User Manual

for S32K3 I2C Driver

Document Number: UM34I2CASRR21-11 Rev0000R3.0.0 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	Prepared for release RTD S32K3 3.0.0

Chapter 2

Introduction

This integration manual describes the integration requirements for I2c Driver for S32K3XX microcontrollers.

2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors:

- s32k310_mqfp100
- s32k310_lqfp48
- s32k311_mqfp100 / MWCT2015S_mqfp100
- s32k311_lqfp48
- s32k312_mqfp100 / MWCT2016S_mqfp100
- s32k312_mqfp172 / MWCT2016S_mqfp172
- s32k314_mqfp172
- s32k314_mapbga257
- s32k322_mqfp100 / MWCT2D16S_mqfp100
- s32k322_mqfp172 / MWCT2D16S_mqfp172
- s32k324_mqfp172 / MWCT2D17S_mqfp172
- s32k324_mapbga257

- s32k341_mqfp100
- s32k341_mqfp172
- s32k342_mqfp100
- s32k342_mqfp172
- s32k344_mqfp172
- s32k344_mapbga257
- s32k394_mapbga289
- s32k396_mapbga289
- s32k358_mqfp172
- s32k358_mapbga289
- s32k328_mqfp172
- s32k328_mapbga289
- s32k338_mqfp172
- s32k338_mapbga289
- s32k348_mqfp172
- s32k348_mapbga289
- s32m274_lqfp64
- s32m276_lqfp64

All of the above microcontroller devices are collectively named as S32K3.

Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.
- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Reference Manual	S32K3xx Reference Manual, Rev.6, Draft B, 01/2023
		S32K39 and S32K37 Reference Manual, Rev. 2 Draft A, 11/2022
		S32M27x Reference Manual, Rev.2, Draft A, - 02/2023
2	Datasheet	S32K3xx Data Sheet, Rev. 6, 11/2022
		S32K396 Data Sheet, Rev. 1.1 — 08/2022
		S32M2xx Data Sheet, Rev. 2 RC — 12/2022
3	Errata	S32K358_0P14E Mask Set Errata – Rev. 28, 9/2022
		S32K396_0P40E Mask Set Errata, Rev. DEC2022, 12/2022
		S32K311_0P98C Mask Set Errata, Rev. 6/March/2023, 3/2023
		S32K312: Mask Set Errata for Mask 0P09C, Rev. 25/April/2022
		S32K342: Mask Set Errata for Mask 0P97C, Rev. 10, 11/2022
		S32K3x4: Mask Set Errata for Mask 0P55A/1P55A, Rev. 14/Oct/2022

Chapter 3

Driver

3.1 Requirements

Requirements for this driver are detailed in the Autosar Driver Software Specification document (See Table Reference List).

For CDD: I2c is a Complex Device Driver (CDD), so there are no AUTOSAR requirements regarding this module.

It has vendor-specific requirements and implementation.

3.2 Driver Design Summary

The I2c driver is implemented as an Autosar complex device driver. It uses the LPI2c and FlexIO hardware peripheral which provides support for implementing the I2c protocol. The I2c driver implements both master and slave mode for LPI2c channels and only master for FlexIO channels. The driver offers a hardware independent API to the upper layer that can be used to configure the I2c and initiate synchronous and asynchronous data transfers. Asynchronous transfer for a master channel use interrupts. The slave operates only in asynchronous mode. Hardware and software settings can be configured using an Autosar standard configuration tool. The information required for an I2c data transfer will be configured in a data structure that will be sent as parameter to the API of the driver. The driver reports errors to the error manager as defined in AUTOSAR.

3.3 Hardware Resources

The hardware configured by the I2c driver is the same between derivatives execepted S32M276.

Physical I2c Channels: LPI2C_0, LPI2C_1, FlexIO.

Note: LPI2C_0 on S32K376 is not usable because it does not have pinouts.

3.4 Deviations from Requirements

The driver deviates from the I2C Driver software specification in some places. The table identifies the I2C requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the I2C Driver.

Term	Definition
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Below table identifies the I2C requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the driver.

Requirement	Status	Description	Notes
SWS_CDD_I2C_00047	N/S	The I2C module shall reset the interrupt flag at the end of the ISR (if not done automatically by hardware)	Replaced by CPR_RTD_00516.

3.5 Driver Limitations

For Lpi2c channel the driver doesn't support:

- Master mode:
  - Host request input can be used to control the start time of an I2c bus transfer.
  - Flexible receive data match can generate interrupt on data match and/or discard unwanted data.
  - Ultra Fast mode
  - Doesn't support 'bExpectNack' = TRUE
- Slave mode:
  - SMBus alert and general call address.
  - software-controllable ACK or NACK, with optional clock stretching on ACK/NACK bit
  - Configurable clock stretching

For FlexIO channel the driver doesn't support:

- Master mode:
  - Due to device limitations, it is not always possible to tell the difference between NACK reception and receiver overflow.
  - The driver does not support multi-master mode. It does not detect arbitration loss
- Slave mode: not supported.

