MCAL User Manual for Smu

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

Note: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.

Intended audience

This document is intended for anyone using the Smu module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)

`Courier`
 	
 `New`
	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets

>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual General

Table of contents

1 Smu driver

1.1 User information

1.1.1 Description

The SMU driver is an abstraction of the SMU peripheral in the AURIX™ microcontroller family. The SMU peripheral centralizes all the alarm signals related to different hardware safety mechanisms. Each alarm can trigger internal actions and/or notify the presence of faults to the external world through a fault signaling protocol. The SMU driver is active before any of the peripherals are active. Therefore, the SMU initialization and de-initialization must be called only on the master core. However, as it encompasses the hardware safety mechanisms distributed across all the cores, the SMU runtime services will be accessible from all cores.

1.1.2 Hardware-software mapping

This section describes the system view of the SMU driver and peripherals administered by it.

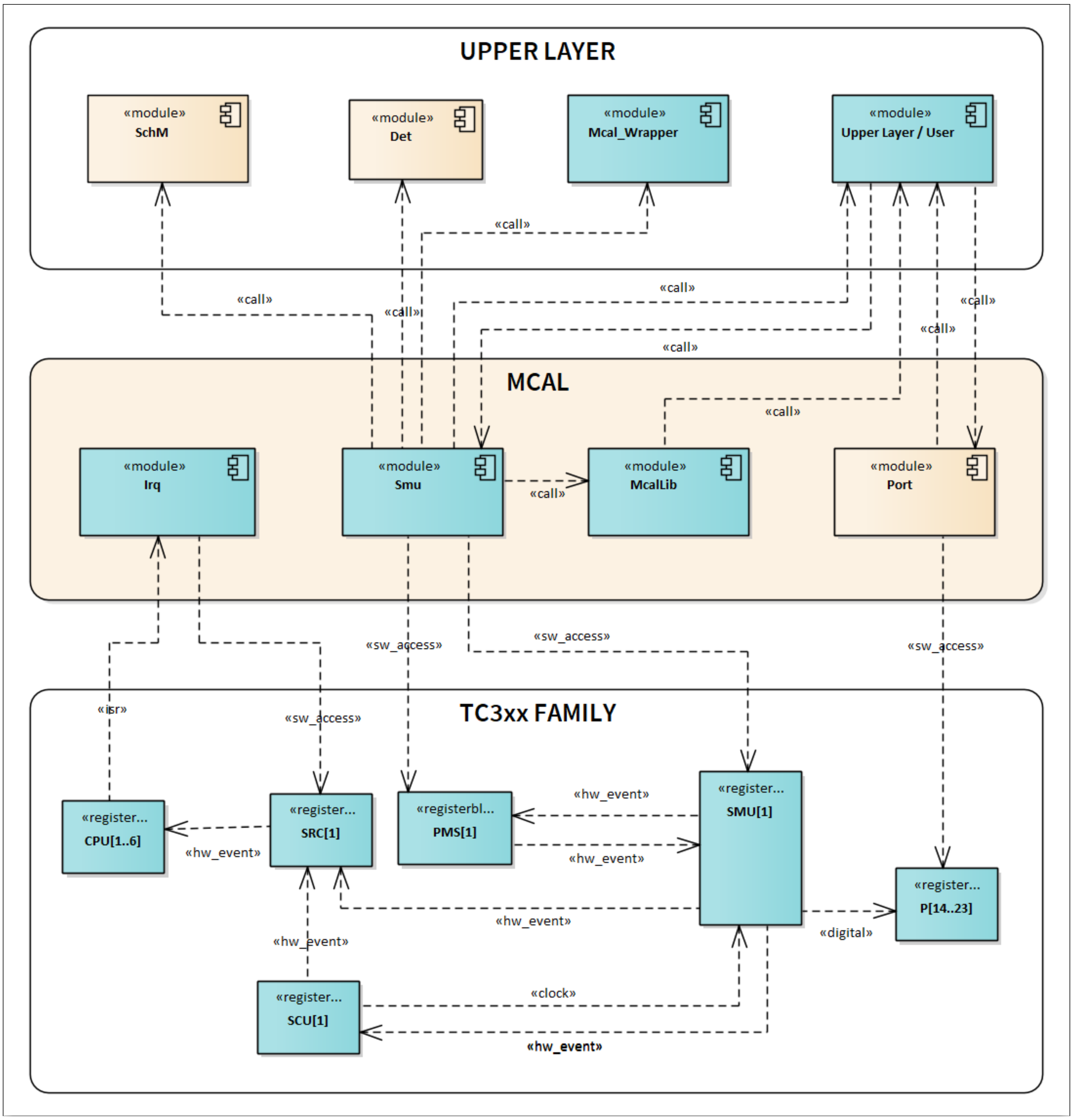


Figure 1 Mapping of hardware-software interfaces

1.1.2.1 SRC: dependent hardware peripheral

Hardware functional features

The SMU driver depends on the interrupt router for raising an interrupt to the CPU based on the internal reactions configured for the alarm events. These interrupts are SMU IR request 0, SMU IR request 1 and SMU IR request 2.

Users of the hardware

The service request nodes SRC_SMUx (x=0 to 2) are exclusively allocated to the SMU peripheral. The service request to be triggered is decided by the SMU peripheral using the interrupt generation set selected by the alarm that is IGSC0, IGSC1 or IGSC2. Each set is a code, which is a combination of the three service requests that need to be triggered; for example, IGSC0 = 011b implies that SRC_SMU0 and SRC_SMU1 service requests are triggered on alarm event.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events raised by the interrupt router are not used by the SMU driver.

1.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The SMU driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB clock signal for functioning. The driver also depends on the SCU IP for external EMS alarm and watchdog timeout.

The internal alarm reactions resulting from the alarm events are interfaced to the SCU. These interface signals can be of the following types:

- NMI request
- Reset request
- CPU reset request
- Emergency stop request
- Run state request

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

Associated SMU alarms, which can detect any hardware safety mechanism failures, exist.

Hardware events

Hardware events from the SCU are not used by the SMU driver.

1.1.2.3 PORT: dependent hardware peripheral

Hardware functional features

The FSP status is routed to the SMU through the port pads. The port pins are driven to the GPIO or SMU mode where the SMU uses the port pins to trigger external reaction mechanism using the FSP.

The SMU driver can also activate the emergency stop feature.

Users of the hardware

The port pads (P33.8 and P33.10) are configured and enabled by the user software through the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from the port pads are not used by the SMU driver.

1.1.2.4 SMU: primary hardware peripheral

Hardware functional features

The SMU driver uses the SMU for providing a generic interface to manage the behavior of the microcontroller under the presence of faults. The SMU centralizes all the alarm signals related to different hardware and software-based safety mechanisms. Each alarm can be individually configured to trigger internal actions and/or to notify externally the presence of faults through a fault signaling protocol.

The SMU driver provides an abstracted interface to the user to access the SMU peripheral.

The SMU peripheral has two parts: the SMU_core and the SMU_stdby. Both operate in different clock and power domains. One of the most important features of SMU_stdby is to monitor the correct functioning of SMU_core. The status of any error is triggered as an alarm from SMU_core to SMU_stdby.

The key hardware functional features used by the driver are:

- Alarm handling: Setting, querying, clearing of alarms and setup of alarm reactions
- FSP handling: Activation, deactivation of the FSP to request the trigger of safe state
- Port emergency handling: Activation of port emergency stop
- Recovery timer handling: Setup, status query of recovery timers
- SMU state machine: Status query of the SMU state machine and invoking transition between states
- SMU Configuration Protection: Activation of permanent lock mechanism
- Smu_ActivatePES() software command
- Register monitoring feature for safety flip flop

The unsupported features of the SMU are:

- OCDS Trigger Bus (OTGB) Interface

Modes or states of the SMU:

The SMU peripheral operates in three states: START, RUN and FAULT states. The application provides the services to invoke the transitions between the states.

The FSP driven by SMU has three states: Power-on, Fault and Fault Free states.

Users of the hardware

The SMU driver exclusively utilizes the SMU IP.

Hardware diagnostic features

The STS register can be read back to ensure the command has been successfully executed after writing the command to the CMD register.

Hardware events

The error status of the hardware and software safety mechanisms is indicated by the alarms. The reaction for these alarms is determined by the configuration parameters.

The SMU is connected to the interrupt router through three service request nodes SRC_SMUx (x = 0 to 2). Each service request can trigger an interrupt on the CPU 0, 1, 2, 3, 4 or 5. The number of CPUs depends on the derivative.

1.1.2.5 PMS: primary hardware peripheral

Hardware functional features

The SMU driver uses the PMS for SMU_stdby mode operation. The SMU_stdby operates in fBCK frequency provided by the PMS. The SMU_stdby is configurable, that is, it can be put to an idle state. All the alarms in the fBCK domain are forwarded to both SMU_stdby and SMU_core.

The key hardware functional features used by the driver are:

- Alarm status with respect to group 20 and 21 can be queried and cleared
- Monitoring of SMU_core through a runtime service for any alarms

The unsupported features of the PMS are:

- SMU_stby built-in self-test

Users of the hardware

The PMS is used by the SMU and MCU drivers. The SMU driver exclusively utilizes the SMU_stdby related registers of the PMS. The MCU driver does not utilize the SMU_stdby related registers and hence resource conflict does not happen.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from the PMS are not used by the SMU driver.

1.1.3 File structure

1.1.3.1 C file structure

The section provides details of the C files of the SMU driver.

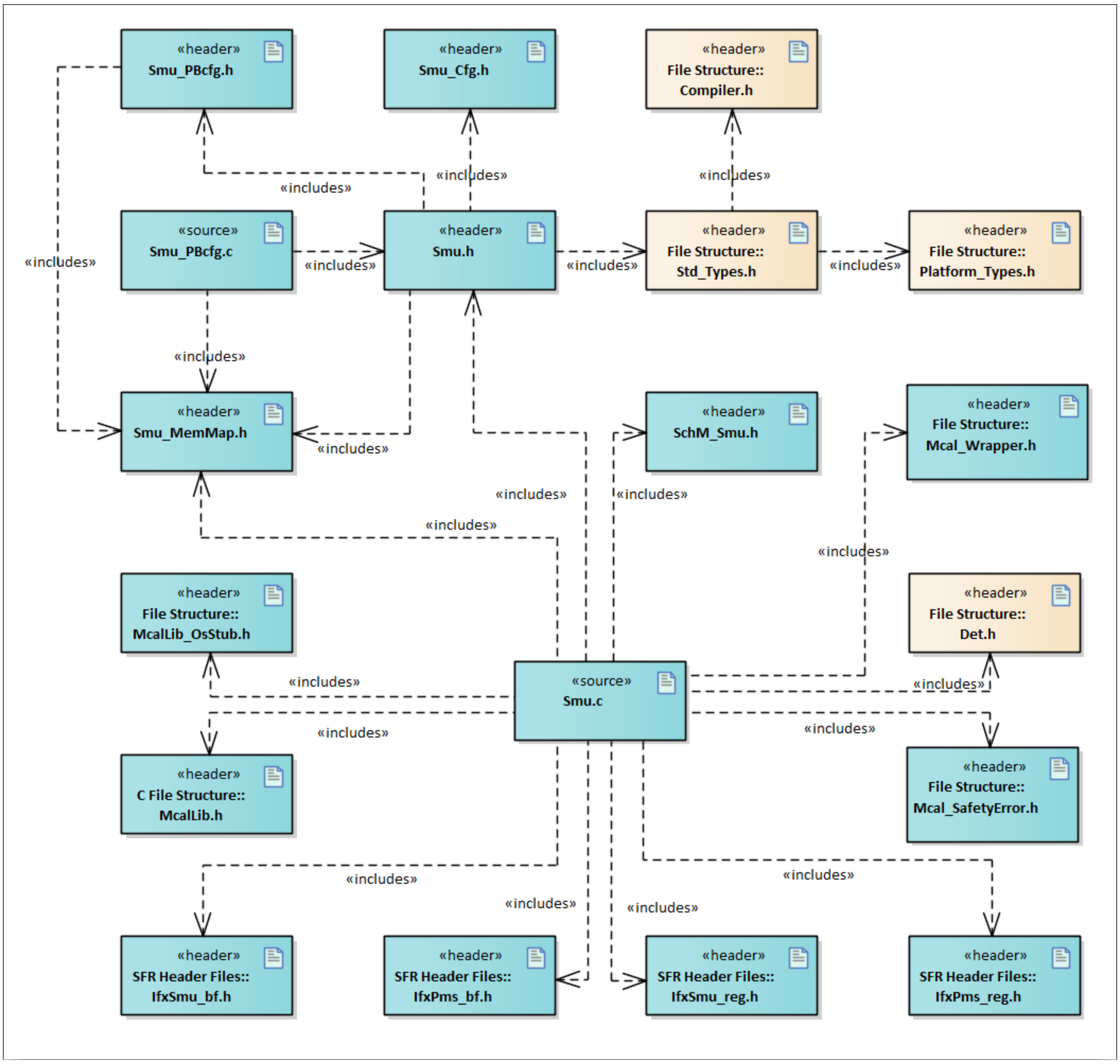

«header» Smu_PBcfg.h
«header» Smu_Cfg.h
«header» File Structure:: Compiler.h
«includes»
«source» Smu_PBcfg.c
«header» Smu.h
«header» File Structure:: Std_Types.h
«header» File Structure:: Platform_Types.h
«header» Smu_MemMap.h
«header» SchM_Smu.h
«header» File Structure:: Mcal_Wrapper.h
«header» File Structure:: McalLib_OsStub.h
«header» File Structure:: Det.h
«source» Smu.c
«header» C File Structure:: McalLib.h
«header» File Structure:: Mcal_SafetyError.h
«header» SFR Header Files:: IfxSmu_bf.h
«header» SFR Header Files:: IfxPms_bf.h
«header» SFR Header Files:: IfxSmu_reg.h
«header» SFR Header Files:: IfxPms_reg.h

Figure 2 **Smu_C_File_Structure-1.png**

Table 2 **C file structure**

File name	Description
`Compiler.h`	Provides abstraction from compiler-specific keywords
`Det.h`	Provides the exported interfaces of Development Error Tracer
`IfxPms_bf.h`	SFR header file for Pms
`IfxPms_reg.h`	SFR header file for Pms
`IfxSmu_bf.h`	SFR header file for SMU
`IfxSmu_reg.h`	SFR header file for SMU

(table continues...)

Table 2 (continued) C file structure

File name	Description
`McalLib.h`	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
`McalLib_OsStub.h`	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
`Mcal_SafetyError.h`	Header file containing the prototype of the API for reporting safety-related errors
`Mcal_Wrapper.h`	Provides the exported interfaces for Production Error and Runtime Development Errors. Implemented by default to include functions of Dem.h and Det.h files. This file can be modified by the user but function prototype is not user modifiable.
`Platform_Types.h`	Platform-specific type declaration file as defined by AUTOSAR
`SchM_Smu.h`	Header file contains the definitions of the SMU critical sections
`Smu.c`	File (static) contains the source code of the SMU software module
`Smu.h`	Header file (static) contains the data types and function prototypes to be exported
`Smu_Cfg.h`	Header file (generated) contains the pre-compile configuration for the SMU driver. The file implements all pre-processor directives.
`Smu_MemMap.h`	Header file contains the mapping of code and data (variables, constants) to specific memory sections for the SMU driver
`Smu_PBcfg.c`	File (generated) contains the post-build configuration for the SMU driver
`Smu_PBcfg.h`	Header file (generated) contains generated configuration data of user
`Std_Types.h`	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

1.1.3.2 Code generator plugin files

The section provides details of the code generator plugin files of the SMU driver.

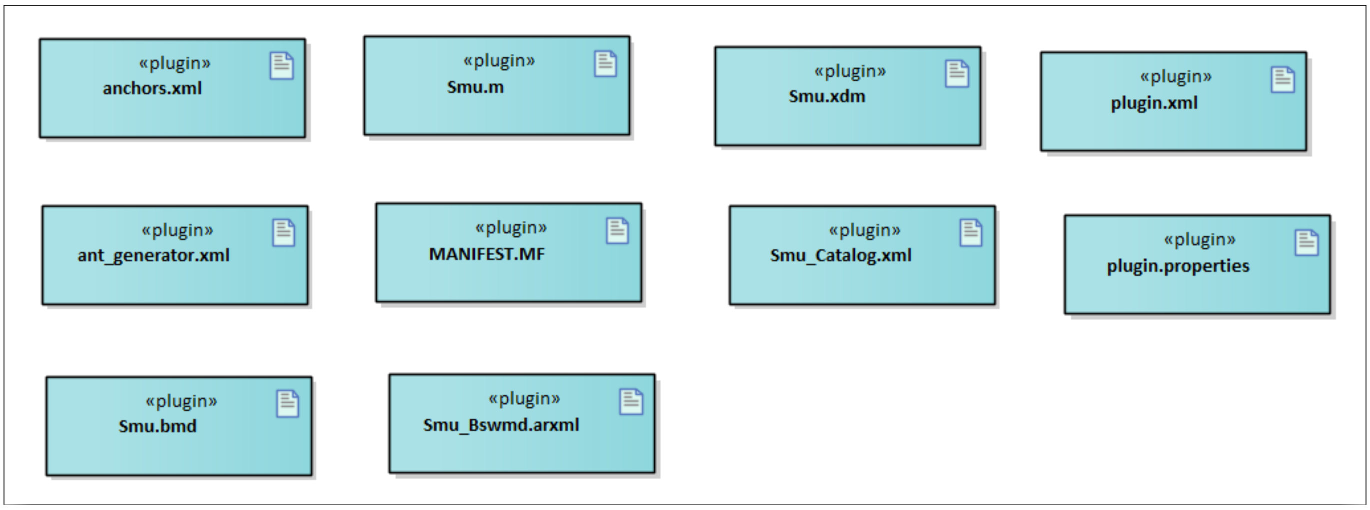


Figure 3 Smu_Code_Generator_Plugin_Files-1.png

Table 3 **Code generator plugin files**

File name	Description
`MANIFEST.MF`	Tresos plugin support file containing the metadata for the SMU driver
`Smu.bmd`	AUTOSAR format XML data model schema file for the SMU driver
`Smu.m`	Code template macro file for the SMU driver
`Smu.xdm`	Tresos format XML data model schema file for the SMU driver
`Smu_Bswmd.arxml`	AUTOSAR format module description file for the SMU driver
`Smu_Catalog.xml`	AUTOSAR format catalog file for the SMU driver
`anchors.xml`	Tresos anchors support file for the SMU driver
`ant_generator.xml`	Tresos support file to generate and rename multiple post-build configurations when using variation point
`plugin.properties`	Tresos plugin support file for the SMU driver
`plugin.xml`	Tresos plugin support file for the SMU driver

1.1.4 Integration hints

The section lists the key points that an integrator or user of the SMU driver must consider.

