Document Title	Specification of Flash Test
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	261
Document Classification	Standard

Document Version	2.1.0
Document Status	Final
Part of Release	4.1
Revision	2

Document Change History

Date	Version	Changed by	Change Description
31.10.2013	2.1.0	AUTOSAR Release Management	• SWS_FlsTst_00066: VARIABLE_CYCLIC_OR_ON_PRECONDITION in table removed • Editorial changes • Removed chapter(s) on change documentation
06.03.2013	2.0.0	AUTOSAR Administration	• Rework according to the new SWS_BSWGeneral document • Added Subchapter 3.x due to SWS General Rollout • Chapter 10: scope of configuration parameters are changed to “local” • SWS_FlsTst_00003: Rename MemMap.h to FlsTst_MemMap.h • SWS_FlsTst_00007: production errors removed • New chapters Production Errors and Extended Production Errors created
05.10.2011	1.2.0	AUTOSAR Administration	• SWS_FlsTst_00026: minor text change • Figure1: IRQ files removed • SWS_FlsTst_00052: parameter range modified • SWS_FlsTst_00053: minor text correction

Document Change History			
Date	**Version**	**Changed by**	**Change Description**

22.11.2010	1.1.0	AUTOSAR Administration	• FlsTst_BlockIdFgndType: type change to uint8-32
			• Limit range of the following parameters to max. value "0xFFFFFFFF"
			  • FlsTstBlockNumberBgnd:
			  • FlsTstBlockNumberFgnd:
			  • FlsTstBlockIndex:
			  • FlsTstBlockSize:
			  • FlsTstNumberOfTestedCells:
			  • FlsTstNumberOfTestedCellsAtomic:
			  • FlsTstTestIntervalIdEndValue:
			• FlsTst015 removed
			• ECUC_FlsTst_00119: configuration for each block
			• ECUC_FlsTst_00158: multiplicity changed to „1“.
			• FlsTstDemEventParameterRefs table included

07.12.2009	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Flash Test driver.
This Flash test module provides algorithm to test invariable memory. Invariable memory can be data/program flash, program SRAM, locked cache and is either embedded in the microcontroller or memory mapped connected to the microcontroller. For simplification the SW module is called Flash Test driver.
The test service can be executed at any time after MCU initialization and it is up to the user of the Flash Test Driver to choose the suitable test algorithm and the right execution place to fulfill the safety requirements of the system. The test service itself is dependant on the storage concept of the system. Therefore the availability of different test algorithms is configurable.
The Flash Test driver is intended to be integrated in the overall safety concept and will not provide the required diagnostic coverage on its own.

2 Acronyms and abbreviations

Acronyms and abbreviations that have a local scope are not contained in the AUTOSAR glossary. These appear in a local glossary below.

Acronym:	*Description:*
BSW	BasicSoftWare
PC	PreCompile
PB	PostBuild

Abbreviation:	*Description:*
DEM	Diagnostic Event Manager.
DET	Development Error Tracer.
MCU	Micro Controller Unit.
PLL	Phase Locked Loop.
ISR	Interrupt Service Routine.

The following table lists important Term and Definition, which are used within this document.

Term:	*Description:*
Background test	Background test is called periodically by a scheduler, and is interruptible. The test is split up over many scheduled tasks.
Foreground test	Foreground test is called via users call.
Invariable memory	Invariable memory can be program flash, program SRAM, locked cache and ROM
Test block	Defined memory area to be tested in foreground and background mode.
Test interval	Interval of a complete Flash test in background mode
Test time	Time for partial test defined within one scheduled task.
Signature	Unique calculation result of the content of a specific memory block
Memory block	Defined memory area
Partial test	Test to be executed in one scheduler interval
Test Interval Id	Identifier of a test interval, which shall be incremented by each start of a new test interval

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] General Requirements on SPAL
AUTOSAR_SRS_SPALGeneral.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[5] Specification of MCU Driver
AUTOSAR_SWS_MCUDriver.pdf

[6] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[7] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[8] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

[9] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [9] (SWS BSW General), which is also valid for Flash Test.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Flash Test.

4 Constraints and assumptions

4.1 Limitations

During Flash Test operation, the Flash area under test shall not be modified.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

The Flash Test module depends on the following modules:

- BSW scheduler is required to trigger main function in background mode

5.1 File structure

5.1.1 Code file structure

Note: Refer to SWS_BSWGeneral document [9].

5.1.2 Header file structure

[SWS_FlsTst_00003] ⌈The include structure for the source code of the Flash Test module shall be as follows:

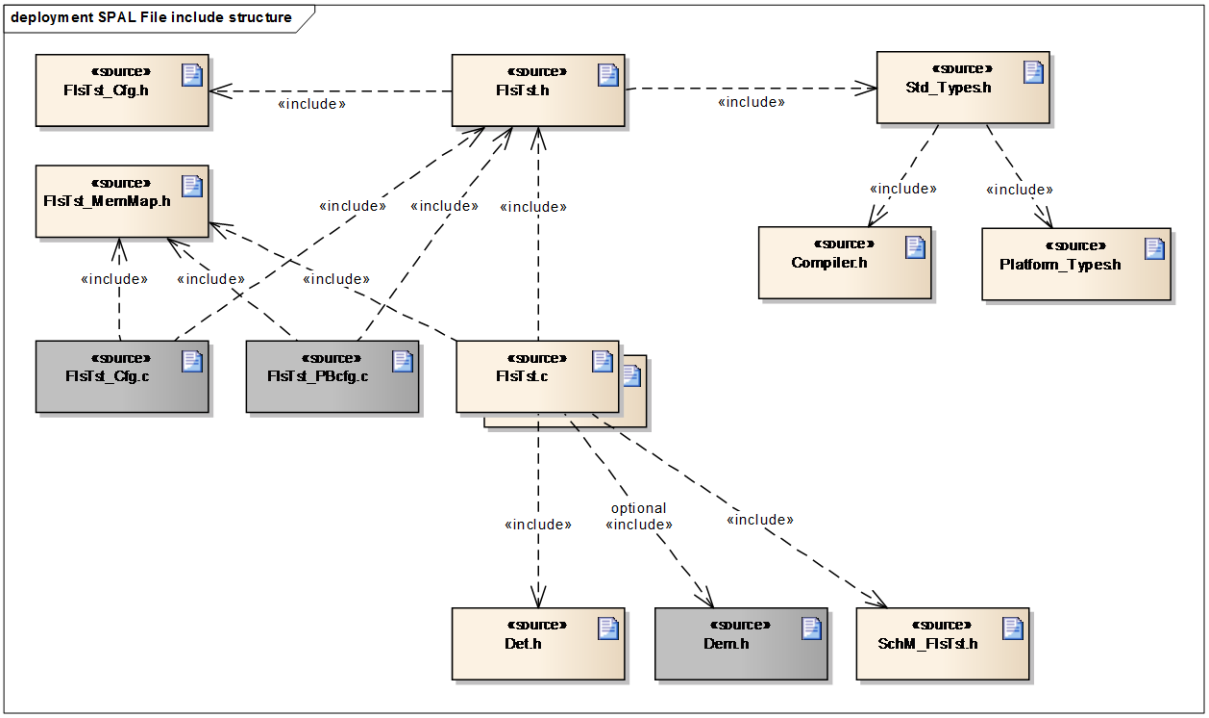

Figure 1: Header file structure

⌋(SRS_BSW_00380, SRS_BSW_00381, SRS_BSW_00412, SRS_BSW_00383, SRS_BSW_00435, SRS_BSW_00436)

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_FlsTst_00026
-	-	SWS_FlsTst_00029
-	-	SWS_FlsTst_00049
-	-	SWS_FlsTst_00052
-	-	SWS_FlsTst_00053
-	-	SWS_FlsTst_00067
-	-	SWS_FlsTst_00068
-	-	SWS_FlsTst_00070
-	-	SWS_FlsTst_00074
-	-	SWS_FlsTst_00075
-	-	SWS_FlsTst_00076
-	-	SWS_FlsTst_00081
-	-	SWS_FlsTst_00108
-	-	SWS_FlsTst_00117
-	-	SWS_FlsTst_00121
-	-	SWS_FlsTst_00138
-	-	SWS_FlsTst_00140
-	-	SWS_FlsTst_00142
-	-	SWS_FlsTst_00148
-	-	SWS_FlsTst_00156
-	-	SWS_FlsTst_00159
-	-	SWS_FlsTst_00161
-	-	SWS_FlsTst_00162
-	-	SWS_FlsTst_00164
BSW00421	-	SWS_FlsTst_00069
BSW00431	-	SWS_FlsTst_00166
BSW00434	-	SWS_FlsTst_00166
SRS_BSW_00003	All software modules shall provide version and identification information	SWS_FlsTst_00166
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_FlsTst_00166
SRS_BSW_00006	The source code of software modules above the æC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_FlsTst_00166

SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2004 Standard.	SWS_FlsTst_00166
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_FlsTst_00166
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_FlsTst_00166
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_FlsTst_00017
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_FlsTst_00166
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_FlsTst_00166
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_FlsTst_00166
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_FlsTst_00166
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_FlsTst_00166
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_FlsTst_00166
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_FlsTst_00166
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_FlsTst_00166
SRS_BSW_00300	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	SWS_FlsTst_00166
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the	SWS_FlsTst_00166

	necessary information	
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_FlsTst_00166
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_FlsTst_00016
SRS_BSW_00305	Data types naming convention	SWS_FlsTst_00166
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_FlsTst_00166
SRS_BSW_00307	Global variables naming convention	SWS_FlsTst_00166
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_FlsTst_00166
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_FlsTst_00166
SRS_BSW_00310	API naming convention	SWS_FlsTst_00166
SRS_BSW_00312	Shared code shall be reentrant	SWS_FlsTst_00166
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_FlsTst_00023, SWS_FlsTst_00033, SWS_FlsTst_00133
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_FlsTst_00166
SRS_BSW_00326	-	SWS_FlsTst_00166
SRS_BSW_00327	Error values naming convention	SWS_FlsTst_00166
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_FlsTst_00166
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_FlsTst_00166
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_FlsTst_00166
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_FlsTst_00166
SRS_BSW_00335	Status values naming convention	SWS_FlsTst_00166

SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_FlsTst_00027
SRS_BSW_00337	Classification of development errors	SWS_FlsTst_00007
SRS_BSW_00339	Reporting of production relevant error status	SWS_FlsTst_00042, SWS_FlsTst_00060, SWS_FlsTst_00112
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_FlsTst_00166
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_FlsTst_00166
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_FlsTst_00166
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_FlsTst_00166
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_FlsTst_00166
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_FlsTst_00166
SRS_BSW_00350	All AUTOSAR Basic Software Modules shall apply a specific naming rule for enabling/disabling the detection and reporting of development errors	SWS_FlsTst_00166
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_FlsTst_00166
SRS_BSW_00355	-	SWS_FlsTst_00100, SWS_FlsTst_00109
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_FlsTst_00063
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_FlsTst_00166
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_FlsTst_00166
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software	SWS_FlsTst_00166

	Modules	
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_FlsTst_00166
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_FlsTst_00166
SRS_BSW_00376	-	SWS_FlsTst_00066
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_FlsTst_00048
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_FlsTst_00166
SRS_BSW_00380	Configuration parameters being stored in memory shall be placed into separate c-files	SWS_FlsTst_00003
SRS_BSW_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_FlsTst_00003
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_FlsTst_00003
SRS_BSW_00385	List possible error notifications	SWS_FlsTst_00007
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_FlsTst_00023, SWS_FlsTst_00025, SWS_FlsTst_00033, SWS_FlsTst_00056, SWS_FlsTst_00059, SWS_FlsTst_00062, SWS_FlsTst_00065, SWS_FlsTst_00089, SWS_FlsTst_00091, SWS_FlsTst_00093, SWS_FlsTst_00114, SWS_FlsTst_00133
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_FlsTst_00166
SRS_BSW_00401	Documentation of multiple instances of configuration parameters shall be available	SWS_FlsTst_00166
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_FlsTst_00018, SWS_FlsTst_00019
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_FlsTst_00011
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_FlsTst_00044
SRS_BSW_00408	All AUTOSAR Basic Software Modules configuration	SWS_FlsTst_00166

	parameters shall be named according to a specific naming rule	
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_FlsTst_00007
SRS_BSW_00410	Compiler switches shall have defined values	SWS_FlsTst_00166
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_FlsTst_00044
SRS_BSW_00412	References to c-configuration parameters shall be placed into a separate h-file	SWS_FlsTst_00003
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_FlsTst_00166
SRS_BSW_00414	The init function may have parameters	SWS_FlsTst_00166
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_FlsTst_00166
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_FlsTst_00166
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_FlsTst_00166
SRS_BSW_00419	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file	SWS_FlsTst_00166
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_FlsTst_00166
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_FlsTst_00166
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_FlsTst_00166
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_FlsTst_00166

SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_FlsTst_00166
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_FlsTst_00166
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_FlsTst_00166
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_FlsTst_00166
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_FlsTst_00166
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_FlsTst_00166
SRS_BSW_00435	Each AUTOSAR Basic Software Module implementation .c shall include its respective header file SchM_.h	SWS_FlsTst_00003
SRS_BSW_00436	Each AUTOSAR Basic Software Module implementation *.c shall include the support memory mapping.	SWS_FlsTst_00003
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_FlsTst_00166
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_FlsTst_00018
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_FlsTst_00166
SRS_BSW_00440	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	SWS_FlsTst_00166
SRS_FlsTst_14208	Background Flash test shall be interruptible	SWS_FlsTst_00066, SWS_FlsTst_00071
SRS_FlsTst_14209	The memory to be tested shall be split into individual smaller pieces	SWS_FlsTst_00066, SWS_FlsTst_00071, SWS_FlsTst_00139
SRS_FlsTst_14211	Flash test execution status shall be available	SWS_FlsTst_00040, SWS_FlsTst_00041, SWS_FlsTst_00091
SRS_FlsTst_14212	Flash test execution completion shall be provided by a notification mechanism	SWS_FlsTst_00077, SWS_FlsTst_00078

SRS_FlsTst_14213	Calculation signature/checksum of a finalized test shall be provided	SWS_FlsTst_00055, SWS_FlsTst_00056, SWS_FlsTst_00057, SWS_FlsTst_00058, SWS_FlsTst_00059, SWS_FlsTst_00115, SWS_FlsTst_00116
SRS_FlsTst_14214	Service for Flash test execution result shall be provided.	SWS_FlsTst_00042, SWS_FlsTst_00043, SWS_FlsTst_00093, SWS_FlsTst_00112, SWS_FlsTst_00113, SWS_FlsTst_00114
SRS_FlsTst_14215	Suspend Flash test execution shall be possible	SWS_FlsTst_00034, SWS_FlsTst_00036, SWS_FlsTst_00037, SWS_FlsTst_00088
SRS_FlsTst_14216	Flash test execution shall be resumed when suspended	SWS_FlsTst_00035, SWS_FlsTst_00038, SWS_FlsTst_00039, SWS_FlsTst_00089
SRS_FlsTst_14217	Flash test execution shall be stopped when wanted	SWS_FlsTst_00030, SWS_FlsTst_00031, SWS_FlsTst_00032
SRS_FlsTst_14219	Foreground Flash test shall be available	SWS_FlsTst_00033, SWS_FlsTst_00050, SWS_FlsTst_00051, SWS_FlsTst_00137, SWS_FlsTst_00143, SWS_FlsTst_00149
SRS_FlsTst_14221	Memory Content to Be Tested Should Not be Valid During the Test	SWS_FlsTst_00166
SRS_FlsTst_14223	Flash Test Error details shall be reported	SWS_FlsTst_00060, SWS_FlsTst_00061, SWS_FlsTst_00062
SRS_FlsTst_14224	ECC Circuitry shall be tested	SWS_FlsTst_00063, SWS_FlsTst_00064, SWS_FlsTst_00065
SRS_FlsTst_14225	Each Flash test Interval shall have an Identifier	SWS_FlsTst_00153, SWS_FlsTst_00154, SWS_FlsTst_00155
SRS_SPAL_00157	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	SWS_FlsTst_00040, SWS_FlsTst_00042, SWS_FlsTst_00057, SWS_FlsTst_00060, SWS_FlsTst_00072, SWS_FlsTst_00073, SWS_FlsTst_00077, SWS_FlsTst_00112
SRS_SPAL_12057	All driver modules shall implement an interface for initialization	SWS_FlsTst_00017, SWS_FlsTst_00020
SRS_SPAL_12064	All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations	SWS_FlsTst_00166
SRS_SPAL_12067	All driver modules shall set their wake-up conditions depending on the selected operation mode	SWS_FlsTst_00166
SRS_SPAL_12068	The modules of the MCAL shall be initialized in a defined sequence	SWS_FlsTst_00166
SRS_SPAL_12069	All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason	SWS_FlsTst_00166
SRS_SPAL_12075	All drivers with random streaming capabilities shall use application buffers	SWS_FlsTst_00166

SRS_SPAL_12077	All drivers shall provide a non blocking implementation	SWS_FlsTst_00166
SRS_SPAL_12078	The drivers shall be coded in a way that is most efficient in terms of memory and runtime resources	SWS_FlsTst_00166
SRS_SPAL_12092	The driver's API shall be accessed by its handler or manager	SWS_FlsTst_00166
SRS_SPAL_12125	All driver modules shall only initialize the configured resources	SWS_FlsTst_00022
SRS_SPAL_12129	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	SWS_FlsTst_00166
SRS_SPAL_12163	All driver modules shall implement an interface for de-initialization	SWS_FlsTst_00027, SWS_FlsTst_00028
SRS_SPAL_12169	All driver modules that provide different operation modes shall provide a service for mode selection	SWS_FlsTst_00166
SRS_SPAL_12265	Configuration data shall be kept constant	SWS_FlsTst_00166
SRS_SPAL_12267	Wakeup sources shall be initialized by MCAL drivers and/or the MCU driver	SWS_FlsTst_00166
SRS_SPAL_12448	All driver modules shall have a specific behavior after a development error detection	SWS_FlsTst_00023, SWS_FlsTst_00025, SWS_FlsTst_00033, SWS_FlsTst_00039, SWS_FlsTst_00133
SRS_SPAL_12461	Specific rules regarding initialization of controller registers shall apply to all driver implementations	SWS_FlsTst_00166
SRS_SPAL_12462	The register initialization settings shall be published	SWS_FlsTst_00166
SRS_SPAL_12463	The register initialization settings shall be combined and forwarded	SWS_FlsTst_00166

7 Functional specification

7.1 General behavior

[SWS_FlsTst_00137] ⌈The Flash test module provides test execution services in background and foreground mode (see also chapter 2). ⌋(SRS_FlsTst_14219)

[SWS_FlsTst_00138] ⌈The memory blocks to be tested shall be configurable for background and foreground mode separately (see FlsTst103, FlsTst104). ⌋( )

[SWS_FlsTst_00139] ⌈In background mode the test blocks shall be tested in the same order they are configured in configuration structure. When all blocks are tested, one test interval is completed (see Figure 2). In background testing the partial tests shall be triggered via `FlsTst_MainFunction` (see SWS_FlsTst_00066). ⌋(SRS_FlsTst_14209)