3.6 Driver usage and configuration tips

3.6.0.1 Master mode

Master could transfer data:

- blocking by using I2c_SyncTransfer() function
- non-blocking by using I2c_AsyncTransfer() function

Master mode supports the following asynchronous methods:

- Interrupts
- DMA

3.6.0.1.1 Example of lpi2c master configuration

```
/* Channel configuration for channel LPI2C_0 - configured as master */
Lpi2c_Ip_MasterConfigType I2c_Lpi2cMasterChannel0_VS_0 =
{
/* Slave address - for HLD layer this field is changed at runtime */
0U,
/* 10-bit address - the transfer will use 7-bit transfer */
FALSE,
/* Operating Mode - the transfer is in standard mode */
LPI2C_STANDARD_MODE,
/* Baudrate parameters - the desired baudrate will be calculated based on this field */
```

```
&baudrateParams0_VS_0,
/* Pin Low Timeout - Pin Low Timeout will be deactivated */
0U,
/* Bus Idle Timeout - Bus Idle Timeout will be deactivated */
0U,
/* Glitch Filter SDA - glitch filter of 1 cycle */
1U,
/* Glitch Filter SDA - glitch filter of 1 cycle */
1U,
/* Master code - used in highspeed mode, in standard mode this field is ignored */
0U,
/* Transfer Type - interrupts will be used for asynchronous transfers*/
LPI2C_USING_INTERRUPTS,
/* Dma Tx Channel - it will be ignored if DMA is not used */
0U,
/* Dma Rx Channel - it will be ignored if DMA is not used */
0U,
/* Master Callback - this field will be updated at runtime in case callback is defined */
NULL_PTR,
/* Master Callback Parameter - represents the I2c logical channel */
0U,
/* State structure used internal by Lpi2c IP */
&Lpi2c_Ip_MasterState[0]
};
```

3.6.0.2 Slave mode

Slave mode supports the following asynchronous methods:

- Interrupts
- DMA

Slave could be used in listening mode or non-listening mode. When non-listening mode is used, I2c_StartListening() function should be called before every transfer.

3.6.0.2.1 Example of lpi2c slave configuration

```
/* Channel configuration for channel LPI2C_1 - configured as slave */
Lpi2c_Ip_SlaveConfigType I2c_Lpi2cSlaveChannel1Config_VS_0 =
{
/* Slave Address */
50U,
/* Selects 7-bit address */
false,
/* Slave mode - slave is in listening mode, no need to call I2c_StartListening() function */
true,
/* Operating Mode */
LPI2C_STANDARD_MODE,
/* Transfer Type */
LPI2C_USING_INTERRUPTS,
/* Glitch Filter SDA - glitch filter is deactivated */
0U,
/* Glitch Filter SCL -glitch filter is deactivated */
0U,
/* Dma Tx Channel - in interrupt mode this field is ignored */
0U,
/* Dma Rx Channel - in interrupt mode this field is ignored */
0U,
/* Slave Callback - this field will be updated at runtime in case callback is defined */
NULL_PTR,
/* Slave Callback Parameter - represents the I2c logical channel number */
1U,
&Lpi2c_Ip_SlaveState[0]
};
```

3.6.1 FLEXIO DMA Optimize

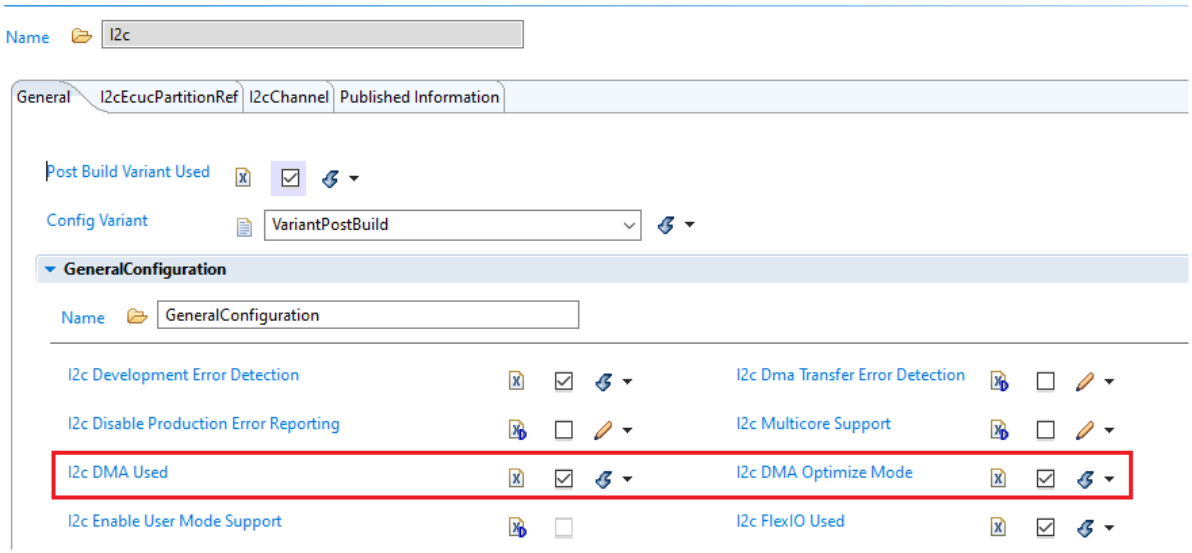
The Flexio I2C driver provides an optional configuration parameter for reducing the number of DMA interrupts required for transmission that are configured with DMA Optimize option. Instead of being interrupted after each end of transmitting or receiving a data block or data amount larger than 13 bytes, only one interrupt will be raised to stop frame and inform to user that the transmission was done. With this feature enabled, There are 2 optimizations will be implemented.

- Replace the function Flexio_I2c_Ip_MasterEndDmaTransfer() with DMA scatter-gather mode.
  - DMA normal: This function is a callback function after each DMA transfer (both transmitting or receiving).
    Master Send: Send STOP or Repeated START signal.
    Master Receive: Configure to generate NACK on the laste byte.
    Send STOP or Repeated START signal.
    Read the last byte.
  - DMA optimize: Remove this function. All activities in Flexio_I2c_Ip_MasterEndDmaTransfer() will be covered by TCD.
- Replace the function Flexio_I2c_Ip_TimerEventHandler with DMA having scatter-gather enabled after transferring 13 bytes.

