MCAL Configuration Verification Manual for Ocu

32-bit TriCore™ AURIX™ TC3xx microcontroller family

About this document

Scope and purpose

This Configuration Data Reference document is applicable to all TC3xx devices in the TriCore™ AURIX™ family of 32-bit microcontrollers.

The purpose of this document is to facilitate the integrator to verify the generated code based on the input configuration parameters. This document describes details of structures, defines, macros and variables generated from the configuration parameters.

Intended audience

This document is intended for integrators who need to understand the logic of the generated configuration code of AURIX™ AUTOSAR MCAL.

Reference documents

This document should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual Ocu

Table of contents

Table of contents

Ocu Driver

This chapter describes the details of the configuration data generated from the OCU driver.

1.1 File: Ocu_Cfg.h

The generated header file contains all pre-compile configuration parameters. Pre-compile time configuration allows decoupling of the static configuration from implementation. The file is generated in 'inc' folder.

1.1.1 Macro: OCU_AR_RELEASE_MAJOR_VERSION

Table 1 OCU_AR_RELEASE_MAJOR_VERSION

Name	OCU_AR_RELEASE_MAJOR_VERSION	
Description	Major version number of AUTOSAR release on which the Ocu implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Ocu_Cfg.h file with ArMajorVersion 4	`#define OCU_AR_RELEASE_MAJOR_VERSION (4U)`

1.1.2 Macro: OCU_AR_RELEASE_MINOR_VERSION

Table 2 OCU_AR_RELEASE_MINOR_VERSION

Name	OCU_AR_RELEASE_MINOR_VERSION	
Description	Minor version number of AUTOSAR release on which the Ocu implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Ocu_Cfg.h file with ArMinorVersion 2	`#define OCU_AR_RELEASE_MINOR_VERSION (2U)`

1.1.3 Macro: OCU_AR_RELEASE_REVISION_VERSION

Table 3 OCU_AR_RELEASE_REVISION_VERSION

Name	OCU_AR_RELEASE_REVISION_VERSION
Description	Revision version number of AUTOSAR release on which the Ocu implementation is based on.

Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArPatchVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Ocu_Cfg.h file with ArPatchVersion 2	`#define OCU_AR_RELEASE_REVISION_VERSION (2U)`

1.1.4 Macro: OCU_SW_MAJOR_VERSION

Table 4 OCU_SW_MAJOR_VERSION

Name	OCU_SW_MAJOR_VERSION	
Description	Major version number of the OCU module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate OCU_Cfg.h file with SwMajorVersion 10	`#define OCU_SW_MAJOR_VERSION (10U)`

1.1.5 Macro: OCU_SW_MINOR_VERSION

Table 5 OCU_SW_MINOR_VERSION

Name	OCU_SW_MINOR_VERSION	
Description	Minor version number of the OCU module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate OCU_Cfg.h file with SwMinorVersion 10	`#define OCU_SW_MINOR_VERSION (10U)`

1.1.6 Macro: OCU_SW_PATCH_VERSION

Table 6 OCU_SW_PATCH_VERSION

Name	OCU_SW_PATCH_VERSION
Description	Patch level version number of the OCU module.
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwPatchVersion'.

	Note: *The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate OCU_Cfg.h file with SwPatchVersion 0	`#define OCU_SW_PATCH_VERSION (0U)`

1.1.7 Macro: OCU_DEV_ERROR_DETECT

Table 7 OCU_DEV_ERROR_DETECT

Name	OCU_DEV_ERROR_DETECT	
Description	Enables/disables the Development Error Detection.	
Verification method	The macro is generated as STD_ON if OcuDevErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuDevErrorDetect = True	`#define OCU_DEV_ERROR_DETECT (STD_ON)`
	OcuDevErrorDetect = False	`#define OCU_DEV_ERROR_DETECT (STD_OFF)`

1.1.8 Macro: OCU_MULTICORE_ERROR_DETECT

Table 8 OCU_MULTICORE_ERROR_DETECT

Name	OCU_MULTICORE_ERROR_DETECT	
Description	Enables/disables MultiCore DET Check	
Verification method	The macro is generated as STD_ON if OcuMultiCoreErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuMultiCoreErrorDetect = True	`#define OCU_MULTICORE_ERROR_DETECT (STD_ON)`
	OcuMultiCoreErrorDetect = False	`#define OCU_MULTICORE_ERROR_DETECT (STD_OFF)`

1.1.9 Macro: OCU_SAFETY_ENABLE

Table 9 OCU_SAFETY_ENABLE

Name	OCU_SAFETY_ENABLE
Description	Enables/disables safety features
Verification method	The macro is generated as STD_ON if OcuSafetyEnable configuration parameter is set to 'True' else the macro is generated as STD_OFF.

