MCAL User Manual for Stm

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 Stm 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

Table of contents

1 STM driver

1.1 User information

1.1.1 Description

The STM provides free running 64 bit timer for real-time operating systems and other system purposes (high precision and long range). The STM complex driver is responsible for enabling STM interrupts and providing alarms at regular intervals.

1.1.2 Hardware-software mapping

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

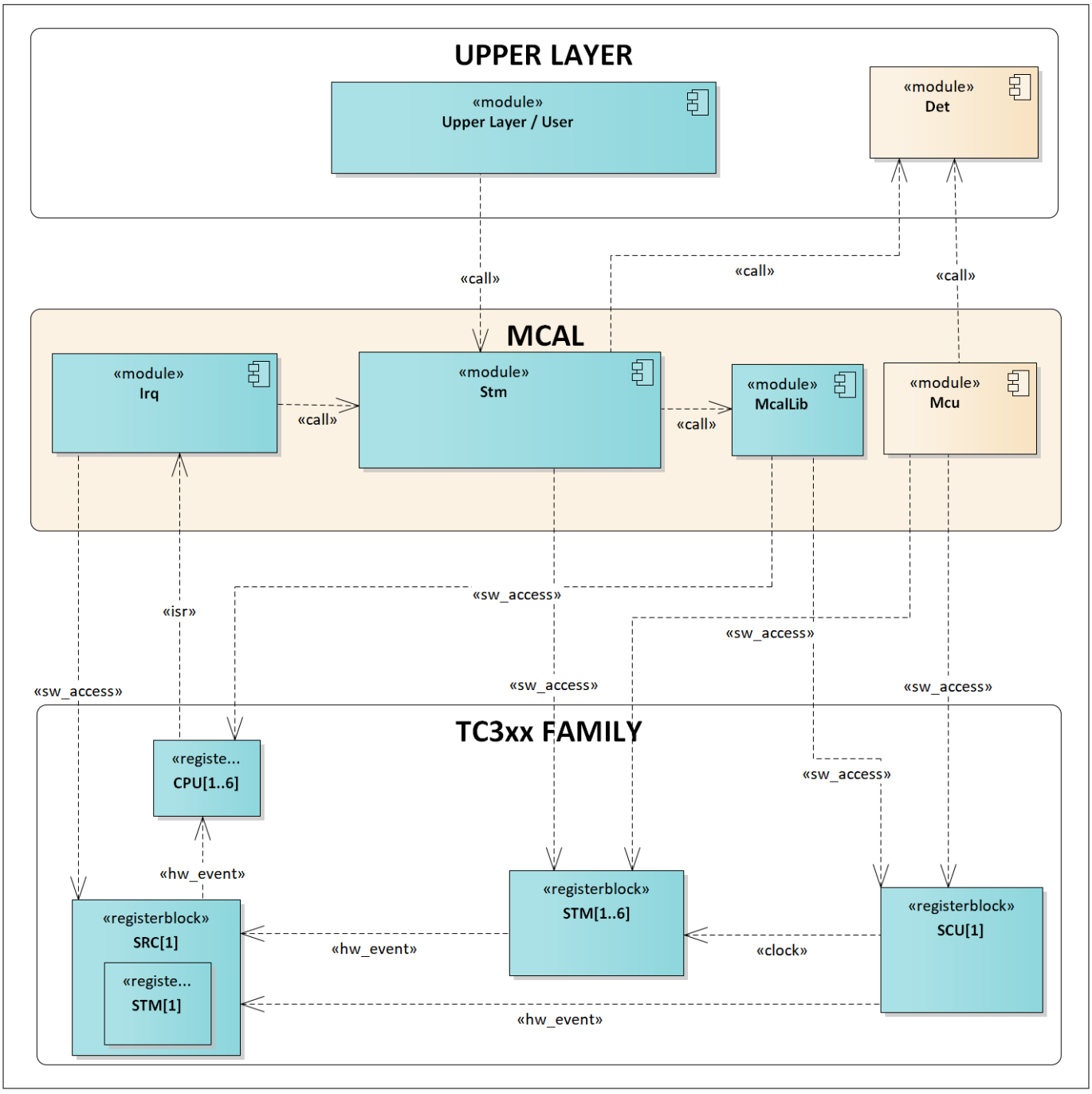

Figure 1 Mapping of hardware-software interfaces

1.1.2.1 STM: primary hardware peripheral

Hardware functional features

- The STM provides free running 64 bit timer for real-time operating systems and other system purposes (high precision and long range)
- STM complex driver is responsible for enabling STM interrupts and providing alarms at regular intervals
- Counting starts automatically after an application reset

Users of the hardware

STM(0..x)(x:number of cores available in the device) is exclusively by STM driver. User can directly access STM registers to get the call back notification.

Hardware diagnostic features

Not applicable.

Hardware events

The content of the 64-bit STM can be compared against the content of compare values stored in the CMP0 or CMP1 register. Service requests can be generated on a compare match of the STM with CMP0 or CMP1 registers.

1.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The STM driver depends upon the SCU for clock and reset functionality.

Users of the hardware

Several peripherals have their clocks supplied by SCU. However it is only the MCU driver that is responsible for configuration of the clock tree.

Hardware diagnostic features

Not applicable.

Hardware events

Not applicable.

1.1.2.3 SRC-IR: dependent hardware peripheral

Hardware functional features

CMP0 and CMP1 register has its compare register flag that is set by hardware on a compare match event.

Users of the hardware

Several peripherals have their interrupt nodes mapped to Interrupt router. STM driver uses two interrupt nodes to trigger STM timer compare match event.

Hardware diagnostic features

Safety mechanism implemented in the Interrupt router module to trigger SMU alarm incase of hardware failure.

Hardware events

Interrupt event generation based on the configured priority and core.

1.1.3 File structure

1.1.3.1 C File Structure

This section provides the details of the C files of the STM driver.