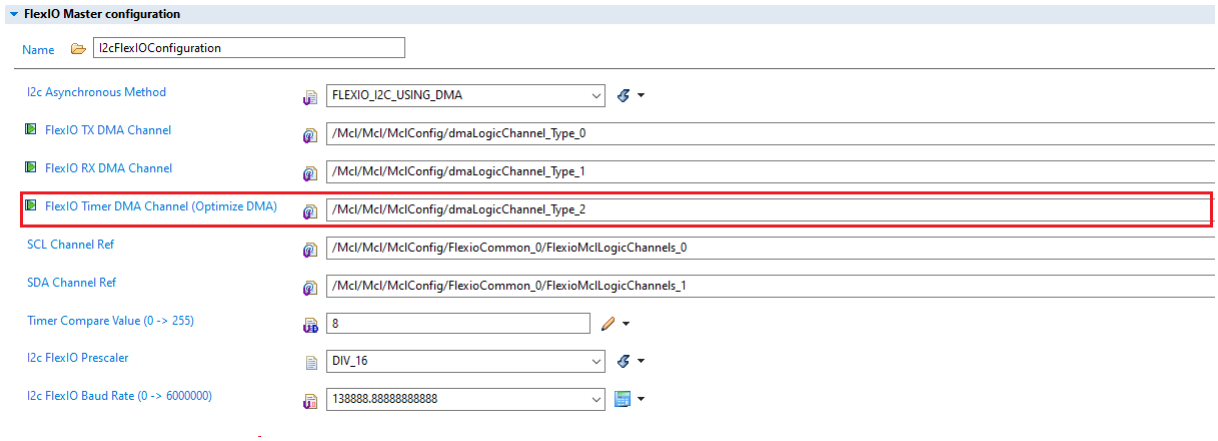
DMA scatter-gather mode will replace these functions to reduce CPU interference in DMA operation.

Configuration-time-prerequisites:

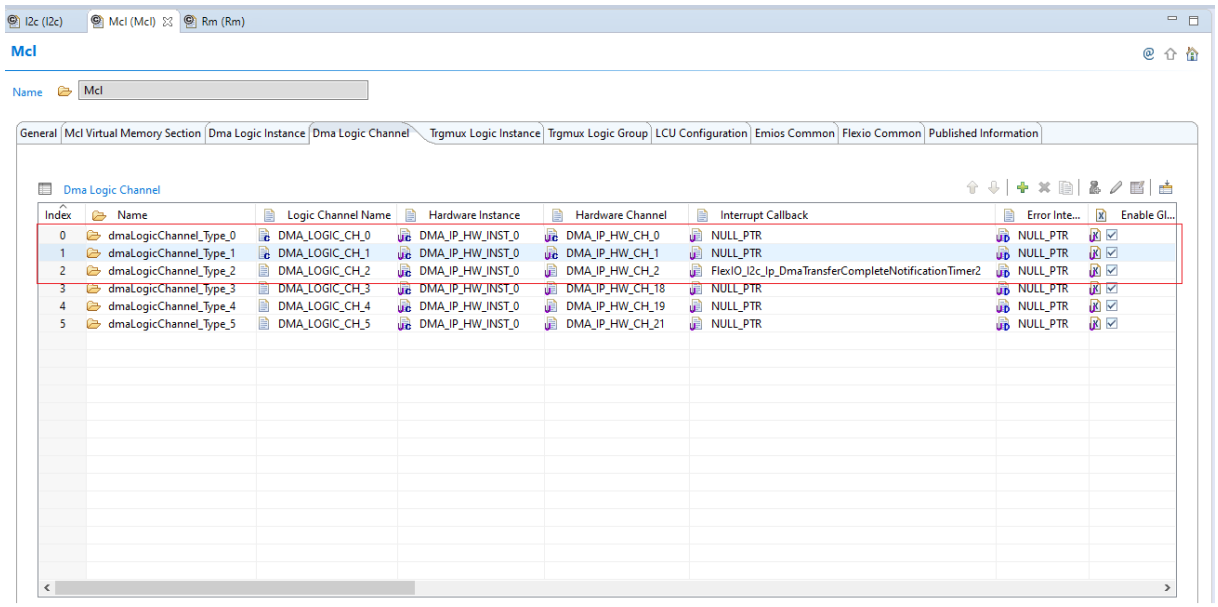
- Enable both I2c Enable DMA support and I2c Optimize DMA Feature(Flexio):



- Configure I2c Asynchronous Method as FLEXIO_I2C_USING_DMA type and select transferring DMA channel(Select channels for DMA: Tx,Rx,Timer Channel):