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of MCAL, but are required to integrate the SMU driver.

- **EcuM**

The EcuM module is not required for integrating SMU driver.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Smu_MemMap.h` file.

The `Smu_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are relocated to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

```
/* User Pragma to be placed here for LMU RAM NC*/
#undef SMU_START_SEC_INIT_VAR_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR
#elif defined SMU_STOP_SEC_INIT_VAR_ASIL_B_GLOBAL_32
/* User Pragma for LMU RAM NC here */
#undef SMU_STOP_SEC_INIT_VAR_ASIL_B_GLOBAL_32
#undef MEMMAP_ERROR

/*Configuration data sections-- to be placed in PFx*/
#elif defined SMU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas to be placed here for PF0 */
#undef SMU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined SMU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
/* User pragmas to be placed here for PFx */
#undef SMU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

/* Code Section ----- to be placed in PFx*/
#elif defined SMU_START_SEC_CODE_ASIL_B_GLOBAL
/* User Pragma to be placed here */
#undef SMU_START_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR
#elif defined SMU_STOP_SEC_CODE_ASIL_B_GLOBAL
/* User Pragma to be placed here */
#undef SMU_STOP_SEC_CODE_ASIL_B_GLOBAL
#undef MEMMAP_ERROR

#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The SMU driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the SMU driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and need to be replaced with a complete DET module during the integration phase.

- **Mcal_Wrapper**

This Driver performs reporting of the Production and Runtime errors. The Handling of the reported errors shall be done by the user. The `Mcal_Wrapper_Det_ReportRuntimeError()` API, `Mcal_Wrapper_Dem_SetEventStatus()` API and `Mcal_Wrapper_Dem_ReportErrorStatus()` API are provided in the `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files as a stub code, and can be updated by the integrator to handle the reported errors. The files `Mcal_Wrapper.c` and `Mcal_Wrapper.h` are user modifiable, where the function prototypes are not user modifable and by default the Mcal Wrapper function shall calls AUTOSAR DEM and DET Modules.

The user of the Smu driver shall process all the production errors reported to the Mcal_Wrapper module. The interface used for reporting production error In AUTOSAR version 4.2.2 is `Mcal_Wrapper_Dem_ReportErrorStatus()` API and for AUTOSAR version 4.4.0 is

`Mcal_Wrapper_Dem_SetEventStatus()` API. The `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files are provided in the MCAL package as a stub code and can be replaced with a user specific production error handling module during the integration phase. Smu driver shall not report runtime errors.

Note: Reentrancy of the `Smu_ClearAlarmStatus()`, `Smu_SetAlarmStatus()`, `Smu_ReleaseFSP()`, `Smu_ActivateFSP()`, `Smu_RTStop()`, `Smu_ActivateRunState()`, `Smu_ActivatePES()`, `Smu_CoreAliveTest()` and `Smu_RegisterMonitor()` APIs is dependent on the reentrancy of the `Mcal_Wrapper_Dem_ReportErrorStatus()` and `Mcal_Wrapper_Dem_SetEventStatus()` APIs. As per the design, the APIs of the module are reentrant. However, in case the `Mcal_Wrapper_Dem_ReportErrorStatus()` or `Mcal_Wrapper_Dem_SetEventStatus()` API is implemented as non-reentrant, the APIs inherit the property of the same.

- **SchM**

The SchM module is a part of the RTE that manages the BSW Scheduler. The SMU driver uses the exclusive areas defined in the `SchM_Smu.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the SMU driver are:

- CmdAccess

- DriverAccess

The `SchM_Smu.h` and `SchM_Smu.c` files are provided in the MCAL package as an example code and need to be updated by the integrator. The user must implement the SchM functions defined by the SMU driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

```
/*Sample implementation*/
#include "IFX_Os.h"
#include "SchM_Smu.h"

void SchM_Enter_CmdAccess(void)
{
/*Suspend all interrupts*/
 SuspendAllInterrupts();
}
void SchM_Exit_CmdAccess(void)
{
/*Resume all interrupts*/
 ResumeAllInterrupts();
}

void SchM_Enter_DriverAccess(void)
{
/*Suspend all interrupts*/
SuspendAllInterrupts();
}

void SchM_Exit_DriverAccess(void)
{
/*Resume all interrupts*/
ResumeAllInterrupts();
}
```

- **Safety error**

The SMU driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notification and callbacks**

The SMU driver does not provide any callbacks or notifications.

- **OS**

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

The SMU driver does not require configuration of any interrupts. The interrupts triggered because of alarm action have to be handled by the user.

1.1.4.2 Multicore and Resource Manager

The SMU driver supports execution of its APIs in parallel from all CPU cores. The following are the key points to be considered with respect to multicore in the SMU driver:

- The runtime services of the SMU driver will be accessible by all cores.
- The hardware and software safety mechanism are associated with specific alarm groups and positions and cannot be reallocated by software or configuration. Hence, the SMU driver does not have any core-specific resource allocation.
- The SMU initialization and de-initialization must be called only from the master core. In case the `Smu_Init()` or `Smu_DeInit()` API is called from any core other than the master core then the API will report an error `E_NOT_OK`. In case DET is enabled, a DET error `SMU_E_CORE_MISMATCH` will also be reported.
- The `Smu_ActivateRunState` and `Smu_ReleaseFSP` APIs shall be invoked from only one core at a time. Invoking from multiple cores simultaneously may lead to significantly high API execution time due to CPU resource starvation.
- Locating of constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores). The following should be considered by the user to ensure better performance of the SMU driver:

  Code section:

  The executable code of the SMU driver is placed under single MemMap section. It can be relocated to any PFlash region.

  Data section:

  The sections marked as GLOBAL should be relocated to the non-cached LMU region.

  Configuration data and constants:

  The sections marked as GLOBAL should be relocated to the PFlash of the master core.

  Note: Relocating of code, data or constants to a distant memory region would impact execution timings.

  Note: If the driver operates from single (master) core, all the sections may be relocated to the PFlash/DSPR/DLMU of the same CPU core.

1.1.4.3 MCU support

The SMU driver does not use any services provided by the MCU driver.

1.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure the port pins 33.8 and 33.10 used by the SMU driver for FSP through the PORT configuration.

1.1.4.5 DMA support

The SMU driver does not use any services provided by the DMA driver.

1.1.4.6 Interrupt connections

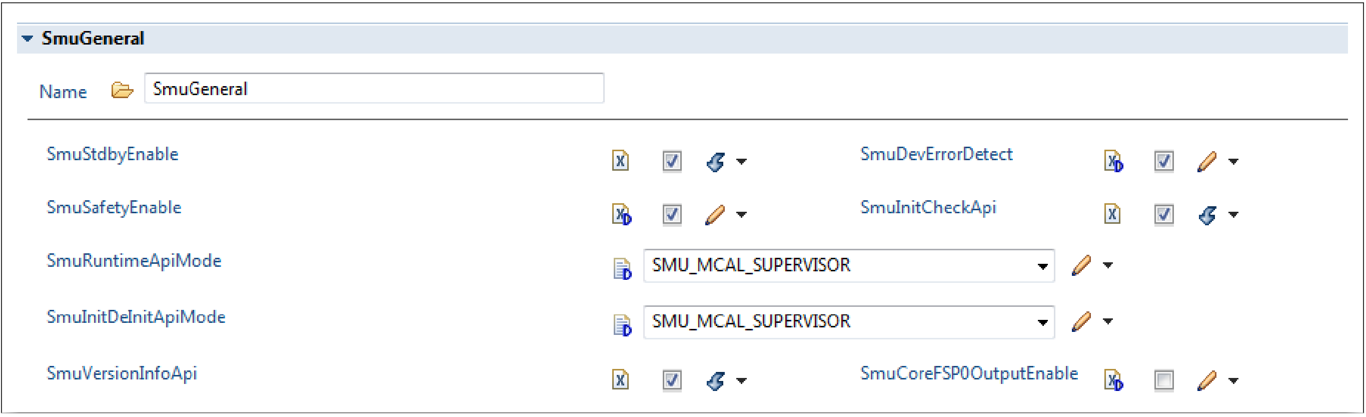
The SMU driver does not use any interrupt source.

1.1.4.7 Example usage

Initializing and de-initializing the SMU driver

The SMU driver is initialized by calling the `Smu_Init()` API. The user must call the `Smu_Init()` API from the master core only. When the `Smu_Init()` API is called from a core other than master core, the API returns an error. The same criteria apply to the `Smu_DeInit()` API. Also in order to check the initialization values, the `Smu_InitCheck()` API can be used. However the `Smu_InitCheck()`API is enabled only when the `SmuInitCheckApi` parameter is enabled. The SMU driver can be initialized and de-initialized as shown follows:

```
#include "Smu_Test.h"
/* Initialize SMU*/
Return = Smu_Init(&(Smu_Config));
/*Check for initialiazation values*/
#if(SMU_INIT_CHECK_API==STD_ON)
Return = Smu_InitCheck(&(Smu_Config));
#endif
/*Call SMU driver functions*/
/*.........................*/
/*Deinitialize the driver*/
Return = Smu_DeInit();
```


SmuGeneral

Name: SmuGeneral

SmuStdbyEnable	✓	SmuDevErrorDetect	✓
SmuSafetyEnable	✓	SmuInitCheckApi	✓
SmuRuntimeApiMode	SMU_MCAL_SUPERVISOR		
SmuInitDeInitApiMode	SMU_MCAL_SUPERVISOR		
SmuVersionInfoApi	✓	SmuCoreFSP0OutputEnable	☐

Figure 4 **Example configuration for enabling Smu_stdby, InitCheck API, DET, version info API and user mode**

In case the `SmuSafetyEnable` parameter is enabled, the user needs to configure the DEM reporting and add the parameter to the `SmuDemEventParameterRefsConf` container.

Configuration register locking

The SMU configuration registers can be protected from unintended access in two ways:

- Temporary lock: The SMU configuration register protection is disabled temporarily to write into the SMU configuration registers and then again enabled.
- Permanent lock: The SMU configuration register protection is enabled to prevent any writing into the SMU configuration register and can be disabled only after application reset.

The `Smu_LockConfigRegs()` API enables the permanent lock on the configuration registers. In case the `Smu_LockConfigRegs()` API fails to turn on the permanent lock and safety error check is enabled, then a DET is reported to let the user know that the configuration registers could not be permanently locked.

User shall ensure that the `Smu_LockConfigRegs()` API is invoked once SMU configuration is completed and no further change in configuration is expected. In case the `Smu_SetAlarmAction()`, `Smu_SetupErrorPin()`, `Smu_ReleaseErrorPin()`, `Smu_RegisterMonitor()` or `Smu_DeInit()` API is called after invoking the `Smu_LockConfigRegs()` API, the API will raise a DET error indicating that the driver is in the LOCKED state. The command-based APIs will work as per their functionality.

An example usage is shown as follows:

```
Std_ReturnType Result = E_NOT_OK;
ResultLockTest = Smu_LockConfigRegs();
```

Alarm status

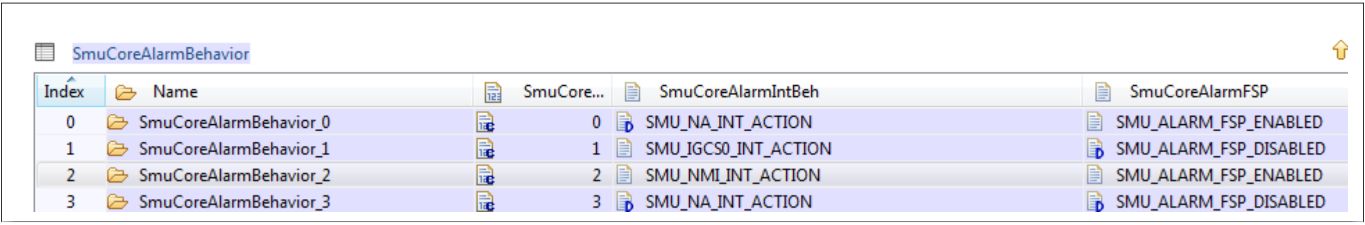
The SMU driver provides services to set, clear and get the alarm status by using the `Smu_SetAlarmStatus()`, `Smu_ClearAlarmStatus()` and `Smu_GetAlarmStatus()` APIs. An example usage is shown as follows:

```
Smu_SetAlarmStatus is valid only for Smu_core.
ResultAlarmStatus = Smu_ClearAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_0);
if (E_OK == ResultAlarmStatus)
{
 ResultAlarmStatus = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_0);
 if (E_OK == ResultAlarmStatus)
 {
 ResultAlarmStatus = Smu_GetAlarmStatus(SMU_ALARM_GROUP10, &AlarmStatus);
 if ((E_OK == ResultAlarmStatus) && (0x01U == (AlarmStatus & 0x01)))
 {
 ResultAlarmStatus = Smu_ClearAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_0);
 if (E_OK == ResultAlarmStatus)
 {
 ResultAlarmStatus = Smu_GetAlarmStatus(SMU_ALARM_GROUP10, &AlarmStatus);
 }
 }
 }
}
```

Alarm action

The SMU driver provides services to set and get the alarm actions by using the `Smu_SetAlarmAction()` and `Smu_GetAlarmAction()` APIs.

Two kinds of alarm actions can be configured for SMU_core: internal action and external action. These are configured as follows:



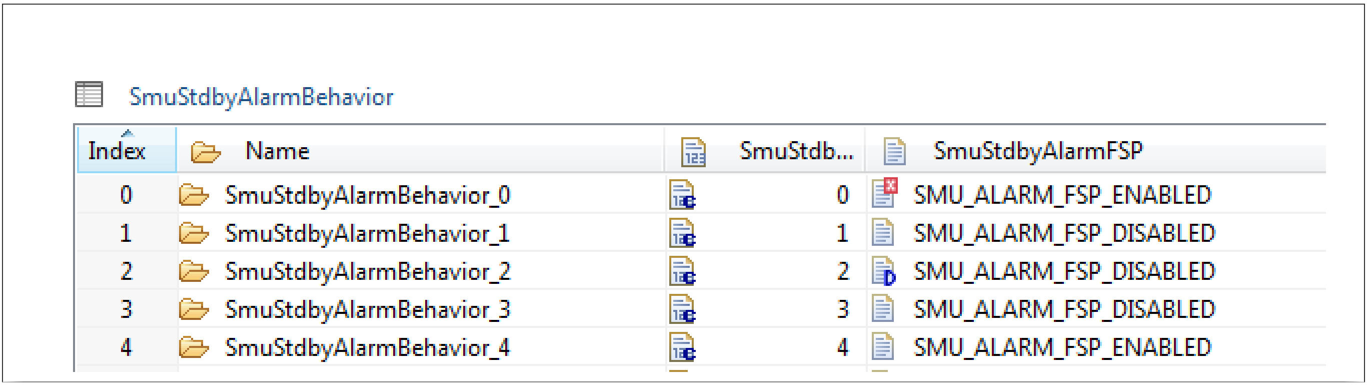
SmuCoreAlarmBehavior

Index	Name	SmuCore...	SmuCoreAlarmIntBeh	SmuCoreAlarmFSP
0	SmuCoreAlarmBehavior_0	0	SMU_NA_INT_ACTION	SMU_ALARM_FSP_ENABLED
1	SmuCoreAlarmBehavior_1	1	SMU_IGCS0_INT_ACTION	SMU_ALARM_FSP_DISABLED
2	SmuCoreAlarmBehavior_2	2	SMU_NMI_INT_ACTION	SMU_ALARM_FSP_ENABLED
3	SmuCoreAlarmBehavior_3	3	SMU_NA_INT_ACTION	SMU_ALARM_FSP_DISABLED

Figure 5 **Internal and external action settings for Smu_core**

For SMU_stdby, the internal reaction is by default `SMU_NA_INT_ACTION`. Only external action can be configured.

For both SMU_core and SMU_stdby, in case an alarm action is tried to be configured for a reserved alarm group, then the configuration will throw an error as shown in the following figure:



SmuStdbyAlarmBehavior

Index	Name	SmuStdb...	SmuStdbyAlarmFSP
0	SmuStdbyAlarmBehavior_0	0	SMU_ALARM_FSP_ENABLED
1	SmuStdbyAlarmBehavior_1	1	SMU_ALARM_FSP_DISABLED
2	SmuStdbyAlarmBehavior_2	2	SMU_ALARM_FSP_DISABLED
3	SmuStdbyAlarmBehavior_3	3	SMU_ALARM_FSP_DISABLED
4	SmuStdbyAlarmBehavior_4	4	SMU_ALARM_FSP_ENABLED

Figure 6 External action setting for Smu_stdby

Register monitoring

The `Smu_RegisterMonitor()` API can be used to enable the SFF tests for the protected registers of particular modules and retrieve the results. The input parameter passed to the API strictly has to follow the sequence of modules as per the bit fields of the RMCTL register. However, the user should ensure that the module for which the SFF test is being requested is present in the derivative being used.

Smu_core alive test

The `Smu_CoreAliveTest()` API provide the means to execute the SMU_AliveTest command that checks the smu_core_alive signal. For the smu_core_alive test to run through the SMU_core command sequence, the SMU_stdby shall be enabled and the SMU_core shall be in START state. In case the SMU_stdby is not enabled or the SMU_core is not in START state, the `Smu_CoreAliveTest()` API returns `E_NOT_OK`.

The user shall read the status flags for the SMU_core alive alarm (alarm 16 of alarm group 21) using the `Smu_GetAlarmStatus()` API to check the result of the smu_core_alive test after execution of the `Smu_CoreAliveTest()` API. The user shall clear the status of the SMU_core alive alarm (alarm 16 of alarm group 21) using the `Smu_ClearAlarmStatus()` API, after checking the result of the smu_core_alive test so that further alarm detection is possible.

The following API sequence should be followed execute the SMU_AliveTest command:

