MCAL User Manual for Ocu

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 Ocu 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
- Specification of OCU Driver, AUTOSAR_SWS_OCU_Driver, AUTOSAR Release 4.2.2
- Specification of OCU Driver, AUTOSAR_SWS_OCU_Driver, AUTOSAR Release 4.4.0

Table of contents

Table of contents

Table of contents

1 Ocu driver

1.1 User information

1.1.1 Description

The OCU driver is responsible for triggering an event on a free- running counter compare match. The event can either be a pin level change, a DMA trigger, an ADC trigger or just a notification. The services provided by the OCU driver are:

- Start/stop a channel
- Configure the compare match values
- Set the pin action on the sub-sequent compare matches
- Enable/disable notifications and to set the pin level on a stopped channel

The TOM and ATOM slices of the GTM will be used to achieve the configured functionality. *Note: The OCU driver does not support GTM-less device (for example, TC35x).*

1.1.2 Hardware-software mapping

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

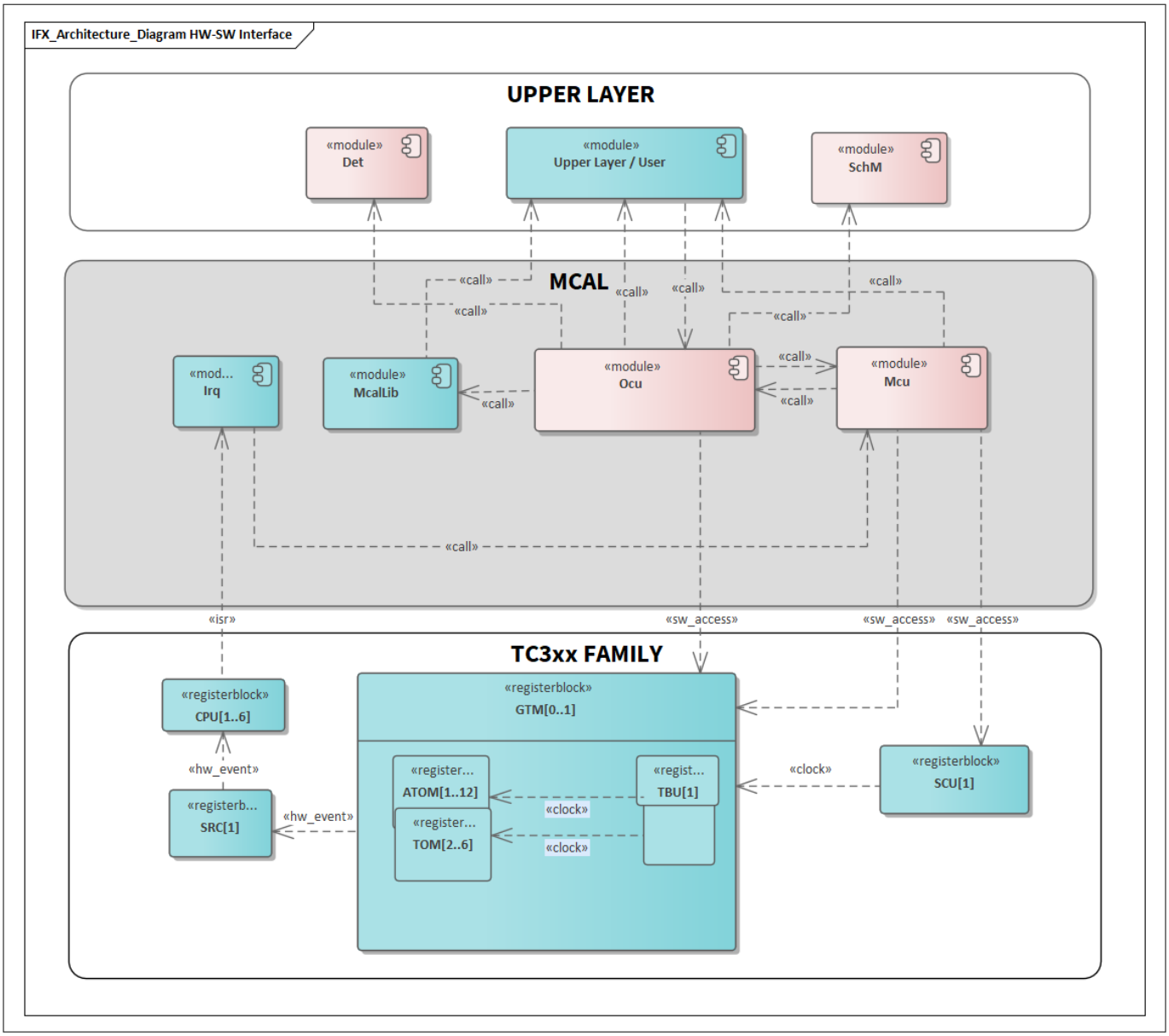


Figure 1 Mapping of hardware-software interfaces

1.1.2.1 GTM: primary hardware peripheral.

Hardware functional features

The OCU driver uses the GTM IP (TOM/ATOM) for realizing the following features,

- TOM CM1 compare match
- ATOM CM1 compare match in SOMP mode
- ATOM CM0/CM1 compare match in SOMC mode
- ATOM in SOMC mode pin level changes, ADC trigger
- ATOM in SOMP mode DMA trigger
- TOM/ATOM notification
- TOM DMA trigger

The key hardware functional features are:

- Channel clock source configuration
- Cyclic comparators for the compare match(ATOM SOMC mode - shared TBU clock)
- Signaling mechanisms: Global enable/disable mechanism, output enable mechanism
- ATOM operation mode are SOMP, SOMC

The unsupported features of the GTM-TOM/ATOM IP are:

- Global force update signaling mechanism
- Continuous down counting down mode, one shot up mode, one shot down mode
- ATOM operational mode: SOMI or SOMS
- TOM/ATOM - SOMP mode triggering port pin or ADC configuration

Users of the hardware

The CMU functional block of the GTM peripheral and TBU_TS[x] timer are exclusively handled by the MCU driver. The MCU driver provides APIs to program the GTM SFRs. The OCU driver uses these APIs to write the GTM SFRs. Additionally, updates to the channel-specific SFRs are performed by the OCU driver. Since these channels are exclusively reserved for the OCU driver, access to the channel-specific SFRs from other drivers or user software is not allowed.

The GTM TOM/ATOM functional blocks are used by PWM, GPT, WDG and OCU driver.

Note: A TOM/ATOM channel of the GTM is exclusively used by its user. While users of TOM/ATOM channels are many, a channel is allocated to and used by exactly by one driver.

Hardware diagnostic features

Not applicable.

Hardware events

The OCU module uses the compare match events generated by the timer channel to provide notification of events to application and perform an action (pin level change, ADC trigger or DMA trigger) without software intervention. Mcu module is responsible for handling interrupts and invoking the callback function to service the interrupt.

1.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The OCU driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB and fGTM clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver, and 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

The SMU alarms configured for the SCU IP are not monitored by the OCU driver.

Hardware events

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

1.1.3 File structure

1.1.3.1 C file structure

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

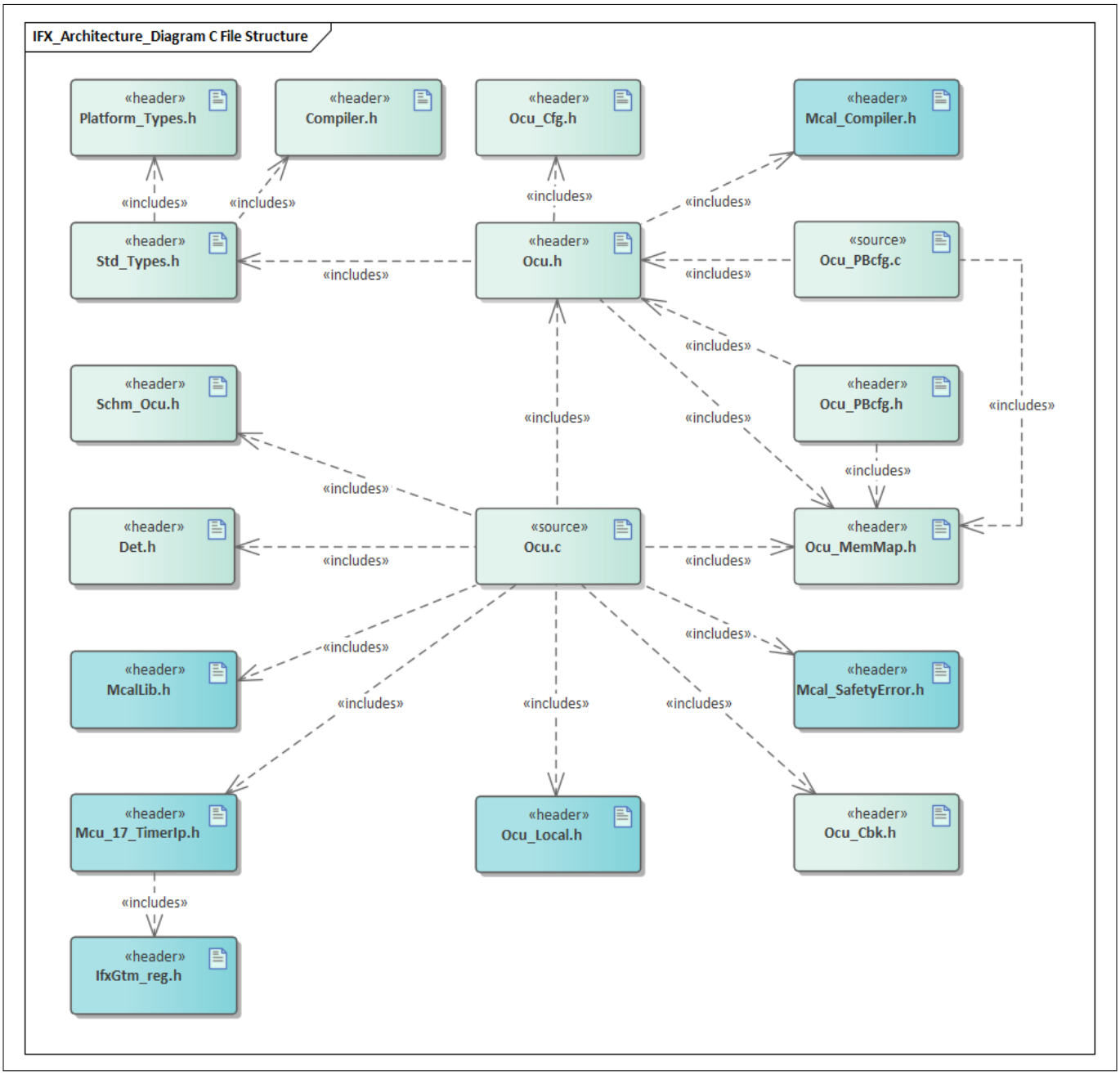


Figure 2 Ocu_C_File_Structure-1.png

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

(table continues...)