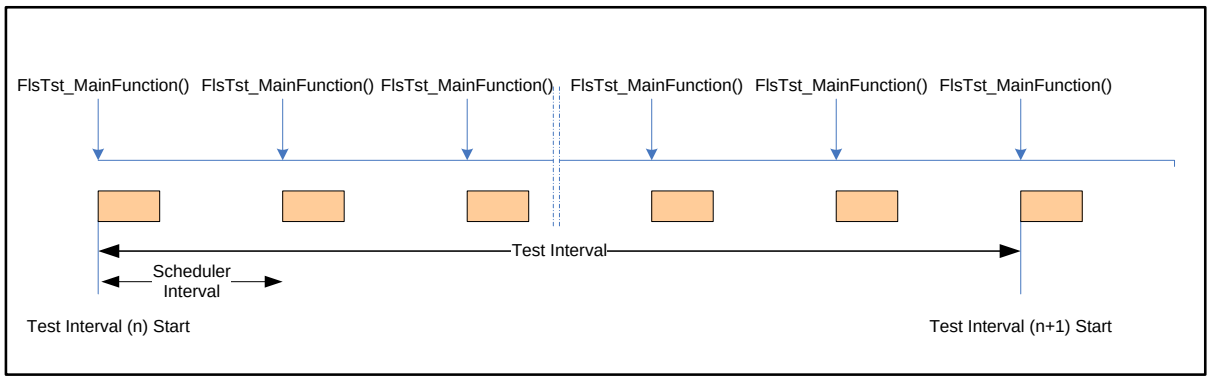

Figure 2: Background Test: Test Interval

[SWS_FlsTst_00140] ⌈The length of a partial test is defined by the number of tested cells, which shall be tested in one scheduled task. (see FlsTst119). The required time for a partial test without interruption is defined as “Test time”. ⌋( )

Note: The partial test can be interrupted by a higher priority task at any time, because the Flash test does not require atomic sequences. It is the responsibility of the user to ensure that the interruptible partial test is finished before the scheduler interval is started(See Figure 3).

[SWS_FlsTst_00142] ⌈A background test shall be aborted or suspended via the API services `FlsTst_Abort() or FlsTst_Suspended()`. The maximum latency time until the API call request is processed, shall be configurable (see FlsTst120). ⌋( )

[SWS_FlsTst_00156] ⌈Each Flash test Interval shall have an Identifier, which shall be incremented by each start of a new test interval in background mode.⌋ ( )

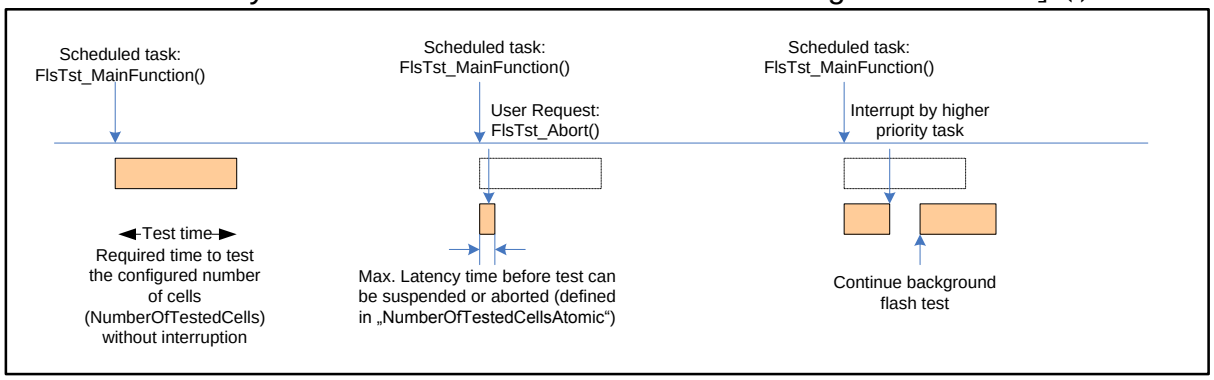

Figure 3: Background Test: Test Process

7.1.1 State Diagram

The Flash test driver states in background mode are described in Figure 4. The described states are driver states in background operation mode.

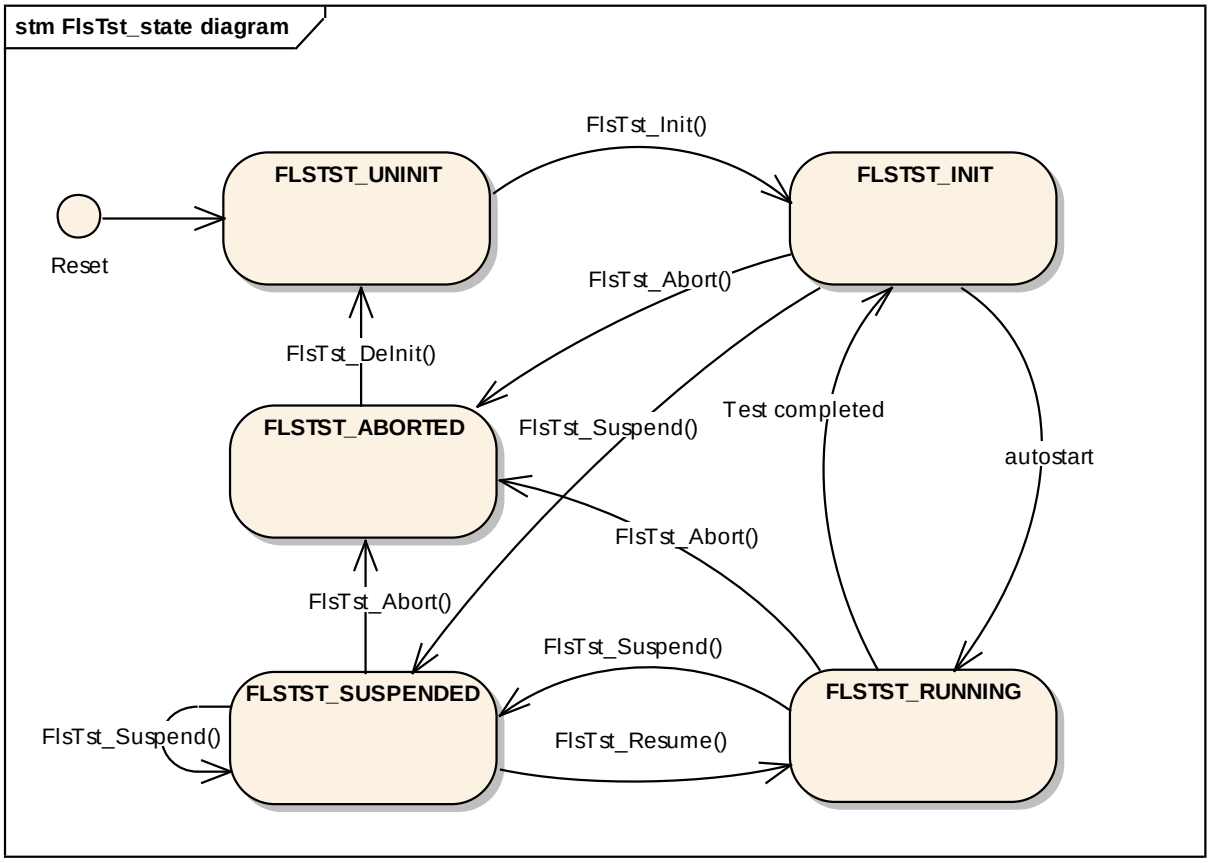

Figure 4: State Diagram – Background mode

[SWS_FlsTst_00143] ⌈Foreground tests are defined as synchronous tests which shall not be interrupted. The execution of Foreground tests is configurable (see

FlsTst086) and can be called after module initialization at any time. ⌋
(SRS_FlsTst_14219)

7.2 Error Classification

[SWS_FlsTst_00007] ⌈The following errors and exceptions shall be detectable by the Flash Test depending on its build version (development/production mode):

Type of error	*Error class*	*Related error code*	*Value [hex]*
Failure within Flash Test execution state	Development	FLSTST_E_STATE_FAILURE	0x01
API parameter out of specified range	Development	FLSTST_E_PARAM_INVALID	0x02
API service used without module initialization	Development	FLSTST_E_UNINIT	0x03
Flash Test module is already initialized	Development	FLSTST_E_ALREADY_INITIALIZED	0x04
For Variant PB: Configuration pointer is a NULL pointer	Development	FLSTST_E_PARAM_CONFIG	0x05
Pointer is a NULL pointer	Development	FLSTST_E_PARAM_POINTER	0x06

To get more details concerning error detection, refer to chapter 8 API specification. ⌋
(SRS_BSW_00337, SRS_BSW_00409, SRS_BSW_00385)

7.3 Production Errors

The Flash Test module does not specify any production errors.

7.4 Extended Production Errors

Type of error	*Related error code*	*Value [hex]*
Flash Failure	FLSTST_E_FLSTST_FAILURE	Assigned by DEM

7.5 Error Detection

[SWS_FlsTst_00011] ⌈The function FlsTst_Init shall be called first before calling any other Flash Test functions except the function FlsTst_GetCurrentState. If this sequence is not respected, the error code FLSTST_E_UNINIT shall be reported to the Development Error Tracer (if development error detection is enabled). ⌋
(SRS_BSW_00406)

7.6 Error Notification

Note: Refer to SWS_BSWGeneral document [9].

7.7 Version Check

Note: Refer to SWS_BSWGeneral document [9].

7.8 Debugging Support

The following requirements deal with the definition of variables and the description of debug information.

[SWS_FlsTst_00148] ⌈The state described in SWS_FlsTst_00048 shall be available for debugging. ⌋ ( )

8 API specification

8.1 Imported types

This chapter lists data type definitions for the included variables and constants.

[SWS_FlsTst_00016] ⌈

Module	*Imported Type*
Dem	Dem_EventIdType
	Dem_EventStatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋(SRS_BSW_00304)

8.2 Type definitions

8.2.1 FlsTst_ConfigType

[SWS_FlsTst_00018] ⌈

Name:	`FlsTst_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	implementation specific
Description:	This type of external data structure shall contain the initialization data for the Flash Test.	