Example(s)	Action	Generated output
	OcuSafetyEnable = True	`#define OCU_SAFETY_ENABLE (STD_ON)`
	OcuSafetyEnable = False	`#define OCU_SAFETY_ENABLE (STD_OFF)`

1.1.10 Macro: OCU_INITCHECK_API

Table 10 OCU_INITCHECK_API

Name	OCU_INITCHECK_API	
Description	Enables/disables OCU_InitCheck API	
Verification method	The macro is generated as STD_ON if OcuInitCheckApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuInitCheckApi = True	`#define OCU_INITCHECK_API (STD_ON)`
	OcuInitCheckApi = False	`#define OCU_INITCHECK_API (STD_OFF)`

1.1.11 Macro: OCU_DE_INIT_API

Table 11 OCU_DE_INIT_API

Name	OCU_DE_INIT_API	
Description	Enables/disables OCU_DeInit API.	
Verification method	The macro is generated as STD_ON if OcuDeInitApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuDeInitApi = True	`#define OCU_DE_INIT_API (STD_ON)`
	OcuDeInitApi = False	`#define OCU_DE_INIT_API (STD_OFF)`

1.1.12 Macro: OCU_SET_PIN_ACTION_API

Table 12 OCU_SET_PIN_ACTION_API

Name	OCU_SET_PIN_ACTION_API	
Description	Enables/disables Ocu_SetPinAction API	
Verification method	The macro is generated as STD_ON if OcuSetPinActionApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuSetPinActionApi = True	`#define OCU_SET_PIN_ACTION_API (STD_ON)`
	OcuSetPinActionApi = False	`#define OCU_SET_PIN_ACTION_API (STD_OFF)`

1.1.13 Macro: OCU_SET_PIN_STATE_API

Table 13 OCU_SET_PIN_STATE_API

Name	OCU_SET_PIN_STATE_API	
Description	Enables/disables Ocu_SetPinState API	
Verification method	The macro is generated as STD_ON if OcuSetPinStateApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuSetPinStateApi = True	`#define OCU_SET_PIN_STATE_API (STD_ON)`
	OcuSetPinStateApi = False	`#define OCU_SET_PIN_STATE_API (STD_OFF)`

1.1.14 Macro: OCU_GET_COUNTER_API

Table 14 OCU_GET_COUNTER_API

Name	OCU_GET_COUNTER_API	
Description	Enables/disables Ocu_GetCounter API	
Verification method	The macro is generated as STD_ON if OcuGetCounterApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuGetCounterApi = True	`#define OCU_GET_COUNTER_API (STD_ON)`
	OcuGetCounterApi = False	`#define OCU_GET_COUNTER_API (STD_OFF)`

1.1.15 Macro: OCU_SET_ABSOLUTE_THRESHOLD_API

Table 15 OCU_SET_ABSOLUTE_THRESHOLD_API

Name	OCU_SET_ABSOLUTE_THRESHOLD_API	
Description	Enables/disables Ocu_SetAbsoluteThreshold API	
Verification method	The macro is generated as STD_ON if OcuSetAbsoluteThresholdApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuSetAbsoluteThresholdApi = True	`#define OCU_SET_ABSOLUTE_THRESHOLD_API (STD_ON)`
	OcuSetAbsoluteThresholdApi = False	`#define OCU_SET_ABSOLUTE_THRESHOLD_API (STD_OFF)`

1.1.16 Macro: OCU_SET_RELATIVE_THRESHOLD_API

Table 16 OCU_SET_RELATIVE_THRESHOLD_API

Name	OCU_SET_RELATIVE_THRESHOLD_API	
Description	Enables/disables Ocu_SetRelativeThreshold API	
Verification method	The macro is generated as STD_ON if OcuSetRelativeThresholdApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuSetRelativeThresholdApi = True	`#define OCU_SET_RELATIVE_THRESHOLD_API (STD_ON)`
	OcuSetRelativeThresholdApi = False	`#define OCU_SET_RELATIVE_THRESHOLD_API (STD_OFF)`

1.1.17 Macro: OCU_GET_VERSION_INFO_API

Table 17 OCU_GET_VERSION_INFO_API

Name	OCU_GET_VERSION_INFO_API	
Description	Enables/disables OCU_GetVersionInfo API	
Verification method	The macro is generated as STD_ON if OcuGetVersionInfoApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuGetVersionInfoApi = True	`#define OCU_GET_VERSION_INFO_API (STD_ON)`
	OcuGetVersionInfoApi = False	`#define OCU_GET_VERSION_INFO_API (STD_OFF)`