Table 2 (continued) C file structure

File name	Description
IfxGtm_reg.h	SFR header file for GTM
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
Mcal_Compiler.h	Header file providing abstraction for TriCore™-intrinsic instruction.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Ocu.c	File (static) containing implementation of APIs.
Ocu.h	Header file (static) defining prototypes of configuration data structures and APIs
Ocu_Cbk.h	Includes callback header definition
Ocu_Cfg.h	Header file (generated) containing constants and pre-processor macros
Ocu_Local.h	Header file defining type definition of global data and inline API's, which can used across source files
Ocu_MemMap.h	File (static) containing the memory section definitions used by the OCU driver
Ocu_PBcfg.c	File (generated) containing objects to data structures
Ocu_PBcfg.h	File (generated) containing declaration of the post-build configuration data structures
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
Schm_Ocu.h	The header file contains the definitions of OCU critical sections
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

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

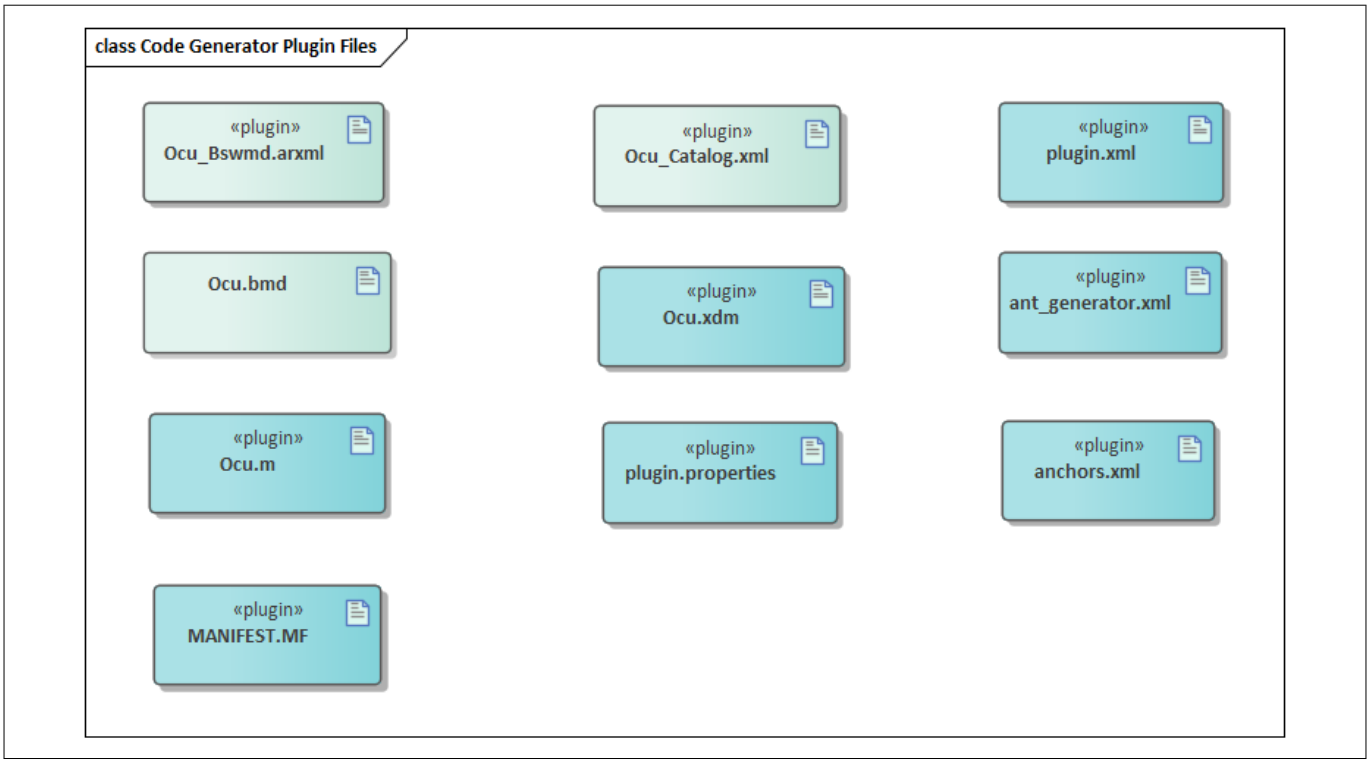


Figure 3 **Ocu_Code_Generator_Plugin_Files-1.png**

Table 3 **Code generator plugin files**

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

1.1.4 Integration hints

This section lists the key points that an integrator or user of the OCU driver must consider.

The OCU features and hardware configurations as follows:

Table 4 OCU channel features supported different HW channels

Feature	TOM channel	ATOM channel in SOMP mode	ATOM channel in SOMC mode
Counter type	Exclusive	Exclusive	Shared (TBU_TS0/1/2)
Notifications	Yes	Yes	Yes
Pin	No	No	Yes
DMA trigger	Yes	Yes	No
ADC trigger	No	No	Yes

1.1.4.1 Integration with AUTOSAR stack

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

- **ECU State Manager (EcuM)**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the 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.

Initialization of OCU:

The user of OCU driver may use APIs of EcuM to initialize the driver. The initialization of the driver should be invoked from each CPU core, which intends to use the services of the OCU driver. All cores can execute initialization simultaneously.

De-initialization of OCU:

The user of OCU driver may use APIs of EcuM to de-initialize the driver. The de-initialization of the driver should be invoked from each CPU core that used the services of the OCU driver. All cores can execute de-initialization simultaneously.

- **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 re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Ocu_MemMap.h` file.

The `Ocu_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 re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

```
/**** GLOBAL RAM DATA -- NON-CACHED LMU ****/
#if defined OCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 /*****User pragmas here for Non-cached LMU*****/
 #undef OCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 #undef MEMMAP_ERROR
#elif defined OCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 /*****User pragmas here for Non-cached LMU*****/
 #undef OCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 #undef MEMMAP_ERROR

/**** CORE[x] RAM DATA -- DSPR ****/ /*[x]=0..5*/
#elif defined OCU_START_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
 /*****User pragmas here for CORE[x] DSPR*****/
 #undef OCU_START_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
 #undef MEMMAP_ERROR
#elif defined OCU_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
 /*****User pragmas here for CORE[x] DSPR*****/
 #undef OCU_STOP_SEC_VAR_CLEARED_ASIL_B_CORE[x]_32
 #undef MEMMAP_ERROR

/**** GLOBAL CONST DATA -- PF[x] ****/
#elif defined OCU_START_SEC_CONST_ASIL_B_GLOBAL_32
 /*****User pragmas here for PF[x]*****/
 #undef OCU_START_SEC_CONST_ASIL_B_GLOBAL_32
 #undef MEMMAP_ERROR
#elif defined OCU_STOP_SEC_CONST_ASIL_B_GLOBAL_32
 /*****User pragmas here for PF[x]*****/
 #undef OCU_STOP_SEC_CONST_ASIL_B_GLOBAL_32
 #undef MEMMAP_ERROR

/**** GLOBAL CONFIG DATA -- PF[x] ****/
#elif defined OCU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef OCU_START_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined OCU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef OCU_STOP_SEC_CONFIG_DATA_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR


/**** CORE[x] CONFIG DATA -- PF[x] ****/ /*[x]=0..5*/
#elif defined OCU_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef OCU_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined OCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef OCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
```

```
  #undef MEMMAP_ERROR


/**** CODE -- PF[x] ****/
#elif defined OCU_START_SEC_CODE_ASIL_B_GLOBAL
 /*****User pragmas here for PF[x]*****/
 #undef OCU_START_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR
#elif defined OCU_STOP_SEC_CODE_ASIL_B_GLOBAL
 /*****User pragmas here for PF[x]*****/
 #undef OCU_STOP_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Ocu_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 BSW modules. The OCU driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the OCU 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 needs to be replaced with a complete DET module during the integration phase.

- **DEM**

DEM module is not required for integrating the OCU driver.

- **BSW Scheduler Module (SchM)**

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

- SetPinAction

- SetThresholdValue

The `SchM_Ocu.h` and `SchM_Ocu.c` files are provided in the MCAL package as an example code and needs to updated by the integrator. The user must implement the SchM functions defined by the OCU 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 of SchM_Ocu.c ****/
#include "Os.h"

/* Disable the interrupts for entering critical section */
void SchM_Enter_Ocu_SetPinAction(void)
{
 SuspendAllInterrupts();
}
/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Ocu_SetPinAction(void)
{
 ResumeAllInterrupts();
}

/* Disable the interrupts for entering critical section */
void SchM_Enter_Ocu_SetThresholdValue(void)
{
 SuspendAllInterrupts();
}
/* Re-enable the interrupt for exiting the critical section */
void SchM_Exit_Ocu_SetThresholdValue(void)
{
 ResumeAllInterrupts();
}
```

- **Safety error**

The OCU 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).

- **Notifications and call-backs**

The OCU driver does not implement any notifications. However, the driver reports the compare match event through notification functions. These notification functions can be configured by the user in EB Tresos for each OCU channel.

The OCU does not expect any call-backs from application. But the OCU driver needs the call-back ISR from the MCU driver.

- **Operating system(OS)**

The OS or the 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 the application.

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

1.1.4.2 Multicore and Resource Manager

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

- Each OCU channel can be allocated to any core using the Resource Manager.
- Interrupts raised by an OCU channel must be serviced by the CPU core to which the channel has been allocated to, if DMA triggering for that channel is not configured.
- OCU channels using GTM-ATOM, channel GTM-ATOM[i]_CH[2x] and ATOM[i]_CH[2x+1], must be allocated to same core as these two channels share same interrupt line.
- OCU channels using GTM-TOM, channel GTM-TOM[i]_CH[2x] and TOM[i]_CH[2x+1], must be allocated to same core as these two channels share same interrupt line.
- Locating 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 the OCU driver is placed under single MemMap section. It can be relocated to any PFlash/DFlash region.

Data section:

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

Configuration data and constants:

The configuration data sections marked as specific to a core should be re-located to the PFlash/DFlash of the same core. The sections marked as global should be relocated to the PFlash/DFlash of the master core.

Note: Re-locating 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 OCU driver is dependent on the MCU driver for clock configuration and timer-IP related services. The initialization of the OCU driver must be started only after completing the MCU initialization. The following must be considered while configuring the MCU driver in EB Tresos:

- The GTM TOM/ATOM timers used by the OCU driver must be reserved in the MCU configuration for exclusive use by the OCU.
- For Pin action, the TOUT configuration should be configured in MCU configuration through parameter "GtmTimerPortPinSelect".

Access of Shared GTM Sfrs

If channels of the same TOM/ATOM module is shared between the application and the OCU driver, user shall ensure shared register of TOM/ATOM modules is accessed using the MCU timer-IP library APIs.

1.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the OCU driver through the PORT configuration as output and initialize the port pins prior to invoking of OCU initialization.

1.1.4.5 DMA support

An OCU channel compare match event can trigger a DMA channel. User is responsible to configure and initialize the DMA channel to accept the hardware trigger from the OCU channel.

1.1.4.6 Interrupt connections

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

User should enable interrupts of the corresponding OCU channel in interrupt configuration register. The interrupt configuration registers of different hardware used by OCU channels are as follows:

Table 5 GTM interrupts

GTM hardware used	SRC register
GTM-TOM	SRC_GTMTOMwx (w= TOM module; shared between 2x and 2x+1 TOM channel)
GTM-ATOM	SRC_GTMATOMwx (w= ATOM module; shared between 2x and 2x+1 ATOM channel)

- For OCU channels not triggering DMA

The priority(SRPN) is chosen by the user but the Type Of Service(TOS) should be the CPU to which the OCU channel is allocated.