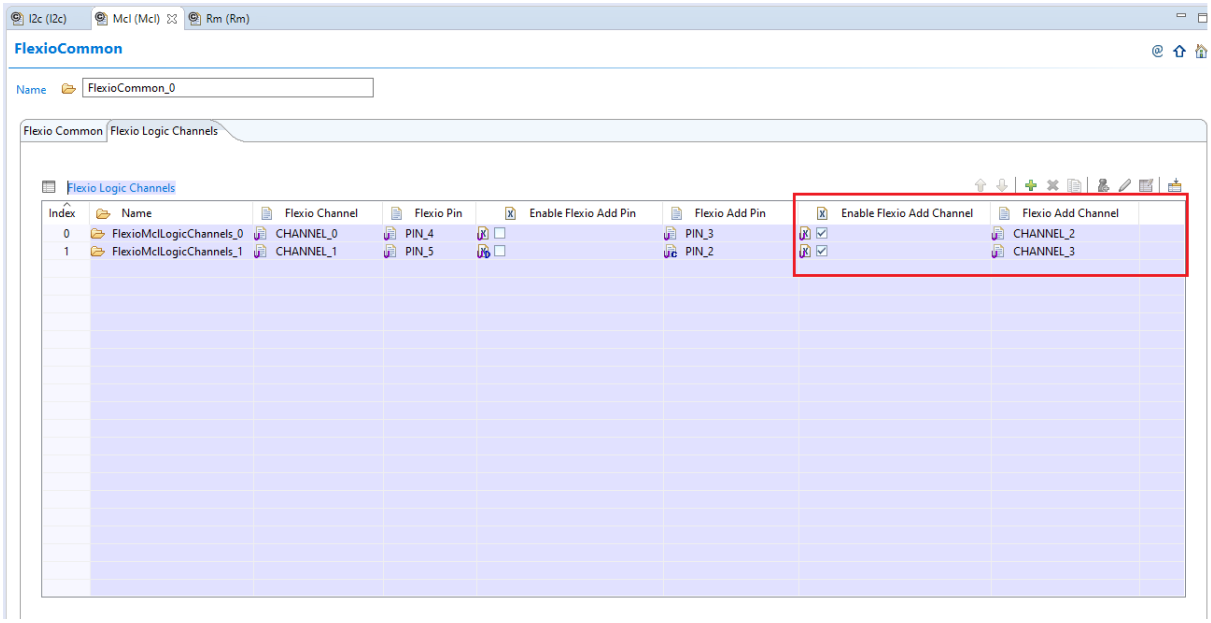
- Also in Mcl, we need to add 3 DMA channels:

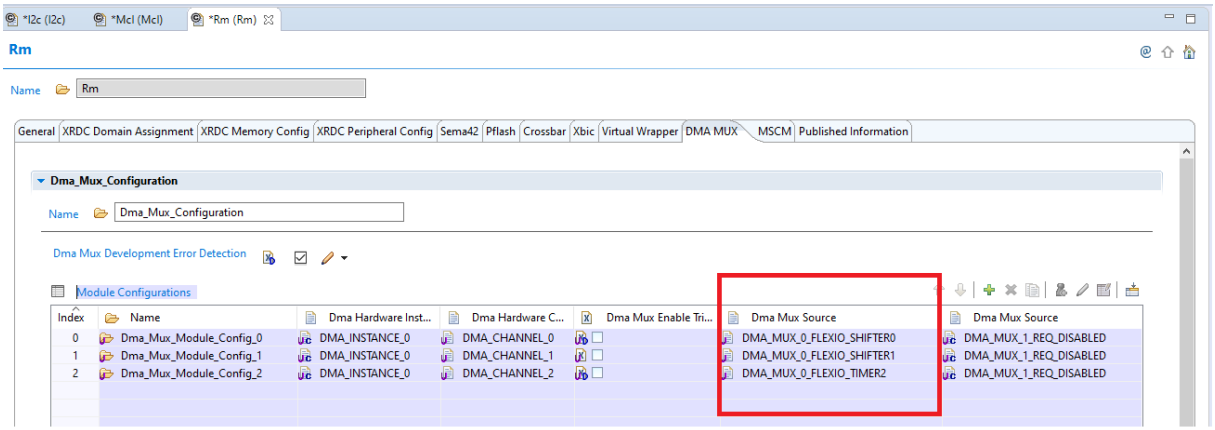


- When Optimize feature enabled, we need to select DMA mux source for Timer channel. Unforturnately, Timer(n) and Shifter(n) have the same DMA mux source so we need to add more channels to use for Timer (e.g.: Shifter(0-1) will be controled by Timer(2-3) and Shifter(4-5) will be controled by Timer(6-7))