⌋(SRS_BSW_00405, SRS_BSW_00438)

[SWS_FlsTst_00019] ⌈The type `FlsTst_ConfigType` shall denote the external data structure which contains the configuration data for the Flash Test module.

List of mandatory configuration parameters:

- Memory block definition to test in foreground mode
- Memory block definition to test in background mode
- Test sequence indication in background mode
- Hardware specific configuration⌋(SRS_BSW_00405)

8.2.2 FlsTst_StateType

[SWS_FlsTst_00048] ⌈

Name:	`FlsTst_StateType`	
Type:	`Enumeration`	
Range:	`FLSTST_UNINIT`	0x00: The Flash Test is not initialized or not usable.
	`FLSTST_INIT`	0x01: The Flash Test is initialized and ready to be started.

	FLSTST_RUNNING	0x02: The Flash Test is currently running.
	FLSTST_ABORTED	0x03: The Flash Test is aborted.
	FLSTST_SUSPENDED	0x04 The Flash Test is waiting to be resumed or is waiting to start foreground mode test
Description:	This is a state value returned by the API service FlsTst_GetCurrentState().	

⌋(SRS_BSW_00377)

[SWS_FlsTst_00049] ⌈For the type FlsTst_StateType, the enumeration value FLSTST_UNINIT shall be the default value after a reset. This enumeration value shall have the numeric value 0.⌋()

8.2.3 FlsTst_TestResultFgndType

[SWS_FlsTst_00052] ⌈

Name:	FlsTst_TestResultFgndType	
Type:	Enumeration	
Range:	FLSTST_NOT_TESTED	0x00: There is no result available.
	FLSTST_OK	0x01: The last Flash Test has been tested with OK result.
	FLSTST_NOT_OK	0x02: The last Flash Test has been tested with NOT_OK result.
Description:	Return type of API service FlsTst_GetResultFgnd().	

⌋()

[SWS_FlsTst_00053] ⌈For the type FlsTst_TestResultFgndType, the enumeration value FLSTST_NOT_TESTED shall be the default value after a reset. This enumeration value shall have the numeric value 0.⌋()

8.2.4 FlsTst_TestResultBgndType

[SWS_FlsTst_00153] ⌈

Name:	FlsTst_TestResultBgndType		
Type:	Structure		
Element:	uint8, uint16, uint32	0..<FlsTstTestIntervalIdEndValue>	current value of FlsTstTestIntervalId, which is incremented by each new start of an test interval.
	FlsTst_TestResultType	result	--
Description:	Return type of API service FlsTst_GetTestResultBgnd().		

⌋(SRS_FlsTst_14225)

[SWS_FlsTst_00154] ⌈For the type FlsTst_TestResultBgndType, the enumeration value FLSTST_RESULT_NOT_TESTED shall be the default value after a reset. This enumeration value shall have the numeric value 0.⌋(SRS_FlsTst_14225)

8.2.5 FlsTst_BlockIdFgndType

[SWS_FlsTst_00100] ⌈

Name:	FlsTst_BlockIdFgndType		
Type:	uint8, uint16, uint32		
Range:	0..<FlsTstBlockNumberFgnd > -1	--	The range is dependent on the number of Foreground Flash blocks defined in the configuration structure. The type shall be chosen depending on the MCU platform for best performance.
Description:	This type specifies the identification (ID) for a Flash block to be tested in foreground mode, which is configured in the configuration structure.		

⌋ (SRS_BSW_00355)

8.2.6 FlsTst_ErrorDetailsType

[SWS_FlsTst_00108] ⌈

Name:	FlsTst_ErrorDetailsType	
Type:	Structure	
Range:	implementation specific	implementation specific
Description:	This type shall specify implementation specific error information monitored in the Flash test module.	

⌋ ( )

8.2.7 FlsTst_TestSignatureFgndType

[SWS_FlsTst_00109] ⌈

Name:	FlsTst_TestSignatureFgndType	
Type:	Structure	
Range:	implementation specific	Implementation specific type
Description:	Type for test signature in foreground mode	

⌋ (SRS_BSW_00355)

8.2.8 FlsTst_TestSignatureBgndType

[SWS_FlsTst_00155] ⌈

Name:	FlsTst_TestSignatureBgndType		
Type:	Structure		
Element:	uint8, uint16, uint32	0..<FlsTstTestIntervalIdEndValue>	current value of FlsTstTestIntervalId, which is incremented by each new start of an test interval.
	uint8, uint16, uint32	Implementation specific	It represents the signature value of the last completed test interval. Value might be generated from several

			block signatures.
Description:	Type for test signature in background mode.		

⌋(SRS_FlsTst_14225)

8.2.9 FlsTst_TestResultType

[SWS_FlsTst_00164] ⌈

Name:	FlsTst_TestResultType	
Type:	Enumeration	
Range:	FLSTST_RESULT_NOT_TESTED	There is no test result available.
	FLSTST_RESULT_OK	The last Flash Test interval has been tested with OK result
	FLSTST_RESULT_NOT_OK	The last Flash Test interval has been tested with NOT-OK result.
Description:	--	

⌋( )

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 FlsTst_Init

[SWS_FlsTst_00017] ⌈

Service name:	FlsTst_Init	
Syntax:	void FlsTst_Init(const FlsTst_ConfigType* ConfigPtr)	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to configuration set
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service for Flash Test initialization.	

⌋(SRS_BSW_00101, SRS_SPAL_12057)

[SWS_FlsTst_00020] ⌈The function FlsTst_Init shall initialize all Flash Test relevant registers and global variables and change the execution state to FLSTST_INIT.⌋(SRS_SPAL_12057)

[SWS_FlsTst_00022] ⌈The function `FlsTst_Init` shall only initialize the configured resources and shall not touch resources that are not configured in the configuration file. ⌋ (SRS_SPAL_12125)

[SWS_FlsTst_00023] ⌈If development error detection is enabled for the Flash Test module, the function `FlsTst_Init` shall raise development error `FLSTST_E_PARAM_CONFIG` if `ConfigPtr` is a null pointer. This is applicable for Variant PB only (see also SWS_FlsTst_00026). The function shall be left without any action. ⌋ (SRS_BSW_00323, SRS_BSW_00386, SRS_SPAL_12448)

[SWS_FlsTst_00025] ⌈If development error detection is enabled, calling the routine `FlsTst_Init` while the Flash Test driver is already initialized shall cause development error `FLSTST_E_ALREADY_INITIALIZED`. The function shall be left without any action. ⌋ (SRS_BSW_00386, SRS_SPAL_12448)

Note: The `FlsTst_Init` function shall be called only once after a reset, unless an `FlsTst_DeInit` call is made before calling `FlsTst_Init` again.

[SWS_FlsTst_00026] ⌈For Variant PC a NULL pointer shall be passed to the initialization routine. In this case the check for this NULL pointer has to be omitted. ⌋ ( )

8.3.2 FlsTst_DeInit

[SWS_FlsTst_00027] ⌈

Service name:	FlsTst_DeInit

Syntax:	`void FlsTst_DeInit(`
	`  void`
	`)`

Service ID[hex]:	0x01
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Service for Flash Test De-Initialization.

⌋ (SRS_BSW_00336, SRS_SPAL_12163)

[SWS_FlsTst_00028] ⌈The function `FlsTst_DeInit` shall de-initialize all Flash Test relevant registers and global variables that were initialized by `FlsTst_Init`. ⌋ (SRS_SPAL_12163)

[SWS_FlsTst_00029] ⌈The function `FlsTst_DeInit` shall set the Flash Test module state to `FLSTST_UNINIT`.⌋()

8.3.3 FlsTst_StartFgnd

[SWS_FlsTst_00149] ⌈

Service name:	FlsTst_StartFgnd	
Syntax:	`Std_ReturnType FlsTst_StartFgnd( FlsTst_BlockIdFgndType FgndBlockId )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FgndBlockId	Number of the foreground test to be executed. This is dependent on configuration.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Foreground test processed E_NOT_OK: Foreground test not accepted
Description:	Service for executing foreground Flash Test.	

⌋(SRS_FlsTst_14219)

[SWS_FlsTst_00050] ⌈The function `FlsTst_StartFgnd` is only applicable for Foreground mode Flash Test operation.⌋(SRS_FlsTst_14219)

[SWS_FlsTst_00051] ⌈The function `FlsTst_StartFgnd` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_StartFgndApi`.⌋(SRS_FlsTst_14219)

[SWS_FlsTst_00033] ⌈If development error detection is enabled and the parameter `FgndBlockId` is out of range, the DET error value `FLSTST_E_PARAM_INVALID` shall be raised and the function shall return without any action with return value `E_NOT_OK`.⌋(SRS_BSW_00323, SRS_BSW_00386, SRS_SPAL_12448, SRS_FlsTst_14219)

8.3.4 FlsTst_Abort

[SWS_FlsTst_00030] ⌈

Service name:	FlsTst_Abort
Syntax:	`void FlsTst_Abort( void )`
Service ID[hex]:	0x03
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters	None

(inout):	
Parameters (out):	None
Return value:	None
Description:	Service for aborting the Flash Test.

⌋ (SRS_FlsTst_14217)

[SWS_FlsTst_00031] ⌈This function shall abort Flash test background operation and set the state to `FLSTST_ABORTED`. When the `FlsTst_Abort` function is called, `FlsTst_MainFunction` shall finish the current atomic sequence it is running. ⌋ (SRS_FlsTst_14217)

[SWS_FlsTst_00032] ⌈After an `FlsTst_Abort` call, `FlsTst_MainFunction` shall not begin testing again when called by the scheduler until after a complete re-initialization of the Flash test module. ⌋ (SRS_FlsTst_14217)

8.3.5 FlsTst_Suspend

[SWS_FlsTst_00034] ⌈

Service name:	FlsTst_Suspend

Syntax:	`void FlsTst_Suspend(`
	`  void`
	`)`

Service ID[hex]:	0x04
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Service for suspending current operation of the Flash Test, until FlsTst_Resume is called.