```
/*Execute smu_core_alive test */
ResultCoreAliveTest = Smu_CoreAliveTest();

/*Read status of alarm 16 of alarm group 21 to check the result of the smu_core_alive test*/
ResultGetAlarmStatus = Smu_GetAlarmStatus(SMU_ALARM_GROUP21, &AlarmStatus16);

/*Clear status of alarm 16 of alarm group 21 after checking the result of smu_core_alive test
so that further alarm detection is possible*/
ResultClearAlarmStatus = Smu_ClearAlarmStatus(SMU_ALARM_GROUP21, SMU_ALARM_16);
```

Alarm execution status

The `Smu_GetAlarmExecutionStatus()` API can be used to retrieve the alarm execution status. Once retrieved, the execution status can be cleared using the `Smu_ClearAlarmExecutionStatus()` API. The alarm reactions are not triggered if the alarm execution status for the particular alarm is not cleared.

In case, the parameter passed to the `Smu_ClearAlarmExecutionStatus()` API is a bit position corresponding to alarm event missed bit, only the alarm event missed bit shall be cleared.

In case, the parameter passed to the `Smu_ClearAlarmExecutionStatus()` API is a bit position corresponding to execution status bit, the execution status bit as well as the corresponding alarm event missed bit shall be cleared.
Hence, if the user is using the alarm missed event functionality, then the user shall first read the alarm event missed bit using the `Smu_GetAlarmExecutionStatus()` API before clearing the execution status bit using the `Smu_ClearAlarmExecutionStatus()` API.

An example code usage is shown as follows:

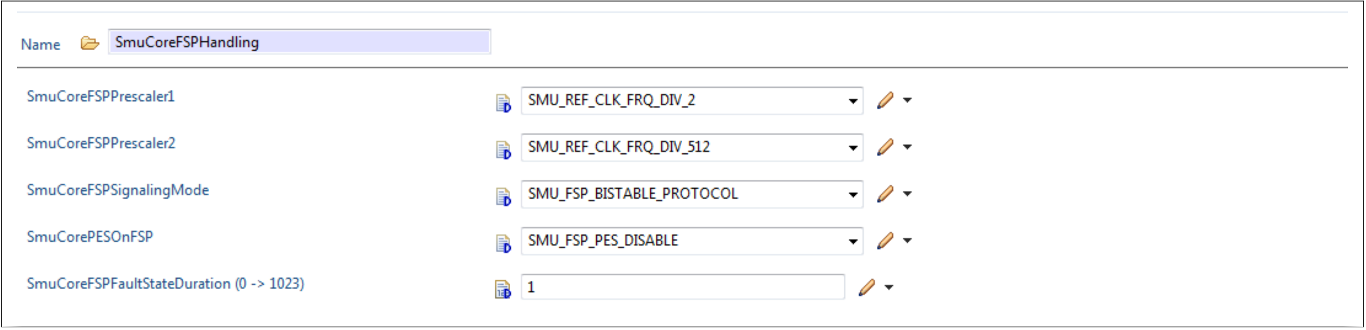
```
Std_ReturnType RetVal = E_NOT_OK;
/*Get the alarm execution status*/
RetVal = Smu_GetAlarmExecutionStatus(ExecReq, &ExecStatus);

if(RetVal == E_OK)
{
/*Clear the alarm execution status of the particular alarm as requested*/
RetVal = Smu_ClearAlarmExecutionStatus(ExecStatusReq);
}
```

FSP handling and Smu_core State Machine

For FSP handling, the glitch filter settings, output pin direction and enable has to set by the user (refer configuration for more details). FSP can be operated by choosing one of the three signaling modes:

- Time switching protocol
- Dual rail protocol
- Bi-stable protocol

FSP output pins can be enabled through configuration for Smu_stdby. PES can be enabled or disabled while using FSP. The prescalar, signaling mode, fault state duration can be selected as per the configuration depicted in the following figure:



Name	SmuCoreFSPHandling
SmuCoreFSPPrescaler1	SMU_REF_CLK_FRQ_DIV_2
SmuCoreFSPPrescaler2	SMU_REF_CLK_FRQ_DIV_512
SmuCoreFSPSignalingMode	SMU_FSP_BISTABLE_PROTOCOL
SmuCorePESOnFSP	SMU_FSP_PES_DISABLE
SmuCoreFSPFaultStateDuration (0 -> 1023)	1

Figure 7 FSP setting for Smu_core

To enable the transition of FAULT to RUN state, the `SmuCoreEnableFaultToRunState` parameter shall be enabled. In addition, the external action with respect to the alarm group and position should be enabled. The internal and external reaction configuration is explained in section **Alarm action**

An example code usage:

```
Smu_CoreStateType SmuState;
/*Setup the error pin in SMU mode*/
ResultFSP = Smu_SetupErrorPin();
if(E_OK == ResultFSP)
{
 /*Set the alarm status of alarm group 10 and position 5*/
 ResultFSP = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_5);
 {
 /*Activate FSP to indicate a fault state*/
 ResultFSP = Smu_ActivateFSP();
 if(E_OK == ResultFSP)
 {
 /*Get the SMU state*/
 SmuState = Smu_GetSmuState();
 if(SMU_FAULT_STATE == SmuState)
 {
 /*In case it is FAULT state then release FSP to transition to RUN state*/
 ResultFSP = Smu_ReleaseFSP();
 if(E_OK == ResultFSP)
 {
 /*Release error pin to transition to GPIO mode*/
 ResultFSP = Smu_ReleaseErrorPin();
 }
 }
 }
}
```

Smu_core has three states- START, RUN, FAULT.

Transition from START to RUN: The transition takes place by executing the SMU_core command to activate RUN state by calling the `Smu_ActivateRunState()` API.

Transition from FAULT to RUN: The transition takes place by executing the SMU_core command by calling the `Smu_ReleaseFSP()` API.

In order to indicate the FAULT state on the error pin, `Smu_ActivateFSP()` API is called.

An example code usage and sequence diagram for state machine and FSP handling is as follows:

```
/*Get the Smu_core state*/
SmuState = Smu_GetSmuState();

switch (SmuState)
{
 case SMU_START_STATE:
 {
 /*Activate the RUN state*/
 Result = Smu_ActivateRunState(SMU_RUN_COMMAND);
 break;
 }
 case SMU_FAULT_STATE:
 {
 /*In case it is FAULT state then release FSP to transition to RUN state*/
 Result = Smu_ReleaseFSP();
 break;
 }
 case SMU_RUN_STATE:
 {
 Result = E_OK;
 break;
 }
 default:
 break;
}
```

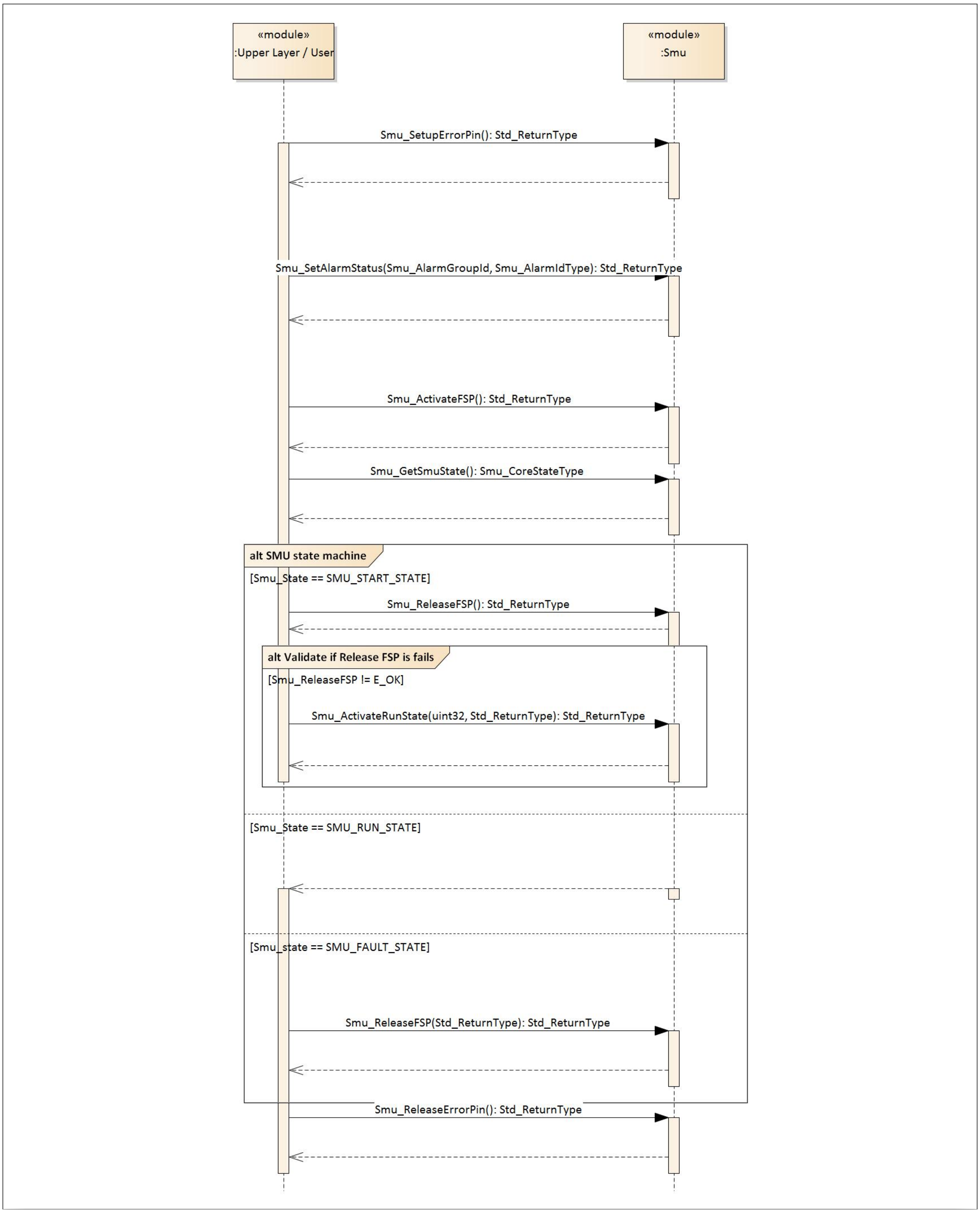
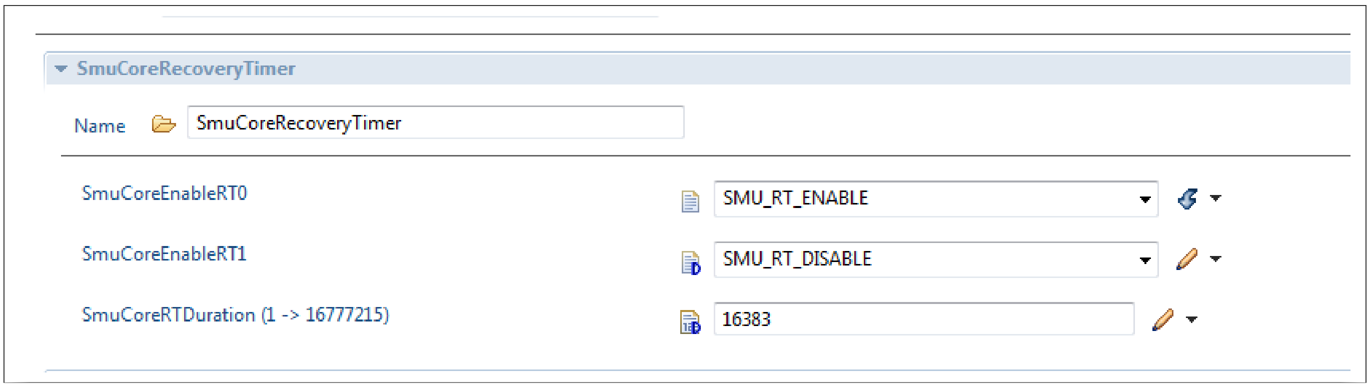

«module»
:Upper Layer / User
«module»
:Smu
Smu_SetupErrorPin(): Std_ReturnType
Smu_SetAlarmStatus(Smu_AlarmGroupId, Smu_AlarmIdType): Std_ReturnType
Smu_ActivateFSP(): Std_ReturnType
Smu_GetSmuState(): Smu_CoreStateType
alt SMU state machine
[Smu_State == SMU_START_STATE]
Smu_ReleaseFSP(): Std_ReturnType
alt Validate if Release FSP is fails
[Smu_ReleaseFSP != E_OK]
Smu_ActivateRunState(uint32, Std_ReturnType): Std_ReturnType
[Smu_State == SMU_RUN_STATE]
[Smu_state == SMU_FAULT_STATE]
Smu_ReleaseFSP(Std_ReturnType): Std_ReturnType
Smu_ReleaseErrorPin(): Std_ReturnType

Figure 8 **Sequence diagram depicting FSP handling and transition of the states of the Smu_core state machine**

Smu_core recovery timer

The recovery timers are configured by first enabling the RT0 and RT1 and setting the RT duration. On enabling, the respective RT group configurations are enabled.

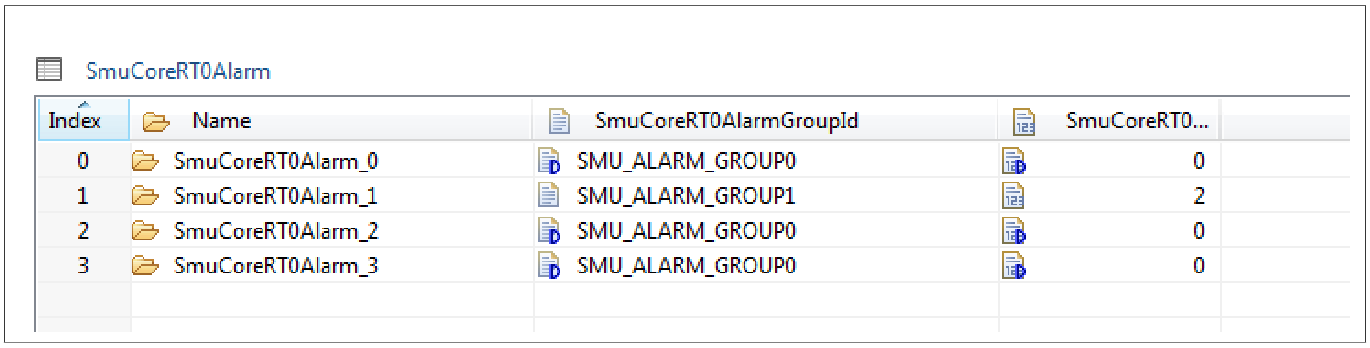


SmuCoreRecoveryTimer

Name	SmuCoreRecoveryTimer
SmuCoreEnableRT0	SMU_RT_ENABLE
SmuCoreEnableRT1	SMU_RT_DISABLE
SmuCoreRTDuration (1 -> 16777215)	16383

Figure 9 **Enable the RT configuration**

After the RT group configuration is enabled, the recovery timers can be assigned to the SMU_core alarm groups and positions.



SmuCoreRT0Alarm

Index	Name	SmuCoreRT0AlarmGroupId	SmuCoreRT0...
0	SmuCoreRT0Alarm_0	SMU_ALARM_GROUP0	0
1	SmuCoreRT0Alarm_1	SMU_ALARM_GROUP1	2
2	SmuCoreRT0Alarm_2	SMU_ALARM_GROUP0	0
3	SmuCoreRT0Alarm_3	SMU_ALARM_GROUP0	0

Figure 10 **Assigning alarm groups and positions to RT**

In order to configure RT, the respective alarm group and position have to be configured for their internal action. In case it is a reserved alarm position, the user must take care of not assigning the alarm position to RT0 or RT1. The information of reserved alarm positions will be evident when an error is encountered while configuring the internal alarm action for that particular alarm group and position. Therefore, the RT configuration can take reference from the error as discussed.

The SMU driver provides the services to stop the recovery timer and to detect any missed events for the configured RT. An example usage with sequence diagram is shown as follows:

```
volatile uint32 DelayCount = SMU_RT_DELAY;
/*Set the alarm status for alarm group 10 and position 2*/
ResultRecoveryTimer = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_2);
if (E_OK == ResultRecoveryTimer)
{
 /*Set the alarm status for alarm group 10 and position 3*/
 ResultRecoveryTimer = Smu_SetAlarmStatus(SMU_ALARM_GROUP10, SMU_ALARM_3);
 if (E_OK == ResultRecoveryTimer)
 {
 /*capture any missed events*/
 ResultRecoveryTimer = Smu_GetRTMissedEvent(SMU_TIMER_NUM, &EventMissed);
 if (0x01 == EventMissed)
 {
 /*Stop the RT is missed events are detected*/
 ResultRecoveryTimer = Smu_RTStop(SMU_TIMER_NUM);
 }
 }
}
```

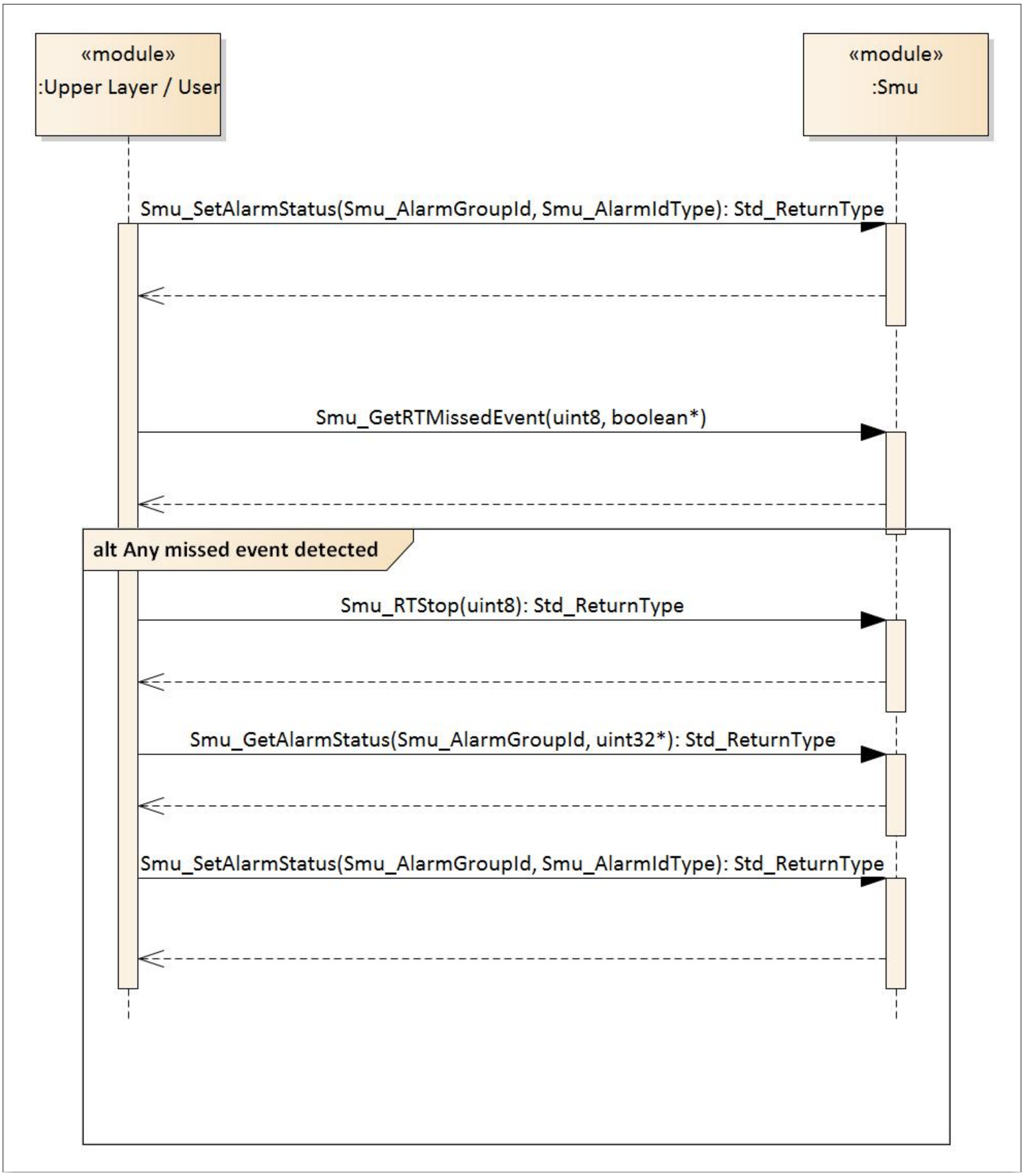

«module»
:Upper Layer / User
«module»
:Smu
Smu_SetAlarmStatus(Smu_AlarmGroupId, Smu_AlarmIdType): Std_ReturnType
Smu_GetRTMissedEvent(uint8, boolean*)
alt Any missed event detected
Smu_RTStop(uint8): Std_ReturnType
Smu_GetAlarmStatus(Smu_AlarmGroupId, uint32*): Std_ReturnType
Smu_SetAlarmStatus(Smu_AlarmGroupId, Smu_AlarmIdType): Std_ReturnType

Figure 11 **Sequence diagram for recovery timer usage**

1.1.5 Key architectural considerations

1.1.5.1 Clearing alarm status during initialization

During initialization, all the alarm statuses are cleared. Therefore, the user must ensure to keep a track of the alarm status before the `Smu_Init()` API is called.

1.1.5.2 Initialization and deinitialization

The `Smu_Init()` and `Smu_Deinit()` APIs shall be called only from the master cores. In case the `Smu_Init()` or `Smu_Deinit()` API is called from any other core besides the master core, the sequence will return an error. There is no resource distribution across the cores and the SMU driver shall be accessible across all cores except for the `Smu_ActivateRunState()` and `Smu_ReleaseFSP()` APIs.

1.1.5.3 SMU_core state machine transitions

The SMU_core state machine transitions are not to be verified by the driver. The user must verify the state before using it.

1.1.5.4 Recovery timer handling

While recovery timer is running, a missed alarm mapped to the same recovery timer is logged in the SFR SMU_STS. These missed alarms have to be explicitly checked and cleared by the application. The missed alarm can be checked using the `Smu_GetRTMissedEvent()` API.

1.1.5.5 SMU register monitoring

The module specific Safety flip-flop (SFF) can be tested using the interface provided to the Register Monitor. The triggering of SFF tests can be achieved by using the `Smu_RegisterMonitor()` API. For this the user shall check whether the module is applicable for the particular device derivative. Additionally, the user shall take care of the prerequisites for safety flip-flop test as mentioned in the HW User Manual before invoking the API. The user can enable the test, record the test results and disable the test using the runtime service of the SMU driver.

1.2 Assumptions of Use (AoU)

The AoU for the SMU driver are as follows.

- **Clearing of RT missed events**

The user shall explicitly clear the RT missed events detected. Currently there is no service provided by the SMU driver to clear the RT missed events.
[cover parentID SMU={79E76BAC-C9CD-409c-8518-B9778796261D}]

- **ConfigPtr passed to InitCheck**

User of SMU shall ensure that InitCheck is invoked with the same ConfigPtr that is used in Init.
[cover parentID SMU={402AA0C2-7E70-4192-B95C-32561B2B846C}]

- **Initialization check**

The user shall call the Smu_InitCheck API after initialization but before calling any SMU Runtime API and before releasing vehicle safe state.
[cover parentID SMU={6779F95B-9003-4e32-A5CA-4E072E2BB8CF}]

- **Non-interference check**

The user shall check that the correct config pointer has been passed and there is no interference to MCAL from other modules.
[cover parentID SMU={B5E820B9-2097-4917-BC6C-60B5A523F429}]

- **SFF test module check**

The user shall check if the module for which SFF test has been requested is available or not in the particular device.
[cover parentID SMU={9A7FBC44-B881-4469-9D42-CA6507F9A712}]

- **SMU FSP functionality**

The Smu_Release FSP API is asynchronous and the transition from the FAULT to the RUN state may require several cycles based on the fault state duration configured by the user, control PAD characteristics and/or recurrent fault occurrences.

The Smu_Activate FSP API is asynchronous and the transition to the FAULT state may require several cycles based on the control PAD characteristics.

Therefore, there is no deterministic time frame within which the state transition can be checked by the driver. The user shall ensure the transition to the intended state has occurred in the SMU_core. The user can check this using the Smu_GetSmuState API.
[cover parentID SMU={D1FB1959-3FB3-4123-92D8-226B551E6129}]

1.3 Reference information

1.3.1 Configuration interfaces

Supported configuration variant: Post-Build

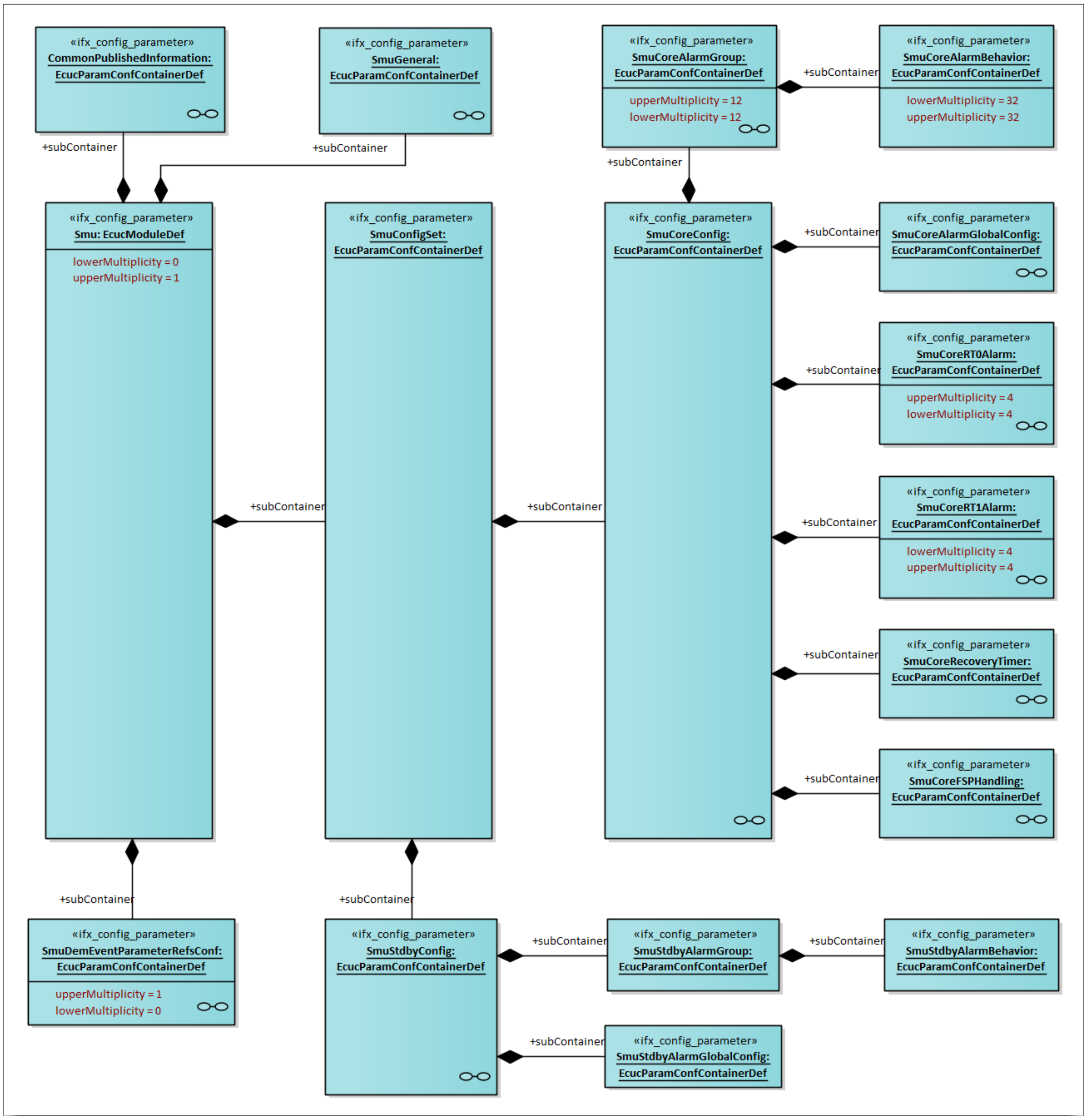

«ifx_config_parameter» CommonPublishedInformation: EcucParamConfContainerDef
«ifx_config_parameter» SmuGeneral: EcucParamConfContainerDef
«ifx_config_parameter» SmuCoreAlarmGroup: EcucParamConfContainerDef
upperMultiplicity = 12
lowerMultiplicity = 12
«ifx_config_parameter» SmuCoreAlarmBehavior: EcucParamConfContainerDef
lowerMultiplicity = 32
upperMultiplicity = 32
«ifx_config_parameter» Smu: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
«ifx_config_parameter» SmuConfigSet: EcucParamConfContainerDef
«ifx_config_parameter» SmuCoreConfig: EcucParamConfContainerDef
«ifx_config_parameter» SmuCoreAlarmGlobalConfig: EcucParamConfContainerDef
«ifx_config_parameter» SmuCoreRT0Alarm: EcucParamConfContainerDef
upperMultiplicity = 4
lowerMultiplicity = 4
«ifx_config_parameter» SmuCoreRT1Alarm: EcucParamConfContainerDef
lowerMultiplicity = 4
upperMultiplicity = 4
«ifx_config_parameter» SmuCoreRecoveryTimer: EcucParamConfContainerDef
«ifx_config_parameter» SmuCoreFSPHandling: EcucParamConfContainerDef
«ifx_config_parameter» SmuDemEventParameterRefsConf: EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
«ifx_config_parameter» SmuStdbyConfig: EcucParamConfContainerDef
«ifx_config_parameter» SmuStdbyAlarmGroup: EcucParamConfContainerDef
«ifx_config_parameter» SmuStdbyAlarmBehavior: EcucParamConfContainerDef
«ifx_config_parameter» SmuStdbyAlarmGlobalConfig: EcucParamConfContainerDef
+subContainer

Figure 12 **Container hierarchy along with their configuration parameters**

1.3.1.1 Container: CommonPublishedInformation

The container gives the published information for SMU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.1.1 ArMajorVersion

Table 4 Specification for ArMajorVersion

Name	`ArMajorVersion`		
Description	The configuration parameter provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.2 ArMinorVersion

Table 5 Specification for ArMinorVersion

Name	`ArMinorVersion`		
Description	The configuration parameter provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.3 ArPatchVersion

Table 6 Specification for ArPatchVersion

Name	`ArPatchVersion`

(table continues...)

Table 6 (continued) Specification for ArPatchVersion

Description	The configuration parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.4 ModuleId

Table 7 Specification for ModuleId

Name	`ModuleId`		
Description	The configuration parameter defines the module ID of SMU module from module list.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	255		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.5 Release

Table 8 Specification for Release

Name	`Release`		
Description	The configuration parameter defines the AURIX derivative used for the implementation.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		

(table continues...)

Table 8 **(continued) Specification for Release**

Default value	As per HW derivative		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.6 SwMajorVersion

Table 9 **Specification for SwMajorVersion**

Name	`SwMajorVersion`		
Description	The configuration parameter defines the major version number of the vendor specific implementation of the module. The numbering is vendor specific.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.7 SwMinorVersion

Table 10 **Specification for SwMinorVersion**

Name	`SwMinorVersion`		
Description	The configuration parameter defines the minor version number of the vendor specific implementation of the module. The numbering is vendor specific.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		

(table continues...)

Table 10 (continued) Specification for SwMinorVersion

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.8 SwPatchVersion

Table 11 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	The configuration parameter defines the patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.9 VendorId

Table 12 Specification for VendorId

Name	`VendorId`		
Description	The configuration parameter defines the vendor ID of the dedicated implementation of the module according to the AUTOSAR vendor list.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 12 (continued) Specification for VendorId

Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2 Container: Smu

The Smu container is the parent container for the SMU module.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

1.3.1.3 Container: SmuConfigSet

The container contains SmuConfigSet configurations for the SMU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.4 Container: SmuCoreAlarmBehavior

The container contains configuration parameters related to alarm behavior. Each alarm group has thirty two alarm configurations. Both the internal and external behavior can be configured for every alarm.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.4.1 SmuCoreAlarmFSP

Table 13 Specification for SmuCoreAlarmFSP

Name	`SmuCoreAlarmFSP`		
Description	The configuration parameter defines the value of the FSP configuration. The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_ALARM_FSP_DISABLED: The configuration parameter literal defines that FSP is disabled. SMU_ALARM_FSP_ENABLED: The configuration parameter literal defines that FSP is enabled.		
Default value	SMU_ALARM_FSP_DISABLED		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 13 (continued) Specification for SmuCoreAlarmFSP

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.2 SmuCoreAlarmIntBeh

Table 14 Specification for SmuCoreAlarmIntBeh

Name	`SmuCoreAlarmIntBeh`		
Description	The configuration parameter defines the internal behavior of an alarm event. The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_CPU_RESET_INT_ACTION: The configuration parameter literal defines the internal behavior as sending CPU reset configuration request. SMU_IGCS0_INT_ACTION: The configuration parameter literal defines the internal behavior as sending an interrupt request to the interrupt system according to the interrupt generation configuration set 0. SMU_IGCS1_INT_ACTION: The configuration parameter literal defines the internal behavior as sending an interrupt request to the interrupt system according to the interrupt generation configuration set 1 SMU_IGCS2_INT_ACTION: The configuration parameter literal defines the internal behavior as sending an interrupt request to the interrupt system according to the interrupt generation configuration set 2 SMU_NA_INT_ACTION: The configuration parameter literal defines the internal behavior as no action (default value). SMU_NMI_INT_ACTION: The configuration parameter literal defines the internal behavior as sending NMI request to the SCU. SMU_RESET_INT_ACTION: The configuration parameter literal defines the internal behavior as sending reset request to the SCU.		
Default value	SMU_NA_INT_ACTION		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.3 SmuCoreAlmBehaviourId

Table 15 Specification for SmuCoreAlmBehaviourId

Name	`SmuCoreAlmBehaviourId`		
Description	The configuration parameter defines the alarm behavior ID corresponding to the particular group. First alarm behavior is selected as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5 Container: SmuCoreAlarmGlobalConfig

The container contains the alarm global configuration parameters. The parameters are used for initializing the SMU_AGC register.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 SmuCoreCpu0ResetRequest

Table 16 Specification for SmuCoreCpu0ResetRequest

Name	`SmuCoreCpu0ResetRequest`		
Description	The configuration parameter is a Boolean which denotes whether the reset request to CPU0 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 16 **(continued) Specification for SmuCoreCpu0ResetRequest**

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.2 SmuCoreCpu1ResetRequest

Table 17 **Specification for SmuCoreCpu1ResetRequest**

Name	`SmuCoreCpu1ResetRequest`		
Description	The configuration parameter is a Boolean which denotes whether the reset request to CPU1 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.3 SmuCoreCpu2ResetRequest

Table 18 **Specification for SmuCoreCpu2ResetRequest**

Name	`SmuCoreCpu2ResetRequest`
Description	The configuration parameter is a Boolean which denotes whether the reset request to CPU2 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.*

(table continues...)

Table 18 (continued) Specification for SmuCoreCpu2ResetRequest

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.4 SmuCoreCpu3ResetRequest

Table 19 Specification for SmuCoreCpu3ResetRequest

Name	`SmuCoreCpu3ResetRequest`		
Description	The configuration parameter is a Boolean which denotes whether the reset request to CPU3 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.5 SmuCoreCpu4ResetRequest

Table 20 Specification for SmuCoreCpu4ResetRequest

Name	`SmuCoreCpu4ResetRequest`		
Description	The configuration parameter is a Boolean which denotes whether the reset request to CPU4 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.6 SmuCoreCpu5ResetRequest

Table 21 Specification for SmuCoreCpu5ResetRequest

Name	`SmuCoreCpu5ResetRequest`		
Description	The configuration parameter is a Boolean which denotes whether the reset request to CPU5 is set or not. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The availability of this parameter is dependent on the availability of the respective CPU in the particular device.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 21 (continued) Specification for SmuCoreCpu5ResetRequest

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.7 SmuCoreCpuResetActivatePES

Table 22 Specification for SmuCoreCpuResetActivatePES

Name	`SmuCoreCpuResetActivatePES`		
Description	The configuration parameter enables the PES on CPU reset. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.8 SmuCoreEnableFaultToRunState

Table 23 Specification for SmuCoreEnableFaultToRunState

Name	`SmuCoreEnableFaultToRunState`		
Description	The configuration parameter defines whether the FAULT state to RUN state transition is enabled or disabled. The state transition is possible only when this parameter is defined with SMU_EFRST_ENABLE. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 23 (continued) Specification for SmuCoreEnableFaultToRunState

Range	SMU_EFRST_DISABLE: The configuration parameter literal defines that the FAULT state to RUN state transition is disabled. SMU_EFRST_ENABLE: The configuration parameter literal defines that the FAULT state to RUN state transition is enabled.		
Default value	SMU_EFRST_DISABLE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.9 SmuCoreIGCS0ActivatePES

Table 24 Specification for SmuCoreIGCS0ActivatePES

Name	`SmuCoreIGCS0ActivatePES`		
Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for IGCS0 internal action. When an IGCS0 internal action is triggered, the hardware triggers automatically the PES, on enabling. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.10 SmuCoreIGCS1ActivatePES

Table 25 Specification for SmuCoreIGCS1ActivatePES

Name	`SmuCoreIGCS1ActivatePES`

(table continues...)

Table 25 (continued) Specification for SmuCoreIGCS1ActivatePES

Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for IGCS1 internal action. When an IGCS1 internal action is triggered, the hardware triggers automatically the PES, on enabling. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.11 SmuCoreIGCS2ActivatePES

Table 26 Specification for SmuCoreIGCS2ActivatePES

Name	`SmuCoreIGCS2ActivatePES`		
Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for IGCS2 internal action. When an IGCS2 internal action is triggered, the hardware triggers automatically the PES, on enabling. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.12 SmuCoreInterruptSet0

Table 27 Specification for SmuCoreInterruptSet0

Name	`SmuCoreInterruptSet0`		
Description	The configuration parameter defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 0. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_SELECT_INT0: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0. SMU_SELECT_INT0_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU1. SMU_SELECT_INT0_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT0_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU2. SMU_SELECT_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1. SMU_SELECT_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU2. SMU_SELECT_INT_NONE: The configuration parameter literal defines the output value of the interrupt request vector as no interrupt selected.		
Default value	SMU_SELECT_INT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.13 SmuCoreInterruptSet1

Table 28 Specification for SmuCoreInterruptSet1

Name	`SmuCoreInterruptSet1`
Description	The configuration parameter defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 1. The default value is this parameter is set to the reset value of the corresponding SFR.

(table continues...)

Table 28 (continued) Specification for SmuCoreInterruptSet1