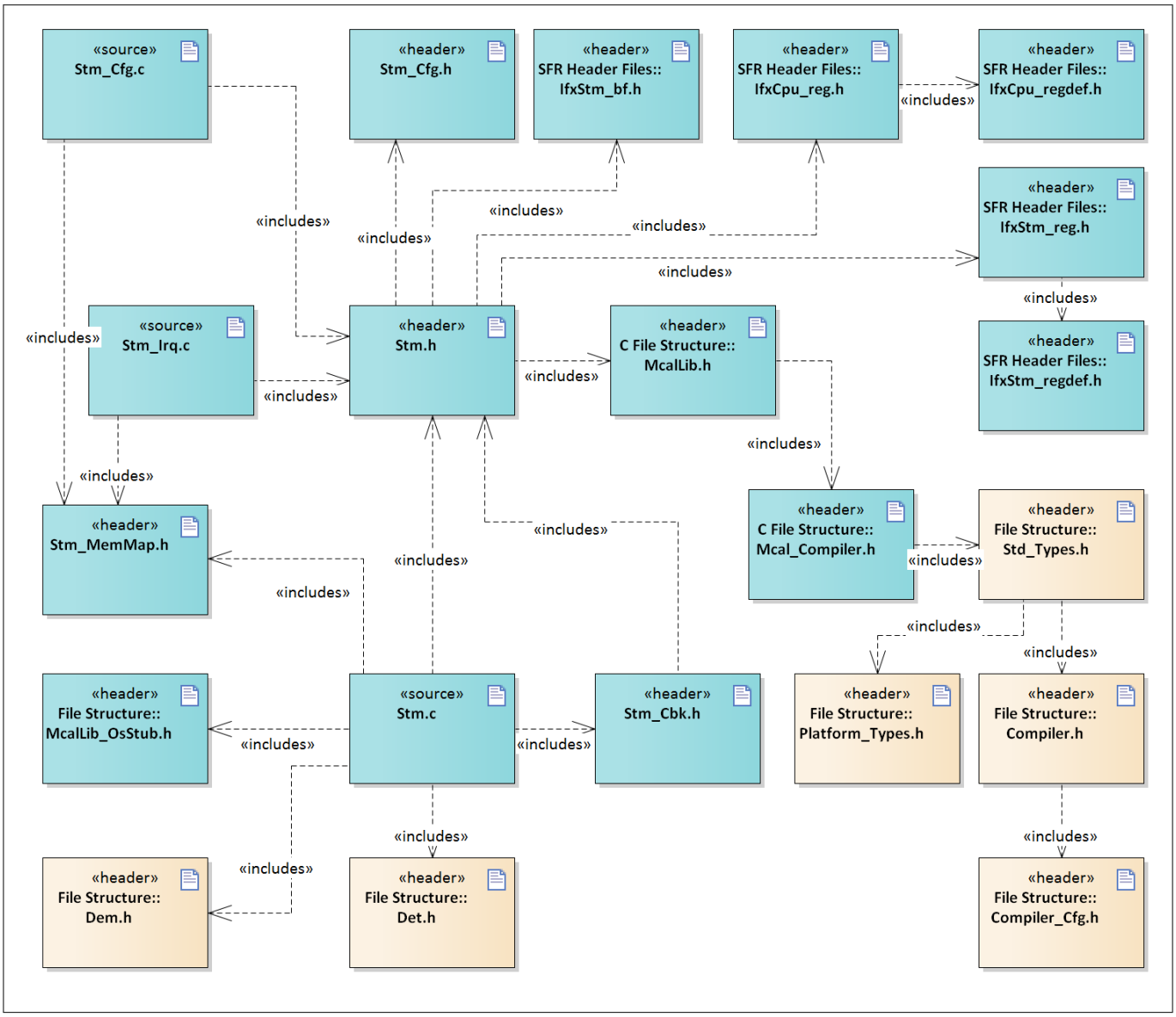

Figure 2 **Stm_C_File_Structure-1.png**

Table 2 **C File Structure**

File name	Description
`Compiler.h`	Provides abstraction from compiler-specific keywords
`Compiler_Cfg.h`	Configuration header file for compiler abstraction
`Dem.h`	Provides the exported interfaces of Diagnostic Event Manager
`Det.h`	Provides the exported interfaces of Development Error Tracer
`IfxCpu_reg.h`	SFR header file for CPU
`IfxCpu_regdef.h`	SFR header file for CPU
`IfxStm_bf.h`	SFR header file for STM
`IfxStm_reg.h`	SFR header file for STM
`IfxStm_regdef.h`	SFR header file for STM

Table 2 C File Structure (continued)

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_Compiler.h	Provides abstraction for TriCore-intrinsic instruction
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.
Stm.c	This file contains functionality of STM driver.
Stm.h	This file contains function declaration and interrupt routine declaration of STM driver.
Stm_Cbk.h	Stm Driver Callback header definition file
Stm_Cfg.c	STM Module configuration file to store STM timer usage information across the cores.
Stm_Cfg.h	STM Module Configuration header file. it contains macro definition for all the pre compile configuration parameters.
Stm_Irq.c	This file contains the interrupt frames for the STM Module.
Stm_MemMap.h	File (Static) containing the memory section definitions used by the STM driver.

1.1.3.2 Code Generator Plugin Files

This section provides the details of the code generator plugin files of the STM driver.

