case	Description
rfe_tc_014	This test case is used to check the RFE API shall support programming maximum of 4096 Chirps

3.14 Test Suite rfe_TS_015

QUALIFICATION Test Suite TS_015 - TS_015.

Table 18. Test Sequence rfe_TS_015

Test Case	Description
rfe_tc_015_configureTxBistPower	The RFE Abstract API shall provide a configuration parameter to set the Tx BIST output power.

3.15 Test Suite rfe_TS_016

QUALIFICATION Test Suite TS_016 - TS_016.

Table 19. Test Sequence rfe_TS_016

Test Case	Description
rfe_tc_016	This test case is to check the RFE SW shall provide setting of FTTI time for RFE SW, via RFE Safety API.

3.16 Test Suite rfe_TS_016_runLRRadarUc

QUALIFICATION Test Suite TS_016_runLRRadarUc - TS_016_runLRRadarUc.

Table 20. Test Sequence rfe_TS_016_runLRRadarUc

Test Case	Description
rfe_tc_016_runLRRadarUc	The RFE SW shall provide functionality to run a configured radar use case, via the RFE Radar Use Case API.

3.17 Test Suite rfe_TS_017

QUALIFICATION Test Suite TS_017 - TS_017.

Table 21. Test Sequence rfe_TS_017

Test Case	Description
rfe_tc_017	This test case is to check the RFE SW shall provide configuration of TCM recoverable errors threshold level, via RFE Safety API

3.18 Test Suite rfe_TS_017_runMRRadarUc

QUALIFICATION Test Suite TS_017_runMRRadarUc - TS_017_runMRRadarUc.

Table 22. Test Sequence rfe_TS_017_runMRRadarUc

Test Case	Description
rfe_tc_017_runMRRadarUc	The RFE SW shall provide functionality to run a configured radar use case, via the RFE Radar Use Case API.

3.19 Test Suite rfe_TS_018_runDDMARadarUc

QUALIFICATION Test Suite TS_018_runDDMARadarUc - TS_018_runDDMARadarUc.

Table 23. Test Sequence rfe_TS_018_runDDMARadarUc

Test Case	Description
rfe_tc_018_runDDMARadarUc	The RFE SW shall provide functionality to run a configured radar use case, via the RFE Radar Use Case API.

3.20 Test Suite rfe_TS_019

QUALIFICATION Test Suite TS_019 - TS_019.

Table 24. Test Sequence rfe_TS_019

Test Case	Description
rfe_tc_019_configureTxBistPowerWithFaultsMasked	The RFE Abstract API shall provide a configuration parameter to set the Tx BIST output power.

3.21 Test Suite rfe_TS_020

QUALIFICATION Test Suite TS_020 - TS_020.

Table 25. Test Sequence rfe_TS_020

Test Case	Description
rfe_tc_020_rfeAutonomousMode	The RFE SW shall support autonomous radar system cycles on RFE-M7.

3.22 Test Suite rfe_TS_021

QUALIFICATION Test Suite TS_021 - TS_021.

Table 26. Test Sequence rfe_TS_021

Test Case	Description
rfe_tc_021_checkLiPvariant	Test LiP sample shall not give BBD faults and CSI2 shall work

3.23 Test Suite rfe_TS_022

QUALIFICATION Test Suite TS_022 - TS_022.

Table 27. Test Sequence rfe_TS_022

Test Case	Description
rfe_tc_022_configureMultipleChirpSequence	The RFE SW shall support defining multiple radar chirp sequences, via RFE Abstract 2.0 API.

3.24 Test Suite rfe_TS_024

QUALIFICATION Test Suite TS_024 - TS_024.

Table 28. Test Sequence rfe_TS_024

Test Case	Description
rfe_tc_024_rxAdcClipThresholds	This test case is used to check the RFE SW shall provide programming of Rx ADC clipping thresholds, clipping counts via RFE Safety API

3.25 Test Suite rfe_TS_025

QUALIFICATION Test Suite TS_025 - TS_025.

Table 29. Test Sequence rfe_TS_025

Test Case	Description
rfe_tc_025	This test case is used to check the RFE SW shall allow testing of GPIO pins, via RFE Safety API

3.26 Test Suite rfe_TS_026

QUALIFICATION Test Suite TS_026 - TS_026.

Table 30. Test Sequence rfe_TS_026

Test Case	Description
rfe_tc_026_validRxEnable	The RFE Radar Use Case API shall support programming configuration parameters to configure 4 RX modules. Here different rx enable is checked.

3.27 Test Suite rfe_TS_027

QUALIFICATION Test Suite TS_027 - TS_027.

Table 31. Test Sequence rfe_TS_027

Test Case	Description
rfe_tc_027_checkCSI2HeaderFooter	Test switching header and footer on/off for CSI2 packet (chirp) via the RFE DFE API.

3.28 Test Suite rfe_TS_028

QUALIFICATION Test Suite TS_028 - TS_028.

Table 32. Test Sequence rfe_TS_028

Test Case	Description
rfe_tc_028_runMultipleRadarCycles	The RFE SW shall execute radar system cycles after the run command called via RFE Radar System Cycle API.

3.29 Test Suite rfe_TS_029

QUALIFICATION Test Suite TS_029 - TS_029.

Table 33. Test Sequence rfe_TS_029

Test Case	Description
rfe_tc_029_multimodeDynamicChirpConfig	The RFE SW shall dynamically program (in-between chirps) the dynamic chirp parameters into the RFE IPs.

3.30 Test Suite rfe_TS_030

QUALIFICATION Test Suite TS_030 - TS_030.

Table 34. Test Sequence rfe_TS_030

Test Case	Description
rfe_tc_030	This test case is used to check the RFE SW while in dynamic chirp configuration mode, the RFE SW shall support chirp sequence lengths of 1 to 4096 with dynamic chirp parameters.

3.31 Test Suite rfe_TS_031

QUALIFICATION Test Suite TS_031 - TS_031.

Table 35. Test Sequence rfe_TS_031

Test Case	Description
rfe_tc_031	This test case is used to check the RFE SW while in dynamic chirp configuration mode, the RFE SW shall support configuration of 1 to 4 sets of static chirp parameters (profiles) for the next system cycle, via the RFE System API

3.32 Test Suite rfe_TS_032

QUALIFICATION Test Suite TS_032 - TS_032.

Table 36. Test Sequence rfe_TS_032

Test Case	Description
rfe_tc_032	This test case is used to check the RFE SW while in dynamic chirp configuration mode, the RFE SW shall provide configuration of the dynamic chirp parameters for the complete chirp sequence of the next radar system cycle, via the RFE System API.

3.33 Test Suite rfe_TS_033

QUALIFICATION Test Suite TS_033 - TS_033.

Table 37. Test Sequence rfe_TS_033

Test Case	Description
rfe_tc_033	This test case is used to check the RFE SW shall support the following dynamic chirp parameters: Phase rotator phase shift for each Tx Fast switch for each Tx Chirp interval time ID of set of static chirp parameters (profile) to be used.

3.34 Test Suite rfe_TS_034

QUALIFICATION Test Suite TS_034 - TS_034.

Table 38. Test Sequence rfe_TS_034

Test Case	Description
rfe_tc_034_configureChirpProfile Sequence	The RFE Abstract API shall provide configuration parameter to set the chirp profile sequencing for each chirp sequence.

3.35 Test Suite rfe_TS_035

QUALIFICATION Test Suite TS_035 - TS_035.

Table 39. Test Sequence rfe_TS_035

Test Case	Description
rfe_tc_035_abstractApiBackwardCompatibility	The RFE SW shall expose RFE Abstract compatible API.

3.36 Test Suite rfe_TS_036

QUALIFICATION Test Suite TS_036 - TS_036.

Table 40. Test Sequence rfe_TS_036

Test Case	Description
rfe_tc_036_singleProfileChirpSequence	The RFE SW shall support programming of single and interleaved progressive chirp sequences, via RFE Abstract 2.0 API.

3.37 Test Suite rfe_TS_037

QUALIFICATION Test Suite TS_037 - TS_037.

Table 41. Test Sequence rfe_TS_037

Test Case	Description
rfe_tc_037_interleavedChirpSequence	The RFE SW shall support programming of single and interleaved progressive chirp sequences, via RFE Abstract 2.0 API.

3.38 Test Suite rfe_TS_040

QUALIFICATION Test Suite TS_040 - TS_040.

Table 42. Test Sequence rfe_TS_040

Test Case	Description
rfe_tc_040_configuremultipleChirpProfile	The RFE SW shall provide programming of multiple active chirp profiles with user defined parameters, via the RFE Radar Use Case API.

3.39 Test Suite rfe_TS_041

QUALIFICATION Test Suite TS_041 - TS_041.

Table 43. Test Sequence rfe_TS_041

Test Case	Description
rfe_tc_041_configuremultipleChirpSequence	The RFE SW shall provide programming of multiple active chirp profiles with user defined parameters, via the RFE Radar Use Case API.

3.40 Test Suite rfe_TS_044

QUALIFICATION Test Suite TS_044 - TS_044.

Table 44. Test Sequence rfe_TS_044

Test Case	Description
rfe_tc_044_autonomousRfe Initialization	The RFE SW shall support running initialization of RFE autonomously and independently from Host processor.

3.41 Test Suite rfe_TS_045

QUALIFICATION Test Suite TS_045 - TS_045.

Table 45. Test Sequence rfe_TS_045

Test Case	Description
rfe_tc_045	This test is used to check the RPC. This test runs on APP-M7

3.42 Test Suite rfe_TS_046

QUALIFICATION Test Suite TS_046 - TS_046.

Table 46. Test Sequence rfe_TS_046

Test Case	Description
rfe_tc_046_configureChirpTime	The RFE Abstract API shall provide configuration parameters to set the chirp timing per profile.

3.43 Test Suite rfe_TS_047

QUALIFICATION Test Suite TS_047 - TS_047.

Table 47. Test Sequence rfe_TS_047

Test Case	Description
rfe_tc_047_configureChirpTime DyTable	The RFE Abstract API shall provide configuration parameters to set the chirp timing per profile.

3.44 Test Suite rfe_TS_050

QUALIFICATION Test Suite TS_050 - TS_050.

Table 48. Test Sequence rfe_TS_050

Test Case	Description
rfe_tc_050_rfe_Abstract_API	This test case is used to check the RFE abstract API.

3.45 Test Suite rfe_TS_051

QUALIFICATION Test Suite TS_051 - TS_051.

Table 49. Test Sequence rfe_TS_051

Test Case	Description
rfe_tc_051	The RFE SW shall provide global RFE Safety configuration, via RFE Safety API

3.46 Test Suite rfe_TS_052

QUALIFICATION Test Suite TS_052 - TS_052.

Table 50. Test Sequence rfe_TS_052

Test Case	Description
rfe_tc_052	This test case verifies if RFE SW shall perform refresh of memory cells with recoverable errors, via RFE Safety API

3.47 Test Suite rfe_TS_054

QUALIFICATION Test Suite TS_054 - TS_054.

Table 51. Test Sequence rfe_TS_054

Test Case	Description
rfe_tc_054	This test case verifies if the RFE SW shall read the Rx ADC clipping counts, via RFE IP API

3.48 Test Suite rfe_TS_055

QUALIFICATION Test Suite TS_055 - TS_055.

Table 52. Test Sequence rfe_TS_055

Test Case	Description
rfe_tc_055	The test case verifies if the RFE SW shall provide programming of pass criteria for self-tests, via RFE System API.

3.49 Test Suite rfe_TS_056

QUALIFICATION Test Suite TS_056 - TS_056.

Table 53. Test Sequence rfe_TS_056

Test Case	Description
rfe_tc_056	This test case verifies if the RFE SW shall provide setting of overtemp/undertemp thresholds for each temperature sensor in RFE, via RFE IP API

3.50 Test Suite rfe_TS_057

QUALIFICATION Test Suite TS_057 - TS_057.

Table 54. Test Sequence rfe_TS_057

Test Case	Description
rfe_tc_057	This test case verifie if the RFE SW shall notify the host upon succesful error recovery

3.51 Test Suite rfe_TS_058

QUALIFICATION Test Suite TS_058 - TS_058.

Table 55. Test Sequence rfe_TS_058

Test Case	Description
rfe_tc_058	This test case verifies if the RFE SW shall notify radar application about communication error via FCCU

3.52 Test Suite rfe_TS_059

QUALIFICATION Test Suite TS_059 - TS_059.

Table 56. Test Sequence rfe_TS_059

Test Case	Description
rfe_tc_059	This test case verifies successful initialization/calibration, RFE SW shall enter normal operation and report this to application

3.53 Test Suite rfe_TS_062

QUALIFICATION Test Suite TS_062 - TS_062.

Table 57. Test Sequence rfe_TS_062

Test Case	Description
rfe_tc_062	This test is used to program interleaved or non-interleaved packing mode per output stream via the RFE Abstract API.

3.54 Test Suite rfe_TS_064

QUALIFICATION Test Suite TS_064 - TS_064.

Table 58. Test Sequence rfe_TS_064

Test Case	Description
rfe_tc_064	This test is used to support defining multiple chirp sequences, via the RFE Radar Frame API

3.55 Test Suite rfe_TS_065

QUALIFICATION Test Suite TS_065 - TS_065.

Table 59. Test Sequence rfe_TS_065

Test Case	Description
rfe_tc_065	This test is used to support configuration of maximum of 20MHz (analog) bandwidth per Chirp.

3.56 Test Suite rfe_TS_066

QUALIFICATION Test Suite TS_066 - TS_066.

Table 60. Test Sequence rfe_TS_066

Test Case	Description
rfe_tc_066	This test is used to support defining a radar frame, a sequence of chirps comprising a number of modes (chirps of a given profile), with modes interleaved, via the RFE Radar Frame API.

3.57 Test Suite rfe_TS_067

QUALIFICATION Test Suite TS_067 - TS_067.

Table 61. Test Sequence rfe_TS_067

Test Case	Description
rfe_tc_067_outputStream	This test is used to provide programming interface for 4 independent output streams via RFE DFE API.

3.58 Test Suite rfe_TS_068

QUALIFICATION Test Suite TS_068 - TS_068.

Table 62. Test Sequence rfe_TS_068

Test Case	Description
rfe_tc_068	This test is used to provide destination programming per output to be CSI-2 or/and PPE via RFE Abstract API.

3.59 Test Suite rfe_TS_069

QUALIFICATION Test Suite TS_069 - TS_069.

Table 63. Test Sequence rfe_TS_069

Test Case	Description
rfe_tc_069	This test is used to execute System Cycle within the programmed time.

3.60 Test Suite rfe_TS_074

QUALIFICATION Test Suite TS_074 - TS_074.

Table 64. Test Sequence rfe_TS_074

Test Case	Description
rfe_tc_074	This test is used to verify that the system can finalize start-up initializations and calibrations within 1[ms].

3.61 Test Suite rfe_TS_075

QUALIFICATION Test Suite TS_075 - TS_075.

Table 65. Test Sequence rfe_TS_075

Test Case	Description
rfe_tc_075_rdrCycStartDelay	This test is used to test start time delay of radar system cycle

3.62 Test Suite rfe_TS_077

QUALIFICATION Test Suite TS_077 - TS_077.

Table 66. Test Sequence rfe_TS_077

Test Case	Description
rfe_tc_077	This test is used to check if it finalize RFE Self-Tests within 2[ms].

3.63 Test Suite rfe_TS_082

QUALIFICATION Test Suite TS_082 - TS_082.

Table 67. Test Sequence rfe_TS_082

Test Case	Description
rfe_tc_082	This test is used to provide functionality to apply radar use case (i.e. configure RFE).

3.64 Test Suite rfe_TS_084

QUALIFICATION Test Suite TS_084 - TS_084.

Table 68. Test Sequence rfe_TS_084

Test Case	Description
rfe_tc_084	Check data from CSI2 TX DMA is streamed out via CSI2 at the specified data rate.

3.65 Test Suite rfe_TS_085

QUALIFICATION Test Suite TS_085 - TS_085.

Table 69. Test Sequence rfe_TS_085

Test Case	Description
rfe_tc_085	Check if user can select the leader-follower synchronization RFE IO via the RFE Abstract API.

3.66 Test Suite rfe_TS_086

QUALIFICATION Test Suite TS_086 - TS_086.

Table 70. Test Sequence rfe_TS_086

Test Case	Description
rfe_tc_086	Test the RFE Abstract API shall provide a configuration parameter for selecting the radar cycle trigger GPIO on the leader device.

3.67 Test Suite rfe_TS_087

QUALIFICATION Test Suite TS_087 - TS_087.

Table 71. Test Sequence rfe_TS_087

Test Case	Description
rfe_tc_087	User can set the LOIF output power of the leader via the RFE Abstract API.

3.68 Test Suite rfe_TS_088

QUALIFICATION Test Suite TS_088 - TS_088.

Table 72. Test Sequence rfe_TS_088

Test Case	Description
rfe_tc_088	Check if the Abstract API support maximum of 40 MHz analog bandwidth per Chirp

3.69 Test Suite rfe_TS_089

QUALIFICATION Test Suite TS_089 - TS_089.

Table 73. Test Sequence rfe_TS_089

Test Case	Description
rfe_tc_089	Test RFE SW shall configure follower devices to use LO signal of leader.

3.70 Test Suite rfe_TS_090

QUALIFICATION Test Suite TS_090 - TS_090.

Table 74. Test Sequence rfe_TS_090

Test Case	Description
rfe_tc_090	Test RFE SW shall synchronize chirp transmission and reception on leader and follower devices to the same clock edge of the shared CLK.

3.71 Test Suite rfe_TS_091

QUALIFICATION Test Suite TS_091 - TS_091.

Table 75. Test Sequence rfe_TS_091

Test Case	Description
rfe_tc_091	Test RFE SW shall configure leader device to derive CLK from XO and calibrate its phase to that of CLK_IN.

3.72 Test Suite rfe_TS_092

QUALIFICATION Test Suite TS_092 - TS_092.

Table 76. Test Sequence rfe_TS_092

Test Case	Description
rfe_tc_092	The phase noise of the LO signal is reduced and the ADC sampling moment of leader and followers are aligned by Leader using XO clock.

3.73 Test Suite rfe_TS_093

QUALIFICATION Test Suite TS_093 - TS_093.

Table 77. Test Sequence rfe_TS_093

Test Case	Description
rfe_tc_093	A cascading example application running on the APP-M7 core of the cascaded devices.

3.74 Test Suite rfe_TS_094

QUALIFICATION Test Suite TS_094 - TS_094.

Table 78. Test Sequence rfe_TS_094

Test Case	Description
rfe_tc_094	Check RFE FW can operate ADC streaming via WDMA

3.75 Test Suite rfe_TS_095

QUALIFICATION Test Suite TS_095 - TS_095.

Table 79. Test Sequence rfe_TS_095

Test Case	Description
rfe_tc_095	Check if RFE SW support radar cycle trigger signal from GMAC within shared clock cycle accurate delay on the leader or standalone case.

3.76 Test Suite rfe_TS_096

QUALIFICATION Test Suite TS_096 - TS_096.

Table 80. Test Sequence rfe_TS_096

Test Case	Description
rfe_tc_096	Check if the Abstract API support PDC filter for 80Mhz sampling rate

3.77 Test Suite rfe_TS_101

QUALIFICATION Test Suite TS_101 - TS_101.

Table 81. Test Sequence rfe_TS_101

Test Case	Description
rfe_tc_101_validTxOutputPower	The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here different tx power in range is checked.

3.78 Test Suite rfe_TS_102

QUALIFICATION Test Suite TS_102 - TS_102.

Table 82. Test Sequence rfe_TS_102

Test Case	Description
rfe_tc_102_lessthanMinTxOutputPower	The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here tx power less than Min range value is checked.

3.79 Test Suite rfe_TS_103

QUALIFICATION Test Suite TS_103 - TS_103.

Table 83. Test Sequence rfe_TS_103

Test Case	Description
rfe_tc_103_greaterthanMaxTxOutputPower	The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here tx power greater than maximum range value is checked.

3.80 Test Suite rfe_TS_104

QUALIFICATION Test Suite TS_104 - TS_104.

Table 84. Test Sequence rfe_TS_104

Test Case	Description
rfe_tc_104_multiProfileTxOutput Power	The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here tx power for multi profile is checked.

3.81 Test Suite rfe_TS_105

QUALIFICATION Test Suite TS_105 - TS_105.

Table 85. Test Sequence rfe_TS_105

Test Case	Description
rfe_tc_105_validTxEnable	The RFE Radar Use Case API shall support programming configuration parameters to configure 4 TX modules. Here different tx enable is checked.

3.82 Test Suite rfe_TS_106

QUALIFICATION Test Suite TS_106 - TS_106.

Table 86. Test Sequence rfe_TS_106

Test Case	Description
rfe_tc_106_validTxPhaseRotation	The RFE Radar Use Case API shall support programming of 0 to 357.1875 degrees for phase rotation (in steps of 2.8125 degrees). Here different phase rotation in range is checked.

3.83 Test Suite rfe_TS_107

QUALIFICATION Test Suite TS_107 - TS_107.

Table 87. Test Sequence rfe_TS_107

Test Case	Description
rfe_tc_107_invalidTxPhase Rotation	The RFE Radar Use Case API shall support programming of 0 to 357.1875 degrees for phase rotation (in steps of 2.8125 degrees). Here invalid phase rotation is checked.

3.84 Test Suite rfe_TS_108

QUALIFICATION Test Suite TS_108 - TS_108.

Table 88. Test Sequence rfe_TS_108

Test Case	Description
rfe_tc_108_multiProfileTxPhase Rotation	The RFE Radar Use Case API shall support programming of 0 to 357.1875 degrees for phase rotation (in steps of 2.8125 degrees). Here invalid phase rotation is checked.

3.85 Test Suite rfe_TS_110

QUALIFICATION Test Suite TS_110 - TS_110.

Table 89. Test Sequence rfe_TS_110

Test Case	Description
rfe_tc_110_dynProf	The RFE Radar Use Case API shall program the id of the profile to be used for the next-next chirp into the TE while dynamically programming.

3.86 Test Suite rfe_TS_111

QUALIFICATION Test Suite TS_111 - TS_111.

Table 90. Test Sequence rfe_TS_111

Test Case	Description
rfe_tc_111_rxGainSingleProfile	The RFE Radar Use Case API shall support programming of 25dB to 46dB rx gain. Here different rx gain in range and out of range is checked.

3.87 Test Suite rfe_TS_112

QUALIFICATION Test Suite TS_112 - TS_112.

Table 91. Test Sequence rfe_TS_112

Test Case	Description
rfe_tc_112_rxGainMultiProfile	The RFE Radar Use Case API shall support programming of 25dB to 46dB rx gain. Here different rx gain in range and out of range is checked.

3.88 Test Suite rfe_TS_113

QUALIFICATION Test Suite TS_113 - TS_113.

Table 92. Test Sequence rfe_TS_113

Test Case	Description
rfe_tc_113_hpfRxSingleProfile	The RFE Radar Use Case API shall support programming of 200kHz to 6400kHz hpf cutoff frequency for rx. Here different rx hpf cutoff frequencies in range and out of range is checked.

3.89 Test Suite rfe_TS_114

QUALIFICATION Test Suite TS_114 - TS_114.

Table 93. Test Sequence rfe_TS_114

Test Case	Description
rfe_tc_114_hpfRxMultiProfile	The RFE Radar Use Case API shall support programming of 200kHz to 6400kHz hpf cutoff frequency for rx. Here different rx hpf cutoff frequencies in range and out of range is checked.

3.90 Test Suite rfe_TS_115

QUALIFICATION Test Suite TS_115 - TS_115.

Table 94. Test Sequence rfe_TS_115

Test Case	Description
rfe_tc_115_lpfRxSingleProfile	The RFE Radar Use Case API shall support programming of 10MHz to 40MHz lpf cutoff frequency for rx. Here different rx lpf cutoff frequencies in range and out of range is checked.

3.91 Test Suite rfe_TS_116

QUALIFICATION Test Suite TS_116 - TS_116.

Table 95. Test Sequence rfe_TS_116

Test Case	Description
rfe_tc_116_lpfRxMultiProfile	The RFE Radar Use Case API shall support programming of 10MHz to 40MHz lpf cutoff frequency for rx. Here different rx lpf cutoff frequencies in range and out of range is checked.

3.92 Test Suite rfe_TS_117

QUALIFICATION Test Suite TS_117 - TS_117.

Table 96. Test Sequence rfe_TS_117

Test Case	Description
rfe_tc_117_txOutputPowerLR	The RFE Radar Use Case API shall support programming of tx power for long range usecase.

3.93 Test Suite rfe_TS_118

QUALIFICATION Test Suite TS_118 - TS_118.

Table 97. Test Sequence rfe_TS_118

Test Case	Description
rfe_tc_118_txOutputPowerSR	The RFE Radar Use Case API shall support programming of tx power for short range usecase.

3.94 Test Suite rfe_TS_119

QUALIFICATION Test Suite TS_119 - TS_119.

Table 98. Test Sequence rfe_TS_119

Test Case	Description
rfe_tc_119_txOutputPowerSaving	The RFE Radar Use Case API shall support programming of tx power for power saving usecase.

3.95 Test Suite rfe_TS_120

QUALIFICATION Test Suite TS_120 - TS_120.

Table 99. Test Sequence rfe_TS_120

Test Case	Description
rfe_tc_120_validRxSatDetThresholdLevel	The RFE Radar Use Case API shall support programming of rx saturation detector threshold level. Here different rx saturation detector threshold level in range is checked.

3.96 Test Suite rfe_TS_121

QUALIFICATION Test Suite TS_121 - TS_121.

Table 100. Test Sequence rfe_TS_121

Test Case	Description
rfe_tc_121_invalidRxSatDetThresholdLevel	The RFE Radar Use Case API shall support programming of rx saturation detector threshold level. Here rx saturation detector threshold level out of range is checked.

3.97 Test Suite rfe_TS_122

QUALIFICATION Test Suite TS_122 - TS_122.

Table 101. Test Sequence rfe_TS_122

Test Case	Description
rfe_tc_122_validChirpFreqSweep	The RFE Radar Use Case API shall support programming to set chirp frequency sweep. Here different center frequency in range is checked. Frequency drift per chirp in 38.14697265625 Hz

3.98 Test Suite rfe_TS_123

QUALIFICATION Test Suite TS_123 - TS_123.

Table 102. Test Sequence rfe_TS_123

Test Case	Description
rfe_tc_123_clkio	The RFE Radar Use Case API shall provide a configuration parameters to configure CLK_IN and CLK_OUT pins.

3.99 Test Suite rfe_TS_124

QUALIFICATION Test Suite TS_124 - TS_124.

Table 103. Test Sequence rfe_TS_124

Test Case	Description
rfe_tc_124_clkioRouted	The RFE Radar Use Case API shall provide a configuration parameters to configure CLK_IN and CLK_OUT pins.

3.100 Test Suite rfe_TS_125

QUALIFICATION Test Suite TS_125 - TS_125.

Table 104. Test Sequence rfe_TS_125

Test Case	Description
rfe_tc_125_chirppllTestPin	The RFE SW shall provide an API to output the ChirpPll signal to a PIN.

3.101 Test Suite rfe_TS_126

QUALIFICATION Test Suite TS_126 - TS_126.

Table 105. Test Sequence rfe_TS_126

Test Case	Description
rfe_tc_126_Calibrate	The RFE Radar Use Case API shall calibrate analog IPs based on customer configuration.

3.102 Test Suite rfe_TS_127

QUALIFICATION Test Suite TS_127 - TS_127.

Table 106. Test Sequence rfe_TS_127

Test Case	Description
rfe_tc_127_recalibrate	The RFE Radar Use Case API shall re-calibrate analog IPs periodically to compensate for temperature over time.

3.103 Test Suite rfe_TS_128

QUALIFICATION Test Suite TS_128 - TS_128.

Table 107. Test Sequence rfe_TS_128

Test Case	Description
rfe_tc_128_rxAdcCalAtbTrim	The RFE SW shall trim the ATB-ADC and calibrate the RX-ADC independently from customer config.

3.104 Test Suite rfe_TS_129

QUALIFICATION Test Suite TS_129 - TS_129.

Table 108. Test Sequence rfe_TS_129

Test Case	Description
rfe_tc_129_8profiles	The RFE SW shall support control of 8 profiles for each IP and bist functionality.

3.105 Test Suite rfe_TS_3001

QUALIFICATION Test Suite TS_3001 - TS_3001.

Table 109. Test Sequence rfe_TS_3001

Test Case	Description
rfe_tc_3001_safetyApiConfig	The RFE SW shall provide RFE Safety configuration, via RFE Safety API

3.106 Test Suite rfe_TS_3002

QUALIFICATION Test Suite TS_3002 - TS_3002.

Table 110. Test Sequence rfe_TS_3002

Test Case	Description
rfe_tc_3002_heartBeatPulse	The RFE SW shall send HB(Heart Beat) once per radar cycle.

3.107 Test Suite rfe_TS_3003

QUALIFICATION Test Suite TS_3003 - TS_3003.

Table 111. Test Sequence rfe_TS_3003

Test Case	Description
rfe_tc_3003_toggleErrorN	This test case is to check the toggling of the ERROR_N pin, via RFE Safety API

3.108 Test Suite rfe_TS_3004

QUALIFICATION Test Suite TS_3004 - TS_3004.

Table 112. Test Sequence rfe_TS_3004

Test Case	Description
rfe_tc_3004_invalidParameter	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.109 Test Suite rfe_TS_3005

QUALIFICATION Test Suite TS_3005 - TS_3005.

Table 113. Test Sequence rfe_TS_3005

Test Case	Description
rfe_tc_3005_frequencyForBist	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.110 Test Suite rfe_TS_3006

QUALIFICATION Test Suite TS_3006 - TS_3006.

Table 114. Test Sequence rfe_TS_3006

Test Case	Description
rfe_tc_3006_rxPhaseDiffThresholdTolerance	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.111 Test Suite rfe_TS_3007

QUALIFICATION Test Suite TS_3007 - TS_3007.

Table 115. Test Sequence rfe_TS_3007

Test Case	Description
rfe_tc_3007_rxGainDiffThresholdTolerance	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.112 Test Suite rfe_TS_3008

QUALIFICATION Test Suite TS_3008 - TS_3008.

Table 116. Test Sequence rfe_TS_3008

Test Case	Description
rfe_tc_3008_injectTestToneBefore Lna	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.113 Test Suite rfe_TS_3009

QUALIFICATION Test Suite TS_3009 - TS_3009.

Table 117. Test Sequence rfe_TS_3009

Test Case	Description
rfe_tc_3009_zeroHourReference ForRxPhaseDiff	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.114 Test Suite rfe_TS_3010

QUALIFICATION Test Suite TS_3010 - TS_3010.

Table 118. Test Sequence rfe_TS_3010

Test Case	Description
rfe_tc_3010_zeroHourReference ForRxGainDiff	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.115 Test Suite rfe_TS_3011

QUALIFICATION Test Suite TS_3011 - TS_3011.

Table 119. Test Sequence rfe_TS_3011

Test Case	Description
rfe_tc_3011_txPhaseDiff ThresholdTolerance	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.116 Test Suite rfe_TS_3012

QUALIFICATION Test Suite TS_3012 - TS_3012.

Table 120. Test Sequence rfe_TS_3012

Test Case	Description
rfe_tc_3012_txPhaseStep ThresholdTolerance	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.117 Test Suite rfe_TS_3013

QUALIFICATION Test Suite TS_3013 - TS_3013.

Table 121. Test Sequence rfe_TS_3013

Test Case	Description
rfe_tc_3013_zeroHourReference ForTxPhaseDiff	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.118 Test Suite rfe_TS_3014

QUALIFICATION Test Suite TS_3014 - TS_3014.

Table 122. Test Sequence rfe_TS_3014

Test Case	Description
rfe_tc_3014_txPowerLevelForBist	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.119 Test Suite rfe_TS_3015

QUALIFICATION Test Suite TS_3015 - TS_3015.

Table 123. Test Sequence rfe_TS_3015

Test Case	Description
rfe_tc_3015_txSelectForTxBist	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.120 Test Suite rfe_TS_3016

QUALIFICATION Test Suite TS_3016 - TS_3016.

Table 124. Test Sequence rfe_TS_3016

Test Case	Description
rfe_tc_3016_getZeroHourData_loop	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.121 Test Suite rfe_TS_3017

QUALIFICATION Test Suite TS_3017 - TS_3017.

Table 125. Test Sequence rfe_TS_3017

Test Case	Description
rfe_tc_3017_getZeroHourData_differentState	The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

3.122 Test Suite rfe_TS_3018

QUALIFICATION Test Suite TS_3018 - TS_3018.

Table 126. Test Sequence rfe_TS_3018

Test Case	Description
rfe_tc_3018_sm3Tx1BallBreak	Testcase to perform ball break detection tx1 r1 fault handling and r2 fault promotion for sm3

3.123 Test Suite rfe_TS_3020

QUALIFICATION Test Suite TS_3020 - TS_3020.

Table 127. Test Sequence rfe_TS_3020

Test Case	Description
rfe_tc_3020_sm5Tx3BallBreak	Testcase to perform ball break detection tx3 r1 fault handling and r2 fault promotion for sm5

3.124 Test Suite rfe_TS_3021

QUALIFICATION Test Suite TS_3021 - TS_3021.

Table 128. Test Sequence rfe_TS_3021

Test Case	Description
rfe_tc_3021_sm6Tx4BallBreak	Testcase to perform ball break detection tx4 r1 fault handling and r2 fault promotion for sm6

3.125 Test Suite rfe_TS_3022

QUALIFICATION Test Suite TS_3022 - TS_3022.

Table 129. Test Sequence rfe_TS_3022

Test Case	Description
rfe_tc_3022_sm7Tx1PPDLevelDetection	Testcase to perform tx1 ppd level det r1 fault handling and r1 to r2 fault promotion for sm7

3.126 Test Suite rfe_TS_3023

QUALIFICATION Test Suite TS_3023 - TS_3023.

Table 130. Test Sequence rfe_TS_3023

Test Case	Description
rfe_tc_3023_sm8Tx2PPDLevelDetection	Testcase to perform tx2 ppd level det r1 fault handling and r1 to r2 fault promotion for sm8

3.127 Test Suite rfe_TS_3024

QUALIFICATION Test Suite TS_3024 - TS_3024.

Table 131. Test Sequence rfe_TS_3024

Test Case	Description
rfe_tc_3024_sm9Tx3PPDLevelDetection	Testcase to perform tx3 ppd level det r1 fault handling and r1 to r2 fault promotion for sm9

3.128 Test Suite rfe_TS_3025

QUALIFICATION Test Suite TS_3025 - TS_3025.

Table 132. Test Sequence rfe_TS_3025

Test Case	Description
rfe_tc_3025_sm10Tx4PPDLevelDetection	Testcase to perform tx4 ppd level det r1 fault handling and r1 to r2 fault promotion for sm10

3.129 Test Suite rfe_TS_3026

QUALIFICATION Test Suite TS_3026 - TS_3026.

Table 133. Test Sequence rfe_TS_3026

Test Case	Description
rfe_tc_3026_sm11TxPhaseDifferenceTx1Tx2R1FaultHandling	Testcase to perform tx bist phase difference tx1tx2 r1 fault handling for sm11

3.130 Test Suite rfe_TS_3027

QUALIFICATION Test Suite TS_3027 - TS_3027.

Table 134. Test Sequence rfe_TS_3027

Test Case	Description
rfe_tc_3027_sm11TxPhaseDifferenceTx1Tx2R2FaultPromotion	Testcase to perform tx bist phase difference tx1tx2 r1 to r2 fault promotion for sm11

3.131 Test Suite rfe_TS_3028

QUALIFICATION Test Suite TS_3028 - TS_3028.

Table 135. Test Sequence rfe_TS_3028

Test Case	Description
rfe_tc_3028_sm11TxPhaseStepTx1R1FaultHandling	Testcase to perform tx bist phase step tx1 r1 fault handling for sm11

3.132 Test Suite rfe_TS_3029

QUALIFICATION Test Suite TS_3029 - TS_3029.

Table 136. Test Sequence rfe_TS_3029

Test Case	Description
rfe_tc_3029_sm11TxPhaseStepTx1R2FaultPromotion	Testcase to perform tx bist phase step tx1 r1 to r2 fault promotion for sm11

3.133 Test Suite rfe_TS_3032

QUALIFICATION Test Suite TS_3032 - TS_3032.

Table 137. Test Sequence rfe_TS_3032

Test Case	Description
rfe_tc_3032_sm12TxPhaseStepTx2R1FaultHandling	Testcase to perform tx bist phase step tx2 r1 fault handling for sm12

3.134 Test Suite rfe_TS_3033

QUALIFICATION Test Suite TS_3033 - TS_3033.

Table 138. Test Sequence rfe_TS_3033

Test Case	Description
rfe_tc_3033_sm12TxPhaseStepTx2R2FaultPromotion	Testcase to perform tx bist phase step tx2 r1 to r2 fault promotion for sm12

3.135 Test Suite rfe_TS_3034

QUALIFICATION Test Suite TS_3034 - TS_3034.

Table 139. Test Sequence rfe_TS_3034

Test Case	Description
rfe_tc_3034_sm13TxPhaseDifferenceTx3Tx4R1FaultHandling	Testcase to perform tx bist phase difference tx3tx4 r1 fault handling for sm13

3.136 Test Suite rfe_TS_3035

QUALIFICATION Test Suite TS_3035 - TS_3035.

Table 140. Test Sequence rfe_TS_3035

Test Case	Description
rfe_tc_3035_sm13TxPhaseDifferenceTx3Tx4R2FaultPromotion	Testcase to perform tx bist phase difference tx3tx4 r1 to r2 fault promotion for sm13

3.137 Test Suite rfe_TS_3038

QUALIFICATION Test Suite TS_3038 - TS_3038.

Table 141. Test Sequence rfe_TS_3038

Test Case	Description
rfe_tc_3038_sm13TxPhaseStepTx4R1FaultHandling	Testcase to perform tx bist phase step tx4 r1 fault handling for sm13

3.138 Test Suite rfe_TS_3039

QUALIFICATION Test Suite TS_3039 - TS_3039.

Table 142. Test Sequence rfe_TS_3039

Test Case	Description
rfe_tc_3039_sm13TxPhaseStepTx4R2FaultPromotion	Testcase to perform tx bist phase step tx4 r1 to r2 fault promotion for sm13

3.139 Test Suite rfe_TS_3040

QUALIFICATION Test Suite TS_3040 - TS_3040.

Table 143. Test Sequence rfe_TS_3040

Test Case	Description
rfe_tc_3040_sm15Rx1BallBreak	Testcase to perform ball break detection rx1 r1 fault handling and r2 fault promotion for sm15

3.140 Test Suite rfe_TS_3041

QUALIFICATION Test Suite TS_3041 - TS_3041.

Table 144. Test Sequence rfe_TS_3041

Test Case	Description
rfe_tc_3041_sm16Rx2BallBreak	Testcase to perform ball break detection rx2 r1 fault handling and r2 fault promotion for sm16

3.141 Test Suite rfe_TS_3042

QUALIFICATION Test Suite TS_3042 - TS_3042.

Table 145. Test Sequence rfe_TS_3042

Test Case	Description
rfe_tc_3042_sm17Rx3BallBreak	Testcase to perform ball break detection rx3 r1 fault handling and r2 fault promotion for sm17

3.142 Test Suite rfe_TS_3043

QUALIFICATION Test Suite TS_3043 - TS_3043.

Table 146. Test Sequence rfe_TS_3043

Test Case	Description
rfe_tc_3043_sm18Rx4BallBreak	Testcase to perform ball break detection rx4 r1 fault handling and r2 fault promotion for sm18

3.143 Test Suite rfe_TS_3046

QUALIFICATION Test Suite TS_3046 - TS_3046.

Table 147. Test Sequence rfe_TS_3046

Test Case	Description
rfe_tc_3046_sm19Rx1LOLevelDetectorR1FaultHandling	Testcase to perform lo lev detector for rx1 r1 fault handling for sm19

3.144 Test Suite rfe_TS_3047

QUALIFICATION Test Suite TS_3047 - TS_3047.

Table 148. Test Sequence rfe_TS_3047

Test Case	Description
rfe_tc_3047_sm19Rx1LOLevelDetectorR2FaultPromotion	Testcase to perform lo lev detector for rx1 r1 to r2 fault promotion for sm19

3.145 Test Suite rfe_TS_3048

QUALIFICATION Test Suite TS_3048 - TS_3048.

Table 149. Test Sequence rfe_TS_3048

Test Case	Description
rfe_tc_3048_sm20Rx2LOLevelDetectorR1FaultHandling	Testcase to perform lo lev detector for rx2 r1 fault handling for sm20

3.146 Test Suite rfe_TS_3049

QUALIFICATION Test Suite TS_3049 - TS_3049.

Table 150. Test Sequence rfe_TS_3049

Test Case	Description
rfe_tc_3049_sm20Rx2LOLevelDetectorR2FaultPromotion	Testcase to perform lo lev detector for rx2 r1 to r2 fault promotion for sm20

3.147 Test Suite rfe_TS_3050

QUALIFICATION Test Suite TS_3050 - TS_3050.

Table 151. Test Sequence rfe_TS_3050

Test Case	Description
rfe_tc_3050_sm21Rx3LOLevelDetectorR1FaultHandling	Testcase to perform lo lev detector for rx3 r1 fault handling for sm21

3.148 Test Suite rfe_TS_3051

QUALIFICATION Test Suite TS_3051 - TS_3051.

Table 152. Test Sequence rfe_TS_3051

Test Case	Description
rfe_tc_3051_sm21Rx3LOLevelDetectorR2FaultPromotion	Testcase to perform lo lev detector for rx3 r1 to r2 fault promotion for sm21

3.149 Test Suite rfe_TS_3052

QUALIFICATION Test Suite TS_3052 - TS_3052.

Table 153. Test Sequence rfe_TS_3052

Test Case	Description
rfe_tc_3052_sm22Rx4LOLevelDetectorR1FaultHandling	Testcase to perform lo lev detector for rx4 r1 fault handling for sm22

3.150 Test Suite rfe_TS_3053

QUALIFICATION Test Suite TS_3053 - TS_3053.

Table 154. Test Sequence rfe_TS_3053

Test Case	Description
rfe_tc_3053_sm22Rx4LOLevelDetectorR2FaultPromotion	Testcase to perform lo lev detector for rx4 r1 to r2 fault promotion for sm22

3.151 Test Suite rfe_TS_3054

QUALIFICATION Test Suite TS_3054 - TS_3054.

Table 155. Test Sequence rfe_TS_3054

Test Case	Description
rfe_tc_3054_sm23RxBistR1FaultHandling	Testcase to perform rx bist r1 fault handling for sm23

3.152 Test Suite rfe_TS_3055

QUALIFICATION Test Suite TS_3055 - TS_3055.

Table 156. Test Sequence rfe_TS_3055

Test Case	Description
rfe_tc_3055_sm23RxBistR2FaultPromotion	Testcase to perform rx bist r1 to r2 fault promotion for sm23

3.153 Test Suite rfe_TS_3056

QUALIFICATION Test Suite TS_3056 - TS_3056.

Table 157. Test Sequence rfe_TS_3056

Test Case	Description
rfe_tc_3056_sm28Gldo1v3OVR1FaultHandling	Testcase to perform gldo 1v3 ov only r1 fault handling for sm28

3.154 Test Suite rfe_TS_3065

QUALIFICATION Test Suite TS_3065 - TS_3065.

Table 158. Test Sequence rfe_TS_3065

Test Case	Description
rfe_tc_3065_sm50AdpllDCOLevelHighDetectR1FaultHandling	Testcase to perform adpll dco lev high detector r1 fault handling for sm50

3.155 Test Suite rfe_TS_3066

QUALIFICATION Test Suite TS_3066 - TS_3066.

Table 159. Test Sequence rfe_TS_3066

Test Case	Description
rfe_tc_3066_sm50AdpllDCOLevelHighDetectR2FaultPromotion	Testcase to perform adpll dco lev high detector r1 to r2 fault promotion for sm50

3.156 Test Suite rfe_TS_3068

QUALIFICATION Test Suite TS_3068 - TS_3068.