1.1.18 Macro: OCU_NOTIFICATION_SUPPORTED

Table 18 OCU_NOTIFICATION_SUPPORTED

Name	OCU_NOTIFICATION_SUPPORTED	
Description	Enables/disables Ocu_EnableNotification and Ocu_DisableNotification API	
Verification method	The macro is generated as STD_ON if OcuNotificationSupported configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	OcuNotificationSupported = True	`#define OCU_NOTIFICATION_SUPPORTED (STD_ON)`
	OcuNotificationSupported = False	`#define OCU_NOTIFICATION_SUPPORTED (STD_OFF)`

1.1.19 Macro: OcuConf_OcuChannel_<channel name>

Table 19 OcuConf_OcuChannel_<channel name>

Name	OcuConf_OcuChannel_<channel name>	
Description	The macro is the symbolic name generated for the configuration parameter 'OcuConfigSet/OcuChannel/OcuChannelId'	
Verification method	The macro is generated as a numeric value which is configured in 'OcuConfigSet/OcuChannel/OcuChannelId'. <channel name> is the name of the OCU channel's conainer name.	
Example(s)	**Action**	**Generated output**

	• Configure 4 OCU channels.
	• Container for Channel ID 0 is named FuelInjectionOutput.
	• Container for Channel ID 1 is named FuelIgnitionOutput.
	• Container for Channel ID 2 is named SolenoidValveOutput
	• Container for Channel ID 3 is named KnockWindowOutput
		`#define OcuConf_OcuChannel_FuelInjectionOutput (0U)`
		`#define OcuConf_OcuChannel_FuelIgnitionOutput (1U)`
		`#define OcuConf_OcuChannel_SolenoidValveOutput (2U)`
		`#define OcuConf_OcuChannel_KnockWindowOutput (3U)`

1.1.20 Macro: OCU_MAX_CHANNELS_CORE<x>

Table 20 OCU_MAX_CHANNELS_CORE<x>

Name	OCU_MAX_CHANNELS_CORE<x> (x ranges from 0 to 5)	
Description	Indicates the total number of channels configured for CORE<x>.	
Verification method	The macro is generated as total number of channels allocated to CORE<x>. *Note: Channels not assigned to any core are assigned to master core (ResourceMMasterCore).*	
Example(s)	**Action**	**Generated output**

	• Configure 4 OCU channels (OcuChannel_0 to OcuChannel_3).
	• Set ResourceMMasterCore as CORE1.
	• Do not assign OCU channels in any ResourceMAllocation
		`#define OCU_MAX_CHANNELS_CORE0 (0U)`
		`#define OCU_MAX_CHANNELS_CORE1 (4U)`
		`#define OCU_MAX_CHANNELS_CORE2 (0U)`
		`#define OCU_MAX_CHANNELS_CORE3 (0U)`
		`#define OCU_MAX_CHANNELS_CORE4 (0U)`
		`#define OCU_MAX_CHANNELS_CORE5 (0U)`

	• Configure 9 OCU channels (OcuChannel_0 to OcuChannel_8).	`#define OCU_MAX_CHANNELS_CORE0 (3U)`
		`#define OCU_MAX_CHANNELS_CORE1 (0U)`
		`#define OCU_MAX_CHANNELS_CORE2 (6U)`

	• Set ResourceMMasterCore as CORE2.
	• Assign OcuChannel_0, OcuChannel_3 and OcuChannel_7 under ResourceMAllocation with ResourceMCoreID as CORE0.
	• Assign OcuChannel_1, OcuChannel_2 and OcuChannel_8 under ResourceMAllocation with ResourceMCoreID as CORE2
		`#define OCU_MAX_CHANNELS_CORE3 (0U)`
		`#define OCU_MAX_CHANNELS_CORE4 (0U)`
		`#define OCU_MAX_CHANNELS_CORE5 (0U)`

	• Configure 4 OCU channels (OcuChannel_0 to OcuChannel_3).
	• Assign all the channels under ResourceMAllocation with ResourceMCoreID as CORE0
		`#define OCU_MAX_CHANNELS_CORE0 (4U)`
		`#define OCU_MAX_CHANNELS_CORE1 (0U)`
		`#define OCU_MAX_CHANNELS_CORE2 (0U)`
		`#define OCU_MAX_CHANNELS_CORE3 (0U)`
		`#define OCU_MAX_CHANNELS_CORE4 (0U)`
		`#define OCU_MAX_CHANNELS_CORE5 (0U)`