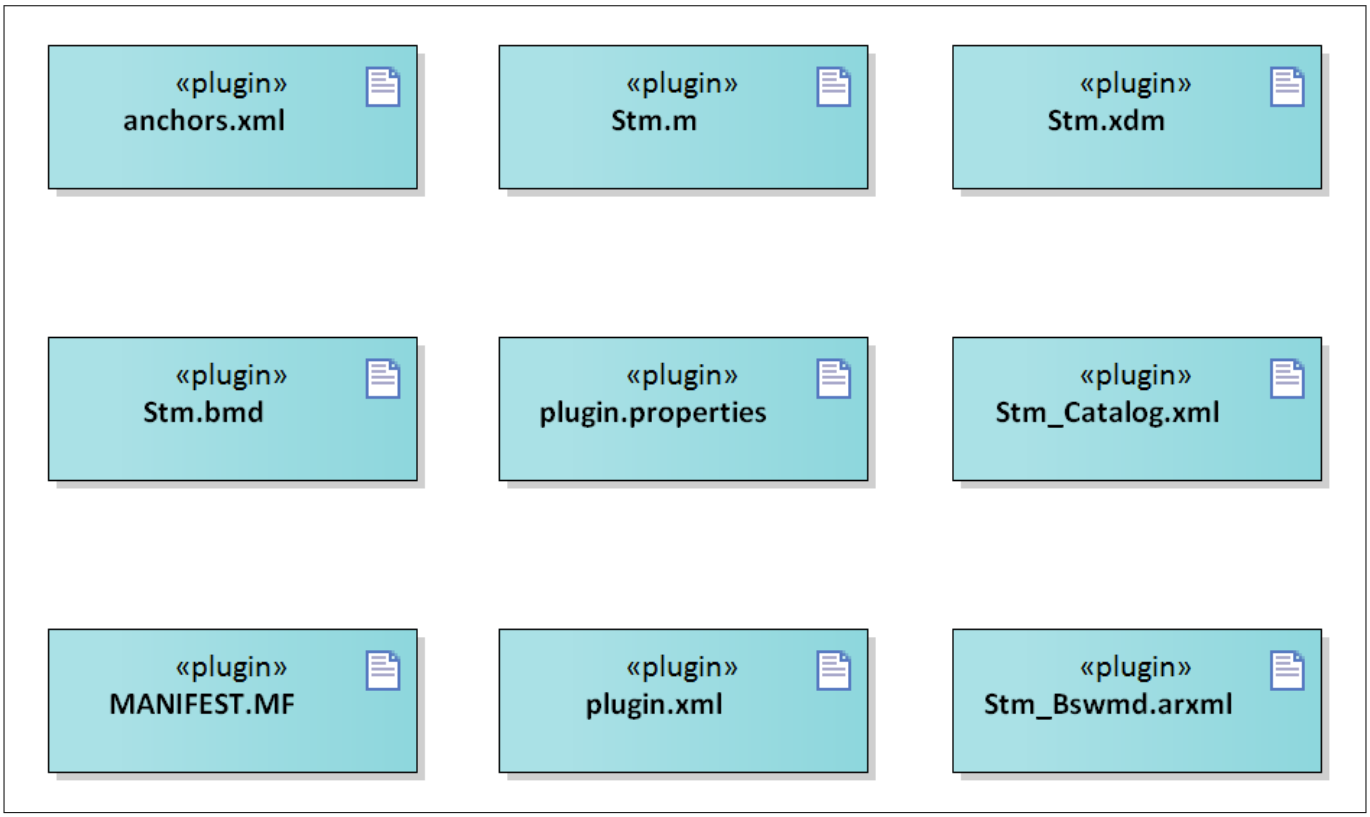

Figure 3 Code Generator Plugin Files

Table 3 Stm_Code_Generator_Plugin_Files-1.png

File name	Description
`MANIFEST.MF`	Tresos plugin support file containing the meta-data for Stm driver.
`Stm.bmd`	AUTOSAR format XML data model schema file (for each device).
`Stm.m`	Code template macro file for Stm driver
`Stm.xdm`	Tresos format XML data model schema file.
`Stm_Bswmd.arxml`	AUTOSAR format module description file.
`Stm_Catalog.xml`	AUTOSAR format catalog file.
`anchors.xml`	Tresos anchors support file for the Stm driver
`plugin.properties`	Tresos plugin support file for the Stm driver.
`plugin.xml`	Tresos plugin support file for the Stm driver.

1.1.4 Integration hints

This section lists the key points that an integrator or the user of the STM 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 STM driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

  Memory mapping is a concept from AUTOSAR that allows re-location of text, variables, constants and configuration data to user specific memory regions. In order to achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Stm_MemMap.h`.

  The file `Stm_MemMap.h` 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 re-located to the correct memory region. A sample implementation listing the memory-section macros is depicted below.

```
/**** CORE[x] GLOBAL RAM DATA -- NON-CACHED LMU ****/
#if defined STM_START_SEC_VAR_CLEARED_QM_CORE[x]_UNSPECIFIED
/***** User pragmas here for Non-cached LMU*****/
#undef STM_START_SEC_VAR_CLEARED_QM_CORE[x]_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined STM_STOP_SEC_VAR_CLEARED_QM_CORE[x]_UNSPECIFIED
/***** User pragmas here for Non-cached LMU*****/
#undef STM_STOP_SEC_VAR_CLEARED_QM_CORE[x]_UNSPECIFIED
#undef MEMMAP_ERROR
/**** CODE -- PF[x] ****/
#elif defined STM_START_SEC_CODE_QM_GLOBAL
/*****User pragmas here for PF[x]*****/
#undef STM_START_SEC_CODE_QM_GLOBAL
#undef MEMMAP_ERROR
#elif defined STM_STOP_SEC_CODE_QM_GLOBAL
/*****User pragmas here for PF[x]*****/
#undef STM_STOP_SEC_CODE_QM_GLOBAL
#undef MEMMAP_ERROR

/**** CORE CONFIG DATA -- PF****/
#elif defined STM_START_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
/*****User pragmas here for PF[x]*****/
#undef STM_START_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR
#elif defined STM_STOP_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
/*****User pragmas here for PF[x]*****/
#undef STM_STOP_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
#undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Stm_MemMap.h, wrong pragma command"
#endif
```

- **DET**

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

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

- **DEM**

  The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The STM driver reports all the production errors through the interfaces provided by the DEM module. The user of the STM driver shall process all the production errors (fail/pass) reported to the DEM module. The interface used for reporting in AUTOSAR version 4.2.2 is `Dem_ReportErrorStatus()` and for AUTOSAR version 4.4.0 is `Dem_SetEventStatus()`. The `Dem.h` and `Dem.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DEM module during the integration phase.
- **SchM**

  STM module does not have any critical section
- **Safety error**

  STM module does not report any safety error.
- **Notifications and callbacks**

  The STM driver gives callback to user when the compare match is detected based on the ticks configured by user.
- **Operating system (OS)**

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

1.1.4.2 Multicore and Resource Manager

The STM driver supports execution of its APIs in parallel from all the CPU cores. The user has to allocate STM timers to CPU cores at pre-compile time using the Resource Manager and MCU module. The following are the key points to be considered with respect to multicore in the driver:

- For each core one STM timer is allowed to configure in the resource manager module.