Table 160. Test Sequence rfe_TS_3068

Test Case	Description
rfe_tc_3068_sm50AdplIDCOLevelLowDetectR2FaultPromotion	Testcase to perform adpll dco lev low detector r1 to r2 fault promotion for sm50

3.157 Test Suite rfe_TS_3069

QUALIFICATION Test Suite TS_3069 - TS_3069.

Table 161. Test Sequence rfe_TS_3069

Test Case	Description
rfe_tc_3069_sm54ChirpPllRMSVcoLevelHighR1FaultHandling	Testcase to perform chirppll rms dco detector and amplitude monitor high r1 fault handling for sm54

3.158 Test Suite rfe_TS_3070

QUALIFICATION Test Suite TS_3070 - TS_3070.

Table 162. Test Sequence rfe_TS_3070

Test Case	Description
rfe_tc_3070_sm54ChirpPllRMSVcoLevelHighR2FaultPromotion	Testcase to perform chirppll rms dco detector and amplitude monitor high r1 to r2 fault promotion for sm54

3.159 Test Suite rfe_TS_3071

QUALIFICATION Test Suite TS_3071 - TS_3071.

Table 163. Test Sequence rfe_TS_3071

Test Case	Description
rfe_tc_3071_sm54ChirpPllRMSVcoLevelLowR1FaultHandling	Testcase to perform chirppll rms dco detector and amplitude monitor low r1 fault handling for sm54

3.160 Test Suite rfe_TS_3072

QUALIFICATION Test Suite TS_3072 - TS_3072.

Table 164. Test Sequence rfe_TS_3072

Test Case	Description
rfe_tc_3072_sm54ChirpPllRMSVcoLevelLowR2FaultPromotion	Testcase to perform chirppll rms dco detector and amplitude monitor low r1 to r2 fault promotion for sm54

3.161 Test Suite rfe_TS_3073

QUALIFICATION Test Suite TS_3073 - TS_3073.

Table 165. Test Sequence rfe_TS_3073

Test Case	Description
rfe_tc_3073_sm55ChirpPllUnlockDetectorR1FaultHandling	Testcase to perform chirppll unlock detector r1 fault handling for sm55

3.162 Test Suite rfe_TS_3074

QUALIFICATION Test Suite TS_3074 - TS_3074.

Table 166. Test Sequence rfe_TS_3074

Test Case	Description
rfe_tc_3074_sm55ChirpPllUnlockDetectorR2FaultPromotion	Testcase to perform chirppll unlock detector r1 to r2 fault promotion for sm55

3.163 Test Suite rfe_TS_3075

QUALIFICATION Test Suite TS_3075 - TS_3075.

Table 167. Test Sequence rfe_TS_3075

Test Case	Description
rfe_tc_3075_sm57ChirpPllCentreFrequencyDriftedR1FaultHandling	Testcase to perform chirppll frequency monitor r1 fault handling for sm57

3.164 Test Suite rfe_TS_3076

QUALIFICATION Test Suite TS_3076 - TS_3076.

Table 168. Test Sequence rfe_TS_3076

Test Case	Description
rfe_tc_3076_sm57ChirpPllCentreFrequencyDriftedR2FaultPromotion	Testcase to perform chirppll frequency monitor r1 to r2 fault promotion for sm57

3.165 Test Suite rfe_TS_3079

QUALIFICATION Test Suite TS_3079 - TS_3079.

Table 169. Test Sequence rfe_TS_3079

Test Case	Description
rfe_tc_3079_sm59ChirpPllLockStepLinearInterR1FaultHandling	Testcase to perform chirppll lockstep linearinter r1 fault handling for sm59

3.166 Test Suite rfe_TS_3080

QUALIFICATION Test Suite TS_3080 - TS_3080.

Table 170. Test Sequence rfe_TS_3080

Test Case	Description
rfe_tc_3080_sm59ChirpPllLock StepLinearInterR2FaultPromotion	Testcase to perform chirppll lockstep linearinter r1 to r2 fault promotion for sm59

3.167 Test Suite rfe_TS_3081

QUALIFICATION Test Suite TS_3081 - TS_3081.

Table 171. Test Sequence rfe_TS_3081

Test Case	Description
rfe_tc_3081_sm60ChirpPllLock StepSigmaDeltaR1FaultHandling	Testcase to perform chirppll lockstep sigmadelta r1 fault handling for sm60

3.168 Test Suite rfe_TS_3082

QUALIFICATION Test Suite TS_3082 - TS_3082.

Table 172. Test Sequence rfe_TS_3082

Test Case	Description
rfe_tc_3082_sm60ChirpPll LockStepSigmaDeltaR2Fault Promotion	Testcase to perform chirppll lockstep sigmadelta r1 to r2 fault promotion for sm60

3.169 Test Suite rfe_TS_3083

QUALIFICATION Test Suite TS_3083 - TS_3083.

Table 173. Test Sequence rfe_TS_3083

Test Case	Description
rfe_tc_3083_sm62TeLockStepR1 FaultHandling	Testcase to perform te lockstep r1 fault handling for sm62

3.170 Test Suite rfe_TS_3084

QUALIFICATION Test Suite TS_3084 - TS_3084.

Table 174. Test Sequence rfe_TS_3084

Test Case	Description
rfe_tc_3084_sm62TeLockStepR2 FaultPromotion	Testcase to perform te lockstep r1 to r2 fault promotion for sm62

3.171 Test Suite rfe_TS_3085

QUALIFICATION Test Suite TS_3085 - TS_3085.

Table 175. Test Sequence rfe_TS_3085

Test Case	Description
rfe_tc_3085_sm63TeSwtRadar PipelineR2Handling	Testcase to perform te swt radar pipeline (software watchdog timer) r2 fault handling for sm63

3.172 Test Suite rfe_TS_3112

QUALIFICATION Test Suite TS_3112 - TS_3112.

Table 176. Test Sequence rfe_TS_3112

Test Case	Description
rfe_tc_3112_sm68RfgCrcAdpll	Testcase to perform rfe register dig crc adpll testing idx e r2 falut handling for sm68

3.173 Test Suite rfe_TS_3113

QUALIFICATION Test Suite TS_3113 - TS_3113.

Table 177. Test Sequence rfe_TS_3113

Test Case	Description
rfe_tc_3113_sm68RfgCrcMcgen	Testcase to perform rfe register dig crc mcgen testing idx e r2 falut handling for sm68

3.174 Test Suite rfe_TS_3114

QUALIFICATION Test Suite TS_3114 - TS_3114.

Table 178. Test Sequence rfe_TS_3114

Test Case	Description
rfe_tc_3114_sm69RfgCrcAdc1	Testcase to perform rfe register dig crc adc1 testing idx e r2 falut handling for sm69

3.175 Test Suite rfe_TS_3115

QUALIFICATION Test Suite TS_3115 - TS_3115.

Table 179. Test Sequence rfe_TS_3115

Test Case	Description
rfe_tc_3115_sm69RfgCrcAdc2	Testcase to perform rfe register dig crc adc2 testing idx e r2 falut handling for sm69

3.176 Test Suite rfe_TS_3116

QUALIFICATION Test Suite TS_3116 - TS_3116.

Table 180. Test Sequence rfe_TS_3116

Test Case	Description
rfe_tc_3116_sm69RfgCrcAdc3	Testcase to perform rfe register dig crc adc3 testing idx e r2 falut handling for sm69

3.177 Test Suite rfe_TS_3117

QUALIFICATION Test Suite TS_3117 - TS_3117.

Table 181. Test Sequence rfe_TS_3117

Test Case	Description
rfe_tc_3117_sm69RfgCrcAdc4	Testcase to perform rfe register dig crc adc4 testing idx e r2 falut handling for sm69

3.178 Test Suite rfe_TS_3118

QUALIFICATION Test Suite TS_3118 - TS_3118.

Table 182. Test Sequence rfe_TS_3118

Test Case	Description
rfe_tc_3118_sm69RfgCrcAtb	Testcase to perform rfe register dig crc atb testing idx e r2 falut handling for sm69

3.179 Test Suite rfe_TS_3119

QUALIFICATION Test Suite TS_3119 - TS_3119.

Table 183. Test Sequence rfe_TS_3119

Test Case	Description
rfe_tc_3119_sm69RfgCrcChirpPll	Testcase to perform rfe register dig crc chirppll testing idx e r2 falut handling for sm69

3.180 Test Suite rfe_TS_3120

QUALIFICATION Test Suite TS_3120 - TS_3120.

Table 184. Test Sequence rfe_TS_3120

Test Case	Description
rfe_tc_3120_sm69RfgCrcGbias	Testcase to perform rfe register dig crc gbias testing idx e r2 falut handling for sm69

3.181 Test Suite rfe_TS_3121

QUALIFICATION Test Suite TS_3121 - TS_3121.

Table 185. Test Sequence rfe_TS_3121

Test Case	Description
rfe_tc_3121_sm69RfgCrcGldo	Testcase to perform rfe register dig crc gldo testing idx e r2 falut handling for sm69

3.182 Test Suite rfe_TS_3122

QUALIFICATION Test Suite TS_3122 - TS_3122.

Table 186. Test Sequence rfe_TS_3122

Test Case	Description
rfe_tc_3122_sm69RfgCrcIsm	Testcase to perform rfe register dig crc ism testing idx e r2 falut handling for sm69

3.183 Test Suite rfe_TS_3123

QUALIFICATION Test Suite TS_3123 - TS_3123.

Table 187. Test Sequence rfe_TS_3123

Test Case	Description
rfe_tc_3123_sm69RfgCrcLldo	Testcase to perform rfe register dig crc lldodig testing idx e r2 falut handling for sm69

3.184 Test Suite rfe_TS_3124

QUALIFICATION Test Suite TS_3124 - TS_3124.

Table 188. Test Sequence rfe_TS_3124

Test Case	Description
rfe_tc_3124_sm69RfgCrcLldoPdc	Testcase to perform rfe register dig crc lldopdc testing idx e r2 falut handling for sm69

3.185 Test Suite rfe_TS_3125

QUALIFICATION Test Suite TS_3125 - TS_3125.

Table 189. Test Sequence rfe_TS_3125

Test Case	Description
rfe_tc_3125_sm69RfgCrcLoif	Testcase to perform rfe register dig crc loif testing idx e r2 falut handling for sm69

3.186 Test Suite rfe_TS_3126

QUALIFICATION Test Suite TS_3126 - TS_3126.

Table 190. Test Sequence rfe_TS_3126

Test Case	Description
rfe_tc_3126_sm69RfgCrcPdc1	Testcase to perform rfe register dig crc pdc1 testing idx e r2 falut handling for sm69

3.187 Test Suite rfe_TS_3127

QUALIFICATION Test Suite TS_3127 - TS_3127.

Table 191. Test Sequence rfe_TS_3127

Test Case	Description
rfe_tc_3127_sm69RfgCrcPdc2	Testcase to perform rfe register dig crc pdc2 testing idx e r2 falut handling for sm69

3.188 Test Suite rfe_TS_3128

QUALIFICATION Test Suite TS_3128 - TS_3128.

Table 192. Test Sequence rfe_TS_3128

Test Case	Description
rfe_tc_3128_sm69RfgCrcPdc3	Testcase to perform rfe register dig crc pdc3 testing idx e r2 falut handling for sm69

3.189 Test Suite rfe_TS_3129

QUALIFICATION Test Suite TS_3129 - TS_3129.

Table 193. Test Sequence rfe_TS_3129

Test Case	Description
rfe_tc_3129_sm69RfgCrcPdc4	Testcase to perform rfe register dig crc pdc4 testing idx e r2 falut handling for sm69

3.190 Test Suite rfe_TS_3130

QUALIFICATION Test Suite TS_3130 - TS_3130.

Table 194. Test Sequence rfe_TS_3130

Test Case	Description
rfe_tc_3130_sm69RfgCrcRcosc	Testcase to perform rfe register dig crc rcosc testing idx e r2 falut handling for sm69

3.191 Test Suite rfe_TS_3131

QUALIFICATION Test Suite TS_3131 - TS_3131.

Table 195. Test Sequence rfe_TS_3131

Test Case	Description
rfe_tc_3131_sm69RfgCrcRx1	Testcase to perform rfe register dig crc rx1 testing idx e r2 falut handling for sm69

3.192 Test Suite rfe_TS_3132

QUALIFICATION Test Suite TS_3132 - TS_3132.

Table 196. Test Sequence rfe_TS_3132

Test Case	Description
rfe_tc_3132_sm69RfgCrcRx2	Testcase to perform rfe register dig crc rx2 testing idx e r2 falut handling for sm69

3.193 Test Suite rfe_TS_3133

QUALIFICATION Test Suite TS_3133 - TS_3133.

Table 197. Test Sequence rfe_TS_3133

Test Case	Description
rfe_tc_3133_sm69RfgCrcRx3	Testcase to perform rfe register dig crc rx3 testing idx e r2 falut handling for sm69

3.194 Test Suite rfe_TS_3134

QUALIFICATION Test Suite TS_3134 - TS_3134.

Table 198. Test Sequence rfe_TS_3134

Test Case	Description
rfe_tc_3134_sm69RfgCrcRx4	Testcase to perform rfe register dig crc rx4 testing idx e r2 falut handling for sm69

3.195 Test Suite rfe_TS_3135

QUALIFICATION Test Suite TS_3135 - TS_3135.

Table 199. Test Sequence rfe_TS_3135

Test Case	Description
rfe_tc_3135_sm69RfgCrcRxBist	Testcase to perform rfe register dig crc rxbist testing idx e r2 falut handling for sm69

3.196 Test Suite rfe_TS_3136

QUALIFICATION Test Suite TS_3136 - TS_3136.

Table 200. Test Sequence rfe_TS_3136

Test Case	Description
rfe_tc_3136_sm69RfgCrcSpim	Testcase to perform rfe register dig crc spim testing idx e r2 falut handling for sm69

3.197 Test Suite rfe_TS_3137

QUALIFICATION Test Suite TS_3137 - TS_3137.

Table 201. Test Sequence rfe_TS_3137

Test Case	Description
rfe_tc_3137_sm69RfgCrcTe	Testcase to perform rfe register dig crc te testing idx e r2 falut handling for sm69

3.198 Test Suite rfe_TS_3138

QUALIFICATION Test Suite TS_3138 - TS_3138.

Table 202. Test Sequence rfe_TS_3138

Test Case	Description
rfe_tc_3138_sm69RfgCrcTempSensor1	Testcase to perform rfe register dig crc tempsensor1 testing idx e r2 falut handling for sm69

3.199 Test Suite rfe_TS_3139

QUALIFICATION Test Suite TS_3139 - TS_3139.

Table 203. Test Sequence rfe_TS_3139

Test Case	Description
rfe_tc_3139_sm69RfgCrcTempSensor2	Testcase to perform rfe register dig crc tempsensor2 testing idx e r2 falut handling for sm69

3.200 Test Suite rfe_TS_3140

QUALIFICATION Test Suite TS_3140 - TS_3140.

Table 204. Test Sequence rfe_TS_3140

Test Case	Description
rfe_tc_3140_sm69RfgCrcTempSensor3	Testcase to perform rfe register dig crc tempsensor3 testing idx e r2 falut handling for sm69

3.201 Test Suite rfe_TS_3141

QUALIFICATION Test Suite TS_3141 - TS_3141.

Table 205. Test Sequence rfe_TS_3141

Test Case	Description
rfe_tc_3141_sm69RfgCrcTempSensor4	Testcase to perform rfe register dig crc tempsensor4 testing idx e r2 falut handling for sm69

3.202 Test Suite rfe_TS_3142

QUALIFICATION Test Suite TS_3142 - TS_3142.

Table 206. Test Sequence rfe_TS_3142

Test Case	Description
rfe_tc_3142_sm69RfgCrcTx1	Testcase to perform rfe register dig crc tx1 testing idx e r2 falut handling for sm69

3.203 Test Suite rfe_TS_3143

QUALIFICATION Test Suite TS_3143 - TS_3143.

Table 207. Test Sequence rfe_TS_3143

Test Case	Description
rfe_tc_3143_sm69RfgCrcTx2	Testcase to perform rfe register dig crc tx2 testing idx e r2 falut handling for sm69

3.204 Test Suite rfe_TS_3144

QUALIFICATION Test Suite TS_3144 - TS_3144.

Table 208. Test Sequence rfe_TS_3144

Test Case	Description
rfe_tc_3144_sm69RfgCrcTx3	Testcase to perform rfe register dig crc tx3 testing idx e r2 falut handling for sm69

3.205 Test Suite rfe_TS_3145

QUALIFICATION Test Suite TS_3145 - TS_3145.

Table 209. Test Sequence rfe_TS_3145

Test Case	Description
rfe_tc_3145_sm69RfgCrcTx4	Testcase to perform rfe register dig crc tx4 testing idx e r2 falut handling for sm69

3.206 Test Suite rfe_TS_3146

QUALIFICATION Test Suite TS_3146 - TS_3146.

Table 210. Test Sequence rfe_TS_3146

Test Case	Description
rfe_tc_3146_sm70MosiCrcMcgen	Testcase to perform spi transaction crc for mcgen ip e mosi r2 fault injection test r2 falut handling for sm70

3.207 Test Suite rfe_TS_3147

QUALIFICATION Test Suite TS_3147 - TS_3147.

Table 211. Test Sequence rfe_TS_3147

Test Case	Description
rfe_tc_3147_sm70MisoCrcMcgen	Testcase to perform spi transaction crc for mcgen ip e miso r2 fault injection test r2 falut handling for sm70

3.208 Test Suite rfe_TS_3148

QUALIFICATION Test Suite TS_3148 - TS_3148.

Table 212. Test Sequence rfe_TS_3148

Test Case	Description
rfe_tc_3148_sm70MosiCrcChirpPll	Testcase to perform spi transaction crc for chirppll ip e mosi r2 fault injection test r2 falut handling for sm70

3.209 Test Suite rfe_TS_3149

QUALIFICATION Test Suite TS_3149 - TS_3149.

Table 213. Test Sequence rfe_TS_3149

Test Case	Description
rfe_tc_3149_sm70MisoCrcChirpPll	Testcase to perform spi transaction crc for chirppll ip e miso r2 fault injection test r2 falut handling for sm70

3.210 Test Suite rfe_TS_3150

QUALIFICATION Test Suite TS_3150 - TS_3150.

Table 214. Test Sequence rfe_TS_3150

Test Case	Description
rfe_tc_3150_sm70MosiCrcIsm	Testcase to perform spi transaction crc for ism ip e mosi r2 fault injection test r2 falut handling for sm70

3.211 Test Suite rfe_TS_3151

QUALIFICATION Test Suite TS_3151 - TS_3151.

Table 215. Test Sequence rfe_TS_3151

Test Case	Description
rfe_tc_3151_sm70MisoCrcIsm	Testcase to perform spi transaction crc for ism ip e miso r2 fault injection test r2 falut handling for sm70

3.212 Test Suite rfe_TS_3152

QUALIFICATION Test Suite TS_3152 - TS_3152.

Table 216. Test Sequence rfe_TS_3152

Test Case	Description
rfe_tc_3152_sm70MosiCrcSpim	Testcase to perform spi transaction crc for spim ip e mosi r2 fault injection test r2 falut handling for sm70

3.213 Test Suite rfe_TS_3153

QUALIFICATION Test Suite TS_3153 - TS_3153.

Table 217. Test Sequence rfe_TS_3153

Test Case	Description
rfe_tc_3153_sm70MisoCrcSpim	Testcase to perform spi transaction crc for spim ip e miso r2 fault injection test r2 falut handling for sm70

3.214 Test Suite rfe_TS_3154

QUALIFICATION Test Suite TS_3154 - TS_3154.

Table 218. Test Sequence rfe_TS_3154

Test Case	Description
rfe_tc_3154_sm70MosiCrcAtb	Testcase to perform spi transaction crc for atb ip e mosi r2 fault injection test r2 falut handling for sm70

3.215 Test Suite rfe_TS_3155

QUALIFICATION Test Suite TS_3155 - TS_3155.

Table 219. Test Sequence rfe_TS_3155

Test Case	Description
rfe_tc_3155_sm70MisoCrcAtb	Testcase to perform spi transaction crc for atb ip e miso r2 fault injection test r2 falut handling for sm70

3.216 Test Suite rfe_TS_3156

QUALIFICATION Test Suite TS_3156 - TS_3156.

Table 220. Test Sequence rfe_TS_3156

Test Case	Description
rfe_tc_3156_sm70MosiCrcTe	Testcase to perform spi transaction crc for te ip e mosi r2 fault injection test r2 falut handling for sm70

3.217 Test Suite rfe_TS_3157

QUALIFICATION Test Suite TS_3157 - TS_3157.

Table 221. Test Sequence rfe_TS_3157

Test Case	Description
rfe_tc_3157_sm70MisoCrcTe	Testcase to perform spi transaction crc for te ip e miso r2 fault injection test r2 falut handling for sm70

3.218 Test Suite rfe_TS_3158

QUALIFICATION Test Suite TS_3158 - TS_3158.

Table 222. Test Sequence rfe_TS_3158

Test Case	Description
rfe_tc_3158_sm70MosiCrcRxBist	Testcase to perform spi transaction crc for rxbist ip e mosi r2 fault injection test r2 falut handling for sm70

3.219 Test Suite rfe_TS_3159

QUALIFICATION Test Suite TS_3159 - TS_3159.

Table 223. Test Sequence rfe_TS_3159

Test Case	Description
rfe_tc_3159_sm70MisoCrcRxBist	Testcase to perform spi transaction crc for rxbist ip e miso r2 fault injection test r2 falut handling for sm70

3.220 Test Suite rfe_TS_3160

QUALIFICATION Test Suite TS_3160 - TS_3160.

Table 224. Test Sequence rfe_TS_3160

Test Case	Description
rfe_tc_3160_sm70MosiCrcGbias	Testcase to perform spi transaction crc for gbias ip e mosi r2 fault injection test r2 falut handling for sm70

3.221 Test Suite rfe_TS_3161

QUALIFICATION Test Suite TS_3161 - TS_3161.

Table 225. Test Sequence rfe_TS_3161

Test Case	Description
rfe_tc_3161_sm70MisoCrcGbias	Testcase to perform spi transaction crc for gbias ip e miso r2 fault injection test r2 falut handling for sm70

3.222 Test Suite rfe_TS_3162

QUALIFICATION Test Suite TS_3162 - TS_3162.

Table 226. Test Sequence rfe_TS_3162

Test Case	Description
rfe_tc_3162_sm70MosiCrcGldo	Testcase to perform spi transaction crc for gldo ip e mosi r2 fault injection test r2 falut handling for sm70

3.223 Test Suite rfe_TS_3163

QUALIFICATION Test Suite TS_3163 - TS_3163.

Table 227. Test Sequence rfe_TS_3163

Test Case	Description
rfe_tc_3163_sm70MisoCrcGldo	Testcase to perform spi transaction crc for gldo ip e miso r2 fault injection test r2 falut handling for sm70

3.224 Test Suite rfe_TS_3164

QUALIFICATION Test Suite TS_3164 - TS_3164.

Table 228. Test Sequence rfe_TS_3164

Test Case	Description
rfe_tc_3164_sm70MosiCrcRcosc	Testcase to perform spi transaction crc for rcosc ip e mosi r2 fault injection test r2 falut handling for sm70

3.225 Test Suite rfe_TS_3165

QUALIFICATION Test Suite TS_3165 - TS_3165.

Table 229. Test Sequence rfe_TS_3165

Test Case	Description
rfe_tc_3165_sm70MisoCrcRcosc	Testcase to perform spi transaction crc for rcosc ip e miso r2 fault injection test r2 falut handling for sm70

3.226 Test Suite rfe_TS_3166

QUALIFICATION Test Suite TS_3166 - TS_3166.

Table 230. Test Sequence rfe_TS_3166

Test Case	Description
rfe_tc_3166_sm70MosiCrcLldo	Testcase to perform spi transaction crc for lldo ip e mosi r2 fault injection test r2 falut handling for sm70

3.227 Test Suite rfe_TS_3167

QUALIFICATION Test Suite TS_3167 - TS_3167.

Table 231. Test Sequence rfe_TS_3167

Test Case	Description
rfe_tc_3167_sm70MisoCrcLldo	Testcase to perform spi transaction crc for lldo ip e miso r2 fault injection test r2 falut handling for sm70

3.228 Test Suite rfe_TS_3168

QUALIFICATION Test Suite TS_3168 - TS_3168.

Table 232. Test Sequence rfe_TS_3168

Test Case	Description
rfe_tc_3168_sm70MosiCrcLldoPdc	Testcase to perform spi transaction crc for lldopdc ip e mosi r2 fault injection test r2 falut handling for sm70

3.229 Test Suite rfe_TS_3169

QUALIFICATION Test Suite TS_3169 - TS_3169.

Table 233. Test Sequence rfe_TS_3169

Test Case	Description
rfe_tc_3169_sm70MisoCrcLldoPdc	Testcase to perform spi transaction crc for lldopdc ip e miso r2 fault injection test r2 falut handling for sm70

3.230 Test Suite rfe_TS_3170

QUALIFICATION Test Suite TS_3170 - TS_3170.

Table 234. Test Sequence rfe_TS_3170

Test Case	Description
rfe_tc_3170_sm70MosiCrcLoif	Testcase to perform spi transaction crc for lo ip e mosi r2 fault injection test r2 falut handling for sm70

3.231 Test Suite rfe_TS_3171

QUALIFICATION Test Suite TS_3171 - TS_3171.

Table 235. Test Sequence rfe_TS_3171

Test Case	Description
rfe_tc_3171_sm70MisoCrcLoif	Testcase to perform spi transaction crc for lo ip e miso r2 fault injection test r2 falut handling for sm70

3.232 Test Suite rfe_TS_3172

QUALIFICATION Test Suite TS_3172 - TS_3172.

Table 236. Test Sequence rfe_TS_3172

Test Case	Description
rfe_tc_3172_sm70MosiCrcAdc1	Testcase to perform spi transaction crc for adc1 ip e mosi r2 fault injection test r2 falut handling for sm70

3.233 Test Suite rfe_TS_3173

QUALIFICATION Test Suite TS_3173 - TS_3173.

Table 237. Test Sequence rfe_TS_3173

Test Case	Description
rfe_tc_3173_sm70MisoCrcAdc1	Testcase to perform spi transaction crc for adc1 ip e miso r2 fault injection test r2 falut handling for sm70

3.234 Test Suite rfe_TS_3174

QUALIFICATION Test Suite TS_3174 - TS_3174.

Table 238. Test Sequence rfe_TS_3174

Test Case	Description
rfe_tc_3174_sm70MosiCrcADc2	Testcase to perform spi transaction crc for adc2 ip e mosi r2 fault injection test r2 falut handling for sm70

3.235 Test Suite rfe_TS_3175

QUALIFICATION Test Suite TS_3175 - TS_3175.

Table 239. Test Sequence rfe_TS_3175

Test Case	Description
rfe_tc_3175_sm70MisoCrcAdc2	Testcase to perform spi transaction crc for adc2 ip e miso r2 fault injection test r2 falut handling for sm70

3.236 Test Suite rfe_TS_3176

QUALIFICATION Test Suite TS_3176 - TS_3176.

Table 240. Test Sequence rfe_TS_3176

Test Case	Description
rfe_tc_3176_sm70MosiCrcAdc3	Testcase to perform spi transaction crc for adc3 ip e mosi r2 fault injection test r2 falut handling for sm70

3.237 Test Suite rfe_TS_3177

QUALIFICATION Test Suite TS_3177 - TS_3177.

Table 241. Test Sequence rfe_TS_3177

Test Case	Description
rfe_tc_3177_sm70MisoCrcAdc3	Testcase to perform spi transaction crc for adc3 ip e miso r2 fault injection test r2 falut handling for sm70

3.238 Test Suite rfe_TS_3178

QUALIFICATION Test Suite TS_3178 - TS_3178.

Table 242. Test Sequence rfe_TS_3178

Test Case	Description
rfe_tc_3178_sm70MosiCrcAdc4	Testcase to perform spi transaction crc for adc4 ip e mosi r2 fault injection test r2 falut handling for sm70

3.239 Test Suite rfe_TS_3179

QUALIFICATION Test Suite TS_3179 - TS_3179.

Table 243. Test Sequence rfe_TS_3179

Test Case	Description
rfe_tc_3179_sm70MisoCrcAdc4	Testcase to perform spi transaction crc for adc4 ip e miso r2 fault injection test r2 falut handling for sm70

3.240 Test Suite rfe_TS_3180

QUALIFICATION Test Suite TS_3180 - TS_3180.

Table 244. Test Sequence rfe_TS_3180

Test Case	Description
rfe_tc_3180_sm70MosiCrcPdc1	Testcase to perform spi transaction crc for pdc1 ip e mosi r2 fault injection test r2 falut handling for sm70

3.241 Test Suite rfe_TS_3181

QUALIFICATION Test Suite TS_3181 - TS_3181.

Table 245. Test Sequence rfe_TS_3181

Test Case	Description
rfe_tc_3181_sm70MisoCrcPdc1	Testcase to perform spi transaction crc for pdc1 ip e miso r2 fault injection test r2 falut handling for sm70

3.242 Test Suite rfe_TS_3182

QUALIFICATION Test Suite TS_3182 - TS_3182.

Table 246. Test Sequence rfe_TS_3182

Test Case	Description
rfe_tc_3182_sm70MosiCrcPdc2	Testcase to perform spi transaction crc for pdc2 ip e mosi r2 fault injection test r2 falut handling for sm70

3.243 Test Suite rfe_TS_3183

QUALIFICATION Test Suite TS_3183 - TS_3183.

Table 247. Test Sequence rfe_TS_3183

Test Case	Description
rfe_tc_3183_sm70MisoCrcPdc2	Testcase to perform spi transaction crc for pdc2 ip e miso r2 fault injection test r2 falut handling for sm70

3.244 Test Suite rfe_TS_3184

QUALIFICATION Test Suite TS_3184 - TS_3184.

Table 248. Test Sequence rfe_TS_3184

Test Case	Description
rfe_tc_3184_sm70MosiCrcPdc3	Testcase to perform spi transaction crc for pdc3 ip e mosi r2 fault injection test r2 falut handling for sm70

3.245 Test Suite rfe_TS_3185

QUALIFICATION Test Suite TS_3185 - TS_3185.

Table 249. Test Sequence rfe_TS_3185

Test Case	Description
rfe_tc_3185_sm70MisoCrcPdc3	Testcase to perform spi transaction crc for pdc3 ip e miso r2 fault injection test r2 falut handling for sm70

3.246 Test Suite rfe_TS_3186

QUALIFICATION Test Suite TS_3186 - TS_3186.

Table 250. Test Sequence rfe_TS_3186

Test Case	Description
rfe_tc_3186_sm70MosiCrcPdc4	Testcase to perform spi transaction crc for pdc4 ip e mosi r2 fault injection test r2 falut handling for sm70

3.247 Test Suite rfe_TS_3187

QUALIFICATION Test Suite TS_3187 - TS_3187.

Table 251. Test Sequence rfe_TS_3187

Test Case	Description
rfe_tc_3187_sm70MisoCrcPdc4	Testcase to perform spi transaction crc for pdc4 ip e miso r2 fault injection test r2 falut handling for sm70

3.248 Test Suite rfe_TS_3188

QUALIFICATION Test Suite TS_3188 - TS_3188.

Table 252. Test Sequence rfe_TS_3188

Test Case	Description
rfe_tc_3188_sm70MosiCrcTx1	Testcase to perform spi transaction crc for tx1 ip e mosi r2 fault injection test r2 falut handling for sm70

3.249 Test Suite rfe_TS_3189

QUALIFICATION Test Suite TS_3189 - TS_3189.

Table 253. Test Sequence rfe_TS_3189

Test Case	Description
rfe_tc_3189_sm70MisoCrcTx1	Testcase to perform spi transaction crc for tx1 ip e miso r2 fault injection test r2 falut handling for sm70

3.250 Test Suite rfe_TS_3190

QUALIFICATION Test Suite TS_3190 - TS_3190.

Table 254. Test Sequence rfe_TS_3190

Test Case	Description
rfe_tc_3190_sm70MosiCrcTx2	Testcase to perform spi transaction crc for tx2 ip e mosi r2 fault injection test r2 falut handling for sm70

3.251 Test Suite rfe_TS_3191

QUALIFICATION Test Suite TS_3191 - TS_3191.

Table 255. Test Sequence rfe_TS_3191

Test Case	Description
rfe_tc_3191_sm70MisoCrcTx2	Testcase to perform spi transaction crc for tx2 ip e miso r2 fault injection test r2 falut handling for sm70

3.252 Test Suite rfe_TS_3192

QUALIFICATION Test Suite TS_3192 - TS_3192.

Table 256. Test Sequence rfe_TS_3192

Test Case	Description
rfe_tc_3192_sm70MosiCrcTx3	Testcase to perform spi transaction crc for tx3 ip e mosi r2 fault injection test r2 falut handling for sm70

3.253 Test Suite rfe_TS_3193

QUALIFICATION Test Suite TS_3193 - TS_3193.

Table 257. Test Sequence rfe_TS_3193

Test Case	Description
rfe_tc_3193_sm70MisoCrcTx3	Testcase to perform spi transaction crc for tx3 ip e miso r2 fault injection test r2 falut handling for sm70

3.254 Test Suite rfe_TS_3194

QUALIFICATION Test Suite TS_3194 - TS_3194.

Table 258. Test Sequence rfe_TS_3194

Test Case	Description
rfe_tc_3194_sm70MosiCrcTx4	Testcase to perform spi transaction crc for tx4 ip e mosi r2 fault injection test r2 falut handling for sm70

3.255 Test Suite rfe_TS_3195

QUALIFICATION Test Suite TS_3195 - TS_3195.

Table 259. Test Sequence rfe_TS_3195

Test Case	Description
rfe_tc_3195_sm70MisoCrcTx4	Testcase to perform spi transaction crc for tx4 ip e miso r2 fault injection test r2 falut handling for sm70

3.256 Test Suite rfe_TS_3196

QUALIFICATION Test Suite TS_3196 - TS_3196.

Table 260. Test Sequence rfe_TS_3196

Test Case	Description
rfe_tc_3196_sm70MosiCrcRx1	Testcase to perform spi transaction crc for rx1 ip e mosi r2 fault injection test r2 falut handling for sm70

3.257 Test Suite rfe_TS_3197

QUALIFICATION Test Suite TS_3197 - TS_3197.

Table 261. Test Sequence rfe_TS_3197

Test Case	Description
rfe_tc_3197_sm70MisoCrcRx1	Testcase to perform spi transaction crc for rx1 ip e miso r2 fault injection test r2 falut handling for sm70

3.258 Test Suite rfe_TS_3198

QUALIFICATION Test Suite TS_3198 - TS_3198.

Table 262. Test Sequence rfe_TS_3198

Test Case	Description
rfe_tc_3198_sm70MosiCrcRx2	Testcase to perform spi transaction crc for rx2 ip e mosi r2 fault injection test r2 falut handling for sm70

3.259 Test Suite rfe_TS_3199

QUALIFICATION Test Suite TS_3199 - TS_3199.

Table 263. Test Sequence rfe_TS_3199

Test Case	Description
rfe_tc_3199_sm70MisoCrcRx2	Testcase to perform spi transaction crc for rx2 ip e miso r2 fault injection test r2 falut handling for sm70

3.260 Test Suite rfe_TS_3200

QUALIFICATION Test Suite TS_3200 - TS_3200.

Table 264. Test Sequence rfe_TS_3200

Test Case	Description
rfe_tc_3200_sm70MosiCrcRx3	Testcase to perform spi transaction crc for rx3 ip e mosi r2 fault injection test r2 falut handling for sm70

3.261 Test Suite rfe_TS_3201

QUALIFICATION Test Suite TS_3201 - TS_3201.

Table 265. Test Sequence rfe_TS_3201

Test Case	Description
rfe_tc_3201_sm70MisoCrcRx3	Testcase to perform spi transaction crc for rx3 ip e miso r2 fault injection test r2 falut handling for sm70

3.262 Test Suite rfe_TS_3202

QUALIFICATION Test Suite TS_3202 - TS_3202.

Table 266. Test Sequence rfe_TS_3202

Test Case	Description
rfe_tc_3202_sm70MosiCrcRx4	Testcase to perform spi transaction crc for rx4 ip e mosi r2 fault injection test r2 falut handling for sm70

3.263 Test Suite rfe_TS_3203

QUALIFICATION Test Suite TS_3203 - TS_3203.

Table 267. Test Sequence rfe_TS_3203

Test Case	Description
rfe_tc_3203_sm70MisoCrcRx4	Testcase to perform spi transaction crc for rx4 ip e miso r2 fault injection test r2 falut handling for sm70

3.264 Test Suite rfe_TS_3204

QUALIFICATION Test Suite TS_3204 - TS_3204.

Table 268. Test Sequence rfe_TS_3204

Test Case	Description
rfe_tc_3204_sm70MosiCrc Tempsensor1	Testcase to perform spi transaction crc for tempsensor1 ip e mosi r2 fault injection test for sm70

3.265 Test Suite rfe_TS_3205

QUALIFICATION Test Suite TS_3205 - TS_3205.

Table 269. Test Sequence rfe_TS_3205

Test Case	Description
rfe_tc_3205_sm70MisoCrc Tempsensor1	Testcase to perform spi transaction crc for tempsensor1 ip e miso r2 fault injection test for sm70

3.266 Test Suite rfe_TS_3206

QUALIFICATION Test Suite TS_3206 - TS_3206.

Table 270. Test Sequence rfe_TS_3206

Test Case	Description
rfe_tc_3206_sm70MosiCrc Tempsensor2	Testcase to perform spi transaction crc for tempsensor2 ip e mosi r2 fault injection test for sm70

3.267 Test Suite rfe_TS_3207

QUALIFICATION Test Suite TS_3207 - TS_3207.

Table 271. Test Sequence rfe_TS_3207

Test Case	Description
rfe_tc_3207_sm70MisoCrc Tempsensor2	Testcase to perform spi transaction crc for tempsensor2 ip e miso r2 fault injection test for sm70

3.268 Test Suite rfe_TS_3208

QUALIFICATION Test Suite TS_3208 - TS_3208.

Table 272. Test Sequence rfe_TS_3208

Test Case	Description
rfe_tc_3208_sm70MosiCrc Tempsensor3	Testcase to perform spi transaction crc for tempsensor3 ip e mosi r2 fault injection test for sm70

3.269 Test Suite rfe_TS_3209

QUALIFICATION Test Suite TS_3209 - TS_3209.

Table 273. Test Sequence rfe_TS_3209

Test Case	Description
rfe_tc_3209_sm70MisoCrc Tempsensor3	Testcase to perform spi transaction crc for tempsensor3 ip e miso r2 fault injection test for sm70

3.270 Test Suite rfe_TS_3210

QUALIFICATION Test Suite TS_3210 - TS_3210.

Table 274. Test Sequence rfe_TS_3210

Test Case	Description
rfe_tc_3210_sm70MosiCrc Tempsensor4	Testcase to perform spi transaction crc for tempsensor4 ip e mosi r2 fault injection test for sm70

3.271 Test Suite rfe_TS_3211

QUALIFICATION Test Suite TS_3211 - TS_3211.

Table 275. Test Sequence rfe_TS_3211

Test Case	Description
rfe_tc_3211_sm70MisoCrc Tempsensor4	Testcase to perform spi transaction crc for tempsensor4 ip e miso r2 fault injection test for sm70

3.272 Test Suite rfe_TS_3212

QUALIFICATION Test Suite TS_3212 - TS_3212.

Table 276. Test Sequence rfe_TS_3212

Test Case	Description
rfe_tc_3212_sm70MosiCrcAdpll	Testcase to perform spi transaction crc for adpll ip e mosi r2 fault injection test for sm70

3.273 Test Suite rfe_TS_3213

QUALIFICATION Test Suite TS_3213 - TS_3213.

Table 277. Test Sequence rfe_TS_3213

Test Case	Description
rfe_tc_3213_sm70MisoCrcAdpll	Testcase to perform spi transaction crc for adpll ip e miso r2 fault injection test for sm70

3.274 Test Suite rfe_TS_3214

QUALIFICATION Test Suite TS_3214 - TS_3214.

Table 278. Test Sequence rfe_TS_3214

Test Case	Description
rfe_tc_3214_sm86FpuR1FaultHandling	Testcase to perform rfe m7 fpu error r1 fault injection and r2 promotion for sm86

3.275 Test Suite rfe_TS_3215

QUALIFICATION Test Suite TS_3215 - TS_3215.

Table 279. Test Sequence rfe_TS_3215

Test Case	Description
rfe_tc_3215_sm86FpuR2FaultPromotion	Testcase to perform rfe m7 fpu error r1 fault injection and r2 promotion for sm86

3.276 Test Suite rfe_TS_3216

QUALIFICATION Test Suite TS_3216 - TS_3216.

Table 280. Test Sequence rfe_TS_3216

Test Case	Description
rfe_tc_3216_sm88DtcmEccR2FaultHandling	Testcase to perform ida cpuctrl cm7 it64 dt32 mem dtcm ecc r2 fault handling for sm88

3.277 Test Suite rfe_TS_3217

QUALIFICATION Test Suite TS_3217 - TS_3217.

Table 281. Test Sequence rfe_TS_3217

Test Case	Description
rfe_tc_3217_sm89ItcmR2FaultHandling	Testcase to perform ida cpuctrl cm7 it64 dt32 mem itcm eccr2 fault handling for sm89

3.278 Test Suite rfe_TS_3218

QUALIFICATION Test Suite TS_3218 - TS_3218.

Table 282. Test Sequence rfe_TS_3218

Test Case	Description
rfe_tc_3218_sm92SoftwareFaultR2Handling	Testcase to perform rfe m7 core swt (software watchdog timer) r2 fault handling for sm92

3.279 Test Suite rfe_TS_3219

QUALIFICATION Test Suite TS_3219 - TS_3219.

Table 283. Test Sequence rfe_TS_3219

Test Case	Description
rfe_tc_3219_sm93FlexNocCrcR2FaultHandling	Testcase to perform ida rfe m7 flexnoc (crc) r2 fault handling for sm93

3.280 Test Suite rfe_TS_3220

QUALIFICATION Test Suite TS_3220 - TS_3220.

Table 284. Test Sequence rfe_TS_3220

Test Case	Description
rfe_tc_3220_sm93Apb2SpiCrcR2FaultHandling	Testcase to perform ida rfe m7 apb2spi (crc) r2 fault handling for sm93

3.281 Test Suite rfe_TS_3221

QUALIFICATION Test Suite TS_3221 - TS_3221.

Table 285. Test Sequence rfe_TS_3221

Test Case	Description
rfe_tc_3221_sm93PitR2FAultHandling	Testcase to perform ida rfe m7 pit (xbic) r2 fault handling for sm93

3.282 Test Suite rfe_TS_3222

QUALIFICATION Test Suite TS_3222 - TS_3222.

Table 286. Test Sequence rfe_TS_3222

Test Case	Description
rfe_tc_3222_sm94MessageE2ER2FaultPromotion	Testcase to perform ida rfe m7 message e2e (r2 fh) special rfe driver code change is needed r1 fault handling and r2 promotion for sm94

3.283 Test Suite rfe_TS_3223

QUALIFICATION Test Suite TS_3223 - TS_3223.

Table 287. Test Sequence rfe_TS_3223

Test Case	Description
rfe_tc_3223_sm94MessageE2EClear	Testcase to perform ida rfe m7 message e2e (clean) special rfe driver code change is needed for sm94

3.284 Test Suite rfe_TS_3224

QUALIFICATION Test Suite TS_3224 - TS_3224.

Table 288. Test Sequence rfe_TS_3224

Test Case	Description
rfe_tc_3224_sm94MessageE2ER1FaultHandling	Testcase to perform ida rfe m7 message e2e (r1 fi) special rfe driver code change is needed r1 fault handling for sm94

3.285 Test Suite rfe_TS_3225

QUALIFICATION Test Suite TS_3225 - TS_3225.