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_SELECT_INT0: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0. SMU_SELECT_INT0_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU1. SMU_SELECT_INT0_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT0_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU2. SMU_SELECT_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1. SMU_SELECT_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU2. SMU_SELECT_INT_NONE: The configuration parameter literal defines the output value of the interrupt request vector as no interrupt selected.		
Default value	SMU_SELECT_INT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.14 SmuCoreInterruptSet2

Table 29 Specification for SmuCoreInterruptSet2

Name	`SmuCoreInterruptSet2`		
Description	The configuration parameter defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 2. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 29 (continued) Specification for SmuCoreInterruptSet2

Range	SMU_SELECT_INT0: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0. SMU_SELECT_INT0_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0 and SRC_SMU1. SMU_SELECT_INT0_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT0_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU0, SRC_SMU2. SMU_SELECT_INT1: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1. SMU_SELECT_INT1_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU1 and SRC_SMU2. SMU_SELECT_INT2: The configuration parameter literal defines the output value of the interrupt request vector as SRC_SMU2. SMU_SELECT_INT_NONE: The configuration parameter literal defines the output value of the interrupt request vector as no interrupt selected.		
Default value	SMU_SELECT_INT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.15 SmuCoreNMIActivatePES

Table 30 Specification for SmuCoreNMIActivatePES

Name	`SmuCoreNMIActivatePES`		
Description	The configuration parameter defines the control of the Port Emergency Stop (PES) feature for NMI internal action. When an NMI internal action is triggered, the hardware triggers automatically the PES, on enabling. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 30 (continued) Specification for SmuCoreNMIActivatePES

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6 Container: SmuCoreAlarmGroup

The container contains the configuration parameters for SMU_core alarm groups.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.6.1 SmuCoreAlmGrpId

Table 31 Specification for SmuCoreAlmGrpId

Name	`SmuCoreAlmGrpId`		
Description	The configuration parameter defines group ID of the SMU alarm group. The value will be assigned to the symbolic name derived from the AlarmGroup container short name. First alarm group is selected as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 11		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7 Container: SmuCoreConfig

The container contains the configuration parameters related to SMU_core.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.8 Container: SmuCoreFSPHandling

The container contains the configuration parameters related to SMU_core FSP handling.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.8.1 SmuCoreFSPFaultStateDuration

Table 32 Specification for SmuCoreFSPFaultStateDuration

Name	`SmuCoreFSPFaultStateDuration`		
Description	The configuration parameter enables the maximum fault state duration of FSP signal. The fault duration value is set at bit field, TFSP_HIGH of FSP Register. The configuration parameter is specified as a number of SMU_FS ticks. The default value is the intermediate value of the SFR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 1023		
Default value	1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.2 SmuCoreFSPPrescaler1

Table 33 Specification for SmuCoreFSPPrescaler1

Name	`SmuCoreFSPPrescaler1`		
Description	The configuration parameter defines the dividing factor to apply to the reference clock fBACK. The divided clock is used as reference to generate the timing of the fault signaling protocol fault state. The frequency of the divided clock is F(SMU_FS). The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 33 **(continued) Specification for SmuCoreFSPPrescaler1**

Range	SMU_REF_CLK_FRQ_DIV_128: The configuration parameter literal defines that the reference clock frequency is divided by 128. SMU_REF_CLK_FRQ_DIV_16: The configuration parameter literal defines that the reference clock frequency is divided by 16. SMU_REF_CLK_FRQ_DIV_256: The configuration parameter literal defines that the reference clock frequency is divided by 256. SMU_REF_CLK_FRQ_DIV_2: The configuration parameter literal defines that the reference clock frequency is divided by 2. SMU_REF_CLK_FRQ_DIV_32: The configuration parameter literal defines that the reference clock frequency is divided by 32. SMU_REF_CLK_FRQ_DIV_4: The configuration parameter literal defines that the reference clock frequency is divided by 4. SMU_REF_CLK_FRQ_DIV_64: The configuration parameter literal defines that the reference clock frequency is divided by 64. SMU_REF_CLK_FRQ_DIV_8: The configuration parameter literal defines that the reference clock frequency is divided by 8.		
Default value	SMU_REF_CLK_FRQ_DIV_2		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.3 SmuCoreFSPPrescaler2

Table 34 **Specification for SmuCoreFSPPrescaler2**

Name	`SmuCoreFSPPrescaler2`		
Description	The configuration parameter defines the dividing factor to apply to the reference clock fBACK. The divided clock is used as reference to generate the timing of the fault free state for the dynamic dual rail and time switching modes of the fault signalling protocol. The frequency of the divided clock is F(SMU_FFS). The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 34 **(continued) Specification for SmuCoreFSPPrescaler2**

Range	SMU_REF_CLK_FRQ_DIV_1024: The configuration parameter literal defines that the reference clock frequency is divided by 1024. SMU_REF_CLK_FRQ_DIV_2048: The configuration parameter literal defines that the reference clock frequency is divided by 2048. SMU_REF_CLK_FRQ_DIV_4096: The configuration parameter literal defines that the reference clock frequency is divided by 4096. SMU_REF_CLK_FRQ_DIV_512: The configuration parameter literal defines that the reference clock frequency is divided by 512.		
Default value	SMU_REF_CLK_FRQ_DIV_512		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.4 SmuCoreFSPSignalingMode

Table 35 **Specification for SmuCoreFSPSignalingMode**

Name	`SmuCoreFSPSignalingMode`		
Description	The configuration parameter defines the type of signal output for FSP on an alarm event. The default value of the parameter is set to the reset value of corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_FSP_BISTABLE_PROTOCOL: The configuration parameter literal defines that bistable protocol is used for FSP handling. SMU_FSP_DUAL_RAIL_PROTOCOL: The configuration parameter literal defines that dual rail protocol is used for FSP handling. SMU_FSP_TIME_SWITCHING_PROTOCOL: The configuration parameter literal defines that time switching protocol is used for FSP handling.		
Default value	SMU_FSP_BISTABLE_PROTOCOL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.5 SmuCorePESOnFSP

Table 36 Specification for SmuCorePESOnFSP

Name	`SmuCorePESOnFSP`		
Description	The configuration parameter defines whether the PES is to be automatically requested when an alarm event configured to start the FSP is detected. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_FSP_PES_DISABLE: The configuration parameter literal defines that PES feature is disabled. SMU_FSP_PES_ENABLE: The configuration parameter literal defines that PES feature is enabled.		
Default value	SMU_FSP_PES_DISABLE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9 Container: SmuCoreRecoveryTimer

The container contains the configuration parameters for SMU_core recovery timer.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.9.1 SmuCoreEnableRT0

Table 37 Specification for SmuCoreEnableRT0

Name	`SmuCoreEnableRT0`		
Description	The configuration parameter defines whether RT0 is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 37 (continued) Specification for SmuCoreEnableRT0

Range	SMU_RT_DISABLE: The configuration parameter literal defines that the recovery timer is disabled. Value: 0 SMU_RT_ENABLE: The configuration parameter literal defines that the recovery timer is enabled. Value: 1		
Default value	SMU_RT_DISABLE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.2 SmuCoreEnableRT1

Table 38 Specification for SmuCoreEnableRT1

Name	`SmuCoreEnableRT1`		
Description	The configuration parameter defines whether RT1 is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_RT_DISABLE: The configuration parameter literal defines that the recovery timer is disabled. Value: 0 SMU_RT_ENABLE: The configuration parameter literal defines that the recovery timer is enabled. Value: 1		
Default value	SMU_RT_DISABLE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.3 SmuCoreRTDuration

Table 39 **Specification for SmuCoreRTDuration**

Name	`SmuCoreRTDuration`		
Description	The configuration parameter defines the maximum duration of SMU_core recovery timer. The maximum duration is specified as a number of the F(SMU_FS) clock ticks. The default value is the intermediate value of the tick duration.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 0xFFFFFFU		
Default value	0x3FFFU		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10 Container: SmuCoreRT0Alarm

The container enables to select the alarms for RT0. Four alarms can be configured per recovery timer instance.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.10.1 SmuCoreRT0AlarmGroupId

Table 40 **Specification for SmuCoreRT0AlarmGroupId**

Name	`SmuCoreRT0AlarmGroupId`		
Description	The configuration parameter defines the alarm group ID associated with the RT0. First alarm group is selected as the default value.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 40 (continued) Specification for SmuCoreRT0AlarmGroupId

Range	SMU_ALARM_GROUP0: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 0. SMU_ALARM_GROUP10: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 10. SMU_ALARM_GROUP11: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 11. SMU_ALARM_GROUP1: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 1. SMU_ALARM_GROUP2: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 2. SMU_ALARM_GROUP3: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 3. SMU_ALARM_GROUP4: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 4. SMU_ALARM_GROUP5: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 5. SMU_ALARM_GROUP6: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 6. SMU_ALARM_GROUP7: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 7. SMU_ALARM_GROUP8: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 8. SMU_ALARM_GROUP9: The configuration parameter literal defines that the alarm group ID associated with the RT0 corresponds to alarm group 9.		
Default value	SMU_ALARM_GROUP0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuCoreEnableRT0		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.2 SmuCoreRT0AlarmId

Table 41 Specification for SmuCoreRT0AlarmId

Name	`SmuCoreRT0AlarmId`		
Description	The configuration parameter defines the alarm ID associated with the RT0. First alarm ID is selected as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 41 (continued) Specification for SmuCoreRT0AlarmId

Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuCoreEnableRT0		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.11 Container: SmuCoreRT1Alarm

The container enables to select the alarms for RT1. Four alarms can be configured per recovery timer instance.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.11.1 SmuCoreRT1AlarmGroupId

Table 42 Specification for SmuCoreRT1AlarmGroupId

Name	`SmuCoreRT1AlarmGroupId`		
Description	The configuration parameter defines the alarm group ID associated with the RT1. First alarm group is selected as the default value.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 42 (continued) Specification for SmuCoreRT1AlarmGroupId

Range	SMU_ALARM_GROUP0: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 0. SMU_ALARM_GROUP10: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 10. SMU_ALARM_GROUP11: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 11. SMU_ALARM_GROUP1: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 1. SMU_ALARM_GROUP2: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 2. SMU_ALARM_GROUP3: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 3. SMU_ALARM_GROUP4: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 4. SMU_ALARM_GROUP5: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 5. SMU_ALARM_GROUP6: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 6. SMU_ALARM_GROUP7: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 7. SMU_ALARM_GROUP8: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 8. SMU_ALARM_GROUP9: The configuration parameter literal defines that the alarm group ID associated with the RT1 corresponds to alarm group 9.		
Default value	SMU_ALARM_GROUP0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuCoreEnableRT1		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.11.2 SmuCoreRT1AlarmId

Table 43 Specification for SmuCoreRT1AlarmId

Name	`SmuCoreRT1AlarmId`		
Description	The configuration parameter defines the alarm ID associated with the RT1. First alarm ID is selected as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 43 (continued) Specification for SmuCoreRT1AlarmId

Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuCoreEnableRT1		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12 Container: SmuDemEventParameterRefsConf

The container lists down the production errors supported by the SMU driver.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

1.3.1.12.1 SmuActivateFSPFailureNotification

Table 44 Specification for SmuActivateFSPFailureNotification

Name	`SmuActivateFSPFailureNotification`		
Description	The configuration parameter tells whether the notification for Production Error incase of failure to activate FSP is enabled or not.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.2 SmuActivatePESFailureNotification

Table 45 Specification for SmuActivatePESFailureNotification

Name	`SmuActivatePESFailureNotification`

(table continues...)

Table 45 (continued) Specification for SmuActivatePESFailureNotification

Description	The configuration parameter tells whether the notification for Production Error related to failure of PES is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.3 SmuActivateRunStateFailureNotification

Table 46 Specification for SmuActivateRunStateFailureNotification

Name	`SmuActivateRunStateFailureNotification`		
Description	The configuration parameter tells whether the notification for Production Error related to failure to activate RUN state is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.4 SmuClearAlarmStatusFailureNotification

Table 47 Specification for SmuClearAlarmStatusFailureNotification

Name	`SmuClearAlarmStatusFailureNotification`

(table continues...)

Table 47 (continued) Specification for SmuClearAlarmStatusFailureNotification

Description	The configuration parameter tells whether the notification for Production Error related to failure to clear alarm status is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.5 SmuCoreAliveFailureNotification

Table 48 Specification for SmuCoreAliveFailureNotification

Name	`SmuCoreAliveFailureNotification`		
Description	The configuration parameter tells whether the notification for Production Error related to SMU_core_alive test failure is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.6 SmuRTStopFailureNotification

Table 49 Specification for SmuRTStopFailureNotification

Name	`SmuRTStopFailureNotification`

(table continues...)

Table 49 (continued) Specification for SmuRTStopFailureNotification

Description	The configuration parameter tells whether the notification for Production Error related to failure to stop recovery timer is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.7 SmuReleaseFSPFailureNotification

Table 50 Specification for SmuReleaseFSPFailureNotification

Name	`SmuReleaseFSPFailureNotification`		
Description	The configuration parameter tells whether the notification for Production Error related to failure to release FSP is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.8 SmuSetAlarmStatusFailureNotification

Table 51 Specification for SmuSetAlarmStatusFailureNotification

Name	`SmuSetAlarmStatusFailureNotification`

(table continues...)

Table 51 (continued) Specification for SmuSetAlarmStatusFailureNotification

Description	The configuration parameter tells whether the notification for Production Error related to failure to set alarm status is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.9 SmuSffFailureNotification

Table 52 Specification for SmuSffFailureNotification

Name	`SmuSffFailureNotification`		
Description	The configuration parameter tells whether the notification for Production Error related to SFF test failure is enabled or disabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	SmuSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13 Container: SmuGeneral

The container contains the general configurations of the SMU driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.13.1 SmuCoreFSP0OutputEnable

Table 53 Specification for SmuCoreFSP0OutputEnable

Name	`SmuCoreFSP0OutputEnable`		
Description	The configuration parameter sets FSP[0] state to output, if true. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.2 SmuCoreFSP0PortEnable

Table 54 Specification for SmuCoreFSP0PortEnable