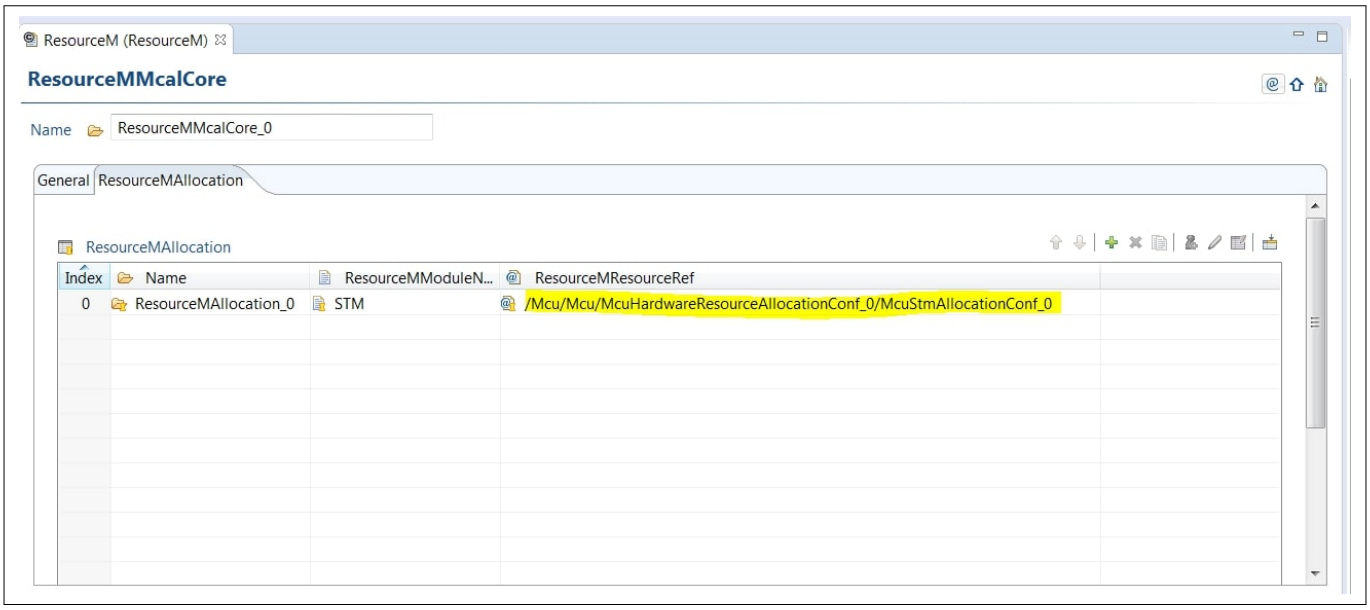

Figure 4 STM timer allocation in Resource Manager

- The STM timer can be allocated to any of the core available on the silicon. The user must ensure that only the allocated STM peripheral resource is accessed in the core.
- It must be ensured that the STM timer number passed as a parameter while invoking a STM API, belong to the same core on which the API is invoked.
- DETs will be raised in case APIs are invoked with mismatch of core and STM timer.
- The 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) and CORE[x] (specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

  Code Section:

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

  Data Section:

  The RAM variable memory sections marked as specific to core, should be re-located to the DSPR of the same core. Those marked as global should be relocated to the non-cached LMU region.

  Configuration data and constants:

  The configuration data section sections marked as specific to core, should be re-located to the PFLASH of the same core. Those marked as global should be relocated to the PFlash of the master core.

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

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

1.1.4.3 MCU support

The STM driver is dependent on MCU driver for clock configuration. The initialization of STM driver must be started only after completion of MCU initialization. The following must be considered while configuring the MCU driver in tresos:

- The CMP register used by STM driver must be reserved in the MCU configuration for exclusive use by STM.
- The CMP register is shared between STM and WDG driver, user must ensure CMP is allocated to STM before using it in STM API's.