⌋ (SRS_FlsTst_14215)

[SWS_FlsTst_00036] ⌈The function `FlsTst_Suspend` is only applicable for Background mode Flash Test operation. ⌋ (SRS_FlsTst_14215)

[SWS_FlsTst_00037] ⌈The function `FlsTst_Suspend` shall set the Flash Test execution state to `FLSTST_SUSPENDED` in case the execution state was `FLSTST_RUNNING` or `FLSTST_INIT`. ⌋ (SRS_FlsTst_14215)

[SWS_FlsTst_00088] ⌈The function `FlsTst_Suspend` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_SuspendResumeApi`. ⌋ (SRS_FlsTst_14215)

8.3.6 FlsTst_Resume

[SWS_FlsTst_00035] ⌈

Service name:	FlsTst_Resume

Syntax:	`void FlsTst_Resume(`
	`  void`
	`)`

Service ID[hex]:	0x05
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Service for continuing the Flash Test at the point it was suspended.

⌋ (SRS_FlsTst_14216)

[SWS_FlsTst_00038] ⌈The function `FlsTst_Resume` shall change the execution state to `FLSTST_RUNNING` when commanded to continue and the current execution state is `FLSTST_SUSPENDED`. ⌋ (SRS_FlsTst_14216)

[SWS_FlsTst_00039] ⌈If development error detection is enabled and the execution state of the Flash Test module is not `FLSTST_SUSPENDED`, the Flash Test module shall report the error value `FLSTST_E_STATE_FAILURE` to the DET, and then immediately return from the function. ⌋ (SRS_SPAL_12448, SRS_FlsTst_14216)

[SWS_FlsTst_00162] ⌈The function `FlsTst_Resume` is only applicable for Background mode Flash Test operation. ⌋ ( )

[SWS_FlsTst_00089] ⌈The function `FlsTst_Resume` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_SuspendResumeApi`. ⌋ (SRS_BSW_00386, SRS_FlsTst_14216)

8.3.7 FlsTst_GetCurrentState

[SWS_FlsTst_00040] ⌈

Service name:	FlsTst_GetCurrentState

Syntax:	`FlsTst_StateType FlsTst_GetCurrentState(`
	`  void`
	`)`

Service ID[hex]:	0x06
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters	None

(inout):		
Parameters (out):	None	
Return value:	FlsTst_StateType	FLSTST_UNINIT The Flash Test is not initialized or not usable. FLSTST_INIT The Flash Test is initialized and ready to be started. FLSTST_RUNNING The Flash Test is currently running. FLSTST_ABORTED The Flash Test is aborted. FLSTST_SUSPENDED The Flash Test is waiting to be resumed or is waiting to start forground mode test
Description:	Service returns the current Flash Test exection state.	

⌋(SRS_SPAL_00157, SRS_FlsTst_14211)

[SWS_FlsTst_00041] ⌈The function `FlsTst_GetCurrentState` shall return the current Flash Test execution state. ⌋(SRS_FlsTst_14211)

[SWS_FlsTst_00091] ⌈The function `FlsTst_GetCurrentState` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_GetCurrentStateApi`. ⌋(SRS_BSW_00386, SRS_FlsTst_14211)

8.3.8 FlsTst_GetTestResultBgnd

[SWS_FlsTst_00042] ⌈

Service name:	FlsTst_GetTestResultBgnd	
Syntax:	`FlsTst_TestResultBgndType FlsTst_GetTestResultBgnd( void )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	FlsTst_TestResultBgndType	See type definition
Description:	Service returns the Background Flash Test result.	

⌋(SRS_BSW_00339, SRS_SPAL_00157, SRS_FlsTst_14214)

[SWS_FlsTst_00043] ⌈The function `FlsTst_GetTestResultBgnd` shall return the Flash test result and Test Interval Id of the last background test. ⌋(SRS_FlsTst_14214)

[SWS_FlsTst_00093] ⌈The function `FlsTst_GetTestResultBgnd` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_GetTestResultBgndApi`. ⌋(SRS_BSW_00386, SRS_FlsTst_14214)

8.3.9 FlsTst_GetTestResultFgnd

[SWS_FlsTst_00112] ⌈

Service name:	FlsTst_GetTestResultFgnd	
Syntax:	`FlsTst_TestResultFgndType FlsTst_GetTestResultFgnd( void )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	FlsTst_TestResultFgndType	See type definition
Description:	Service returns the Foreground Flash Test result.	

⌋ (SRS_BSW_00339, SRS_SPAL_00157, SRS_FlsTst_14214)

[SWS_FlsTst_00113] ⌈The function `FlsTst_GetTestResultFgnd` shall return the Flash test result of the last foreground test. ⌋ (SRS_FlsTst_14214)

[SWS_FlsTst_00114] ⌈The function `FlsTst_GetTestResultFgnd` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_GetTestResultFgndApi`. ⌋ (SRS_BSW_00386, SRS_FlsTst_14214)

8.3.10 FlsTst_GetVersionInfo

[SWS_FlsTst_00044] ⌈

Service name:	FlsTst_GetVersionInfo	
Syntax:	`void FlsTst_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Service returns the version information of this module.	

⌋ (SRS_BSW_00407, SRS_BSW_00411)

-

[SWS_FlsTst_00133] ⌈If development error detection is enabled for the Flash Test module, the function `FlsTst_GetVersionInfo` shall raise development error `FLSTST_E_PARAM_POINTER` if parameter versioninfo is a null pointer. ⌋ (SRS_BSW_00323, SRS_BSW_00386, SRS_SPAL_12448)

8.3.11 FlsTst_GetTestSignatureBgnd

[SWS_FlsTst_00054] ⌈

Service name:	FlsTst_GetTestSignatureBgnd	
Syntax:	`FlsTst_TestSignatureBgndType FlsTst_GetTestSignatureBgnd( void )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	FlsTst_TestSignatureBgndType	See type definition
Description:	Service returns the Flash Test result in background mode.	

⌋ (SRS_FlsTst_14213, SRS_SPAL_00157)

[SWS_FlsTst_00055] ⌈The function `FlsTst_GetTestSignatureBgnd` shall return the signature and Test Interval Id of the last background test. ⌋ (SRS_FlsTst_14213)

[SWS_FlsTst_00056] ⌈The function `FlsTst_GetTestSignatureBgnd` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_GetTestSignatureBgndApi`. ⌋ (SRS_BSW_00386, SRS_FlsTst_14213)

[SWS_FlsTst_00115] ⌈If no signature is available, the function `FlsTst_GetTestSignatureBgnd` shall return the default value “0x0”. ⌋ (SRS_FlsTst_14213)

8.3.12 FlsTst_GetTestSignatureFgnd

[SWS_FlsTst_00057] ⌈

Service name:	FlsTst_GetTestSignatureFgnd	
Syntax:	`FlsTst_TestSignatureFgndType FlsTst_GetTestSignatureFgnd( void )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	FlsTst_TestSignatureFgndType	See type definition
Description:	Service returns the Flash Test result in foreground mode.	

⌋ (SRS_FlsTst_14213, SRS_SPAL_00157)

[SWS_FlsTst_00058] ⌈The function `FlsTst_GetTestSignatureFgnd` shall return the signature of the last foreground test. ⌋(SRS_FlsTst_14213)

[SWS_FlsTst_00059] ⌈The function `FlsTst_GetTestSignatureFgnd` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_GetTestSignatureFgndApi`. ⌋(SRS_BSW_00386, SRS_FlsTst_14213)

[SWS_FlsTst_00116] ⌈If no signature is available, the function `FlsTst_GetTestSignatureFgnd` shall return the default value “0x0”. ⌋(SRS_FlsTst_14213)

8.3.13 FlsTst_GetErrorDetails

[SWS_FlsTst_00060] ⌈

Service name:	FlsTst_GetErrorDetails	
Syntax:	`FlsTst_ErrorDetailsType FlsTst_GetErrorDetails( void )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	FlsTst_ErrorDetailsType	See type definition
Description:	Service returns error detais monitored from the Flash module.	

⌋(SRS_BSW_00339, SRS_SPAL_00157, SRS_FlsTst_14223)

[SWS_FlsTst_00061] ⌈The function `FlsTst_GetErrorDetails` shall return the error details monitored from the Flash Test driver. ⌋(SRS_FlsTst_14223)

[SWS_FlsTst_00062] ⌈The function `FlsTst_GetErrorDetails` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_GetErrorDetailsApi`. ⌋(SRS_BSW_00386, SRS_FlsTst_14223)

8.3.14 FlsTst_TestEcc

[SWS_FlsTst_00063] ⌈

Service name:	FlsTst_TestEcc
Syntax:	`Std_ReturnType FlsTst_TestEcc( void )`
Service ID[hex]:	0x0c
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	see type definition
Description:	Service executes a test of ECC hardware. This is only applicable in case the hardware provices such functionality.	

⌋(SRS_BSW_00357, SRS_FlsTst_14224)

[SWS_FlsTst_00064] ⌈The function `FlsTst_TestEcc` shall execute a test of the ECC circuitry. ⌋(SRS_FlsTst_14224)

[SWS_FlsTst_00065] ⌈The function `FlsTst_TestEcc` shall be pre compile time configurable On/Off by the configuration parameter: `FlsTst_TestEccApi`. ⌋(SRS_BSW_00386, SRS_FlsTst_14224)

8.4 Callback notifications

Since the Flash Test is a driver module, it does not provide any callback functions for lower layer modules.

8.5 Scheduled functions

8.5.1 FlsTst_MainFunction

[SWS_FlsTst_00066] ⌈

Service name:	FlsTst_MainFunction

Syntax:	`void FlsTst_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x0d
Description:	Service for executing the Flash Test in background mode.

