MCAL Configuration Verification Manual for Uart

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 Uart

Table of contents

1 Uart driver

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

1.1 File: Uart_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: UART_AR_RELEASE_MAJOR_VERSION

Table 1 UART_AR_RELEASE_MAJOR_VERSION

Name	UART_AR_RELEASE_MAJOR_VERSION	
Description	Major version number of AUTOSAR release on which the Uart 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 Uart_Cfg.h file with ArMajorVersion 4	`#define UART_AR_RELEASE_MAJOR_VERSION (4U)`

1.1.2 Macro: UART_AR_RELEASE_MINOR_VERSION

Table 2 UART_AR_RELEASE_MINOR_VERSION

Name	UART_AR_RELEASE_MINOR_VERSION	
Description	Minor version number of AUTOSAR release on which the Uart 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 Uart_Cfg.h file with ArMajorVersion 2	`#define UART_AR_RELEASE_MINOR_VERSION (2U)`
	Generate Uart_Cfg.h file with ArMajorVersion 4	`#define UART_AR_RELEASE_MINOR_VERSION (4U)`

1.1.3 Macro: UART_AR_RELEASE_PATCH_VERSION

Table 3 UART_AR_RELEASE_PATCH_VERSION

Name	UART_AR_RELEASE_PATCH_VERSION	
Description	Revision version number of AUTOSAR release on which the Uart 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 Uart_Cfg.h file with ArPatchVersion 2	`#define UART_AR_RELEASE_PATCH_VERSION  (2U)`
	Generate Uart_Cfg.h file with ArPatchVersion 0	`#define UART_AR_RELEASE_PATCH_VERSION  (0U)`

1.1.4 Macro: UART_SW_MAJOR_VERSION

Table 4 UART_SW_MAJOR_VERSION

Name	UART_SW_MAJOR_VERSION	
Description	Major version number of the Uart 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 Uart_Cfg.h file with SwMajorVersion 2	`#define UART_SW_MAJOR_VERSION  (2U)`

1.1.5 Macro: UART_SW_MINOR_VERSION

Table 5 UART_SW_MINOR_VERSION

Name	UART_SW_MINOR_VERSION	
Description	Minor version number of the Uart 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 Uart_Cfg.h file with SwMinorVersion 0	`#define UART_SW_MINOR_VERSION  (0U)`

1.1.6 Macro: UART_SW_PATCH_VERSION

Table 6 UART_PATCH_VERSION

Name	UART_SW_PATCH_VERSION	
Description	Patch level version number of the Uart 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 Uart_Cfg.h file with SwPatchVersion 0	`#define UART_SW_PATCH_VERSION   (0U)`

1.1.7 Macro: UART_SAFETY_ENABLE

Table 7 UART_SAFETY_ENABLE

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

1.1.8 Macro: UART_INITCHECK_API

Table 8 UART_INITCHECK_API

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

1.1.9 Macro: UART_INIT_DEINIT_API_MODE

Table 9 UART_INIT_DEINIT_API_MODE

Name	UART_INIT_DEINIT_API_MODE
Description	Decides the mode of execution of Init and DeInit API's.
Verification method	The macro is generated as UART_SUPERVISOR_MODE if UartInitDeInitApiMode

	configuration parameter is set to 'MCAL_SUPERVISOR' else the macro is generated as UART_USER1_MODE.	
Example(s)	**Action**	**Generated output**
	UartInitDeInitApiMode = MCAL_SUPERVISOR	`#define UART_INIT_DEINIT_API_MODE (UART_SUPERVISOR_MODE)`
	UartInitDeInitApiMode = MCAL_USER1	`#define UART_INIT_DEINIT_API_MODE (UART_USER1_MODE)`

1.1.10 Macro: UART_DEV_ERROR_DETECT

Table 10 UART_DEV_ERROR_DETECT

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

1.1.11 Macro: UART_RUNTIME_ERROR_DETECT

Table 11 UART_RUNTIME_ERROR_DETECT

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

1.1.12 Macro: UART_DEINIT_API

Table 12 UART_DEINIT_API

Name	UART_DEINIT_API	
Description	Enables/disables Uart_DeInit API.	
Verification method	The macro is generated as STD_ON if UartDeinitApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**

	UartDeInitApi = True	`#define UART_DEINIT_API (STD_ON)`
	UartDeInitApi = False	`#define UART_DEINIT_API (STD_OFF)`