Table 289. Test Sequence rfe_TS_3225

Test Case	Description
rfe_tc_3225_sm98SpiAccessMosiMcgen	Testcase to perform m7 spi access ana for mcgen ip e mosi r2 fault injection test for sm98

3.286 Test Suite rfe_TS_3226

QUALIFICATION Test Suite TS_3226 - TS_3226.

Table 290. Test Sequence rfe_TS_3226

Test Case	Description
rfe_tc_3226_sm98SpiAccessMisoMcgen	Testcase to perform m7 spi access ana for mcgen ip e miso r2 fault injection test for sm98

3.287 Test Suite rfe_TS_3227

QUALIFICATION Test Suite TS_3227 - TS_3227.

Table 291. Test Sequence rfe_TS_3227

Test Case	Description
rfe_tc_3227_sm98SpiAccessMosiChirppll	Testcase to perform m7 spi access ana for chirppll ip e mosi r2 fault injection test for sm98

3.288 Test Suite rfe_TS_3228

QUALIFICATION Test Suite TS_3228 - TS_3228.

Table 292. Test Sequence rfe_TS_3228

Test Case	Description
rfe_tc_3228_sm98SpiAccessMisoChirppll	Testcase to perform m7 spi access ana for chirppll ip e miso r2 fault injection test for sm98

3.289 Test Suite rfe_TS_3229

QUALIFICATION Test Suite TS_3229 - TS_3229.

Table 293. Test Sequence rfe_TS_3229

Test Case	Description
rfe_tc_3229_sm98SpiAccessMosiIsm	Testcase to perform m7 spi access ana for ism ip e mosi r2 fault injection test for sm98

3.290 Test Suite rfe_TS_3230

QUALIFICATION Test Suite TS_3230 - TS_3230.

Table 294. Test Sequence rfe_TS_3230

Test Case	Description
rfe_tc_3230_sm98SpiAccessMisoIsm	Testcase to perform m7 spi access ana for ism ip e miso r2 fault injection test for sm98

3.291 Test Suite rfe_TS_3231

QUALIFICATION Test Suite TS_3231 - TS_3231.

Table 295. Test Sequence rfe_TS_3231

Test Case	Description
rfe_tc_3231_sm98SpiAccessMosiSpim	Testcase to perform m7 spi access ana for spim ip e mosi r2 fault injection test for sm98

3.292 Test Suite rfe_TS_3232

QUALIFICATION Test Suite TS_3232 - TS_3232.

Table 296. Test Sequence rfe_TS_3232

Test Case	Description
rfe_tc_3232_sm98SpiAccessMisoSpim	Testcase to perform m7 spi access ana for spim ip e miso r2 fault injection test for sm98

3.293 Test Suite rfe_TS_3233

QUALIFICATION Test Suite TS_3233 - TS_3233.

Table 297. Test Sequence rfe_TS_3233

Test Case	Description
rfe_tc_3233_sm98SpiAccessMosiAtb	Testcase to perform m7 spi access ana for atb ip e mosi r2 fault injection test for sm98

3.294 Test Suite rfe_TS_3234

QUALIFICATION Test Suite TS_3234 - TS_3234.

Table 298. Test Sequence rfe_TS_3234

Test Case	Description
rfe_tc_3234_sm98SpiAccessMisoAtb	Testcase to perform m7 spi access ana for atb ip e miso r2 fault injection test for sm98

3.295 Test Suite rfe_TS_3235

QUALIFICATION Test Suite TS_3235 - TS_3235.

Table 299. Test Sequence rfe_TS_3235

Test Case	Description
rfe_tc_3235_sm98SpiAccessMosiTe	Testcase to perform m7 spi access ana for te ip e mosi r2 fault injection test for sm98

3.296 Test Suite rfe_TS_3236

QUALIFICATION Test Suite TS_3236 - TS_3236.

Table 300. Test Sequence rfe_TS_3236

Test Case	Description
rfe_tc_3236_sm98SpiAccessMisoTe	Testcase to perform m7 spi access ana for te ip e miso r2 fault injection test for sm98

3.297 Test Suite rfe_TS_3237

QUALIFICATION Test Suite TS_3237 - TS_3237.

Table 301. Test Sequence rfe_TS_3237

Test Case	Description
rfe_tc_3237_sm98SpiAccessMosiRxbist	Testcase to perform m7 spi access ana for rxbist ip e mosi r2 fault injection test for sm98

3.298 Test Suite rfe_TS_3238

QUALIFICATION Test Suite TS_3238 - TS_3238.

Table 302. Test Sequence rfe_TS_3238

Test Case	Description
rfe_tc_3238_sm98SpiAccessMisoRxbist	Testcase to perform m7 spi access ana for rxbist ip e miso r2 fault injection test for sm98

3.299 Test Suite rfe_TS_3239

QUALIFICATION Test Suite TS_3239 - TS_3239.

Table 303. Test Sequence rfe_TS_3239

Test Case	Description
rfe_tc_3239_sm98SpiAccessMosiGbias	Testcase to perform m7 spi access ana for gbias ip e mosi r2 fault injection test for sm98

3.300 Test Suite rfe_TS_3240

QUALIFICATION Test Suite TS_3240 - TS_3240.

Table 304. Test Sequence rfe_TS_3240

Test Case	Description
rfe_tc_3240_sm98SpiAccessMisoGbias	Testcase to perform m7 spi access ana for gbias ip e miso r2 fault injection test for sm98

3.301 Test Suite rfe_TS_3241

QUALIFICATION Test Suite TS_3241 - TS_3241.

Table 305. Test Sequence rfe_TS_3241

Test Case	Description
rfe_tc_3241_sm98SpiAccessMosiGldo	Testcase to perform m7 spi access ana for gldo ip e mosi r2 fault injection test for sm98

3.302 Test Suite rfe_TS_3242

QUALIFICATION Test Suite TS_3242 - TS_3242.

Table 306. Test Sequence rfe_TS_3242

Test Case	Description
rfe_tc_3242_sm98SpiAccessMisoGldo	Testcase to perform m7 spi access ana for gldo ip e miso r2 fault injection test for sm98

3.303 Test Suite rfe_TS_3243

QUALIFICATION Test Suite TS_3243 - TS_3243.

Table 307. Test Sequence rfe_TS_3243

Test Case	Description
rfe_tc_3243_sm98SpiAccessMosiRcosc	Testcase to perform m7 spi access ana for rcosc ip e mosi r2 fault injection test for sm98

3.304 Test Suite rfe_TS_3244

QUALIFICATION Test Suite TS_3244 - TS_3244.

Table 308. Test Sequence rfe_TS_3244

Test Case	Description
rfe_tc_3244_sm98SpiAccessMisoRcosc	Testcase to perform m7 spi access ana for rcosc ip e miso r2 fault injection test for sm98

3.305 Test Suite rfe_TS_3245

QUALIFICATION Test Suite TS_3245 - TS_3245.

Table 309. Test Sequence rfe_TS_3245

Test Case	Description
rfe_tc_3245_sm98SpiAccessMosiLldo	Testcase to perform m7 spi access ana for lldo ip e mosi r2 fault injection test for sm98

3.306 Test Suite rfe_TS_3246

QUALIFICATION Test Suite TS_3246 - TS_3246.

Table 310. Test Sequence rfe_TS_3246

Test Case	Description
rfe_tc_3246_sm98SpiAccessMiso Lldo	Testcase to perform m7 spi access ana for lldo ip e miso r2 fault injection test for sm98

3.307 Test Suite rfe_TS_3247

QUALIFICATION Test Suite TS_3247 - TS_3247.

Table 311. Test Sequence rfe_TS_3247

Test Case	Description
rfe_tc_3247_sm98SpiAccessMosi Lldopdc	Testcase to perform m7 spi access ana for lldopdc ip e mosi r2 fault injection test for sm98

3.308 Test Suite rfe_TS_3248

QUALIFICATION Test Suite TS_3248 - TS_3248.

Table 312. Test Sequence rfe_TS_3248

Test Case	Description
rfe_tc_3248_sm98SpiAccessMiso Lldopdc	Testcase to perform m7 spi access ana for lldopdc ip e miso r2 fault injection test for sm98

3.309 Test Suite rfe_TS_3249

QUALIFICATION Test Suite TS_3249 - TS_3249.

Table 313. Test Sequence rfe_TS_3249

Test Case	Description
rfe_tc_3249_sm98SpiAccessMosi Lo	Testcase to perform m7 spi access ana for lo ip e mosi r2 fault injection test for sm98

3.310 Test Suite rfe_TS_3250

QUALIFICATION Test Suite TS_3250 - TS_3250.

Table 314. Test Sequence rfe_TS_3250

Test Case	Description
rfe_tc_3250_sm98SpiAccessMiso Lo	Testcase to perform m7 spi access ana for lo ip e miso r2 fault injection test for sm98

3.311 Test Suite rfe_TS_3251

QUALIFICATION Test Suite TS_3251 - TS_3251.

Table 315. Test Sequence rfe_TS_3251

Test Case	Description
rfe_tc_3251_sm98SpiAccessMosi Adc1	Testcase to perform m7 spi access ana for adc1 ip e mosi r2 fault injection test for sm98

3.312 Test Suite rfe_TS_3252

QUALIFICATION Test Suite TS_3252 - TS_3252.

Table 316. Test Sequence rfe_TS_3252

Test Case	Description
rfe_tc_3252_sm98SpiAccessMisoAdc1	Testcase to perform m7 spi access ana for adc1 ip e miso r2 fault injection test for sm98

3.313 Test Suite rfe_TS_3253

QUALIFICATION Test Suite TS_3253 - TS_3253.

Table 317. Test Sequence rfe_TS_3253

Test Case	Description
rfe_tc_3253_sm98SpiAccessMosiAdc2	Testcase to perform m7 spi access ana for adc2 ip e mosi r2 fault injection test for sm98

3.314 Test Suite rfe_TS_3254

QUALIFICATION Test Suite TS_3254 - TS_3254.

Table 318. Test Sequence rfe_TS_3254

Test Case	Description
rfe_tc_3254_sm98SpiAccessMisoAdc2	Testcase to perform m7 spi access ana for adc2 ip e miso r2 fault injection test for sm98

3.315 Test Suite rfe_TS_3255

QUALIFICATION Test Suite TS_3255 - TS_3255.

Table 319. Test Sequence rfe_TS_3255

Test Case	Description
rfe_tc_3255_sm98SpiAccessMosiAdc3	Testcase to perform m7 spi access ana for adc3 ip e mosi r2 fault injection test for sm98

3.316 Test Suite rfe_TS_3256

QUALIFICATION Test Suite TS_3256 - TS_3256.

Table 320. Test Sequence rfe_TS_3256

Test Case	Description
rfe_tc_3256_sm98SpiAccessMisoAdc3	Testcase to perform m7 spi access ana for adc3 ip e miso r2 fault injection test for sm98

3.317 Test Suite rfe_TS_3257

QUALIFICATION Test Suite TS_3257 - TS_3257.

Table 321. Test Sequence rfe_TS_3257

Test Case	Description
rfe_tc_3257_sm98SpiAccessMosiAdc4	Testcase to perform m7 spi access ana for adc4 ip e mosi r2 fault injection test for sm98

3.318 Test Suite rfe_TS_3258

QUALIFICATION Test Suite TS_3258 - TS_3258.

Table 322. Test Sequence rfe_TS_3258

Test Case	Description
rfe_tc_3258_sm98SpiAccessMisoAdc4	Testcase to perform m7 spi access ana for adc4 ip e miso r2 fault injection test for sm98

3.319 Test Suite rfe_TS_3259

QUALIFICATION Test Suite TS_3259 - TS_3259.

Table 323. Test Sequence rfe_TS_3259

Test Case	Description
rfe_tc_3259_sm98SpiAccessMosiPdc1	Testcase to perform m7 spi access ana for pdc1 ip e mosi r2 fault injection test for sm98

3.320 Test Suite rfe_TS_3260

QUALIFICATION Test Suite TS_3260 - TS_3260.

Table 324. Test Sequence rfe_TS_3260

Test Case	Description
rfe_tc_3260_sm98SpiAccessMisoPdc1	Testcase to perform m7 spi access ana for pdc1 ip e miso r2 fault injection test for sm98

3.321 Test Suite rfe_TS_3261

QUALIFICATION Test Suite TS_3261 - TS_3261.

Table 325. Test Sequence rfe_TS_3261

Test Case	Description
rfe_tc_3261_sm98SpiAccessMosiPdc2	Testcase to perform m7 spi access ana for pdc2 ip e mosi r2 fault injection test for sm98

3.322 Test Suite rfe_TS_3262

QUALIFICATION Test Suite TS_3262 - TS_3262.

Table 326. Test Sequence rfe_TS_3262

Test Case	Description
rfe_tc_3262_sm98SpiAccessMisoPdc2	Testcase to perform m7 spi access ana for pdc2 ip e miso r2 fault injection test for sm98

3.323 Test Suite rfe_TS_3263

QUALIFICATION Test Suite TS_3263 - TS_3263.

Table 327. Test Sequence rfe_TS_3263

Test Case	Description
rfe_tc_3263_sm98SpiAccessMosiPdc3	Testcase to perform m7 spi access ana for pdc3 ip e mosi r2 fault injection test for sm98

3.324 Test Suite rfe_TS_3264

QUALIFICATION Test Suite TS_3264 - TS_3264.

Table 328. Test Sequence rfe_TS_3264

Test Case	Description
rfe_tc_3264_sm98SpiAccessMisoPdc3	Testcase to perform m7 spi access ana for pdc3 ip e miso r2 fault injection test for sm98

3.325 Test Suite rfe_TS_3265

QUALIFICATION Test Suite TS_3265 - TS_3265.

Table 329. Test Sequence rfe_TS_3265

Test Case	Description
rfe_tc_3265_sm98SpiAccessMosiPdc4	Testcase to perform m7 spi access ana for pdc4 ip e mosi r2 fault injection test for sm98

3.326 Test Suite rfe_TS_3266

QUALIFICATION Test Suite TS_3266 - TS_3266.

Table 330. Test Sequence rfe_TS_3266

Test Case	Description
rfe_tc_3266_sm98SpiAccessMisoPdc4	Testcase to perform m7 spi access ana for pdc4 ip e miso r2 fault injection test for sm98

3.327 Test Suite rfe_TS_3267

QUALIFICATION Test Suite TS_3267 - TS_3267.

Table 331. Test Sequence rfe_TS_3267

Test Case	Description
rfe_tc_3267_sm98SpiAccessMosiTx1	Testcase to perform m7 spi access ana for tx1 ip e mosi r2 fault injection test for sm98

3.328 Test Suite rfe_TS_3268

QUALIFICATION Test Suite TS_3268 - TS_3268.

Table 332. Test Sequence rfe_TS_3268

Test Case	Description
rfe_tc_3268_sm98SpiAccessMisoTx1	Testcase to perform m7 spi access ana for tx1 ip e miso r2 fault injection test for sm98

3.329 Test Suite rfe_TS_3269

QUALIFICATION Test Suite TS_3269 - TS_3269.

Table 333. Test Sequence rfe_TS_3269

Test Case	Description
rfe_tc_3269_sm98SpiAccessMosiTx2	Testcase to perform m7 spi access ana for tx2 ip e mosi r2 fault injection test for sm98

3.330 Test Suite rfe_TS_3270

QUALIFICATION Test Suite TS_3270 - TS_3270.

Table 334. Test Sequence rfe_TS_3270

Test Case	Description
rfe_tc_3270_sm98SpiAccessMisoTx2	Testcase to perform m7 spi access ana for tx2 ip e miso r2 fault injection test for sm98

3.331 Test Suite rfe_TS_3271

QUALIFICATION Test Suite TS_3271 - TS_3271.

Table 335. Test Sequence rfe_TS_3271

Test Case	Description
rfe_tc_3271_sm98SpiAccessMosiTx3	Testcase to perform m7 spi access ana for tx3 ip e mosi r2 fault injection test for sm98

3.332 Test Suite rfe_TS_3272

QUALIFICATION Test Suite TS_3272 - TS_3272.

Table 336. Test Sequence rfe_TS_3272

Test Case	Description
rfe_tc_3272_sm98SpiAccessMisoTx3	Testcase to perform m7 spi access ana for tx3 ip e miso r2 fault injection test for sm98

3.333 Test Suite rfe_TS_3273

QUALIFICATION Test Suite TS_3273 - TS_3273.

Table 337. Test Sequence rfe_TS_3273

Test Case	Description
rfe_tc_3273_sm98SpiAccessMosiTx4	Testcase to perform m7 spi access ana for tx4 ip e mosi r2 fault injection test for sm98

3.334 Test Suite rfe_TS_3274

QUALIFICATION Test Suite TS_3274 - TS_3274.

Table 338. Test Sequence rfe_TS_3274

Test Case	Description
rfe_tc_3274_sm98SpiAccessMiso Tx4	Testcase to perform m7 spi access ana for tx4 ip e miso r2 fault injection test for sm98

3.335 Test Suite rfe_TS_3275

QUALIFICATION Test Suite TS_3275 - TS_3275.

Table 339. Test Sequence rfe_TS_3275

Test Case	Description
rfe_tc_3275_sm98SpiAccessMosi Rx1	Testcase to perform m7 spi access ana for rx1 ip e mosi r2 fault injection test for sm98

3.336 Test Suite rfe_TS_3276

QUALIFICATION Test Suite TS_3276 - TS_3276.

Table 340. Test Sequence rfe_TS_3276

Test Case	Description
rfe_tc_3276_sm98SpiAccessMiso Rx1	Testcase to perform m7 spi access ana for rx1 ip e miso r2 fault injection test for sm98

3.337 Test Suite rfe_TS_3277

QUALIFICATION Test Suite TS_3277 - TS_3277.

Table 341. Test Sequence rfe_TS_3277

Test Case	Description
rfe_tc_3277_sm98SpiAccessMosi Rx2	Testcase to perform m7 spi access ana for rx2 ip e mosi r2 fault injection test for sm98

3.338 Test Suite rfe_TS_3278

QUALIFICATION Test Suite TS_3278 - TS_3278.

Table 342. Test Sequence rfe_TS_3278

Test Case	Description
rfe_tc_3278_sm98SpiAccessMiso Rx2	Testcase to perform m7 spi access ana for rx2 ip e miso r2 fault injection test for sm98

3.339 Test Suite rfe_TS_3279

QUALIFICATION Test Suite TS_3279 - TS_3279.

Table 343. Test Sequence rfe_TS_3279

Test Case	Description
rfe_tc_3279_sm98SpiAccessMosiRx3	Testcase to perform m7 spi access ana for rx3 ip e mosi r2 fault injection test for sm98

3.340 Test Suite rfe_TS_3280

QUALIFICATION Test Suite TS_3280 - TS_3280.

Table 344. Test Sequence rfe_TS_3280

Test Case	Description
rfe_tc_3280_sm98SpiAccessMisoRx3	Testcase to perform m7 spi access ana for rx3 ip e miso r2 fault injection test for sm98

3.341 Test Suite rfe_TS_3281

QUALIFICATION Test Suite TS_3281 - TS_3281.

Table 345. Test Sequence rfe_TS_3281

Test Case	Description
rfe_tc_3281_sm98SpiAccessMosiRx4	Testcase to perform m7 spi access ana for rx4 ip e mosi r2 fault injection test for sm98

3.342 Test Suite rfe_TS_3282

QUALIFICATION Test Suite TS_3282 - TS_3282.

Table 346. Test Sequence rfe_TS_3282

Test Case	Description
rfe_tc_3282_sm98SpiAccessMisoRx4	Testcase to perform m7 spi access ana for rx4 ip e miso r2 fault injection test for sm98

3.343 Test Suite rfe_TS_3283

QUALIFICATION Test Suite TS_3283 - TS_3283.

Table 347. Test Sequence rfe_TS_3283

Test Case	Description
rfe_tc_3283_sm98SpiAccessMosiTempsensor1	Testcase to perform m7 spi access ana for tempsensor1 ip e mosi r2 fault injection test for sm98

3.344 Test Suite rfe_TS_3284

QUALIFICATION Test Suite TS_3284 - TS_3284.

Table 348. Test Sequence rfe_TS_3284

Test Case	Description
rfe_tc_3284_sm98SpiAccessMisoTempsensor1	Testcase to perform m7 spi access ana for tempsensor1 ip e miso r2 fault injection test for sm98

3.345 Test Suite rfe_TS_3285

QUALIFICATION Test Suite TS_3285 - TS_3285.

Table 349. Test Sequence rfe_TS_3285

Test Case	Description
rfe_tc_3285_sm98SpiAccessMosiTempsensor2	Testcase to perform m7 spi access ana for tempsensor2 ip e mosi r2 fault injection test for sm98

3.346 Test Suite rfe_TS_3286

QUALIFICATION Test Suite TS_3286 - TS_3286.

Table 350. Test Sequence rfe_TS_3286

Test Case	Description
rfe_tc_3286_sm98SpiAccessMisoTempsensor2	Testcase to perform m7 spi access ana for tempsensor2 ip e miso r2 fault injection test for sm98

3.347 Test Suite rfe_TS_3287

QUALIFICATION Test Suite TS_3287 - TS_3287.

Table 351. Test Sequence rfe_TS_3287

Test Case	Description
rfe_tc_3287_sm98SpiAccessMosiTempsensor3	Testcase to perform m7 spi access ana for tempsensor3 ip e miso r2 fault injection test for sm98

3.348 Test Suite rfe_TS_3288

QUALIFICATION Test Suite TS_3288 - TS_3288.

Table 352. Test Sequence rfe_TS_3288

Test Case	Description
rfe_tc_3288_sm98SpiAccessMisoTempsensor3	Testcase to perform m7 spi access ana for tempsensor3 ip e mosi r2 fault injection test for sm98

3.349 Test Suite rfe_TS_3289

QUALIFICATION Test Suite TS_3289 - TS_3289.

Table 353. Test Sequence rfe_TS_3289

Test Case	Description
rfe_tc_3289_sm98SpiAccessMosiTempsensor4	Testcase to perform m7 spi access ana for tempsensor4 ip e mosi r2 fault injection test for sm98

3.350 Test Suite rfe_TS_3290

QUALIFICATION Test Suite TS_3290 - TS_3290.

Table 354. Test Sequence rfe_TS_3290

Test Case	Description
rfe_tc_3290_sm98SpiAccessMisoTempsensor4	Testcase to perform m7 spi access ana for tempsensor4 ip e miso r2 fault injection test for sm98

3.351 Test Suite rfe_TS_3291

QUALIFICATION Test Suite TS_3291 - TS_3291.

Table 355. Test Sequence rfe_TS_3291

Test Case	Description
rfe_tc_3291_sm98SpiAccessMosiAdpll	Testcase to perform m7 spi access ana for adpll ip e mosi r2 fault injection test for sm98

3.352 Test Suite rfe_TS_3292

QUALIFICATION Test Suite TS_3292 - TS_3292.

Table 356. Test Sequence rfe_TS_3292

Test Case	Description
rfe_tc_3292_sm98SpiAccessMisoAdpll	Testcase to perform m7 spi access ana for adpll ip e miso r2 fault injection test for sm98

3.353 Test Suite rfe_TS_3293

QUALIFICATION Test Suite TS_3293 - TS_3293.

Table 357. Test Sequence rfe_TS_3293

Test Case	Description
rfe_tc_3293_sm99SpiAccessMosiSpi	Testcase to perform rfe access m7 spi access (special hook with error n) r2 fault handling for sm99

3.354 Test Suite rfe_TS_3294

QUALIFICATION Test Suite TS_3294 - TS_3294.

Table 358. Test Sequence rfe_TS_3294

Test Case	Description
rfe_tc_3294_sm99SpiAccessMisoSpi	Testcase to perform rfe access m7 spi access (special hook no error n) r2 fault handling for sm99

4 Test Cases

4.1 rfe_tc_001_validSampleCountConfiguration

The RFE Radar Use Case API shall support programming of 8-8192 samples per Chirp. Here different samples values in range are checked.

4.1.1 Detailed Description

Table 359. rfe_tc_001_validSampleCountConfiguration Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1290125

4.1.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for '2048' Samples per chirp, in a chirp profile. Parameters : Chirp Acquisition time = 51.2us or 2048 ticks , Sampling frequency = 40MHz Samples = Chirp Acquisition time(51.2) * Sampling frequency(40) = 2048 Note : us = ticks * 25 / 1000
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
6. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
7. Repeat the test for different Samples configurations count of 8, 32, 100, 250 ,256, 512, 800, 1024, 8000, 8192.

4.2 rfe_tc_002_sampleCountLessThanMinRange

The RFE Radar Use Case API shall support programming of 8-8192 samples per Chirp. Here Sample count less than Min range value is checked.

4.2.1 Detailed Description

Table 360. rfe_tc_002_sampleCountLessThanMinRange Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1290125

4.2.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for '2048' Samples per chirp, in a chirp profile. Parameters : Chirp Acquisition time = 51.2us or 2048 ticks , Sampling frequency = 40MHz Samples = Chirp Acquisition time(51.2) * Sampling frequency(40) = 2048 Note : us = ticks * 25 / 1000
3. Manually update the blob content with Samples/Acquisition ticks count value lower than 8 e.g 7
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_ CONFIGURATION_PARAMETERVALUE'

4.3 rfe_tc_003_sampleCountGreaterThanMaxRange

The RFE Radar Use Case API shall support programming of 8-8192 samples per Chirp. Here Sample count greater than max range value is checked.

4.3.1 Detailed Description

Table 361. rfe_tc_003_sampleCountGreaterThanMaxRange Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional

Table 361. rfe_tc_003_sampleCountGreaterThanMaxRange Specification*...continued*

Property	Description
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1290125

4.3.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Asset check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for '2048' Samples per chirp, in a chirp profile. Parameters : Chirp Acquisition time = 51.2us or 2048 ticks , Sampling frequency = 40MHz Samples = Chirp Acquisition time(51.2) * Sampling frequency(40) = 2048 Note : us = ticks * 25 / 1000
3. Manually update the blob content with Samples/Acquisition ticks count value greater than 8192 e.g 9000, Invalid blob to be generated.
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e API_INVALID_ CONFIGURATION_PARAMETERVALUE

4.4 rfe_tc_004_reConfigureRfeWithDifferentSamplesCount

This test case is used to programming Samples when RFE is in 'CONFIGURED' state.

4.4.1 Detailed Description

Table 362. rfe_tc_004_reConfigureRfeWithDifferentSamplesCount Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1290125

4.4.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case by programming for '2048' Samples per chirp, in a chirp profile. Parameters : Chirp Acquisition time = 51.2us or 2048 ticks , Sampling frequency = 40MHz Samples = Chirp Acquisition time(51.2) * Sampling frequency(40) = 2048 Note : us = ticks * 25 / 1000
3. 3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
   - On successful RFE configuration Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
5. 5. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
6. 6. Create a configuration with the RFE static use case by programming for '1024' Samples per chirp, in a chirp profile. Parameters : Chirp Acquisition time = 25.6us or 1024 ticks , Sampling frequency = 40MHz Samples = Chirp Acquisition time(25.6) * Sampling frequency(40) = 1024 Note : us = ticks * 25 / 1000
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
7. 7. Perform steps 4 & 5.

4.5 rfe_tc_005_validChirpCountConfiguration

This test case is used to check the RFE API shall support programming maximum of 4096 Chirps. Here a valid chirp count range check is done.

4.5.1 Detailed Description

Table 363. rfe_tc_005_validChirpCountConfiguration Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280657

4.5.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case by programming chirpCount to '4096' chirps for a chirp sequence configuration.
3. 2. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
5. 5. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
6. Repeat the test steps from 1-5 for different chirp configurations count of 1, 20, 32, 64, 128, 512, 2048.

4.6 rfe_tc_006_chirpCountZero

This test case is used to check the RFE API shall support programming maximum of 4096 Chirps.Here a chirp count '0' check is done.

4.6.1 Detailed Description

Table 364. rfe_tc_006_chirpCountZero Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280657

4.6.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.

- On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.

2. Create a configuration with the RFE static use case by programming chirpCount to '4096' chirps for a chirp sequence configuration.
3. Manually update the blob content with chirp count value to '0' or optionally use
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
5. Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e API_INVALID_ CONFIGURATION_PARAMETERVALUE

4.7 rfe_tc_007_chirpCountGreaterThanMaxRange

This test case is used to check the RFE API shall support programming maximum of 4096 Chirps.Here a count range greater than 4096 check is done.

4.7.1 Detailed Description

Table 365. rfe_tc_007_chirpCountGreaterThanMaxRange Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280657

4.7.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming chirpCount to '4096' chirps for a chirp sequence configuration.
3. Manually update the blob content with chirp count value greater than 4096 e.g 8192
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e API_INVALID_ CONFIGURATION_PARAMETERVALUE

4.8 rfe_tc_008_reConfigureRfeWithDifferentChirpCount

This test case is used to programming Chirps when RFE is in 'CONFIGURED' state.

4.8.1 Detailed Description

Table 366. rfe_tc_008_reConfigureRfeWithDifferentChirpCount Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280657

4.8.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case by programming chirpCount to '4096' chirps for a chirp sequence configuration.
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
5. 5. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
6. 6. Create a configuration with the RFE static use case by programming chirpCount to '2048' chirps for a chirp sequence configuration.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
7. 7. Perform steps 4 & 5.

4.9 rfe_tc_010_readCurrentTempBeforeAfterChirpSequence

The RFE SW shall provide an API to get the current temperatures, the temperatures before and after the last chirp sequence.

4.9.1 Detailed Description

Table 367. rfe_tc_010_readCurrentTempBeforeAfterChirpSequence Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1301497

4.9.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
6. Execute rfe_monitorRead() API. Parameters : Set rfe_monitorSelect_t with 'TEMPBEFORECHIRPSEQ' = 1 and 'TEMPAFTERCHIRPSEQ' = 1 to enable the temp sensors readout. Set empty rfe_monitorValues_t was the reply.
7. Assert to check the Error parameter of the rfe_monitorRead() API reply to be '0'. i.e No Error
8. Read the out parameter rfe_monitorValues_t to check the temperatures before and after the last chirp sequence.
   - Assert to check if the monitor values i.e. temperature values of all the sensors are non-zero and greater than zero.

4.10 rfe_tc_011_stopRadarCycle

The RFE SW shall provide functionality to stop an running radar use case, via the RFE Radar Use Case API.

4.10.1 Detailed Description

Table 368. rfe_tc_011_stopRadarCycle Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1280639

4.10.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
   2. Create a configuration with the RFE static use case. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
   4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
   5. During the Radar Cycle(s) in progress, execute rfe_radarCycleStop() API.
   - Assert to check the Error parameter of the rfe_radarCycleStop() API reply to be '0x0' or '0x4'. i.e. No Error or API_BUSY i.e RFE API Busy
   6. If the Error parameter is '0x04' i.e. API Busy then repeat step '5' until you have rfe_radarCycleStop() API Error Parameter to be '0x0'. 7. Execute rfe_getRadarCycleCount() API to check the number of executed cycles.
   - Assert to check the Error parameter of the rfe_getRadarCycleCount() API reply to be '0x0'. i.e No Error
   - Assert to check the 'radarCycleCount' to be less than configured 'RFE_CYCLE_COUNT' cycles.
   - Assert to check the 'chirpSequenceCount' to be less than the total configured chirp sequences of 'RFE_CYCLE_COUNT' Cycles.

4.11 rfe_tc_012

This test case is used to check the RFE restrict access to the defined API groups and to individual API calls depending on its state.

4.11.1 Detailed Description

Table 369. rfe_tc_012 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280283, 1290108, 1381784

4.11.2 Test Procedure

Steps:

1. Initialize the RFE SW(rfe_sync)
2. Check if the status is in the Initialized status, if yes:
3. Check all of the abstract API calls that don't support the initialized status, e,p. rfe_radarCycleStart()...
4. error is expected after the API calls
5. Check the status shall remain in the same status
6. Then do rfe_Config() with valid parameters, check if it is in configured status, if yes:
7. Check all of the abstract API calls that don't support the configured status, e,p. rfe_getNextRadarCycleStartTime()
8. error is expected after the API calls
9. Check the status shall remain in the same status
10. Clean PPE buffer
11. for 1290108 read PPE buffer and check that still they are still zero
12. Then do rfe_startRadarCycle() with valid parameters for single radar cycle, check if it is in busy status, if yes:
13. Check all of the abstract API calls that don't support the busy status, e,p. rfe_getFuSaFaults()
14. error is expected after the API calls
15. Check the status shall remain in the same status
16. Check FUSA faults, there shall be no FUSA faults
17. Check that current radar cycle is not interrupted. Check that ADC data is non-zero, ADC shall provide valid data, no zero padding(set PPE to store the data from address 0x33E80000), (do when possible: max ADC absolute value is more than 2000, Amplitude after FFT is more than -20dBm, clear target detection)
18. Repeat above tests also for the 'radarCycleIdle', 'testContinuousWaveTransmission'

4.12 rfe_tc_013

This test case is used to check the RFE SW shall support definition of multiple radar use cases.

4.12.1 Detailed Description

Table 370. rfe_tc_013 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280375, 1280639, 1280173, 1280646, 1290110

4.12.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Check RFE calibrate, frame and BIST configuration with valid parameters.
3. Check configuring RFE multiple radar use cases using RFE Radar use case api.
4. Check RFE IP block Re-calibration app specific
5. Check RFE frame start
6. Check stopping the radar frame acquisition by using RFE Radar use case api and select a different use case and repeat this at certain intervals.
7. Raw data is transferred to binary files or streamed to local host on PC and either visualized real-time or analysed offline with default data visualizer.

4.13 rfe_tc_014

This test case is used to check the RFE API shall support programming maximum of 4096 Chirps

4.13.1 Detailed Description

Table 371. rfe_tc_014 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox

Table 371. rfe_tc_014 Specification*...continued*

Property	Description
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1290125, 1280657

4.13.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Check RFE calibrate, frame and BIST configuration with valid parameters.
3. Check configuring RFE static use cases by programming chirpCount to 4096 chirps per radar frame and verify the response code.
4. Vary through all combinations of acquisition-time-ticks vs. effectiveSamplingFrequency to check programming of 8-8192 samples per chirp and verify the response code.
5. Verify at APP-M7 either the number of interrupts or the chirpSequenceCount within rfe_monitorRead() to check the number of chirps programmed correctly. Check the buffer size for verifying samples/chirp.

4.14 rfe_tc_015_configureTxBistPower

The RFE Abstract API shall provide a configuration parameter to set the Tx BIST output power.

4.14.1 Detailed Description

Table 372. rfe_tc_015_configureTxBistPower Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1376566

4.14.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.

- On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.

2. 2. Create a configuration with the RFE static use case with below parameters. Note: TxBist is enabled and performed only when one of the Tx's should be "Enabled and transmission should be 'ON' , also either of the below fusa faults are un-masked . i.e TX1-TX2 phase diff error,TX2-TX3 phase diff error,TX3-TX4 phase diff error,TX1 phase step error,TX2 phase step error, TX3 phase step error & TX4 phase step error Parameters : e.g. txEnableElement Tx-1 is 'Enabled' a for given chirp sequence. txTransmissionEnableElement Tx-1 is 'Enabled' for a given chirp profile. 'txPowerLevelForBist' with value between -90 to 150 i.e. -9 to 15 dBm under the monitor configuration. e.g. 10dBm. Faults 'TX1-TX2 phase diff error', 'TX1 phase step error' should be unmasked
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'
4. 4. Execute rfe_getBistZeroHourReferenceData() API
   - Assert to check the Error parameter of the rfe_getBistZeroHourReferenceData() API reply to be '0x0'. i.e No Error
   - Assert to check the 'txPowerLevelForBist' value to be '10dBm' from rfe_txBistZeroHourRefData.
5. 5. Repeat the steps 2 to 4 with different values of 'txPowerLevelForBist' e.g , -9, -5, 0, 3, 12 & 15 dBm.
6. 6. Repeat the steps 2 to 4 with unmasking the each of the 7 faults one by one with the either Tx's "Enabled and transmission should be 'ON'. i.e TX1-TX2 phase diff error,TX2-TX3 phase diff error,TX3-TX4 phase diff error,TX1 phase step error,TX2 phase step error,TX3 phase step error & TX4 phase step error.
7. 7. Repeat the steps 2 to 4 with unmasking the each of the 7 faults one by one with all the Tx's "Disabled and transmission should be 'OFF'. i.e TX1-TX2 phase diff error,TX2-TX3 phase diff error,TX3-TX4 phase diff error,TX1 phase step error,TX2 phase step error,TX3 phase step error & TX4 phase step error.

4.15 rfe_tc_016

This test case is to check the RFE SW shall provide setting of FTTI time for RFE SW, via RFE Safety API.

4.15.1 Detailed Description

Table 373. rfe_tc_016 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299512

4.15.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Check setting the FTTI for RFE SW using Safety API with invalid value and verify the response error code.
3. Check setting the FTTI for RFE SW using Safety API with valid value.
4. Verify self-test for every FTTI durations.

4.16 rfe_tc_016_runLRRadarUc

The RFE SW shall provide functionality to run a configured radar use case, via the RFE Radar Use Case API.

4.16.1 Detailed Description

LR Radar Usecase.

Table 374. rfe_tc_016_runLRRadarUc Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1381784

4.16.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case and update it with below parameters for LR (Long Range Usecase). Parameters : ChirpCount=256, Center Frequency=76.5GHz, Sampling Frequency=40MHz, ChirpBandwidth=380MHz, Acquisition time=51.2us, Chirp Time=55.4us (Rest timing parameters can be default or sum to chirp time), Samples=2048 and atleast one Tx enabled.
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
5. 5. Check for any FUSA faults using rfe_getFuSaFaults() API.
   - Assert to check the Error parameter of the rfe_getFuSaFaults() API reply to be '0'. i.e No Error
   - Assert to check the pFuSaR1R2FaultList size to be '0'.

6. 6. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.

4.17 rfe_tc_017

This test case is to check the RFE SW shall provide configuration of TCM recoverable errors threshold level, via RFE Safety API

4.17.1 Detailed Description

Table 375. rfe_tc_017 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305755

4.17.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Check using Safety API the configuration of TCM recoverable error threshold with an invalid level and verify the response error code.
3. Check setting the configuration of TCM recoverable error for RFE SW using Safety API with valid value.
4. Verify by injecting faults and check the behaviour.

4.18 rfe_tc_017_runMRRadarUc

The RFE SW shall provide functionality to run a configured radar use case, via the RFE Radar Use Case API.

4.18.1 Detailed Description

MR Radar Usecase.

Table 376. rfe_tc_017_runMRRadarUc Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.

Table 376. rfe_tc_017_runMRRadarUc Specification*...continued*

Property	Description
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1381784

4.18.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case and update it below parameters for MR (Mid Range Usecase). Parameters : ChirpCount=58, Center Frequency=76.5GHz, Sampling Frequency=40MHz, ChirpBandwidth=800MHz, Acquisition time=25.6us, Chirp Time=33.95us (Rest timing parameters can be default or sum to chirp time), Samples=1024 and atleast one Tx enabled.
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
5. 5. Check for any FUSA faults using rfe_getFuSaFaults() API.
   - Assert to check the Error parameter of the rfe_getFuSaFaults() API reply to be '0'. i.e No Error
   - Assert to check the pFuSaR1R2FaultList size to be '0'.
6. 6. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.

4.19 rfe_tc_018_runDDMARadarUc

The RFE SW shall provide functionality to run a configured radar use case, via the RFE Radar Use Case API.

4.19.1 Detailed Description

DDMA Radar Usecase.

Table 377. rfe_tc_018_runDDMARadarUc Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.

Table 377. rfe_tc_018_runDDMARadarUc Specification*...continued*

Property	Description
	RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1381784

4.19.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the default RFE static use case and update it below parameters for DDMA usecase. Parameters : Chirp Interval Time=38.175, Dwell Time=0.150, Settle Time=2.300, Acquisition Time=25.6, Reset Time=5.7.(All in us) Center Frequency=76.5GHz, Sampling Frequency=40MHz, ChirpBandwidth=500MHz, Pll loop filter bandwidth=250MHz, Samples=1024, Chirps=58, csi2VirtualChannel:channel=0, all Tx's enabled and transmission enabled. Create a chirp profile '0' with above parameters with txPhaseRotation tx-1="0" tx-2="0" tx-3="0" tx-4="0". Create a chirp profile '1' with above parameters with txPhaseRotation tx-1="180" tx-2="270" tx-3="0" tx-4="0". Create a chirp profile '2' with above parameters with txPhaseRotation tx-1="0" tx-2="180" tx-3="0" tx-4="180". Create a chirp profile '3' with above parameters with txPhaseRotation tx-1="180" tx-2="90" tx-3="0" tx-4="270". Create a chirpProfileSequence config '0' with profile sequence="0 1 2 3"
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
5. 5. Check for any FUSA faults using rfe_getFuSaFaults() API.
   - Assert to check the Error parameter of the rfe_getFuSaFaults() API reply to be '0'. i.e No Error
   - Assert to check the pFuSaR1R2FaultList size to be '0'.
6. 6. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
   Optionally perform 1st and 2nd FFT processing to visualize the data in Charts.

4.20 rfe_tc_019_configureTxBistPowerWithFaultsMasked

The RFE Abstract API shall provide a configuration parameter to set the Tx BIST output power.

4.20.1 Detailed Description

Test to check when fault unmasked doesn't correspond to Tx enabled and transmitting.

Table 378. rfe_tc_019_configureTxBistPowerWithFaultsMasked Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1376566

4.20.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case with below parameters. Note: TxBist is enabled and performed only when one of the Tx's should be "Enabled and transmission should be 'ON' , also either of the below fusa faults are un-masked. i.e TX1-TX2 phase diff error,TX2-TX3 phase diff error,TX3-TX4 phase diff error,TX1 phase step error,TX2 phase step error, TX3 phase step error & TX4 phase step error Parameters : e.g. txEnableElement Tx-1 is 'Enabled' a for given chirp sequence. txTransmissionEnableElement Tx-1 is 'Enabled' for a given chirp profile. 'txPowerLevelForBist' with value between -90 to 150 i.e. -9 to 15 dBm under the monitor configuration. e.g. 12.5dBm. Mask all the above 7 Faults
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'
4. 4. Execute rfe_getBistZeroHourReferenceData() API
   - Assert to check the Error parameter of the rfe_getBistZeroHourReferenceData() API reply to be '0x0'. i.e No Error
   - Assert to check the 'txPowerLevelForBist' value to be '0dBm' from rfe_txBistZeroHourRefData.
5. 5. Repeat the steps 2 to 4 with different Tx's enabled and Transmission 'ON' i.e Tx2,Tx3,T4.
6. 6. Repeat the steps 2 to 4 with different Tx's disabled and Transmission 'OFF' i.e Tx2,Tx3,T4.

4.21 rfe_tc_020_rfeAutonomousMode

The RFE SW shall support autonomous radar system cycles on RFE-M7.

4.21.1 Detailed Description

Table 379. rfe_tc_020_rfeAutonomousMode Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to execute the commands.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1301085

4.21.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case.
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration.
   Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0 Note : This configures RFE in autonomous mode
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
5. 5. Keep it for a duration approximately 10 mins, duration can be long but not recommended for the daily tests. This is typically a long duration test.
6. 6. Capture the Raw ADC data , based on the Samples and chirps configured for every cycle executed.
7. 6. During the Radar Cycle in progress, execute rfe_radarCycleStop() API.
   - Assert to check the Error parameter of the rfe_radarCycleStop() API reply to be '0x0' or '0x4'. i.e. No Error or API_BUSY i.e RFE API Busy
8. 7. If the Error parameter is '0x04' i.e. API Busy then repeat step '5' until you have rfe_radarCycleStop() API Error Parameter to be '0x0'.
9. 8. Execute rfe_getRadarCycleCount() API to check the number of executed cycles.
   - Assert to check the Error parameter of the rfe_getRadarCycleCount() API reply to be '0x0'. i.e No Error
   - Assert to check the 'radarCycleCount' to be greater than '0' cycles.
   Here assertion should also be on number of cycles that could execute with the above configuration. e.g With Samples=1024, Chirps=58, Dwell time=.150, Settle time=2.30,Reset time=5.7 ,Reset time=5.7, Jumpback time=0.2, ChirpInterval time =38.175(All times in us), Fs=40MHz,Cf=76.5GHz we could get around 450 cycles in 40 seconds time.