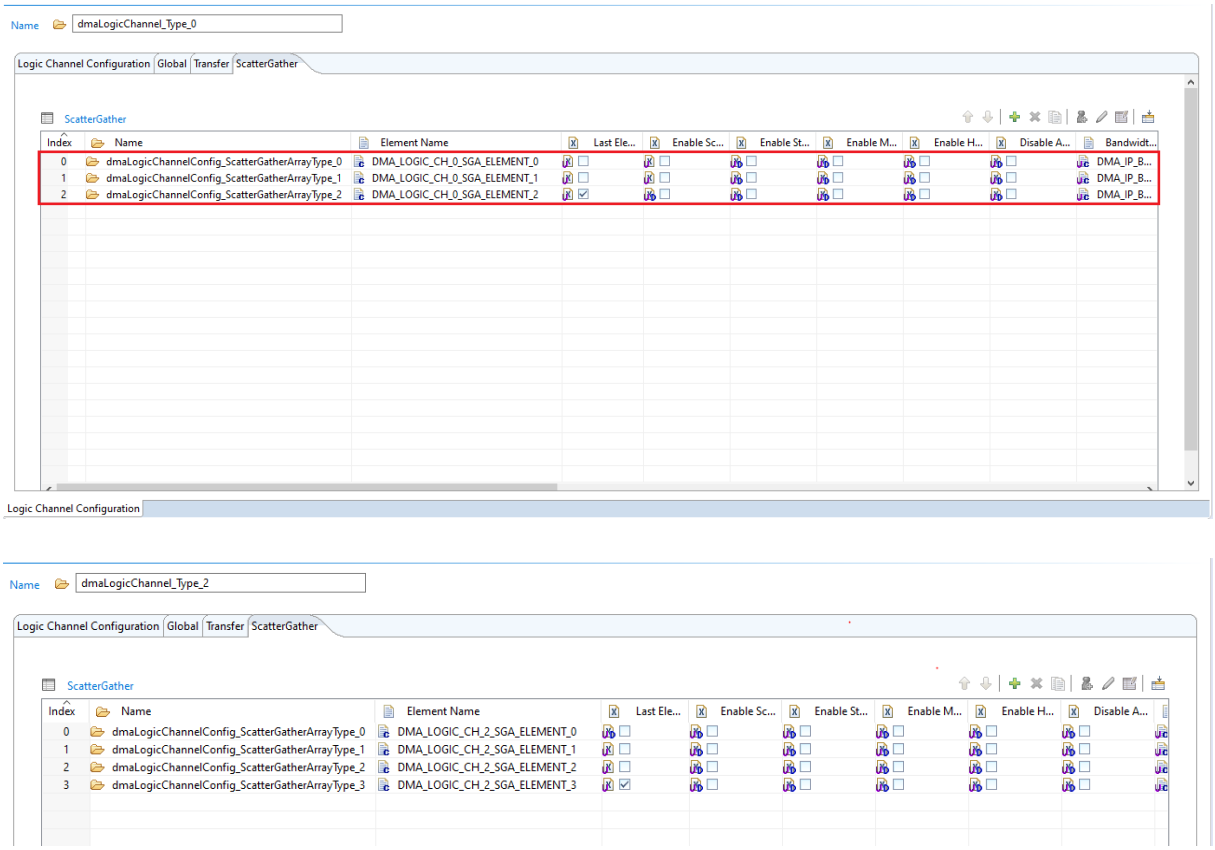
35	33	FlexIO Shifter0 DMA Request	FLEXIO	YES
36	33	FlexIO Timer0 DMA Request	FLEXIO	YES
37	34	FlexIO Shifter1 DMA Request	FLEXIO	YES
38	34	FlexIO Timer1 DMA Request	FLEXIO	YES
39	35	FlexIO Shifter2 DMA Request	FLEXIO	YES
40	35	FlexIO Timer2 DMA Request	FLEXIO	YES
41	36	FlexIO Shifter3 DMA Request	FLEXIO	YES
42	36	FlexIO Timer3 DMA Request	FLEXIO	YES

(i.e. from S32K3xx_DMAMUX_map.xlsx)





- Transferring DMA channel will be configured in Scatter-Gather mode for Tx channel (3 elements) and Timer channel (4 elements) when both transmitting and receiving:



3.7 Runtime errors

The driver generates the following DEM errors at runtime.

Function	Error Code	Condition triggering the error
I2c_SyncTransmit	I2C_E_TIMEOUT_FAILURE	Bus is busy when trying to send the slave address for a period of time larger than the configured timeout

3.8 Symbolic Names Disclaimer

All containers having symbolicNameValue set to TRUE in the AUTOSAR schema will generate defines like:

#define <Mip>Conf_<Container_ShortName>_<Container_ID>

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations or to use names that may trigger other compile issues (e.g. match existing `#ifdefs` arguments).

Chapter 4

Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module I2c
  - Container GeneralConfiguration
    - Parameter I2cDevErrorDetect
    - Parameter I2cDmaTransferErrorDetect
    - Parameter I2cDisableDemReportErrorStatus
    - Parameter I2cMulticoreSupport
    - Parameter I2cDmaUsed
    - Parameter I2cDmaOptimize
    - Parameter I2cEnableUserModeSupport
    - Parameter I2cFlexIOUsed
    - Parameter I2cTimeoutDuration
    - Parameter I2cTimeoutMethod
    - Parameter I2cVersionInfoApi
    - Parameter I2cCallback
    - Parameter I2cErrorCallback
  - Container I2cGlobalConfig
    - Reference I2cEcucPartitionRef
    - Container I2cFlexIOModuleConfiguration
      - Reference I2cClockRef
    - Container I2cChannel
      - Parameter I2cChannelId
      - Parameter I2cHwChannel
      - Parameter I2cMasterSlaveConfiguration
      - Parameter I2cOperatingMode
      - Reference I2cChannelEcucPartitionRef
      - Container I2cMasterConfiguration
      - Parameter I2cAsyncMethod
      - Parameter I2cPrescaler
      - Parameter I2cGlitchFilterSDA
      - Parameter I2cGlitchFilterSCL

- Parameter I2cPinLowTimeout
- Parameter I2cBusIdleTimeout
- Parameter I2cDataValidDelay
- Parameter I2cSetupHoldDelay
- Parameter I2cClockHighPeriod
- Parameter I2cClockLowPeriod
- Parameter I2cBaudRate
- Reference I2cClockRef
- Reference I2cDmaTxChannelRef
- Reference I2cDmaRxChannelRef
- Container I2cHighSpeedModeConfiguration
- Parameter I2cMasterCode
- Parameter I2cDataValidDelay
- Parameter I2cSetupHoldDelay
- Parameter I2cClockHighPeriod
- Parameter I2cClockLowPeriod
- Parameter I2cHighSpeedBaudRate
- Container I2cSlaveConfiguration
- Parameter I2cSlaveAddress
- Parameter I2cSlaveIs10BitAddress
- Parameter Lpi2cSlaveListening
- Parameter I2cAsyncMethod
- Parameter I2cSlaveFilterEnable
- Parameter I2cGlitchFilterSDA
- Parameter I2cGlitchFilterSCL
- Reference I2cSlaveDmaTxChannelRef
- Reference I2cSlaveDmaRxChannelRef
- Container I2cFlexIOConfiguration
- Parameter I2cAsyncMethod
- Parameter I2cFlexIOCompareValue
- Parameter I2cFlexIOChannelPrescaler
- Parameter I2cBaudRate
- Reference I2cDmaTxChannelRef
- Reference I2cDmaRxChannelRef
- Reference I2cTimerDmaRef
- Reference SclFlexioRef
- Reference SdaFlexioRef

* Container I2cDemEventParameterRefs
  - Reference I2C_E_TIMEOUT_FAILURE

– Container CommonPublishedInformation

* Parameter ArReleaseMajorVersion
* Parameter ArReleaseMinorVersion
* Parameter ArReleaseRevisionVersion
* Parameter ModuleId
* Parameter SwMajorVersion
* Parameter SwMinorVersion

* Parameter SwPatchVersion
* Parameter VendorApiInfix
* Parameter VendorId

4.1 Module I2c

Configuration of the Inter-integrated circuit (I2c) module.