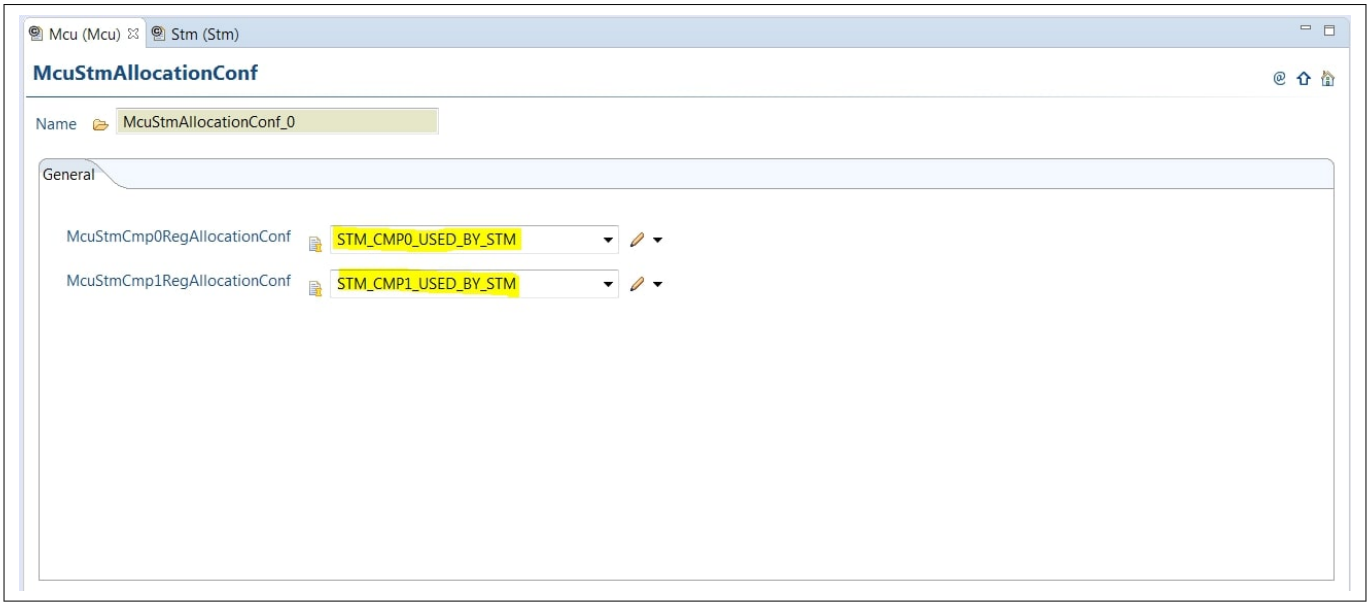

Figure 5 STM compare register allocation for STM in MCU

1.1.4.4 Port support

The STM driver does not use any services provided by the PORT driver.

1.1.4.5 DMA support

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

1.1.4.6 Interrupt connections

The interrupt connections of the STM driver are described in this section.

Compare Match Interrupts

The compare match interrupt is generated based on the ticks configured by the user. The interrupts for CMP registers are enabled based on the allocation of CMP registers for STM. The interrupts from CMP0 is routed to SR0 and CMP1 is routed to SR1.

```
/* include MCU timer header file */
#include "Mcu_17_TimerIp.h"
/*********SRC_ STM0SR0********/
ISR(STM0SR0_ISR)
{
/* Enable Global Interrupts */
ENABLE();
/* Call Interrupt function */
Mcu_17_Stm_CompareMatchIsr(STM_NUMBER, STM_INTERRUPT_NODE);
/* STM_NUMBER = 0, STM_INTERRUPT_NODE = 0 */
}
/********* SRC_ STM0SR1********/
ISR(STM0SR1_ISR)
{
/* Enable Global Interrupts */
ENABLE();
/* Call Interrupt function */
 Mcu_17_Stm_CompareMatchIsr(STM_NUMBER, STM_INTERRUPT_NODE);
}
```

1.1.4.7 Example usage

Enable STM interrupts

The code Listing depicts the steps involved to enable the STM driver interrupts.

```
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0);
while(Mcu_GetPllStatus() == MCU_PLL_LOCKED);
Mcu_DistributePllClock();
/* Configure Interrupt priority */
IrqStm_Init();
/*Enable Stm interrupt*/
Stm_EnableModule(StmModuleNumber);
```

After the call of `Stm_EnableModule()` it enables the STM interrupts and interrupts node mapping.

Enable Alarm

The code Listing depicts the steps involved to invoke a Call-back function after the elapse of configured amount of time.

```
Stm_EnableModule(StmModuleNumber);
/*Enable interrupt and call-back */
Stm_EnableAlarm(StmModuleNumber,CompareRegisterId,TimerMode,Ticks,Stm_Applicati
onfunction);
```

Read timer value

The following code snippet shows call to `Stm_ReadTotalTimerValue()` and `Stm_ReadTimerValue()` API.

```
/* Read total timer value */
Timer = Stm_ReadTotalTimerValue(StmModuleNumber);
/*Read timer value of STM_TIM */
TimerValue= Stm_ReadTimerValue(StmModuleNumber,TimerNumber);
```

Compare Match Control

The following code snippet shows call to `Stm_SetCompareMatchControl()`

```
Stm_EnableModule(StmModuleNumber);
/* Set Compare Match Control Register*/
Stm_SetCompareMatchControl(StmModuleNumber,CompareRegisterId,Mstart,MSize);
/*Enable interrupt and call-back */
Stm_EnableAlarm(StmModuleNumber,CompareRegisterId,TimerMode,Ticks,Stm_Applicati
onfunction);
```

Note:

If Stm_EnableAlarm() is called without calling Stm_SetCompareMatchControl(), STM timer bit range [0-31] is used for comparison.

In Stm_SetCompareMatchControl(), Mstart and Msize should be used appropriately for other STM timer bit range.

Eg. For Stm timer bit range[4-35], MStart should be set to 0x4.

For CMP register bit range[0-30], Msize should be set to 0x1E.

Disable Alarm

After the call of `Stm_DisableAlarm()` it disables the STM interrupts and stops invoking a call-back function.

```
Stm_EnableAlarm(StmModuleNumber,CompareRegisterId,TimerMode,Ticks,Stm_Applicati
onfunction);
/*Stop invoking call-back function */
Stm_DisableAlarm(StmModuleNumber,CompareRegisterId);
```

1.1.5 Key architectural considerations

There are no key architectural considerations for the STM driver.

1.2 Assumptions of Use (AoU)

There are no AoU for the STM driver.