1.1.13 Macro: UART_VERSION_INFO_API

Table 13 UART_VERSION_INFO_API

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

1.1.14 Macro: UART_ABORT_READ_API

Table 14 UART_ABORT_READ_API

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

1.1.15 Macro: UART_ABORT_WRITE_API

Table 15 UART_ABORT_WRITE_API

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

	UartAbortWriteApi = False	`#define UART_ABORT_WRITE_API (STD_OFF)`

1.1.16 Macro: UART_RX_MODE

Table 16 UART_RX_MODE

Name	UART_RX_MODE	
Description	Configures the mode of receive operation in polling / interrupt / mixed (both interrupt and polling) mode.	
Verification method	The macro is generated as UART_POLLING_MODE if UartRxChannelMode configuration parameter is set to 'POLLING' for all configured channels. The macro is generated as UART_INTERRUPT_MODE if UartRxChannelMode configuration parameter is set to 'INTERRUPT' for all configured channels. The macro is generated as UART_MIXED_MODE if UartRxChannelMode configuration parameter is set to 'INTERRUPT' for at least one configured channel and 'POLLING' for at least one configured channel.	
Example(s)	**Action**	**Generated output**
	UartRxChannelMode = POLLING (For all configured channels)	`#define UART_RX_MODE (UART_POLLING_MODE)`
	UartRxChannelMode = INTERRUPT(For all configured channels)	`#define UART_RX_MODE (UART_INTERRUPT_MODE)`
	UartRxChannelMode = INTERRUPT(For at least one configured channel) and UartRxChannelMode = POLLING(For at least one configured channel)	`#define UART_RX_MODE (UART_MIXED_MODE)`

1.1.17 Macro: UART_SLEEP_MODE_SUPPORT

Table 17 UART_SLEEP_MODE_SUPPORT

Name	UART_SLEEP_MODE_SUPPORT	
Description	Enables/disables UART driver sleep mode.	
Verification method	The macro is generated as numeric value 0U if configuration parameter UartSleepEnable is set to 'True' else the macro is generated as 0x08.	
Example(s)	**Action**	**Generated output**
	UartSleepEnable = True	`#define UART_SLEEP_MODE_SUPPORT (0U)`
	UartSleepEnable = False	`#define UART_SLEEP_MODE_SUPPORT (0x08U)`

1.1.18 Macro: UART_TX_MODE

Table 18 UART_TX_MODE

Name	UART_TX_MODE	
Description	Configures the mode of transmit operation in polling / interrupt / mixed (both interrupt and polling) mode.	
Verification method	The macro is generated as UART_POLLING_MODE if UartTxChannelMode configuration parameter is set to 'POLLING' for all configured channels. The macro is generated as UART_INTERRUPT_MODE if UartTxChannelMode configuration parameter is set to 'INTERRUPT' for all configured channels. The macro is generated as UART_MIXED_MODE if UartTxChannelMode configuration parameter is set to 'INTERRUPT' for at least one configured channel and 'POLLING' for at least one configured channel.	
Example(s)	**Action**	**Generated output**
	UartTxChannelMode = POLLING (For all configured channels)	`#define UART_TX_MODE (UART_POLLING_MODE)`
	UartTxChannelMode = INTERRUPT(For all configured channels)	`#define UART_TX_MODE (UART_INTERRUPT_MODE)`
	UartTxChannelMode = INTERRUPT(For at least one configured channel) and UartTxChannelMode = POLLING(For at least one configured channel)	`#define UART_TX_MODE (UART_MIXED_MODE)`

1.1.19 Macro: UART_MAXTIMEOUT_COUNT

Table 19 UART_MAXTIMEOUT_COUNT

Name	UART_MAXTIMEOUT_COUNT	
Description	Specifies the maximum time in nanoseconds to wait for reporting hardware timeout errors.	
Verification method	The macro is generated as a numeric value set in the configuration parameter 'UartTimeoutCount'.	
Example(s)	**Action**	**Generated output**
	Set UartTimeoutCount as 100	`#define UART_MAXTIMEOUT_COUNT (100U)`
	Set UartTimeoutCount as 240000	`#define UART_MAXTIMEOUT_COUNT (240000U)`

1.1.20 Macro: UART_INDEX

Table 20 UART_INDEX

Name	UART_INDEX	
Description	Instance ID of UART module.	
Verification method	The macro is generated as a numeric value set in the configuration parameter UartIndex	
Example(s)	**Action**	**Generated output**
	Set UartIndex as 0	`#define UART_INDEX (0U)`
	Set UartIndex as 240	`#define UART_INDEX (240U)`

1.1.21 Macro: UART_NUM_OF_CHANNEL_CONFIGURED

Table 21 UART_NUM_OF_CHANNEL_CONFIGURED

Name	UART_NUM_OF_CHANNEL_CONFIGURED	
Description	Indicates the total number of UART channels configured.	
Verification method	The macro is generated as total number of UART channels configured.	
Example(s)	**Action**	**Generated output**
	• Configure 4 UART channels	`#define UART_NUM_OF_CHANNEL_CONFIGURED (4U)`
	• Configure 1 UART channel	`#define UART_NUM_OF_CHANNEL_CONFIGURED (1U)`