	• Configure 9 OCU channels (OcuChannel_0 to OcuChannel_8).
	• ResourceMMasterCore is CORE4.
	• Assign OcuChannel_0, OcuChannel_3 and OcuChannel_7 under ResourceMAllocation with ResourceMCoreID as CORE3
		`#define OCU_MAX_CHANNELS_CORE0 (0U)`
		`#define OCU_MAX_CHANNELS_CORE1 (0U)`
		`#define OCU_MAX_CHANNELS_CORE2 (0U)`
		`#define OCU_MAX_CHANNELS_CORE3 (3U)`
		`#define OCU_MAX_CHANNELS_CORE4 (6U)`
		`#define OCU_MAX_CHANNELS_CORE5 (0U)`

1.1.21 Macro: OCU_MAX_CHANNELS

Table 21 OCU_MAX_CHANNELS

Name	OCU_MAX_CHANNELS	
Description	Indicates the total number of channels configured.	
Verification method	The macro is generated as a numeric value which corresponds to the number of elements in the list 'OcuConfigSet/OcuChannel'.	
Example(s)	**Action**	**Generated output**
	Configure 4 OCU channels (Ocu_Channel0 to Ocu_Channel3)	`#define OCU_MAX_CHANNELS (4U)`
	Configure 9 OCU channels (Ocu_Channel0 to Ocu_Channel8)	`#define OCU_MAX_CHANNELS (9U)`

1.1.22 Macro: OCU_SINGLE_CORE

Table 22 OCU_SINGLE_CORE

Name	OCU_SINGLE_CORE	
Description	Enables/disables multi-core features	
Verification method	The macro is generated as STD_ON if all the channels are configured to a single core else the macro is generated as STD_OFF. *Note: Channels not assigned to any core are assigned to master core (ResourceMMasterCore).*	
Example(s)	**Action**	**Generated output**
	• Configure 4 OCU channels. • Do not configure any channel to any core.	`#define OCU_SINGLE_CORE (STD_ON)`
	• Configure 4 OCU channels. • Configure Channel0 to non-master core.	`#define OCU_SINGLE_CORE (STD_OFF)`

1.1.23 Macro: OCU_SINGLE_CORE_ID

Table 23 OCU_SINGLE_CORE_ID

Name	OCU_SINGLE_CORE_ID	
Description	Core ID of the core in case of a single core configuration.	
Verification method	The macro is generated as a numeric value which denotes the Core ID to which all the OCU channel are allocated. If OCU channels are split between multiple cores, the macro is generated as 0.	
Example(s)	**Action**	**Generated output**
	• Configure 4 OCU channels. • Do not configure any channel to any core. • Master Core is 4	`#define OCU_SINGLE_CORE_ID (4)`
	• Configure 4 OCU channels. • Configure all channels to Core 2. • Master Core is 1	`#define OCU_SINGLE_CORE_ID (2)`
	• Configure 4 OCU channels. • Configure Channel0 to non-master core.	`#define OCU_SINGLE_CORE_ID (0)`

1.2 File: OCU[_<variant>]_PBcfg.c

The generated source file contains all post-build configuration parameters. Post-build time configuration mechanism allows configurable functionality of OCU driver that is deployed as object code. The file is generated in 'src' folder.

1.2.1 Callback function declaration

Callback function declaration

Name	<User configured callback function name>	
Type	Ocu_NotifiPtrType	
Description	Decalration of the notification callback function configured by the user for a notification capable OCU channel. *Note: The declaration is not generated if the user configures NULL or the address of the callback function instead of function name.*	
Verification method	The declaration is generated with function name configured in the configuration parameter OcuNotification.	
Example(s)	**Action**	**Generated output**
	Configure Channel 2's notification as 23245	`/* No declaration is available */`
	Configure Channel 3's notification as 0.	`/* No declaration is available */`
	Configure Channel 3's notification as NULL.	`/* No declaration is available */`
	Configure Channel 4's notification as Notification_Ocu_Chan ().	`extern void Notification_Ocu_Chan (void);`

1.2.2 Structure: Ocu_Config[_<variant>]

Table 24 Ocu_Config[_<variant>]

Name	Ocu_Config[_<variant>]	
Type	Ocu_ConfigType	
Description	Root configuration structure of OCU driver which will be used during initialization.	
Verification method	The generated structure is present in OCU[_<variant>]_PBcfg.c file. The <variant> indicates the name of the post-build variant. For a variant-aware configuration the structure name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**
	Configure 1 OCU Channel 0 to Core0 (variant-unaware)	`const Ocu_ConfigType Ocu_Config = {`

	/* Pointer to channel configuration set per core */
	{
	&Ocu_kConfigCore_0,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR
	},
	#if (OCU_SINGLE_CORE == STD_OFF)
	{
	(uint16)(OCU_CORE0 | (uint8)0),
	},
	#endif
	};