Name	`SmuCoreFSP0PortEnable`		
Description	The configuration parameter sets FSP[0] PORT enable to true. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.3 SmuCoreFSP1OutputEnable

Table 55 **Specification for SmuCoreFSP1OutputEnable**

Name	`SmuCoreFSP1OutputEnable`		
Description	The configuration parameter sets FSP[1] state to output, if true. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.4 SmuCoreFSP1PortEnable

Table 56 **Specification for SmuCoreFSP1PortEnable**

Name	`SmuCoreFSP1PortEnable`		
Description	The configuration parameter sets FSP[1] port enable to true. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.5 SmuCoreGlitchFilterSCU

Table 57 Specification for SmuCoreGlitchFilterSCU

Name	`SmuCoreGlitchFilterSCU`		
Description	The configuration parameter sets glitch filter for SCU to enabled state. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.6 SmuCoreGlitchFilterSTS

Table 58 Specification for SmuCoreGlitchFilterSTS

Name	`SmuCoreGlitchFilterSTS`		
Description	The configuration parameter sets glitch filter for SMU_STS to be enabled. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.7 SmuDevErrorDetect

Table 59 **Specification for SmuDevErrorDetect**

Name	`SmuDevErrorDetect`		
Description	The configuration parameter enables or disables DET checks.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.8 SmuInitCheckApi

Table 60 **Specification for SmuInitCheckApi**

Name	`SmuInitCheckApi`		
Description	The configuration parameter enables or disables the Smu_InitCheck API. The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.9 SmuInitDeInitApiMode

Table 61 Specification for SmuInitDeInitApiMode

Name	`SmuInitDeInitApiMode`		
Description	The configuration parameter defines the mode in which the Init and Deinit API will be used. Since SMU driver accesses the SFRs, it is more efficient to operate the SMU driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_MCAL_SUPERVISOR: The configuration parameter implies that SUPERVISOR mode is used. The parameter takes value 0 when assigned. SMU_MCAL_USER1: The configuration parameter implies that USER1 mode is used. The parameter takes values 1 when assigned.		
Default value	SMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.10 SmuRuntimeApiMode

Table 62 Specification for SmuRuntimeApiMode

Name	`SmuRuntimeApiMode`		
Description	The configuration parameter gives the mode in which the runtime API will be used. Since SMU driver accesses the SFRs, it is more efficient to operate the SMU driver in supervisor mode. Hence, the default mode of operation is supervisor. When the parameter is in supervisor mode, then the SmuInitDeIntMode is in supervisor mode.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_MCAL_SUPERVISOR: The configuration parameter implies that SUPERVISOR mode is used. The parameter takes value 0 when assigned. SMU_MCAL_USER1: The configuration parameter implies that USER1 mode is used. The parameter takes value 1 when assigned.		
Default value	SMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 62 (continued) Specification for SmuRuntimeApiMode

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuInitDeInitApiMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.11 SmuSafetyEnable

Table 63 Specification for SmuSafetyEnable

Name	`SmuSafetyEnable`		
Description	The configuration parameter defines whether the safety checks mandated by safety standards are enabled or disabled. The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.12 SmuStdbyEnable

Table 64 Specification for SmuStdbyEnable

Name	`SmuStdbyEnable`		
Description	The configuration parameter defines whether the SMU_stdby unit is enabled or disabled. The default value is this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

(table continues...)

Table 64 (continued) Specification for SmuStdbyEnable

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.13 SmuVersionInfoApi

Table 65 Specification for SmuVersionInfoApi

Name	`SmuVersionInfoApi`		
Description	The configuration parameter enables or disables the VersionInfo API. The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14 Container: SmuStdbyAlarmBehavior

The container contains configuration parameters corresponding to alarm behavior. The behavior type is external if FSP is enabled or no reaction.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.14.1 SmuStdbyAlarmFSP

Table 66 Specification for SmuStdbyAlarmFSP

Name	`SmuStdbyAlarmFSP`

(table continues...)

Table 66 (continued) Specification for SmuStdbyAlarmFSP

Description	The configuration parameter defines whether the FSP is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SMU_ALARM_FSP_DISABLED: The configuration parameter literal defines that FSP is disabled. SMU_ALARM_FSP_ENABLED: The configuration parameter literal defines that the FSP is enabled.		
Default value	SMU_ALARM_FSP_DISABLED		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuStdbyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14.2 SmuStdbyAlmBehaviourId

Table 67 Specification for SmuStdbyAlmBehaviourId

Name	`SmuStdbyAlmBehaviourId`		
Description	The configuration parameter defines the alarm ID corresponding to the particular group. First alarm behavior id is selected as the default value..		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuStdbyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.15 Container: SmuStdbyAlarmGlobalConfig

The container contains the configuration parameters related to SMU_stdby global configurations.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.15.1 SmuStdbyEnableFSP0

Table 68 **Specification for SmuStdbyEnableFSP0**

Name	`SmuStdbyEnableFSP0`		
Description	The configuration parameter defines whether the use of FSP[0] P33.8 pin is enabled or disabled for FSP handling. The default value of the parameter is the reset value of the SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuStdbyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.15.2 SmuStdbyEnableFSP1

Table 69 **Specification for SmuStdbyEnableFSP1**

Name	`SmuStdbyEnableFSP1`		
Description	The configuration parameter defines whether the use of FSP[1] P33.10 pin is enabled or disabled for FSP handling. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuStdbyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.16 Container: SmuStdbyAlarmGroup

The container contains the configuration parameters related to SMU_stdby alarm groups.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.16.1 SmuStdbyAlmGrpId

Table 70 Specification for SmuStdbyAlmGrpId

Name	`SmuStdbyAlmGrpId`		
Description	The configuration parameter defines the alarm group ID of the alarm group to be configured. First group id is selected as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SmuStdbyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.17 Container: SmuStdbyConfig

The container contains the configuration parameters related to SMU_stdby.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.2 Functions - Type definitions

This section lists all the data type of the SMU driver.

1.3.2.1 Smu_AlarmGroupId

Table 71 Specification for Smu_AlarmGroupId

Syntax	`Smu_AlarmGroupId`	
Type	Enumeration	
File	`Smu.h`	
Range	0 - SMU_GROUP_0	None
	1 - SMU_GROUP_1	None

(table continues...)

Table 71 **(continued) Specification for Smu_AlarmGroupId**

	2 - SMU_GROUP_2	None
	3 - SMU_GROUP_3	None
	4 - SMU_GROUP_4	None
	5 - SMU_GROUP_5	None
	6 - SMU_GROUP_6	None
	7 - SMU_GROUP_7	None
	8 - SMU_GROUP_8	None
	9 - SMU_GROUP_9	None
	10 - SMU_GROUP_10	None
	11 - SMU_GROUP_11	None
	20 - SMU_GROUP_20	None
	21 - SMU_GROUP_21	None
Description	Smu_AlarmGroupId enumeration gives the alarm group ID for each group in SMU_core and SMU_stdby.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.2 Smu_AlarmIdType

Table 72 **Specification for Smu_AlarmIdType**

Syntax	`Smu_AlarmIdType`	
Type	Enumeration	
File	`Smu.h`	
Range	0 - SMU_ALARM_0	None
	1 - SMU_ALARM_1	None
	2 - SMU_ALARM_2	None
	3 - SMU_ALARM_3	None
	4 - SMU_ALARM_4	None
	5 - SMU_ALARM_5	None
	6 - SMU_ALARM_6	None
	7 - SMU_ALARM_7	None
	8 - SMU_ALARM_8	None
	9 - SMU_ALARM_9	None
	10 - SMU_ALARM_10	None
	11 - SMU_ALARM_11	None

(table continues...)

Table 72 **(continued) Specification for Smu_AlarmIdType**

	12 - SMU_ALARM_12	None
	13 - SMU_ALARM_13	None
	14 - SMU_ALARM_14	None
	15 - SMU_ALARM_15	None
	16 - SMU_ALARM_16	None
	17 - SMU_ALARM_17	None
	18 - SMU_ALARM_18	None
	19 - SMU_ALARM_19	None
	20 - SMU_ALARM_20	None
	21 - SMU_ALARM_21	None
	22 - SMU_ALARM_22	None
	23 - SMU_ALARM_23	None
	24 - SMU_ALARM_24	None
	25 - SMU_ALARM_25	None
	26 - SMU_ALARM_26	None
	27 - SMU_ALARM_27	None
	28 - SMU_ALARM_28	None
	29 - SMU_ALARM_29	None
	30 - SMU_ALARM_30	None
	31 - SMU_ALARM_31	None
Description	Smu_AlarmIdType enumeration gives the alarm ID associated with each alarm group.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.3 Smu_ConfigType

Table 73 **Specification for Smu_ConfigType**

Syntax	`Smu_ConfigType`	
Type	Structure	
File	`Smu.h`	
Range	-	None
Description	Smu_ConfigType defines the type of data structure containing the set of configuration parameters required for initializing the SMU driver and the SMU hardware unit.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.4 Smu_CoreAlarmActionType

Table 74 Specification for Smu_CoreAlarmActionType

Syntax	`Smu_CoreAlarmActionType`	
Type	uint8	
File	`Smu.h`	
Range	SMU_ALARM_ACTION_NONE	SMU_NA_ALARM_CONFIG implies that no action has to be taken on receiving an alarm.
	SMU_ALARM_ACTION_RSVD	SMU_RSVD_ALARM_CONFIG is reserved and no action is taken. Alarm is disabled.
	SMU_ALARM_ACTION_IGCS0	SMU_IGCS0_ALARM_CONFIG sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 0 from AGC register.
	SMU_ALARM_ACTION_IGCS1	SMU_IGCS1_ALARM_CONFIG sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 1 from AGC register.
	SMU_ALARM_ACTION_IGCS2	SMU_IGCS2_ALARM_CONFIG sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 2 from AGC register.
	SMU_ALARM_ACTION_NMI	SMU_NMI_ALARM_CONFIG sends an NMI request to the SCU.
	SMU_ALARM_ACTION_RESET	SMU_RESET_ALARM_CONFIG sends a RESET request to the SCU. SCU shall be configured to generate an application or system reset.
	SMU_ALARM_ACTION_CPU_RESET	SMU_CPU_RST_ALARM_CONFIG sets a CPU reset using CPU Reset Configuration set from the AGC register.
Description	Smu_CoreAlarmActionType defines the internal action behaviour for the alarms in SMU_core.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.5 Smu_CoreCommandType

Table 75 Specification for Smu_CoreCommandType

Syntax	`Smu_CoreCommandType`
Type	uint8

(table continues...)

Table 75 (continued) Specification for Smu_CoreCommandType

File	`Smu.h`	
Range	SMU_RUN_COMMAND	SMU_RUN_COMMAND makes the SMU_core enter the RUN state.
	SMU_ACTIVATEFSP_COMMAND	SMU_ACTIVATEFSP_COMMAND activates FSP for SMU_core.
	SMU_RELEASEFSP_COMMAND	SMU_RELEASEFSP_COMMAND releases FSP for SMU_core.
	SMU_ACTIVATE_PES	SMU_ACTIVATE_PES activates the PES feature for SMU_core.
	SMU_STOPREC_COMMAND	SMU_STOPREC_COMMAND stops the recovery timer for SMU_core.
	SMU_ASCE_COMMAND	SMU_ASCE_COMMAND is alarm status clear enable command for SMU_core. Software shall execute SMU_ASCE_COMMAND prior to clearing of a AG<n> alarm status bit. SMU_ASCE_COMMAND sets the ASCE bit in the STS register.
	SMU_ALARM_COMMAND	SMU_ALARM_COMMAND triggers a software based alarm. ARG specifies the alarm index.
	SMU_ALIVETEST_COMMAND	SMU_ALIVETEST_COMMAND enables the testing of the smu_core_alive signal. Sending SMU_ALIVETEST_COMMAND will forward the smu_core_alive alarm to the SMU_stdby. Argument ARG shall be set to 0x05 to start the test and to 0x0A to end the test.
Description	Smu_CoreCommandType describes the SMU_core command sets.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.6 Smu_CoreStateType

Table 76 Specification for Smu_CoreStateType

Syntax	`Smu_CoreStateType`
Type	uint8
File	`Smu.h`

(table continues...)

Table 76 **(continued) Specification for Smu_CoreStateType**

Range	SMU_START_STATE	SMU_START_STATE corresponds to the START state in the SSM. Value:0. The alarms shall be logged in but not processed during START state. Exception to this is the RT and Watchdog timeout alarms. Entry condition: PORST Exit condition: Releasing FSP and activating RUN state through SMU_core command.
	SMU_RUN_STATE	SMU_RUN_STATE corresponds to the RUN state in the SSM. Value:1 The alarms logged are processed. Entry condition: When FSP is released, RUN state is activated through SMU_core commands or FSP fault state timing is expired. Exit condition: When FSP is activated or alarm detected with FSP enabled.
	SMU_FAULT_STATE	SMU_FAULT_STATE corresponds to the FAULT state in the SSM. Value:2 SMU input alarm events are processed according to their configurations. FSP drives according to the configured reaction and timing. If a new FSP is detected, the FSP fault state timing is restarted. Entry condition: When alarm is detected. Exit condition: When FSP is released or FSP fault state timing is expired.
	SMU_UNDEFINED_STATE	SMU_UNDEFINED_STATE corresponds to the UNDEFINED state in the SSM. Value:3
Description	Smu_CoreStateType defines the various state types of SMU State Machine (SSM).	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.7 Smu_EnableRunStateType

Table 77 **Specification for Smu_EnableRunStateType**

Syntax	`Smu_EnableRunStateType`	
Type	Enumeration	
File	`Smu.h`	
Range	0 - SMU_EFRST_DISABLE	The Enable Fault To RUN state is disabled
	1 - SMU_EFRST_ENABLE	The Enable Fault To RUN state is enabled.

(table continues...)

Table 77 **(continued) Specification for Smu_EnableRunStateType**

Description	Smu_EnableRunStateType enumeration defines whether the fault to run state is enabled or disabled.
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.8 Smu_FSPActionType

Table 78 **Specification for Smu_FSPActionType**

Syntax	`Smu_FSPActionType`	
Type	uint32	
File	`Smu.h`	
Range	0 - 1	
Description	Smu_FSPActionType defines the FSP action type for the alarm group and ID.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.9 Smu_SffTestResType

Table 79 **Specification for Smu_SffTestResType**

Syntax	`Smu_SffTestResType`	
Type	uint8	
File	`Smu.h`	
Range	0 - 255	
Description	Smu_SffTestResType gives the SFF test results.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3 Functions - APIs

This section lists all the APIs of the SMU driver.

1.3.3.1 Smu_Init

Table 80 **Specification for `Smu_Init` API**

Syntax	`Std_ReturnType Smu_Init ( const Smu_ConfigType * const ConfigPtr )`
Service ID	0xA8

(table continues...)

Table 80 (continued) Specification for `Smu_Init` API

Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the SMU configuration for initialization.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is initialization of resources of AURIX SMU peripheral is successful. E_NOT_OK: Operation failed that is initialization of resources of AURIX SMU peripheral is not successful, for example, when driver is already initialized.
Description	The purpose of the API is to setup the SMU peripheral based on the configuration. The SMU driver initializes the resources of the AURIX SMU peripheral, for example the error reaction and the Fault Signaling Protocol (FSP). Initialization should be done only from the master core. During initialization, all the alarm statuses are cleared, hence the user must ensure to keep a track of the alarm status before Smu_Init is called. *Note: The API uses SMU temporary locking mechanism after configuration of SFRs to prevent accidental change in values of the SFRs.*	
Source	IFX	
Error handling	SMU_E_ALREADY_INITIALIZED, SMU_E_INIT_FAILED, SMU_E_CORE_MISMATCH, SMU_E_LOCKED	
Configuration dependencies	-	
User hints	Smu_Init API is responsible to clear the status of all SMU alarms. However, if the source of an alarm is not cleared or disabled, then the status of that alarm may remain set after execution of Smu_Init API even though the alarm had been cleared in the API.	
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), PMS_AGFSP_STDBY(w), PMS_AG_STDBY(w), PMS_CMD_STDBY(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AG(w), SMU_AGC(w), SMU_AGCF(w), SMU_AGFSP(w), SMU_CMD(w), SMU_FSP(w), SMU_KEYS(rw), SMU_RTAC00(w), SMU_RTAC01(w), SMU_RTAC10(w), SMU_RTAC11(w), SMU_RTC(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.2 Smu_DeInit

Table 81 Specification for `Smu_DeInit` API

Syntax	`Std_ReturnType Smu_DeInit ( void )`	
Service ID	0xAA	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful: de-initialization of SMU driver by resetting the module registers is successful. E_NOT_OK: Operation failed that is de-initialization of SMU driver by resetting the module registers is not successful, for example when driver is already reset.
Description	The purpose of the API is to de-initialize the SMU driver by resetting the module registers. De-initialization shall be done only from the master core. *Note: The API uses SMU temporary locking mechanism after configuration of SFRs to prevent accidental change in values of the SFRs.*	
Source	IFX	
Error handling	SMU_E_UNINIT, SMU_E_CORE_MISMATCH, SMU_E_LOCKED	
Configuration dependencies	-	
User hints	Smu_DeInit API is responsible to clear the status of all SMU alarms. However, if the source of an alarm is not cleared or disabled, then the status of that alarm may remain set after execution of Smu_DeInit API even though the alarm had been cleared in the API.	
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), PMS_AGFSP_STDBY(w), PMS_AG_STDBY(w), PMS_CMD_STDBY(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AG(w), SMU_AGC(w), SMU_AGCF(w), SMU_AGFSP(w), SMU_CMD(w), SMU_FSP(w), SMU_KEYS(rw), SMU_RTAC00(w), SMU_RTAC01(w), SMU_RTAC10(w), SMU_RTAC11(w), SMU_RTC(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	