1.1.22 Macro: UART_MAX_HW_UNIT

Table 22 UART_MAX_HW_UNIT

Name	UART_MAX_HW_UNIT	
Description	Indicates the maximum number of UART channels supported by device variant.	
Verification method	The macro is generated as last index of UART channel supported by device variant plus 1.	
Example(s)	**Action**	**Generated output**
	Generate Uart_Cfg.h file for a TC333 device.	`#define UART_MAX_HW_UNIT (9U)`
	Generate Uart_Cfg.h file for a TC399 device.	`#define UART_MAX_HW_UNIT (12U)`

1.1.23 Macro: UART_ASCLIN<x>

Table 23 UART_ASCLIN<x>

Name	UART_ASCLIN<x>	
Description	Specify the the ASCLIN channel number. *Note: This macro is not configurable by the user.*	
Verification method	The macro is generated as a numeric value which corresponds to the hardware unit identifier. For non-consecutive ASCLIN devices, this macro is not generated for ASCLIN hardware unit which is not available in the device.	
Example(s)	**Action**	**Generated output**

	Generate Uart_Cfg.h file for a TC333 device	`#define UART_ASCLIN0 (0U)`
		`#define UART_ASCLIN1 (1U)`
		`#define UART_ASCLIN2 (2U)`
		`#define UART_ASCLIN3 (3U)`
		`#define UART_ASCLIN8 (8U)`

	Generate Uart_Cfg.h file for a TC399 device.	`#define UART_ASCLIN0 (0U)`
		`#define UART_ASCLIN1 (1U)`
		`#define UART_ASCLIN2 (2U)`
		`#define UART_ASCLIN3 (3U)`
		`#define UART_ASCLIN4 (4U)`
		`#define UART_ASCLIN5 (5U)`
		`#define UART_ASCLIN6 (6U)`
		`#define UART_ASCLIN7 (7U)`
		`#define UART_ASCLIN8 (8U)`
		`#define UART_ASCLIN9 (9U)`
		`#define UART_ASCLIN10 (10U)`
		`#define UART_ASCLIN11 (11U)`
		`#define UART_ASCLIN12 (12U)`
		`#define UART_ASCLIN13 (13U)`
		`#define UART_ASCLIN14 (14U)`
		`#define UART_ASCLIN15 (15U)`
		`#define UART_ASCLIN16 (16U)`
		`#define UART_ASCLIN17 (17U)`
		`#define UART_ASCLIN18 (18U)`
		`#define UART_ASCLIN19 (19U)`
		`#define UART_ASCLIN20 (20U)`
		`#define UART_ASCLIN21 (21U)`
		`#define UART_ASCLIN22 (22U)`
		`#define UART_ASCLIN23 (23U)`

1.1.24 Macro: UART_ASCLIN_REG_ADDR

Table 24 UART_ASCLIN_REG_ADDR

Name	UART_ASCLIN_REG_ADDR	
Description	SFR base address of ASCLIN hardware modules available in device variant. *Note: This macro is not configurable by the user.*	
Verification method	The macro is generated as a SFR base address (`&MODULE_ASCLINx`). Where 'x' is varies from 0 to number of ASCLIN hardware modules available in device variant. NULL pointer is generated for the ASCLIN hardware module which is not available in the device.	
Example(s)	**Action**	**Generated output**

	TC399 device has 12 ASCLIN hardware modules	`#define UART_ASCLIN_REG_ADDR\`
		`&MODULE_ASCLIN0,`
		`&MODULE_ASCLIN1,`
		`&MODULE_ASCLIN2,`
		`&MODULE_ASCLIN3,`
		`&MODULE_ASCLIN4,`
		`&MODULE_ASCLIN5,`
		`&MODULE_ASCLIN6,`
		`&MODULE_ASCLIN7,`
		`&MODULE_ASCLIN8,`
		`&MODULE_ASCLIN9,`
		`&MODULE_ASCLIN10,`
		`&MODULE_ASCLIN11`

	TC333 device has 5 ASCLIN hardware modules	`#define UART_ASCLIN_REG_ADDR\`
		`&MODULE_ASCLIN0,`
		`&MODULE_ASCLIN1,`
		`&MODULE_ASCLIN2,`
		`&MODULE_ASCLIN3,`
		`NULL_PTR,`
		`NULL_PTR,`
		`NULL_PTR,`
		`NULL_PTR,`
		`&MODULE_ASCLIN8`