10. 9. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=1, is_scheduled=False, start_time=0
11. 10. Execute rfe_getRadarCycleCount() API to check the number of executed cycles.
    - Assert to check the Error parameter of the rfe_getRadarCycleCount() API reply to be '0x0'. i.e No Error
    - Assert to check the 'radarCycleCount' to be equal to 1 cycle.

4.22 rfe_tc_021_checkLiPvariant

Test LiP sample shall not give BBD faults and CSI2 shall work

4.22.1 Detailed Description

Test the STRX LIP variant, the BBD must be disabled, no BBD (TX and RX) faults are reported, the CSI2 must have pin swap.

Table 380. rfe_tc_021_checkLiPvariant Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. This test is only for the Launcher in Package(LiP) Variant
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	Qualification
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	-# No API errors, no FUSA errors, no BBD FUSA errors. -# SRAM contains enough non-zero data as programmed.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board; both board and sample shall be LiP variant.
Execution	Automated
Satisfied Requirements	3221753

4.22.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) and check the sync status with rfe_sync() API.
2. Check FCCU errors with rfe_getFuSaFaults(), no error shall happen, BBD errors are part of the FCCU errors.
3. Check Register in CSI2 module from host core: RFE_HW_CSI2PHY_CFG_MIXEL_PNCH_REG, the 1st lsb shall be opposite to the non-LiP Variant. Need to measure the non-LiP Variant once.
4. Configure RFE with single sequence and single profile use case and enable CSI2 output. Use 1024 samples number and 128 chirps number.
5. Configure Host core to store the Data from CSI2 to SRAM
6. Clean the data in target SRAM area, if fresh start from the power cycle, this step can be skiped.
7. Trigger single radar cycle, and check the RFE status and FUSA errors
8. Check the data in SRAM, the programmed SRAM area shall contain 1024*128*4 non-zero data.

4.23 rfe_tc_022_configureMultipleChirpSequence

The RFE SW shall support defining multiple radar chirp sequences, via RFE Abstract 2.0 API.

4.23.1 Detailed Description

Multiple chirp sequences with different chirp profile.

Table 381. rfe_tc_022_configureMultipleChirpSequence Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. Connect the board with FSW with appropriate connections.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board, FSW.
Execution	Automated
Satisfied Requirements	1280346

4.23.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
   2. Create a configuration with the RFE static use case. Update with below parameters: Create 4 chirp profile configurations i.e, Profile0,Profile1,Profile2 & Profile3. Set CenterFrequency as Profile0 =76.3GHz,Profile1 =76.5GHz,Profile2 =76.7GHz,Profile3 =76.45GHz Create 4 corresponding chirp sequences for each profile and set the following <chirpProfileSequence sequence="0" />, <chirpProfileSequence sequence="1" />, <chirpProfileSequence sequence="2" /> & <chirpProfileSequence sequence="3" /> Use 58 chirps and 1024 samples, for all chirpSequences.
   - Set all Tx's ON with Transmission enabled. Use Virtual Channel = 0
   set the chirpSequences as chirp-sequence-config-index="0 1 2 3". set the start-time-offset between the chirp sequence as 450000 682000 914000 1146000 ticks. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
   4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_ RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
   5. Plot the FM Time domain data to visualize the configured center frequency, chirp interval times and chirp sequence offset time. Assert the if all the profile have the center frequency as configured. Assert the chirp interval times for all the 4 chirp sequences. Assert the start time offset for all the chirp sequences. Attached are image for references in RFE_1280346.docx. Test_rfe\integration\specific\STRX\doc\reference

4.24 rfe_tc_024_rxAdcClipThresholds

This test case is used to check the RFE SW shall provide programming of Rx ADC clipping thresholds, clipping counts via RFE Safety API

4.24.1 Detailed Description

Table 382. rfe_tc_024_rxAdcClipThresholds Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311224

4.24.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming limit-adc1="4194303", in a single chirp profile. Parameters : limit-adc1="4194303"
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Use test spi read, to check the programmed limit is distributed in PDC module using register PDC_ADC_OL_CNT1 and bitfield ADC_OL_CNT_THRSH.
6. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
7. Use test spi read, to check the programmed limit is distributed in PDC module using register PDC_ADC_OL_CNT1 and bitfield ADC_OL_CNT_THRSH.
8. Repeat the test for different limit of 0, 100, 1000, 10000. Check if error triggered during radar cycle for lower limit.
9. Repeat the test for different adcs. Eg., adc2, adc3,adc4.
10. Create a configuration with the RFE static use case by programming limit-adc1="4194304", in a single chirp profile by using rfeCfg api. Parameters : limit-adc1="4194304"
11. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.
    - Assert to check the Error parameter of the rfe_configure() API reply to be '7'. i.e invalid configuration
12. On invalid RFE configuration check for the current RFE state to be 'INTIALIAZED'.

4.25 rfe_tc_025

This test case is used to check the RFE SW shall allow testing of GPIO pins, via RFE Safety API

4.25.1 Detailed Description

Table 383. rfe_tc_025 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305771

4.25.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Check configuring GPIO pins with invalid value using safety API and verify the error response code.
3. Check configuring GPIO pins with valid value using safety API and verify the error response code.

4.26 rfe_tc_026_validRxEnable

The RFE Radar Use Case API shall support programming configuration parameters to configure 4 RX modules. Here different rx enable is checked.

4.26.1 Detailed Description

The configuration parameters such as rx gain and rx baseband filter corner frequencies are tested through requirements 1280787 and 1296344 respectively.

Table 384. rfe_tc_026_validRxEnable Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to collect the adc data.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox

Table 384. rfe_tc_026_validRxEnable Specification*...continued*

Property	Description
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280203, 1290129, 1280309

4.26.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming by enabling rxEnable, in a single chirp profile. Parameters : rxEnable = true for rx-1 and false for other rx's and atleast one tx is enabled.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
6. Collect the adc data
   - Assert that ADC data is non-zero. And assert the data received is number of chirps * number of samples * 2 bytes.
7. Repeat the test for different rx enable configurations. Eg. rx-2 true others false, rx-3 true others false, mulitple rx enabled etc.
8. Repeat the test for no rx enable configurations. All rxEnable should be false.
   - Assert that ADC data received is Zero. (Zero padding)

4.27 rfe_tc_027_checkCSI2HeaderFooter

Test switching header and footer on/off for CSI2 packet (chirp) via the RFE DFE API.

4.27.1 Detailed Description

The RFE SW shall enable switching header and footer on/off for CSI2 packet (chirp) via the RFE DFE API. Need to check both the Header and Footer enable and disable cases. Due to the Header and Footer data are same for both Packet Processor(PPE) and CSI2, and using PPE is easier to get data in SRAM, therefore we test via PPE instead of CSI2.

Table 385. rfe_tc_027_checkCSI2HeaderFooter Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. Due to validate data from CSI2 is difficult, testing from PPE and CSI2 can get same header and footer settings,
Post	RFE FW should be in non-error state and should be able to perform execute new tests.

Table 385. rfe_tc_027_checkCSI2HeaderFooter Specification*...continued*

Property	Description
Test Level	Qualification
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	-# No API errors, no FUSA errors in every step -# When Header and Footer are enabled, the 16bytes Header and Footer are none-zero; When Header and Footer are disabled, the 16bytes data are zeros.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280702

4.27.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) and check the sync status with rfe_sync() API.
2. Check FCCU errors with rfe_getFuSaFaults().
3. Configure RFE with single sequence and single profile use case and enable Packet Processor output. Use 1024 samples number and 64 chirps number.
4. step a: In RFE Configuration, the data Header and Footer option shall be enabled
5. Enable the Header and Footer storage from PPE via Control Core, Register Name: RX_STAT_CONFIG, Offset in PPE module: FCh, write this register to value 0x6
6. Configure Host core to store the Data from Packet Processor to SRAM
7. step b: Trigger single radar cycle, and check the RFE status and FUSA errors
8. step c: Check the data in SRAM, the programmed SRAM area shall contain 1024*64*4*2 + 16*64 bytes data. Header and Footer data are stored interleaved between ADC data, they are after each chirps' ADC data.
9. step d: Check the data in SRAM, every chirp has ADC data with 1024*4*2 Bytes, the Header and Footer info is right after this ADC raw data, and length is 16 bytes. Check Header and Footer data are non-zeros.
10. step e: Clean the Data in SRAM to be zeros
11. step f: use rfeConfig() to load an almost same parameters, but with disabled header and footer.
12. step g: Trigger new radar cycle, the the programmed SRAM area shall contain 1024*64*4*2 + 16*64 bytes data.
13. step h: Check the data in SRAM, every chirp has ADC data with 1024*4*2 Bytes, the Header and Footer info is right after this ADC raw data, and length is 16 bytes. Check Header and Footer data are zeros.
14. step l: Clean the Data in SRAM to be zeros
15. repeat step a,b,c,d,e,f,g,h,l, and use parameter update abstract API to enable/disable the Header and Footer in step a and f.

4.28 rfe_tc_028_runMultipleRadarCycles

The RFE SW shall execute radar system cycles after the run command called via RFE Radar System Cycle API.

4.28.1 Detailed Description

Table 386. rfe_tc_028_runMultipleRadarCycles Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. QTA or RFE Proxy can be used to test.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1290108

4.28.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case.
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
5. 5. Execute rfe_getRadarCycleCount() API to check the number of executed cycles.
   - Assert to check the Error parameter of the rfe_getRadarCycleCount() API reply to be '0x0'. i.e No Error
   - Assert to check the 'radarCycleCount' to be equal to RFETEST_RADAR_CYCLE_COUNT cycles.
6. 6. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=1, is_scheduled=False, start_time=0
7. 7. Execute rfe_getRadarCycleCount() API to check the number of executed cycles.
   - Assert to check the Error parameter of the rfe_getRadarCycleCount() API reply to be '0x0'. i.e No Error
   - Assert to check the 'radarCycleCount' to be equal to 1 cycle.

4.29 rfe_tc_029_multimodeDynamicChirpConfig

The RFE SW shall dynamically program (in-between chirps) the dynamic chirp parameters into the RFE IPs.

4.29.1 Detailed Description

Dynamic parameters configuration in Multi-mode configuration/profile.

Table 387. rfe_tc_029_multimodeDynamicChirpConfig Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311315

4.29.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
   2. Create a configuration with the RFE static use case. Parameters: Create 8 chirp profiles (0-8) configurations with other default chirp configuration Use 128 chirps and 512 samples, with single chirpSequences (1 frame) including the all the 8 profiles. Enable dynamicUpdate dynamic-updates=enabled Create a dynamic table entry for all the 128 chirps, with same parameter configurations as defined in above profile ie. phase,tx tranmission, Chirpinterval time, dwell time and frequency. E.g Set static chirp timing parameters,profile,frequency as in profile configuration. Set first 16 chirps with all TX transmission enabled, next 16 chirps TX transmission disabled, alternate settings till 128 chirps. i.e 16 * 8(profile) = 128 chirps 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and dynamic contents with length.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
   4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
2. 5. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
3. 6. Perform Range doppler post processing to visualize the available at different doppler's/velocity, here each mode/profile data should be checked i.e. 16*512 for each profile. If the Tx was enabled in mode then target is seen or if Tx was disabled then no target should be seen.

4.30 rfe_tc_030

This test case is used to check the RFE SW while in dynamic chirp configuration mode, the RFE SW shall support chirp sequence lengths of 1 to 4096 with dynamic chirp parameters.

4.30.1 Detailed Description

Table 388. rfe_tc_030 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311317

4.30.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Start up Calibrations (RFE-M7) app independent.
3. Check RFE frame configuration with static and dynamic chirp valid parameters
4. Check RFE frame start
5. Raw data is transferred to binary files or streamed to local host on PC and either visualized real-time or analysed offline with default data visualizer.

4.31 rfe_tc_031

This test case is used to check the RFE SW while in dynamic chirp configuration mode, the RFE SW shall support configuration of 1 to 4 sets of static chirp parameters (profiles) for the next system cycle, via the RFE System API

4.31.1 Detailed Description

Table 389. rfe_tc_031 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311318

4.31.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Start up Calibrations (RFE-M7) app independent.
3. Check RFE frame configuration static chirp parameter of each set of profile with invalid parameters and verify the error response code.
4. Check RFE frame configuration static chirp parameter of each set of profile with valid parameters.
5. Implement 4 profiles with completely different chirpTiming in it. Use dynamicTableEntry to vary through all parameters (phase-rotation-tx,...)
6. Check RFE frame start
7. Change the profile parameter for every frame and verify

4.32 rfe_tc_032

This test case is used to check the RFE SW while in dynamic chirp configuration mode, the RFE SW shall provide configuration of the dynamic chirp parameters for the complete chirp sequence of the next radar system cycle, via the RFE System API.

4.32.1 Detailed Description

Table 390. rfe_tc_032 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280368, 1290112

4.32.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Start up Calibrations (RFE-M7) app independent.
3. Check RFE frame configuration static chirp parameter of each set of profile with valid parameters.
4. Use rfe_configure() and read back memory from rfe_sysMemAddress. Check if read back data corresponds to set one.
5. rfe_radarCycleStart() with radarCycleCount = 0 (infinite)
6. Check using RFE SW System API for frame configuration of the dynamic chirp parameters for the complete chirp sequence of the next radar system cycle
7. Use rfe_updateDynamicTable() in any allowed state and check the response code.
8. Use rfe_updatePush() in any allowed state and check the response code.

9. Read back memory for dynamic tabele and check if read back data corresponds to set one.

4.33 rfe_tc_033

This test case is used to check the RFE SW shall support the following dynamic chirp parameters: Phase rotator phase shift for each Tx Fast switch for each Tx Chirp interval time ID of set of static chirp parameters (profile) to be used.

4.33.1 Detailed Description

Table 391. rfe_tc_033 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311320

4.33.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Start up Calibrations (RFE-M7) app independent
3. Check RFE frame configuration static chirp parameter of each set of profile with valid parameters.
4. Check RFE frame start
5. Check using RFE SW System API for frame dynamic chirp parameters: Phase rotator phase shift for each Tx Fast switch for each Tx Chirp interval time ID of set of static chirp parameters (profile) to be used.

4.34 rfe_tc_034_configureChirpProfileSequence

The RFE Abstract API shall provide configuration parameter to set the chirp profile sequencing for each chirp sequence.

4.34.1 Detailed Description

Sequence the order of the profile.

Table 392. rfe_tc_034_configureChirpProfileSequence Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	

Table 392. rfe_tc_034_configureChirpProfileSequence Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1375542

4.34.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
   2. Create a configuration with the RFE static use case. Parameters: Create 8 chirp profiles (0-8) configurations with other default chirp configuration Use 128 chirps and 512 samples, with single chirpSequences (1 frame) including the all the 8 profiles. i.e 16 * 8(profile) = 128 chirps Set chirpProfileSequence sequence="0 1 2 3 4 5 6 7". Set the Tx PPD(Peak power detector) in dBm for each . profile0-dBm="1.0" profile1-dBm="3.0" profile2-dBm="5.0" profile3-dBm="7.0" profile4-dBm="9.0" profile5-dBm="11.0" profile6-dBm="13.0" profile7-dBm="14.0". Set Tx power of the profiles P0=1.5, P1=3.5,P2=5.5,P3=7.5,P4=9.5,P5=11.5,P6=13.5,P7=15. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
   4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
2. 5. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
3. 6. Perform Average FFT Range post processing to visualize the averaged amplitude of different profiles. i.e. Data size of 16*512 for each profile. If the Tx power is low the averaged amplitude of peak will be low, here with above test average amplitude increases from P0-P7. Attached are images for references in RFE_TC_30_configureChirpProfileSequence.docx. Test_rfe\integration\specific\STRX\doc\reference Relative target power change is P1-P0=2 dB,P2-P1=2dB,P3-P2=2dB,P4-P3=2dB,P5-P4=2dB,P6-P5=2dB,P7-P6=1.5dB
   - # 7. Repeat the steps 1-6 by with chirpProfileSequence sequence="7 6 5 4 3 2 1 0".

4.35 rfe_tc_035_abstractApiBackwardCompatibility

The RFE SW shall expose RFE Abstract compatible API.

4.35.1 Detailed Description

Test for API backward compatibility. Backwards compatibility works from RFE FW 0.8.18 Config related API are tested.

Table 393. rfe_tc_035_abstractApiBackwardCompatibility Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. QTA OR RFE Proxy can be used for this test.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1305665

4.35.2 Test Procedure

Steps:

1. Pre-requisite- load the know version of RFE under Test e.g RFE FW 0.8.18 Note: Backward compatibility works from RFE v0.8.18
2. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
3. 2. Invoke rfe_getVersion() API
   - Assert to check the version to correct depending on the RFE FW version loaded.
   - Assert the hash value.
4. 3. Create a configuration with the RFE static use case with dynamic tables.
5. 4. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and Dynamic table data with length.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
6. 5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_ RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
7. 6. Test all other Abstract API's defined in SAF85xx_RFE_SW_Reference_Manual.
8. 7. Repeat the steps from 1-6 with previous version RFE config blob etc. i.e RFE FW 0.8.17 (0.8.16 and earlier will not work).

4.36 rfe_tc_036_singleProfileChirpSequence

The RFE SW shall support programming of single and interleaved progressive chirp sequences, via RFE Abstract 2.0 API.

4.36.1 Detailed Description

Single profile Chirp Sequence

Table 394. rfe_tc_036_singleProfileChirpSequence Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy is required, SpectrumAnalyzer/FSW to analyze the FM time domain data.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1375547

4.36.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case. Parameters: Create 1 chirp profiles '0' configurations with other default chirp configuration Use 128 chirps and 512 samples, with single chirpSequences (1 frame). Set chirpProfileSequence sequence="0". Set the center frequency for chirp profiles 76500000
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_ RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
5. 5. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
6. 6. Perform Average Amplitude Distance plot to check Ceiling test (~2m). These test can be performed in lab conditions.
7. 7. Plot the FM Time domain data to visualize the configured center frequency Assert the if the profile have the center frequency as configured. Attached are image for references in RFE_1280346.docx. Test_rfe \integration\specific\STRX\doc\reference

4.37 rfe_tc_037_interleavedChirpSequence

The RFE SW shall support programming of single and interleaved progressive chirp sequences, via RFE Abstract 2.0 API.

4.37.1 Detailed Description

Interleaved multi-mode(profile) Chirp Sequence.

Table 395. rfe_tc_037_interleavedChirpSequence Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy is required, SpectrumAnalyzer/FSW to analyze the FM time domain data.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board
Execution	Automated
Satisfied Requirements	1375547

4.37.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case. Parameters: Create 8 chirp profiles (0-7) configurations with other default chirp configuration Use 128 chirps and 512 samples, with single chirpSequences (1 frame) including the all the 8 profiles. i.e 16 * 8(profile) = 128 chirps Set chirpProfileSequence sequence="0 1 2 3 4 5 6 7". Set the different center frequency for chirp profiles 76500000,76800000,76900000,77000000,77250000,77500000,77800000 & 78000000.
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
5. 5. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
6. 6. Perform Average Amplitude Distance plot to check Ceiling test (~2m) for all the modes/profiles. These test can be performed in lab conditions.
7. 7. Plot the FM Time domain data to visualize the configured center frequency. Assert the if all the profile have the center frequency as configured. Attached are image for references in RFE_1280346.docx. Test_rfe \integration\specific\STRX\doc\reference

4.38 rfe_tc_040_configuremultipleChirpProfile

The RFE SW shall provide programming of multiple active chirp profiles with user defined parameters, via the RFE Radar Use Case API.

4.38.1 Detailed Description

Multiple chirp profile.

Table 396. rfe_tc_040_configuremultipleChirpProfile Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. Packet Processor setting to be enabled to support multiple sequence data saving on shared memory. RFE Proxy and matlab processing is required.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280607

4.38.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case. Parameters: Create 8 chirp profiles (0-7) configurations with other default chirp configuration Use 64 chirps and 256 samples, with single chirpSequences (1 frame) including the all the 8 profiles. i.e 8 * 8(profile) = 64 chirps Set chirpProfileSequence sequence="0 1 2 3 4 5 6 7". Enable Tx ON and transmission enabled for Profile 0. Enable Tx OFF and transmission disabled for Profile 1. Enable Tx ON and transmission enabled for Profile 2. Enable Tx OFF and transmission disabled for Profile 3. Enable Tx ON and transmission enabled for Profile 4. Enable Tx OFF and transmission disabled for Profile 5. Enable Tx ON and transmission enabled for Profile 6. Enable Tx OFF and transmission disabled for Profile 7.
3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
5. 5. Check for any FUSA faults using rfe_getFuSaFaults() API.
   - Assert to check the Error parameter of the rfe_getFuSaFaults() API reply to be '0'. i.e No Error
   - Assert to check the pFuSaR1R2FaultList size to be '0'.
6. 6. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.

7. 7. Perform Average FFT Range post processing to visualize the averaged amplitude of different profiles. i.e. Data size of 8*256 for each profile. If the Tx power is low the averaged amplitude of peak will be low, here with above test average amplitude increases from P0-P7. Attached are images for references in RFE_TC_30_configureChirpProfileSequence.docx. Test_rfe\integration\specific\STRX\doc\reference Relative target power change is P1-P0=2 dB,P2-P1=2dB,P3-P2=2dB,P4-P3=2dB,P5-P4=2dB,P6-P5=2dB,P7-P6=1.5dB
8. 8. Repeat the steps 1-7 by with chirpProfileSequence sequence="7 6 5 4 3 2 1 0",sequence="3 6 5 0 3 2 1 4",sequence="4 7 3 6 0 2 1 5"

4.39 rfe_tc_041_configuremultipleChirpSequence

The RFE SW shall provide programming of multiple active chirp profiles with user defined parameters, via the RFE Radar Use Case API.

4.39.1 Detailed Description

Multiple chirp profile with different chirp sequence.

Table 397. rfe_tc_041_configuremultipleChirpSequence Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. Packet Processor setting to be enabled to support multiple sequence data saving on shared memory. RFE Proxy and matlab processing is required.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280607

4.39.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. 2. Create a configuration with the RFE static use case. Parameters: Create 1st chirp profiles configurations with other default chirp configuration - Sequence '0' Use 58 chirps and 1024 samples, with single chirpSequence (1st frame).
   - Set only one Tx ON with Transmission enabled. Use Virtual Channel = 0
   Create a 2nd chirp profile configuration with other default chirp configurations' - Sequence '1' Use 58 chirps and 1024 samples, with second chirpSequence (2nd frame).
   - Set all the Tx's ON and Transmission enabled. Use Virtual Channel = 0
   set the chirpSequences chirp-sequence-config-index="0 1" set the start-time-offset between the chirp sequence as 800000 ticks

3. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
4. 4. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
5. 5. Check for any FUSA faults using rfe_getFuSaFaults() API.
   - Assert to check the Error parameter of the rfe_getFuSaFaults() API reply to be '0'. i.e No Error
   - Assert to check the pFuSaR1R2FaultList size to be '0'.
6. 6. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
7. 7. Perform Average FFT Range post processing to visualize the averaged amplitude for both the profiles. i.e. Data size of 58*1024 for each profile/mode. Attached are images for references in RFE_1280607.docx. Test_rfe\integration\specific\STRX\doc\reference a. Plots with Tx ON & enabled b. Tx OFF & disabled.
8. 8. Repeat the steps 1-7 by with chirpProfileSequence sequence="1 0".

4.40 rfe_tc_044_autonomousRfeInitialization

The RFE SW shall support running initialization of RFE autonomously and independently from Host processor.

4.40.1 Detailed Description

Table 398. rfe_tc_044_autonomousRfeInitialization Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and needs to be powered (fresh) on with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to sync and perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1290113

4.40.2 Test Procedure

Steps:

1. 1. Power up the board, Read the register 0x33fffff4.ie. CLK_PLL_STATUS
   - Assert to check the value of register 0xC0DE0001.
   If assert is passed then RFE is autonomously initialised independent of host processor. 2. Invoke Radar Front End(RFE) using rfe_sync() API.
   - Assert to check the Error parameter of the rfe_sync() API reply to be '0'. i.e No Error

- On successful RFE initialization/sync, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.

4.41 rfe_tc_045

This test is used to check the RPC. This test runs on APP-M7

4.41.1 Detailed Description

Table 399. rfe_tc_045 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1300394

4.41.2 Test Procedure

Steps:

1. Check RFE abstract API with valid parameters and check the return status.
2. Reduce scope to RPCs that are communicating with RFE M7. (e.g. rfe_getVersion(),rfe_configure(),...).
3. Check that RFE abstract command is not coming back with a return status immediately.

4.42 rfe_tc_046_configureChirpTime

The RFE Abstract API shall provide configuration parameters to set the chirp timing per profile.

4.42.1 Detailed Description

Chirp Timing parameters update.

Table 400. rfe_tc_046_configureChirpTime Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.

Table 400. rfe_tc_046_configureChirpTime Specification*...continued*

Property	Description
Execution	Automated
Satisfied Requirements	1296451

4.42.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
   2. Create a configuration with the RFE static use case. Parameters: Create 8 chirp profiles (0-8) configurations with other default chirp configuration Use 128 chirps and 512 samples, with single chirpSequences (1 frame) including the all the 8 profiles. i.e 16 * 8(profile) = 128 chirps Configure chirp timing parameters i.e., DwellTime,SettleTime, AcquisitionTime,Jumpbacktime,ResetTime,ChirpIntervalTime within the defined ranges in us as per SW reference manual. Note : us = ticks * 25 / 1000 Note : Range of the timing parameters will checked as part of TS/TC for requirement 1280173. Define different set values for each chirp profile. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
   4. Read the TimingEngine(TE) registers for each profile using the SPI Read API. e.g. DWELL_TIME_ PROFILE0,SETTLE_TIME_PROFILE0 ,CHIRP_INTERVAL_TIMER_PROFILE0 etc. Note : Value of 0xFFE = 102us. 0xFFE=4094, 4094/40= 102us. Here 40= 40 MHz clock
   - Assert to check the value of the registers to what was configured.
   5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
   6. Invoke rfe_getState() API.
   - Assert to check the Error parameter of the rfe_getState() API reply to be '0x0'. i.e No Error
   - Assert to check the state to be in 'rfe_state_configured_e'
   - In case of fault occurred, invoke rfe_getFuSaFaults() API and log/report the FuSa faults.

4.43 rfe_tc_047_configureChirpTimeDyTable

The RFE Abstract API shall provide configuration parameters to set the chirp timing per profile.

4.43.1 Detailed Description

Chirp Timing parameters update.

Table 401. rfe_tc_047_configureChirpTimeDyTable Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.

Table 401. rfe_tc_047_configureChirpTimeDyTable Specification*...continued*

Property	Description
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296451

4.43.2 Test Procedure

Steps:

1. 1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
   2. Create a configuration with the RFE static use case. Parameters: Create 8 chirp profiles (0-8) configurations with other default chirp configuration Use 128 chirps and 512 samples, with single chirpSequences (1 frame) including the all the 8 profiles. i.e 16 * 8(profile) = 128 chirps Configure chirp timing parameters i.e., DwellTime,SettleTime, AcquisitionTime,Jumpbacktime,ResetTime,ChirpIntervalTime within the defined ranges in us as per SW reference manual. Note : us = ticks * 25 / 1000 Note : Range of the timing parameters will checked as part of TestSpecification for requirement 1280173. Define different set values for each chirp profile. Create Dynamic tables for each of the 8 profile(s) with ChirpIntervalTime time and Dwelltime. i.e First 16 chirps with same valid values, define different values for next 16 chirps and so on. 3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and dynamic contents with length.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
   4. Read back memory for dynamic table. e.g. Dwelltime and ChirpIntervalTime.
   - Assert to check the value to what was configured/set.
   5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=50, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
   6. Update the values of the Dynamic table data i.e., ChirpIntervalTime time and Dwelltime with rfe_updateDynamicTable() and rfe_updatePush() API.
   - Assert to check the Error parameter of the APIs reply to be '0x0'. i.e No Error
   - Repeat the step 4.
   7. Invoke rfe_getState() API.
   - Assert to check the Error parameter of the rfe_getState() API reply to be '0x0'. i.e No Error
   - Assert to check the state to be in 'rfe_state_configured_e'
   - In case of fault occurred, invoke rfe_getFuSaFaults() API and log/report the FuSa faults.

4.44 rfe_tc_050_rfe_Abstract_API

This test case is used to check the RFE abstract API.

4.44.1 Detailed Description

Table 402. rfe_tc_050_rfe_Abstract_API Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.

Table 402. rfe_tc_050_rfe_Abstract_API Specification*...continued*

Property	Description
	QTA to be used to test.
Post	RFE FW should be in non-error state and should be able to perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280200

4.44.2 Test Procedure

Steps:

1. Check RFE abstract API with valid parameters and check the return status.
2. Check RFE abstract API with invalid parameters and verify error returned.
3. Check RFE abstract API function with equivalence and boundary input configurations.

4.45 rfe_tc_051

The RFE SW shall provide global RFE Safety configuration, via RFE Safety API

4.45.1 Detailed Description

Table 403. rfe_tc_051 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1381846

4.45.2 Test Procedure

Steps:

1. Check if RFE initialization is succesfull.
2. Check RFE is in initialized state for accepting frame configuration.
3. Configure RFE calibrate

4. Configure RFE frame
5. Configure the selected Self test with valid parameters and check the response code.
6. Start RFE Radar System Cycle.
7. Inject an error
8. Check if the error signalling matches the intended behaviour
9. Change error signalling for the error
10. Start RFE Radar System Cycle
11. Inject same error in Step 5
12. Check if error signallng matches intended behaviour

4.46 rfe_tc_052

This test case verifies if RFE SW shall perform refresh of memory cells with recoverable errors, via RFE Safety API

4.46.1 Detailed Description

Table 404. rfe_tc_052 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3062432

4.46.2 Test Procedure

Steps:

1. Check if RFE initialization is succesfull.
2. Check RFE is in initialized state for accepting frame configuration.
3. Configure RFE calibrate
4. Configure RFE frame
5. Configure selected Self test with valid parameters and check the response code.
6. Start RFE Radar System Cycle.
7. Inject single bit memory error
8. Check error statictics reported by RFE indicates single bit error occured
9. Inject multibit bit memory error
10. Verify if Error_N is asserted followed by chip reset

4.47 rfe_tc_054

This test case verifies if the RFE SW shall read the Rx ADC clipping counts, via RFE IP API

4.47.1 Detailed Description

Table 405. rfe_tc_054 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311225

4.47.2 Test Procedure

Steps:

1. Check if RFE initialization is succesfull.
2. Check RFE is in initialized state for accepting frame configuration.
3. Configure RFE calibrate
4. Configure RFE frame
5. Configure selected Self test with valid parameters and check the response code.
6. Start RFE Radar System Cycle.
7. Inject a ADC clipping event
8. Check if error statictics reported by RFE indicates ADC clipping occured

4.48 rfe_tc_055

The test case verifies if the RFE SW shall provide programming of pass criteria for self-tests, via RFE System API.

4.48.1 Detailed Description

Table 406. rfe_tc_055 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated

Table 406. rfe_tc_055 Specification*...continued*

Property	Description
Satisfied Requirements	1296779

4.48.2 Test Procedure

Steps:

1. Check if RFE initialization is succesfull.
2. Check if RFE is in initialized state for accepting frame configuration.
3. Configure RFE calibrate
4. Configure RFE frame
5. Configure selected Self test with valid parameters and check the response code
6. Start RFE Radar System Cycle.
7. Inject error in RX IP
8. Check if error statictics reported by RFE indicates an error is reported and error is recovered succesfully

4.49 rfe_tc_056

This test case verifies if the RFE SW shall provide setting of overtemp/undertemp thresholds for each temperature sensor in RFE, via RFE IP API

4.49.1 Detailed Description

Table 407. rfe_tc_056 Specification

Property	Description
Pre	-# Note Over and Under Temperature settings in OTP -# Put DUT in a temperature chamberN/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1310721, 1296734, 1311204

4.49.2 Test Procedure

Steps:

1. Check if RFE initialization is succesfull.
2. Check if RFE is in initialized state for accepting frame configuration.
3. Read back temperature.
4. Configure rfe_configureInterrupt() with RFE_EVENTS_IRQ_NONE
5. Configure the under/over temperature thresholds for all sensors to be triggered at room temperature.
6. Check if temperature fault occured for all combinations via checking rfe_monitorRead().

7. Configure rfe_configureInterrupt() with RFE_EVENTS_IRQ_ALL
8. Configure the under/over temperature thresholds for all sensors to be triggered at room temperature.
9. Check if temperature trigger occured for all combinations.
10. Configure the under/over temperature thresholds to default value (-40 .. 150 degree C)
11. Check that temperature trigger did not occur.
12. Configure rfe_configureInterrupt() with RFE_EVENTS_IRQ_RADAR_CYCLE_CHANGE
13. Start RFE Radar System Cycle.
14. Check if interrupt was triggered due to radar cycle change.
15. Repeat for all possible trigger types.

4.50 rfe_tc_057

This test case verifie if the RFE SW shall notify the host upon succesful error recovery

4.50.1 Detailed Description

Table 408. rfe_tc_057 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1377107

4.50.2 Test Procedure

Steps:

1. Check if RFE initialization is succesfull.
2. Check if RFE is in initialized state for accepting frame configuration.
3. Configure RFE calibrate
4. Configure RFE frame
5. Configure selected Self test with valid parameters and check the response code
6. Start RFE Radar System Cycle.
7. Inject HW recoverable error
8. Verify Error_N is asserted and deasserted within recovery interval

4.51 rfe_tc_058

This test case verifies if the RFE SW shall notify radar application about communication error via FCCU

4.51.1 Detailed Description

Table 409. rfe_tc_058 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safetyl
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311213, 1376611

4.51.2 Test Procedure

Steps:

1. Check if RFE initialization is succesfull.
2. Check if RFE is in initialized state for accepting frame configuration.
3. Configure RFE calibrate
4. Configure RFE frame
5. Configure selected Self test with valid parameters and check the response code
6. Start RFE Radar System Cycle.
7. Send a command over IPCF, simultaneously inject a CRC error.
8. Check if retramission is requested.
9. Resend the command, repeat steps 7,8,9 "N" times till retransmission times out.
10. Verify that Error_N is asserted

4.52 rfe_tc_059

This test case verifies successful initialization/calibration, RFE SW shall enter normal operation and report this to application

4.52.1 Detailed Description

Table 410. rfe_tc_059 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated

Table 410. rfe_tc_059 Specification*...continued*

Property	Description
Satisfied Requirements	1311216

4.52.2 Test Procedure

Steps:

1. Power on the RFE
2. Check if Error_N is asserted at power on and deasserted after RFE initialization
3. Check if RFE initialization is succesfull by configuring a Radar System Cycle and runs succesfuly over multiple cycles.

4.53 rfe_tc_062

This test is used to program interleaved or non-interleaved packing mode per output stream via the RFE Abstract API.

4.53.1 Detailed Description

Table 411. rfe_tc_062 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280214, 1280702

4.53.2 Test Procedure

Steps:

1. 1280214 - The RFE SW shall program interleaved or non-interleaved packing mode per output stream via the RFE Abstract API.
2. Power on the RFE
3. Initialize RFE
4.
5. Receive RFE single mode radar system cycle configuration
6. with interleaved packer configuration
7. with data test pattern generation configuration
8. Configure RFE using the received configuration
9. Check that RFE is configured correctly by reading back configured registers and configured variables
10.

11. Run radar system cycle 1x
12. Check that data is interleaved correctly
13.
14. Receive RFE single mode radar system cycle configuration
15. with non-interleaved packer configuration
16. with data test pattern generation configuration
17. Configure RFE using the received configuration
18. Check that RFE is configured correctly by reading back configured registers and configured variables
19.
20. Run radar system cycle 1x
21. Check that data is not-interleaved correctly
22.
23. repeat test case for all combinations of data outputs
24. 3 possible modes (PPE, CSI2, PPE/CSI2)
25. times
26. header / footer / interleaving
27. end of test

4.54 rfe_tc_064

This test is used to support defining multiple chirp sequences, via the RFE Radar Frame API

4.54.1 Detailed Description

Table 412. rfe_tc_064 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Safety
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280346

4.54.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Check RFE calibrate, frame and BIST configuration with valid parameters.
3. Check configuring RFE for single profile with multiple chirp sequences.
4. Check rfe_startRadarCycle().
5. Raw data is transferred to binary files or streamed to local host on PC and either visualized real-time or analysed offline with default data visualizer.

4.55 rfe_tc_065

This test is used to support configuration of maximum of 20MHz (analog) bandwidth per Chirp.

4.55.1 Detailed Description

Table 413. rfe_tc_065 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280438

4.55.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Start up Calibrations (RFE-M7) app independent
3. Configure RFE frame with maxmimum number 1024 samples per chirp
4. Start radar cycle
5. Check if the acquisition time (chirping time) is 51.2us

4.56 rfe_tc_066

This test is used to support defining a radar frame, a sequence of chirps comprising a number of modes (chirps of a given profile), with modes interleaved, via the RFE Radar Frame API.

4.56.1 Detailed Description

Table 414. rfe_tc_066 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A

Table 414. rfe_tc_066 Specification*...continued*

Property	Description
Execution	Automated
Satisfied Requirements	1280534

4.56.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Start up Calibrations (RFE-M7) app independent
3. Check RFE frame configuration static chirp parameter of each set of 8 profiles assigned to each chirp in sequence with valid parameters.
4. Check RFE frame start
5. Check using RFE SW System API for dynamic chirp interleaved multimode mode, overwrite the dynamic parameters of the TE profile registers used for the next-next chirp.

4.57 rfe_tc_067_outputStream

This test is used to provide programming interface for 4 independent output streams via RFE DFE API.

4.57.1 Detailed Description

Table 415. rfe_tc_067_outputStream Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and needs to be powered (fresh) on with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy is required to collect adc data.
Post	RFE FW should be in non-error state and should be able to sync and perform execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280542, 1280376

4.57.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.

2. Create a configuration with the RFE static use case. Parameters: Create 1 chirp profiles '0' configurations with other default chirp configuration Use 128 chirps and 512 samples, with single chirpSequences (1 frame). Set chirpProfileSequence sequence="0". Set the center frequency for chirp profiles 76500000
3. Configure the RFE using the rfe_configure() API with the blob contents of the above configuration. Parameters: Configblob contents with length and no dynamic table.
4. Program a different physical output for each RX channel (csi2VirtualChannel channel)
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x0'. i.e No Error
   - On successful RFE configuration, Assert to check for the current RFE state to be 'CONFIGURED'.
5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFETEST_RADAR_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
6. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.
7. Start radar cycle
8. Check the ADC data captured in each RX channel in a different buffer.

4.58 rfe_tc_068

This test is used to provide destination programming per output to be CSI-2 or/and PPE via RFE Abstract API.

4.58.1 Detailed Description

Table 416. rfe_tc_068 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	All checks are okay
HW Environment	STRX Board with CSI-2 RX
Execution	Automated
Satisfied Requirements	1280608

4.58.2 Test Procedure

Steps:

1. Initialize RFE and check response code
2. Configure RFE with a valid chirp sequence and check response code
3. Configure CSI-2 and check response code
4. Set data output destination to CSI-2 and check response code
5. Trigger a chirp sequence and check response code
6. Check that data is transfered via CSI-2
7. Configure PPE and check response code
8. Set data output destination to PPE and check response code

9. Trigger a chirp sequence and check response code
10. Check that data is transfered to System Ram
11. Configure CSI-2 and PPE check response code
12. Set data output destination to CSI-2 and PPE check response code
13. Trigger a chirp sequence and check response code
14. Check that data is transfered to CSI-2 and PPE
15.
16. Negative check: Configure dataOut to 'packet-processor="false" and csi2="false"' and verify the error response code.

4.59 rfe_tc_069

This test is used to execute System Cycle within the programmed time.

4.59.1 Detailed Description

Table 417. rfe_tc_069 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280635

4.59.2 Test Procedure

Steps:

1. Check RFE initialization with valid value and check the response code. - function test
2. Start up Calibrations (RFE-M7) app independent parameters and check the response code. - function test
3. Check RFE frame configuration with valid parameters and check the response code..
4. Check RFE frame start parameters and check the response code.
5. Check behaviour of RFE BIST parameters and check the response code.
6. Set radar system repetition rate
7. Start radar system cycle and the repetition rate is == programmed repetition rate +/- 1[us]

4.60 rfe_tc_074

This test is used to verify that the system can finalize start-up initializations and calibrations within 1[ms].

4.60.1 Detailed Description

Table 418. rfe_tc_074 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Measured time is <= 1ms
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299283

4.60.2 Test Procedure

Steps:

1. Load RFE and AppM7.
2. Start app at AppM7. Keep RFE M7 stalled
3. Start a timer.
4. Execute startup initializations and startup calibrations.
5. Stop timer.

4.61 rfe_tc_075_rdrCycStartDelay

This test is used to test start time delay of radar system cycle

4.61.1 Detailed Description

To synchronize with other sensors, the RFE radar cycle shall support starting delay functions in order to pause for a certain time, this test is to ensure the radar cycle pause as programmed.

Table 419. rfe_tc_075_rdrCycStartDelay Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	Qualification
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	-# No API error and No FUSA error reported in every step of the execution. -# Check the radar cycle start time is programmed start time +/- 1[us]
HW Environment	For Shark project need external MCU.
Execution	Automated
Satisfied Requirements	1301089

4.61.2 Test Procedure

Steps:

1. Test application host is A55, or AppM7, or MCU.
2. Initialize RFE and A53/AppM7/MCU.
3. Check RFE initialization with valid value.
4. Configure RFE with a proper/customer's configuration to send Radar Cycle Start Signal via RFE_GPIO_0 and check RFE errors
5. Enable A53/AppM7/MCU to read RFE_GPIO_0 status
6. On A53/AppM7/MCU, start timer, read RFE_GPIO_0 once, and stop timer after it, record the time, and repeat 100 times to get the averaged timing overhead H1.
7. step a: On A53/AppM7/MCU, start timer
8. step b: Start radar cycle with start_time specified to 10 us. In step h, this delay is programmed to different values.
9. step c: A53/AppM7 keep reading the RFE_GPIO_0 status, until detect the RFE_GPIO_0 is toggled
10. step d: A53/AppM7 stop timer and record the time
11. step e: repeat step abcd 100 time, and get an averaged start time delay D1
12. step f: divide the average delay D1 with the averaged overhead H1, get R1, which is R1 = D1 - H1.
13. step g: Check the radar cycle start time R1 was programmed start time +/- 1[us]
14.
15. step h: repeat step a, b, c, d, e, f, g, with 20us and 30us radar cycle starting time delay.

4.62 rfe_tc_077

This test is used to check if it finalize RFE Self-Tests within 2[ms].

4.62.1 Detailed Description

Table 420. rfe_tc_077 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1301539

4.62.2 Test Procedure

Steps:

1. Record / store start time (via timer)
2. Start RFE safety tests via BIST API
3. Upon completion record / store end time (via timer)

4. compare (end time - start time) to 2ms

4.63 rfe_tc_082

This test is used to provide functionality to apply radar use case (i.e. configure RFE).

4.63.1 Detailed Description

Table 421. rfe_tc_082 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1381831

4.63.2 Test Procedure

Steps:

1. Bring the system to a known good state. Call rfe_Configure() several times.
2. Run test case for near range radar (1007886).
3. Run test case for short range radar (1007900).
4. Run test case for medium range radar (1007901).
5. Run test case for long range radar (1007903).
6. Run test case for NCAP front sensor Adaptive Cruise Control (1007896).
7. Run test case for NCAP front sensor Autonomous Emergency Break (cities) (1007899).
8. Run test case for NCAP front sensor Autonomous Emergency Break (VRU) (1007897).
9. Run test case for near range radar gap filler (1007887).
10. Run test case for front corner sensor parking assist (1007892).
11. Run test case for front corner sensor traffic alert in cities (1007893).
12. Run test case for front corner sensor traffic alert in rural areas (1007894).
13. Run test case for rear corner sensor blind spot detection (1007888).
14. Run test case for rear corner sensor lange change assist (1007889).
15. Run test case for rear corner sensor parking assist (1007890).
16. Run test case for front corner sensor traffic alert (1007891).