Included containers:

- GeneralConfiguration
- I2cGlobalConfig
- CommonPublishedInformation

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantSupport	true
supportedConfigVariants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

4.2 Container GeneralConfiguration

GeneralConfiguration

This container contains the global configuration parameters of the Non-Autosar I2c driver.

Note: Implementation Specific Parameter.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.3 Parameter I2cDevErrorDetect

I2cDevErrorDetect

Switches the Development Error Detection and Notification ON or OFF.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.4 Parameter I2cDmaTransferErrorDetect

I2cDmaTransferErrorDetect

Switches the Dma transfer error of the i2c module ON or OFF.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.5 Parameter I2cDisableDemReportErrorStatus

I2cDisableDemReportErrorStatus

Switches the Diagnostic Error Reporting and Notification OFF.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.6 Parameter I2cMulticoreSupport

This parameter globally enables the possibility to support multicore.

If I2cMulticoreSupport is disabled, then for all the variants no partition shall be defined.

If I2cMulticoreSupport is enabled, at least one EcucPartition needs to be defined (in all variants).

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.7 Parameter I2cDmaUsed

I2cDmaUsed

Check this in order to be able to use DMA in the I2c driver.

Leaving this unchecked will allow the I2c driver to compile with no dependencies from the Mcl driver.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.8 Parameter I2cDmaOptimize

I2cDmaOptimizeMode

Check this in order to be able optimize feature for DMA in the I2c driver.

The Flexio I2C driver provides an optional configuration parameter for reducing the number of DMA interrupts required for transmission that are configured with DMA Optimize option. Instead of being interrupted after each end of transmitting or receiving a data block or data amount larger than 13 bytes, only one interrupt will be raised to stop frame and inform to user that the transmission was done. DMA scatter-gather mode will replace these functions to reduce CPU interference in DMA operation. For more details please refer to driver user manual.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.9 Parameter I2cEnableUserModeSupport

When this parameter is enabled, the I2C module will adapt to run from User Mode.

Note: I2C module does not include registers protection. So, It is accessible to all registers in any public mode.

I2C is not affected by this field.

For additional details, please refer to chapter '5.6 User Mode Support' in the IM.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.10 Parameter I2cFlexIOUsed

I2cFlexIOUsed

Check this in order to be able to use FlexIO channels.

Leaving this unchecked will allow the I2c driver to generate code without configuring the FlexIO module.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.11 Parameter I2cTimeoutDuration

Specifies the maximum number of loops for blocking function until a timeout is raised in short term wait loops

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1000
max	65535
min	1

4.12 Parameter I2cTimeoutMethod

Configures the timeout method.

Based on this selection a certain timeout method from OsIf will be used in the driver.

Note: If SystemTimer or CustomTimer are selected make sure the corresponding timer is enabled in OsIf General configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COU↩ NTER_CUSTOM']

4.13 Parameter I2cVersionInfoApi

I2cVersionInfoApi

Switches the I2c_GetVersionInfo function ON or OFF.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.14 Parameter I2cCallback

I2c general callback. This function will be called for all i2c events.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	I2c_Callback

4.15 Parameter I2cErrorCallback

I2c error callback. This function will be called for i2c error events.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	I2c_ErrorCallback

4.16 Container I2cGlobalConfig

This container contains the global configuration parameter of the I2c driver. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exit once per configuration set.

Included subcontainers:

- I2cFlexIOModuleConfiguration

- I2cChannel
- I2cDemEventParameterRefs

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.17 Reference I2cEcucPartitionRef

Maps the I2C driver to zero or multiple ECUC partitions to make the

modules API available in this partition. The I2C driver will operate as

an independent instance in each of the partitions.

Tags: atp.Status=draft

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.18 Container I2cFlexIOModuleConfiguration

This container contains the configuration (parameters) of the Master Configuration.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.19 Reference I2cClockRef

Reference to the FlexIO clock source configuration, which is set in the MCU driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↩ Config/McuClockReferencePoint

4.20 Container I2cChannel

This container contains the configuration (parameters) of the I2c Controller(s).

Note:"User should use unique names for naming the I2c channels across different I2cGlobalConfig Sets."

Included subcontainers:

- I2cMasterConfiguration
- I2cSlaveConfiguration
- I2cFlexIOConfiguration

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	6
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE

4.21 Parameter I2cChannelId

Identifies the I2c channel.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	6
min	0

4.22 Parameter I2cHwChannel

Selects the physical I2c Channel.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	LPI2C_0
literals	['LPI2C_0', 'LPI2C_1', 'FLEXIO_0_CH_0_1', 'FLEXIO_0_CH_2_3', 'FL↩ EXIO_0_CH_4_5', 'FLEXIO_0_CH_6_7']

4.23 Parameter I2cMasterSlaveConfiguration

Selects the master slave configuration.