1.1.25 Macro: UART_CSRREG_CLKSEL_CLC

Table 25 UART_CSRREG_CLKSEL_CLC

Name	UART_CSRREG_CLKSEL_CLC
Description	This macro determines whether the ASCLIN peripheral frequency is configured in fast or slow mode.
Verification method	The macro is generated as UART_CSRREG_CLKSEL_FASTCLK if UartCsrClksel

	configuration parameter is set to 'ASCLINF' for ASCLIN module. The macro is generated as UART_CSRREG_CLKSEL_SLOWCLK if UartCsrClksel configuration parameter is set to 'ASCLINS' for ASCLIN module.	
Example(s)	**Action**	**Generated output**
	UartCsrClksel = ASCLINF	`#define UART_ CSRREG_CLKSEL_CLC (UART_CSRREG_CLKSEL_FASTCLK)`
	UartCsrClksel = ASCLINS	`#define UART_ CSRREG_CLKSEL_CLC (UART_CSRREG_CLKSEL_SLOWCLK)`

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

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

1.2.1 Structure: Uart_Config[_<variant>]

Table 26 Uart_Config[_<variant>]

Name	Uart_Config[_<variant>]	
Type	Uart_ConfigType	
Description	Array of structure which contains configuration of each of UART channels. The base address of this structure array will be will be referenced in root configuration structure.	
Verification method	The generated structure is present in Uart[_<variant>]_PBcfg.c file. <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 2 UART channels (variant unaware)	`const Uart_ConfigType Uart_Config =`
		`{`
		`  &Uart_ChannelConfig[0],`
		`  &Uart_ChannelIdLookup[0],`
		`  2U`
		`};`

	Configure 2 UART channels (variant aware. Variant name is 'Petrol')	`const Uart_ConfigType Uart_Config_Petrol =`
		`{`
		`  &Uart_ChannelConfig_Petrol[0],`
		`  &Uart_ChannelIdLookup_Petrol[0],`
		`  2U`
		`};`

1.2.1.1 Member: ChannelConfigPtr

Table 27 ChannelConfigPtr

Name	ChannelConfigPtr	
Type	Uart_ChannelConfigType *	
Description	Pointer to the base of array which stores the data of each channel configured.	
Verification method	The generated structure member is present in the Uart_Config[_<variant>] structure.	
Example(s)	**Action**	**Generated output**
	Configure 1 UART channel (variant unaware)	`&Uart_ChannelConfig[0],`
	Configure 1 UART channel (variant aware. Variant name is 'Petrol')	`&Uart_ChannelConfig_Petrol[0],`

1.2.1.2 Member: Uart_ChannelIdLookupPtr

Table 28 Uart_ChannelIdLookupPtr

Name	Uart_ChannelIdLookupPtr	
Type	uint8 *	
Description	Pointer to the base of array which stores the data of UART channel lookup table.	
Verification method	The generated structure member is present in the Uart_Config[_<variant>] structure.	
Example(s)	**Action**	**Generated output**
	Configure 1 UART channel (variant unaware)	`&Uart_ChannelIdLookup[0],`
	Configure 1 UART channel (variant aware. Variant name is 'Petrol')	`&Uart_ChannelIdLookup_Petrol[0],`

1.2.1.3 Member: NoOfChannels

Table 29 NoOfChannels

Name	NoOfChannels	
Type	uint8	
Description	Number of UART channels configured.	
Verification method	The structure member is generated as numeric value based on number of channel configured in container 'UartConfigSet/UartChannel'	
Example(s)	**Action**	**Generated output**
	Configure 1 UART channel	`1U`
	Configure 10 UART channels	`10U`

1.2.2 Structure: Uart_ChannelConfig[_<variant>][<ChannelCount>]

Table 30 Uart_ChannelConfig[_<variant>][<ChannelCount>]

Name	Uart_ChannelConfig[_<variant>][<ChannelCount>]	
Type	Uart_ChannelConfigType	
Description	Array of structure which contains configuration of each of UART channels. The base address of this structure array will be referenced in root configuration structure.	
Verification method	The generated file has this structure if at least one channel. <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. <ChannelCount> is number of channel configured.	
Example(s)	**Action**	**Generated output**

	Configure 1 UART channel (variant unaware)	`static const Uart_ChannelConfigType Uart_ChannelConfig[1] =`
		`{`
		`  /* UART Channel ID: 0 Configuration */`
		`  { /* Notification function */`
		`    {`
		`      /* Call-back notification function for write operation */`
		`        &Ch0Transmit,`
		`      /* Call-back notification function for read operation */`
		`        &Ch0Receive,`
		`      /* Call-back notification function for abort write operation */`
		`        NULL_PTR,`
		`      /* Call-back notification function for abort read operation */`
		`        NULL_PTR,`
		`    },`
		`    /* BaudRate : 1920.0 Hz  */`
		`    /* Channel baud rate numerator */`
		`    24U,`
		`    /* Channel baud rate denominator */`
		`    1000U,`
		`    /* Channel baud rate prescalar */`
		`    24U,`
		`    /* Channel oversampling */`
		`    9U,`
		`    /* Hardware channel id */`
		`    UART_ASCLIN0,`
		`    /* Number of stop Bits */`