1.3 Reference information

1.3.1 Configuration interfaces

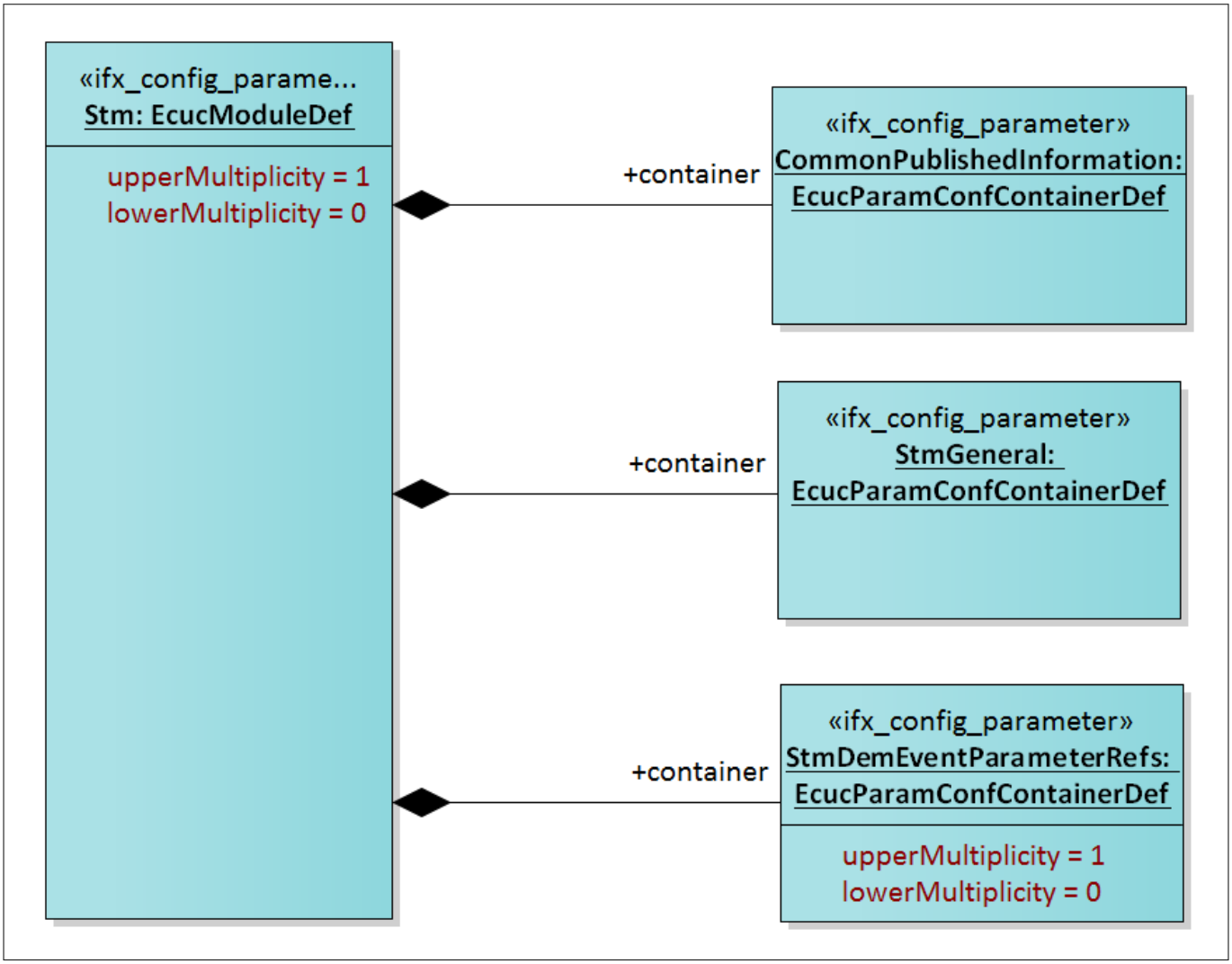

Figure 6 Container hierarchy along with their configuration parameters

1.3.1.1 Container: StmDemEventParameterRefs

Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: Pre-Compile

1.3.1.1.1 STM_E_CLC_ENABLE_ERR

Table 4 Specification for STM_E_CLC_ENABLE_ERR

Name	STM_E_CLC_ENABLE_ERR		
Description	DEM enable/disable for STM module enable failure.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef

Table 4 Specification for STM_E_CLC_ENABLE_ERR (continued)

Range	Reference to Node:		
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	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0		

1.3.1.2 Container: CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.
Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: -

1.3.1.2.1 ArMajorVersion

Table 5 Specification for ArMajorVersion

Name	`ArMajorVersion`		
Description	STM Autosar major version		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.1.2.2 ArMinorVersion

Table 6 Specification for ArMinorVersion

Name	`ArMinorVersion`
Description	STM Autosar minor version

Table 6 Specification for ArMinorVersion (continued)

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**	FALSE
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		

1.3.1.2.3 ArPatchVersion

Table 7 Specification for ArPatchVersion

Name	`ArPatchVersion`		
Description	STM Autosar patch version		
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**	FALSE
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		

1.3.1.2.4 ModuleId

Table 8 Specification for ModuleId

Name	`ModuleId`		
Description	STM Module ID		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	255		

Table 8 Specification for ModuleId (continued)

Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.1.2.5 Release

Table 9 Specification for Release

Name	`Release`		
Description	Aurix derivative used for the implementation,		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.1.2.6 SwMajorVersion

Table 10 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	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 driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE

Table 10 Specification for SwMajorVersion (continued)

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		

1.3.1.2.7 SwMinorVersion

Table 11 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	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 driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.1.2.8 SwPatchVersion

Table 12 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	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 driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE

Table 12 Specification for SwPatchVersion (continued)

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		

1.3.1.2.9 VendorId

Table 13 Specification for VendorId

Name	`VendorId`		
Description	STM Vendor ID		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.1.3 Container: StmGeneral

Stm General configuration container
Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: -

1.3.1.3.1 StmDevErrorDetect

Table 14 Specification for StmDevErrorDetect

Name	`StmDevErrorDetect`		
Description	STM Development error detection enable/disable		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		

Table 14 Specification for StmDevErrorDetect (continued)

Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.1.3.2 StmRuntimeApiMode

Table 15 Specification for StmRuntimeApiMode

Name	`StmRuntimeApiMode`		
Description	Stm Runtime Api Mode Configuration		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	STM_MCAL_SUPERVISOR: STM Mcal Supervisor Mode Selection STM_MCAL_USER1: STM Mcal User1 Mode Selection		
Default value	STM_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.1.3.3 StmVersionInfoApi

Table 16 Specification for StmVersionInfoApi

Name	`StmVersionInfoApi`		
Description	Enable/Disable Stm_GetVersionInfo() API.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		

Table 16	Specification for StmVersionInfoApi (continued)		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
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		

1.3.2 Functions - Type definitions

1.3.2.1 Stm_CallbackFnPtrType

Table 17	Specification for Stm_CallbackFnPtrType
Syntax	`Stm_CallbackFnPtrType`
Type	Pointer to a function of type void Function_Name (void)
File	`Stm.h`
Description	Function pointer used for Call-back function.
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0

1.3.2.2 Stm_TotalTimerCaptureType

Table 18	Specification for Stm_TotalTimerCaptureType	
Syntax	`Stm_TotalTimerCaptureType`	
Type	Structure	
File	`Stm.h`	
Range	uint32 LowerPart	Lower 32 bit value of 64 bit timer.
	uint32 UpperPart	Upper 32 bit value of 64 bit timer.
Description	Holds the complete 64 bit STM timer value.	
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 STM driver.