Configure 1 OCU to Core0 (variant-aware. Variant name is 'Petrol')	const Ocu_ConfigType Ocu_Config_Petrol =
	{
	/* Pointer to channel configuration set per core */
	{
	&Ocu_kConfigCore_0_Petrol,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR,
	NULL_PTR
	},
	#if (OCU_SINGLE_CORE == STD_OFF)
	{
	(uint16)(OCU_CORE0 | (uint8)0),
	},
	#endif
	};

1.2.2.1 Member: CoreConfig[6]

Table 25 CoreConfig[6]

Name	CoreConfig[6]

Type	Ocu_CoreConfigType *	
Description	Array of core-specific configuration.	
Verification method	The generated structure member is present in the Ocu_Config[_<variant>] structure. If a Core<x> is allocated at least one channel, then the element <x> shall be generated as '&Ocu_kConfigCore_<x>' else 'NULL_PTR' is generated.(x in range 0 to 5).	
Example(s)	**Action**	**Generated output**

	All the OCU channels are allocated to Core 0 (variant-unaware)	`{`
		`  &Ocu_kConfigCore_0,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR`
		`}`

	All the OCU channels are allocated to Core 0 (variant-aware. Variant name is 'Petrol')	`{`
		`  &Ocu_kConfigCore_0_Petrol,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR,`
		`  NULL_PTR`
		`}`

	All the OCU channels are split between all cores except Core 0. (variant-unaware)	`{`
		`  NULL_PTR,`
		`  &Ocu_kConfigCore_1,`
		`  &Ocu_kConfigCore_2,`
		`  &Ocu_kConfigCore_3,`
		`  &Ocu_kConfigCore_4,`
		`  &Ocu_kConfigCore_5`
		`}`

1.2.2.2 Member: ChannelMapping[OCU_MAX_CHANNELS]

Table 26 ChannelMapping[OCU_MAX_CHANNELS]

Name	ChannelMapping[OCU_MAX_CHANNELS]
Type	uint16
Description	Array of channel specific data, which stores information of the core and index. Lower 8-bit for core specific channel identifier. Upper 8-bit to identify which core is using that channel

Verification method	The generated structure member contains an array entry for each configured channel at 'OcuConfigSet\OcuChannelId' index. The core specific channel identifier is the index of the channel in the list ordered in ascending order of "OcuConfigSet\OcuChannelId" for the channels allocated to that core.	
Example(s)	**Action**	**Generated output**

	Allocation order:
	OCU channel 0 is allocated to Core 0
	OCU channel 1 is allocated to Core 1.
		`{`
		`  0x000, /* Core 0 Index 0*/`
		`  0x100  /* Core 1 Index 0*/`
		`}`

	9 OCU channels. Channel2, Channel4 allocated to Core 4. Channel 3, Channel8 allocated to Core 1. Rest of the channels allocated to Core 2.
	Allocation order:
	Ocu Channel 0 is allocated to Core 2
	Ocu Channel 1 is allocated to Core 2
	Ocu Channel 2 is allocated to Core 4
	Ocu Channel 3 is allocated to Core 1
	Ocu Channel 4 is allocated to Core 4
	Ocu Channel 5 is allocated to Core 2
	Ocu Channel 6 is allocated to Core 2
	Ocu Channel 7 is allocated to Core 2
	Ocu Channel 8 is allocated to Core 1
		`{`
		`  0x200, /* Core 2 Index 0*/`
		`  0x201, /* Core 2 Index 1*/`
		`  0x400, /* Core 4 Index 0*/`
		`  0x100, /* Core 1 Index 0*/`
		`  0x401, /* Core 4 Index 1*/`
		`  0x202, /* Core 2 Index 2*/`
		`  0x203, /* Core 2 Index 3*/`
		`  0x204, /* Core 2 Index 4*/`
		`  0x101  /* Core 1 Index 1*/`
		`}`

1.2.3 Structure: Ocu_kConfigCore_<x>[_<variant>]

Table 27 Ocu_kConfigCore_<x>[_<variant>]

Name	Ocu_kConfigCore_<x>[_<variant>]	
Type	Ocu_CoreConfigType	
Description	Configuration structure of OCU driver for Core <x> which will be referenced in root configuration structure. (x ranges from 0 to 5)	
Verification method	The generated file has this structure if atleast one channel is assigned to Core <x>. <Variant> indicates the name of the post-build variant. For a variant-aware configuration, the structure name is appended with the variant name. For variant-unaware configuration, <variant> is ignored.	
Example(s)	**Action**	**Generated output**

	Configure 3 OCU channels to Core0 (variant-unaware)	`static const Ocu_CoreConfigType Ocu_kConfigCore_0 =`
		`{`
		`&Ocu_kChannelConfigCore_0[0],`
		`  3`
		`};`

	Configure 10 OCU channels to Core2 (variant-aware. Variant name is 'Petrol')	`static const Ocu_CoreConfigType Ocu_kConfigCore_2_Petrol = { &Ocu_kChannelConfigCore_2_Petrol[0], 10 };`