	`1U,`
	`/* Frame length */`
	`8U,`
	`/* Alternate receive pin */`
	`UART_SELECT_A,`
	`/* Identifies the parity is enabled */`
	`0U,`
	`/* Identifies the parity is even or odd */`
	`0U,`
	`/* Alternate CTS pin */`
	`UART_SELECT_CTS_A,`
	
	`/* CTS is enabled */`
	`0U,`
	`/* RTS/CTS polarity if CTS is enabled */`
	`0U,`
	`/* Receive operation mode polling/interrupt */`
	`UART_RX_INTERRUPT,`
	`/* Transmit operation mode polling/interrupt  */`
	`UART_TX_INTERRUPT`
	`}`
	`};`

Configure 1 UART channel (variant aware, Variant name is ‘Petrol’)	`static const Uart_ChannelConfigType Uart_ChannelConfig_Petrol[1] =`
	`{`
	`/* UART Channel ID: 0 Configuration */`
	`{ /* Notification function */`
	`{`
	`/* Call-back notification function for write operation */`
	`&Ch0Transmit,`
	`/* Call-back notification function for read operation */`
	`&Ch0Receive,`
	`/* Call-back notification function for abort write operation */`
	`NULL_PTR,`
	`/* Call-back notification function for abort read operation */`
	`NULL_PTR,`
	`},`
	`/* BaudRate : 1920.0 Hz  */`

```
    /* Channel baud rate numerator */
    24U,
    /* Channel baud rate denominator */
    1000U,
    /* Channel baud rate prescalar */
    24U,
    /* Channel oversampling */
    9U,
    /* Hardware channel id */
    UART_ASCLIN0,
    /* Number of stop Bits */
    1U,
    /* Frame length */
    8U,
    /* Alternate receive pin */
    UART_SELECT_A,
    /* Identifies the parity is enabled */
    0U,
    /* Identifies the parity is even or odd
*/
    0U,
     /* Alternate CTS pin */
    UART_SELECT_CTS_A,

    /* CTS is enabled */
    0U,
    /* RTS/CTS polarity if CTS is enabled */
    0U,
    /* Receive operation mode
polling/interrupt */
    UART_RX_INTERRUPT,
     /* Transmit operation mode
polling/interrupt  */
    UART_TX_INTERRUPT
  }
};
```

1.2.2.1 Member: UartNotif

Table 31 UartNotif

Name	UartNotif
Type	Uart_NotifType

Description	Structure member which stores the callback notification pointer.	
Verification method	The structure member is generated as callback function address which is configured for channel UartNotification/UartTransmitNotifPtr, UartNotification/UartReceiveNotifPtr, UartNotification/UartAbortTransmitNotifPtr and UartNotification/UartAbortReceiveNotifPtr.	
Example(s)	**Action**	**Generated output**

	Configure at least 1 UART channel with tx notification with Ch0Transmit, rx notification with Ch0Receive, abort tx with Ch0AbortTransmit and abort rx with Ch0AbortReceive.	`/* Notification function */`
		`{`
		`/* Call-back notification function for write operation */`
		`&Ch0Transmit,`
		`/* Call-back notification function for read operation */`
		`&Ch0Receive,`
		`/* Call-back notification function for abort write operation */`
		`&Ch0AbortTransmit,`
		`/* Call-back notification function for abort read operation */`
		`&Ch0AbortReceive,`
		`}`

	Configure at least 1 UART channel with tx notification with NULL_PTR, rx notification with NULL_PTR, abort tx with NULL_PTR and abort rx with NULL_PTR.	`/* Notification function */`
		`{`
		`/* Call-back notification function for write operation */`
		`NULL_PTR,`
		`/* Call-back notification function for read operation */`
		`NULL_PTR,`
		`/* Call-back notification function for abort write operation */`
		`NULL_PTR,`
		`/* Call-back notification function for abort read operation */`
		`NULL_PTR,`
		`},`

1.2.2.2 Member: ChanBaudRateNumerator

Table 32 ChanBaudRateNumerator

Name	ChanBaudRateNumerator
Type	uint16
Description	Indicates the UART channel baud rate numerator value for BRG.
Verification method	The structure member is generated as numeric value.

	• If UartAutoCalcBaudParams 'False' then generates with value configured in parameter 'UartChannel/UartChanBaudNumerator'. • If UartAutoCalcBaudParams is set 'True' then value is calculated based on input frequency and baud rate value. ChanBaudRateNumerator is generated as per following formula: $f^{PD} = f^{A}$ / (ChanBaudRatePrescalar + 1) $f^{OVS} = f^{PD}$ * (ChanBaudRateNumerator / ChanBaudRateDenominator) f^{SHIFT} (Baud Rate)= f^{OVS} / (UartChanBaudOverSampling + 1) $f^{ASCLINF}$ or $f^{ASCLINS}$ is used as input clock frequency (f^{A}).	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartAutoCalcBaudParams = False • UartChanBaudNumerator = 10.	`10U`
	• Configure 1 UART channel. • UartChanBaudNumerator = True. • UartBaudRate = 9600. • Input frequency $f^{ASCLINS}$ set to 20 MHz.	`24U`