1.3.3.1 Stm_EnableModule

Table 19 Specification for `Stm_EnableModule` API

Syntax	`void Stm_EnableModule ( const uint8 ModuleNumber )`	
Service ID	0x0	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	ModuleNumber	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	none
Description	By default, all the STM timers will run after power on reset. The API does: i) Enable the STM in such a way that all the 32 bits of CMP1 register values are compared with first 32 bit values (TIM0) of 64 bit STM. ii) Enable the compare match interrupt. By default, tie the interrupt node with compare register - - CMP0 compare match interrupt directed to interrupt node 0 - CMP1 compare match interrupt directed to interrupt node 1 v) This service resets the RAM corresponding to the STM module number. In multi core context, user shall ensure that only the allocated STM peripheral resource is accessed in the core.	
Source	IFX	
Error handling	STM_E_CORE_TIMER_MISMATCH, STM_E_CLC_ENABLE_ERR	
Configuration dependencies	-	
User hints	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0	

1.3.3.2 Stm_EnableAlarm

Table 20 Specification for `Stm_EnableAlarm` API

Syntax	`void Stm_EnableAlarm ( const uint8 ModuleNumber, const uint8 CompareRegisterId, const uint8 TimerMode, const uint32 Ticks, const Stm_CallbackFnPtrType Stm_Applicationfunction )`	
Service ID	0x1	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different STM timers	
Parameters (in)	ModuleNumber CompareRegisterId TimerMode Ticks Stm_Applicationfunction	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon. Compare register number. CompareRegisterId value can be 0 or 1. Defines the STM running mode: 0: One Shot, 1: Continuous Timer ticks w.r.t selected timer. By default TIM0 is used as Timernumber for reference to invoke a Call-back function based on values of Ticks. If user wants to change the TimerNumber then user should call the API Stm_SetCompareMatchControl () with appropriate Mstart number. Function pointer used for Call-back function. Stm_CallbackFnPtrType is defined as void (*Stm_CallbackFnPtrType) (void). NULL_PTR is a invalid function
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	A service shall be provided to invoke a call-back function after the elapse of configured amount of time (scheduling a function). This callback function shall be called either once or continuously after the elapse of the time. The timeout shall be ticks. In multi core context, user shall ensure that only the allocated STM peripheral resource is accessed in the core.	
Source	IFX	
Error handling	STM_E_CORE_TIMER_MISMATCH, STM_E_CMPREG_FAILED, STM_E_TIMER_MODE_FAILED, STM_E_PARAM_POINTER	
Configuration dependencies	-	

Table 20	Specification for `Stm_EnableAlarm` API (continued)
User hints	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0

1.3.3.3 Stm_DisableAlarm

Table 21	Specification for `Stm_DisableAlarm` API	
Syntax	`void Stm_DisableAlarm ( const uint8 ModuleNumber, const uint8 CompareRegisterId )`	
Service ID	0x2	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different STM timers	
Parameters (in)	ModuleNumber CompareRegisterId	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon. Compare register number. CompareRegisterId value can be 0 or 1.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	none
Description	This function stops invoking call-back function. In multi core context, user shall ensure that only the allocated STM peripheral resource is accessed in the core.	
Source	IFX	
Error handling	STM_E_CMPREG_FAILED, STM_E_CORE_TIMER_MISMATCH	
Configuration dependencies	-	
User hints	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0	

1.3.3.4 Stm_SetCompareMatchControl

Table 22 Specification for `Stm_SetCompareMatchControl` API

Syntax	`void Stm_SetCompareMatchControl ( const uint8 ModuleNumber, const uint8 CompareRegisterId, const uint8 Mstart, const uint8 MSize )`	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different STM timers	
Parameters (in)	ModuleNumber CompareRegisterId Mstart MSize	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon. Compare register number. CompareRegisterId value can be 0 or 1. The lowest bit number of the 64-bit STM that is compared with the content of Compare register. Mstart values can be 0 to 31. MSize values can be 0 to 31. The number of bits in register CMP0 or CMP1 (starting from bit 0) that are used for the compare operation with the System Timer.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	API Service shall be provided to write the CMCON register: Match Start and Match Size for a particular STM module. In multi core context, user shall ensure that only the allocated STM peripheral resource is accessed in the core.	
Source	IFX	
Error handling	STM_E_MSTART_FAILED, STM_E_CMPREG_FAILED, STM_E_CORE_TIMER_MISMATCH, STM_E_MSIZE_FAILED	
Configuration dependencies	-	
User hints	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0	

1.3.3.5 Stm_ReadTimerValue

Table 23 Specification for `Stm_ReadTimerValue` API

Syntax	`uint32 Stm_ReadTimerValue ( const uint8 ModuleNumber, const uint8 TimerNumber )`	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different STM timers	
Parameters (in)	ModuleNumber TimerNumber	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon. The TimerNumber values shall be 0 (TIM0), 1(TIM1), 2(TIM2), 3(TIM3), 4 (TIM4), 5(TIM5) and 6(TIM6).
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	uint32	Current running timer value in ticks.
Description	Service to read individual TIM values of particular STM modules shall be provided. In multi core context, user shall ensure that only the allocated STM peripheral resource is accessed in the core.	
Source	IFX	
Error handling	STM_E_CORE_TIMER_MISMATCH, STM_E_INV_TIMER_NUMBER	
Configuration dependencies	-	
User hints	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0	

1.3.3.6 Stm_ReadTotalTimerValue

Table 24 Specification for `Stm_ReadTotalTimerValue` API

Syntax	`Stm_TotalTimerCaptureType Stm_ReadTotalTimerValue ( const uint8 ModuleNumber )`
Service ID	0x05