Select wether the selected channel will be used as Master, Slave or both.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	MASTER_MODE
literals	['MASTER_MODE', 'SLAVE_MODE']

4.24 Parameter I2cOperatingMode

Selects the pin configuration.

Configures the pin mode.

000b - LPI2C configured for 2-pin open drain mode (Master and Slave using SDA/SCL).

001b - LPI2C configured for 2-pin output only mode in UFM (NOT SUPPORTED!).

010b - LPI2C configured for 2-pin push-pull mode (Master and Slave using SDA/SCL).

100b - LPI2C configured for 2-pin open drain mode with separate LPI2C slave (Master using SDA/SCL, Slave using SDAS/SCLS).

110b - LPI2C configured for 2-pin push-pull mode with separate LPI2C slave (Master using SDA/SCL, Slave using SDAS/SCLS).

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	LPI2C_STANDARD_MODE
literals	['LPI2C_STANDARD_MODE', 'LPI2C_FAST_MODE', 'LPI2C_FASTPL↩ US_MODE', 'LPI2C_HIGHSPEED_MODE']

4.25 Reference I2cChannelEcucPartitionRef

Maps one single I2C channel to zero or one ECUC partitions.

The ECUC partition referenced is a subset of the ECUC partitions

where the I2C driver is mapped to.

Tags: atp.Status=draft

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	true

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.26 Container I2cMasterConfiguration

This container contains the configuration (parameters) of the Master Configuration.

Included subcontainers:

- I2cHighSpeedModeConfiguration

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.27 Parameter I2cAsyncMethod

Configures the asynchronous mechanism used by the 'AsyncTransmit' function (interrupts or DMA).

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	LPI2C_USING_INTERRUPTS
literals	['LPI2C_USING_INTERRUPTS', 'LPI2C_USING_DMA']

4.28 Parameter I2cPrescaler

PRESCALE: Configures LPI2C_MCFGR1[PRESCALE]

Configures the clock prescaler used for all LPI2C master logic, except the digital glitch filters.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	LPI2C_MASTER_PRESC_DIV_1
literals	['LPI2C_MASTER_PRESC_DIV_1', 'LPI2C_MASTER_PRESC_DIV_2', 'LPI2C_MASTER_PRESC_DIV_4', 'LPI2C_MASTER_PRESC_DIV_8', 'LPI2C_MASTER_PRESC_DIV_16', 'LPI2C_MASTER_PRESC_DIV_32', 'LPI2C_MASTER_PRESC_DIV_64', 'LPI2C_MASTER_PRESC_DIV_128']

4.29 Parameter I2cGlitchFilterSDA

Glitch Filter SDA: Configures LPI2C_MCFGR2[FILTSDA]

Configures the I2c master digital glitch filters for SDA input, a configuration of 0 will disable the glitch filter.

Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored. The latency through the glitch filter is equal to FILTSDA cycles and must be configured less than the minimum SCL low or high period.

The glitch filter cycle count is not affected by the PRESCALE configuration and is automatically bypassed in High Speed mode.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.30 Parameter I2cGlitchFilterSCL

Glitch Filter SCL: Configures LPI2C_MCFGR2[FILTSCL]

Configures the I2c master digital glitch filters for SCL input, a configuration of 0 will disable the glitch filter.

Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored. The latency through the glitch filter is equal to FILTSCL cycles and must be configured less than the minimum SCL low or high period.

The glitch filter cycle count is not affected by the PRESCALE configuration and is automatically bypassed in High Speed mode.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.31 Parameter I2cPinLowTimeout

Pin Low Timeout: Configures LPI2C_MCFGR3[PINLOW]

Configures the pin low timeout flag in clock cycles. If SCL and/or SDA is low for longer than (PINLOW * 256) cycles then PLTF is set. When set to zero, this feature is disabled.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4095
min	0

4.32 Parameter I2cBusIdleTimeout

Bus Idle Timeout: Configures LPI2C_MCFGR2[BUSIDLE]

Configures the bus idle timeout period in clock cycles. If both SCL and SDA are higher than BUSIDLE cycles, then the I2C bus is assumed to be idle and the master can generate a START condition. When set to zero, this feature is disabled.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4095
min	0

4.33 Parameter I2cDataValidDelay

Data Valid Delay: Configures LPI2C_MCCR0[DATAVD]

Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	63
min	1

4.34 Parameter I2cSetupHoldDelay

Setup Hold Delay: Configures LPI2C_MCCR0[SETHOLD]

Minimum number of cycles (minus one) that is used by the master as the setup and hold time for a

(repeated) START condition and setup time for a STOP condition.

The setup time is extended by the time it takes to detect a rising edge on the external SCL pin.

Ignoring any additional board delay due to external loading, this is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	63
min	2

4.35 Parameter I2cClockHighPeriod

Clock High Period: Configures LPI2C_MCCR0[CLKHI]

Minimum number of cycles (minus one) that the SCL clock is driven high by the master.

The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin.

Ignoring any additional board delay due to external loading, this is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	63
min	1

4.36 Parameter I2cClockLowPeriod

Clock Low Period: Configures LPI2C_MCCR0[CLKLO]

Minimum number of cycles (minus one) that the SCL clock is driven low by the master.