4.64 rfe_tc_084

Check data from CSI2 TX DMA is streamed out via CSI2 at the specified data rate.

4.64.1 Detailed Description

Table 422. rfe_tc_084 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Data from CSI2 TX DMA is streamed out via CSI2 at the specified data rate. -# Check if ADC data value as programmed.
HW Environment	nan
Execution	Automated
Satisfied Requirements	2889870

4.64.2 Test Procedure

Steps:

1. * Develop test application on App M7
2. - Load RFE FW onto RFE M7 and init RFEM7
3. - Init BBE
4. - Call Abstract API sync to check the initialization status
5. - Use the CSI2 TX DMA driver to configure csi2 TX DMA
6. - Configure CSI2 TX DMA data path as follows:
7. SRAM -> CSI2 TX DMA -> CSI2 TX -> CSI2 RX of external MCU
8. - Configure the ADC to generate the test pattern and trigger the radar cycle to generate data
9. - Export ADC data from MCU to analyze
10. - Check if ADC data value as programmed ADC test pattern

4.65 rfe_tc_085

Check if user can select the leader-follower synchronization RFE IO via the RFE Abstract API.

4.65.1 Detailed Description

Table 423. rfe_tc_085 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error.

Table 423. rfe_tc_085 Specification*...continued*

Property	Description
	-# -Repeat the above test for 8 configurations with different RFE GPIOs Output Trigger of leader&Follower. Connection on HW needs to be changed in between(not interconnect all IOs at the same time) to prove that the right IOs are actually used. -# Analyze follower FFT data, if the target bin is at the expected distance for all chirps in 1st FFT result..
HW Environment	cascaded board
Execution	Automated
Satisfied Requirements	2883279

4.65.2 Test Procedure

Steps:

1. * Use Cascaded board
2. * Connection on HW needs to be changed in between(not interconnect all IOs at the same time): Leader RFE GPIO1 connects to Follower RFE GPIO1, Leader RFE GPIO2 connects to Follower RFE GPIO2,..., Leader RFE GPIO N connects to Follower RFE GPIO N. 8 connections.
3. * Develop test application on AppM7 to control both leader and follower(L&F)
4. - Load RFE FW onto RFE M7 on both L&F
5. - Call Abstract API sync to check the initialization status on both L&F
6. - Call Abstract API monitorRead to check the chip status on both L&F: All Temperatures info, TX output power, and all other monitorRead outputs. This step passes when all parameters as programmed or experienced value.
7. - One of Leader RFE GPIOs is configured as output for trigger follower
8. - One of the Follower RFE GPIOs is configured as an input trigger for radar cycle or configuration.
9. - Call Abstract API to configure to upload the configuration on both L&F
10. - Call Abstract API radarCycleStart to trigger chirp on Leader
11. - Analyze follower FFT data, if the target bin is at the expected distance for all chirps in 1st FFT result.
12. - Call Abstract API monitorRead to check the chip status on both L&F
13.
14. * Repeat the above test for 8 configurations with different RFE GPIOs Output Trigger of leader&Follower. Connection on HW needs to be changed in between(not interconnect all IOs at the same time) to prove that the right IOs are actually used. If the change of the HW setup is not desired, it should be checked on the scope that configured IOs are the ones actually used.
15. 8 variants.

4.66 rfe_tc_086

Test the RFE Abstract API shall provide a configuration parameter for selecting the radar cycle trigger GPIO on the leader device.

4.66.1 Detailed Description

Table 424. rfe_tc_086 Specification

Property	Description
Pre	N/A

Table 424. rfe_tc_086 Specification*...continued*

Property	Description
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Check if Time Domain data of all ADCs are in Sinus shape and with non-zero data -# Check if the detected target is at the expected distance for all chirps in 1st FFT result. -# Check Oscilloscope, the delay between channel 0(from signal generator) and channel 1(from RFE) shall be less than 25ns -# step g): Configure the input trigger GPIO as RFE GPIO3, the signal generator shall be connected to the RFE GPIO3, and the connection between the signal generator and scope stays, repeat steps b), c),d) This is to prove the RFE GPIO Trigger are selectable..
HW Environment	cascaded board, standalone Board
Execution	Automated
Satisfied Requirements	2883281, 2883290

4.66.2 Test Procedure

Steps:

1. * Use Cascaded Board
2. * Connect one of the free RFE GPIO(the largest index number is recommended) to the Radar Cycle Start IO input for standalone setup.
3. - Optional: TXs face to a ceiling or RTS, Ceiling test target shall be between 2~3 m, or RTS test target set to 50m
4. - Connect the RFE GPIO0 of the leader to the signal generator, await for the trigger signal
5. - Signal Generator connects to Oscilloscope and RFE GPIO0 at the same time with same length cable.
6. - RFE GPIO1 set as the output of TE Chirp Sequence signal. RFE GPIO1 is connected to the Oscilloscope Channel 2.
7. - Oscilloscope uses channel1 as the trigger signal
8. - Load RFE FW onto RFE M7 for both L&F
9. - Init Leader and Follower(L&F), configure L&F with general settings until both L&F are in configured state.
10. - Configure the PPOE to store data on SRAM
11. - step a): Configure Leader to receive trigger from RFE GPIO pin
12. - step b): use signal generator to generate a pulse as trigger source to trigger one radar cycle
13. - step c): Read Monitor and FUSA Statues
14. - Step d): Download ADC data from shared Memory and analyze data for both L&F
15. - Check if Time Domain data of all ADCs are in Sinus shape and with non zero data
16. - Check if the detected target is at expected distance for all chirps in 1st FFT result.
17. - Step e): Check Oscilloscope, the delay between channel 1 and channel 2 is the measured delay of chirp sequences, this measured delay and programmed delay difference shall be within 25ns(cycle-accurate)
18. * step f): Configure the Leader to receive trigger from Abstract API, send start radar cycle trigger via Abstract API, then repeat step c) and d).

19. * step g): Configure the input trigger GPIO as RFE GPIO3, the signal generator shall be connect to the RFE GPIO3, the connection between the signal generator and scope stays, repeat step b), c),d) This is to prove the RFE GPIO Trigger is selectable.
20. * step h): Do power cycle on the whole board, init setup in standalone mode, repeat step a), b), c), d), e), f).

4.67 rfe_tc_087

User can set the LOIF output power of the leader via the RFE Abstract API.

4.67.1 Detailed Description

Table 425. rfe_tc_087 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Measure the LO absolute output power of each loop and setting are matching the datasheet/validation result. -# Measured relative LO power difference between all of the possible settings.
HW Environment	cascaded board, FSW85/60
Execution	Automated
Satisfied Requirements	2889868

4.67.2 Test Procedure

Steps:

1. * Use Cascaded Board
2. - Connect LO_out of leader to FSW85/60
3. - Load RFE FW onto RFE M7 via Lauterbach
4. - Call Abstract API sync to check the initialization status
5. - Call Abstract API monitorRead to check the chip status
6. - Call Abstract API to configure to upload the configuration file
7. - Call Abstract API testContinuousWaveTransmissionStart, check FSW spectrum if there's a peak at target LO Frequency
8. - Check LO peak Power
9. * Collect the de-embedded loss information and cable loss information, these 2 pieces of information shall be added to the readout from FSW to calculate the real absolute LO output power from the STRX sample. Then check the absolute LO output power of each loop and setting are matching the datasheet/validation result.
10. * Repeat the Above test with all other possible LO power Settings, and check if the relative difference matches the setting difference
11.

12. General Settings:
13. 1 profile 1 sequence
14. Profile 0:
15. Center Frequency 76.8 GHz(Integer Frequency)
16. BW: 0GHz CW mode

4.68 rfe_tc_088

Check if the Abstract API support maximum of 40 MHz analog bandwidth per Chirp

4.68.1 Detailed Description

Table 426. rfe_tc_088 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# step e) Download sampling data from SRAM, which shall be 2048 sample data per chirp -# use FFT to process each chirp of data, and get the index of the highest peak as the measured target bin frequency. Also log this target bin index for step g). -# step g) Download sampling data from SRAM, which shall be 2048 sample data per chirp -# use FFT to process each chirp of data, and get the power at the target bin in step e), the power value shall be much lower than the value in step e) due to it shall be filtered out by LPF. The power difference shall match the datasheet or HW validation result..
HW Environment	stand-alone board
Execution	Automated
Satisfied Requirements	1280778

4.68.2 Test Procedure

Steps:

1. Use Stand-Alone Board/ or cascaded board in stand-alone mode.
2. TXs face RTS and program a target on RTS which will appear at 39MHz in the IF domain. Which is 39/40 of the radar detection range for a specific setting.
3. Test Application implemented on the AppM7
4. Load RFE FW
5. Sync with RFE, check initialization status and FUSA errors
6. Configure RFE with 80MHz sampling rate, and configured to store data in SRAM.
7. Configure 25.6us sampling time,
8. configure PDC decimation filter at 80MHz sampling rate
9. step a)Configure RX LPF shall be at least 40MHz

10. step b) Check configuration status and FUSA errors
11. step c) Trigger radar cycle with Abstract API
12. step d) Check RFE status and FUSA errors
13. step e) Download sampling data from SRAM, which shall be 2048 sample data per chirp
14. use FFT to process each chirp of data, and get the index of the highest peak as the measured target bin frequency. Also log this target bin index for step g).
15. The peak shall be visible only at 39MHz in the IF domain after FFT.
16. step f) Configure the RX LPF to be 20 MHZ, repeat step b,c,d)
17. step g) Download sampling data from SRAM, which shall be 2048 sample data per chirp
18. use FFT to process each chirp of data, and get the power at the target bin in step e), the power value shall be much lower than the value in step e) due to it shall be filtered out by LPF. The power difference shall match the datasheet or HW validation result.

4.69 rfe_tc_089

Test RFE SW shall configure follower devices to use LO signal of leader.

4.69.1 Detailed Description

Table 427. rfe_tc_089 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Check the follower is at expected frequency during recalibration for each profile..
HW Environment	cascaded board, FSW85/60
Execution	Automated
Satisfied Requirements	2883286, 2883289

4.69.2 Test Procedure

Steps:

1. * Use Cascaded Board
2. - Connect the target physical GPIO of the leader to the signal generator, await for the trigger signal
3. - Connect TX1 and GPIO signal to FSW,
4. - Optional: other TXs face a ceiling or Radar Target Simulator(RTS), Ceiling test target shall be between 2~3 m, RTS test target set to 50m, RTS is optional
5. - Load RFE FW onto RFE M7 via Lauterbach
6. - Call Abstract API sync to check the initialization status
7. - Call Abstract API monitorRead to check the chip status
8. - Call Abstract API to configure to upload the configuration file. Profile-dependent calibration must be enabled for all profiles.

9. - Configure the PPOE to store data on SRAM
10. - use a signal generator to generate a pulse as the trigger to the GPIO of the leader, to trigger a radar cycle
11. - Call Abstract API monitorRead to check the chip status
12. - Check FSW FM Time Domain, there shall be output at calibration period, calibration frequency shall be at the start frequency of each profile and chirp waveform during chirping
13. - check the delay between the GPIO trigger signal and before the first chirps start each sequence.
14. - Check FSW PM Time Domain, there shall be output at TX calibration at the start frequency and stable output at the chirping period
15. - Download ADC data
16. - Process FFT on each Profile individually, and check if FFT Domain has the highest peak at the target frequency for each profile.
17.
18. * Repeat Above the test with Cascaded Board, FSW is connected to TX1 of the follower, to check the follower is at the expected frequency during recalibration for each profile.
19.
20. Leader and Followers' settings are the same:
21. -profile 0~7 has the center freq: 76.5, 77, 77.5, 78,76.5, 77, 77.5, 78 GHz
22. -profile 0~7 has the chip slope: down down down down up up up up
23. -BW:500MHz
24. -8 profiles in 8 sequences, 1 profile per sequence
25. - Chirp Number: 16 for each sequence sequence
26. - Sample Number: 512
27. - All TX on, gain Max
28. - All RX on, gain Max
29. - HPF: 200kHz, LPF:20MHz

4.70 rfe_tc_090

Test RFE SW shall synchronize chirp transmission and reception on leader and follower devices to the same clock edge of the shared CLK.

4.70.1 Detailed Description

Table 428. rfe_tc_090 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# - Check if FFT Domain has highest peak at target frequency -# - Check if the Target bin is stable over chirp to chirp, sequence to sequence -# Check the phase variations of 4 ADCs on both Leader and follower(L&F) are stable, the variation of difference shall be smaller than the phase jump that is caused by the clock jitter value at the target bin.

Table 428. rfe_tc_090 Specification*...continued*

Property	Description
	-# - The Phase difference between all RXs on Leader and Follower shall be constant over constant over chirps and multiple sequences & radar cycles. -# -The phase difference between all RXs on both leader and follower shall be constant over test with the different distances -# - Trigger radar cycle multiple times and check phase for each time..
HW Environment	cascaded board
Execution	Automated
Satisfied Requirements	2883288

4.70.2 Test Procedure

Steps:

1. * Use Cascaded board
2. - other TXs face RTS
3. - RTS test target set to 50/80/100m
4. - Host application running on AppM7
5. - Load RFE FW onto RFE M7
6. - Call Abstract API sync to check the initialization status
7. - Call Abstract API monitorRead to check the chip status
8. - Call Abstract API to configure to upload the configuration file of test matrix 1
9. - Configure the PPOE to store data on SRAM
10. - Call Abstract API radarCycleStart to trigger chirp
11. - Call Abstract API monitorRead to check the chip status
12. - Download ADC data via Lauterbach
13. - Check if FFT Domain has the highest peak at a target frequency
14. - Check if the Target bin is stable over chirp to chirp, sequence to sequence
15. - Check the phase variations of 4 ADCs on both Leader and follower(L&F) are stable, the variation of difference shall be smaller than the phase jump that is caused by the clock jitter value at the target bin. The phase jump can be calculated with :
16. p_jitter = (d*2*BW)/(c*T_s*F_Clk)*360
17. Where p_jitter is the phase jump caused by the clock jitter, d is the target distance(100m), BW is the effective BW(1GHz), c is the light speed, T_s is the Sampling duration in one chirp, F_Clk is the clock frequency(40MHz), 360 is used to convert it into deg unit.
18. - Phase difference between all RXs on both leader and follower shall be constant over chirps and multiple sequences & radar cycles, the variation of this difference shall be within the calculated phase jump for all ADCs.
19. - Trigger radar cycle multiple times and check phase for each time.
20. - Repeated test over 3 different distance settings with RTS: 50/80/100m
21. - Phase difference between all RXs on both leader and follower shall be constant over test with the different distances, the variation of this difference shall be within the calculated phase jump for all ADCs.
22. * Repeat above Test Matrix 1, 2, 3, 4.
23. Test Matrix1:
24. Leader TX1 Power on, gain max, all other TXs from Leader and Follwers are off
25. Single Profile 0 is used on L&F
26. Test Matrix2:
27. Follower1 TX1 Power on, gain max, all other TXs from Leader and Follwers are off

28. Single Profile 0 is used on L&F
29. Test Matrix3:
30. Leader TX1 Power on, gain max, all other TXs from Leader and Follwers are off
31. 1 sequence 8 Profiles are used on L&F, do signal processing individually for each profile's ADC data, which means 8 FFT on 1 ADC data, L&F has 8 ADCs, which means 8*8= 64 FFT results shall be checked, no phase jump is expected.
32. Test Matrix4:
33. Follower1 TX1 Power on, gain max, all other TXs from Leader and Follwers are off
34. 1 sequence 8 Profiles are used on L&F, do signal processing individually for each profile's ADC data, which means 8 FFT on 1 ADC data, L&F have 8 ADCs, which means 8*8= 64 FFT results shall be checked , no phase jump is expected.
35.
36. General Test cases Leader and Followers' settings are the same:
37. -profile 0~7 has the same center freq: 78GHz
38. -profile 0~7 has the chirp slope: down chirp(falling chirp)
39. -BW:1GHz
40. -8 profiles in 8 sequences, 1 profile per sequence
41. - Chirp Number: 16 for each sequence sequence
42. - Sample Number: 512
43. - All RX on, gain Max
44. - HPF: 800kHz, LPF:20MHz

4.71 rfe_tc_091

Test RFE SW shall configure leader device to derive CLK from XO and calibrate its phase to that of CLK_IN.

4.71.1 Detailed Description

Table 429. rfe_tc_091 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Check Oscilloscope if there's stable CLK_IN signal and draw eye pattern on scope, check how narrow is the overlapping part..
HW Environment	cascaded board, Oscilloscope
Execution	Automated
Satisfied Requirements	2883291

4.71.2 Test Procedure

Steps:

1. * Use Cascaded board
2. - Connect the TX1 output to FSW, enable TX1 output. FSW configured to phase noise analysis mode
3. - Develop host application on AppM7
4. - Load RFE FW onto RFE M7 for both Leader and follower(L&F)
5. - Call Abstract API sync to check the initialization status of L&F
6. - Call Abstract API monitorRead to check the chip status
7. - Configure L&F to work in CW mode
8. - Configure the Leader to enable CLK_IN, CLK_IN input signal link to XO.
9. - Trigger the CW mode signal and observe the phase noise level from FSW
10. - Phase noise level in this test shall be compatible with the best result of System validation.

4.72 rfe_tc_092

The phase noise of the LO signal is reduced and the ADC sampling moment of leader and followers are aligned by Leader using XO clock.

4.72.1 Detailed Description

Table 430. rfe_tc_092 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Phase noise level in this test shall match the best result of IP validation..
HW Environment	Oscilloscope
Execution	Automated
Satisfied Requirements	3100226

4.72.2 Test Procedure

Steps:

1. * Host Application running on Local host PC
2. * Local Host shall have a driver to download the Oscilloscope result
3. * RFE IO of STRX is connected to one of the Oscilloscope channels. Configure the Oscilloscope to measure the signal frequency
4. * Load rfeFW and init RFE M7, check if RFEM7 in initialized state
5. * Use abstract API to enable the 40MHz clock output.
6. * Check the signal frequency from Oscilloscope.

4.73 rfe_tc_093

A cascading example application running on the APP-M7 core of the cascaded devices.

4.73.1 Detailed Description

Table 431. rfe_tc_093 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# Check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error..
HW Environment	cascaded board
Execution	Automated
Satisfied Requirements	3100231

4.73.2 Test Procedure

Steps:

1. * Use a cascaded board.
2. * Execute the Example application on the APP-M7
3. * Check if the status of each output of APP-M7 is as expected.
4. * Check if the test can finish without issues(e.g. no errors, data correct, etc.)

4.74 rfe_tc_094

Check RFE FW can operate ADC streaming via WDMA

4.74.1 Detailed Description

Table 432. rfe_tc_094 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# Check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Check if BBE report interrupt by each chirp.
HW Environment	nan
Execution	Automated
Satisfied Requirements	3100227, 3100228, 3100229, 3100230

4.74.2 Test Procedure

Steps:

1. * Develop a Test Application on AppM7 to check the below state one step by one step, the state shall be as described, and no API error.
2. * Host application is running on Standalone AppM7
3. * BBE shall be initialized
4. * Init STRX in standalone mode.
5. * Configure the RFE and enable N WDMA streams for N ADCs.
6. * Configure each WDMA to use 1/N of the maximum memory.
7. * Enable the ADC test sample mode and generate increasing test example data as output for all N ADCs.
8. * Enable BBE to deliver the interrupt info to AppM7, AppM7 log when BBE report interrupt signal is received
9. * Trigger ADC to send test pattern. And check the data value in memory shall increase for N sections.
10. * Check if the BBE report is interrupted by each chirp.
11. * Iterate over "N" from 1 to 4 for N WDMA streams and ADCs.

4.75 rfe_tc_095

Check if RFE SW support radar cycle trigger signal from GMAC within shared clock cycle accurate delay on the leader or standalone case.

4.75.1 Detailed Description

Table 433. rfe_tc_095 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	-# check RFE state one step by one step, the state shall be as described, and no API error nor Fusa Error or Internal Error. -# Check if the Time Domain data of all ADCs are in Sinus shape and with non-zero data -# Check if the detected target is at the expected distance for all chirps in 1st FFT result. -# Check Oscilloscope, the delay between channel 1 and channel 2 is the measured delay of chirp sequences, this measured delay and programmed delay difference shall be within 25ns(cycle-accurate) -# The test shall pass both on Cascaded and Standalone boards.
HW Environment	cascaded board, stand-alone board, Oscilloscope
Execution	Automated
Satisfied Requirements	3100232, 3100234

4.75.2 Test Procedure

Steps:

1. * Use Cascaded Board first, later use the Standalone board to repeat the same tests.
2. * Radar Cycle Start Signal and the Chirp Sequence Start Signal from TE is used.
3. - Optional: TXs face to ceiling or RTS. If a Ceiling test is used, the target shall be between 2~3 m, if RTS is used, the test target is set to 50m.
4. - Connect the RFE GPIO0 of the leader to Oscilloscope channel 1, and await the trigger signal.
5. - RFE GPIO1 set as Chirp Sequence signal output. RFE GPIO1 is connected to the Oscilloscope Channel 2.
6. - Oscilloscope uses channel1 as the trigger signal
7. - Load RFE FW onto RFE M7 for both L&F
8. - Init Leader and Follower(L&F), configure L&F with general settings until both L&F are in configured state.
9. - Configure STRX to store data on SRAM
10. - step a): Configure Leader to receive trigger from GMAC, and output TE radar cycle start signal via RFE GPIO0, TE Chirp Sequence Start Signal outputs via RFE GPIO1. The follower receives the radar cycle trigger from the TE of the Leader and configures the Leader to trigger the follower via RFE IO.
11. - step b): Configure GMAC to generate a radar cycle trigger to the leader
12. - step c): Read RFE Monitor and FUSA Statues
13. - Step d): Download ADC data from shared Memory and analyze data for both L&F
14. - Check if the Time Domain data of all ADCs are in Sinus shape and with non-zero data
15. - Check if the detected target is at the expected distance for all chirps in the 1st FFT result.
16. - Step e): Check Oscilloscope, the delay between channel 1 and channel 2 is the measured delay of chirp sequences, this measured delay and programmed delay difference shall be within 25ns(cycle-accurate)
17. - Step f): repeat the above steps on the Standalone board
18.
19. 1 profile 1 sequence
20. Profile 0:
21. Center Frequency 77 GHz
22. BW: 2GHz
23. Chirp Number: 16
24. Sample Number: 512
25. - All TX on, gain Max
26. - All RX on, gain Max
27. - HPF: 200kHz, LPF:20MHz

4.76 rfe_tc_096

Check if the Abstract API support PDC filter for 80Mhz sampling rate

4.76.1 Detailed Description

Table 434. rfe_tc_096 Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox

Table 434. rfe_tc_096 Specification*...continued*

Property	Description
Pass Criteria	-# check RFE state one step by one step, state shall be as described and no API error nor Fusa Error nor Internal Error. -# Download sampling data from SRAM, shall be 2048 sample data per chirp -# use FFT to process each chirp of data, get the index of the highest peak as the measured target bin frequency. -# The peak shall be visible only at 39MHz in the IF domain after FFT. SNR shall be larger than 30dB.
HW Environment	nan
Execution	Automated
Satisfied Requirements	3100235

4.76.2 Test Procedure

Steps:

1. Use Stand-Alone Board/ or cascaded board in stand-alone mode.
2. TXs face RTS and program a target on RTS which will appear at 39MHz in the IF domain. Which is 39/40 of the radar detection range for a specific setting.
3. Test Application implemented on the AppM7
4. Configure the PDC filter to 'real decimation filter', and this test is to prove this PDC filter mode works with an 80MHz sampling rate.
5. Load RFE FW
6. Sync with RFE, check initialization status and FUSA errors
7. Configure RFE with an 80MHz sampling rate, and configure to store data in SRAM.
8. Configure 25.6us sampling time,
9. configure PDC decimation filter at 80MHz sampling rate
10. Configure RX LPF shall be at least 40MHz
11. Check configuration status and FUSA errors
12. Trigger radar cycle with Abstract API
13. Check RFE status and FUSA errors
14. Download sampling data from SRAM, shall be 2048 sample data per chirp
15. use FFT to process each chirp of data, and get the index of the highest peak as the measured target bin frequency.
16. The peak shall be visible only at 39MHz in the IF domain after FFT. SNR shall be larger than 30 dB.

4.77 rfe_tc_101_validTxOutputPower

The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here different tx power in range is checked.

4.77.1 Detailed Description

The range is obtained from datasheet document (doc618065)

Table 435. rfe_tc_101_validTxOutputPower Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables.

Table 435. rfe_tc_101_validTxOutputPower Specification*...continued*

Property	Description
	RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and executenew tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296375

4.77.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it is not equal to -100.
3. Create a configuration with the RFE static use case by programming for '70' (7dBm) tx power, in a single chirp profile. Parameters : txPower = 7dBm, Sampling frequency = 40MHz and with only Tx1 enabled. txPpdThreshold should be 3dB less than target power. In recalibration, profile dependent should be enabled. Note : 1 tick = 25 ns for timing parameters.
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Select the transmitter power for monitor read-out using api rfe_monitorRead for TX1.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm
7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
   - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
8. Select the transmitter power for monitor read-out using api rfe_monitorRead for TX1.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm
9. Repeat the test for different tx power configurations for 5, 10, 13 in dBm.
10. Repeat the test for 10 times.

4.78 rfe_tc_102_lessthanMinTxOutputPower

The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here tx power less than Min range value is checked.

4.78.1 Detailed Description

The range is obtained from datasheet document (doc618065)

Table 436. rfe_tc_102_lessthanMinTxOutputPower Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in a state which does not prevent from next set of test cases from being executed.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296375

4.78.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it is not equal to -100.
3. Create a configuration with the RFE static use case by programming for '70' (7dBm) tx power, in a single chirp profile. Parameters : txPower = 7dBm, Sampling frequency = 40MHz and with only Tx1 enabled. txPpdThreshold should be 3dB less than target power..
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration.
5. Update the blob content with tx power value lower than 50 (5dBm) e.g 45 (4.5 dBm) Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'
   - Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
6. Clear the error and update the blob content with tx power value with 40 (4dBm). Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'
   - Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.

4.79 rfe_tc_103_greaterthanMaxTxOutputPower

The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here tx power greater than maximum range value is checked.

4.79.1 Detailed Description

The range is obtained from datasheet document (doc618065)

Table 437. rfe_tc_103_greaterthanMaxTxOutputPower Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in a state which does not prevent from next set of test cases from being executed.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296375

4.79.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it is not equal to -100.
3. Create a configuration with the RFE static use case by programming for '70' (7dBm) tx power, in a single chirp profile. Parameters : txPower = 7dBm, Sampling frequency = 40MHz and with only Tx1 enabled. txPpdThreshold should be 3dB less than target power..
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration.
5. Update the blob content with tx power value higher than 130 (13dBm) e.g 131 (13.1 dBm). Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'
   - Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
6. Clear the error and update the blob content with tx power value with 150 (15dBm). Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'
   - Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.

4.80 rfe_tc_104_multiProfileTxOutputPower

The RFE Radar Use Case API shall support programming of 5 to 13 dBm tx power. Here tx power for multi profile is checked.

4.80.1 Detailed Description

The range is obtained from datasheet document (doc618065)

Table 438. rfe_tc_104_multiProfileTxOutputPower Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and executenew tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296375

4.80.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it is not equal to -100.
3. Create a configuration with the RFE static use case in a multi chirp profile by programming for '50'(5dBm) tx power for profile 0, '60'(6dBm) tx power for profile 1, '70'(7dBm) tx power for profile 2, '80'(8dBm) tx power for profile 3, '90'(9dBm) tx power for profile 4, '110'(11dBm) tx power for profile 5, '120'(12dBm) tx power for profile 6, '130'(13dBm) tx power for profile 7. Parameters : txPower = 7dBm or 70 (in 0.1 steps) , Sampling frequency = 40MHz and enable all TX. txPpdThreshold should be 3dB less than target power..
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Select the transmitter power for monitor read-out using api rfe_monitorRead for all tx and all profiles.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm

7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for all profiles in sequence 0 (both profile independent and dependent)
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
   - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
8. Select the transmitter power for monitor read-out using api rfe_monitorRead for all tx and all profiles.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm
9. Repeat the test for 10 times.

4.81 rfe_tc_105_validTxEnable

The RFE Radar Use Case API shall support programming configuration parameters to configure 4 TX modules. Here different tx enable is checked.

4.81.1 Detailed Description

Different parameters of tx such as tx power and tx phase rotation is tested during requirements 1296375 and 1296376 respectively. The ppd threshold is tested as part of safety checks in other tests,

Table 439. rfe_tc_105_validTxEnable Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables on a setup which generates a reflection signal (ceiling test, corner reflector in anechoic chamber etc.,) and it's tx signal shouldn't be routed to any signal analyser. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to collect the adc data. Matlab/Python should be able to collect data and process for FFT.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280680

4.81.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Assert check the OTP contents, there are different TX variant samples, some samples only have 3TX enabled. Need to read OTP to confirm which TXs can be tested.

3. Create a configuration with the RFE static use case by programming by enabling txTransmissionEnable and txEnable, in a single chirp profile. Parameters : txTransmissionEnable = true, txEnable = true for tx-1 and false for other tx's
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
7. Collect the adc data
   - Assert that ADC data is non-zero.
8. Normalized ADC data: Normalized data = (Data-2^15)/(SampleNum* ChirpNum * 4). Sum the ADC data: Sumed data = sum(Normalized data)
   - If sum the ADC data below 20 (or 30, based on noise), means TX is disabled, otherwise TX is enabled.
9. Repeat the test for different tx enable configurations. Eg. tx-2 true others false etc.
   - Assert that adc data is above noise level. Noise level is calculated when all tx are disabled for reference.
10. Repeat the test for no tx enable configurations. All tx txTransmissionEnable and txEnable should be false.
    - Should return no error.
11. Normalized ADC data: Normalized data = (Data-2^15)/(SampleNum* ChirpNum * 4). There should not be any transmissions, so received data should be noise. Sum the ADC data: Sumed data = sum(Normalized data)
    - If sum the ADC data below 20 (or 30, based on noise), means TX is not transmitting, otherwise TX is transmitting.
12. Repeat the test for no tx txTransmissionEnable configurations. All tx txTransmissionEnable should be false but tx enable should be true.
    - Should return no error.
13. Normalized ADC data: Normalized data = (Data-2^15)/(SampleNum* ChirpNum * 4). There should not be any transmissions, so received data should be noise. Sum the ADC data: Sumed data = sum(Normalized data)
    - If sum the ADC data below 20 (or 30, based on noise), means TX is not transmitting, otherwise TX is transmitting.

4.82 rfe_tc_106_validTxPhaseRotation

The RFE Radar Use Case API shall support programming of 0 to 357.1875 degrees for phase rotation (in steps of 2.8125 degrees). Here different phase rotation in range is checked.

4.82.1 Detailed Description

Table 440. rfe_tc_106_validTxPhaseRotation Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables on a setup which generates a reflection signal (ceiling test, corner reflector in anechoic chamber etc.,). RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to collect the adc data. Matlab/Python should be able to collect data and process for FFT.

Table 440. rfe_tc_106_validTxPhaseRotation Specification*...continued*

Property	Description
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296376

4.82.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1/2/3/4_PR_PHASE_CONTROL_PROFILE0.
   - Assert that the phase rotation read-out is 0 degrees. Note: Maximum value for bit field is 128, so I lsb is 360/128 *1 = 2.8125 degree.
3. Create a configuration with the RFE static use case by programming for '90' tx phase rotation, in a single chirp profile for all Tx. Parameters : txPhaseRotation = 90 degrees, Sampling frequency = 40MHz
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1_PR_PHASE_CONTROL_PROFILE0.
   - Assert that the phase rotation read-out is 32 (bitfield value) or 90 degrees. Note: Maximum value for bit field is 128, so I lsb is 360/128 *1 = 2.8125 degree.
7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
8. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1/2/3/4_PR_PHASE_CONTROL_PROFILE0.
   - Assert that the phase rotation read-out is 32 (bitfield value) or 90 degrees. Note: Maximum value for bit field is 128, so I lsb is 360/128 *1 = 2.8125 degree.
9. Collect the adc data
   - Assert that ADC data is non-zero.
   - Assert that FFT-1 data with matlab/python and check if phase shift is correct for tx1 (or the tx with which antenna is connected).
10. Repeat the test for different tx phase rotation configurations of 0, 90, 180, 270, 337.5, 357.1875.
11. Repeat the test for different tx phase rotation configurations for different txs.

4.83 rfe_tc_107_invalidTxPhaseRotation

The RFE Radar Use Case API shall support programming of 0 to 357.1875 degrees for phase rotation (in steps of 2.8125 degrees). Here invalid phase rotation is checked.

4.83.1 Detailed Description

Table 441. rfe_tc_107_invalidTxPhaseRotation Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to collect the adc data.
Post	RFE FW should be in a state which does not prevent from next set of test cases from being executed
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296376

4.83.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1_PR_PHASE_CONTROL_PROFILE0.
   - Assert that the phase rotation read-out is 0 degrees. Note: Maximum value for bit field is 128, so I lsb is 360/128 *1 = 2.8125 degree.
3. Create a configuration with the RFE static use case by programming for '90' tx phase rotation, in a single chirp profile. Parameters : txPhaseRotation = 90 degrees, Sampling frequency = 40MHz
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1_PR_PHASE_CONTROL_PROFILE0.
   - Assert that the phase rotation read-out is 32 (bitfield value) or 90 degrees. Note: Maximum value for bit field is 128, so I lsb is 360/128 *1 = 2.8125 degree.
7. Update the blob content with tx phase rotation value higher than 357.1875 e.g 360. Parameters : Config blob contents with length and No Dynamic table data.

- Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_ CONFIGURATION_PARAMETERVALUE'

4.84 rfe_tc_108_multiProfileTxPhaseRotation

The RFE Radar Use Case API shall support programming of 0 to 357.1875 degrees for phase rotation (in steps of 2.8125 degrees). Here invalid phase rotation is checked.

4.84.1 Detailed Description

Table 442. rfe_tc_108_multiProfileTxPhaseRotation Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables on a setup which generates a reflection signal (ceiling test, corner reflector in anechoic chamber etc.,). RFE M7, APP M7 and A53 cores should up. RFE Proxy should be started to collect the adc data. Matlab/Python should be able to collect data and process for FFT.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296376

4.84.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1_PR_PHASE_CONTROL_PROFILE0. Similarly check for all the profiles and all the tx's.
   - Assert that the phase rotation read-out is 0 degrees. Note: Maximum value for bit field is 128, so I lsb is 360/128 *1 = 2.8125 degree.
3. Create a configuration with the RFE static use case in a multi chirp profile by programming for '0' tx phase rotation for profile 0, '45' tx phase rotation for profile 1, '90' tx phase rotation for profile 2, '135' tx phase rotation for profile 3, '180' tx phase rotation for profile 4, '225' tx phase rotation for profile 5, '270' tx phase rotation for profile 6, '357.1875' tx phase rotation for profile 7.
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.

- On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.

6. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1_PR_PHASE_CONTROL_PROFILE0.
   - Assert that the phase rotation read-out for each profile based on input. Note: Maximum value for bit field is 128, so l lsb is 360/128 *1 = 2.8125 degree.
7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
8. Select the SPI read for testSetParam to read phase rotation value in the register TE_PR_PHASE_CTL_PROF0 and bit field TX1/2/3/4_PR_PHASE_CONTROL_PROFILE0.
   - Assert that the phase rotation read-out for each profile based on input. Note: Maximum value for bit field is 128, so l lsb is 360/128 *1 = 2.8125 degree.
9. Collect the adc data
   - Assert that ADC data is non-zero.
   - Assert that FFT-1 data with matlab/python and check if phase shift is correct for tx1 (or the tx with which antenna is connected).

4.85 rfe_tc_110_dynProf

The RFE Radar Use Case API shall program the id of the profile to be used for the next-next chirp into the TE while dynamically programming.

4.85.1 Detailed Description

Table 443. rfe_tc_110_dynProf Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE proxy and matlab should be enabled to collect adc data. Spectrum analyser needs to be connected to observe the dynamic chirp updated or not for different profiles.
Post	RFE FW should be in non-error state and should be able to perform and executenew tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1311321

4.85.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.

- On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.

2. Create a configuration with the RFE static use case in a 8 profiles single chirp sequence with 8 chirps and dynamic updates enabled with phase rotation and frequency drift should be 0 in static configuration. chirp-profile="0" phase-rotation-tx-1="0" phase-rotation-tx-2="270" phase-rotation-tx-3="0" phase-rotation-tx-4="90" transmission-enable-tx-1="enabled" transmission-enable-tx-2="disabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="800" chirp-frequency-drift-steps="188" chirp-profile="1" phase-rotation-tx-1="180" phase-rotation-tx-2="180" phase-rotation-tx-3="90" phase-rotation-tx-4="0" transmission-enable-tx-1="disabled" transmission-enable-tx-2="enabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="600" chirp-frequency-drift-steps="232" chirp-profile="2" phase-rotation-tx-1="0" phase-rotation-tx-2="270" phase-rotation-tx-3="0" phase-rotation-tx-4="90" transmission-enable-tx-1="enabled" transmission-enable-tx-2="disabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="800" chirp-frequency-drift-steps="188" chirp-profile="3" phase-rotation-tx-1="180" phase-rotation-tx-2="180" phase-rotation-tx-3="90" phase-rotation-tx-4="0" transmission-enable-tx-1="disabled" transmission-enable-tx-2="enabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="600" chirp-frequency-drift-steps="232" chirp-profile="4" phase-rotation-tx-1="0" phase-rotation-tx-2="270" phase-rotation-tx-3="0" phase-rotation-tx-4="90" transmission-enable-tx-1="enabled" transmission-enable-tx-2="disabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="800" chirp-frequency-drift-steps="188" chirp-profile="5" phase-rotation-tx-1="180" phase-rotation-tx-2="180" phase-rotation-tx-3="90" phase-rotation-tx-4="0" transmission-enable-tx-1="disabled" transmission-enable-tx-2="enabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="600" chirp-frequency-drift-steps="232" chirp-profile="6" phase-rotation-tx-1="0" phase-rotation-tx-2="270" phase-rotation-tx-3="0" phase-rotation-tx-4="90" transmission-enable-tx-1="enabled" transmission-enable-tx-2="disabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="800" chirp-frequency-drift-steps="188" chirp-profile="7" phase-rotation-tx-1="180" phase-rotation-tx-2="180" phase-rotation-tx-3="90" phase-rotation-tx-4="0" transmission-enable-tx-1="disabled" transmission-enable-tx-2="enabled" transmission-enable-tx-3="enabled" transmission-enable-tx-4="enabled" chirp-interval-time-ticks="600" chirp-frequency-drift-steps="232"
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
   - Use spi read to check for all the profiles if tx phase rotation and chirp frequency drift registers are set to 0.
5. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for all profiles in sequence 0 (both profile independent and dependent)
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
   - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
   - Use spi read to check for all the profiles if tx phase rotation and chirp frequency drift registers are updated correctly based on dynamic updateds.
   - Assert using spectrum analyser for dynamic updated correctly for different profiles.
   - Assert using matlab for fft-1 after collecting the adc-data to check the tx phase rotation updates for the tx where antenna is connected.
6. Repeat the test for profile id 8
   - Assert to check the rfe_configure returns error 7.
   - Assert to check the current RFE state to be 'Initialised' using rfe_getState() API.

4.86 rfe_tc_111_rxGainSingleProfile

The RFE Radar Use Case API shall support programming of 25dB to 46dB rx gain. Here different rx gain in range and out of range is checked.

4.86.1 Detailed Description

Table 444. rfe_tc_111_rxGainSingleProfile Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in a state which does not prevent from next set of test cases from being executed.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280787

4.86.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for '0' rx gain, in a single chirp profile.
   Parameters : rxGain = 0 (25dB), Sampling frequency = 40MHz
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration.
   Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Select the rx gain registers P0_IF_ST1_CIN, P0_IF_ST2_CIN, P0_IF_ST1_CF and P0_IF_ST2_CF in module RX1 using spi read.
   - Assert that register read out are different from reset values of the register.
6. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
7. Collect the adc data
   - Assert that ADC data is non-zero.
8. Normalized ADC data: Normalized data = (Data-2^15)/(SampleNum* ChirpNum * 4). Sum the ADC data: Sumed data = sum(Normalized data)
   - Assert that ADC data is observing the gain increase until saturation is reached through sum of the ADC data.

9. Repeat the test for different rx gain configurations for 3 (34dB), 5 (40dB), 6 (43dB), 7 (46dB).
   - Assert that register read out are different from previous gain values of the register.
10. Repeat the test for invalid rx gain configurations for 8.
    - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'
    - Assert to check for the current RFE state to be 'INITIALIZED'.

4.87 rfe_tc_112_rxGainMultiProfile

The RFE Radar Use Case API shall support programming of 25dB to 46dB rx gain. Here different rx gain in range and out of range is checked.

4.87.1 Detailed Description

Table 445. rfe_tc_112_rxGainMultiProfile Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1280787

4.87.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case in a multi chirp profile by programming for '0' (25dB) rx gain for profile 0, '1' (28dB) rx gain for profile 1, '2' (31dB) rx gain for profile 2, '3' (34dB) rx gain for profile 3, '4' (37dB) rx gain for profile 4, '5' (40dB) rx gain for profile 5, '6' (43dB) rx gain for profile 6, '7' (46dB) rx gain for profile 7.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Select the rx gain registers P0_IF_ST1_CIN, P0_IF_ST2_CIN, P0_IF_ST1_CF and P0_IF_ST2_CF in module RX1 using spi read. Similar for all the profiles.
   - Assert that register read out are different from reset values of the register.
   - Assert that register read out are different for different profiles.

6. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
7. Collect the adc data
   - Assert that ADC data is non-zero.
8. Normalized ADC data: Normalized data = (Data-2^15)/(SampleNum* ChirpNum * 4). Sum the ADC data: Sumed data = sum(Normalized data)
   - Assert that ADC data is observing the gain increase until saturation is reached through sum of the ADC data.

4.88 rfe_tc_113_hpfRxSingleProfile

The RFE Radar Use Case API shall support programming of 200kHz to 6400kHz hpf cutoff frequency for rx. Here different rx hpf cutoff frequencies in range and out of range is checked.

4.88.1 Detailed Description

Table 446. rfe_tc_113_hpfRxSingleProfile Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. RFE proxy to capture adc data. Matlab to process the captured adc data.
Post	RFE FW should be in a state which does not prevent from next set of test cases from being executed.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296344

4.88.2 Test Procedure

Steps:

1. a. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for '0'(200kHz) hpf cutoff frequency, in a single chirp profile. Parameters : high-pass-filter= 0 (200kHz), Sampling frequency = 40MHz
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.

5. Select the rx hpf cutoff frequency registers P0_IF_R in module RX1 using spi read.
   - Assert that register read out are different from reset values of the register.
6. Repeat the test for different rx hpf cutoff frequency configurations for 3 (800kHz), 5 (3200kHz).
   - Assert that register read out are different from previous hpf cutoff frequency values of the register.
7. b. Create a configuration with the RFE static use case by programming for '4'(`1600kHz) hpf cutoff frequency, in a single chirp profile. Parameters : high-pass-filter= 4 (`1600kHz), Sampling frequency = 40MHz, effective-chirp-bandwidth-kHz="1000000" (BW = 1Ghz), acquisition-time-ticks="512"
8. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
9. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
10. Capture the Raw ADC data.
11. Perform Average FFT Range post processing to visualize the averaged amplitude for both the profiles. f = (2*2) / 299792458 * 1000000000 /(512 * 25e-09) = 1.0424 MHz Attached are calculations for references in RFE_1296344.docx. Test_rfe\integration\specific\STRX\doc\reference
   - Assert that IF frequencies higher then HPF are staying constant in power.
12. c. Create a configuration with the RFE static use case by programming for '4'(`1600kHz) hpf cutoff frequency, in a single chirp profile. Parameters : high-pass-filter= 4 (`1600kHz), Sampling frequency = 40MHz, effective-chirp-bandwidth-kHz="250000" (BW = 250MHz), acquisition-time-ticks="512"
13. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
14. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
15. Capture the Raw ADC data.
16. Perform Average FFT Range post processing to visualize the averaged amplitude for both the profiles. f = (2*2) / 299792458 * 250000000/(512 * 25e-09) = 260.600 kHz Attached are calculations for references in RFE_1296344.docx. Test_rfe\integration\specific\STRX\doc\reference
   - Assert that IF frequencies lower then HPF are suppressed.
17. Repeat the test from step c with varying the hpf.
   - Assert that the higher delta between IF frequency and the HPF frequency the bigger the suppression.
18. Repeat the test from step a for invalid rx hpf cutoff frequency configurations for 7.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'
   - Assert to check for the current RFE state to be 'INITIALIZED'.