(table continues...)

Table 81 (continued) Specification for `Smu_DeInit` API

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.3 Smu_GetAlarmAction

Table 82 Specification for `Smu_GetAlarmAction` API

Syntax	`Std_ReturnType Smu_GetAlarmAction ( const Smu_AlarmGroupId AlarmGroup, const Smu_AlarmIdType AlarmPos, Smu_CoreAlarmActionType * const IntAlarmAction, Smu_FSPActionType * const FSPAction )`	
Service ID	0xAB	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup AlarmPos	Alarm group number (0 - 11, 20, 21) Alarm position within the requested group (0 - 31)
Parameters (out)	IntAlarmAction FSPAction	Alarm action for the requested alarm FSP action for the requested alarm. (0- Disabled , 1- Enabled)
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of the internal alarm, FSP action currently configured for the requested alarm is successful. E_NOT_OK: Operation not successful that is invalid alarm action is returned, retrieval of the internal alarm, FSP action currently configured for the requested alarm is not successful, for example due to invalid parameters.
Description	The purpose of the API is to provide the internal alarm, FSP action currently configured for the requested alarm.	
Source	IFX	
Error handling	SMU_E_PARAM_POINTER, SMU_E_PARAM_GROUP, SMU_E_UNINIT, SMU_E_STDBY_DISABLED	
Configuration dependencies	-	
User hints	None	

(table continues...)

Table 82 (continued) Specification for `Smu_GetAlarmAction` API

SFR accessed	PMS_AGFSP_STDBY(r), SMU_AGCF(r), SMU_AGFSP(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.4 Smu_SetAlarmAction

Table 83 Specification for `Smu_SetAlarmAction` API

Syntax	`Std_ReturnType  Smu_SetAlarmAction`
	`(`
	`    const Smu_AlarmGroupId AlarmGroup,`
	`    const Smu_AlarmIdType AlarmPos,`
	`    const Smu_CoreAlarmActionType AlarmAction,`
	`    const Smu_FSPActionType FSPAction`
	`)`	

Service ID	0xAC	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup AlarmPos AlarmAction FSPAction	Alarm group number (0 - 11, 20, 21) Alarm position within the requested group (0-31) The internal alarm action for the requested alarm group and position The FSP action to be set. (0- Disabled, 1- Enabled)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: The alarm action is set. E_NOT_OK: The alarm action could not be set.
Description	The purpose of the API is to set the desired alarm action for the group and position specified. *Note: The API uses SMU temporary locking mechanism after configuration of SFRs to prevent accidental change in values of the SFRs.*	
Source	IFX	
Error handling	SMU_E_PARAM_GROUP, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE, SMU_E_LOCKED, SMU_E_INVALID_ALARM_ACTION, SMU_E_STDBY_DISABLED	
Configuration dependencies	-	

(table continues...)

Table 83 (continued) Specification for `Smu_SetAlarmAction` API

User hints	None
SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), PMS_AGFSP_STDBY(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AGCF(w), SMU_AGFSP(w), SMU_KEYS(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.5 Smu_ClearAlarmStatus

Table 84 Specification for `Smu_ClearAlarmStatus` API

Syntax	`Std_ReturnType  Smu_ClearAlarmStatus`
	`(`
	`    const Smu_AlarmGroupId AlarmGroup,`
	`    const Smu_AlarmIdType AlarmPos`
	`)`	

Service ID	0xAD	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup AlarmPos	Alarm group number. (0 - 11, 20, 21) Alarm position within the requested group (0 - 31)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: The alarm status is cleared successfully E_NOT_OK: The alarm status is not cleared successfully
Description	The purpose of the API is to clear SMU alarm status of the requested alarm.	
Source	IFX	
Error handling	SMU_E_CLEAR_ALARM_STATUS_FAILURE, SMU_E_PARAM_GROUP, SMU_E_STDBY_DISABLED, SMU_E_INVALID_DRIVER_STATE, SMU_E_UNINIT	
Configuration dependencies	-	
User hints	If the source of the requested alarm is not cleared or disabled, then the status of the alarm may remain set after execution of Smu_ClearAlarmStatus API even though the alarm had been cleared in the API.	

(table continues...)

Table 84 (continued) Specification for `Smu_ClearAlarmStatus` API

SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), PMS_AGFSP_STDBY(w), PMS_AG_STDBY(rw), PMS_CMD_STDBY(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AG(rw), SMU_CMD(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.6 Smu_GetAlarmStatus

Table 85 Specification for `Smu_GetAlarmStatus` API

Syntax	`Std_ReturnType Smu_GetAlarmStatus ( const Smu_AlarmGroupId AlarmGroup, uint32 * const AlarmStatus )`	
Service ID	0xAF	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup	Group id of the alarm raised. (0 - 11, 20, 21)
Parameters (out)	AlarmStatus	Status of the alarm raised
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of SMU alarm status of the requested alarm is successful. E_NOT_OK: Operation unsuccessful that is SMU alarm status of the requested alarm is not retrieved.
Description	The purpose of the API is to provide the alarm status of the requested alarm group.	
Source	IFX	
Error handling	SMU_E_PARAM_POINTER, SMU_E_PARAM_GROUP, SMU_E_UNINIT, SMU_E_STDBY_DISABLED	
Configuration dependencies	-	
User hints	None	

(table continues...)

Table 85 (continued) Specification for `Smu_GetAlarmStatus` API

SFR accessed	PMS_AG_STDBY(r), SMU_AG(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.7 Smu_SetAlarmStatus

Table 86 Specification for `Smu_SetAlarmStatus` API

Syntax	`Std_ReturnType  Smu_SetAlarmStatus`
	`(`
	`    const Smu_AlarmGroupId AlarmGroup,`
	`    const Smu_AlarmIdType AlarmPos`
	`)`	

Service ID	0xAE	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup AlarmPos	Alarm group number (0 - 11) Alarm position within the requested group (0-31)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is SMU alarm status of the requested alarm is set successfully. E_NOT_OK: Operation unsuccessful that is SMU alarm status of the requested alarm is not set.
Description	The purpose of the API is to set the requested alarm status. This service can be used by the user software to trigger software SMU alarm. For SMU_core during the START state of the SMU, it shall be possible to set any of the alarms. However, during the RUN state, only the software alarms shall be set. The API is applicable only for SMU_core alarm groups and positions.	
Source	IFX	
Error handling	SMU_E_SET_ALARM_STATUS_FAILURE, SMU_E_INVALID_DRIVER_STATE, SMU_E_PARAM_GROUP, SMU_E_UNINIT	
Configuration dependencies	-	
User hints	None	

(table continues...)

Table 86 (continued) Specification for `Smu_SetAlarmStatus` API

SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AG(rw), SMU_CMD(w), SMU_DBG(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.8 Smu_GetAlarmDebugStatus

Table 87 Specification for `Smu_GetAlarmDebugStatus` API

Syntax	`Std_ReturnType  Smu_GetAlarmDebugStatus`
	`(`
	`    const Smu_AlarmGroupId AlarmGroup,`
	`    uint32 * const AlarmStatus`
	`)`	

Service ID	0xB0	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmGroup	Alarm group number (0 - 11)
Parameters (out)	AlarmStatus	Alarm Debug Status register value
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of alarm status from debug register is successful. E_NOT_OK: Operation unsuccessful that is retrieval of SMU alarm debug status of the requested alarm is not successful.
Description	The purpose of the API is to provide the alarm status for the requested alarm group from the stored debug registers. The debug status is applicable only for SMU_core.	
Source	IFX	
Error handling	SMU_E_PARAM_POINTER, SMU_E_PARAM_GROUP, SMU_E_UNINIT	
Configuration dependencies	-	
User hints	This is required by the application to know the reason of the malfunction esp. In case the internal reaction was configured to be a reset. This is only for Smu_core.	

(table continues...)

Table 87 **(continued) Specification for `Smu_GetAlarmDebugStatus` API**

SFR accessed	SMU_AD(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.9 Smu_LockConfigRegs

Table 88 **Specification for `Smu_LockConfigRegs` API**

Syntax	`Std_ReturnType  Smu_LockConfigRegs`
	`(`
	`  void`
	`)`	

Service ID	0xB1	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is SMU configuration registers are locked successfully. E_NOT_OK: Operation not successful that is SMU configuration registers are not locked successfully.
Description	The purpose of the API is to permanently lock the SMU configuration registers to prevent any modification to configuration register content. The API can be called from any core. However, it is recommended to call the API from only one core at any instance of time to ensure consistent behavior.	
Source	IFX	
Error handling	SMU_E_INVALID_DRIVER_STATE, SMU_E_UNINIT, SMU_E_LOCKED, SMU_E_SF_CFG_LOCKED	
Configuration dependencies	-	
User hints	None	

(table continues...)

Table 88 (continued) Specification for `Smu_LockConfigRegs` API

SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AGC(rw), SMU_KEYS(rw), SMU_RTC(rw), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.10 Smu_ReleaseFSP

Table 89 Specification for `Smu_ReleaseFSP` API

Syntax	`Std_ReturnType  Smu_ReleaseFSP (   void )`	
Service ID	0xB2	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is setting the PCS bit is successful. E_NOT_OK: Operation not successful or SMU is already in RUN state.
Description	The purpose of the API is to switch the SMU peripheral from the FAULT state to the RUN state. This also switches the error pin from the FAULT state to FAULT-FREE state. Additionally, this API can be used to change the FSP state from the power-on state to the Fault-free state. This is essential to setup the error pin to drive the FSP. It is also required for testing of FSP pin. The transitions of states require certain clock cycles to reflect. The API returns before this transition is observed. The API can be called from any core. However, it is recommended to call the API from only one core at any instance of time to ensure consistent behavior.	
Source	IFX	
Error handling	SMU_E_RELEASE_FSP_FAILURE, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE	

(table continues...)

Table 89 (continued) Specification for `Smu_ReleaseFSP` API

Configuration dependencies	-
User hints	None
SFR accessed	SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AGC(r), SMU_CMD(w), SMU_DBG(r), SMU_STS(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.11 Smu_ActivateFSP

Table 90 Specification for `Smu_ActivateFSP` API

Syntax	`Std_ReturnType  Smu_ActivateFSP`
	`(`
	`  void`
	`)`	

Service ID	0xB3	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is activation of FSP is successful E_NOT_OK: Operation not successful.
Description	The purpose of the API is to activate the FSP to indicate a FAULT state on the error pin to the safe state switching device. When FSP is activated the SMU reaches the fault state. This can be confirmed by reading the SMU state in hardware. Also, In the SMU START state, activation of FSP is only possible using this API as alarms are not processed. Additionally, this is required for the testing of the FSP timing. The transitions of states require certain clock cycles to reflect. The API returns before this transition is observed.	
Source	IFX	
Error handling	SMU_E_ACTIVATE_FSP_FAILURE, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE	

(table continues...)

Table 90 **(continued) Specification for `Smu_ActivateFSP` API**

Configuration dependencies	-	
User hints	None	
SFR accessed	SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_CMD(w), SMU_STS(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.12 Smu_SetupErrorPin

Table 91 **Specification for `Smu_SetupErrorPin` API**

Syntax	`Std_ReturnType  Smu_SetupErrorPin`
	`(`
	`  void`
	`)`	

Service ID	0xB4	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is switching of the error pin from GPIO mode to SMU mode is successful. E_NOT_OK: Operation not successful or the error pin is already set.
Description	The purpose of the API is to enable the SMU to control the error pin. This API switches the error pin from GPIO mode to SMU mode. Only after switching to the SMU mode, SMU can control the error pin. *Note: The transitions of states require certain clock cycles to reflect. The API returns before this transition is observed.* *Note: The API uses SMU temporary locking mechanism after configuration of SFRs to prevent accidental change in values of the SFRs.*	
Source	IFX	
Error handling	SMU_E_LOCKED, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE	

(table continues...)

Table 91 (continued) Specification for `Smu_SetupErrorPin` API

Configuration dependencies	-
User hints	None
SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), PMS_CMD_STDBY(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_KEYS(w), SMU_PCTL(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.13 Smu_ReleaseErrorPin

Table 92 Specification for `Smu_ReleaseErrorPin` API

Syntax	`Std_ReturnType  Smu_ReleaseErrorPin`
	`(`
	`  void`
	`)`	

Service ID	0xB5	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is control of error pin is successfully released from SMU E_NOT_OK: Operation not successful.
Description	The purpose of the API is to release the control of the error pin. *Note: The transitions of modes require certain clock cycles to reflect. The API returns before this transition is observed.* *Note: The API uses SMU temporary locking mechanism after configuration of SFRs to prevent accidental change in values of the SFRs.*	
Source	IFX	
Error handling	SMU_E_LOCKED, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE	

(table continues...)

Table 92 (continued) Specification for `Smu_ReleaseErrorPin` API

Configuration dependencies	-
User hints	None
SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), PMS_CMD_STDBY(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_KEYS(w), SMU_PCTL(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.14 Smu_RTStop

Table 93 Specification for `Smu_RTStop` API

Syntax	`Std_ReturnType  Smu_RTStop ( const uint8 TimerNum )`	
Service ID	0xB6	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	TimerNum	Recovery Timer unit to be stopped (0,1)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is requested recovery timer is stopped successfully E_NOT_OK: Operation not successful that is requested recovery timer could not be stopped successfully, for example due to invalid parameters
Description	The purpose of the API is to stop the requested recovery timer unit. Possible use case: when a fault occurs, error handler might be triggered. However, this error handler should setup a recovery mechanism or error mitigation mechanism within a finite interval of time to prevent the system from failing.	
Source	IFX	

(table continues...)

Table 93 (continued) Specification for `Smu_RTStop` API

Error handling	SMU_E_RT_STOP_FAILURE, SMU_E_INVALID_TIMER, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE
Configuration dependencies	-
User hints	None
SFR accessed	SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_CMD(w), SMU_STS(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.15 Smu_GetRTMissedEvent

Table 94 Specification for `Smu_GetRTMissedEvent` API

Syntax	`Std_ReturnType Smu_GetRTMissedEvent`
	`(`
	`    const uint8 TimerNum,`
	`    boolean * const EventMissed`
	`)`	

Service ID	0xB7	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	TimerNum	Recovery Timer unit for which the status has to be procured (0,1).
Parameters (out)	EventMissed	EventMissed : TRUE: Event has been missed FALSE: Event has NOT been missed
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is check for missed events successful. E_NOT_OK: Operation not successful that is check for missed events not successful, for example due to invalid parameters
Description	The purpose of the API is to know if any alarms requiring the requested recovery timer were set while the recovery timer was running.	
Source	IFX	

(table continues...)

Table 94 (continued) Specification for `Smu_GetRTMissedEvent` API

Error handling	SMU_E_INVALID_TIMER, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE, SMU_E_PARAM_POINTER
Configuration dependencies	-
User hints	None
SFR accessed	SMU_STS(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.16 Smu_ActivatePES

Table 95 Specification for `Smu_ActivatePES` API

Syntax	`Std_ReturnType  Smu_ActivatePES ( void )`	
Service ID	0xB8	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is activation of the Port Emergency Stop (PES) is successful. E_NOT_OK: Operation not successful.
Description	The purpose of this API is to trigger the activation of the Port Emergency Stop (PES). The PES is also directly controlled by the SMU_core when entering the FAULT state.	
Source	IFX	
Error handling	SMU_E_ACTIVATE_PES_FAILURE, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE	
Configuration dependencies	-	
User hints	None	

(table continues...)

Table 95 **(continued) Specification for `Smu_ActivatePES` API**

SFR accessed	SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_CMD(w), SMU_STS(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.17 Smu_RegisterMonitor

Table 96 **Specification for `Smu_RegisterMonitor` API**

Syntax	`Std_ReturnType  Smu_RegisterMonitor`
	`(`
	`    const uint16 * const RegMonPtr,`
	`    Smu_SffTestResType * const RegMonResult`
	`)`	

Service ID	0xB9	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	RegMonPtr	Pointer to the array, which holds the modules, which have SFF test enabled. The total number of elements is the total number of modules, which can undergo SFF tests. The elements have to be as per the bits specified in RMCTL register. Additionally, the module needs to be present in the specific device.