⌋(SRS_BSW_00376, SRS_FlsTst_14208, SRS_FlsTst_14209)

[SWS_FlsTst_00067] ⌈The function `FlsTst_MainFunction` shall test the defined flash blocks in background mode, starting with the first flash block in the FlsTstConfigParams. ⌋()

[SWS_FlsTst_00068] ⌈The function `FlsTst_MainFunction` shall set the Flash Test execution state from `FLSTST_INIT` to `FLSTST_RUNNING` when calling the function the first time after initialization or after a complete test interval. ⌋()

[SWS_FlsTst_00069] ⌈When FlsTstTestResultSignature is true, the function `FlsTst_MainFunction` shall provide the test signatures of all blocks within a test interval. ⌋ (BSW00421)

[SWS_FlsTst_00161] ⌈When FlsTstTestResultSignature is disabled, the function `FlsTst_MainFunction` shall set the overall result status to `FLSTST_RESULT_OK` if all blocks are tested with result status OK. If at least one block test result is not ok, then the function shall set the overall test result status to `FLSTST_RESULT_NOT_OK` regardless whether all blocks are already tested or not and report the production error `FLSTST_E_FLSTST_FAILURE` to the DEM. ⌋()

[SWS_FlsTst_00070] ⌈After the function `FlsTst_MainFunction` has completed testing all flash blocks, the next call of the function `FlsTst_MainFunction` shall restart the test from the beginning. ⌋()

[SWS_FlsTst_00071] ⌈The function `FlsTst_MainFunction` shall test a defined number of flash cells within one call. The defined number is specified by configuration (see FlsTst119). ⌋ (SRS_FlsTst_14208, SRS_FlsTst_14209)

[SWS_FlsTst_00117] ⌈The function `FlsTst_MainFunction` shall test a defined number of flash cells without checking user request for Abort or Suspend. The defined number is specified by configuration (see FlsTst120). ⌋()

[SWS_FlsTst_00121] ⌈The function `FlsTst_MainFunction` shall increment the Test Interval Id by 1 before start of a new test interval. The first test interval shall have the Test Interval Id = “0”. If the end value = `FlsTstIntervalIdEndValue` is reached, Test Interval Id shall start with value “0” again. The value shall be provided as part of the return values of `FlsTst_GetTestResultBgnd and FlsTst_GetTestSignatureBgnd.` ⌋()

8.6 Expected Interfaces

In this chapter all interfaces required from other modules are listed.

8.6.1 Mandatory Interfaces

This chapter defines all interfaces that are required to fulfill the core functionality of the module.

[SWS_FlsTst_00072] ⌈

API function	*Description*
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main

	function. OBD Events Suppression shall be ignored for this computation.

⌋(SRS_SPAL_00157)

8.6.2 Optional Interfaces

This chapter defines all interfaces that are required to fulfill an optional functionality of the module.

[SWS_FlsTst_00073] ⌈

API function	***Description***
Det_ReportError	Service to report development errors.

⌋(SRS_SPAL_00157)

8.6.3 Configurable interfaces

In this chapter, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of these kinds of interfaces are not fixed because they are configurable.

[SWS_FlsTst_00074] ⌈The callback notifications shall be configurable as function pointers within the initialization data structure (FlsTst_ConfigType). ⌋( )

[SWS_FlsTst_00075] ⌈The callback notifications shall have no parameters and no return value. ⌋( )

[SWS_FlsTst_00076] ⌈If a callback notification is configured as null pointer, the Flash Test module shall not execute the callback. ⌋( )

8.6.3.1 FlsTst_TestCompleted Notification

[SWS_FlsTst_00077] ⌈

Service name:	FlsTst_TestCompletedNotification

Syntax:	`void FlsTst_TestCompletedNotification(`
	`  void`
	`)`

Service ID[hex]:	0x0e
Sync/Async:	Synchronous
Reentrancy:	Don't care
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	The function FlsTst_TestCompleted shall be called every time when a complete

	test cycle had been tested.

⌋(SRS_SPAL_00157, SRS_FlsTst_14212)

[SWS_FlsTst_00078] ⌈The Flash Test module shall call the callback notification `FlsTst_TestCompleted` every time when it has tested a complete test cycle of a flash test in background mode. ⌋(SRS_FlsTst_14212)

[SWS_FlsTst_00159] ⌈The call of function `FlsTst_TestCompleted` shall be pre compile time configurable On/Off by the configuration parameter `FlsTstTestCompletedNotificationSupported`. ⌋()