4.89 rfe_tc_114_hpfRxMultiProfile

The RFE Radar Use Case API shall support programming of 200kHz to 6400kHz hpf cutoff frequency for rx. Here different rx hpf cutoff frequencies in range and out of range is checked.

4.89.1 Detailed Description

Table 447. rfe_tc_114_hpfRxMultiProfile Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables.

Table 447. rfe_tc_114_hpfRxMultiProfile Specification*...continued*

Property	Description
	RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296344

4.89.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case in a multi chirp profile by programming for '0' (200kHz) rx hpf cutoff frequency for profile 0, '1' (300kHz) rx hpf cutoff frequency for profile 1, '2' (400kHz) rx hpf cutoff frequency for profile 2, '3' (800kHz) rx hpf cutoff frequency for profile 3, '4' (1600kHz) rx hpf cutoff frequency for profile 4, '5' (3200kHz) rx hpf cutoff frequency for profile 5, '6' (6400kHz) rx hpf cutoff frequency for profile 6.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Select the rx hpf cutoff frequency registers P0_IF_R in module RX1 using spi read.
   - Assert that register read out are different from reset values of the register.
   - Assert that register read out are different for different profiles.

4.90 rfe_tc_115_lpfRxSingleProfile

The RFE Radar Use Case API shall support programming of 10MHz to 40MHz lpf cutoff frequency for rx. Here different rx lpf cutoff frequencies in range and out of range is checked.

4.90.1 Detailed Description

Table 448. rfe_tc_115_lpfRxSingleProfile Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in a state which does not prevent from next set of test cases from being executed.

Table 448. rfe_tc_115_lpfRxSingleProfile Specification*...continued*

Property	Description
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296344

4.90.2 Test Procedure

Steps:

1. a. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for '0'(10MHz) lpf cutoff frequency, in a single chirp profile. Parameters : low-pass-filter= 0 (10MHz), Sampling frequency = 40MHz
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Select the lpf cutoff frequency registers P0_IF_R in module RX1 using spi read.
   - Assert that register read out are different from reset values of the register.
6. Repeat the test for different lpf cutoff frequency configurations for 3 (25MHz), 5 (40MHz).
   - Assert that register read out are different from previous gain values of the register.
7. b. Create a configuration with the RFE static use case by programming for '0'(10MHz) lpf cutoff frequency in a single chirp profile. Parameters : low-pass-filter= 0 (10MHz), Sampling frequency = 40MHz, effective-chirp-bandwidth-kHz="4000000" (BW = 4Ghz), acquisition-time-ticks="128"
8. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
9. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API Parameters : cycle_count=RFE_CYCLE_COUNT, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0x0'. i.e No Error
10. Capture the Raw ADC data.
11. Perform Average FFT Range post processing to visualize the averaged amplitude for both the profiles. F = (2*2) / 299792458 * 4000000000 /(128 / 40000000) = 16.678 Mhz Attached are calculations for references in RFE_1296344.docx. Test_rfe\integration\specific\STRX\doc\reference
    - Assert that IF frequencies higher then LPF are suppressed.
12. c. Repeat the test from step b with low-pass-filter= 2 (20MHz),
    - Assert that IF frequencies lower then LPF are staying constant in power.
13. Repeat the test for invalid lpf cutoff frequency configurations for 6.
    - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'
    - Assert to check for the current RFE state to be 'INITIALIZED'.

4.91 rfe_tc_116_lpfRxMultiProfile

The RFE Radar Use Case API shall support programming of 10MHz to 40MHz lpf cutoff frequency for rx. Here different rx lpf cutoff frequencies in range and out of range is checked.

4.91.1 Detailed Description

Table 449. rfe_tc_116_lpfRxMultiProfile Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296344

4.91.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case in a multi chirp profile by programming for '0' (10MHz) rx lpf cutoff frequency for profile 0, '1' (15MHz) rx lpf cutoff frequency for profile 1, '2' (20MHz) rx lpf cutoff frequency for profile 2, '3' (25MHz) rx lpf cutoff frequency for profile 3, '4' (30MHz) rx lpf cutoff frequency for profile 4, '5' (40MHz) rx lpf cutoff frequency for profile 5.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Select the lpf cutoff frequency registers P0_IF_R in module RX1 using spi read.
   - Assert that register read out are different from reset values of the register.
6. Repeat the test for different lpf cutoff frequency configurations for 3 (25MHz), 5 (40MHz).
   - Assert that register read out are different from previous gain values of the register.
7. Repeat the test for invalid lpf cutoff frequency configurations for 6.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_CONFIGURATION_PARAMETERVALUE'

4.92 rfe_tc_117_txOutputPowerLR

The RFE Radar Use Case API shall support programming of tx power for long range usecase.

4.92.1 Detailed Description

Table 450. rfe_tc_117_txOutputPowerLR Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and executenew tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296375

4.92.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it is not equal to -100.
3. Create a configuration with the RFE static use case by programming for 13dBm tx power, in a single chirp profile. Parameters : txPower = 13dBm, ChirpCount=128, Center Frequency=76.3GHz, Sampling Frequency=40MHz, ChirpBandwidth=320MHz, Samples=2048 and all Tx enabled.
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm
7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
   - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
8. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it crosses the limit.

a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm

4.93 rfe_tc_118_txOutputPowerSR

The RFE Radar Use Case API shall support programming of tx power for short range usecase.

4.93.1 Detailed Description

Table 451. rfe_tc_118_txOutputPowerSR Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and executenew tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296375

4.93.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it is not equal to -100.
3. Create a configuration with the RFE static use case by programming for 7dBm tx power, in a single chirp profile. Parameters : txPower = 7dBm, ChirpCount=8, Center Frequency=76.45GHz, Sampling Frequency=40MHz, ChirpBandwidth=810MHz, Samples=2048 and all Tx enabled.
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm
7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error

  - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
8. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm

4.94 rfe_tc_119_txOutputPowerSaving

The RFE Radar Use Case API shall support programming of tx power for power saving usecase.

4.94.1 Detailed Description

Table 452. rfe_tc_119_txOutputPowerSaving Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and executenew tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1296375

4.94.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the transmitter power for monitor read-out using api rfe_monitorRead.
   - Assert that tx power read-out if it is not equal to -100.
3. Create a configuration with the RFE static use case by programming for 5dBm tx power, in a single chirp profile. Parameters : txPower = 5dBm, ChirpCount=128, Center Frequency=76.45GHz, Sampling Frequency=40MHz, ChirpBandwidth=810MHz, Samples=2048 and all Tx is enabled.
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Select the transmitter power for monitor read-out using api rfe_monitorRead for all tx.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm

7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
   - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
8. Select the transmitter power for monitor read-out using api rfe_monitorRead for all tx.
   - Assert that tx power read-out if it crosses the limit.
   a. Allowed range is +- 0.5 dBm of input power target = [5, 13] dBm. For example if input power target is 7dBm, allowed range is 7dB +- 0.5 dBm

4.95 rfe_tc_120_validRxSatDetThresholdLevel

The RFE Radar Use Case API shall support programming of rx saturation detector threshold level. Here different rx saturation detector threshold level in range is checked.

4.95.1 Detailed Description

Table 453. rfe_tc_120_validRxSatDetThresholdLevel Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1375526

4.95.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the rx saturation count of all enabled rx for monitor read-out using api rfe_monitorRead.
   - Assert that rx saturation count of all enabled rx for monitor read-out is 0.
3. Create a configuration with the RFE static use case by programming for '0'(-13_05dB) threshold in a single chirp profile. Parameters : rxSaturationThresholdRx1 power-stage-1 and power-stage-2 to 0(-13_05dB), max-count-stage-1 and max-count-stage-2 to 1000.
4. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
5. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.

6. Select the rx saturation count of all enabled rx for monitor read-out using api rfe_monitorRead.
   - Assert that rx saturation count of all enabled rx for monitor read-out is 0.
   - Assert that rx saturation thresold by using SPI read of register SAT_DET_CTL and bit field st1Level or st2Level depending on stage for corresponding rxc.
7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
8. Select the rx saturation count of all enabled rx for monitor read-out using api rfe_monitorRead.
   - Assert that rx saturation count of all enabled rx for monitor read-out is 0.
   - Assert that rx saturation thresold by using SPI read of register SAT_DET_CTL and bit field st1Level or st2Level depending on stage for corresponding rx.
9. Repeat the test for different rx thresholds for power-stage-1 to 3(-10_44dB), 6(-7_83dB), 9(-5_22dB), 12(-2_61dB), 15(0dB).
10. Repeat the test for different rx thresholds for power-stage-2 to 3(-10_44dB), 6(-7_83dB), 9(-5_22dB), 12(-2_61dB), 15(0dB).
11. Repeat the test for different rx's. Eg., rxSaturationThresholdRx2, rxSaturationThresholdRx3, rxSaturationThresholdRx4.

4.96 rfe_tc_121_invalidRxSatDetThresholdLevel

The RFE Radar Use Case API shall support programming of rx saturation detector threshold level. Here rx saturation detector threshold level out of range is checked.

4.96.1 Detailed Description

Table 454. rfe_tc_121_invalidRxSatDetThresholdLevel Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in a state which does not prevent from next set of test cases from being executed.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1375526

4.96.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Select the rx saturation count of all enabled rx for monitor read-out using api rfe_monitorRead.

- Assert that rx saturation count of all enabled rx for monitor read-out is 0.

3. Create a configuration with the RFE static use case by programming for '0'(-13_05dB) threshold in a single chirp profile. Parameters : rxSaturationThresholdRx1 power-stage-1 and power-stage-2 to 0, max-count-stage-1 and max-count-stage-2 to 1000.
4. Update the blob content with rxSaturationThresholdRx1 power-stage-1 to 16
5. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0x07'. i.e 'API_INVALID_ CONFIGURATION_PARAMETERVALUE'
   - Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
6. Repeat the test for different rx's. Eg., rxSaturationThresholdRx2, rxSaturationThresholdRx3, rxSaturationThresholdRx4.

4.97 rfe_tc_122_validChirpFreqSweep

The RFE Radar Use Case API shall support programming to set chirp frequency sweep. Here different center frequency in range is checked. Frequency drift per chirp in 38.14697265625 Hz

4.97.1 Detailed Description

Table 455. rfe_tc_122_validChirpFreqSweep Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1310833

4.97.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for '76500000' center frequency in a single chirp profile. Parameters : center-frequency-kHz="76500000" effective-chirp-bandwidth-kHz="1000000" vco="1 GHz" slope="rising" chirp-interval-time-ticks="3000" dwell-time-ticks="6" settle-time-ticks="92" acquisition-time-ticks="512" reset-time-ticks="300" chirpFrequency->frequency-drift-steps="0"
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.

- Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Calculate change in reset slope value based on below equation: chirpSlopeCode = roundFloat(effChirpBandwidth * 1000 / acquisitionTimeTicks * 0.0262144); driftFrequency = (slopeDirection == rfe_chirpSlopeDirection_rising_e) ? frequencyDriftSteps : -frequencyDriftSteps; resetBandwidth = chirpSlopeCode * (settleTimeTicks + acquisitionTimeTicks) - driftFrequency; resetSlopeCode = roundFloat(resetBandwidth / resetTimeTicks);
   - Assert that reset slope code by using SPI read of register P0_RESET in chirppll.
6. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
   - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
7. Repeat the test for different chirp frequencies sweep Eg; 10, 1000 etc., till it crosses the end frequency / 38.14697265625.
8. Repeat the test for different center frequencies 78000000, 78500000, 79000000, 80000000, 80500000, 81000000
9. Repeat the test for slope="falling" and different VCO.

4.98 rfe_tc_123_clkio

The RFE Radar Use Case API shall provide a configuration parameters to configure CLK_IN and CLK_OUT pins.

4.98.1 Detailed Description

Table 456. rfe_tc_123_clkio Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. Clk out pin connected to clk in pin.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1310709

4.98.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API if not already intialized by previous tests,

- On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Call rfeConfigure with basic configuration and wait till configuration is complete with below CLK IO configuration. clkInOut in="enabled" out="enabled" reduce-driver-level="false"
   - rfeConfigure is success and returns no error.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
3. Call test SPI read to read module MCGEN and register XO_CTL and bitfield XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD and XO_CLKOUTDRVLVL_RED.
   - XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD should be TRUE/FALSE based on if clkInOut in,out is enabled else false. Similar for XO_CLKOUTDRVLVL_RED will be 0/1 based on reduce-driver-level.
4. Repeat the test for clkInOut in="enabled" out="enabled" reduce-driver-level="true".
   - rfeConfigure is success and returns no error.
5. Call test SPI read to read module MCGEN and register XO_CTL and bitfield XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD and XO_CLKOUTDRVLVL_RED.
   - XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD should be TRUE/FALSE based on if clkInOut in,out is enabled else false. Similar for XO_CLKOUTDRVLVL_RED will be 0/1 based on reduce-driver-level.
6. Repeat the test for clkInOut in="enabled" out="disabled" reduce-driver-level="true".
   - rfeConfigure is failure and returns error.
7. Call test SPI read to read module MCGEN and register XO_CTL and bitfield XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD and XO_CLKOUTDRVLVL_RED.
   - XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD should be TRUE/FALSE based on if clkInOut in,out is enabled else false. Similar for XO_CLKOUTDRVLVL_RED will be 0/1 based on reduce-driver-level.

4.99 rfe_tc_124_clkioRouted

The RFE Radar Use Case API shall provide a configuration parameters to configure CLK_IN and CLK_OUT pins.

4.99.1 Detailed Description

Table 457. rfe_tc_124_clkioRouted Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. Clk out pin connected to oscilloscope.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	1310709

4.99.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API if not already intialized by previous tests,
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Call rfeConfigure with basic configuration and wait till configuration is complete with below CLK IO configuration. clkInOut in="disabled" out="enabled" reduce-driver-level="false"
   - rfeConfigure is success and returns no error.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
3. Call test SPI read to read module MCGEN and register XO_CTL and bitfield XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD and XO_CLKOUTDRVLVL_RED.
   - XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD should be TRUE/FALSE based on if clkInOut in,out is enabled else false. Similar for XO_CLKOUTDRVLVL_RED will be 0/1 based on reduce-driver-level.
4. Repeat the test for clkInOut in="disabled" out="enabled" reduce-driver-level="true"
   - rfeConfigure is success and returns no error.
5. Call test SPI read to read module MCGEN and register XO_CTL and bitfield XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD and XO_CLKOUTDRVLVL_RED.
   - XO_CLKIN_PWD, XO_CLKOUTP_PWD and XO_CLKOUTN_PWD should be TRUE/FALSE based on if clkInOut in,out is enabled else false. Similar for XO_CLKOUTDRVLVL_RED will be 0/1 based on reduce-driver-level.
6. Check with 25pF capacitor and measure slope via oscilloscope for reduce-driver-level="false" and reduce-driver-level="true"
   - Assert that there is change in slope incase of reduce-driver-level="true".

4.100 rfe_tc_125_chirppllTestPin

The RFE SW shall provide an API to output the ChirpPll signal to a PIN.

4.100.1 Detailed Description

The RFE SW shall provide an API to output the ChirpPll signal to a PIN.

The API is for for internal validation and validation by customers.

Table 458. rfe_tc_125_chirppllTestPin Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up. FSW should be connected.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A

Table 458. rfe_tc_125_chirppllTestPin Specification*...continued*

Property	Description
Execution	Automated
Satisfied Requirements	1374624

4.100.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API if not already intialized by previous tests,
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. set rfe_testSetParam rfe_testParam_chirpPllTestPinEnable_e with value 0/1 to disable/enable the output of the divided (Fout/16) ChirpPLL VCO output.
   - expected no error.
3. set rfe_testSetParam for rfe_testParam_chirpPllTestPinEnable_e and value 3
   - expected rfe_error_api_invalidArgumentValue_e
4. Clear error
5. Call rfeConfigure with basic configuration and wait till configuration is complete
   - rfeConfigure is success and returns no error.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
6. Call test SPI read to read module ATB and register RF_BUF_1 and bitfield ENA_RF_BUF
   - ENA_RF_BUF should be TRUE if rfe_testParam_chirpPllTestPinEnable_e is enabled else false.
7. Call test SPI read to read module ATB and register RF_BUF_2 and bitfield CTRL_RFBUF_SPARE
   - CTRL_RFBUF_SPARE should be 0xC (bits 3:2 = output power programmability bits for chirpPLL_clk_p) if rfe_testParam_chirpPllTestPinEnable_e is enabled else 0.
8. Generate CW signal with rfe_continuousWaveTransmissionStart with 76/78/81 GHz (low/mid/high) .
   - Assert to check the Error parameter of the rfe_continuousWaveTransmissionStart() API reply to be '0'. i.e No Error
   - By connecting PLL test pin (pin available on WG v3 boards), check expected frequency at pin, Frequency = 4.75 GHz to 5.0625 GHz for 76 to 81 GHz.

4.101 rfe_tc_126_Calibrate

The RFE Radar Use Case API shall calibrate analog IPs based on customer configuration.

4.101.1 Detailed Description

The variation of tx power, tx phase rotation, rx gain, rx filter frequencies, center frequency in calibration is tested in other requirements.

In current requirement chirppll, rx lox2, loi rx, loi tx, tx buff2a, rfbist lox2 and rfbist ssb calibrations are considered.

Table 459. rfe_tc_126_Calibrate Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.

Table 459. rfe_tc_126_Calibrate Specification*...continued*

Property	Description
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	3243484

4.101.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case in a single chirp profile. Parameters : tcenter-frequency-kHz="76000000" effective-chirp-bandwidth-kHz="1000000" vco="1 GHz" slope="rising" and with all Tx enabled.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Please wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Select the calibration register bitfields In LOIF, LOIF_DRIVER_CTL-> GAIN_TX, LOIF_DRIVER_CTL-> GAIN_RX, In TX, P0_POWER-> LOX2_DAC_CTRL, PR_CTL -> DAC_IREF_PR, In RX, P0_IBIAS-> AMP80_IREF, In RXBIST, RXBIST_RF_PWR-> LOX2_CTRL, RXBIST_RF_PWR-> SSB_GAIN_BUFFER1, SSB_GAIN_BUFFER2, DAC_TRIM_BIAS In CHIRPPLL, P0_ANA_VCO_CTL1 -> TUNING, P0_ANA_VCO_CTL1 -> CURRENT, using spi read.
   - Assert that register read out are different from reset values of the register.
6. Repeat the test for different center frequency Eg., 80GHz.
   - Assert that register read out are different from previous calibrated values of the register.

4.102 rfe_tc_127_recalibrate

The RFE Radar Use Case API shall re-calibrate analog IPs periodically to compensate for temperature over time.

4.102.1 Detailed Description

The variation of tx power (1296375), tx phase rotation (1296376) in calibration is tested in other requirements.

In current requirement chirppll, rx lox2, loi rx, loi tx, tx buff2a, rfbist lox2 and rfbist ssb calibrations are considered.

Table 460. rfe_tc_127_recalibrate Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables.

Table 460. rfe_tc_127_recalibrate Specification*...continued*

Property	Description
	RFE M7, APP M7 and A53 cores should up. RFE proxy to capture the adc data. Matlab to process the captured the adc data.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	Radar Front End(RFE) IC plus supporting device like IF_V1 board.
Execution	Automated
Satisfied Requirements	3243484

4.102.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case in a single chirp profile. Parameters : center-frequency-kHz="76500000" effective-chirp-bandwidth-kHz="1000000" vco="1 GHz" slope="rising" and with all Tx enabled. In recalibration, enable profile independent calibration for sequence 0 In recalibration, enable profile dependent calibration for sequence 0, profile 0.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Config blob contents with length and No Dynamic table data.
   - Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error
4. Wait with timeout till rfe_getState is busy till state transition is done to 'CONFIGURED'.
   - On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Select the calibration register bitfields In LOIF, LOIF_DRIVER_CTL-> GAIN_TX, LOIF_DRIVER_CTL-> GAIN_RX, In TX, P0_POWER-> LOX2_DAC_CTRL, PR_CTL -> DAC_IREF_PR, In RX, P0_IBIAS-> AMP80_IREF, In RXBIST, RXBIST_RF_PWR-> LOX2_CTRL, RXBIST_RF_PWR-> SSB_GAIN_BUFFER1, SSB_GAIN_BUFFER2, DAC_TRIM_BIAS In CHIRPPLL, P0_ANA_VCO_CTL1 -> TUNING, P0_ANA_VCO_CTL1 -> CURRENT, using spi read.
   - Assert that register read out are different from reset values of the register.
6. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
   - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
7. Collect the adc data
   - Assert that ADC data is non-zero.
   - Assert that processed data with matlab and check if target is correctly detected (e.g. distance, velocity) for the tx with which antenna is connected.
8. Select the calibration register bitfields In LOIF, LOIF_DRIVER_CTL-> GAIN_TX, LOIF_DRIVER_CTL-> GAIN_RX, In TX, P0_POWER-> LOX2_DAC_CTRL, PR_CTL -> DAC_IREF_PR, In RX, P0_IBIAS-> AMP80_IREF, In RXBIST, RXBIST_RF_PWR-> LOX2_CTRL, RXBIST_RF_PWR-> SSB_GAIN_BUFFER1,

SSB_GAIN_BUFFER2, DAC_TRIM_BIAS In CHIRPPLL, P0_ANA_VCO_CTL1 -> TUNING, P0_ANA_VCO_CTL1 -> CURRENT, using spi read.
- Assert that register read out are different from reset values of the register.

9. Repeat the test for different center frequency Eg., 80GHz.
   - Assert that register read out are different from previous calibrated values of the register.
10. Repeat the test from configuration, where in recalibration only profile independent is for sequence 0 is enabled.
    - Assert that register read out are different between configuration and radar cycle calibrated values of the register except for chirppll VCO.
11. Repeat the test from configuration, where in recalibration only profile dependent is for sequence 0 and profile 0 is enabled.
    - Assert that register read out are different between configuration and radar cycle calibrated values of the register only for chirppll VCO. Other registers values shouldn't be changed between configuration and radar cycle.
12. Repeat the test from configuration, where there is no recalibration is enabled.
    - Assert that register read out for calibrated values of the registers values shouldn't be changed between configuration and radar cycle.
13. Destroy calibration via fault injection (write wrong values to the calibration registers) within the radar cycle. Recalibration should fix the calibration and correctly detect the target again, "destroyed calibration" should not detect the target.
14. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0, recalibration for profile 0 in sequence 0
    - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
    - Assert to check the current RFE state to be 'CONFIGURED' using rfe_getState() API.
15. Collect the adc data
    - Assert that ADC data is non-zero.
    - Assert that processed data with matlab and check if target is correctly detected (e.g. distance, velocity) for the tx with which antenna is connected.

4.103 rfe_tc_128_rxAdcCalAtbTrim

The RFE SW shall trim the ATB-ADC and calibrate the RX-ADC independently from customer config.

4.103.1 Detailed Description

Table 461. rfe_tc_128_rxAdcCalAtbTrim Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated

Table 461. rfe_tc_128_rxAdcCalAtbTrim Specification*...continued*

Property	Description
Satisfied Requirements	1301541

4.103.2 Test Procedure

Steps:

1. Power on the board and initialize the Radar Front End(RFE) using rfe_sync() API.
   - Assert to check no error. This checks otp distributed and atb adc is trimmed.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
   - Assert that the memory read is 0x33fffff4.ie. CLK_PLL_STATUS is 0xC0DE0001.
2. Call test SPI read to read module RX ADC and register LF_LPF_CTL and bitfield CLPF_I and CLPF_Q.
   - CLPF_I and CLPF_Q values should be different than reset values.

4.104 rfe_tc_129_8profiles

The RFE SW shall support control of 8 profiles for each IP and bist functionality.

4.104.1 Detailed Description

Table 462. rfe_tc_129_8profiles Specification

Property	Description
Pre	Radar Front End(RFE) IC should be connected and powered up with necessary connectors/cables. RFE M7, APP M7 and A53 cores should up.
Post	RFE FW should be in non-error state and should be able to perform and execute new tests.
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280276

4.104.2 Test Procedure

Steps:

1. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
2. Create a configuration with the RFE static use case by programming for 8 distinct profiles with bist enabled for every radar cycle and bist fusa flags are not masked.
3. Configure the RFE using the rfe_configure() API using the blob contents of the above configuration. Parameters : Configblob contents with length and No Dynamic table data.

- Assert to check the Error parameter of the rfe_configure() API reply to be '0'. i.e No Error

4. On successful RFE configuration check for the current RFE state to be 'CONFIGURED'.
5. Call the rfe_getBistZeroHourReferenceData() api, to get the bist zero hour reference data.
   - Assert to check the Error parameter of the rfe_getBistZeroHourReferenceData() API reply to be '0'. i.e No Error
6. Update the bist using rfe_update api with the bist zero hour reference data received.
7. Trigger/Start the Radar Cycle using rfe_radarCycleStart() API. Parameters : cycle_count=1, is_scheduled=False, start_time=0
   - Assert to check the Error parameter of the rfe_radarCycleStart() API reply to be '0'. i.e No Error
8. Capture the Raw ADC data.
   - Assert that ADC data to be non-zero and size (Samples * Chirps * 4 * 2) bytes e.g. 512 samples * 4096 chirps * 4 ADC's * 2(bytes) = 16,777,216 bytes.

4.105 rfe_tc_3001_safetyApiConfig

The RFE SW shall provide RFE Safety configuration, via RFE Safety API

4.105.1 Detailed Description

The RFE SW shall provide RFE Safety configuration, via RFE Safety API

- Mask or unmask the R1 and R2 faults.

- Program threshold for Monitors - PPD, ADC Clipping Counter, RX Saturation Detectors

- BIST: Zero-hour data and its threshold (will be covered in rfe_tc_3004_invalidParameter, rfe_tc_3009_zeroHourReferenceForRxPhaseDiff, rfe_tc_3010_zeroHourReferenceForRxGainDiff, rfe_tc_3013_zeroHourReferenceForTxPhaseDiff)

- Fault propagation threshold to treat R1 faults as R2 fault.

Table 463. rfe_tc_3001_safetyApiConfig Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1381846

4.105.2 Test Procedure

Steps:

1. Make sure that RFE is synced and initialized or in configured state.
2. Configure RFE with a rfeConfig that does not mask any FuSa fault.
3. Wait until configuration is done.
   - Make sure no unexpected RFE_ERROR flags were raised so far.

4. Loop through all FuSa fault mask indices and all available FuSa fault mask bits:
5. Enable the FuSa fault mask via rfe_update* API.
   - Check that no RFE_ERROR flag was raised.
   - Make sure that RFE is still in state configured.
6. Disable the FuSa fault and repeat the two checks above.
7. Start three radar cycles.
8. Wait until RFE is in radarCycleIdle state.
9. Enable FuSa fault mask via rfe_update* API.
   - Check that no RFE_ERROR flag was raised.
   - Check that RFE is still in state radarCycleIdle and has completed only one cycle so far (using rfe_getRadarCycleCount() API).
10. Wait until radarCycle is in radarCycleIdle and the second radar cycle is finished.
11. Disable FuSa fault mask via rfe_update* API.
    - Check that no RFE_ERROR flag was raised.
12. Wait until RFE is in state configured.
13. Use update API to set an invalid FuSaFault mask.
    - Check that this triggers a rfe_error_api_invalidConfigurationParameterValue_e.
14. Clear the error.
15. Set the fault propagation threshold (thresholdValueToPromoteR1Faults) to 0xFF via rfe_update* API.
    - Check for error flags.
16. Set the fault propagation threshold (thresholdValueToPromoteR1Faults) to all values in between via rfe_update* API.
    - In each case, check for error flags.
17. Set the fault propagation threshold (thresholdValueToPromoteR1Faults) to 0x0 via rfe_update* API.
    - Check for error flags.
18. Use update API to set an a threshold that is out of range.
    - Check that this triggers a rfe_error_api_invalidConfigurationParameterValue_e.
19. Clear the error.
20. Start radarCycles and repeat the steps above while RFE is in state radarCycleIdle.
21. In configured state:
22. Update txPpdThreshold iteratively from its minimum up to its maximum value. Use a stepsize that ensures that no more then 20 parameter values within the range were applied.
    - Check that no error flags were raised and RFE remains in state configured.
23. Update txPpdThreshold to minimum-1 and maximum+1.
    - Check that ERROR flag rfe_error_api_invalidConfigurationParameterValue_e is raised.
24. Then clear the RFE errors.
25. Start radarCycles and repeat the steps above while RFE is in state radarCycleIdle.
26. Repeat the steps above for the following parameters as well:
27. adcClippingCountResetEveryChirpSequence
28. adcClippingCountLimit, but skip the check for minimum-1
29. rxSaturationThreshold, but skip the check for minimum-1
30. rxSaturationCountLimit, but skip the check for minimum-1

4.106 rfe_tc_3002_heartBeatPulse

The RFE SW shall send HB(Heart Beat) once per radar cycle.

4.106.1 Detailed Description

The RFE SW shall send HB(Heart Beat) once per radar cycle.

The RFE SW shall provide setting of FHTI time for RFE SW, via RFE Safety API

Table 464. rfe_tc_3002_heartBeatPulse Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280394, 1299512,

4.106.2 Test Procedure

Steps:

1. Enable Interrupt and register ISR for Heartbeat Interrupt by calling interrupts_register_ISR_enable/ interrupts_cm7_install with edge trigger option to count heartbeat pulse
2. Initialize the Radar Front End(RFE) using rfe_sync() API.
   - On successful RFE initialization, Assert to check the current RFE state to be 'INITIALIZED' using rfe_getState() API.
3. Clear HeartBeat count.
4. set rfe_testSetParam with value 1 to trigger Heartbeat pulse from rfeM7
   - expected heartbeat count is set to one in ISR.
5. set rfe_testSetParam with value 0
   - expected rfe_error_api_invalidArgumentValue_e
6. Clear error
7. Clear HeartBeat count.
8. Call rfeConfigure with basic configuration and wait till configuration is complete
   - rfeConfigure is success
9. Call getBistZeroHourReferenceData and update the zero hour reference data
   - getBistZeroHourReferenceData is success
10. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush to update TX Phase diff, RX Phase diff and RX gain diff received from the getBistZeroHourReferenceData
11. Call rfe_radarCycleStart with 10 radar cycle
    - Once radar Cycle is complete total HeartBeat count matches with radar cycle count 10
    - Periodicity Heartbeat pulse period should match with radar cycle period 50ms(Default radarCycleDuration).
12. Update radarCycleDuration to different value min 400000(10ms) and max 20000000 (500ms) in steps of 10ms with 1 radar cycle by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush
13. Call rfe_radarCycleStart with 1 radar cycle
    - Once radar Cycle is complete total HeartBeat count matches with radar cycle count 1
    - Periodicity Heartbeat pulse period should match with configured radarCycleDuration.

4.107 rfe_tc_3003_toggleErrorN

This test case is to check the toggling of the ERROR_N pin, via RFE Safety API

4.107.1 Detailed Description

This test case is to check the toggling of the ERROR_N pin, via RFE Safety API On successfully completing safety checks, the RFE SW shall clear all the fault from RFE FCCU.

Table 465. rfe_tc_3003_toggleErrorN Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	
Test Type	Functional
Test Technique	BlackBox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280570, 1311256,

4.107.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_Sync().
2. Initialize RTS Fccu.
3. Clear ErrorN Fault.
   - Check ErrorN Fault status is 0 .
4. Call rfe_testSetParam with parameter rfe_testParam_assertErrorNSignal_e and value 1 to set ErrorN.
   - Check ErrorN Fault status is 1 .
5. Call rfe_testSetParam with parameter rfe_testParam_deAssertErrorNSignal_e and value 1 to set ErrorN.
   - Check ErrorN Fault status is 0 .
6. Call rfe_testSetParam with parameter rfe_testParam_assertErrorNSignal_e and value 1 to set ErrorN.
   - Check ErrorN Fault status is 1 .
7. Call rfe_testSetParam with parameter rfe_testParam_deAssertErrorNSignal_e and value 1 to set ErrorN.
   - Check ErrorN Fault status is 0 .
8. Call rfe_testSetParam with parameter rfe_testParam_assertErrorNSignal_e and value 0 to set ErrorN.
   - expected rfe_testSetParam return rfe_error_api_invalidArgumentValue_e
9. Clear Error
10. Call rfe_testSetParam with parameter rfe_testParam_deAssertErrorNSignal_e and value 0 to set ErrorN.
    - expected rfe_testSetParam return rfe_error_api_invalidArgumentValue_e
11. Call rfe_configure wih default configuration
    - rfe_configure executed successfully
12. Call rfe_testSetParam with parameter rfe_testParam_assertErrorNSignal_e and value 1 to set ErrorN.
    - expected rfe_testSetParam return rfe_error_api_operationNotAllowed_e
13. Clear Error

4.108 rfe_tc_3004_invalidParameter

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.108.1 Detailed Description

This testcase does invalid parameter test for all the parameter for getBistZeroHourReferenceData API

Table 466. rfe_tc_3004_invalidParameter Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658, 1381846

4.108.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
   - Check rfeConfigure is executed successfully
3. Call getBistZeroHourReferenceData with pTxReferenceData as NULL
4. Check rfe_error_api_invalidArgumentValue_e error is reported
5. Call getBistZeroHourReferenceData with pRxReferenceData as NULL
6. Clear Error
7. Call getBistZeroHourReferenceData
8. Check rfe_error_api_invalidArgumentValue_e error is reported
9. Clear Error
10. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration outside the accepted range for frequency-in-kHz bist configuration parameter
11. Check rfe_updatePush is executed with error
12. Check rfe_error_api_invalidArgumentValue_e error is reported
13. Clear Error
14. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration outside the accepted range for RX phase-diff-threshold-tolerance bist configuration parameter
15. Check rfe_updatePush is executed with error
16. Check rfe_error_api_invalidArgumentValue_e error is reported
17. Clear Error
18. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration RX gain-diff-threshold-tolerance bist configuration parameter outside the accepted range
19. Check rfe_updatePush is executed with error

20. Check rfe_error_api_invalidArgumentValue_e error is reported
21. Clear Error
22. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-rx12 Phase Diff bist configuration parameter outside the accepted range
23. Check rfe_updatePush is executed with error
24. Check rfe_error_api_invalidArgumentValue_e error is reported
25. Clear Error
26. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-rx13 Phase Diff bist configuration parameter outside the accepted range
27. Check rfe_updatePush is executed with error
28. Check rfe_error_api_invalidArgumentValue_e error is reported
29. Clear Error
30. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-rx14 Phase Diff bist configuration parameter outside the accepted range
31. Check rfe_updatePush is executed with error
32. Check rfe_error_api_invalidArgumentValue_e error is reported
33. Clear Error
34. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-rx13 Gain Diff bist configuration parameter outside the accepted range
35. Check rfe_updatePush is executed with error
36. Check rfe_error_api_invalidArgumentValue_e error is reported
37. Clear Error
38. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-rx12 Gain Diff bist configuration parameter outside the accepted range
39. Check rfe_updatePush is executed with error
40. Check rfe_error_api_invalidArgumentValue_e error is reported
41. Clear Error
42. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-rx14 Gain Diff bist configuration parameter outside the accepted range
43. Check rfe_updatePush is executed with error
44. Check rfe_error_api_invalidArgumentValue_e error is reported
45. Clear Error
46. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with Tx phase-step-threshold-tolerance bist configuration parameter outside the accepted range
47. Check rfe_updatePush is executed with error
48. Check rfe_error_api_invalidArgumentValue_e error is reported
49. Clear Error
50. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with Tx phase-diff-threshold-tolerance bist configuration parameter outside the accepted range
51. Check rfe_updatePush is executed with error
52. Check rfe_error_api_invalidArgumentValue_e error is reported
53. Clear Error
54. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-tx12 Phase Diff bist configuration parameter outside the accepted range
55. Check rfe_updatePush is executed with error
56. Check rfe_error_api_invalidArgumentValue_e error is reported
57. Clear Error

58. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-tx23 Phase Diff bist configuration parameter outside the accepted range
59. Check rfe_updatePush is executed with error
60. Check rfe_error_api_invalidArgumentValue_e error is reported
61. Clear Error
62. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with zero-hr-ref-tx34 Phase Diff bist configuration parameter outside the accepted range
63. Check rfe_updatePush is executed with error
64. Check rfe_error_api_invalidArgumentValue_e error is reported
65. Clear Error
66. rfe_updateBegin, rfe_updateParam, rfe_updatePush basic configuration with txPowerlevelForBist bist configuration parameter outside the accepted range
67. Check rfe_updatePush is executed with error
68. Check rfe_error_api_invalidArgumentValue_e error is reported
69. Clear Error
70. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
71. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
72. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
73. Run All the above scenario by updating the config parameter sweep via rfe_configure()

4.109 rfe_tc_3005_frequencyForBist

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.109.1 Detailed Description

This testcase does all possible configuration frequencyForBist parameter getBistZeroHourReferenceData API

Table 467. rfe_tc_3005_frequencyForBist Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658

4.109.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with 76.5GHz to 82GHz in steps of 100KHz for frequency-in-kHzbist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with 76.5GHz to 82GHz in steps of 100KHz for frequency-in-kHzbist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Run All the above scenario by updating the config parameter sweep via rfe_configure()
16. Run All the above scenario with injectTestToneBeforeLna true

4.110 rfe_tc_3006_rxPhaseDiffThresholdTolerance

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.110.1 Detailed Description

This testcase does all possible configuration for rxPhaseDiffThresholdTolerance parameter getBistZeroHourReferenceData API

Table 468. rfe_tc_3006_rxPhaseDiffThresholdTolerance Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A

Table 468. rfe_tc_3006_rxPhaseDiffThresholdTolerance Specification*...continued*

Property	Description
Execution	Automated
Satisfied Requirements	3069658

4.110.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with 0 to 400 in steps of 0.1dBm for RX phase-diff-threshold-tolerance bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with 0 to 400 in steps of 0.1dBm for RX phase-diff-threshold-tolerance bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Run All the above scenario by updating the config parameter sweep via rfe_configure()
16. Run All the above scenario with injectTestToneBeforeLna true

4.111 rfe_tc_3007_rxGainDiffThresholdTolerance

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.111.1 Detailed Description

This testcase does all possible configuration for rxGainDiffThresholdTolerance parameter getBistZeroHourReferenceData API

Table 469. rfe_tc_3007_rxGainDiffThresholdTolerance Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658

4.111.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with 0 to 400 in steps pof 0.1dBm for RX gain-diff-threshold-tolerance bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with 0 to 400 in steps pof 0.1dBm for RX gain-diff-threshold-tolerance bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Run All the above scenario by updating the config parameter sweep via rfe_configure()
16. Run All the above scenario with injectTestToneBeforeLna true

4.112 rfe_tc_3008_injectTestToneBeforeLna

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.112.1 Detailed Description

This testcase does all possible configuration for injectTestToneBeforeLna parameter getBistZeroHourReferenceData API

Table 470. rfe_tc_3008_injectTestToneBeforeLna Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658

4.112.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with before-lna as false
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Run All the above scenario by updating the config parameter sweep via rfe_configure()
12. Run All the above scenario with injectTestToneBeforeLna true

4.113 rfe_tc_3009_zeroHourReferenceForRxPhaseDiff

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.113.1 Detailed Description

This testcase does all possible configuration for zeroHourReferenceForRxPhaseDiff parameter getBistZeroHourReferenceData API

Table 471. rfe_tc_3009_zeroHourReferenceForRxPhaseDiff Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658, 1381846

4.113.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with -1800 till 1800 in steps of 0.1 degree for zero-hr-ref-rx12 Phase Diff bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with -1800 till 1800 in steps of 0.1 degree for zero-hr-ref-rx13 Phase Diff bist configuration parameter
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff

13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Call rfeConfigure basic configuration with -1800 till 1800 in steps of 0.1 degree for zero-hr-ref-rx14 Phase Diff bist configuration parameter
    - Check rfeConfigure is executed successfully
16. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
17. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
18. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
19. Call rfeConfigure basic configuration with -1800 till 1800 in steps of 0.1 degree for zero-hr-ref-rx12 Phase Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
20. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
21. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
22. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
23. Call rfeConfigure basic configuration with -1800 till 1800 in steps of 0.1 degree for zero-hr-ref-rx13 Phase Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
24. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
25. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
26. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
27. Call rfeConfigure basic configuration with -1800 till 1800 in steps of 0.1 degree for zero-hr-ref-rx14 Phase Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
28. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
29. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
30. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
31. Run All the above scenario by updating the config parameter sweep via rfe_configure()
32. Run All the above scenario with injectTestToneBeforeLna true

4.114 rfe_tc_3010_zeroHourReferenceForRxGainDiff

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.114.1 Detailed Description

This testcase does all possible configuration for zeroHourReferenceForRxGainDiff parameter getBistZeroHourReferenceData API

Table 472. rfe_tc_3010_zeroHourReferenceForRxGainDiff Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658, 1381846

4.114.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call rfeConfigure basic configuration with -400 till 400 in steps of 0.1dB for zero-hr-ref-rx12 Gain Diff bist configuration parameter
   - Check rfeConfigure is executed successfully
5. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
6. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
7. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
8. Call rfeConfigure basic configuration with -400 till 400 in steps of 0.1dB for zero-hr-ref-rx13 Gain Diff bist configuration parameter
   - Check rfeConfigure is executed successfully
9. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
10. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
11. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
12. Call rfeConfigure basic configuration with -400 till 400 in steps of 0.1dB for zero-hr-ref-rx14 Gain Diff bist configuration parameter
   - Check rfeConfigure is executed successfully
13. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
14. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
15. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
16. Call rfeConfigure basic configuration with -400 till 400 in steps of 0.1dB for zero-hr-ref-rx12 Gain Diff bist configuration parameter with injectTestToneBeforeLna false
   - Check rfeConfigure is executed successfully
17. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff

18. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
19. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
20. Call rfeConfigure basic configuration with -400 till 400 in steps of 0.1dB for zero-hr-ref-rx13 Gain Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfullywith injectTestToneBeforeLna false
21. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
22. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
23. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
24. Call rfeConfigure basic configuration with -400 till 400 in steps of 0.1dB for zero-hr-ref-rx14 Gain Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
25. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
26. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
27. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
28. Run All the above scenario by updating the config parameter sweep via rfe_configure()
29. Run All the above scenario with injectTestToneBeforeLna true

4.115 rfe_tc_3011_txPhaseDiffThresholdTolerance

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.115.1 Detailed Description

This testcase does all possible configuration for txPhaseDiffThresholdTolerance parameter getBistZeroHourReferenceData API

Table 473. rfe_tc_3011_txPhaseDiffThresholdTolerance Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658

4.115.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with 0 to 400 in steps of 0.1 degree for Tx phase-diff-threshold-tolerance bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with 0 to 400 in steps of 0.1 degree for Tx phase-diff-threshold-tolerance bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Run All the above scenario by updating the config parameter sweep via rfe_configure()
16. Run All the above scenario with injectTestToneBeforeLna true

4.116 rfe_tc_3012_txPhaseStepThresholdTolerance

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.116.1 Detailed Description

This testcase does all possible configuration for txPhaseStepThresholdTolerance parameter getBistZeroHourReferenceData API

Table 474. rfe_tc_3012_txPhaseStepThresholdTolerance Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A

Table 474. rfe_tc_3012_txPhaseStepThresholdTolerance Specification*...continued*

Property	Description
Execution	Automated
Satisfied Requirements	3069658

4.116.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with 0 to 400 in steps of 0.1 degree for Tx phase-step-threshold-tolerance bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with 0 to 400 in steps of 0.1 degree for Tx phase-step-threshold-tolerance bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Run All the above scenario by updating the config parameter sweep via rfe_configure()
16. Run All the above scenario with injectTestToneBeforeLna true