- For OCU channels triggering DMA

The priority(SRPN) should be the DMA channel which needs to be triggered on a compare match event of that OCU channel and Type Of Service(TOS) should be DMA.

1.1.4.7 Example usage

Configuration:

User can confugre either a TOM or an ATOM channel for each OCU channel.

The mode of the ATOM channel is selected during the generation phase depending on the timer selected for that channel. If one of the TBU timer is selected, ATOM channel is configured for the SOMC mode else channel is configured in the SOMP mode.

TOM channel supports only one mode and no selection is required. Also, TBU timer cannot be used as a timer for TOM channel.

DMA triggers:

Each OCU channel using TOM or ATOM(in SOMP) can trigger one DMA channel(configured as SRPN). The compare match event of the OCU channel is routed as DMA trigger. Hence if an OCU channel is triggering a DMA channel, notifications cannot be issued by the OCU channel.

```
Ocu_Init(<Configration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* DMA gets triggered on each compare match of default threshold until the call to
Ocu_SetAbsoluteTreshold/Ocu_SetRelativeTreshold */

<compare value write in/out interval> = Ocu_SetAbsoluteTreshold(<logical channel ID>,
<reference value>, <absolute threshold value>);
/* Return value indicates if the new compare value is written within the reference interval or
outside */

/* DMA gets triggered on each compare match of new absolute threshold set by the previous call
of Ocu_SetAbsoluteTreshold */

<compare value write in/out interval> = Ocu_SetRelativeTreshold(<logical channel ID>, <relative
threshold value>);
/* Return value indicates if the new compare value is written within the reference interval or
outside */

/* DMA gets triggered on each compare match of new threshold set by the previous call of
Ocu_SetRelativeTreshold */

Ocu_StopChannel(<logical channel ID>);

/* DMA triggering is stopped */

Ocu_DeInit();
```

ADC triggering:

The output level of the ATOM channel (in SOMC mode) is routed to the ADC as a trigger line.

When the OCU channel is configured for both pin and ADC triggering and the compare match events are set, both ADC trigger and pin state are affected.

Note: When the `Ocu_SetPinState` API is called, both pin and ADC trigger (if configured) are affected. ADC triggering is an unintended consequence when this API is invoked.

The user of the ADC driver can configure the trigger from the OCU driver either as a level trigger or as an edge trigger with the following behavioral considerations:

- For OCU channel configured for both pin and ADC triggering, default pin level selected by the user is the level of the ADC trigger line after initialization. The trigger line level can be changed on a compare match event using the `Ocu_SetPinAction` API. The change of pin level gets reflected in the ADC trigger line.

```
Ocu_Init(<Configration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* ADC trigger line is unchanged */

Ocu_SetPinAction(<logical channel ID>, OCU_SET_HIGH);

/* ADC trigger line is set to high on the next compare match of default threshold */

<compare value write in/out interval> = Ocu_SetRelativeTreshold(<logical channel ID>, <relative
threshold value>);
Ocu_SetPinAction(<logical channel ID>, OCU_LOW); /* Applicable if pin is configured*/
/* Return value indicates if the new compare value is written within the reference interval or
outside */

/* ADC trigger line is set to low on the next compare match of new threshold */

Ocu_StopChannel(<logical channel ID>);

/* ADC trigger line is unchanged */

Ocu_SetPinState(<logical channel ID>, OCU_HIGH);

/* ADC trigger line is set to high */

Ocu_DeInit();
```

- For OCU channel configured for ADC triggering but not supporting pin, default ADC trigger line level is LOW after initialisation. The ADC trigger line gets toggled on every compare match event. Hence, for an edge

triggered ADC, user should enable both edge triggering(in ADC) to trigger ADC on each compare match. For a level trigger, user can configure threshold values to start/stop the ADC triggering.

```
Ocu_Init(<Configration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* ADC trigger line is toggled on each compare match(default threshold) */

<compare value write in/out interval> = Ocu_SetRelativeTreshold(<logical channel ID>, <relative
threshold value>);

/* Return value indicates if the new compare value is written within the reference interval or
outside */

/* ADC trigger line is toggled on each compare match(new threshold) */

Ocu_StopChannel(<logical channel ID>);

/* ADC trigger line is unchanged */

Ocu_DeInit();
```

Notification:

Each OCU channel can issue a notification callback to the application on a compare match event.

```
Ocu_Init(<Configration pointer>);

Ocu_StartChannel(<logical channel ID>);

/* Notifications are issued on each compare match(default threshold) */

<compare value write in/out interval> = Ocu_SetAbsoluteTreshold(<logical channel ID>,
<reference value>, <absolute threshold value>);
/* Return value indicates if the new compare value is written within the reference interval or
outside */

/* Notifications are issued on each compare match(new threshold) */

<compare value write in/out interval> = Ocu_SetRelativeTreshold(<logical channel ID>, <relative
threshold value>);
/* Return value indicates if the new compare value is written within the reference interval or
outside */

/* Notifications are issued on each compare match(new threshold) */

Ocu_StopChannel(<logical channel ID>);

/* Notifications are no longer issued */

Ocu_DeInit();
```

1.1.5 Key architectural considerations

1.1.5.1 DMA support

Each OCU channel(using TOM/ATOM in SOMP) is capable to trigger a DMA transfer with the below restrictions:

-To enable a DMA transaction on a compare match, user shall configure the Type Of Service(TOS) of the corresponding OCU channel's service node (TOM or ATOM) as "DMA" and "SRPN" with the DMA channel number which needs to be triggered.

-The hardware resource which shares the interrupt line with OCU channel which supports DMA should not be used.

TOM(2[x]) and TOM(2[x]+1) share same interrupt line. x in range 0 to 7

ATOM(2[x]) and ATOM(2[x]+1) share same interrupt line. x in range 0 to 3

Hence, if TOM0 is used for OCU channel triggering DMA, TOM1 should not be used for any other OCU channel/ module.

1.1.5.2 Default pin action

The pin action after initialization is set to OCU_DISABLE. This is to ensure that there is no unintended pin level changes on starting a channel after initialization.

Note: If ADC triggering with no pin support is configured, the OCU channel's output toggles on each compare match.

1.1.5.3 Default pin level

If OcuOutputPinDefaultState is not configured, the default state of the OCU channel pin is set to OCU_LOW.

1.1.5.4 Threshold values at Start channel

OCU_StartChannel does not update the threshold values, but uses the previously programmed threshold values.

1.2 Assumptions of Use (AoU)

The AoU for the Ocu driver are as follows:

- **Configuration pointer across multiple cores**

User shall ensure, Configuration pointer passed for Ocu initialization should be same across all the configured cores.

[cover parentID OCU={D2C31E23-BDAD-4e28-83DD-69B0BF9EAF7D}]

- **ADC trigger on Ocu_SetPinState**

User shall be aware of, on configuring both ADC and port pin as a trigger on the OCU compare match. On Ocu_SetPinState API/function call, changing the port pin state may cause a change in the channel output signal. This would potentially cause an unexpected ADC trigger based on trigger edge configured for the ADC channel.

[cover parentID OCU={2A98C475-744B-45fb-8CC4-FE7AE42DBB0A}]

- **Code generation**

User shall ensure the generated configuration structures are correct against the intended GUI configurations.

[cover parentID OCU={20319FAC-437C-4362-B9AE-FBF21CD9E3B0}]

- **ADC trigger on both edges**

In case the OCU channel is configured only to support the ADC, the default compare match action is set to "Toggle".

Rationale: 1. As "Ocu_SetPinAction" and "Ocu_SetPinState" function are pre compile time configurable API ("Ocu_SetPinAction" and "Ocu_SetPinState" API are disabled, if pin actions are not configured on compare match). Therefore, users do not have an have option to set/change the Edge level(falling/raising) for triggering the ADC on a compare match. Hence, the compare match action is set to Toggle.

[cover parentID OCU={C5B71056-8154-4379-A5A7-471EF2C5A58D}]

- **Correctness of configuration pointer**

User shall ensure the correct configuration pointer is being passed for initializing the Ocu driver.

[cover parentID OCU={7A50611B-46D9-43a3-9342-18DC2364D603}]

- **EVADC MUX configurations**

User shall ensure that EVADC MUX configurations (that is, ADC channel connected to ATOM through EVADC MUX) are according to the group selection in the OCU configuration.

[cover parentID OCU={3C177397-9C0D-4b61-816B-B8DC52734D6D}]

- **Compare match event between Ocu_StartChannel and Ocu_EnableNotification**

User shall be aware that if the compare match event occurs after Ocu_StartChannel but before Ocu_EnableNotification, no callback will be reported for that event. However, future compare match events will be notified through the callback.

[cover parentID OCU={6BF58900-A642-4fb8-AC36-E726504DC406}]

- **No ADC conversion**

In case OCU channel is configured to support both ADC and PORT, the default compare match action is set to NO ACTION. As PORT is enabled/configured to get triggered on compare match through "Ocu_SetPinAction" and "Ocu_SetPinState" APIs, Compare match action and current ADC signal level can be modified. Hence, user can configure ADC conversion on raising edge or falling edge or both the edges.

Scenario: When the initial state (ADC Trigger Signal) is not set to a desired value, the first compare match event may generate an edge, which may not trigger ADC conversion. For example: ADC configured for conversion on falling edge, initial state is already low and signal is toggled on next compare match(that is, The next compare match is the raising edge).

Hence user shall ensure output level is initialized before the Ocu start channel.

[cover parentID OCU={3316FCFA-E4C8-4105-A79B-EF93BFC70982}]

- **No ADC trigger**

In case the OCU channel is configured to support both ADC and PORT, the default compare-match action is set to NO ACTION. As PORT is enabled/configured to get triggered on compare match, through "Ocu_SetPinAction" and "Ocu_SetPinState" APIs, compare -match action and current ADC signal level can be modified. Hence, user can configure ADC conversion on raising edge or falling edge or both edges.

Scenario: When the initial state (ADC trigger signal) is not set to a desired value, the first compare match event may not generate an edge required to trigger ADC conversion. For Example.: ADC configured for falling edge, initial state is already low and on next compare match the signal is configured to set LOW.

The user shall ensure output level is initialized before the start channel.

[cover parentID OCU={7EB6515B-2749-4e6c-9788-25265C24AE80}]

- **Pin/ADC signal state on stop channel**

User shall be aware that on calling the Ocu_StopChannel API, only compare match event notifications are disabled, but PIN state or ADC signal levels are unchanged. (For Example, When the ADC is configured as level trigger, on calling the Ocu_StopChannle API the ADC trigger levels are not changed).

[cover parentID OCU={0D2BF3D1-A029-4ed5-84C9-DEC6B75554CD}]

- **Port Pin state on deinit**

User shall ensure, port pin is SET to the required state in GPIO mode before invoking the deinit Api.

Rational: When deinit is invoked, all SFRs are placed to power on reset state and hence there may be a change on the output state of the channel.

[cover parentID OCU={7128C809-9265-40dd-8FB8-7D099E979BCC}]

- **Return value of Ocu_SetAbsoluteThreshold/ Ocu_SetRelativeThreshold**

In the Ocu_SetAbsoluteThreshold/ Ocu_SetRelativeThreshold Api, the OCU_CM_OUT_REF_INTERVAL is returned even in case of an error and it may lead to wrong system reaction. Hence user shall perform an explicit check of DET or SE error to validate the return value.