1.2.2.3 Member: ChanBaudRateDenominator

Table 33 ChanBaudRateDenominator

Name	ChanBaudRateDenominator
Type	uint16
Description	This structure member value is used to configure the DENOMINATOR field of BRG register.
Verification method	The structure member is generated as numeric value. • If UartAutoCalcBaudParams 'False' then generates with value configured in parameter 'UartChannel/UartChanBaudDenominator'. • If UartAutoCalcBaudParams is set 'True' then value is calculated based on input frequency and baud rate value. ChanBaudRateDenominator is generated as per following formula: $f^{PD} = f^{A}$ / (ChanBaudRatePrescalar + 1) $f^{OVS} = f^{PD}$ * (ChanBaudRateNumerator / ChanBaudRateDenominator) f^{SHIFT} (Baud Rate)= f^{OVS} / (UartChanBaudOverSampling + 1)

	$f^{ASCLINF}$ or $f^{ASCLINS}$ is used as input clock frequency (f^{A}).	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartAutoCalcBaudParams = False • UartChanBaudDenominator = 10.	`10U`
	• Configure 1 UART channel. • UartAutoCalcBaudParams = True. • UartBaudRate = 9600. • Input frequency $f^{ASCLINS}$ set to 20 MHz.	`1000U`

1.2.2.4 Member: ChanBaudRatePrescalar

Table 34 ChanBaudRatePrescalar

Name	ChanBaudRatePrescalar	
Type	uint16	
Description	This structure member value is used to configure the PRESCALAR of BITCON register.	
Verification method	The structure member is generated as numeric value. • If UartAutoCalcBaudParams 'False' then generates with value configured in parameter 'UartChannel/UartChanBaudPrescalar'. • If UartAutoCalcBaudParams is set 'True' then value is calculated based on input frequency and baud rate value. ChanBaudRatePrescalar is generated as per following formula: $f^{PD} = f^{A} / (ChanBaudRatePrescalar + 1)$ $f^{OVS} = f^{PD} * (ChanBaudRateNumerator / ChanBaudRateDenominator)$ f^{SHIFT} (Baud Rate)= f^{OVS} / (UartChanBaudOverSampling + 1) $f^{ASCLINF}$ or $f^{ASCLINS}$ is used as input clock frequency (f^{A}).	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartAutoCalcBaudParams = False • Set UartChanBaudPrescalar = 10.	`10U`

	• Configure 1 UART channel. • UartAutoCalcBaudParams = True. • UartBaudRate = 9600. • Input frequency $f^{ASCLINS}$ set to 20 MHz.	`4U`

1.2.2.5 Member: UartChanBaudOverSampling

Table 35 UartChanBaudOverSampling

Name	UartChanBaudOverSampling	
Type	uint8	
Description	This structure member value is used to configure the OVERSAMPLING of BITCON register.	
Verification method	The structure member is generated as numeric value. • If UartAutoCalcBaudParams 'False' then generates with value configured in parameter 'UartChannel/UartChanBaudOverSampling'. • If UartAutoCalcBaudParams is set 'True' then value is calculated based on input frequency and baud rate value. UartChanBaudOverSampling is generated as per following formula: $f^{PD} = f^{A}$ / (ChanBaudRatePrescalar + 1) $f^{OVS} = f^{PD}$ * (ChanBaudRateNumerator / ChanBaudRateDenominator) f^{SHIFT} (Baud Rate)= f^{OVS} / (UartChanBaudOverSampling + 1) $f^{ASCLINF}$ or $f^{ASCLINS}$ is used as input clock frequency (f^{A}).	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartAutoCalcBaudParams = False • UartChanBaudOverSampling = 10.	`10U`
	• Configure 1 UART channel. • UartAutoCalcBaudParams = True. • UartBaudRate = 9600. • Input frequency $f^{ASCLINS}$ set to 20 MHz.	`9U`

1.2.2.6 Member: HwModule

Table 36 HwModule

Name	HwModule	
Type	uint8	
Description	ASCLIN hardware channel identifier.	
Verification method	The structure member is generated as `UART_`<UartHwUnit>, value of UartHwUnit is retrieved from configuration parameter 'UartChannel/UartHwUnit'.	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartHwUnit = ASCLIN0.	`/* Hardware channel id */` `UART_ASCLIN0,`
	Configure 1 UART channel with UartHwUnit = ASCLIN11.	`/* Hardware channel id */` `UART_ASCLIN11,`