1.2.3.1 Member: ChannelConfigPtr

Table 28 ChannelConfigPtr

Name	ChannelConfigPtr	
Type	Ocu_ChannelConfigType *	
Description	Pointer to the base of array which stores the data of each channel configured to Core<x>.	
Verification method	The structure member is generated with base address of array which stores the channel data of Core <x>.	
Example(s)	**Action**	**Generated output**
	Configure atleast 1 OCU channel to Core 3.(variant-unaware)	`&Ocu_kChannelConfigCore_3[0]`
	Configure atleast 1 OCU channel to Core 4. (variant-aware. Variant name is 'Petrol')	`&Ocu_kChannelConfigCore_4_Petrol[0]`

1.2.3.2 Member: MaxChannelCore

Table 29 MaxChannelCore

Name	MaxChannelCore	
Type	uint8	
Description	Indicates the total number of channels assigned to Core for which the structure is generated.	
Verification method	The structure member is generated as total number of channels allocated to CORE<x>. *Note: Channels not assigned to any core are assigned to master core (ResourceMMasterCore).*	
Example(s)	**Action**	**Generated output**
	• Configure 4 OCU channels. 3 channels are allocated to Core 0. • 1 channel is allocated to Core 1. Output is shown for Core 0	3

	• Configure 14 OCU channels. 3 channels are allocated to Core 1. • ResourceMMasterCore is CORE0. • Rest of the channels are not allocated to any core. Output is shown for Core 0	11

1.2.4 Structure: Ocu_kChannelConfigCore_<x>[_<variant>]

Table 30 Ocu_kChannelConfigCore_<x>[_<variant>]

Name	Ocu_kChannelConfigCore_<x>[_<variant>]	
Type	Ocu_ChannelConfigType	
Description	Configuration structure of OCU driver for all channels belonging to Core <x> which will be referenced in core specific configuration structure (OCU_kConfigCore_<x>[_<variant>]). (x ranges from 0 to 5)	
Verification method	The generated file has this structure if at least one channel is assigned to Core <x>. <variant> indicates the name of the post-build variant. For a variant-aware configuration, the structure name is appended with the variant name. For variant-unaware configuration, <variant> is ignored.	
Example(s)	**Action**	**Generated output**
	Configure 1 OCU channel to Core0 (variant-unaware)	`static const Ocu_ChannelConfigType Ocu_kChannelConfigCore_0[ ] = { { /* OCU Channel 0 */ (Ocu_NotifiPtrType)0, (uint32)100, /*DefaultThreshold*/ (uint32)300, /*MaxCounterValue*/ (uint8)(0U), /* portpinout*/ { OCU_GTM_TOM, /* Assigned Hw Unit*/ 0x0000,/*Assigned Hw Unit Number*/ (uint8)OCU_LOW,/*Pin defaultstate*/ OCU_FALSE, /* Pin Used */ OCU_TRUE, /* Dma Used */ OCU_FALSE, /* Adc Used */ OCU_GTM_FIXED_CLOCK_0 /*Clock Select */ }, } };`

	Configure 1 OCU (1GTM) channel to Core 2(variant-aware. Variant name is 'Petrol')	`static const Ocu_ChannelConfigType Ocu_kChannelConfigCore_2_Petrol[ ] = { { /* Ocu Channel 0 */ (Ocu_NotifiPtrType)0, (uint32)100, /*DefaultThreshold*/ (uint32)300, /*MaxCounterValue*/ (uint8)(0U), /* portpinout*/ { OCU_GTM_TOM, /* Assigned Hw Unit*/ 0x0000,/*Assigned Hw Unit Number*/ (uint8)OCU_LOW,/*Pin defaultstate*/ OCU_FALSE, /* Pin Used */ OCU_TRUE, /* Dma Used */ OCU_FALSE, /* Adc Used */ OCU_GTM_FIXED_CLOCK_0 /*Clock Select */ }, } }`

1.2.4.1 Member: NotificationPointer

Table 31 NotificationPointer

Name	NotificationPointer	
Type	Ocu_NotifiPtrType	
Description	Pointer to the callback functions configured by the user.	
Verification method	The structure member is generated with function name or address configured in the configuration parameter OcuNotification.	
Example(s)	**Action**	**Generated output**
	Configure Channel 2's notification as 23245.	`(Ocu_NotifiPtrType)23245`
	Configure Channel 3's notification as 0.	`(Ocu_NotifiPtrType)0`
	Configure Channel 3's notification as NULL.	`(Ocu_NotifiPtrType)0`
	Configure Channel 4's notification as Notification_Ocu_Chanl4 ().	`&(Ocu_NotifiPtrType) Notification_Ocu_Chanl4`