[cover parentID OCU={2392C9C6-5EEC-45be-9971-4040BC89C527}]

- **Software Sequence on Ocu_Init**

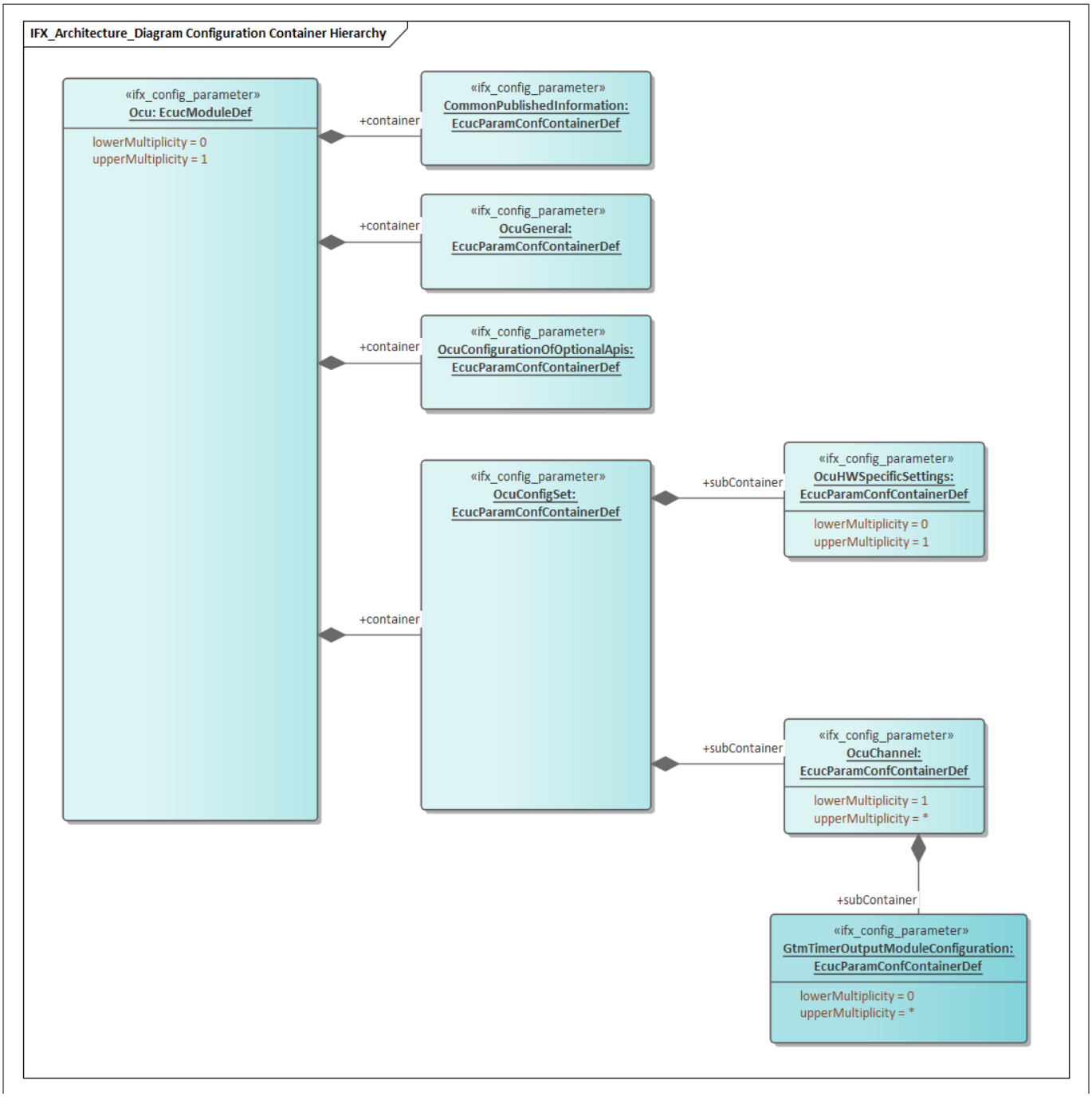
User shall ensure that the Mcu_Init and Port_Init API's are called before the Ocu_Init.

[cover parentID OCU={058E1602-6790-4304-AF7B-C0D06CE9D262}]

1.3 Reference information

1.3.1 Configuration interfaces

Supported configuration variant: Post-Build


IFX_Architecture_Diagram Configuration Container Hierarchy
«ifx_config_parameter»
Ocu: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
«ifx_config_parameter»
CommonPublishedInformation:
EcucParamConfContainerDef
+container
«ifx_config_parameter»
OcuGeneral:
EcucParamConfContainerDef
+container
«ifx_config_parameter»
OcuConfigurationOfOptionalApis:
EcucParamConfContainerDef
+container
«ifx_config_parameter»
OcuConfigSet:
EcucParamConfContainerDef
+subContainer
«ifx_config_parameter»
OcuHWSpecificSettings:
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
«ifx_config_parameter»
OcuChannel:
EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
«ifx_config_parameter»
GtmTimerOutputModuleConfiguration:
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *

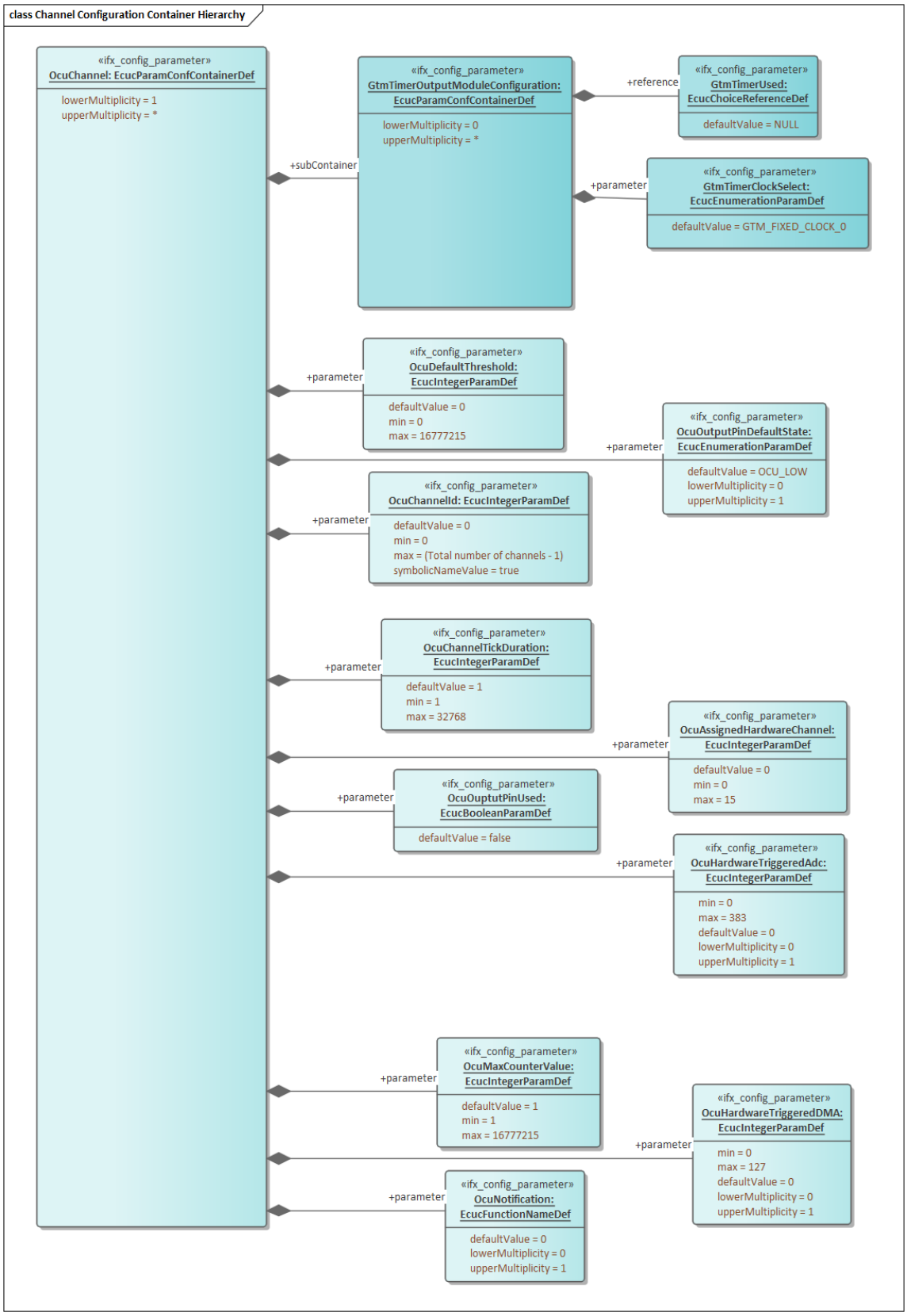


Figure 4 Container hierarchy along with their configuration parameters

1.3.1.1 Container: GtmTimerOutputModuleConfiguration

This container contains the parameters for configuring the selected TOM/ATOM channel.
Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: Post-Build

1.3.1.1.1 GtmTimerClockSelect

Table 6 Specification for GtmTimerClockSelect

Name	`GtmTimerClockSelect`		
Description	This parameter decides the Clock Source for TOM/ATOM timer.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 is selected for ATOM channel. GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 is selected for ATOM channel. GTM_FIXED_CLOCK_0: Fixed Clock 0 is selected for TOM channel. GTM_FIXED_CLOCK_1: Fixed Clock 1 is selected for TOM channel. GTM_FIXED_CLOCK_2: Fixed Clock 2 is selected for TOM channel. GTM_FIXED_CLOCK_3: Fixed Clock 3 is selected for TOM channel. GTM_TBU_TS0: TBU_TS0 is selected for ATOM channel. GTM_TBU_TS1: TBU_TS1 is selected for ATOM channel. GTM_TBU_TS2: TBU_TS2 is selected for ATOM channel.		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerUsed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.2 GtmTimerUsed

Table 7 Specification for GtmTimerUsed

Name	`GtmTimerUsed`

(table continues...)

Table 7 (continued) Specification for GtmTimerUsed

Description	The TOM/ATOM Channel resource assigned to the Ocu channel. The HW resource used should be unique in a configuration set of OCU. This parameter is list of all the GTM timer channels (TOM/ATOM) used by OCU Driver. Referred timer channel in MCU should have TomChannelUsage/ AtomChannelUsage as USED_BY_OCU_DRIVER		
Multiplicity	1..1	**Type**	EcucChoiceReferenceDef
Range	Reference to Node: McuGtmTomChannelAllocationConf, McuGtmAtomChannelAllocationConf		
Default value	NULL		
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.		

1.3.1.2 Container: Ocu

Configuration of Ocu (Output Compare Unit) module

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.3 Container: OcuChannel

Configuration of an individual OCU channel.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

1.3.1.3.1 OcuAssignedHardwareChannel

Table 8 Specification for OcuAssignedHardwareChannel

Name	`OcuAssignedHardwareChannel`		
Description	The physical hardware channel that is assigned to this logical channel. *NOTE: This parameter is non-editable and not used. The hardware channel is configured through "GtmTimerUsed" parameter.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 15		

(table continues...)