1.2.2.7 Member: StopBits

Table 37 StopBits

Name	StopBits	
Type	uint8	
Description	This structure member value is used to configure number of stop bits for UART channel.	
Verification method	The structure member is generated as per value configured in parameter 'UartChannel/UartStopBits'.	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartStopBits = 1.	`/* Number of stop Bits */` `1U,`
	Configure 1 UART channel with UartStopBits = 2.	`/* Number of stop Bits */` `2U,`

1.2.2.8 Member: DataLength

Table 38 DataLength

Name	DataLength	
Type	uint8	
Description	This structure member value is used to configure frame length for UART channel.	
Verification method	The structure member is generated as value configured in parameter 'UartChannel/UartDataLength'.	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartDataLength = 2.	`/* Frame length */` `2U,`
	• Configure 1 UART channel with UartDataLength = 16.	`/* Frame length */` `16U,`

1.2.2.9 Member: RxPin

Table 39 RxPin

Name	RxPin	
Type	uint8	
Description	This structure member value is used to configure alternate receive pin for ASCLIN channel.	
Verification method	The structure member is generated as RxPin value configured in parameter 'UartChannel/UartRxPinSelection'.	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartRxPinSelection = SELECT_A_PORT14_PIN1.	`/* Alternate receive pin */` `UART_SELECT_A,`
	• Configure 1 UART channel with UartRxPinSelection = SELECT_E_PORT13_PIN11	`/* Alternate receive pin */` `UART_SELECT_E,`

1.2.2.10 Member: ParityEnable

Table 40 ParityEnable

Name	ParityEnable	
Type	uint8	
Description	This structure member value is used to configure enable/disable parity check/generation for ASCLIN channel.	
Verification method	The structure member is generated as per value configured for parameter 'UartChannel/UartParityBit'. If UartParityBit is set to 'NOPARITY' then member generated with value 0 else it generate with value 1.	
Example(s)	**Action**	**Generated output**
	• Configure 1 UART channel with UartParityBit = ODDPARITY.	`/* Identifies the parity is enabled */` `1U,`
	• Configure 1 UART channel with UartParityBit = EVENPARITY	`/* Identifies the parity is enabled */` `1U,`
	• Configure 1 UART channel with UartParityBit = NOPARITY.	`/* Identifies the parity is enabled */` `0U,`

1.2.2.11 Member: Parity

Table 41 Parity

Name	Parity	
Type	uint8	
Description	This structure member value is used to configure even/odd parity check/generation for ASCLIN channel.	
Verification method	The structure member is generated as per value configured in parameter 'UartChannel/UartParityBit'. If UartParityBit is set to 'EVENPARITY' then member generated with value 0. If UartParityBit is set to 'ODDPARITY' then member generated with value 1.	
Example(s)	**Action**	**Generated output**

	• Configure 1 UART channel with UartParityBit = EVENPARITY.	`/* Identifies the parity is even or odd */`
		`0U,`

	• Configure 1 UART channel with UartParityBit = ODDPARITY.	`/* Identifies the parity is even or odd */`
		`1U,`

1.2.2.12 Member: CTSPin

Table 42 CTSPin

Name	CTSPin	
Type	uint8	
Description	This structure member value is used to configure alternate CTS pin selection for ASCLIN channel.	
Verification method	The structure member is generated as value configured in parameter 'UartChannel/UartCTSPinSelection'.	
Example(s)	**Action**	**Generated output**

	• Configure 1 UART channel with UartCTSPinSelection = SELECT_CTS_A_PORT14_PIN9.	`/* Alternate CTS pin */`
		`UART_SELECT_CTS_A,`

1.2.2.13 Member: CtsEnable

Table 43 CtsEnable

Name	CtsEnable
Type	uint8
Description	This structure member value is used to configure enable/disable CTS for ASCLIN channel.
Verification method	The structure member is generated as per value configured in parameter

	'UartChannel/UartCTSEnable'. If UartCTSEnable is set to 'True' then member generated with value 1U. If UartCTSEnable is set to 'False' then member generated with value 0U.	
Example(s)	**Action**	**Generated output**
	• Configure UART channel with UartCTSEnable = True.	`1U,`
	• Configure UART channel with UartCTSEnable = False.	`0U,`

1.2.2.14 Member: CtsPolarity

Table 44 CtsPolarity

Name	CtsPolarity	
Type	uint8	
Description	This structure member value is used to configure polarity of CTS pin for ASCLIN channel.	
Verification method	The structure member is generated as per value configured in parameter 'UartChannel/UartCTSPolarity'. If UartCTSPolarity is set to 'HIGH' then member generated with value 0U. If UartCTSPolarity is set to 'LOW' then member generated with value 1U.	
Example(s)	**Action**	**Generated output**

	• Configure UART channel with UartCTSPolarity = HIGH.	`/* RTS/CTS polarity if CTS is enabled */`
		`0U,`

	• Configure UART channel with UartCTSPolarity = LOW.	`/* RTS/CTS polarity if CTS is enabled */`
		`1U,`