Parameters (out)	RegMonResult	Pointer to array for the SFF test results for the modules. In case a module test was not enabled but has an error recorded, it will indicate that failure as well.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is the SFF test execution was completed successfully, irrespective of the SFF test result E_NOT_OK: Operation not successful that is the SFF test execution was not completed successfully, irrespective of the SFF test result

(table continues...)

Table 96 (continued) Specification for `Smu_RegisterMonitor` API

Description	The purpose of the API is to provide the initialization, execution and termination of the safety flip-flop tests to be executed for different modules as enabled in the RegMonPtr parameter. The user shall take care of the prerequisites for safety flip-flop test as mentioned in the HW UM before invoking the API. The API returns whether the safety flip-flop test execution has been successfully completed or not, irrespective of the safety flip-flop test results. The result of the safety flip-flop tests can be obtained through the RegMonResult parameter. The RegMonResult parameter needs to be checked only when the API returns E_OK, which implies that the safety flip-flop test execution has been successfully completed. *Note: The API uses SMU temporary locking mechanism after configuration of SFRs to prevent accidental change in values of the SFRs.*
Source	IFX
Error handling	SMU_E_SFF_TEST_FAILURE, SMU_E_INVALID_DRIVER_STATE, SMU_E_UNINIT, SMU_E_LOCKED, SMU_E_PARAM_POINTER
Configuration dependencies	-
User hints	None
SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_KEYS(w), SMU_RMCTL(w), SMU_RMEF(rw), SMU_RMSTS(rw), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.18 Smu_GetSmuState

Table 97 Specification for `Smu_GetSmuState` API

Syntax	`Smu_CoreStateType  Smu_GetSmuState`
	`(`
	`  void`
	`)`	

Service ID	0xBA	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-

(table continues...)

Table 97 (continued) Specification for `Smu_GetSmuState` API

Parameters (in - out)	-	-
Return	Smu_CoreStateType	State of SMU core state machine
Description	The purpose of the API is to provide the current state of the SMU core. This is referred to as the safety status of the system as all critical faults will cause SMU to go to the FAIL state.	
Source	IFX	
Error handling	SMU_E_UNINIT	
Configuration dependencies	-	
User hints	None	
SFR accessed	SMU_DBG(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.19 Smu_ActivateRunState

Table 98 Specification for `Smu_ActivateRunState` API

Syntax	`Std_ReturnType Smu_ActivateRunState ( const uint32 Cmd )`	
Service ID	0xBB	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	Cmd	Command to switch the SMU to the RUN state
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is activation of fault free RUN state is successful. E_NOT_OK: Operation not successful that is activation of fault free RUN state is not successful, for example SMU is not initially in START state.

(table continues...)

Table 98 (continued) Specification for `Smu_ActivateRunState` API

Description	The purpose of the API is to allow switching the SMU peripheral into the RUN fault-free state as requested by the caller. The SMU validates the request based on integrity checks of SMU (that is check of the command value). The API can be called from any core. However, it is recommended to call the API from only one core at any instance of time to ensure consistent behavior.
Source	IFX
Error handling	SMU_E_ACTIVATE_RUN_STATE_FAILURE, SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE
Configuration dependencies	-
User hints	-
SFR accessed	SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_CMD(w), SMU_DBG(r), SMU_STS(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.20 Smu_GetVersionInfo

Table 99 Specification for `Smu_GetVersionInfo` API

Syntax	`void Smu_GetVersionInfo ( Std_VersionInfoType * const VersionInfoPtr )`	
Service ID	0xBC	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to store information about the module
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to return the version information of the SMU driver. The version information includes Module ID, Vendor ID and vendor specific version numbers. This function is available only if the SMU_VERSION_INFO_API is set ON.	
Source	IFX	

(table continues...)

Table 99 (continued) Specification for `Smu_GetVersionInfo` API

Error handling	SMU_E_PARAM_POINTER
Configuration dependencies	SmuVersionInfoApi
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.21 Smu_CoreAliveTest

Table 100 Specification for `Smu_CoreAliveTest` API

Syntax	`Std_ReturnType  Smu_CoreAliveTest ( void )`	
Service ID	0xBD	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: The SMU_AliveTest start command is executed successfully E_NOT_OK: The SMU_AliveTest start command is not executed successfully
Description	The purpose of the API is to provide the means to execute the SMU_AliveTest command that checks the smu_core_alive signal. The API returns whether the SMU_AliveTest command to start the test has been successfully executed or not. It does not return the result of the smu_core_alive test. The result of the smu_core_alive test can be obtained by reading the status flag for the SMU_core alive alarm (alarm 16 of alarm group 21) by means of the Smu_GetAlarmStatus API. The Smu_CoreAliveTest API also executes the SMU_AliveTest command to stop the test, which provides the user with flexibility to call the API cyclically during runtime. The SMU_stdby has to remain enabled and the SMU_core has to be in the START state to execute this command.	
Source	IFX	
Error handling	SMU_E_CORE_ALIVE_FAILURE, SMU_E_INVALID_DRIVER_STATE, SMU_E_UNINIT, SMU_E_STDBY_DISABLED	

(table continues...)

Table 100 (continued) Specification for `Smu_CoreAliveTest` API

Configuration dependencies	-
User hints	The user shall read the status flags for SMU_core alive alarm (alarm 16 of alarm group 21) using the Smu_GetAlarmStatus() API to check the result of the smu_core_alive test after execution of the Smu_CoreAliveTest() API. The user shall clear the status of SMU_core alive alarm (alarm 16 of alarm group 21) using the Smu_ClearAlarmStatus() API, after checking the result of smu_core_alive test so that further alarm detection is possible.
SFR accessed	SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_CMD(w), SMU_DBG(r), SMU_STS(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.22 Smu_InitCheck

Table 101 Specification for `Smu_InitCheck` API

Syntax	`Std_ReturnType Smu_InitCheck`
	`(`
	`    const Smu_ConfigType * const ConfigPtr`
	`)`	

Service ID	0xA9	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different CPU cores	
Parameters (in)	ConfigPtr	Pointer to the SMU configuration for initialization
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK - if initialization comparison is success. E_NOT_OK - In Case of - Driver is not initialized - Input config Pointer is Null - Global Variables or SFR is not set as expected.
Description	The purpose of the API is to check the initialization values after SMU is initialized. The API should be called after the SMU driver is initialized to check the initialization values.	
Source	IFX	

(table continues...)

Table 101 (continued) Specification for `Smu_InitCheck` API

Error handling	-
Configuration dependencies	SmuInitCheckApi
User hints	None
SFR accessed	PMS_AGFSP_STDBY(r), PMS_CMD_STDBY(r), SMU_AGC(r), SMU_AGCF(r), SMU_AGFSP(r), SMU_CMD(r), SMU_FSP(r), SMU_RTAC00(r), SMU_RTAC01(r), SMU_RTAC10(r), SMU_RTAC11(r), SMU_RTC(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.23 Smu_GetAlarmExecutionStatus

Table 102 Specification for `Smu_GetAlarmExecutionStatus` API

Syntax	`Std_ReturnType  Smu_GetAlarmExecutionStatus`
	`(`
	`    const uint32 AlarmExecStatusReq,`
	`    uint32 * const AlarmExecStatus`
	`)`	

Service ID	0xBE	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmExecStatusReq	Requested alarm reaction execution status bit or alarm event missed bit
Parameters (out)	AlarmExecStatus	Pointer that stores the alarm execution status or the alarm event missed result
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is retrieval of requested alarm reaction execution status or the alarm event missed result is successful E_NOT_OK: Operation not successful that is retrieval of requested alarm reaction execution status or the alarm event missed result is not successful, for example due to invalid parameters
Description	The purpose of the API is to retrieve the requested alarm reaction execution status or the alarm event missed result.	
Source	IFX	

(table continues...)

Table 102 (continued) Specification for `Smu_GetAlarmExecutionStatus` API

Error handling	SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE, SMU_E_INVALID_EXECUTION_STATUS, SMU_E_PARAM_POINTER
Configuration dependencies	-
User hints	None
SFR accessed	SMU_AEX(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.24 Smu_ClearAlarmExecutionStatus

Table 103 Specification for `Smu_ClearAlarmExecutionStatus` API

Syntax	`Std_ReturnType  Smu_ClearAlarmExecutionStatus`
	`(`
	`    const uint32 AlarmExecStatusReq`
	`)`	

Service ID	0xBF	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	AlarmExecStatusReq	Alarm reaction execution status bit or alarm event missed bit to be cleared
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Operation successful that is requested alarm reaction execution status bit or the alarm event missed bit is cleared successfully E_NOT_OK: Operation not successful that is requested alarm reaction execution status bit or the alarm event missed bit is not cleared successfully, for example due to invalid parameters
Description	The purpose of the API is to clear the requested alarm reaction execution status bit or the alarm event missed bit.	
Source	IFX	
Error handling	SMU_E_UNINIT, SMU_E_INVALID_DRIVER_STATE, SMU_E_INVALID_EXECUTION_STATUS	

(table continues...)

Table 103 (continued) Specification for `Smu_ClearAlarmExecutionStatus` API

Configuration dependencies	-
User hints	If the API parameter corresponds to execution status bit of an alarm reaction, the API additionally clears the corresponding alarm event missed bit. If alarm event missed bit is requested, only that bit status will be cleared. To determine if alarm event missed bit is set, the user shall invoke the Smu_GetAlarmExecutionStatus API.
SFR accessed	CPU_COMPAT(w), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SMU_AEXCLR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.4 Notifications and Callbacks

The SMU driver does not provide any notifications or callbacks.

1.3.5 Scheduled functions

The SMU driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

The SMU driver does not provide any interrupt handlers.

1.3.7 Callout

The driver does not support any callout functions.

1.3.8 Errors Handling

This section describes the various errors reported by the SMU driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
SMU_E_UNINIT: SMU_E_UNINIT DET is reported when any API is called while the driver is not in initialized state.	IFX	0x01	DET_SAFETY	0x01	DET_SAFETY
SMU_E_ALREADY_INITIALIZED: SMU_E_ALREADY_INITIALIZED DET is reported when SMU is already initialized.	IFX	0x02	DET_SAFETY	0x02	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
SMU_E_INIT_FAILED: SMU_E_INIT_FAILED DET is reported when initialization of SMU driver fails due to incorrect configuration parameter.	IFX	0x03	DET_SAFETY	0x03	DET_SAFETY
SMU_E_PARAM_POINTER: SMU_E_PARAM_POINTER DET is reported when the pointer passed as a parameter to an API is a NULL pointer.	IFX	0x04	DET_SAFETY	0x04	DET_SAFETY
SMU_E_PARAM_GROUP: SMU_E_PARAM_GROUP DET is reported when the group ID or the alarm position passed as a parameter to an API is not valid.	IFX	0x05	DET_SAFETY	0x05	DET_SAFETY
SMU_E_INVALID_DRIVER_STATE: SMU_E_INVALID_DRIVER_STATE DET is reported when the SMU driver state is SMU_FAILED.	IFX	0x06	DET_SAFETY	0x06	DET_SAFETY
SMU_E_INVALID_TIMER: SMU_E_INVALID_TIMER DET is reported when the timer value passed as a parameter to an API is not valid.	IFX	0x07	DET_SAFETY	0x07	DET_SAFETY
SMU_E_STDBY_DISABLED: SMU_E_STDBY_DISABLED DET is reported when any alarm action is configured or performed with respect to the standby domain of SMU without enabling the SMU standby mode.	IFX	0x08	DET_SAFETY	0x08	DET_SAFETY
SMU_E_LOCKED: SMU_E_LOCKED DET is reported when the SMU is already in locked state.	IFX	0x09	DET_SAFETY	0x09	DET_SAFETY
SMU_E_INVALID_ALARM_ACTION: SMU_E_INVALID_ALARM_ACTION DET is reported when the alarm action to be configured is not valid.	IFX	0x0A	DET_SAFETY	0x0A	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
SMU_E_INVALID_EXECUTION_STATUS: SMU_E_INVALID_EXECUTION_STATUS DET is reported when an invalid error mechanism is requested for alarm execution status.	IFX	0x0B	DET_SAFETY	0x0B	DET_SAFETY
SMU_E_CORE_MISMATCH: SMU_E_CORE_MISMATCH DET is reported when the Init or De-Init is called from any core other than the master core.	IFX	0x68	DET_SAFETY	0x68	DET_SAFETY
SMU_E_SF_CFG_LOCKED: The safety error is reported when the configuration registers do not get locked after applying permanent lock.	IFX	0xC8	SAFETY	0xC8	SAFETY
SMU_E_ACTIVATE_RUN_STATE_FAILURE: SMU_E_ACTIVATE_RUN_STATE_FAILURE Production Error is reported when the activation of RUN state fails.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
SMU_E_SET_ALARM_STATUS_FAILURE: SMU_E_SET_ALARM_STATUS_FAILURE Production Error is reported when the setting of alarm status fails.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
SMU_E_RELEASE_FSP_FAILURE: SMU_E_RELEASE_FSP_FAILURE Production Error is reported when the FSP cannot be released.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
SMU_E_SFF_TEST_FAILURE: SMU_E_SFF_TEST_FAILURE Production Error is reported when timeout occurs before SFF test status is reflected.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
SMU_E_RT_STOP_FAILURE: SMU_E_RT_STOP_FAILURE Production Error is reported when the Recovery timer cannot be stopped.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
SMU_E_CLEAR_ALARM_STATUS_FAILURE: SMU_E_CLEAR_ALARM_STATUS_FAILURE Production Error is reported when the clearing of alarm status fails.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
SMU_E_ACTIVATE_FSP_FAILURE: SMU_E_ACTIVATE_FSP_FAILURE Production Error is reported when activation of FSP fails.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
SMU_E_CORE_ALIVE_FAILURE: SMU_E_CORE_ALIVE_FAILURE Production Error is reported when the the SMU_AliveTest start command fails.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
SMU_E_ACTIVATE_PES_FAILURE: SMU_E_ACTIVATE_PES_FAILURE Production Error is reported when activation of the PES feature fails.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error

1.3.9 Deviations and limitations

This section describes the deviations and limitations of the SMU driver.

1.3.9.1 Deviations

This section describes the deviations of the SMU driver.

1.3.9.1.1 Software specification deviations

The SMU driver does not have any deviations.

1.3.9.1.2 AMDC Violations

The SMU driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

The SMU driver does not have any VSMD violations.

1.3.9.2 Limitations

This section describes the limitations of the SMU driver.

Table 104 Known limitations

Reference	Limitation
Usability of Port Pin P33.8 for FSP and Emergency Stop Input features	Due to hardware errata SMU_TC.H016, If user enables the emergency stop feature then, Port pin P33.8 (FSP[0]) shall not be configured for FSP, instead Port pin P33.10(FSP[1]) shall be used for the same.

Revision history

Table 105 **Revision history**

Date	Version	Description
2023-07-03	6.0	Document is released.
2023-06-13	5.1	- 1.1.2 Hardware-software mapping section, Figure 1 Mapping of hardware-software interfaces updated to add Mcal_Wrapper and remove Dem - In 1.1.3.1 C file structure, Figure 2 Smu_C_File_Structure-1.png updated by inclusion Mcal_Wrapper.h and removing Dem.h - 1.1.3.1 C File Structure section, Table 2 C File Structure updated to add Mcal_Wrapper.h and remove Dem.h - DEM module removed and Mcal_Wrapper module added in 1.1.4.1 Integration with AUTOSAR stack section - ASIL Level has been updated to Safety level and the description of the safety level is updated in Section 1.3.3 - DEM changed to Production Error In Section 1.3.8 - ConfigPtr passed to InitCheck AOU added in section 1.2 - Updated the description of return type E_NOT_OK for API Smu_InitCheck in section 1.3.3.22
2022-07-29	5.0	Document is released.
2022-07-01	4.1	Limitation section is updated for FSP and Emergency stop input feature enabled on port pin P33.8.
2021-11-12	4.0	Document is released.
2021-11-11	3.1	Config variant attribute table information is removed and added this information in 'Configuration interfaces' section.
2021-02-15	3.0	Document is released.
2021-02-11	2.1	- Example usage for Configuration register locking and Alarm execution status updated. - Initialization Check AoU modified. - Note regarding usage of temporary lock added in description of `Smu_Init()`, `Smu_DeInit()`, `Smu_SetAlarmAction()`, `Smu_SetupErrorPin()`, `Smu_ReleaseErrorPin()` and `Smu_RegisterMonitor()` APIs. - API and parameter description updated for `Smu_GetAlarmExecutionStatus()` and `Smu_ClearAlarmExecutionStatus()` APIs. - User hint added for the `Smu_ClearAlarmExecutionStatus()` API.

(table continues...)

Table 105 (continued) Revision history

2020-11-10	2.0	Document is released.
2020-11-09	1.1	SFR access information for APIs updated.
2020-08-13	1.0	Document is released.
2020-08-07	0.1	- Initial draft. - The SMU driver chapter moved from MC-ISAR_TC3xx_UM_CD to this document.

Edition 2023-07-03

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-ocr1484806431059