Table 8 (continued) Specification for OcuAssignedHardwareChannel

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

1.3.1.3.2 OcuChannelEcucPartitionRef

Table 9 Specification for OcuChannelEcucPartitionRef

Name	`OcuChannelEcucPartitionRef`		
Description	Maps an OCU channel to zero or multiple ECUC partitions to limit the access to this channel. The ECUC partitions referenced are a subset of the ECUC partitions where the OCU driver is mapped to. *Note: Parameter support is added only for AUTOSAR schema compliance, this parameter is not used in code generation logic, hence this parameter is made editable false.*		
Multiplicity	0..*	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.3.3 OcuChannelId

Table 10 Specification for OcuChannelId

Name	`OcuChannelId`

(table continues...)

Table 10 (continued) Specification for OcuChannelId

Description	Channel Id of the OCU channel. This value will be assigned to the symbolic name derived from the OcuChannel container short name. It defines the assignment of the channel to the physical OCU hardware channel. The value of the parameter should be unique in a configuration set. *NOTE: Minimum value is set as default value.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - (Total number of channels - 1)		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.4 OcuChannelTickDuration

Table 11 Specification for OcuChannelTickDuration

Name	`OcuChannelTickDuration`		
Description	Specifies the tick duration of the counter of the channel. This parameter is the number of the input clock edges (rising edges or falling edges exclusively) counted each time to increase the counter by one unit. *Note: This parameter is not-used,not editable and default value is set to 1.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 32768		
Default value	1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.5 OcuDefaultThreshold

Table 12 Specification for OcuDefaultThreshold

Name	`OcuDefaultThreshold`		
Description	Value of comparison threshold used for Initialization.(in ticks) The value should be less than the maximum counter value. For ATOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 24-bit range. For TOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 16-bit range. *NOTE: Minimum value is set as default value.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	GtmTimerClockSelect, GtmTimerUsed, OcuChannelTickDuration, OcuMaxCounterValue		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.6 OcuHardwareTriggeredAdc

Table 13 Specification for OcuHardwareTriggeredAdc

Name	`OcuHardwareTriggeredAdc`		
Description	This parameter is used to allow the OCU channel to trigger an ADC channel upon compare match. The value of the parameter represents the ADC Group to trigger. *NOTE: Minimum value is set as default value.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 383		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	GtmTimerClockSelect		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.7 OcuHardwareTriggeredDMA

Table 14 Specification for OcuHardwareTriggeredDMA

Name	`OcuHardwareTriggeredDMA`		
Description	This parameter is used to allow the OCU channel to trigger a DMA channel upon compare match The value of the parameter represents the DMA physical channel to trigger. *NOTE: Minimum value is set as default value.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 127		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	GtmTimerClockSelect, OcuNotification		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.8 OcuMaxCounterValue

Table 15 Specification for OcuMaxCounterValue

Name	`OcuMaxCounterValue`		
Description	Maximum value in ticks, the counter of the OCU channel is able to count. For ATOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 24-bit range. For TOM channel, OcuDefaultThreshold scaled up by amount of ticks should be in 16-bit range. *NOTE: Minimum value is set as default value.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 16777215		
Default value	1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	GtmTimerClockSelect, GtmTimerUsed, OcuChannelTickDuration		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.9 OcuNotification

Table 16 Specification for OcuNotification

Name	`OcuNotification`		
Description	The parameter is used by the OCU driver to invoke the user-defined function if the Compare match is detected. The parameter can be configured as a name or an address(numeric value) of the notification function. *Note1: By default, the notification parameter will be NULL , to remove dependency from user defined functions.* *Note2: The OCU driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.*		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	OcuNotificationSupported, OcuHardwareTriggeredDMA		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.10 OcuOuptutPinUsed

Table 17 Specification for OcuOuptutPinUsed

Name	`OcuOuptutPinUsed`		
Description	Information about the usage of an output pin on this channel. True: the channel uses an output pin. False: the channel does not use an output pin. *NOTE: As the default HW used by the channel is set to TOM, and TOM cannot support any pin functionality, the default value is set to false.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 17 (continued) Specification for OcuOuptutPinUsed

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	GtmTimerPortPinSelect, GtmTimerClockSelect		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.11 OcuOutputPinDefaultState

Table 18 Specification for OcuOutputPinDefaultState

Name	`OcuOutputPinDefaultState`		
Description	The parameter OcuOutputPinDefaultState represents the state that a pin associated with a channel shall be set to after initialization. *NOTE: OCU_LOW is set as default value as it represents the minimum numeric value.*		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	OCU_HIGH: The OCU channel output pin will be set to high (3 or 5 V) when requested. OCU_LOW: The OCU channel output pin will be set to low (0V) when requested.		
Default value	OCU_LOW		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	OcuOuptutPinUsed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4 Container: OcuConfigSet

This container is the base of a Configuration Set, which contains the configured OCU channels. This way, different configuration sets can be defined for post-build process.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Post-Build

1.3.1.4.1 OcuCountdirection

Table 19 Specification for OcuCountdirection

Name	`OcuCountdirection`

(table continues...)

Table 19 (continued) Specification for OcuCountdirection

Description	This parameter indicates the count direction for the whole OCU driver. The parameter is non-editable and always configured as "OCU_UPCOUNTING".		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	OCU_UPCOUNTING: The OCU counter will reckon from the minimum to the maximum value.		
Default value	OCU_UPCOUNTING		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5 Container: OcuConfigurationOfOptionalApis

Configuration of optional APIs

NOTE: By default all the optional API's except InitCheck will not be available to optimise the executable size. InitCheck will be available as Safety error reporting is enabled by default.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 OcuDeInitApi

Table 20 Specification for OcuDeInitApi

Name	`OcuDeInitApi`		
Description	Adds / removes the service Ocu_DeInit() from the code.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 20	(continued) Specification for OcuDeInitApi
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.5.2 OcuGetCounterApi

Table 21	Specification for OcuGetCounterApi		
Name	`OcuGetCounterApi`		
Description	Adds / removes the service Ocu_GetCounter() from the code.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.3 OcuInitCheckApi

Table 22	Specification for OcuInitCheckApi		
Name	`OcuInitCheckApi`		
Description	Adds / removes the service Ocu_InitCheck() from the code.		
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.		

1.3.1.5.4 OcuNotificationSupported

Table 23 Specification for OcuNotificationSupported

Name	`OcuNotificationSupported`		
Description	Adds / removes the services Ocu_EnableNotification() and Ocu_DisableNotification() from the code.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.5 OcuSetAbsoluteThresholdApi

Table 24 Specification for OcuSetAbsoluteThresholdApi

Name	`OcuSetAbsoluteThresholdApi`		
Description	Adds / removes the service Ocu_SetAbsoluteThreshold() from the code.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.6 OcuSetPinActionApi

Table 25 Specification for OcuSetPinActionApi

Name	`OcuSetPinActionApi`		
Description	Adds / removes the service Ocu_SetPinAction() from the code.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.7 OcuSetPinStateApi

Table 26 Specification for OcuSetPinStateApi

Name	`OcuSetPinStateApi`		
Description	Adds / removes the service Ocu_SetPinState() from the code.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.8 OcuSetRelativeThresholdApi

Table 27 Specification for OcuSetRelativeThresholdApi

Name	`OcuSetRelativeThresholdApi`		
Description	Adds / removes the service Ocu_SetRelativeThreshold() from the code.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.9 OcuVersionInfoApi

Table 28 Specification for OcuVersionInfoApi

Name	`OcuVersionInfoApi`		
Description	Switch to indicate that the Ocu_GetVersionInfo() is supported.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6 Container: OcuGeneral

This container contains the module-wide configuration parameters of the OCU Driver.

Note: By default all the error reporting (Development, Safety and Multi-core) are enabled, to ensure proper driver functionality.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.6.1 OcuDevErrorDetect

Table 29 Specification for OcuDevErrorDetect

Name	`OcuDevErrorDetect`		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. true: enabled (ON) false: disabled (OFF)		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.2 OcuEcucPartitionRef

Table 30 Specification for OcuEcucPartitionRef

Name	`OcuEcucPartitionRef`		
Description	Maps the OCU driver to zero or multiple ECUC partitions to make the driver API available in the according partition. *Note: Parameter support is added only for AUTOSAR schema compliance, this parameter is not used in code generation logic, hence this parameter is made editable false.*		
Multiplicity	0..*	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE

(table continues...)

Table 30 (continued) Specification for OcuEcucPartitionRef

Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.6.3 OcuKernelEcucPartitionRef

Table 31 Specification for OcuKernelEcucPartitionRef

Name	`OcuKernelEcucPartitionRef`		
Description	Maps the OCU kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the OCU driver is mapped to. *Note: Parameter support is added only for AUTOSAR schema compliance, this parameter is not used in code generation logic, hence this parameter is made editable false.*		
Multiplicity	0..1	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.6.4 OcuMultiCoreErrorDetect

Table 32 Specification for OcuMultiCoreErrorDetect

Name	`OcuMultiCoreErrorDetect`		
Description	The parameter enables or disables the multi core related default error tracer (DET) detection and reporting. It is applicable only when DETs are enabled.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

(table continues...)

Table 32 (continued) Specification for OcuMultiCoreErrorDetect

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

1.3.1.6.5 OcuSafetyEnable

Table 33 Specification for OcuSafetyEnable

Name	`OcuSafetyEnable`		
Description	Pre-processor switch for enabling the safety features of OCU driver.		
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.		

1.3.1.7 Container: OcuGroup

This container contains the parameters for configuring an OCU group.

NOTE: The container is not supported. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.7.1 OcuGroupDefinition

Table 34 Specification for OcuGroupDefinition

Name	`OcuGroupDefinition`		
Description	Assignment of OcuChannels to an OcuGroup.		
Multiplicity	1..*	**Type**	EcucReferenceDef
Range	Reference to Node: OcuChannel		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.7.2 OcuGroupId

Table 35 Specification for OcuGroupId

Name	`OcuGroupId`		
Description	Numeric ID of the group. This parameter is the symbolic name of the group.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.8 Container: OcuHWSpecificSettings

This container contains Ocu-specific parameters for selecting the clock source and setting optional prescalers.

NOTE: The container is not supported. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.8.1 OcuClockSource

Table 36 Specification for OcuClockSource

Name	`OcuClockSource`		
Description	The OCU driver specific clock input for the unit can statically be configured to select different clock sources. *NOTE: The parameter is not supported as the clock source selection will be configured using MCU. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.*		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	None		
Default value	UNUSED		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.2 OcuPrescale

Table 37 Specification for OcuPrescale

Name	`OcuPrescale`		
Description	Optional OCU driver specific clock prescale factor. *NOTE: The parameter is not supported as the clock prescale factor will be configured using MCU. But the parameter will be maintained nonetheless to maintain AUTOSAR compatibility.*		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	None		
Default value	UNUSED		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9 Container: CommonPublishedInformation

Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.9.1 ArMajorVersion

Table 38 Specification for ArMajorVersion

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

1.3.1.9.2 ArMinorVersion

Table 39 Specification for ArMinorVersion

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

1.3.1.9.3 ArPatchVersion

Table 40 Specification for ArPatchVersion

Name	`ArPatchVersion`		
Description	AUTOSAR patch version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the AUTOSAR version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.4 ModuleId

Table 41 Specification for ModuleId

Name	`ModuleId`		
Description	Parameter to provide the module identifier.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	125		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.5 Release

Table 42 Specification for Release

Name	`Release`
Description	Aurix derivative used for the implementation.