4.117 rfe_tc_3013_zeroHourReferenceForTxPhaseDiff

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.117.1 Detailed Description

This testcase does all possible configuration for zeroHourReferenceForTxPhaseDiff parameter getBistZeroHourReferenceData API

Table 475. rfe_tc_3013_zeroHourReferenceForTxPhaseDiff Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658, 1381846

4.117.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with -1800 to 1800 in steps of 0.1 degree for zero-hr-ref-tx12 Phase Diff bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with -1800 to 1800 in steps of 0.1 degree for zero-hr-ref-tx23 Phase Diff bist configuration parameter
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Call rfeConfigure basic configuration with -1800 to 1800 in steps of 0.1 degree for zero-hr-ref-tx34 Phase Diff bist configuration parameter
    - Check rfeConfigure is executed successfully
16. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
17. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData

18. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
19. Call rfeConfigure basic configuration with -1800 to 1800 in steps of 0.1 degree for zero-hr-ref-tx12 Phase Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
20. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
21. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
22. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
23. Call rfeConfigure basic configuration with -1800 to 1800 in steps of 0.1 degree for zero-hr-ref-tx23 Phase Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
24. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
25. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
26. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
27. Call rfeConfigure basic configuration with -1800 to 1800 in steps of 0.1 degree for zero-hr-ref-tx34 Phase Diff bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
28. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
29. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
30. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
31. Run All the above scenario by updating the config parameter sweep via rfe_configure()
32. Run All the above scenario with injectTestToneBeforeLna true

4.118 rfe_tc_3014_txPowerLevelForBist

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.118.1 Detailed Description

This testcase does all possible configuration for txPowerLevelForBist parameter getBistZeroHourReferenceData API

Table 476. rfe_tc_3014_txPowerLevelForBist Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A

Table 476. rfe_tc_3014_txPowerLevelForBist Specification*...continued*

Property	Description
Execution	Automated
Satisfied Requirements	3069658

4.118.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration with -90 to 150 in steps of 10(1dBm) for txPowerlevelForBist bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration with -90 to 150 in steps of 10(1dBm) for txPowerlevelForBist bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Run All the above scenario by updating the config parameter sweep via rfe_configure()
16. Run All the above scenario with injectTestToneBeforeLna true

4.119 rfe_tc_3015_txSelectForTxBist

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.119.1 Detailed Description

This testcase does all possible configuration for txSelectForTxBist parameter getBistZeroHourReferenceData API

Table 477. rfe_tc_3015_txSelectForTxBist Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658

4.119.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call rfeConfigure basic configuration by enabling all TX1/2/3/4 individually bist configuration parameter
   - Check rfeConfigure is executed successfully
8. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
9. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
10. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
11. Call rfeConfigure basic configuration by enabling all TX1/2/3/4 individually bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
12. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
13. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
14. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
15. Call rfeConfigure basic configuration by enabling TX1 individually bist configuration parameter
    - Check rfeConfigure is executed successfully
16. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX2/3/4 is 0
17. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData

18. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
19. Call rfeConfigure basic configuration by enabling TX2 individually bist configuration parameter
    - Check rfeConfigure is executed successfully
20. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/3/4 is 0
21. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
22. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
23. Call rfeConfigure basic configuration by enabling TX3 individually bist configuration parameter
    - Check rfeConfigure is executed successfully
24. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/2/4 is 0
25. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
26. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
27. Call rfeConfigure basic configuration by enabling TX4 individually bist configuration parameter
    - Check rfeConfigure is executed successfully
28. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/2/3 is 0
29. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
30. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
31. Call rfeConfigure basic configuration by enabling TX1 individually bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
32. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX2/3/4 is 0
33. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
34. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
35. Call rfeConfigure basic configuration by enabling TX2 individually bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
36. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/3/4 is 0
37. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
38. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
39. Call rfeConfigure basic configuration by enabling TX3 individually bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
40. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/2/4 is 0

41. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
42. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
43. Call rfeConfigure basic configuration by enabling TX4 individually bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
44. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/2/3 is 0
45. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
46. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
47. Call rfeConfigure basic configuration by disabling all TX bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
48. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/2/3/4 is 0
49. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
50. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
51. Call rfeConfigure basic configuration by disabling all TX bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
52. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/2/3/4 is 0
53. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
54. Call rfe_radarCycleStart with 10 radar cycle
55. Bist should fail during radar cycle
56. Call rfeConfigure basic configuration by disabling all TX bist configuration parameter with injectTestToneBeforeLna false
    - Check rfeConfigure is executed successfully
57. Call getBistZeroHourReferenceData to get bist zero hour rx phase diff, gain diff and tx phase diff
    - Check zero hour data for TX1/2/3/4 is 0
58. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
59. Enable all TX1/2/3/4 for txBist by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
60. Call rfe_radarCycleStart with 10 radar cycle
61. Bist should fail during radar cycle and move to fusa fault state
62. Run All the above scenario by updating the config parameter sweep via rfe_configure()
63. Run All the above scenario with injectTestToneBeforeLna true

4.120 rfe_tc_3016_getZeroHourData_loop

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.120.1 Detailed Description

This testcase does all possible configuration for txSelectForTxBist parameter getBistZeroHourReferenceData API

Table 478. rfe_tc_3016_getZeroHourData_loop Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658

4.120.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
3. Unmask FuSa Faults for BIST measurement
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
7. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff in a loop of 100 times
8. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
   - Check the value measured are within the range.
9. Call rfe_radarCycleStart with 10 radar cycle
   - All requested radar cycle to run without fault.
10. Call getBistZeroHourReferenceData in a loop of 100
11. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
12. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
    - Check the value measured are within the range.
13. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
14. Call getBistZeroHourReferenceData in a loop of 100 followed for rfe_radarCycleStart

15. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff with injectTestToneBeforeLna false
16. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
    - Check the value measured are within the range.
17. Call rfe_radarCycleStart with 10 radar cycle
    - All requested radar cycle to run without fault.
18. Run All the above scenario by updating the config parameter sweep via rfe_configure()

4.121 rfe_tc_3017_getZeroHourData_differentState

The RFE SW shall provide an API to trigger and retrieve the Zero-Hour data measurement performed by RFE SW

4.121.1 Detailed Description

This testcase does all possible configuration for txSelectForTxBist parameter getBistZeroHourReferenceData API

Table 479. rfe_tc_3017_getZeroHourData_differentState Specification

Property	Description
Pre	N/A
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069658

4.121.2 Test Procedure

Steps:

1. Check RFE initialization with valid value.
2. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff in Initialized state
   - getBistZeroHourReferenceData execution failed with rfe_error_api_operationNotAllowed_e
3. Call rfeConfigure basic configuration with default bist configuration and wait till configuration is complete
   - Check rfeConfigure is executed successfully
4. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff
5. Update ZeroHourReferenceData by calling rfe_updateBegin, rfe_updateParam, rfe_updatePush received from the getBistZeroHourReferenceData
6. Call rfe_radarCycleStart with 10 radar cycle
7. When system is in radarCycle state call getBistZeroHourReferenceData
   - getBistZeroHourReferenceData execution failed with rfe_error_api_operationNotAllowed_e
8. Call rfe_testSetParam with parameter rfe_testParam_assertHeartBeatSignal_e and value 1 to set ErrorN.

- Check ErrorN Fault status is 1 .

9. Call rfe_radarCycleStart with 10 radar cycle
   - Check state moved to fusaFault state.
10. Call getBistZeroHourReferenceData to get bist zero hour reference data for rx phase diff, gain diff and tx phase diff in Initialized state
    - getBistZeroHourReferenceData execution failed with rfe_error_api_operationNotAllowed_e

4.122 rfe_tc_3018_sm3Tx1BallBreak

Testcase to perform ball break detection tx1 r1 fault handling and r2 fault promotion for sm3

4.122.1 Detailed Description

On successfully completing safety checks, the RFE SW shall clear all the fault from RFE FCCU.

Table 480. rfe_tc_3018_sm3Tx1BallBreak Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311256,

4.122.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob

10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active

- Fault is not active for unmasked SM

34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail.

4.123 rfe_tc_3020_sm5Tx3BallBreak

Testcase to perform ball break detection tx3 r1 fault handling and r2 fault promotion for sm5

4.123.1 Detailed Description

On successfully completing safety checks, the RFE SW shall clear all the fault from RFE FCCU.

Table 481. rfe_tc_3020_sm5Tx3BallBreak Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311256,

4.123.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM

30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail.

4.124 rfe_tc_3021_sm6Tx4BallBreak

Testcase to perform ball break detection tx4 r1 fault handling and r2 fault promotion for sm6

4.124.1 Detailed Description

On successfully completing safety checks, the RFE SW shall clear all the fault from RFE FCCU.

Table 482. rfe_tc_3021_sm6Tx4BallBreak Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311256,

4.124.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail.

4.125 rfe_tc_3022_sm7Tx1PPDLevelDetection

Testcase to perform tx1 ppd level det r1 fault handling and r1 to r2 fault promotion for sm7

4.125.1 Detailed Description

When configured the RFE SW shall perform recovery for TX power detector faults RFE SW shall notify radar application about communication fault via FCCU

Table 483. rfe_tc_3022_sm7Tx1PPDLevelDetection Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A

Table 483. rfe_tc_3022_sm7Tx1PPDLevelDetection Specification*...continued*

Property	Description
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311264, 1311213,

4.125.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active

- No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.3 - PPD Level Detector - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553858

4.126 rfe_tc_3023_sm8Tx2PPDLevelDetection

Testcase to perform tx2 ppd level det r1 fault handling and r1 to r2 fault promotion for sm8

4.126.1 Detailed Description

When configured the RFE SW shall perform recovery for TX power detector faults

Table 484. rfe_tc_3023_sm8Tx2PPDLevelDetection Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311264, 1311213,

4.126.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active

16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active

- Fault is active for unmasked SM

37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
- Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
- rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
- ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.3 - PPD Level Detector - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553858

4.127 rfe_tc_3024_sm9Tx3PPDLevelDetection

Testcase to perform tx3 ppd level det r1 fault handling and r1 to r2 fault promotion for sm9

4.127.1 Detailed Description

When configured the RFE SW shall perform recovery for TX power detector faults

Table 485. rfe_tc_3024_sm9Tx3PPDLevelDetection Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311264, 1311213,

4.127.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
- RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
- RTS FCCU Init success
3. Clear RTS FCCU faults
- ErrorN is not asserted
4. call rfe_configure with default settings
- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
- rfe_configure() is success
6. update bistZeroHourReferenceData
- update bistZeroHourReferenceData is success
7. call rfe_configure with default settings

- rfe_configure() is success

8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count

- No Faults is promoted to R2
- Radar cycle requested are complete

33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.3 - PPD Level Detector - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553858

4.128 rfe_tc_3025_sm10Tx4PPDLevelDetection

Testcase to perform tx4 ppd level det r1 fault handling and r1 to r2 fault promotion for sm10

4.128.1 Detailed Description

When configured the RFE SW shall perform recovery for TX power detector faults

Table 486. rfe_tc_3025_sm10Tx4PPDLevelDetection Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311264, 1311213,

4.128.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization

- RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count

- No Faults is promoted to R2
- Radar cycle requested are complete

29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.3 - PPD Level Detector - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553858

4.129 rfe_tc_3026_sm11TxPhaseDifferenceTx1Tx2R1FaultHandling

Testcase to perform tx bist phase difference tx1tx2 r1 fault handling for sm11

4.129.1 Detailed Description

The R1 fault counter shall be cleared (reset to zero) when R1 faults do not appear in consecutive radar cycles.

The RFE SW shall notify the host upon successful fault recovery The RFE SW shall attempt recovery of R1 faults only when the fault count is lesser than the threshold count programmed.

Table 487. rfe_tc_3026_sm11TxPhaseDifferenceTx1Tx2R1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox

Table 487. rfe_tc_3026_sm11TxPhaseDifferenceTx1Tx2R1FaultHandling Specification*...continued*

Property	Description
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3121419, 1377107, 2899054

4.129.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active

- No Faults count is incremented

23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.130 rfe_tc_3027_sm11TxPhaseDifferenceTx1Tx2R2FaultPromotion

Testcase to perform tx bist phase difference tx1tx2 r1 to r2 fault promotion for sm11

4.130.1 Detailed Description

The RFE SW shall provide the count of R1 faults via API.

The RFE SW shall provide fault status and fault statistics readout via API.

The RFE FW shall provide an RFE Safety API to program the threshold count for R1 faults after which it shall be treated as R2 fault.

Table 488. rfe_tc_3027_sm11TxPhaseDifferenceTx1Tx2R2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3069657 , 3168361 , 2899048 ,

4.130.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete

19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted

40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.131 rfe_tc_3028_sm11TxPhaseStepTx1R1FaultHandling

Testcase to perform tx bist phase step tx1 r1 fault handling for sm11

4.131.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 489. rfe_tc_3028_sm11TxPhaseStepTx1R1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.131.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob

12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
    - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active

- Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.132 rfe_tc_3029_sm11TxPhaseStepTx1R2FaultPromotion

Testcase to perform tx bist phase step tx1 r1 to r2 fault promotion for sm11

4.132.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 490. rfe_tc_3029_sm11TxPhaseStepTx1R2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.132.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
 - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
 - RTS FCCU Init success
3. Clear RTS FCCU faults
 - ErrorN is not asserted
4. call rfe_configure with default settings
 - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
 - rfe_configure() is success
6. update bistZeroHourReferenceData
 - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
 - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob

10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active

- Fault is not active for unmasked SM

34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.133 rfe_tc_3032_sm12TxPhaseStepTx2R1FaultHandling

Testcase to perform tx bist phase step tx2 r1 fault handling for sm12

4.133.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

The RFE SW shall notify the host upon successful fault recovery

Table 491. rfe_tc_3032_sm12TxPhaseStepTx2R1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.133.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success

3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings

- No Faults is promoted to R2
- Radar cycle requested are complete

30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.134 rfe_tc_3033_sm12TxPhaseStepTx2R2FaultPromotion

Testcase to perform tx bist phase step tx2 r1 to r2 fault promotion for sm12

4.134.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 492. rfe_tc_3033_sm12TxPhaseStepTx2R2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.134.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().

- RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2

- Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.135 rfe_tc_3034_sm13TxPhaseDifferenceTx3Tx4R1FaultHandling

Testcase to perform tx bist phase difference tx3tx4 r1 fault handling for sm13

4.135.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 493. rfe_tc_3034_sm13TxPhaseDifferenceTx3Tx4R1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox

Table 493. rfe_tc_3034_sm13TxPhaseDifferenceTx3Tx4R1FaultHandling Specification*...continued*

Property	Description
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.135.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active

- No Faults count is incremented

23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.136 rfe_tc_3035_sm13TxPhaseDifferenceTx3Tx4R2FaultPromotion

Testcase to perform tx bist phase difference tx3tx4 r1 to r2 fault promotion for sm13

4.136.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 494. rfe_tc_3035_sm13TxPhaseDifferenceTx3Tx4R2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.136.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete

19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted

40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.137 rfe_tc_3038_sm13TxPhaseStepTx4R1FaultHandling

Testcase to perform tx bist phase step tx4 r1 fault handling for sm13

4.137.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 495. rfe_tc_3038_sm13TxPhaseStepTx4R1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.137.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob

12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
    - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active

- Fault is not active for unmasked SM

36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.138 rfe_tc_3039_sm13TxPhaseStepTx4R2FaultPromotion

Testcase to perform tx bist phase step tx4 r1 to r2 fault promotion for sm13

4.138.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 496. rfe_tc_3039_sm13TxPhaseStepTx4R2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507, 1377107,

4.138.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob

10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active

- Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
 - Fault is promoted to R2
 - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
 - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
 - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
 - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
 - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.4-TX_BIST_Phase-Diff_Step.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc562530

4.139 rfe_tc_3040_sm15Rx1BallBreak

Testcase to perform ball break detection rx1 r1 fault handling and r2 fault promotion for sm15

4.139.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 497. rfe_tc_3040_sm15Rx1BallBreak Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.139.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
 - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
 - RTS FCCU Init success
3. Clear RTS FCCU faults
 - ErrorN is not asserted

4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
   - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
   - Fault is promoted to R2
   - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
   - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
   - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
   - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
   - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail.

4.140 rfe_tc_3041_sm16Rx2BallBreak

Testcase to perform ball break detection rx2 r1 fault handling and r2 fault promotion for sm16

4.140.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 498. rfe_tc_3041_sm16Rx2BallBreak Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.140.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail.

4.141 rfe_tc_3042_sm17Rx3BallBreak

Testcase to perform ball break detection rx3 r1 fault handling and r2 fault promotion for sm17

4.141.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 499. rfe_tc_3042_sm17Rx3BallBreak Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 499. rfe_tc_3042_sm17Rx3BallBreak Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.141.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail.

4.142 rfe_tc_3043_sm18Rx4BallBreak

Testcase to perform ball break detection rx4 r1 fault handling and r2 fault promotion for sm18

4.142.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 500. rfe_tc_3043_sm18Rx4BallBreak Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.142.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e

- rfe state should match rfe_state_fuSaFault_e

39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail.

4.143 rfe_tc_3046_sm19Rx1LOLevelDetectorR1FaultHandling

Testcase to perform lo lev detector for rx1 r1 fault handling for sm19

4.143.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 501. rfe_tc_3046_sm19Rx1LOLevelDetectorR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.143.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success

9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete

34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.144 rfe_tc_3047_sm19Rx1LOLevelDetectorR2FaultPromotion

Testcase to perform lo lev detector for rx1 r1 to r2 fault promotion for sm19

4.144.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 502. rfe_tc_3047_sm19Rx1LOLevelDetectorR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.144.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success

6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings

  - No Faults is promoted to R2
  - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
  - No Faults is promoted to R2
  - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
  - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
  - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
  - Fault is promoted to R2
  - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
  - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
  - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
  - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
  - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.145 rfe_tc_3048_sm20Rx2LOLevelDetectorR1FaultHandling

Testcase to perform lo lev detector for rx2 r1 fault handling for sm20

4.145.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 503. rfe_tc_3048_sm20Rx2LOLevelDetectorR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.145.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete

26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.146 rfe_tc_3049_sm20Rx2LOLevelDetectorR2FaultPromotion

Testcase to perform lo lev detector for rx2 r1 to r2 fault promotion for sm20

4.146.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 504. rfe_tc_3049_sm20Rx2LOLevelDetectorR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional

Table 504. rfe_tc_3049_sm20Rx2LOLevelDetectorR2FaultPromotion Specification*...continued*

Property	Description
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.146.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active

22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.147 rfe_tc_3050_sm21Rx3LOLevelDetectorR1FaultHandling

Testcase to perform lo lev detector for rx3 r1 fault handling for sm21

4.147.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 505. rfe_tc_3050_sm21Rx3LOLevelDetectorR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.147.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4

16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
    - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM

37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.148 rfe_tc_3051_sm21Rx3LOLevelDetectorR2FaultPromotion

Testcase to perform lo lev detector for rx3 r1 to r2 fault promotion for sm21

4.148.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 506. rfe_tc_3051_sm21Rx3LOLevelDetectorR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.148.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM

34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.149 rfe_tc_3052_sm22Rx4LOLevelDetectorR1FaultHandling

Testcase to perform lo lev detector for rx4 r1 fault handling for sm22

4.149.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 507. rfe_tc_3052_sm22Rx4LOLevelDetectorR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.149.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success

3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings

   - No Faults is promoted to R2
   - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
   - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.150 rfe_tc_3053_sm22Rx4LOLevelDetectorR2FaultPromotion

Testcase to perform lo lev detector for rx4 r1 to r2 fault promotion for sm22

4.150.1 Detailed Description

When configured to perform SW based fault recovery, RFE FW shall 1. Set fault in FCCU to trigger ERROR_N to LOW : For SW safety mechanism 2. Perform fault recovery and if fault recovery is successful then trigger FCCU to set ERROR_N to HIGH else do nothing so that Host can take necessary action to trigger recovery via REBOOT

Table 508. rfe_tc_3053_sm22Rx4LOLevelDetectorR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305759

4.150.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.5- LO_LVL_DETECTOR.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc568235

4.151 rfe_tc_3054_sm23RxBistR1FaultHandling

Testcase to perform rx bist r1 fault handling for sm23

4.151.1 Detailed Description

The RFE SW shall set RFE FCCU fault flag when fault is detected by SW.

Table 509. rfe_tc_3054_sm23RxBistR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.

Table 509. rfe_tc_3054_sm23RxBistR1FaultHandling Specification*...continued*

Property	Description
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1299507

4.151.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count

- No Faults is promoted to R2
- Radar cycle requested are complete

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.6-RX_BIST_Phase_and_Gain_Diff.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc563067

4.152 rfe_tc_3055_sm23RxBistR2FaultPromotion

Testcase to perform rx bist r1 to r2 fault promotion for sm23

4.152.1 Detailed Description

When configured the RFE FW shall handle the RFE FCCU faults/warnings as well as SW detected faults

- fault recovery action

- fault signaling to main FCCU

Table 510. rfe_tc_3055_sm23RxBistR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1305758

4.152.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

  - Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e

- rfe state should match rfe_state_fuSaFault_e

39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.6-RX_BIST_Phase_and_Gain_Diff.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc563067

4.153 rfe_tc_3056_sm28Gldo1v3OVR1FaultHandling

Testcase to perform gldo 1v3 ov only r1 fault handling for sm28

4.153.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 511. rfe_tc_3056_sm28Gldo1v3OVR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.153.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings

- rfe_configure() is success

10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
    - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count

- No Faults is promoted to R2
- Radar cycle requested are complete

35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.7.1.a - GLDO OV 1V3 - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc563338

4.154 rfe_tc_3065_sm50AdpllDCOLevelHighDetectR1FaultHandling

Testcase to perform adpll dco lev high detector r1 fault handling for sm50

4.154.1 Detailed Description

When configured the RFE SW shall perform recovery for Clock PLL faults

Table 512. rfe_tc_3065_sm50AdpllDCOLevelHighDetectR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311262

4.154.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success

8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
    - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings

- No Faults is promoted to R2
- Radar cycle requested are complete

34. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
 - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.19 - ADPLL_DCO_LevelDetector.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553494

4.155 rfe_tc_3066_sm50AdpllDCOLevelHighDetectR2FaultPromotion

Testcase to perform adpll dco lev high detector r1 to r2 fault promotion for sm50

4.155.1 Detailed Description

When configured the RFE SW shall perform recovery for Clock PLL faults

Table 513. rfe_tc_3066_sm50AdpllDCOLevelHighDetectR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311262

4.155.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
 - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
 - RTS FCCU Init success
3. Clear RTS FCCU faults
 - ErrorN is not asserted
4. call rfe_configure with default settings
 - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
 - rfe_configure() is success
6. update bistZeroHourReferenceData

   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
   - No Faults is promoted to R2

  - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.19 - ADPLL_DCO_LevelDetector.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553494

4.156 rfe_tc_3068_sm50AdpllDCOLevelLowDetectR2FaultPromotion

Testcase to perform adpll dco lev low detector r1 to r2 fault promotion for sm50

4.156.1 Detailed Description

When configured the RFE SW shall perform recovery for Clock PLL faults

Table 514. rfe_tc_3068_sm50AdpllDCOLevelLowDetectR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311262

4.156.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().

- RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2

- Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.19 - ADPLL_DCO_LevelDetector.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553494

4.157 rfe_tc_3069_sm54ChirpPllRMSVcoLevelHighR1FaultHandling

Testcase to perform chirppll rms dco detector and amplitude monitor high r1 fault handling for sm54

4.157.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 515. rfe_tc_3069_sm54ChirpPllRMSVcoLevelHighR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.157.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle

- No Faults is promoted to R2
- Radar cycle requested are complete

20. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.10 - ChirpPLL_RMS VCO Detector_Amplitude Monitor.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553482

4.158 rfe_tc_3070_sm54ChirpPllRMSVcoLevelHighR2FaultPromotion

Testcase to perform chirppll rms dco detector and amplitude monitor high r1 to r2 fault promotion for sm54

4.158.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 516. rfe_tc_3070_sm54ChirpPllRMSVcoLevelHighR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.158.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4

14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion

   - Fault is promoted to R2
   - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
   - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
   - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
   - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
   - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.10 - ChirpPLL_RMS VCO Detector_Amplitude Monitor.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553482

4.159 rfe_tc_3071_sm54ChirpPllRMSVcoLevelLowR1FaultHandling

Testcase to perform chirppll rms dco detector and amplitude monitor low r1 fault handling for sm54

4.159.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 517. rfe_tc_3071_sm54ChirpPllRMSVcoLevelLowR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.159.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization

- RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM

29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
 - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
 - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.10 - ChirpPLL_RMS VCO Detector_Amplitude Monitor.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553482

4.160 rfe_tc_3072_sm54ChirpPllRMSVcoLevelLowR2FaultPromotion

Testcase to perform chirppll rms dco detector and amplitude monitor low r1 to r2 fault promotion for sm54

4.160.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 518. rfe_tc_3072_sm54ChirpPllRMSVcoLevelLowR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A

Table 518. rfe_tc_3072_sm54ChirpPllRMSVcoLevelLowR2FaultPromotion Specification*...continued*

Property	Description
Execution	Automated
Satisfied Requirements	1311269

4.160.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2

- Radar cycle requested are complete

24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
   - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
   - Fault is promoted to R2
   - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
   - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
   - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
   - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
   - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.10 - ChirpPLL_RMS VCO Detector_Amplitude Monitor.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553482

4.161 rfe_tc_3073_sm55ChirpPllUnlockDetectorR1FaultHandling

Testcase to perform chirppll unlock detector r1 fault handling for sm55

4.161.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 519. rfe_tc_3073_sm55ChirpPllUnlockDetectorR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.161.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3

15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
    - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM

37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.11 - ChirpPLL_Unlock_Detectors.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553490

4.162 rfe_tc_3074_sm55ChirpPllUnlockDetectorR2FaultPromotion

Testcase to perform chirppll unlock detector r1 to r2 fault promotion for sm55

4.162.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 520. rfe_tc_3074_sm55ChirpPllUnlockDetectorR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.162.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings

- rfe_configure() is success

8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count

- No Faults is promoted to R2
- Radar cycle requested are complete

33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.11 - ChirpPLL_Unlock_Detectors.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553490

4.163 rfe_tc_3075_sm57ChirpPllCentreFrequencyDriftedR1FaultHandling

Testcase to perform chirppll frequency monitor r1 fault handling for sm57

4.163.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 521. rfe_tc_3075_sm57ChirpPllCentreFrequencyDriftedR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.163.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete

26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.17 - ChirpPLL frequency monitor.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553491

4.164 rfe_tc_3076_sm57ChirpPllCentreFrequencyDriftedR2FaultPromotion

Testcase to perform chirppll frequency monitor r1 to r2 fault promotion for sm57

4.164.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 522. rfe_tc_3076_sm57ChirpPllCentreFrequencyDriftedR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.

Table 522. rfe_tc_3076_sm57ChirpPllCentreFrequencyDriftedR2FaultPromotion Specification*...continued*

Property	Description
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.164.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active

20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.17 - ChirpPLL frequency monitor.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc553491

4.165 rfe_tc_3079_sm59ChirpPllLockStepLinearInterR1FaultHandling

Testcase to perform chirppll lockstep linearinter r1 fault handling for sm59

4.165.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 523. rfe_tc_3079_sm59ChirpPllLockStepLinearInterR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.165.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success

9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete

34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.12 - ChirpPLL_LockStep_X.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582723

4.166 rfe_tc_3080_sm59ChirpPllLockStepLinearInterR2FaultPromotion

Testcase to perform chirppll lockstep linearinter r1 to r2 fault promotion for sm59

4.166.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 524. rfe_tc_3080_sm59ChirpPllLockStepLinearInterR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.166.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted

4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

30. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
 - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
 - Fault is promoted to R2
 - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
 - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
 - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
 - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
 - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.12 - ChirpPLL_LockStep_X.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582723

4.167 rfe_tc_3081_sm60ChirpPllLockStepSigmaDeltaR1FaultHandling

Testcase to perform chirppll lockstep sigmadelta r1 fault handling for sm60

4.167.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection

- Linearity detection

- Frequency monitor

- Delayed lock step(Timing Engine)

Table 525. rfe_tc_3081_sm60ChirpPllLockStepSigmaDeltaR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 525. rfe_tc_3081_sm60ChirpPllLockStepSigmaDeltaR1FaultHandling Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.167.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
    - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.12 - ChirpPLL_LockStep_X.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582723

4.168 rfe_tc_3082_sm60ChirpPllLockStepSigmaDeltaR2FaultPromotion

Testcase to perform chirppll lockstep sigmadelta r1 to r2 fault promotion for sm60

4.168.1 Detailed Description

When configured the RFE SW shall perform recovery for Chirp PLL faults

- Unlock detection

- Level detection
- Linearity detection
- Frequency monitor
- Delayed lock step(Timing Engine)

Table 526. rfe_tc_3082_sm60ChirpPllLockStepSigmaDeltaR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311269

4.168.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active

- No Faults count is incremented

17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM

37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.12 - ChirpPLL_LockStep_X.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582723

4.169 rfe_tc_3083_sm62TeLockStepR1FaultHandling

Testcase to perform te lockstep r1 fault handling for sm62

4.169.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 527. rfe_tc_3083_sm62TeLockStepR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.169.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
4. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
5. Clear RTS FCCU faults
   - ErrorN is not asserted
6. call rfe_configure with default settings
   - rfe_configure() is success
7. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success

8. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
9. call rfe_configure with default settings
   - rfe_configure() is success
10. unmask Fusa Fault in rfeConfigure_modify config blob
11. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
12. configure 1st hook to read the register no fault is active in radar cycle 1
13. configure 2nd hook to perform fault injection in radar cycle 2
14. configure 3rd hook to read register to check fault injection is active in radar cycle 3
15. configure 4th hook to clear fault injection in radar cycle 4
16. Read register to check no faults are active
17. call rfe_getFuSaFaults to check no faults active
   - No Faults active
18. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
19. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
20. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_getFuSaFaults to check no faults active
   - No Faults active
24. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
25. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
26. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
27. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
28. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
29. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
30. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
31. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
32. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
33. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings

- No Faults is promoted to R2
- Radar cycle requested are complete

34. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
35. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
36. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
37. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.12 - ChirpPLL_LockStep_X.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582723

4.170 rfe_tc_3084_sm62TeLockStepR2FaultPromotion

Testcase to perform te lockstep r1 to r2 fault promotion for sm62

4.170.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 528. rfe_tc_3084_sm62TeLockStepR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.170.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData

- update bistZeroHourReferenceData is success

7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
   - No Faults is promoted to R2

- Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.12 - ChirpPLL_LockStep_X.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582723

4.171 rfe_tc_3085_sm63TeSwtRadarPipelineR2Handling

Testcase to perform te swt radar pipeline (software watchdog timer) r2 fault handling for sm63

4.171.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 529. rfe_tc_3085_sm63TeSwtRadarPipelineR2Handling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.171.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().

- RFE moves to rfe_state_initialized_e state

2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP_SM63_TE_swt_radar_pipeline.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc612998

4.172 rfe_tc_3112_sm68RfgCrcAdpll

Testcase to perform rfe register dig crc adpll testing idx e r2 falut handling for sm68

4.172.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 530. rfe_tc_3112_sm68RfgCrcAdpll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.172.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active

16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.173 rfe_tc_3113_sm68RfgCrcMcgen

Testcase to perform rfe register dig crc mcgen testing idx e r2 falut handling for sm68

4.173.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 531. rfe_tc_3113_sm68RfgCrcMcgen Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated

Table 531. rfe_tc_3113_sm68RfgCrcMcgen Specification*...continued*

Property	Description
Satisfied Requirements	1311226

4.173.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete

24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.174 rfe_tc_3114_sm69RfgCrcAdc1

Testcase to perform rfe register dig crc adc1 testing idx e r2 falut handling for sm69

4.174.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 532. rfe_tc_3114_sm69RfgCrcAdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.174.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success

8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.175 rfe_tc_3115_sm69RfgCrcAdc2

Testcase to perform rfe register dig crc adc2 testing idx e r2 falut handling for sm69

4.175.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 533. rfe_tc_3115_sm69RfgCrcAdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.

Table 533. rfe_tc_3115_sm69RfgCrcAdc2 Specification*...continued*

Property	Description
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.175.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active

20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.176 rfe_tc_3116_sm69RfgCrcAdc3

Testcase to perform rfe register dig crc adc3 testing idx e r2 falut handling for sm69

4.176.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 534. rfe_tc_3116_sm69RfgCrcAdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.176.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults

- ErrorN is not asserted

4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.177 rfe_tc_3117_sm69RfgCrcAdc4

Testcase to perform rfe register dig crc adc4 testing idx e r2 falut handling for sm69

4.177.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 535. rfe_tc_3117_sm69RfgCrcAdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.177.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.178 rfe_tc_3118_sm69RfgCrcAtb

Testcase to perform rfe register dig crc atb testing idx e r2 falut handling for sm69

4.178.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 536. rfe_tc_3118_sm69RfgCrcAtb Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.178.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.179 rfe_tc_3119_sm69RfgCrcChirpPll

Testcase to perform rfe register dig crc chirppll testing idx e r2 falut handling for sm69

4.179.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 537. rfe_tc_3119_sm69RfgCrcChirpPll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.179.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.180 rfe_tc_3120_sm69RfgCrcGbias

Testcase to perform rfe register dig crc gbias testing idx e r2 falut handling for sm69

4.180.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 538. rfe_tc_3120_sm69RfgCrcGbias Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 538. rfe_tc_3120_sm69RfgCrcGbias Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.180.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.181 rfe_tc_3121_sm69RfgCrcGldo

Testcase to perform rfe register dig crc gldo testing idx e r2 falut handling for sm69

4.181.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 539. rfe_tc_3121_sm69RfgCrcGldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.181.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.182 rfe_tc_3122_sm69RfgCrcIsm

Testcase to perform rfe register dig crc ism testing idx e r2 falut handling for sm69

4.182.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 540. rfe_tc_3122_sm69RfgCrcIsm Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.182.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.183 rfe_tc_3123_sm69RfgCrcLldo

Testcase to perform rfe register dig crc lldodig testing idx e r2 falut handling for sm69

4.183.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 541. rfe_tc_3123_sm69RfgCrcLldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.183.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.184 rfe_tc_3124_sm69RfgCrcLldoPdc

Testcase to perform rfe register dig crc lldopdc testing idx e r2 falut handling for sm69

4.184.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 542. rfe_tc_3124_sm69RfgCrcLldoPdc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.184.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.185 rfe_tc_3125_sm69RfgCrcLoif

Testcase to perform rfe register dig crc loif testing idx e r2 falut handling for sm69

4.185.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 543. rfe_tc_3125_sm69RfgCrcLoif Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 543. rfe_tc_3125_sm69RfgCrcLoif Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.185.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.186 rfe_tc_3126_sm69RfgCrcPdc1

Testcase to perform rfe register dig crc pdc1 testing idx e r2 falut handling for sm69

4.186.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 544. rfe_tc_3126_sm69RfgCrcPdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.186.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.187 rfe_tc_3127_sm69RfgCrcPdc2

Testcase to perform rfe register dig crc pdc2 testing idx e r2 falut handling for sm69

4.187.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 545. rfe_tc_3127_sm69RfgCrcPdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.187.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.188 rfe_tc_3128_sm69RfgCrcPdc3

Testcase to perform rfe register dig crc pdc3 testing idx e r2 falut handling for sm69

4.188.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 546. rfe_tc_3128_sm69RfgCrcPdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.188.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.189 rfe_tc_3129_sm69RfgCrcPdc4

Testcase to perform rfe register dig crc pdc4 testing idx e r2 falut handling for sm69

4.189.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 547. rfe_tc_3129_sm69RfgCrcPdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.189.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.190 rfe_tc_3130_sm69RfgCrcRcosc

Testcase to perform rfe register dig crc rcosc testing idx e r2 falut handling for sm69

4.190.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 548. rfe_tc_3130_sm69RfgCrcRcosc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 548. rfe_tc_3130_sm69RfgCrcRcosc Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.190.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.191 rfe_tc_3131_sm69RfgCrcRx1

Testcase to perform rfe register dig crc rx1 testing idx e r2 falut handling for sm69

4.191.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 549. rfe_tc_3131_sm69RfgCrcRx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.191.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.192 rfe_tc_3132_sm69RfgCrcRx2

Testcase to perform rfe register dig crc rx2 testing idx e r2 falut handling for sm69

4.192.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 550. rfe_tc_3132_sm69RfgCrcRx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.192.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.193 rfe_tc_3133_sm69RfgCrcRx3

Testcase to perform rfe register dig crc rx3 testing idx e r2 falut handling for sm69

4.193.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 551. rfe_tc_3133_sm69RfgCrcRx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.193.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.194 rfe_tc_3134_sm69RfgCrcRx4

Testcase to perform rfe register dig crc rx4 testing idx e r2 falut handling for sm69

4.194.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 552. rfe_tc_3134_sm69RfgCrcRx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.194.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.195 rfe_tc_3135_sm69RfgCrcRxBist

Testcase to perform rfe register dig crc rxbist testing idx e r2 falut handling for sm69

4.195.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 553. rfe_tc_3135_sm69RfgCrcRxBist Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 553. rfe_tc_3135_sm69RfgCrcRxBist Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.195.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.196 rfe_tc_3136_sm69RfgCrcSpim

Testcase to perform rfe register dig crc spim testing idx e r2 falut handling for sm69

4.196.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 554. rfe_tc_3136_sm69RfgCrcSpim Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.196.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.197 rfe_tc_3137_sm69RfgCrcTe

Testcase to perform rfe register dig crc te testing idx e r2 falut handling for sm69

4.197.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 555. rfe_tc_3137_sm69RfgCrcTe Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.197.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.198 rfe_tc_3138_sm69RfgCrcTempSensor1

Testcase to perform rfe register dig crc tempsensor1 testing idx e r2 falut handling for sm69

4.198.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 556. rfe_tc_3138_sm69RfgCrcTempSensor1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.198.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.199 rfe_tc_3139_sm69RfgCrcTempSensor2

Testcase to perform rfe register dig crc tempsensor2 testing idx e r2 falut handling for sm69

4.199.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 557. rfe_tc_3139_sm69RfgCrcTempSensor2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.199.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.200 rfe_tc_3140_sm69RfgCrcTempSensor3

Testcase to perform rfe register dig crc tempsensor3 testing idx e r2 falut handling for sm69

4.200.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 558. rfe_tc_3140_sm69RfgCrcTempSensor3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 558. rfe_tc_3140_sm69RfgCrcTempSensor3 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.200.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
 - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
 - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.201 rfe_tc_3141_sm69RfgCrcTempSensor4

Testcase to perform rfe register dig crc tempsensor4 testing idx e r2 falut handling for sm69

4.201.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 559. rfe_tc_3141_sm69RfgCrcTempSensor4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.201.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
 - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
 - RTS FCCU Init success
3. Clear RTS FCCU faults
 - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.202 rfe_tc_3142_sm69RfgCrcTx1

Testcase to perform rfe register dig crc tx1 testing idx e r2 falut handling for sm69

4.202.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 560. rfe_tc_3142_sm69RfgCrcTx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.202.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.203 rfe_tc_3143_sm69RfgCrcTx2

Testcase to perform rfe register dig crc tx2 testing idx e r2 falut handling for sm69

4.203.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 561. rfe_tc_3143_sm69RfgCrcTx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.203.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.204 rfe_tc_3144_sm69RfgCrcTx3

Testcase to perform rfe register dig crc tx3 testing idx e r2 falut handling for sm69

4.204.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 562. rfe_tc_3144_sm69RfgCrcTx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.204.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.205 rfe_tc_3145_sm69RfgCrcTx4

Testcase to perform rfe register dig crc tx4 testing idx e r2 falut handling for sm69

4.205.1 Detailed Description

The RFE SW shall trigger CRC check of RFE registers (for SPI-based RFE IPs, Infra IPs and RFE Digital IPs)

Table 563. rfe_tc_3145_sm69RfgCrcTx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 563. rfe_tc_3145_sm69RfgCrcTx4 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311226

4.205.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.24 - SM68_MCGEN_REG_DIG_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582836

4.206 rfe_tc_3146_sm70MosiCrcMcgen

Testcase to perform spi transaction crc for mcgen ip e mosi r2 fault injection test r2 falut handling for sm70

4.206.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 564. rfe_tc_3146_sm70MosiCrcMcgen Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.206.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.207 rfe_tc_3147_sm70MisoCrcMcgen

Testcase to perform spi transaction crc for mcgen ip e miso r2 fault injection test r2 falut handling for sm70

4.207.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 565. rfe_tc_3147_sm70MisoCrcMcgen Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.207.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.208 rfe_tc_3148_sm70MosiCrcChirpPll

Testcase to perform spi transaction crc for chirppll ip e mosi r2 fault injection test r2 falut handling for sm70

4.208.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 566. rfe_tc_3148_sm70MosiCrcChirpPll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.208.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.209 rfe_tc_3149_sm70MisoCrcChirpPll

Testcase to perform spi transaction crc for chirppll ip e miso r2 fault injection test r2 falut handling for sm70

4.209.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 567. rfe_tc_3149_sm70MisoCrcChirpPll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.209.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.210 rfe_tc_3150_sm70MosiCrcIsm

Testcase to perform spi transaction crc for ism ip e mosi r2 fault injection test r2 falut handling for sm70

4.210.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 568. rfe_tc_3150_sm70MosiCrcIsm Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 568. rfe_tc_3150_sm70MosiCrcIsm Specification_...continued_

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.210.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.211 rfe_tc_3151_sm70MisoCrcIsm

Testcase to perform spi transaction crc for ism ip e miso r2 fault injection test r2 falut handling for sm70

4.211.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 569. rfe_tc_3151_sm70MisoCrcIsm Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.211.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.212 rfe_tc_3152_sm70MosiCrcSpim

Testcase to perform spi transaction crc for spim ip e mosi r2 fault injection test r2 falut handling for sm70

4.212.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 570. rfe_tc_3152_sm70MosiCrcSpim Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.212.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.213 rfe_tc_3153_sm70MisoCrcSpim

Testcase to perform spi transaction crc for spim ip e miso r2 fault injection test r2 falut handling for sm70

4.213.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 571. rfe_tc_3153_sm70MisoCrcSpim Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.213.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.214 rfe_tc_3154_sm70MosiCrcAtb

Testcase to perform spi transaction crc for atb ip e mosi r2 fault injection test r2 falut handling for sm70

4.214.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 572. rfe_tc_3154_sm70MosiCrcAtb Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.214.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.215 rfe_tc_3155_sm70MisoCrcAtb

Testcase to perform spi transaction crc for atb ip e miso r2 fault injection test r2 falut handling for sm70

4.215.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 573. rfe_tc_3155_sm70MisoCrcAtb Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 573. rfe_tc_3155_sm70MisoCrcAtb Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.215.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.216 rfe_tc_3156_sm70MosiCrcTe

Testcase to perform spi transaction crc for te ip e mosi r2 fault injection test r2 falut handling for sm70

4.216.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 574. rfe_tc_3156_sm70MosiCrcTe Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.216.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.217 rfe_tc_3157_sm70MisoCrcTe

Testcase to perform spi transaction crc for te ip e miso r2 fault injection test r2 falut handling for sm70

4.217.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 575. rfe_tc_3157_sm70MisoCrcTe Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.217.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.218 rfe_tc_3158_sm70MosiCrcRxBist

Testcase to perform spi transaction crc for rxbist ip e mosi r2 fault injection test r2 falut handling for sm70

4.218.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 576. rfe_tc_3158_sm70MosiCrcRxBist Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.218.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.219 rfe_tc_3159_sm70MisoCrcRxBist

Testcase to perform spi transaction crc for rxbist ip e miso r2 fault injection test r2 falut handling for sm70

4.219.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 577. rfe_tc_3159_sm70MisoCrcRxBist Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.219.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.220 rfe_tc_3160_sm70MosiCrcGbias

Testcase to perform spi transaction crc for gbias ip e mosi r2 fault injection test r2 falut handling for sm70