9 Sequence diagrams

9.1 Initialization

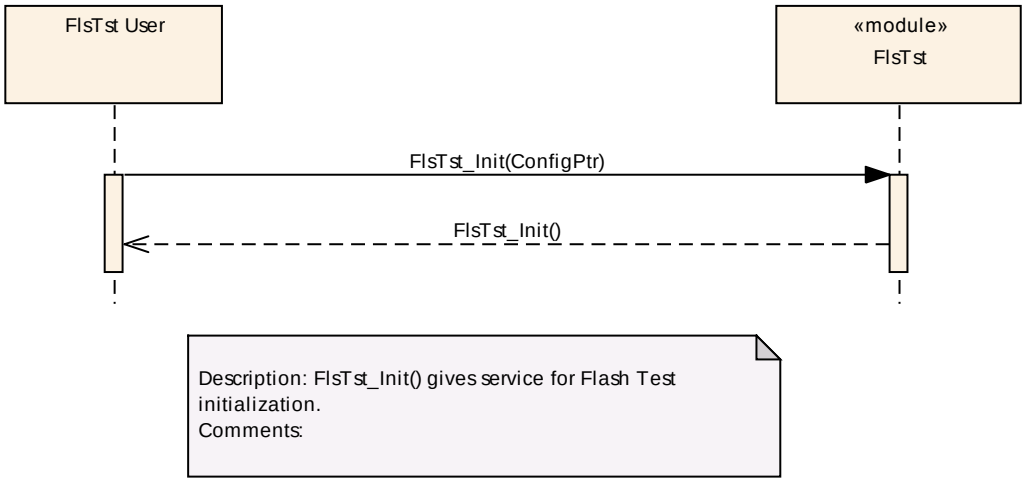

Figure 5: Flash test driver initialization

9.2 De-initialization

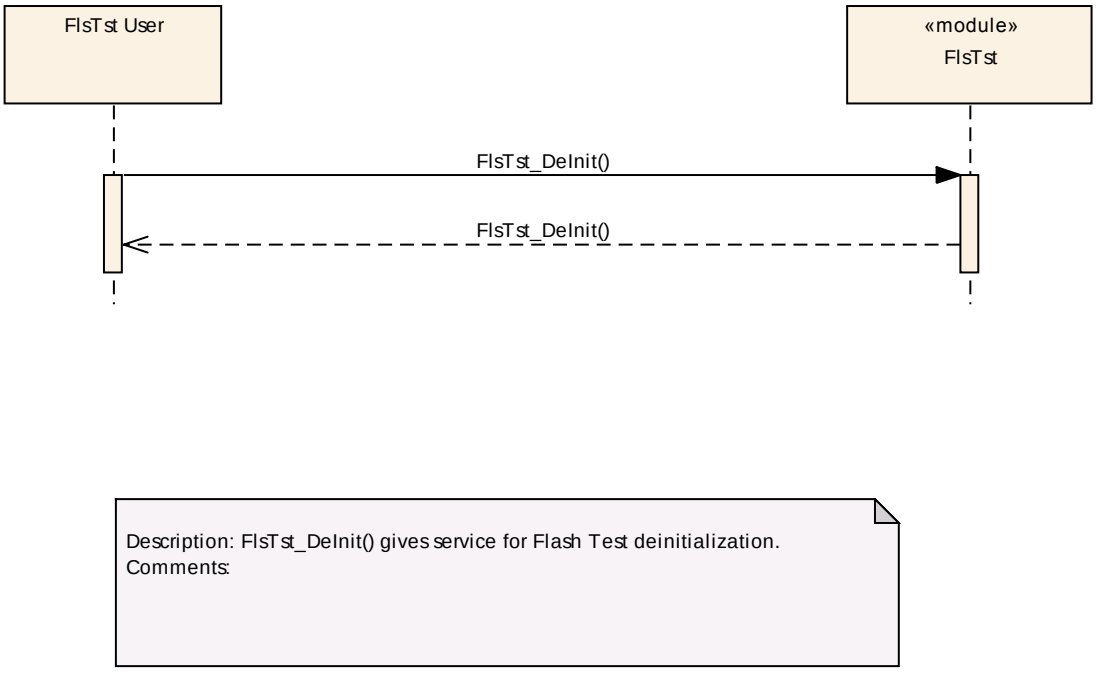

Figure 6: Flash test driver de-initialization

9.3 Background Test

9.3.1 Test Result Calculation within Flash test driver

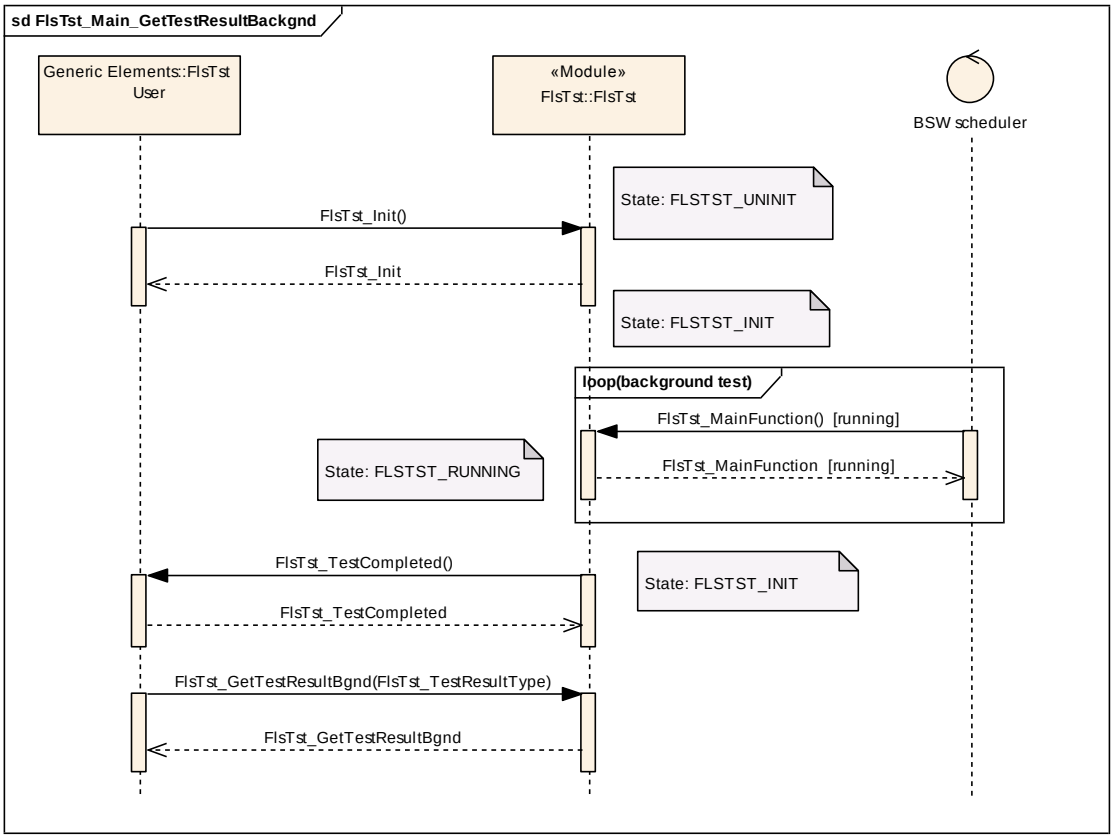

Figure 7: Background Test – Test result calculation in Flash test driver

9.3.2 Test signature provided to caller

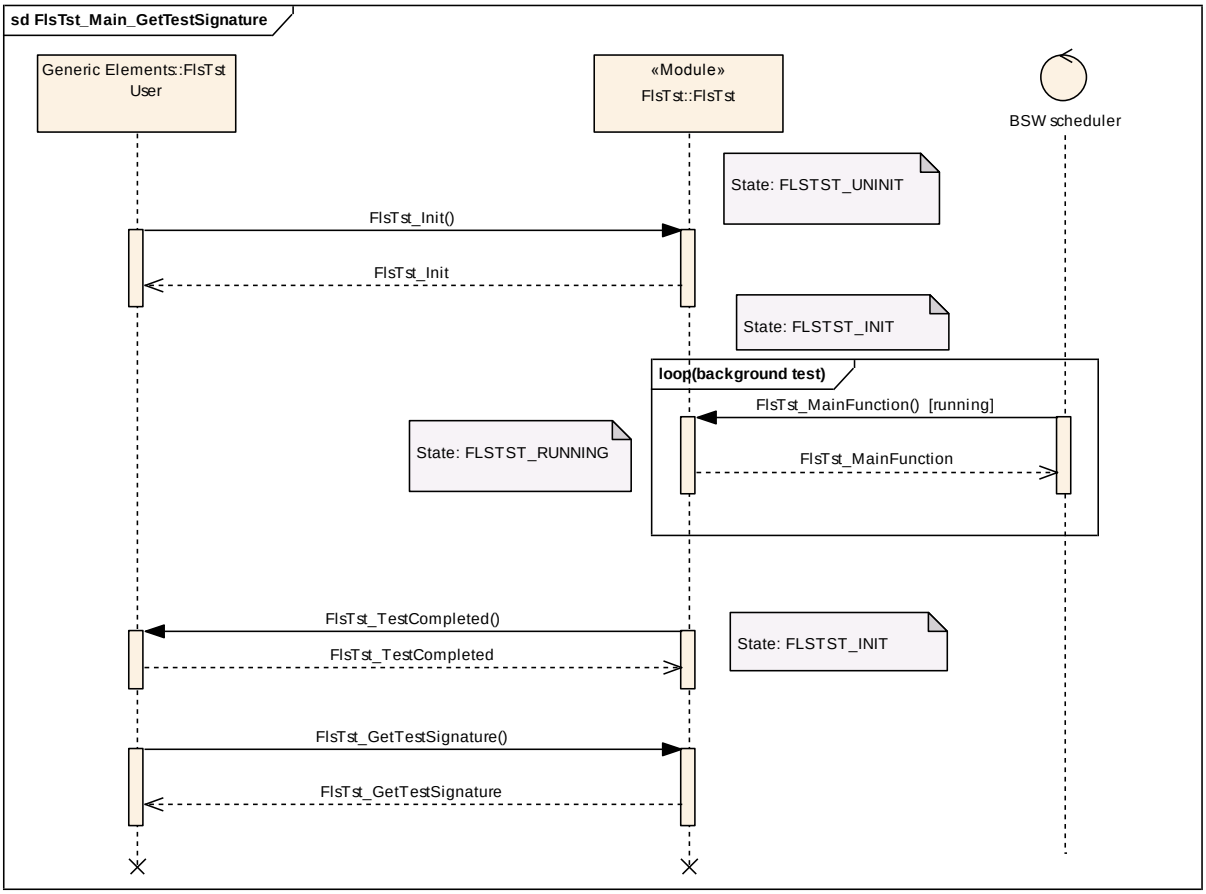

Figure 8: Background Test – Test signature provided to caller

9.4 Suspend and Resume Background Testing

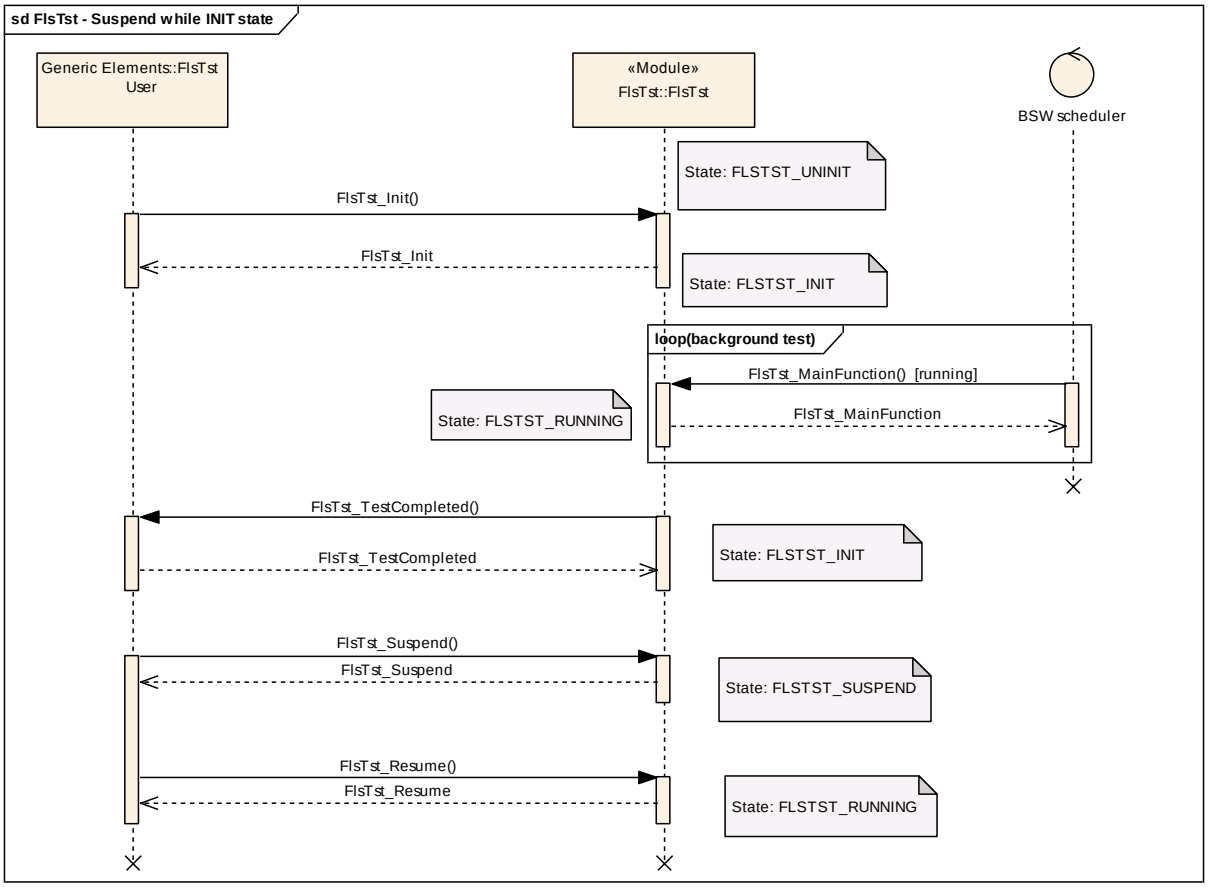

Figure 9: Suspend and Resume Background Testing

9.5 Foreground Task interrupts Background Task

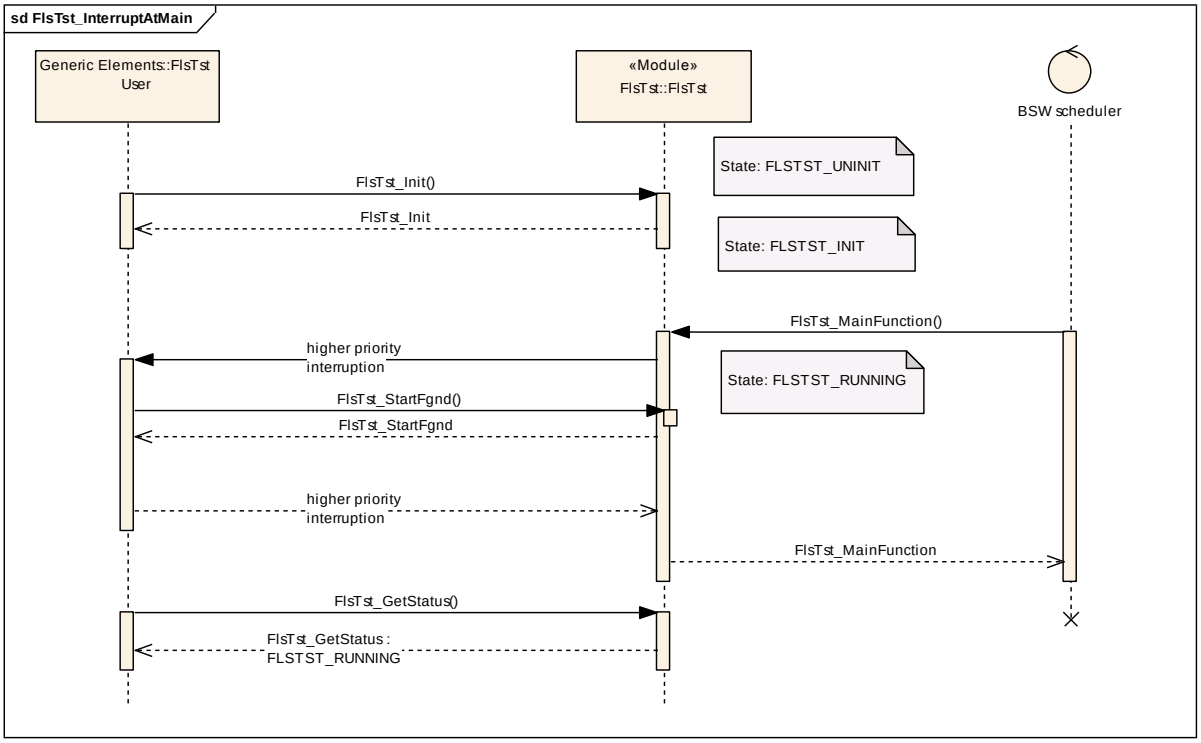

Figure 10: Foreground task interrupts Background Task

10 Configuration specification

10.1 Specification template for configuration parameters

Label	Description
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	Description
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	Description
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters Functional specification and Chapter API specification.

10.2.1 Variants

[SWS_FlsTst_00081] ⌈Variant PB: This variant allows a mix of pre-compile time-, post build-time configuration parameters. The intention of this variant is to optimize the parameters configuration for a re-loadable binary⌋ ( )

10.2.2 FlsTst

SWS Item	ECUC_FlsTst_00135 :
Module Name	*FlsTst*

Module Description	--

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FlsTstConfigSet	1	Multiple Configuration Set Container
FlsTstConfigurationOfOptApiServices	1	--
FlsTstDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
FlsTstGeneral	1	--

10.2.3 FlsTstGeneral

SWS Item	**ECUC_FlsTst_00082 :**
Container Name	FlsTstGeneral
Description	--
Configuration Parameters	

SWS Item	**ECUC_FlsTst_00083 :**		
Name	FlsTstDevErrorDetect {FLSTST_DEV_ERROR_DETECT}		
Description	Switch for enabling the development error detection.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00120 :**		
Name	FlsTstNumberOfTestedCellsAtomic {FLSTST_NUMBER_OF_TESTED_CELLS_ATOMIC}		
Description	Configures the Number of cells to be tested in background mode without checking user requests (Abort, Suspend).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00084 :**
Name	FlsTstTestCompletedNotificationSupported {FLSTST_TEST_COMPLETED_NOTIFICATION_SUPPORTED}
Description	Switch to indicate that the notification is supported.
Multiplicity	1

Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00158 :**		
Name	FlsTstTestIntervalIdEndValue {FLSTST_TEST_INTERVAL_ID_END_VALUE}		
Description	Defines the end value of the Test Interval Id.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00160 :**		
Name	FlsTstTestResultSignature {FLSTST_TEST_RESULT_SIGNATURE}		
Description	Configures the result of the test in background mode: True: Test Result is a signature (see FlsTst155, FlsTst054) False: Test Result is ok/not ok (see FlsTst153, FlsTst042)		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.4 FlsTstDemEventParameterRefs

SWS Item	**ECUC_FlsTst_00170 :**
Container Name	FlsTstDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**ECUC_FlsTst_00171 :**
Name	FLSTST_E_FLSTST_FAILURE
Description	Reference to the DemEventParameter which shall be issued when the error "Flash Failure" has occurred.
Multiplicity	0..1

Type	Symbolic name reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.5 FlsTstConfigSet

SWS Item	**ECUC_FlsTst_00152 :**
Container Name	FlsTstConfigSet [Multi Config Container]
Description	Multiple Configuration Set Container
Configuration Parameters	

SWS Item	**ECUC_FlsTst_00122 :**		
Name	FlsTstBlockNumberBgnd {FLSTST_BLOCK_NUMBER_BGND}		
Description	This parameter shall represent the number of test blocks available for the background test. calculationFormula = Number of configured FlsTstBlocks in the FlsTstBlockBgndConfigSet (or 0 if no FlsTstBlocks are configured).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00124 :**		
Name	FlsTstBlockNumberFgnd {FLSTST_BLOCK_NUMBER_FGND}		
Description	This parameter shall represent the number of test blocks available for the foreground test. calculationFormula = Number of configured FlsTstBlocks in the FlsTstBlockFgndConfigSet (or 0 if no FlsTstBlocks are configured).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00102 :**