(table continues...)

Table 42 (continued) Specification for Release

Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	As per the configuration.		
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.		

1.3.1.9.6 SwMajorVersion

Table 43 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	Module major version.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.7 SwMinorVersion

Table 44 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	Module minor version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		

(table continues...)

Table 44 (continued) Specification for SwMinorVersion

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

1.3.1.9.8 SwPatchVersion

Table 45 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	Module patch version.		
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	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.9 Vendor ID

Table 46 Specification for Vendor ID

Name	`Vendor ID`		
Description	None		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 46 (continued) Specification for Vendor ID

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

1.3.2 Functions - Type definitions

This section list all the data types of the Ocu driver.

1.3.2.1 Ocu_ChannelType

Table 47 Specification for Ocu_ChannelType

Syntax	`Ocu_ChannelType`	
Type	uint8	
File	`Ocu.h`	
Range	0-[{maximum TOM channels}+{maximum ATOM channels}]	Maximum value depends on the device.
Description	Numeric identifier of an OCU channel. As the maximum number for HW channels available is 192(96 TOM and 96 ATOM) the data type chosen is 8-bit (maximum possible of 255).	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.2 Ocu_ValueType

Table 48 Specification for Ocu_ValueType

Syntax	`Ocu_ValueType`	
Type	uint32	
File	`Ocu.h`	
Range	0-16777215	To support both TOM and ATOM, data type is chosen as a 32-bit variable.
Description	Type for reading the counter and writing the threshold values (in number of ticks). ATOM channels use 24-bit wide compare registers are TOM channels use 16-bit wide compare registers. Hence to support both 32-bit data type is chosen.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.3 Ocu_PinStateType

Table 49 Specification for Ocu_PinStateType

Syntax	`Ocu_PinStateType`	
Type	Enumeration	
File	`Ocu.h`	
Range	0 - OCU_LOW	The pin associated to an OCU channel is in low state.
	1 - OCU_HIGH	The pin associated to an OCU channel is in high state.
Description	Output state of the pin linked to an OCU channel.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.4 Ocu_PinActionType

Table 50 Specification for Ocu_PinActionType

Syntax	`Ocu_PinActionType`	
Type	Enumeration	
File	`Ocu.h`	
Range	0 - OCU_DISABLE	The channel pin will remain at its current level upon compare match.
	1 - OCU_SET_HIGH	The channel pin will be set HIGH upon compare match.
	2 - OCU_SET_LOW	The channel pin will be set LOW upon compare match.
	3 - OCU_TOGGLE	The channel pin will be set to the opposite of its current level upon compare match.
Description	Automatic action (by hardware) to be performed on a pin attached to an OCU channel.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.5 Ocu_ConfigType

Table 51 Specification for Ocu_ConfigType

Syntax	`Ocu_ConfigType`
Type	Structure
File	`Ocu.h`

(table continues...)

Table 51 (continued) Specification for Ocu_ConfigType

Range	Hardware dependent[]	The contents of the initialization data structure are hardware specific.
Description	Defines the type of data structure containing the set of configuration parameters required for initializing the OCU driver.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.6 Ocu_ReturnType

Table 52 Specification for Ocu_ReturnType

Syntax	`Ocu_ReturnType`	
Type	Enumeration	
File	`Ocu.h`	
Range	0 - OCU_CM_OUT_REF_INTERVAL	The compare match will not occur inside the current Reference Interval.
	1 - OCU_CM_IN_REF_INTERVAL	The compare match will occur inside the current Reference Interval.
Description	Return information after setting a new threshold value.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.7 Ocu_NotifiPtrType

Table 53 Specification for Ocu_NotifiPtrType

Syntax	`Ocu_NotifiPtrType`
Type	Pointer to a function of type void Function_Name (void)
File	`Ocu.h`
Description	Channel notification function pointer
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 Ocu driver.

1.3.3.1 Ocu_Init

Table 54 Specification for `Ocu_Init` API

Syntax	`void Ocu_Init ( const Ocu_ConfigType * const ConfigPtr )`	
Service ID	0x00	
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 configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to initialize all relevant registers configured hardware (AssignedHWUnit) with the values of structure referenced by the parameter ConfigPtr. The API will also disable all notifications and the OCU channel status will be set to OCU_STOPPED. For multicore, the function will initialize those channels allocated to the core in which this function is invoked. Additionally for master core, the function will initialize the resources which are shared among cores.	
Source	AUTOSAR	
Error handling	OCU_E_ALREADY_INITIALIZED, OCU_E_INIT_FAILED, OCU_E_CORE_NOT_CONFIGURED	
Configuration dependencies	-	
User hints	-	

(table continues...)

Table 54 (continued) Specification for `Ocu_Init` API

SFR accessed	CPU_CORE_ID(r), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(w), GTM_ATOM_CH_CTRL(w), GTM_ATOM_CH_IRQ_EN(w), GTM_ATOM_CH_IRQ_MODE(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_ATOM_CH_SOMC(rw), GTM_ATOM_CH_SR0(w), GTM_ATOM_CH_SR1(w), GTM_TOM_CH_CM0(w), GTM_TOM_CH_CM1(w), GTM_TOM_CH_CN0(w), GTM_TOM_CH_CTRL(w), GTM_TOM_CH_IRQ_EN(w), GTM_TOM_CH_IRQ_MODE(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_SR0(w), GTM_TOM_CH_SR1(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(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.

1.3.3.2 Ocu_DeInit

Table 55 Specification for `Ocu_DeInit` API

Syntax	`void Ocu_DeInit ( void )`	
Service ID	0x01	
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	void	-

(table continues...)

Table 55 (continued) Specification for `Ocu_DeInit` API

Description	The purpose of this API is to de-initialize the OCU driver. The used peripherals/registers will be set to power-on reset state. The API will disable all used interrupts and notifications. For multicore, the function will de-initialize those channels allocated to the core in which the function is invoked.
Source	AUTOSAR
Error handling	OCU_E_UNINIT, OCU_E_PARAM_INVALID_STATE
Configuration dependencies	OcuDeInitApi
User hints	-
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_AGC_ENDIS_CTRL(w), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(w), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(w), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(w), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(w), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(w), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(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.

1.3.3.3 Ocu_EnableNotification

Table 56 Specification for `Ocu_EnableNotification` API

Syntax	`void Ocu_EnableNotification ( const Ocu_ChannelType ChannelNumber )`	
Service ID	0x0B	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
Parameters (out)	-	-
Parameters (in - out)	-	-

(table continues...)

Table 56 (continued) Specification for `Ocu_EnableNotification` API

Return	void	-
Description	The purpose of the API is to enable notifications from an OCU channel. For multicore, the OCU channel shall be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	OCU_E_UNINIT, OCU_E_PARAM_INVALID_CHANNEL, OCU_E_NO_VALID_NOTIF, OCU_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	OcuNotificationSupported	
User hints	None	
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_IRQ_EN(rw), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_EN(rw), GTM_TOM_CH_IRQ_NOTIFY(w) *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 Ocu_DisableNotification

Table 57 Specification for `Ocu_DisableNotification` API

Syntax	`void Ocu_DisableNotification ( const Ocu_ChannelType ChannelNumber )`	
Service ID	0x0A	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to disable notifications from an OCU channel. For multicore, the OCU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	

(table continues...)

Table 57 (continued) Specification for `Ocu_DisableNotification` API

Error handling	OCU_E_PARAM_INVALID_CHANNEL, OCU_E_UNINIT, OCU_E_NO_VALID_NOTIF, OCU_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	OcuNotificationSupported
User hints	None
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_IRQ_EN(rw), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_EN(rw), GTM_TOM_CH_IRQ_NOTIFY(w) *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 Ocu_GetCounter

Table 58 Specification for `Ocu_GetCounter` API

Syntax	`Ocu_ValueType Ocu_GetCounter ( const Ocu_ChannelType ChannelNumber )`	
Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Ocu_ValueType	Current counter value
Description	The purpose of the API is to read the current value of the counter. For multicore, the OCU channel shall be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	OCU_E_PARAM_INVALID_CHANNEL, OCU_E_UNINIT, OCU_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	OcuGetCounterApi	
User hints	None	

(table continues...)

Table 58 (continued) Specification for `Ocu_GetCounter` API

SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_CN0(r), GTM_TBU_CH0_BASE(r), GTM_TBU_CH0_CTRL(r), GTM_TBU_CH1_BASE(r), GTM_TBU_CH2_BASE(r), GTM_TOM_CH_CN0(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 Ocu_SetAbsoluteThreshold

Table 59 Specification for `Ocu_SetAbsoluteThreshold` API

Syntax	`Ocu_ReturnType Ocu_SetAbsoluteThreshold ( const Ocu_ChannelType ChannelNumber, const Ocu_ValueType ReferenceValue, const Ocu_ValueType AbsoluteValue )`	
Service ID	0x07	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber ReferenceValue AbsoluteValue	Numeric identifier of the OCU channel Value given by the upper layer and used as a base to determine whether, writing the threshold value to the compare register was within or outside the reference Interval. Value to compare with the content of the counter. This value is in ticks.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Ocu_ReturnType	To indicate whether, writing the threshold value to the compare register was within or outside the reference Interval.
Description	The purpose of the API is to set the value of the channel threshold using an absolute input data. For multicore, the OCU channel should be allocated to the core in which the function is invoked. Decision: Ocu_SetAbsoluteThreshold API will return OCU_CM_OUT_REF_INTERVAL if a DET/ Safety Error is reported. Rationale : As the enumeration value of OCU_CM_OUT_REF_INTERVAL is numerically minimum (ZERO), it is chosen as a default return value if an error is identified.	

(table continues...)

Table 59 (continued) Specification for `Ocu_SetAbsoluteThreshold` API

Source	AUTOSAR
Error handling	OCU_E_UNINIT, OCU_E_PARAM_INVALID_CHANNEL, OCU_E_CORE_CHANNEL_MISMATCH, OCU_E_PARAM_COMPARE_VALUE, OCU_E_PARAM_REF_VALUE
Configuration dependencies	OcuSetAbsoluteThresholdApi
User hints	None
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(r), GTM_ATOM_CH_SOMC(rw), GTM_ATOM_CH_SR0(r), GTM_ATOM_CH_SR1(rw), GTM_TBU_CH0_BASE(r), GTM_TBU_CH0_CTRL(r), GTM_TBU_CH1_BASE(r), GTM_TBU_CH2_BASE(r), GTM_TOM_CH_CM1(w), GTM_TOM_CH_CN0(r), GTM_TOM_CH_SR1(w) *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 Ocu_SetPinAction

Table 60 Specification for `Ocu_SetPinAction` API

Syntax	`void Ocu_SetPinAction ( const Ocu_ChannelType ChannelNumber, const Ocu_PinActionType PinAction )`	
Service ID	0x05	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber PinAction	Numeric identifier of the OCU OCU_SET_LOW, OCU_SET_HIGH, OCU_TOGGLE, OCU_DISABLE
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to indicate to the driver the action to be taken by the hardware upon compare match. The action will be disable, high, low or toggle. For multicore, the OCU channel should be allocated to the core in which the function is invoked.	