Table 24 Specification for `Stm_ReadTotalTimerValue` API (continued)

Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant for different STM timers	
Parameters (in)	ModuleNumber	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Stm_TotalTimerCaptureType	LowerPart : Lower 32 bit value of 64 bit timer. UpperPart : Upper 32 bit value of 64 bit timer
Description	Service to read complete 64 bit STM Timer value of a particular STM module shall be provided. In multi core context, user shall ensure that only the allocated STM peripheral resource is accessed in the core.	
Source	IFX	
Error handling	STM_E_CORE_TIMER_MISMATCH	
Configuration dependencies	-	
User hints	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0	

1.3.3.7 Stm_SleepModeHandle

Table 25 Specification for `Stm_SleepModeHandle` API

Syntax	`void  Stm_SleepModeHandle`
	`(`
	`  const uint8 ModuleNumber,`
	`  const uint8 SleepmodeControl`
	`)`	

Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	ModuleNumber SleepmodeControl	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon.

Table 25 Specification for `Stm_SleepModeHandle` API (continued)

		The values can be 0 : Enable STM during Controller sleep mode. 1 : Disable STM during Controller sleep mode.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to enable/disable of STM during controller sleep mode. In multi core context, user shall ensure that only the allocated STM peripheral resource is accessed in the core.	
Source	IFX	
Error handling	STM_E_CORE_TIMER_MISMATCH, STM_E_SLEEP_MODE_FAILED	
Configuration dependencies	-	
User hints	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0	

1.3.3.8 Stm_GetVersionInfo

Table 26 Specification for `Stm_GetVersionInfo` API

Syntax	`void Stm_GetVersionInfo ( Std_VersionInfoType * const versioninfo )`	
Service ID	0x7	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.
Parameters (in - out)	-	-
Return	void	-
Description	This function returns the version information of this module. The version information include: Module ID, Vendor ID and Vendor specific version numbers.	
Source	IFX	

Table 26 Specification for `Stm_GetVersionInfo` API (continued)

Error handling	STM_E_PARAM_POINTER
Configuration dependencies	StmVersionInfoApi
User hints	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0

1.3.4 Notifications and Callbacks

This section lists all the notifications and callbacks of the STM driver.

1.3.4.1 Stm_CallbackFunction

Table 27 Specification for `Stm_CallbackFunction` API

Syntax	`void Stm_CallbackFunction ( void )`
Service ID	
Sync/Async	Synchronous
ASIL Level	QM
Re-entrancy	Non Reentrant
Parameters (in)	-
Parameters (out)	-
Parameters (in - out)	-
Return	void
Description	Stm call back notification: After configured elapsed amount of time, user will get the call back notification.
Source	IFX
Error handling	-
Configuration dependencies	-
User hints	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0

1.3.5 Scheduled functions

The STM driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

This section lists all the Interrupt handlers of the STM driver.

1.3.6.1 Stm_Isr

Table 28 Specification for `Stm_Isr` API

Syntax	`void Stm_Isr ( const uint8 ModuleNumber, const uint8 InterruptNode )`	
Service ID	0x8	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	ModuleNumber InterruptNode	System timer peripheral number. The values shall be 0, 1, 2,3,4 and 5 based on the number of STM peripheral available on the silicon. 0 or 1
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	none
Description	This function is the interrupt handler and collects the interrupt node and invoke the call-back function	
Source	IFX	
Error handling	-	
Configuration dependencies	-	
User hints	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0	

1.3.7 Callout

The STM driver does not provide any callout.

1.3.8 Error Handling

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

Error Name: Description	Source	Error ID	Type
STM_E_CORE_TIMER_MISMATCH: Wrong STM Module number is passed to the function.	IFX	0x65	DET
STM_E_CMPREG_FAILED: Invalid STM compare register number is passed.	IFX	0x1	DET
STM_E_TIMER_MODE_FAILED: Invalid timer mode is passed to the function.	IFX	0x2	DET
STM_E_PARAM_POINTER: Invalid Call back function is passed to the function.	IFX	0x3	DET
STM_E_MSTART_FAILED: Invalid lowest bit number of the 64-bit STM is passed	IFX	0x4	DET
STM_E_SLEEP_MODE_FAILED: Invalid sleep mode control is passed to the function.	IFX	0x5	DET
STM_E_INV_TIMER_NUMBER: Invalid STM timer number is passed.	IFX	0x6	DET
STM_E_MSIZE_FAILED: Invalid msize is passed to the API.	IFX	0x7	DET
STM_E_CLC_ENABLE_ERR: The hardware does not react in the expected time (hardware malfunction).	IFX	Assigned by DEM	DEM

1.3.9 Deviations and limitations

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

1.3.9.1 Deviations

This section describes the deviations of the STM driver.

1.3.9.1.1 Software specification

The STM driver does not have any deviations.

1.3.9.1.2 AMDC Violations

The STM driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

The STM driver does not have any VSMD violations.

1.3.9.2 Limitations

The STM driver does not have any limitations.

Revision history

Major changes since the last revision

Date	Version	Description
2020-11-18	2.0	Document is released
2020-11-11	1.1	• Error handling format of all the APIs and functions in Functions – APIs, Notifications and Callbacks and Interrupt service routines updated • Reference to Dem_SetEventStatus API for AUTOSAR 4.4.0 added in section DEM under Integration with AUTOSAR stack • Autosar Version applicability information added in Configuration interfaces,Functions - Type definitions, Functions – APIs, Notifications and Callbacks and Interrupt service routines sections • User hints added for all the APIs and functions in Functions – APIs, Notifications and Callbacks and Interrupt service routines sections • Error Handling section format modified by consolidating all the errors to a single table • Deviations and limitations section format updated
2020-08-13	1.0	Document is released
2020-08-10	0.1	• Initial version • Stm driver chapter moved from TC3xx_SW_MCAL_UM_DEMO to this document

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-11-18

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-kcy1596794565979

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.