Name	FlsTstTestCompletedNotification {FLSTST_TEST_COMPLETED_NOTIFICATION}
Description	Pointer to function, which shall be called after finishing the background Flash test interval.
Multiplicity	1

Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FlsTstBlockBgndConfigSet	0..1	This container defines the blocks in background mode.
FlsTstBlockFgndConfigSet	0..1	This container defines the blocks in foreground mode.

10.2.6 FlsTstBlockBgndConfigSet

SWS Item	**ECUC_FlsTst_00103 :**
Container Name	FlsTstBlockBgndConfigSet
Description	This container defines the blocks in background mode.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FlsTstBlock	1..*	This container specifies configuration parameters for an individual test block.

10.2.7 FlsTstBlockFgndConfigSet

SWS Item	**ECUC_FlsTst_00104 :**
Container Name	FlsTstBlockFgndConfigSet
Description	This container defines the blocks in foreground mode.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FlsTstBlock	1..*	This container specifies configuration parameters for an individual test block.

10.2.8 FlsTstBlock

SWS Item	**ECUC_FlsTst_00105 :**
Container Name	FlsTstBlock
Description	This container specifies configuration parameters for an individual test block.
Configuration Parameters	

SWS Item	**ECUC_FlsTst_00106 :**		
Name	FlsTstBlockBaseAddress {FLSTST_BLOCK_BASE_ADDRESS}		
Description	Start Address of the Flash block.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00151 :**		
Name	FlsTstBlockIndex {FLSTST_BLOCK_INDEX}		
Description	Foreground Test: Index identifies block to be tested by FlsTst_StartFgnd(); Background Test: The scheduling for background test shall follow an order defined by this index. '0' means highest priority.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00107 :**		
Name	FlsTstBlockSize {FLSTST_BLOCK_SIZE}		
Description	This parameter shall represent the Flash Test block size.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00119 :**		
Name	FlsTstNumberOfTestedCells {FLSTST_NUMBER_OF_TESTED_CELLS}		
Description	Configures the Number of cells to be tested in background mode during one scheduled task (FlsTst_MainFunction() call).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00123 :**

Name	FlsTstSignatureAddress {FLSTST_SIGNATURE_ADDRESS}		
Description	Address of the signature reference value of the Flash test block.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FlsTst_00101 :**		
Name	FlsTstTestAlgorithm {FLSTST_TEST_ALGORITHM}		
Description	This is the configuration of the test algorithm for foreground mode and background mode. The availability of algorithm is implementation specific.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FLSTST_16BIT_CRC	--	
	FLSTST_32BIT_CRC	--	
	FLSTST_8BIT_CRC	--	
	FLSTST_CHECKSUM	--	
	FLSTST_DUPLICATED_MEMORY	--	
	FLSTST_ECC	--	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

Note: The content of this chapter is specified in SWS_BSWGeneral document [9].

11 Not applicable requirements

[SWS_FlsTst_00166] ⌈These requirements are not applicable to this specification.⌋

(SRS_BSW_00344, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00170, SRS_BSW_00419, SRS_BSW_00398, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, BSW00431, SRS_BSW_00432, SRS_BSW_00433, BSW00434, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00007, SRS_BSW_00300, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00305, SRS_BSW_00307, SRS_BSW_00310, SRS_BSW_00373, SRS_BSW_00327, SRS_BSW_00335, SRS_BSW_00350, SRS_BSW_00408, SRS_BSW_00410, SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00361, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00378, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00371, SRS_BSW_00358, SRS_BSW_00414, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00009, SRS_BSW_00401, SRS_BSW_00172, SRS_BSW_00010, SRS_BSW_00003, SRS_BSW_00341, SRS_BSW_00334, SRS_BSW_00437, SRS_BSW_00439, SRS_BSW_00440, SRS_SPAL_12267, SRS_SPAL_12461, SRS_SPAL_12462, SRS_SPAL_12463, SRS_SPAL_12068, SRS_SPAL_12069, SRS_SPAL_12169, SRS_SPAL_12075, SRS_SPAL_12129, SRS_SPAL_12064, SRS_SPAL_12067, SRS_SPAL_12077, SRS_SPAL_12078, SRS_SPAL_12092, SRS_SPAL_12265, SRS_FlsTst_14221)