(table continues...)

Table 60 (continued) Specification for `Ocu_SetPinAction` API

Source	AUTOSAR
Error handling	OCU_E_PARAM_NO_PIN, OCU_E_PARAM_INVALID_ACTION, OCU_E_PARAM_INVALID_CHANNEL, OCU_E_UNINIT, OCU_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	OcuSetPinActionApi
User hints	None
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_SOMC(rw) *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 Ocu_SetPinState

Table 61 Specification for `Ocu_SetPinState` API

Syntax	`void Ocu_SetPinState ( const Ocu_ChannelType ChannelNumber, const Ocu_PinStateType PinState )`	
Service ID	0x04	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber PinState	Numeric identifier of the OCU State of the pin to set. OCU_HIGH or OCU_LOW
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to set the level of the pin associated to an OCU channel to high or low. For multicore, the OCU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	OCU_E_PARAM_NO_PIN, OCU_E_PARAM_INVALID_CHANNEL, OCU_E_UNINIT, OCU_E_PARAM_INVALID_STATE, OCU_E_CORE_CHANNEL_MISMATCH	

(table continues...)

Table 61 (continued) Specification for `Ocu_SetPinState` API

Configuration dependencies	OcuSetPinStateApi
User hints	None
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_SOMC(rw) *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 Ocu_SetRelativeThreshold

Table 62 Specification for `Ocu_SetRelativeThreshold` API

Syntax	`Ocu_ReturnType  Ocu_SetRelativeThreshold`
	`(`
	`  const Ocu_ChannelType ChannelNumber,`
	`  const Ocu_ValueType RelativeValue`
	`)`	

Service ID	0x08	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber RelativeValue	Numeric identifier of the OCU channel Value to use for computing the new threshold.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Ocu_ReturnType	To indicate whether, writing the threshold value to the compare register was within or outside the reference Interval.
Description	The purpose of the API is to set the value of the channel threshold to the relative current value of the counter. For multicore, the OCU channel should be allocated to the core in which the function is invoked. Decision: Ocu_SetRelativeThreshold API will return OCU_CM_OUT_REF_INTERVAL if a DET/ Safety Error is reported. Rationale : As the enumeration value of OCU_CM_OUT_REF_INTERVAL is numerically minimum (ZERO), it is chosen as a default return value if an error is identified.	
Source	AUTOSAR	

(table continues...)

Table 62 (continued) Specification for `Ocu_SetRelativeThreshold` API

Error handling	OCU_E_PARAM_INVALID_CHANNEL, OCU_E_UNINIT, OCU_E_CORE_CHANNEL_MISMATCH, OCU_E_PARAM_COMPARE_VALUE
Configuration dependencies	OcuSetRelativeThresholdApi
User hints	None
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(r), GTM_ATOM_CH_SOMC(rw), GTM_ATOM_CH_SR0(r), GTM_ATOM_CH_SR1(rw), GTM_TBU_CH0_BASE(r), GTM_TBU_CH0_CTRL(r), GTM_TBU_CH1_BASE(r), GTM_TBU_CH2_BASE(r), GTM_TOM_CH_CM1(w), GTM_TOM_CH_CN0(r), GTM_TOM_CH_SR1(w) *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 Ocu_StartChannel

Table 63 Specification for `Ocu_StartChannel` API

Syntax	`void Ocu_StartChannel ( const Ocu_ChannelType ChannelNumber )`	
Service ID	0x02	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to start the OCU channel. For multicore, the OCU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	OCU_E_BUSY, OCU_E_PARAM_INVALID_CHANNEL, OCU_E_UNINIT, OCU_E_CORE_CHANNEL_MISMATCH	

(table continues...)

Table 63 (continued) Specification for `Ocu_StartChannel` API

Configuration dependencies	-
User hints	None
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(r), GTM_ATOM_CH_IRQ_EN(rw), GTM_ATOM_CH_IRQ_NOTIFY(rw), GTM_ATOM_CH_SOMC(rw), GTM_ATOM_CH_SR0(r), GTM_ATOM_CH_SR1(r), GTM_TBU_CH0_BASE(r), GTM_TBU_CH0_CTRL(r), GTM_TBU_CH1_BASE(r), GTM_TBU_CH2_BASE(r), GTM_TOM_CH_CN0(r), GTM_TOM_CH_IRQ_EN(rw), GTM_TOM_CH_IRQ_NOTIFY(w) *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 Ocu_StopChannel

Table 64 Specification for `Ocu_StopChannel` API

Syntax	`void Ocu_StopChannel ( const Ocu_ChannelType ChannelNumber )`	
Service ID	0x03	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The purpose of the API is to stop the OCU channel. For multicore, the OCU channel should be allocated to the core in which the function is invoked.	
Source	AUTOSAR	
Error handling	OCU_E_PARAM_INVALID_CHANNEL, OCU_E_UNINIT, OCU_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	
User hints	None	

(table continues...)

Table 64 (continued) Specification for `Ocu_StopChannel` API

SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_IRQ_EN(rw), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_ATOM_CH_SOMC(rw), GTM_ATOM_CH_STAT(r), GTM_TOM_CH_IRQ_EN(rw), GTM_TOM_CH_IRQ_NOTIFY(w) *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 Ocu_InitCheck

Table 65 Specification for `Ocu_InitCheck` API

Syntax	`Std_ReturnType  Ocu_InitCheck`
	`(`
	`  const Ocu_ConfigType * const ConfigPtr`
	`)`	

Service ID	0x0C	
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 configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: if initialization comparison is success. E_NOT_OK: In case of - Input configuration pointer is NULL. - ConfigPtr received in InitCheck is not same as used in Init. - Driver is not initialized. - If any of the GTM TOM and ATOM channel is not initialized - SFRs not having expected value.
Description	The purpose of the API is to check the values set during the OCU driver initialization are as per the config structure. The API does not modify any SFR/variable and only a read operation is performed. The API is called after Ocu_Init() is done to check for the correctness of initialization. In case any failure is observed in comparison, the API returns E_NOT_OK.	
Source	IFX	
Error handling	-	

(table continues...)

Table 65 (continued) Specification for `Ocu_InitCheck` API

Configuration dependencies	OcuInitCheckApi
User hints	-
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_AGC_ENDIS_CTRL(r), GTM_ATOM_AGC_ENDIS_STAT(r), GTM_ATOM_AGC_FUPD_CTRL(r), GTM_ATOM_AGC_GLB_CTRL(r), GTM_ATOM_AGC_OUTEN_CTRL(r), GTM_ATOM_AGC_OUTEN_STAT(r), GTM_ATOM_CH_CM0(r), GTM_ATOM_CH_CM1(r), GTM_ATOM_CH_CN0(r), GTM_ATOM_CH_CTRL(r), GTM_ATOM_CH_IRQ_EN(r), GTM_ATOM_CH_IRQ_MODE(r), GTM_ATOM_CH_IRQ_NOTIFY(r), GTM_ATOM_CH_SR0(r), GTM_ATOM_CH_SR1(r), GTM_TOM_CH_CM0(r), GTM_TOM_CH_CM1(r), GTM_TOM_CH_CN0(r), GTM_TOM_CH_CTRL(r), GTM_TOM_CH_IRQ_EN(r), GTM_TOM_CH_IRQ_MODE(r), GTM_TOM_CH_SR0(r), GTM_TOM_CH_SR1(r), GTM_TOM_TGC0_ENDIS_CTRL(r), GTM_TOM_TGC0_ENDIS_STAT(r), GTM_TOM_TGC0_FUPD_CTRL(r), GTM_TOM_TGC0_GLB_CTRL(r), GTM_TOM_TGC0_OUTEN_CTRL(r), GTM_TOM_TGC0_OUTEN_STAT(r), GTM_TOM_TGC1_ENDIS_CTRL(r), GTM_TOM_TGC1_ENDIS_STAT(r), GTM_TOM_TGC1_FUPD_CTRL(r), GTM_TOM_TGC1_GLB_CTRL(r), GTM_TOM_TGC1_OUTEN_CTRL(r), GTM_TOM_TGC1_OUTEN_STAT(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 Ocu_GetVersionInfo

Table 66 Specification for `Ocu_GetVersionInfo` API

Syntax	`void Ocu_GetVersionInfo ( Std_VersionInfoType * const versioninfo )`	
Service ID	0x09	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to store the version information of the module
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 66 (continued) Specification for `Ocu_GetVersionInfo` API

Description	The purpose of the API is to return the version information of the OCU driver. The version information includes: Module ID, Vendor ID., vendor specific version numbers. This function is available only if the OCU_VERSION_INFO_API is ON.
Source	AUTOSAR
Error handling	OCU_E_PARAM_POINTER
Configuration dependencies	OcuVersionInfoApi
User hints	The API can be called before OCU initialization.
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.4 Notifications and Callbacks

This section lists all the notification and callbacks of the Ocu driver.

1.3.4.1 Ocu_Timer_Isr

Table 67 Specification for `Ocu_Timer_Isr` API

Syntax	`void Ocu_Timer_Isr ( const uint32 LogicalChannelId, const uint32 flags )`	
Service ID	0x20	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channels	
Parameters (in)	LogicalChannelId flags	Logical channel identifier. Interrupt flags responsible for ISR
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Callback function from MCU to service timer (GTM-TOM and GTM-ATOM) interrupts.	
Source	IFX	
Error handling	OCU_E_INVALID_ISR_UNINIT, OCU_E_INVALID_ISR_INACTIVE_CHANNEL, OCU_E_INVALID_ISR_COMP_INVALID, OCU_E_INVALID_ISR_CHANNEL_INVALID, OCU_E_INVALID_ISR_CHANNEL_CORE_MISMATCH	

(table continues...)

Table 67 (continued) Specification for `Ocu_Timer_Isr` API	
Configuration dependencies	-
User hints	None
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(r), GTM_ATOM_CH_SOMC(rw), GTM_ATOM_CH_SR0(r), GTM_ATOM_CH_SR1(r), GTM_TBU_CH0_BASE(r), GTM_TBU_CH0_CTRL(r), GTM_TBU_CH1_BASE(r), GTM_TBU_CH2_BASE(r), GTM_TOM_CH_CN0(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.