1.2.4.2 Member: DefaultThreshold

Table 32 DefaultThreshold

Name	DefaultThreshold	
Type	Ocu_ValueType	
Description	Value of comparison threshold used for Initialization.(in ticks)	
Verification method	The structure member is generated as value configured in the configuration parameter OcuDefaultThreshold.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with OcuDefaultThreshold = 100	`(uint32)100`
	Configure an OCU channel with OcuDefaultThreshold = 300	`(uint32)100`

1.2.4.3 Member: MaxCounterValue

Table 33 MaxCounterValue

Name	MaxCounterValue	
Type	Ocu_ValueType	
Description	Maximum value in ticks, the counter of the OCU channel is able to count.	
Verification method	The structure member is generated as value configured in the configuration parameter OcuMaxCounterValue. Note: MaxCounterValue is not editable for ATOM-SOMC mode (i.e, GtmTimerClockSelect as TBU clock) and the default configured value is 0xFFFFFF.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with OcuMaxCounterValue = 100	`(uint32)100`
	Configure an OCU channel with OcuMaxCounterValue = 300	`(uint32)300`

1.2.4.4 Member: PinSelection

Table 34 PinSelection

Name	PinSelection	
Type	uint32	
Description	This parameter decides the output port and pin for the GTM timer.	
Verification method	The structure member is generated as value configured in the configuration parameter GtmTimerPortPinSelect. Lower 16 bit contains the TOUT number and Upper 16 bit contains the Timer Ouput Selection value.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with GtmTimerPortPinSelect = TOUT48_SELC_PORT22_PIN1	`(uint32)(48U | ((uint32)(OCU_ALT_SELC << 16U)))`
	Configure an OCU channel with GtmTimerPortPinSelect = TOUT53_SELC_PORT21_PIN2	`(uint32)(53U | ((uint32)(OCU_ALT_SELC << 16U)))`

1.2.4.5 Member: AssignedHwUnit

Table 35 AssignedHwUnit

Name	AssignedHwUnit	
Type	unsigned_int : 2	
Description	Hardware type selected for the OCU channel.	
Verification method	The structure member is generated based on the GtmTimerUsed and GtmTimerClockSelect configuration parameter. Assigned Hardware Unit is TOM or ATOM is decided based on the GtmTimerUsed configuration parameter. ATOM (SOMP) Channel or ATOM (SOMC) channel (Shared (TBU_TS0/1/2)) clock) is decided based on the GtmTimerClockSelect configuration parameter.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with GtmTimerUsed = (../McuGtmTomAllocationConf_0/ McuGtmTomChannelAllocationConf_0).	`OCU_GTM_TOM`
	Configure an OCU channel with GtmTimerUsed = (../McuGtmAtomAllocationConf_0/ McuGtmAtomChannelAllocationConf_0). and GtmTimerClockSelect = GTM_TBU_TS0	`OCU_GTM_ATOM_SHARED`
	Configure an OCU channel with GtmTimerUsed = (../McuGtmAtomAllocationConf_0/ McuGtmAtomChannelAllocationConf_1) and GtmTimerClockSelect = GTM_CONFIGURABLE_CLOCK_1	`OCU_GTM_ATOM`

1.2.4.6 Member: AssignedHwUnitNumber

Table 36 AssignedHwUnitNumber

Name	AssignedHwUnitNumber	
Type	unsigned_int : 16	
Description	Hardware Unit number used for the OCU channel.	
Verification method	The structure member is generated based on the string configured in the configuration parameter GtmTimerUsed. Upper 16bit contains the module number of the ATOM or TOM and the Lower 16bit contains the channel number. • (Module number << 3) + Channel number	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with • GtmTimerUsed = /Mcu/Mcu/McuHardwareResourceAllocationConf_0/ McuGtmAllocationConf_0/McuGtmAtomAllocationConf_0/McuGtmAtomChannelAllocationConf_1 (ATOM module is 0 and channel is 1)	`0x0001`

	Configure an OCU channel with • GtmTimerUsed = /Mcu/Mcu/McuHardwareResourceAllocationConf_0/McuGtmAllocationConf_0/McuGtmTomAllocationConf_5/McuGtmTomChannelAllocationConf_6 (ATOM module is 5 and channel is 6)	`0x0506`

1.2.4.7 Member: PinDefaultState

Table 37 PinDefaultState

Name	PinDefaultState	
Type	unsigned_int : 1	
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.	
Verification method	The structure member is generated as value configured in the configuration parameter OcuOutputPinDefaultState.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with OcuOutputPinDefaultState = OCU_HIGH	`(uint8)OCU_HIGH,`
	Configure an OCU channel with OcuOutputPinDefaultState = OCU_LOW	`(uint8)OCU_LOW,`