1.2.2.15 Member: RxMode

Table 45 RxMode

Name	RxMode	
Type	uint8	
Description	This structure member value is used to configure the receive operation mode polling/interrupt.	
Verification method	The structure member is generated as per value configured in parameter 'UartChannel/UartRxChannelMode'. If UartRxChannelMode is set to 'INTERRUPT' then member is generated with value UART_INTERRUPT_MODE. If UartRxChannelMode is set to 'POLLING' then member is generated with value UART_POLLING_MODE.	
Example(s)	**Action**	**Generated output**

	• Configure UART channel with UartRxChannelMode = INTERRUPT.	`/* Receive operation mode polling/interrupt */`
		`UART_INTERRUPT_MODE,`

	• Configure UART channel with UartRxChannelMode = POLLING.	`/* Receive operation mode polling/interrupt */`
		`UART_POLLING_MODE,`

1.2.2.16 Member: TxMode

Table 46 TxMode

Name	TxMode	
Type	uint8	
Description	This structure member value is used to configure the transmit operation mode polling/interrupt.	
Verification method	The structure member is generated as per value configured in parameter 'UartChannel/UartTxChannelMode'. If UartTxChannelMode is set to 'INTERRUPT' then member is generated with value UART_INTERRUPT_MODE. If UartTxChannelMode is set to 'POLLING' then member is generated with value UART_POLLING_MODE.	
Example(s)	**Action**	**Generated output**

	• Configure UART channel with UartTxChannelMode = INTERRUPT.	`/* Transmit operation mode polling/interrupt */`
		`UART_INTERRUPT_MODE,`

	• Configure UART channel with UartTxChannelMode = POLLING.	`/* Transmit operation mode polling/interrupt */`
		`UART_POLLING_MODE,`

1.2.3 Array: Uart_ChannelIdLookup_<variant>[UART_MAX_HW_UNIT]

Table 47 Uart_ChannelIdLookup_<variant>[UART_MAX_HW_UNIT]

Name	Uart_ChannelIdLookup_<variant>[UART_MAX_HW_UNIT]	
Type	uint8	
Description	Array to maintain physical to logical channel mapping.	
Verification method	The generated file has this structure if at least one channel configured. <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. UART_MAX_HW_UNIT is number of hardware channel supported by device variant. Array member generated with logical channel index for which ASCLIN hardware is configured, if ASCLIN hardware is not configured then array index generated as 0xFFU.	
Example(s)	**Action**	**Generated output**

	• Configure 3 UART channels
	(variant unaware)
	(UartChannel_0, UartChannel_1, UartChannel_2)
	• UartChannel_0/ UartHwUnit = ASCLIN0
	• UartChannel_1/ UartHwUnit = ASCLIN1
	• UartChannel_2/ UartHwUnit = ASCLIN11
		`static const uint8 Uart_ChannelIdLookup[UART_MAX_HW_UNIT] =`
		`{`
		`    0U,`
		`    1U,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    2U`
		`};`

	Configure 3 UART channel (variant aware, Variant name is 'Petrol')
	(UartChannel_0, UartChannel_1, UartChannel_2)
	• UartChannel_0/ UartHwUnit = ASCLIN0
	• UartChannel_1/ UartHwUnit = ASCLIN1
	• UartChannel_2/ UartHwUnit = ASCLIN11
		`static const uint8 Uart_ChannelIdLookup_Petrol[UART_MAX_HW_UNIT] =`
		`{`
		`    0U,`
		`    1U,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    0xFFU,`
		`    2U`
		`};`

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

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

1.3.1 Structure: Uart_Config[_<variant>]

Table 48 Uart_Config[_<varaint>]

Name	Uart_Config[_<variant>]

Type	Uart_ConfigType	
Description	Declaration of root configuration structure of UART driver which will be used during initialization.	
Verification method	The generated structure is present in Uart[_<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 at least 1 Uart channel (variant-unaware)	`extern const Uart_ConfigType Uart_Config;`
	Configure at least 1 Uart channel (variant-aware. Variant name is 'Petrol')	`extern const Uart_ConfigType Uart_Config_Petrol;`

Revision history

Major changes since the last revision

Date	Version	Description
2020-11-02	1.0	• Released.
2020-11-02	0.1	• Removed UART_RX_POLLING_ENABLE, UART_TX_POLLING_ENABLE Macros and Added UART_RX_MODE, UART_TX_MODE macros. • Verification method and Example(s) are changed in UART_ASCLIN<x>, • UART_ASCLIN_REG_ADDR and UART_MAX_HW_UNIT macros. • Added UART_CSRREG_CLKSEL_CLC Macro. • Verification Example(s) updated for Autosar and Software version macros • Uart driver chapter moved from MC-ISAR_TC3xx_Config_Verification_Manual_CD.pdf to this document

Edition 2018-02-26
Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Doc_Number