1.3.5 Scheduled functions

The Ocu driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

The Ocu 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 Ocu driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
OCU_E_UNINIT: Error code is reported when any OCU API services other than Ocu_GetVersionInfo() and Ocu_Init() are used without module initialization.	AUTOSAR	0x02	DET_SAFETY	0x02	DET_SAFETY
OCU_E_PARAM_INVALID_CHANNEL: Error code is reported when OCU API used with an invalid channel identifier.	AUTOSAR	0x03	DET_SAFETY	0x03	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
OCU_E_PARAM_INVALID_STATE: Error code is reported when API Ocu_SetPinState() called with an invalid pin state or when the channel is in the RUNNING state. Also in case, API Ocu_DeInit called when at least one of the OCU channel is in RUNNING state.	AUTOSAR	0x04	DET_SAFETY	0x04	DET_SAFETY
OCU_E_PARAM_INVALID_ACTION: Error code is reported when API Ocu_SetPinAction() called with an invalid pin action.	AUTOSAR	0x05	DET_SAFETY	0x05	DET_SAFETY
OCU_E_NO_VALID_NOTIF: Error code is reported on usage of Ocu_DisableNotification() or Ocu_EnableNotification() on a channel where a NULL pointer is configured as the notification function.	AUTOSAR	0x06	DET_SAFETY	0x06	DET_SAFETY
OCU_E_ALREADY_INITIALIZED: Error code is reported when OCU driver is already initialized and Ocu_Init() is called.	AUTOSAR	0x07	DET_SAFETY	0x07	DET_SAFETY
OCU_E_PARAM_POINTER: Error code is reported when a NULL pointer is passed an input parameter.	AUTOSAR	0x08	DET_SAFETY	0x08	DET_SAFETY
OCU_E_BUSY: Error code is reported when API Ocu_StartChannel() called on a channel that is in state RUNNING.	AUTOSAR	0x09	DET_SAFETY	0x09	DET_SAFETY
OCU_E_PARAM_NO_PIN: Error code is reported when Ocu_SetPinState() or Ocu_SetPinAction() called for a channel that does not have an associated output pin.	AUTOSAR	0x0A	DET_SAFETY	0x0A	DET_SAFETY
OCU_E_INIT_FAILED: Error code is reported when OCU initialization has failed. Example, selected configuration set does not exist.	AUTOSAR	0x0B	DET_SAFETY	0x0B	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
OCU_E_CORE_NOT_CONFIGURED: Error code is reported when OCU module is not configured for the core in which an API is invoked.	IFX	0x64	DET_SAFETY	0x64	DET_SAFETY
OCU_E_CORE_CHANNEL_MISMATCH: Error code is reported when an API is called with the channel not allocated to executing core.	IFX	0x65	DET_SAFETY	0x65	DET_SAFETY
OCU_E_PARAM_COMPARE_VALUE: "OCU_E_PARAM_COMPARE_VALUE" error is reported to indicate invalid input compare values.	IFX	0xC8	SAFETY	0xC8	SAFETY
OCU_E_PARAM_REF_VALUE: If the reference value is greater than the maximum threshold value then function Ocu_SetAbsoluteThreshold shall raise the error "OCU_E_PARAM_REF_VALUE".	IFX	0xC9	SAFETY	0xC9	SAFETY
OCU_E_INVALID_ISR_UNINIT: If the ISR is invoked before OCU_Init then Ocu driver shall report a safety error "OCU_E_INVALID_ISR_UNINIT".	IFX	0xCA	SAFETY	0xCA	SAFETY
OCU_E_INVALID_ISR_INACTIVE_CHANNEL: If the ISR is invoked when the channel is not active then Ocu driver shall report a safety error "OCU_E_INVALID_ISR_INACTIVE_CHANNEL"	IFX	0xCB	SAFETY	0xCB	SAFETY
OCU_E_INVALID_ISR_COMP_INVALID: The Ocu driver shall report a safety error, if ISR is invoked with wrong flags (Compare match notification flag) indicating unexpected compare match.	IFX	0xCC	SAFETY	0xCC	SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
OCU_E_INVALID_ISR_CHANNEL_INVALID: The Ocu driver shall report Safety error "OCU_E_INVALID_ISR_CHANNEL_INVALID", if the ISR is invoked when the passed channel ID is not configured.	IFX	0xCD	SAFETY	0xCD	SAFETY
OCU_E_INVALID_ISR_CHANNEL_CORE_MISMATCH: The Ocu driver shall report DET/Safety error "OCU_E_INVALID_ISR_CHANNEL_CORE_MISMATCH", if the ISR is invoked when the passed channel and core id mismatch.	IFX	0xCE	SAFETY	0xCE	SAFETY

1.3.9 Deviations and limitations

The section describes the deviations and limitations of the Ocu driver.

1.3.9.1 Deviations

The section describes the deviations of the Ocu driver.

1.3.9.1.1 Software specification deviations

The Ocu driver does not have any deviations.

1.3.9.1.2 AMDC Violations

This section describes the violations reported by the vector AMDC checker tool with respect to AUTOSAR.

Table 68 Violations reported by AMDC checker tool for A207

AMDC Rule	A207
Description	Maximum value of parameter 'Ocu/OcuConfigSet/OcuChannel/OcuHardwareTriggeredAdc' in VSMD (383) may not be larger than maximum value defined in StMD (255). [Ocu.bmd]
Justification	The Max number of Adc channels are supported by the hardware TC39x is 383. To provide this feature to user, the autosar parameter range check has been violated. Hence the range of the parameter is extended from 255 to 383. *Note: "OcuHardwareTriggeredAdc" is only for the visual indication to identify which ADC group is configured(there is no functional impact)*

1.3.9.1.3 VSMD Violations

This section describes the violations reported by the EB VSMD checker tool with respect to AUTOSAR.

Table 69 Violations reported by VSMD checker tool for EB03

Rule ID:	EB03
VSMD Node(s):	/AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuHardwareTriggeredAdc /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuHardwareTriggeredDMA /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuNotification /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuOutputPinDefaultState /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuCountdirection /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuHWSpecificSettings /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuHWSpecificSettings/OcuClockSource /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuHWSpecificSettings/OcuPrescale /AURIX2G/EcucDefs/Ocu/OcuGeneral/OcuKernelEcucPartitionRef
Description:	The StMD node has LOWER-MULTIPLICITY=0 and UPPER-MULTIPLICITY=1. The VSMD-node shall get the OPTIONAL-attribute instead of creating a list!
Additional Information:	-

Table 70 Violations reported by VSMD checker tool for EB09

Rule ID:	EB09
VSMD Node(s):	/AURIX2G/EcucDefs/Ocu
Description:	EB specific rule to check consistency of parameter postBuildVariantUsed.
Additional Information:	-

Table 71 Violations reported by VSMD checker tool for EcucSws_1007

Rule ID:	EcucSws_1007
VSMD Node(s):	/AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuHardwareTriggeredAdc
Description:	For Integer and Float Parameters the MIN values must be >= and the MAX values <= as in the StMD.
Additional Information:	-

Table 72 Violations reported by VSMD checker tool for EcucSws_1014

Rule ID:	EcucSws_1014
VSMD Node(s):	/AURIX2G/EcucDefs/Ocu /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis
Description:	Additional vendor specific parameter definitions (using ParameterTypes), container definitions and references shall be added to the VSMD according to the alphabetical order.
Additional Information:	-

Table 73 Violations reported by VSMD checker tool for EcucSws_1035

Rule ID:	EcucSws_1035

(table continues...)

Table 73 (continued) Violations reported by VSMD checker tool for EcucSws_1035

VSMD Node(s):	/AURIX2G/EcucDefs/Ocu /AURIX2G/EcucDefs/Ocu/OcuConfigSet /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuAssignedHardwareChannel /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuChannelEcucPartitionRef /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuChannelId /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuChannelTickDuration /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuDefaultThreshold /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuHardwareTriggeredAdc /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuHardwareTriggeredDMA /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuMaxCounterValue /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuNotification /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuOuptutPinUsed /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuChannel/OcuOutputPinDefaultState /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuCountdirection /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuGroup /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuGroup/OcuGroupDefinition /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuGroup/OcuGroupId /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuHWSpecificSettings /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuHWSpecificSettings/OcuClockSource /AURIX2G/EcucDefs/Ocu/OcuConfigSet/OcuHWSpecificSettings/OcuPrescale /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuDeInitApi /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuGetCounterApi

(table continues...)

Table 73 (continued) Violations reported by VSMD checker tool for EcucSws_1035

	/AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuNotificationSupported /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuSetAbsoluteThresholdApi /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuSetPinActionApi /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuSetPinStateApi /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuSetRelativeThresholdApi /AURIX2G/EcucDefs/Ocu/OcuConfigurationOfOptionalApis/OcuVersionInfoApi /AURIX2G/EcucDefs/Ocu/OcuGeneral /AURIX2G/EcucDefs/Ocu/OcuGeneral/OcuDevErrorDetect /AURIX2G/EcucDefs/Ocu/OcuGeneral/OcuEcucPartitionRef /AURIX2G/EcucDefs/Ocu/OcuGeneral/OcuKernelEcucPartitionRef
Description:	For Containers, Parameters and References elements UUID must be unique (also between StMD and VSMD).
Additional Information:	-

Table 74 Violations reported by VSMD checker tool for EcucSws_2101

Rule ID:	EcucSws_2101
VSMD Node(s):	/AURIX2G/EcucDefs/Ocu/POST_BUILD_VARIANT_USED
Description:	For each ConfigurationVariant supported by the ModuleDef, there must be one ImplementationConfigClass element. In VSMD, the ImplementationConfigClass is mandatory.
Additional Information:	-

Table 75 Violations reported by VSMD checker tool for EcucSws_6003

Rule ID:	EcucSws_6003
VSMD Node(s):	/AURIX2G/EcucDefs/Ocu
Description:	The SHORT-NAME of the AR-PACKAGEs of StMD and VSMD must be different to ensure a unique SHORT-NAME-path.
Additional Information:	-

Table 76 Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41

Rule ID:	TpsEcuc_06051_ASR41
VSMD Node(s):	/AURIX2G/EcucDefs/Ocu/ POST_BUILD_VARIANT_USED
Description:	The implementationConfigClass of an EcucParameterDef or EcucAbstractReferenceDef in VSMD shall be the same or higher (where PreCompile configuration class is considered to be the lowest and PostBuild the highest) as in StMD with respect to the selected subset defined by the actually implemented supportedConfigVariant.
Additional Information:	-

1.3.9.2 Limitations

The section describes the limitations of the Ocu driver.

Table 77 Known limitations

Reference	Limitation
`Ocu_EnableNotification`, `Ocu_DisableNotification`	If an OCU channel is configured to trigger a DMA channel, notifications cannot be issued by the OCU channel.
`Ocu_SetPinState`	Unintended ADC triggering might be issued when the `Ocu_SetPinState` API is invoked on a channel which is configured for both ADC triggering and pin.
`Overloading of DMA channels`	When DMA channels are overloaded then there can be a delay of receiving the DMA interrupt, which could further lead to missing of DMA interrupts. In this corner case the driver which is using this DMA channel may lose data or get stuck.

Revision history

Table 78 Revision History

Date	Version	Description
2024-08-09	5.0	Document is released
2024-08-05	4.1	- Parameters and Syntax updated in section 1.3.4.1 Ocu_Timer_Isr function. - Limitation updated for overloading of DMA channels in section 1.3.9.2
2023-06-08	4.0	Document is released
2023-05-25	3.1	- ASIL level field changed to Safety level with description as "refer to release notes" for all APIs under 1.3.3 Functions - APIs and 1.3.4 Notifications and Callbacks. - In Ocu_InitCheck function, E_NOT_OK description updated to list all the conditions when function returns E_NOT_OK.
2021-11-11	3.0	Document is released
2021-11-09	2.1	- Config variant attribute table information is removed and added this information in 'Configuration interfaces' section - Updated SFR information in Ocu_InitCheck API
2020-11-18	2.0	Document is released
2020-11-17	1.1	- Updated HSI of APIs in section 1.3.3
2020-08-13	1.0	Document is released
2020-08-10	0.1	- Initial Version - Ocu driver chapter moved from MC-ISAR_TC3xx_UM_Basic to this document - Ocu_InitCheck API's Re-entrancy field - Added AMDC and VSMD violations

Trademarks

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

Edition 2024-08-09

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

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.