4.220.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 578. rfe_tc_3160_sm70MosiCrcGbias Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 578. rfe_tc_3160_sm70MosiCrcGbias Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.220.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.221 rfe_tc_3161_sm70MisoCrcGbias

Testcase to perform spi transaction crc for gbias ip e miso r2 fault injection test r2 falut handling for sm70

4.221.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 579. rfe_tc_3161_sm70MisoCrcGbias Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.221.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.222 rfe_tc_3162_sm70MosiCrcGldo

Testcase to perform spi transaction crc for gldo ip e mosi r2 fault injection test r2 falut handling for sm70

4.222.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 580. rfe_tc_3162_sm70MosiCrcGldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.222.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.223 rfe_tc_3163_sm70MisoCrcGldo

Testcase to perform spi transaction crc for gldo ip e miso r2 fault injection test r2 falut handling for sm70

4.223.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 581. rfe_tc_3163_sm70MisoCrcGldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.223.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.224 rfe_tc_3164_sm70MosiCrcRcosc

Testcase to perform spi transaction crc for rcosc ip e mosi r2 fault injection test r2 falut handling for sm70

4.224.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 582. rfe_tc_3164_sm70MosiCrcRcosc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.224.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.225 rfe_tc_3165_sm70MisoCrcRcosc

Testcase to perform spi transaction crc for rcosc ip e miso r2 fault injection test r2 falut handling for sm70

4.225.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 583. rfe_tc_3165_sm70MisoCrcRcosc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 583. rfe_tc_3165_sm70MisoCrcRcosc Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.225.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.226 rfe_tc_3166_sm70MosiCrcLldo

Testcase to perform spi transaction crc for lldo ip e mosi r2 fault injection test r2 falut handling for sm70

4.226.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 584. rfe_tc_3166_sm70MosiCrcLldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.226.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.227 rfe_tc_3167_sm70MisoCrcLldo

Testcase to perform spi transaction crc for lldo ip e miso r2 fault injection test r2 falut handling for sm70

4.227.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 585. rfe_tc_3167_sm70MisoCrcLldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.227.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.228 rfe_tc_3168_sm70MosiCrcLldoPdc

Testcase to perform spi transaction crc for lldopdc ip e mosi r2 fault injection test r2 falut handling for sm70

4.228.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 586. rfe_tc_3168_sm70MosiCrcLldoPdc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.228.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.229 rfe_tc_3169_sm70MisoCrcLldoPdc

Testcase to perform spi transaction crc for lldopdc ip e miso r2 fault injection test r2 falut handling for sm70

4.229.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 587. rfe_tc_3169_sm70MisoCrcLldoPdc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.229.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.230 rfe_tc_3170_sm70MosiCrcLoif

Testcase to perform spi transaction crc for lo ip e mosi r2 fault injection test r2 falut handling for sm70

4.230.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 588. rfe_tc_3170_sm70MosiCrcLoif Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 588. rfe_tc_3170_sm70MosiCrcLoif Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.230.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.231 rfe_tc_3171_sm70MisoCrcLoif

Testcase to perform spi transaction crc for lo ip e miso r2 fault injection test r2 falut handling for sm70

4.231.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 589. rfe_tc_3171_sm70MisoCrcLoif Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.231.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

  - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.232 rfe_tc_3172_sm70MosiCrcAdc1

Testcase to perform spi transaction crc for adc1 ip e mosi r2 fault injection test r2 falut handling for sm70

4.232.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 590. rfe_tc_3172_sm70MosiCrcAdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.232.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.233 rfe_tc_3173_sm70MisoCrcAdc1

Testcase to perform spi transaction crc for adc1 ip e miso r2 fault injection test r2 falut handling for sm70

4.233.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 591. rfe_tc_3173_sm70MisoCrcAdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.233.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.234 rfe_tc_3174_sm70MosiCrcADc2

Testcase to perform spi transaction crc for adc2 ip e mosi r2 fault injection test r2 falut handling for sm70

4.234.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 592. rfe_tc_3174_sm70MosiCrcADc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.234.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.235 rfe_tc_3175_sm70MisoCrcAdc2

Testcase to perform spi transaction crc for adc2 ip e miso r2 fault injection test r2 falut handling for sm70

4.235.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 593. rfe_tc_3175_sm70MisoCrcAdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 593. rfe_tc_3175_sm70MisoCrcAdc2 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.235.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.236 rfe_tc_3176_sm70MosiCrcAdc3

Testcase to perform spi transaction crc for adc3 ip e mosi r2 fault injection test r2 falut handling for sm70

4.236.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 594. rfe_tc_3176_sm70MosiCrcAdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.236.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.237 rfe_tc_3177_sm70MisoCrcAdc3

Testcase to perform spi transaction crc for adc3 ip e miso r2 fault injection test r2 falut handling for sm70

4.237.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 595. rfe_tc_3177_sm70MisoCrcAdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.237.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.238 rfe_tc_3178_sm70MosiCrcAdc4

Testcase to perform spi transaction crc for adc4 ip e mosi r2 fault injection test r2 falut handling for sm70

4.238.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 596. rfe_tc_3178_sm70MosiCrcAdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.238.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.239 rfe_tc_3179_sm70MisoCrcAdc4

Testcase to perform spi transaction crc for adc4 ip e miso r2 fault injection test r2 falut handling for sm70

4.239.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 597. rfe_tc_3179_sm70MisoCrcAdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.239.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.240 rfe_tc_3180_sm70MosiCrcPdc1

Testcase to perform spi transaction crc for pdc1 ip e mosi r2 fault injection test r2 falut handling for sm70

4.240.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 598. rfe_tc_3180_sm70MosiCrcPdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 598. rfe_tc_3180_sm70MosiCrcPdc1 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.240.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.241 rfe_tc_3181_sm70MisoCrcPdc1

Testcase to perform spi transaction crc for pdc1 ip e miso r2 fault injection test r2 falut handling for sm70

4.241.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 599. rfe_tc_3181_sm70MisoCrcPdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.241.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.242 rfe_tc_3182_sm70MosiCrcPdc2

Testcase to perform spi transaction crc for pdc2 ip e mosi r2 fault injection test r2 falut handling for sm70

4.242.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 600. rfe_tc_3182_sm70MosiCrcPdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.242.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.243 rfe_tc_3183_sm70MisoCrcPdc2

Testcase to perform spi transaction crc for pdc2 ip e miso r2 fault injection test r2 falut handling for sm70

4.243.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 601. rfe_tc_3183_sm70MisoCrcPdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.243.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.244 rfe_tc_3184_sm70MosiCrcPdc3

Testcase to perform spi transaction crc for pdc3 ip e mosi r2 fault injection test r2 falut handling for sm70

4.244.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 602. rfe_tc_3184_sm70MosiCrcPdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.244.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.245 rfe_tc_3185_sm70MisoCrcPdc3

Testcase to perform spi transaction crc for pdc3 ip e miso r2 fault injection test r2 falut handling for sm70

4.245.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 603. rfe_tc_3185_sm70MisoCrcPdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 603. rfe_tc_3185_sm70MisoCrcPdc3 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.245.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.246 rfe_tc_3186_sm70MosiCrcPdc4

Testcase to perform spi transaction crc for pdc4 ip e mosi r2 fault injection test r2 falut handling for sm70

4.246.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 604. rfe_tc_3186_sm70MosiCrcPdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.246.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.247 rfe_tc_3187_sm70MisoCrcPdc4

Testcase to perform spi transaction crc for pdc4 ip e miso r2 fault injection test r2 falut handling for sm70

4.247.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 605. rfe_tc_3187_sm70MisoCrcPdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.247.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.248 rfe_tc_3188_sm70MosiCrcTx1

Testcase to perform spi transaction crc for tx1 ip e mosi r2 fault injection test r2 falut handling for sm70

4.248.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 606. rfe_tc_3188_sm70MosiCrcTx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.248.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.249 rfe_tc_3189_sm70MisoCrcTx1

Testcase to perform spi transaction crc for tx1 ip e miso r2 fault injection test r2 falut handling for sm70

4.249.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 607. rfe_tc_3189_sm70MisoCrcTx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.249.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.250 rfe_tc_3190_sm70MosiCrcTx2

Testcase to perform spi transaction crc for tx2 ip e mosi r2 fault injection test r2 falut handling for sm70

4.250.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 608. rfe_tc_3190_sm70MosiCrcTx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 608. rfe_tc_3190_sm70MosiCrcTx2 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.250.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.251 rfe_tc_3191_sm70MisoCrcTx2

Testcase to perform spi transaction crc for tx2 ip e miso r2 fault injection test r2 falut handling for sm70

4.251.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 609. rfe_tc_3191_sm70MisoCrcTx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.251.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success

5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.252 rfe_tc_3192_sm70MosiCrcTx3

Testcase to perform spi transaction crc for tx3 ip e mosi r2 fault injection test r2 falut handling for sm70

4.252.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 610. rfe_tc_3192_sm70MosiCrcTx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.252.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
 - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
 - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
 - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.253 rfe_tc_3193_sm70MisoCrcTx3

Testcase to perform spi transaction crc for tx3 ip e miso r2 fault injection test r2 falut handling for sm70

4.253.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 611. rfe_tc_3193_sm70MisoCrcTx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.253.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.254 rfe_tc_3194_sm70MosiCrcTx4

Testcase to perform spi transaction crc for tx4 ip e mosi r2 fault injection test r2 falut handling for sm70

4.254.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 612. rfe_tc_3194_sm70MosiCrcTx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.254.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.255 rfe_tc_3195_sm70MisoCrcTx4

Testcase to perform spi transaction crc for tx4 ip e miso r2 fault injection test r2 falut handling for sm70

4.255.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 613. rfe_tc_3195_sm70MisoCrcTx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 613. rfe_tc_3195_sm70MisoCrcTx4 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.255.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.256 rfe_tc_3196_sm70MosiCrcRx1

Testcase to perform spi transaction crc for rx1 ip e mosi r2 fault injection test r2 falut handling for sm70

4.256.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 614. rfe_tc_3196_sm70MosiCrcRx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.256.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.257 rfe_tc_3197_sm70MisoCrcRx1

Testcase to perform spi transaction crc for rx1 ip e miso r2 fault injection test r2 falut handling for sm70

4.257.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 615. rfe_tc_3197_sm70MisoCrcRx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.257.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.258 rfe_tc_3198_sm70MosiCrcRx2

Testcase to perform spi transaction crc for rx2 ip e mosi r2 fault injection test r2 falut handling for sm70

4.258.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 616. rfe_tc_3198_sm70MosiCrcRx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.258.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.259 rfe_tc_3199_sm70MisoCrcRx2

Testcase to perform spi transaction crc for rx2 ip e miso r2 fault injection test r2 falut handling for sm70

4.259.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 617. rfe_tc_3199_sm70MisoCrcRx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.259.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.260 rfe_tc_3200_sm70MosiCrcRx3

Testcase to perform spi transaction crc for rx3 ip e mosi r2 fault injection test r2 falut handling for sm70

4.260.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 618. rfe_tc_3200_sm70MosiCrcRx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 618. rfe_tc_3200_sm70MosiCrcRx3 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.260.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.261 rfe_tc_3201_sm70MisoCrcRx3

Testcase to perform spi transaction crc for rx3 ip e miso r2 fault injection test r2 falut handling for sm70

4.261.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 619. rfe_tc_3201_sm70MisoCrcRx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.261.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.262 rfe_tc_3202_sm70MosiCrcRx4

Testcase to perform spi transaction crc for rx4 ip e mosi r2 fault injection test r2 falut handling for sm70

4.262.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 620. rfe_tc_3202_sm70MosiCrcRx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.262.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.263 rfe_tc_3203_sm70MisoCrcRx4

Testcase to perform spi transaction crc for rx4 ip e miso r2 fault injection test r2 falut handling for sm70

4.263.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 621. rfe_tc_3203_sm70MisoCrcRx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.263.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.264 rfe_tc_3204_sm70MosiCrcTempsensor1

Testcase to perform spi transaction crc for tempsensor1 ip e mosi r2 fault injection test for sm70

4.264.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 622. rfe_tc_3204_sm70MosiCrcTempsensor1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.264.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.265 rfe_tc_3205_sm70MisoCrcTempsensor1

Testcase to perform spi transaction crc for tempsensor1 ip e miso r2 fault injection test for sm70

4.265.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 623. rfe_tc_3205_sm70MisoCrcTempsensor1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 623. rfe_tc_3205_sm70MisoCrcTempsensor1 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.265.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.266 rfe_tc_3206_sm70MosiCrcTempsensor2

Testcase to perform spi transaction crc for tempsensor2 ip e mosi r2 fault injection test for sm70

4.266.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 624. rfe_tc_3206_sm70MosiCrcTempsensor2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.266.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.267 rfe_tc_3207_sm70MisoCrcTempsensor2

Testcase to perform spi transaction crc for tempsensor2 ip e miso r2 fault injection test for sm70

4.267.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 625. rfe_tc_3207_sm70MisoCrcTempsensor2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.267.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.268 rfe_tc_3208_sm70MosiCrcTempsensor3

Testcase to perform spi transaction crc for tempsensor3 ip e mosi r2 fault injection test for sm70

4.268.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 626. rfe_tc_3208_sm70MosiCrcTempsensor3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.268.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.269 rfe_tc_3209_sm70MisoCrcTempsensor3

Testcase to perform spi transaction crc for tempsensor3 ip e miso r2 fault injection test for sm70

4.269.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 627. rfe_tc_3209_sm70MisoCrcTempsensor3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.269.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.270 rfe_tc_3210_sm70MosiCrcTempsensor4

Testcase to perform spi transaction crc for tempsensor4 ip e mosi r2 fault injection test for sm70

4.270.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 628. rfe_tc_3210_sm70MosiCrcTempsensor4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 628. rfe_tc_3210_sm70MosiCrcTempsensor4 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.270.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
 - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
 - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.271 rfe_tc_3211_sm70MisoCrcTempsensor4

Testcase to perform spi transaction crc for tempsensor4 ip e miso r2 fault injection test for sm70

4.271.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 629. rfe_tc_3211_sm70MisoCrcTempsensor4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.271.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
 - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
 - RTS FCCU Init success
3. Clear RTS FCCU faults
 - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.272 rfe_tc_3212_sm70MosiCrcAdpll

Testcase to perform spi transaction crc for adpll ip e mosi r2 fault injection test for sm70

4.272.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 630. rfe_tc_3212_sm70MosiCrcAdpll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.272.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.273 rfe_tc_3213_sm70MisoCrcAdpll

Testcase to perform spi transaction crc for adpll ip e miso r2 fault injection test for sm70

4.273.1 Detailed Description

RFE SW shall protect the integrity of the frame by CRC

Table 631. rfe_tc_3213_sm70MisoCrcAdpll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311211

4.273.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.274 rfe_tc_3214_sm86FpuR1FaultHandling

Testcase to perform rfe m7 fpu error r1 fault injection and r2 promotion for sm86

4.274.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 632. rfe_tc_3214_sm86FpuR1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.274.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM

34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP_SM86_RFE_M7_FPU_error.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc614593

4.275 rfe_tc_3215_sm86FpuR2FaultPromotion

Testcase to perform rfe m7 fpu error r1 fault injection and r2 promotion for sm86

4.275.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 633. rfe_tc_3215_sm86FpuR2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.275.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM

34. call rfe_getFuSaFaultStatistics to check no faults active
- Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
- Fault is promoted to R2
- Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
- Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
- Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
- rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
- ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP_SM86_RFE_M7_FPU_error.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc614593

4.276 rfe_tc_3216_sm88DtcmEccR2FaultHandling

Testcase to perform ida cpuctrl cm7 it64 dt32 mem dtcm ecc r2 fault handling for sm88

4.276.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 634. rfe_tc_3216_sm88DtcmEccR2FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.276.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
- RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
- RTS FCCU Init success
3. Clear RTS FCCU faults
- ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
- rfe_configure() is success
6. update bistZeroHourReferenceData
- update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
- rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
- No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
- No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
- No Faults is promoted to R2
- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
- No Faults is promoted to R2
- Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
- No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
- No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
- No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
- No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
- No Faults is promoted to R2
- Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
- No Faults is promoted to R2
- Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.21 -IDA_CPUCTRL_CM7_IT64_DT32_MEM_x TCM.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582818

4.277 rfe_tc_3217_sm89ItcmR2FaultHandling

Testcase to perform ida cpuctrl cm7 it64 dt32 mem itcm eccr2 fault handling for sm89

4.277.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 635. rfe_tc_3217_sm89ltcmR2FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.277.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.21 -IDA_CPUCTRL_CM7_IT64_DT32_MEM_x TCM.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582818

4.278 rfe_tc_3218_sm92SoftwareFaultR2Handling

Testcase to perform rfe m7 core swt (software watchdog timer) r2 fault handling for sm92

4.278.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 636. rfe_tc_3218_sm92SoftwareFaultR2Handling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.278.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.92 - SM92_RFE_M7_CORE_SWT.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc600826

4.279 rfe_tc_3219_sm93FlexNocCrcR2FaultHandling

Testcase to perform ida rfe m7 flexnoc (crc) r2 fault handling for sm93

4.279.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 637. rfe_tc_3219_sm93FlexNocCrcR2FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.279.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.23 - SM93_IDA_RFE_M7_XBIC - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582840

4.280 rfe_tc_3220_sm93Apb2SpiCrcR2FaultHandling

Testcase to perform ida rfe m7 apb2spi (crc) r2 fault handling for sm93

4.280.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 638. rfe_tc_3220_sm93Apb2SpiCrcR2FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 638. rfe_tc_3220_sm93Apb2SpiCrcR2FaultHandling Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.280.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.23 - SM93_IDA_RFE_M7_XBIC - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582840

4.281 rfe_tc_3221_sm93PitR2FAultHandling

Testcase to perform ida rfe m7 pit (xbic) r2 fault handling for sm93

4.281.1 Detailed Description

The RFE SW shall provide fault status and fault statistics readout via API.

Table 639. rfe_tc_3221_sm93PitR2FAultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	3168361

4.281.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.23 - SM93_IDA_RFE_M7_XBIC - rev2.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582840

4.282 rfe_tc_3222_sm94MessageE2ER2FaultPromotion

Testcase to perform ida rfe m7 message e2e (r2 fh) special rfe driver code change is needed r1 fault handling and r2 promotion for sm94

4.282.1 Detailed Description

RFE SW shall notify radar application about communication fault via FCCU

Table 640. rfe_tc_3222_sm94MessageE2ER2FaultPromotion Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311213

4.282.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e

- rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
 - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.94 - SM94_IDA_RFE_M7_MESSAGE_E2E" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc617543

4.283 rfe_tc_3223_sm94MessageE2EClear

Testcase to perform ida rfe m7 message e2e (clean) special rfe driver code change is needed for sm94

4.283.1 Detailed Description

RFE SW shall notify radar application about communication fault via FCCU

Table 641. rfe_tc_3223_sm94MessageE2EClear Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311213

4.283.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
 - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
 - RTS FCCU Init success
3. Clear RTS FCCU faults
 - ErrorN is not asserted
4. call rfe_configure with default settings
 - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
 - rfe_configure() is success
6. update bistZeroHourReferenceData
 - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
 - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob

10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active

- Fault is not active for unmasked SM

34. call rfe_getFuSaFaultStatistics to check no faults active
 - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
 - Fault is promoted to R2
 - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
 - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
 - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
 - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
 - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.94 - SM94_IDA_RFE_M7_MESSAGE_E2E" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc617543

4.284 rfe_tc_3224_sm94MessageE2ER1FaultHandling

Testcase to perform ida rfe m7 message e2e (r1 fi) special rfe driver code change is needed r1 fault handling for sm94

4.284.1 Detailed Description

RFE SW shall notify radar application about communication fault via FCCU

Table 642. rfe_tc_3224_sm94MessageE2ER1FaultHandling Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1311213

4.284.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
 - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
 - RTS FCCU Init success
3. Clear RTS FCCU faults

- ErrorN is not asserted

4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getFuSaFaultStatistics to check no faults active
   - Fault count is incremented to 1 for unmasked SM
27. call rfe_radarCycleStart with 1 radar cycle count for 3rd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
28. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete

29. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
30. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is incremented to 2 for unmasked SM
31. call rfe_radarCycleStart with 1 radar cycle count for 4rd radar cycle by clearing Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
32. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
33. call rfe_getFuSaFaults to check no faults active
    - Fault is not active for unmasked SM
34. call rfe_getFuSaFaultStatistics to check no faults active
    - Fault count is cleared for unmasked SM
35. call rfe_radarCycleStart with radar cycle count greater than r1FaultPromotion
    - Fault is promoted to R2
    - Radar cycle requested are complete
36. call rfe_getFuSaFaults to check unmasked SM fault active
    - Fault is active for unmasked SM
37. call rfe_getFuSaFaultStatistics to check count matches with r1FaultPromotion for active fault
    - Fault count is equal to r1FaultPromotion
38. call rfe_getState to check state is rfe_state_fuSaFault_e
    - rfe state should match rfe_state_fuSaFault_e
39. Read ErrorN state in RTS FCCU
    - ErrorN is asserted
40. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.94 - SM94_IDA_RFE_M7_MESSAGE_E2E" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc617543

4.285 rfe_tc_3225_sm98SpiAccessMosiMcgen

Testcase to perform m7 spi access ana for mcgen ip e mosi r2 fault injection test for sm98

4.285.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 643. rfe_tc_3225_sm98SpiAccessMosiMcgen Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated

Table 643. rfe_tc_3225_sm98SpiAccessMosiMcgen Specification*...continued*

Property	Description
Satisfied Requirements	1280679

4.285.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete

24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.286 rfe_tc_3226_sm98SpiAccessMisoMcgen

Testcase to perform m7 spi access ana for mcgen ip e miso r2 fault injection test for sm98

4.286.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 644. rfe_tc_3226_sm98SpiAccessMisoMcgen Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.286.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success

8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.287 rfe_tc_3227_sm98SpiAccessMosiChirppll

Testcase to perform m7 spi access ana for chirppll ip e mosi r2 fault injection test for sm98

4.287.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 645. rfe_tc_3227_sm98SpiAccessMosiChirppll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.

Table 645. rfe_tc_3227_sm98SpiAccessMosiChirppll Specification*...continued*

Property	Description
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.287.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active

20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.288 rfe_tc_3228_sm98SpiAccessMisoChirppll

Testcase to perform m7 spi access ana for chirppll ip e miso r2 fault injection test for sm98

4.288.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 646. rfe_tc_3228_sm98SpiAccessMisoChirppll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.288.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults

   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.289 rfe_tc_3229_sm98SpiAccessMosiIsm

Testcase to perform m7 spi access ana for ism ip e mosi r2 fault injection test for sm98

4.289.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 647. rfe_tc_3229_sm98SpiAccessMosiIsm Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.289.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.290 rfe_tc_3230_sm98SpiAccessMisoIsm

Testcase to perform m7 spi access ana for ism ip e miso r2 fault injection test for sm98

4.290.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 648. rfe_tc_3230_sm98SpiAccessMisoIsm Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.290.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.291 rfe_tc_3231_sm98SpiAccessMosiSpim

Testcase to perform m7 spi access ana for spim ip e mosi r2 fault injection test for sm98

4.291.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 649. rfe_tc_3231_sm98SpiAccessMosiSpim Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.291.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.292 rfe_tc_3232_sm98SpiAccessMisoSpim

Testcase to perform m7 spi access ana for spim ip e miso r2 fault injection test for sm98

4.292.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 650. rfe_tc_3232_sm98SpiAccessMisoSpim Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 650. rfe_tc_3232_sm98SpiAccessMisoSpim Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.292.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.293 rfe_tc_3233_sm98SpiAccessMosiAtb

Testcase to perform m7 spi access ana for atb ip e mosi r2 fault injection test for sm98

4.293.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 651. rfe_tc_3233_sm98SpiAccessMosiAtb Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.293.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
 - rfe_configure() is success
6. update bistZeroHourReferenceData
 - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
 - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
 - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
 - No Faults is promoted to R2
 - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
 - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
 - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
 - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.294 rfe_tc_3234_sm98SpiAccessMisoAtb

Testcase to perform m7 spi access ana for atb ip e miso r2 fault injection test for sm98

4.294.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 652. rfe_tc_3234_sm98SpiAccessMisoAtb Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.294.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
 - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
 - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
 - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
 - No Faults is promoted to R2
 - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
 - No Faults is promoted to R2
 - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
 - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.295 rfe_tc_3235_sm98SpiAccessMosiTe

Testcase to perform m7 spi access ana for te ip e mosi r2 fault injection test for sm98

4.295.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 653. rfe_tc_3235_sm98SpiAccessMosiTe Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.295.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.296 rfe_tc_3236_sm98SpiAccessMisoTe

Testcase to perform m7 spi access ana for te ip e miso r2 fault injection test for sm98

4.296.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 654. rfe_tc_3236_sm98SpiAccessMisoTe Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.296.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.297 rfe_tc_3237_sm98SpiAccessMosiRxbist

Testcase to perform m7 spi access ana for rxbist ip e mosi r2 fault injection test for sm98

4.297.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 655. rfe_tc_3237_sm98SpiAccessMosiRxbist Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 655. rfe_tc_3237_sm98SpiAccessMosiRxbist Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.297.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.298 rfe_tc_3238_sm98SpiAccessMisoRxbist

Testcase to perform m7 spi access ana for rxbist ip e miso r2 fault injection test for sm98

4.298.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 656. rfe_tc_3238_sm98SpiAccessMisoRxbist Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.298.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.299 rfe_tc_3239_sm98SpiAccessMosiGbias

Testcase to perform m7 spi access ana for gbias ip e mosi r2 fault injection test for sm98

4.299.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 657. rfe_tc_3239_sm98SpiAccessMosiGbias Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.299.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

  - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
  - No Faults is promoted to R2
  - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
  - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
  - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
  - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
  - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
  - No Faults is promoted to R2
  - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
  - No Faults is promoted to R2
  - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
  - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.300 rfe_tc_3240_sm98SpiAccessMisoGbias

Testcase to perform m7 spi access ana for gbias ip e miso r2 fault injection test for sm98

4.300.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 658. rfe_tc_3240_sm98SpiAccessMisoGbias Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.300.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.301 rfe_tc_3241_sm98SpiAccessMosiGldo

Testcase to perform m7 spi access ana for gldo ip e mosi r2 fault injection test for sm98

4.301.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 659. rfe_tc_3241_sm98SpiAccessMosiGldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.301.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.302 rfe_tc_3242_sm98SpiAccessMisoGldo

Testcase to perform m7 spi access ana for gldo ip e miso r2 fault injection test for sm98

4.302.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 660. rfe_tc_3242_sm98SpiAccessMisoGldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 660. rfe_tc_3242_sm98SpiAccessMisoGldo Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.302.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.303 rfe_tc_3243_sm98SpiAccessMosiRcosc

Testcase to perform m7 spi access ana for rcosc ip e mosi r2 fault injection test for sm98

4.303.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 661. rfe_tc_3243_sm98SpiAccessMosiRcosc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.303.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.304 rfe_tc_3244_sm98SpiAccessMisoRcosc

Testcase to perform m7 spi access ana for rcosc ip e miso r2 fault injection test for sm98

4.304.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 662. rfe_tc_3244_sm98SpiAccessMisoRcosc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.304.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.305 rfe_tc_3245_sm98SpiAccessMosiLldo

Testcase to perform m7 spi access ana for lldo ip e mosi r2 fault injection test for sm98

4.305.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 663. rfe_tc_3245_sm98SpiAccessMosiLldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.305.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.306 rfe_tc_3246_sm98SpiAccessMisoLldo

Testcase to perform m7 spi access ana for lldo ip e miso r2 fault injection test for sm98

4.306.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 664. rfe_tc_3246_sm98SpiAccessMisoLldo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.306.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.307 rfe_tc_3247_sm98SpiAccessMosiLldopdc

Testcase to perform m7 spi access ana for lldopdc ip e mosi r2 fault injection test for sm98

4.307.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 665. rfe_tc_3247_sm98SpiAccessMosiLldopdc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 665. rfe_tc_3247_sm98SpiAccessMosiLldopdc Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.307.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.308 rfe_tc_3248_sm98SpiAccessMisoLldopdc

Testcase to perform m7 spi access ana for lldopdc ip e miso r2 fault injection test for sm98

4.308.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 666. rfe_tc_3248_sm98SpiAccessMisoLldopdc Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.308.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.309 rfe_tc_3249_sm98SpiAccessMosiLo

Testcase to perform m7 spi access ana for lo ip e mosi r2 fault injection test for sm98

4.309.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 667. rfe_tc_3249_sm98SpiAccessMosiLo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.309.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC. docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.310 rfe_tc_3250_sm98SpiAccessMisoLo

Testcase to perform m7 spi access ana for lo ip e miso r2 fault injection test for sm98

4.310.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 668. rfe_tc_3250_sm98SpiAccessMisoLo Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.310.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.311 rfe_tc_3251_sm98SpiAccessMosiAdc1

Testcase to perform m7 spi access ana for adc1 ip e mosi r2 fault injection test for sm98

4.311.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 669. rfe_tc_3251_sm98SpiAccessMosiAdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.311.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.312 rfe_tc_3252_sm98SpiAccessMisoAdc1

Testcase to perform m7 spi access ana for adc1 ip e miso r2 fault injection test for sm98

4.312.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 670. rfe_tc_3252_sm98SpiAccessMisoAdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 670. rfe_tc_3252_sm98SpiAccessMisoAdc1 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.312.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.313 rfe_tc_3253_sm98SpiAccessMosiAdc2

Testcase to perform m7 spi access ana for adc2 ip e mosi r2 fault injection test for sm98

4.313.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 671. rfe_tc_3253_sm98SpiAccessMosiAdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.313.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.314 rfe_tc_3254_sm98SpiAccessMisoAdc2

Testcase to perform m7 spi access ana for adc2 ip e miso r2 fault injection test for sm98

4.314.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 672. rfe_tc_3254_sm98SpiAccessMisoAdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.314.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.315 rfe_tc_3255_sm98SpiAccessMosiAdc3

Testcase to perform m7 spi access ana for adc3 ip e mosi r2 fault injection test for sm98

4.315.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 673. rfe_tc_3255_sm98SpiAccessMosiAdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.315.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.316 rfe_tc_3256_sm98SpiAccessMisoAdc3

Testcase to perform m7 spi access ana for adc3 ip e miso r2 fault injection test for sm98

4.316.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 674. rfe_tc_3256_sm98SpiAccessMisoAdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.316.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.317 rfe_tc_3257_sm98SpiAccessMosiAdc4

Testcase to perform m7 spi access ana for adc4 ip e mosi r2 fault injection test for sm98

4.317.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 675. rfe_tc_3257_sm98SpiAccessMosiAdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 675. rfe_tc_3257_sm98SpiAccessMosiAdc4 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.317.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.318 rfe_tc_3258_sm98SpiAccessMisoAdc4

Testcase to perform m7 spi access ana for adc4 ip e miso r2 fault injection test for sm98

4.318.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 676. rfe_tc_3258_sm98SpiAccessMisoAdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.318.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.319 rfe_tc_3259_sm98SpiAccessMosiPdc1

Testcase to perform m7 spi access ana for pdc1 ip e mosi r2 fault injection test for sm98

4.319.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 677. rfe_tc_3259_sm98SpiAccessMosiPdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.319.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.320 rfe_tc_3260_sm98SpiAccessMisoPdc1

Testcase to perform m7 spi access ana for pdc1 ip e miso r2 fault injection test for sm98

4.320.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 678. rfe_tc_3260_sm98SpiAccessMisoPdc1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.320.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.321 rfe_tc_3261_sm98SpiAccessMosiPdc2

Testcase to perform m7 spi access ana for pdc2 ip e mosi r2 fault injection test for sm98

4.321.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 679. rfe_tc_3261_sm98SpiAccessMosiPdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.321.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.322 rfe_tc_3262_sm98SpiAccessMisoPdc2

Testcase to perform m7 spi access ana for pdc2 ip e miso r2 fault injection test for sm98

4.322.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 680. rfe_tc_3262_sm98SpiAccessMisoPdc2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 680. rfe_tc_3262_sm98SpiAccessMisoPdc2 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.322.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.323 rfe_tc_3263_sm98SpiAccessMosiPdc3

Testcase to perform m7 spi access ana for pdc3 ip e mosi r2 fault injection test for sm98

4.323.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 681. rfe_tc_3263_sm98SpiAccessMosiPdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.323.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success

5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.324 rfe_tc_3264_sm98SpiAccessMisoPdc3

Testcase to perform m7 spi access ana for pdc3 ip e miso r2 fault injection test for sm98

4.324.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 682. rfe_tc_3264_sm98SpiAccessMisoPdc3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.324.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.325 rfe_tc_3265_sm98SpiAccessMosiPdc4

Testcase to perform m7 spi access ana for pdc4 ip e mosi r2 fault injection test for sm98

4.325.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 683. rfe_tc_3265_sm98SpiAccessMosiPdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.325.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.326 rfe_tc_3266_sm98SpiAccessMisoPdc4

Testcase to perform m7 spi access ana for pdc4 ip e miso r2 fault injection test for sm98

4.326.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 684. rfe_tc_3266_sm98SpiAccessMisoPdc4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.326.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.327 rfe_tc_3267_sm98SpiAccessMosiTx1

Testcase to perform m7 spi access ana for tx1 ip e mosi r2 fault injection test for sm98

4.327.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 685. rfe_tc_3267_sm98SpiAccessMosiTx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 685. rfe_tc_3267_sm98SpiAccessMosiTx1 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.327.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.328 rfe_tc_3268_sm98SpiAccessMisoTx1

Testcase to perform m7 spi access ana for tx1 ip e miso r2 fault injection test for sm98

4.328.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 686. rfe_tc_3268_sm98SpiAccessMisoTx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.328.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.329 rfe_tc_3269_sm98SpiAccessMosiTx2

Testcase to perform m7 spi access ana for tx2 ip e mosi r2 fault injection test for sm98

4.329.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 687. rfe_tc_3269_sm98SpiAccessMosiTx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.329.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.330 rfe_tc_3270_sm98SpiAccessMisoTx2

Testcase to perform m7 spi access ana for tx2 ip e miso r2 fault injection test for sm98

4.330.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 688. rfe_tc_3270_sm98SpiAccessMisoTx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.330.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.331 rfe_tc_3271_sm98SpiAccessMosiTx3

Testcase to perform m7 spi access ana for tx3 ip e mosi r2 fault injection test for sm98

4.331.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 689. rfe_tc_3271_sm98SpiAccessMosiTx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.331.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.332 rfe_tc_3272_sm98SpiAccessMisoTx3

Testcase to perform m7 spi access ana for tx3 ip e miso r2 fault injection test for sm98

4.332.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 690. rfe_tc_3272_sm98SpiAccessMisoTx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 690. rfe_tc_3272_sm98SpiAccessMisoTx3 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.332.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.333 rfe_tc_3273_sm98SpiAccessMosiTx4

Testcase to perform m7 spi access ana for tx4 ip e mosi r2 fault injection test for sm98

4.333.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 691. rfe_tc_3273_sm98SpiAccessMosiTx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.333.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.334 rfe_tc_3274_sm98SpiAccessMisoTx4

Testcase to perform m7 spi access ana for tx4 ip e miso r2 fault injection test for sm98

4.334.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 692. rfe_tc_3274_sm98SpiAccessMisoTx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.334.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.335 rfe_tc_3275_sm98SpiAccessMosiRx1

Testcase to perform m7 spi access ana for rx1 ip e mosi r2 fault injection test for sm98

4.335.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 693. rfe_tc_3275_sm98SpiAccessMosiRx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.335.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.336 rfe_tc_3276_sm98SpiAccessMisoRx1

Testcase to perform m7 spi access ana for rx1 ip e miso r2 fault injection test for sm98

4.336.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 694. rfe_tc_3276_sm98SpiAccessMisoRx1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.336.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.337 rfe_tc_3277_sm98SpiAccessMosiRx2

Testcase to perform m7 spi access ana for rx2 ip e mosi r2 fault injection test for sm98

4.337.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 695. rfe_tc_3277_sm98SpiAccessMosiRx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 695. rfe_tc_3277_sm98SpiAccessMosiRx2 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.337.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.338 rfe_tc_3278_sm98SpiAccessMisoRx2

Testcase to perform m7 spi access ana for rx2 ip e miso r2 fault injection test for sm98

4.338.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 696. rfe_tc_3278_sm98SpiAccessMisoRx2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.338.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.339 rfe_tc_3279_sm98SpiAccessMosiRx3

Testcase to perform m7 spi access ana for rx3 ip e mosi r2 fault injection test for sm98

4.339.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 697. rfe_tc_3279_sm98SpiAccessMosiRx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.339.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.340 rfe_tc_3280_sm98SpiAccessMisoRx3

Testcase to perform m7 spi access ana for rx3 ip e miso r2 fault injection test for sm98

4.340.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 698. rfe_tc_3280_sm98SpiAccessMisoRx3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.340.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.341 rfe_tc_3281_sm98SpiAccessMosiRx4

Testcase to perform m7 spi access ana for rx4 ip e mosi r2 fault injection test for sm98

4.341.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 699. rfe_tc_3281_sm98SpiAccessMosiRx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.341.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.342 rfe_tc_3282_sm98SpiAccessMisoRx4

Testcase to perform m7 spi access ana for rx4 ip e miso r2 fault injection test for sm98

4.342.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 700. rfe_tc_3282_sm98SpiAccessMisoRx4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 700. rfe_tc_3282_sm98SpiAccessMisoRx4 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.342.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.343 rfe_tc_3283_sm98SpiAccessMosiTempsensor1

Testcase to perform m7 spi access ana for tempsensor1 ip e mosi r2 fault injection test for sm98

4.343.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 701. rfe_tc_3283_sm98SpiAccessMosiTempsensor1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.343.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.344 rfe_tc_3284_sm98SpiAccessMisoTempsensor1

Testcase to perform m7 spi access ana for tempsensor1 ip e miso r2 fault injection test for sm98

4.344.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 702. rfe_tc_3284_sm98SpiAccessMisoTempsensor1 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.344.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

  - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.345 rfe_tc_3285_sm98SpiAccessMosiTempsensor2

Testcase to perform m7 spi access ana for tempsensor2 ip e mosi r2 fault injection test for sm98

4.345.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 703. rfe_tc_3285_sm98SpiAccessMosiTempsensor2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.345.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM

26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.346 rfe_tc_3286_sm98SpiAccessMisoTempsensor2

Testcase to perform m7 spi access ana for tempsensor2 ip e miso r2 fault injection test for sm98

4.346.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 704. rfe_tc_3286_sm98SpiAccessMisoTempsensor2 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.346.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.347 rfe_tc_3287_sm98SpiAccessMosiTempsensor3

Testcase to perform m7 spi access ana for tempsensor3 ip e miso r2 fault injection test for sm98

4.347.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 705. rfe_tc_3287_sm98SpiAccessMosiTempsensor3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 705. rfe_tc_3287_sm98SpiAccessMosiTempsensor3 Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.347.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.348 rfe_tc_3288_sm98SpiAccessMisoTempsensor3

Testcase to perform m7 spi access ana for tempsensor3 ip e mosi r2 fault injection test for sm98

4.348.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 706. rfe_tc_3288_sm98SpiAccessMisoTempsensor3 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.348.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.349 rfe_tc_3289_sm98SpiAccessMosiTempsensor4

Testcase to perform m7 spi access ana for tempsensor4 ip e mosi r2 fault injection test for sm98

4.349.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 707. rfe_tc_3289_sm98SpiAccessMosiTempsensor4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.349.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.350 rfe_tc_3290_sm98SpiAccessMisoTempsensor4

Testcase to perform m7 spi access ana for tempsensor4 ip e miso r2 fault injection test for sm98

4.350.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 708. rfe_tc_3290_sm98SpiAccessMisoTempsensor4 Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.350.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active

- Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.351 rfe_tc_3291_sm98SpiAccessMosiAdpll

Testcase to perform m7 spi access ana for adpll ip e mosi r2 fault injection test for sm98

4.351.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 709. rfe_tc_3291_sm98SpiAccessMosiAdpll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.351.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1

11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.352 rfe_tc_3292_sm98SpiAccessMisoAdpll

Testcase to perform m7 spi access ana for adpll ip e miso r2 fault injection test for sm98

4.352.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 710. rfe_tc_3292_sm98SpiAccessMisoAdpll Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification

Table 710. rfe_tc_3292_sm98SpiAccessMisoAdpll Specification*...continued*

Property	Description
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.352.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
    - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
    - No Faults is promoted to R2
    - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented

21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "STRX-OC_TP9.25 - SM70_SM98_SPI_transaction_access_CRC.docx" Collabnet link for the document: https://www.collabnet.nxp.com/sf/go/doc582936

4.353 rfe_tc_3293_sm99SpiAccessMosiSpi

Testcase to perform rfe access m7 spi access (special hook with error n) r2 fault handling for sm99

4.353.1 Detailed Description

If an SPI write fault is detected the RFE SW shall transition to FuSa Fault State.

Table 711. rfe_tc_3293_sm99SpiAccessMosiSpi Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280679

4.353.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings

- rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2
   - Radar cycle requested are complete
18. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
   - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
   - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
   - No Faults is promoted to R2
   - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
   - No Faults is promoted to R2
   - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
   - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "nan" Collabnet link for the document: nan

4.354 rfe_tc_3294_sm99SpiAccessMisoSpi

Testcase to perform rfe access m7 spi access (special hook no error n) r2 fault handling for sm99

4.354.1 Detailed Description

If an SPI read fault is detected the RFE SW shall transition to FuSa Fault State.

Table 712. rfe_tc_3294_sm99SpiAccessMisoSpi Specification

Property	Description
Pre	For executing this testcase rfeFuSaSysValFw rfe firmware build variant needs to be used.
Post	N/A
Test Level	qualification
Test Type	Functional
Test Technique	Blackbox
Pass Criteria	Verification Points are successful.
HW Environment	N/A
Execution	Automated
Satisfied Requirements	1280743

4.354.2 Test Procedure

Steps:

1. Check RFE initialization by calling rfe_sync().
   - RFE moves to rfe_state_initialized_e state
2. In rfe_state_initialized_e perform RTS FCCU initialization
   - RTS FCCU Init success
3. Clear RTS FCCU faults
   - ErrorN is not asserted
4. call rfe_configure with default settings
   - rfe_configure() is success
5. call rfe_getBistZeroHourReferenceData to get bist zero hour data
   - rfe_configure() is success
6. update bistZeroHourReferenceData
   - update bistZeroHourReferenceData is success
7. call rfe_configure with default settings
   - rfe_configure() is success
8. unmask Fusa Fault in rfeConfigure_modify config blob
9. configure r1FaultPromotion to 5 in rfeConfigure_modify config blob
10. configure 1st hook to read the register no fault is active in radar cycle 1
11. configure 2nd hook to perform fault injection in radar cycle 2
12. configure 3rd hook to read register to check fault injection is active in radar cycle 3
13. configure 4th hook to clear fault injection in radar cycle 4
14. Read register to check no faults are active
15. call rfe_getFuSaFaults to check no faults active
   - No Faults active
16. call rfe_getFuSaFaultStatistics to check no faults active
   - No Faults count is incremented
17. call rfe_radarCycleStart with 1 radar cycle count for first radar cycle
   - No Faults is promoted to R2

- Radar cycle requested are complete

18. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
19. call rfe_getFuSaFaults to check no faults active
    - No Faults active
20. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
21. call rfe_getFuSaFaults to check no faults active
    - No Faults active
22. call rfe_getFuSaFaultStatistics to check no faults active
    - No Faults count is incremented
23. call rfe_radarCycleStart with 1 radar cycle count for 2nd radar cycle with Fault injection settings
    - No Faults is promoted to R2
    - Radar cycle requested are complete
24. call rfe_radarCycleStart with 1 radar cycle count
    - No Faults is promoted to R2
    - Radar cycle requested are complete
25. call rfe_getFuSaFaults to check no faults active
    - Fault is active for unmasked SM
26. call rfe_getState to check rfe state is rfe_state_fuSaFault_e
27. Read the data from all requested hook to create testcase Pass/Fail. Register setting for fault Injection refer FuSa Sys Val Test Procedure document "nan" Collabnet link for the document: nan

Tables

Contents