1.2.4.8 Member: PinUsed

Table 38 PinUsed

Name	PinUsed	
Type	unsigned_int : 1	
Description	Information about the usage of an output pin on this channel.	
Verification method	The structure member is generated OCU_TRUE if OcuOuptutPinUsed configuration parameter is set to ‘True’ else the structure member is generated as OCU_FALSE.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with OcuOuptutPinUsed = True	`OCU_TRUE`
	Configure an OCU channel with OcuOuptutPinUsed = False	`OCU_FALSE`

1.2.4.9 Member: DmaUsed

Table 39 DmaUsed

Name	DmaUsed	
Type	unsigned_int : 1	
Description	Information about the usage of a DMA event trigger on this channel.	
Verification method	The structure member is generated as OCU_TRUE if OcuHardwareTriggeredDMA configuration parameter has atleast one element in its container (../Ocu/OcuConfigSet /OcuChannel/<Channel name>/OcuHardwareTriggeredDMA) else the structure member is generated as OCU_FALSE.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with OcuHardwareTriggeredDMA configuration parameter with elements in its container.	`OCU_TRUE`
	Configure an OCU channel with OcuHardwareTriggeredDMA configuration parameter with zero (0) elements in its container.	`OCU_FALSE`

1.2.4.10 Member: AdcUsed

Table 40 AdcUsed

Name	AdcUsed	
Type	unsigned_int : 1	
Description	Information about the usage of ADC event trigger on this channel.	
Verification method	The structure member is generated as OCU_TRUE if OcuHardwareTriggeredAdc configuration parameter has atleast one element in its container (../Ocu/OcuConfigSet /OcuChannel/<Channel name>/OcuHardwareTriggeredAdc) else the structure member is generated as OCU_FALSE.	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with OcuHardwareTriggeredAdc configuration parameter with elements in its container.	`OCU_TRUE`
	Configure an OCU channel with OcuHardwareTriggeredAdc configuration parameter with zero (0) elements in its container.	`OCU_FALSE`

1.2.4.11 Member: ClockSelect

Table 41 ClockSelect

Name	ClockSelect
Type	unsigned_int : 4
Description	This parameter decides the Clock Source for TOM/ATOM timer.

Verification method	The structure member is generated as value configured in the configuration parameter GtmTimerClockSelect .	
Example(s)	**Action**	**Generated output**
	Configure an OCU channel with GtmTimerClockSelect = GTM_FIXED_CLOCK_0	`GTM_FIXED_CLOCK_0`
	Configure an OCU channel with GtmTimerClockSelect = GTM_CONFIGURABLE_CLOCK_0	`GTM_CONFIGURABLE_CLOCK_0`
	Configure an OCU channel with GtmTimerClockSelect = GTM_TBU_TS0	`GTM_TBU_TS0`

1.2.4.12 Member: Reserved

Table 42 Reserved

Name	Reserved	
Type	unsigned_int : 6	
Description	Reserved bit field for 32-bit padding.	
Verification method	The structure member is generated as 0. *Note: The member is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate the configuration file of OCU	`0U`

1.3 File: OCU[_<variant>]_PBcfg.h

The generated header file contains the declaration of the root configuration structure. Post-build time configuration mechanism allows configurable functionality of OCU driver that is deployed as object code. The file is generated in ‘inc’ folder.

1.3.1 Structure: Ocu_Config[_<variant>]

Table 43 Ocu_Config[_<varaint>]

Name	Ocu_Config[_<variant>]	
Type	Ocu_ConfigType	
Description	Declaration of Root configuration structure of Ocu driver which will be used during initialization.	
Verification method	The generated structure is present in OCU[_<variant>]_PBcfg.h file. The <variant> indicates the name of the post-build variant. For a variant-aware configuration the structure name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**

	Configure atleast one OCU channel and generate (variant-unaware)	`extern const Ocu_ConfigType Ocu_Config;`
	Configure atleast one OCU channel and generate (variant-aware. Variant name is ‘Petrol’)	`extern const Ocu_ConfigType Ocu_Config_Petrol;`

Revision history

Major changes since the last revision

Date	Version	Description
2020-08-11	2.0	Document Released
2020-08-07	1.1	• Ocu driver chapter moved from MC-ISAR_TC3xx_Config_Verification_Manual_Basic.pdf to this document. • TOUT configuration is performed centrally in MCU module. Hence, “PinSelection” parameter is removed from the configuration structure “Ocu_kChannelConfigCore_<x>”.
2019-07-22	1.0	Document Released
2019-07-22	0.1	Initial Version.

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

Edition 2020-08-11
Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Doc_Number

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.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

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.