This value is also used for the minimum bus free time between a STOP and a START condition.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	63
min	3

4.37 Parameter I2cBaudRate

Calculated as Frequency/(((CLKLO+CLKHI+2)*2^PRESCALER)+ROUNDDOWN((2+FILTSCL)/2^PRESCALER))

The functional clock must be at least 8 times faster than the I2c bus bandwidth.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0.0
max	6000000.0
min	0.0

4.38 Reference I2cClockRef

Reference to the I2c clock source configuration, which is set in the MCU driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↩ Config/McuClockReferencePoint

4.39 Reference I2cDmaTxChannelRef

Reference to the DMA TX channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type

4.40 Reference I2cDmaRxChannelRef

Reference to the DMA RX channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type

4.41 Container I2cHighSpeedModeConfiguration

This container contains the configuration (parameters) of the Master Clock Configuration Register 1.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.42 Parameter I2cMasterCode

Master code

Master code used in high speed mode. Should be in range 0-7.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	7
min	0

4.43 Parameter I2cDataValidDelay

Data Valid Delay: Configures LPI2C_MCCR1[DATAVD]

Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	63
min	1

4.44 Parameter I2cSetupHoldDelay

Setup Hold Delay: Configures LPI2C_MCCR1[SETHOLD]

Minimum number of cycles (minus one) that is used by the master as the setup and hold time for a (repeated) START condition and setup time for a STOP condition.

The setup time is extended by the time it takes to detect a rising edge on the external SCL pin.

Ignoring any additional board delay due to external loading, this is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	63
min	2

4.45 Parameter I2cClockHighPeriod

Clock High Period: Configures LPI2C_MCCR1[CLKHI]

Minimum number of cycles (minus one) that the SCL clock is driven high by the master.

The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin.

Ignoring any additional board delay due to external loading, this is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	63
min	1

4.46 Parameter I2cClockLowPeriod

Clock Low Period: Configures LPI2C_MCCR1[CLKLO]

Minimum number of cycles (minus one) that the SCL clock is driven low by the master.

This value is also used for the minimum bus free time between a STOP and a START condition.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	3
max	63
min	3

4.47 Parameter I2cHighSpeedBaudRate

Calculated as Frequency/((((CLKLO+CLKHI+2)*2^PRESCALER)+ROUNDDOWN((2+FILTSCL)/2^PRESCALER))

The functional clock must be at least 8 times faster than the I2c bus bandwith.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0.0
max	6000000.0
min	0.0

4.48 Container I2cSlaveConfiguration

This container contains the configuration (parameters) of the Slave Channel.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.49 Parameter I2cSlaveAddress

The address of the slave: Configures LPI2C_SAMR[ADDR0]

Configures the I2c slave address.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	1023
min	0

4.50 Parameter I2cSlaveIs10BitAddress

I2cSlaveDisableFilterInDoze configures LPI2C_SCR[FILTDZ]

Check to disable the filter in Doze mode. Uncheck to have the filters enabled in Doze mode.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.51 Parameter Lpi2cSlaveListening

I2cSlaveListening

Check this in order to chose slave mode (always listening or on demand only).

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.52 Parameter I2cAsyncMethod

Configures the asynchronous mechanism used by the 'AsyncTransmit' function (interrupts or DMA).
Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	LPI2C_USING_INTERRUPTS
literals	['LPI2C_USING_INTERRUPTS', 'LPI2C_USING_DMA']

4.53 Parameter I2cSlaveFilterEnable

I2cSlaveFilterEnable configures LPI2C_SCR[FILTEN]

Check to enable the digital filter and output delay counter for slave mode.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.54 Parameter I2cGlitchFilterSDA

Glitch Filter SDA: Configures LPI2C_MCFGR2[FILTSDA]

Configures the I2c master digital glitch filters for SDA input, a configuration of 0 will disable the glitch filter.

Glitches equal to or less than FILTSDA cycles long will be filtered out and ignored. The latency through the glitch filter is equal to FILTSDA cycles and must be configured less than the minimum SCL low or high period.

The glitch filter cycle count is not affected by the PRESCALE configuration and is automatically bypassed in High Speed mode.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.55 Parameter I2cGlitchFilterSCL

Glitch Filter SCL: Configures LPI2C_MCFGR2[FILTSCL]

Configures the I2c master digital glitch filters for SCL input, a configuration of 0 will disable the glitch filter.

Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored. The latency through the glitch filter is equal to FILTSCL cycles and must be configured less than the minimum SCL low or high period.

The glitch filter cycle count is not affected by the PRESCALE configuration and is automatically bypassed in High Speed mode.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.56 Reference I2cSlaveDmaTxChannelRef

Reference to the DMA TX channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Mcl/MclConfig/dmaLogicChannel_Type

4.57 Reference I2cSlaveDmaRxChannelRef

Reference to the DMA RX channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
~~multiplicityConfigClasses~~	VARIANT-POST-BUILD: POST-BUILD

Property	Value
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Mcl/MclConfig/dmaLogicChannel_Type

4.58 Container I2cFlexIOConfiguration

This container contains the configuration (parameters) of the Master Configuration.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.59 Parameter I2cAsyncMethod

Configures the asynchronous mechanism used by the 'AsyncTransmit' function (interrupts or DMA).

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	FLEXIO_I2C_USING_INTERRUPTS
literals	['FLEXIO_I2C_USING_INTERRUPTS', 'FLEXIO_I2C_USING_DMA']

4.60 Parameter I2cFlexIOCompareValue

This configures FLEXIO_TIMCMPa[CMP[7:0]]

This is used to calculate the Baud rate of the I2c. The baud rate divider is equal to (CMP[7:0] + 1) * 2.

The divider will be (input_clock + desired_baud_rate) div (2* desired_baud_rate) - 2. The extra -1 is from the timer reset setting used for clock stretching. Round to nearest integer.

This must be manually inserted, and the baud rate will be calculated based on it.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8
max	255
min	0

4.61 Parameter I2cFlexIOChannelPrescaler

PRESCALE: Configures FLEXIO_TIMCFGn[TIMDEC]

Configures the clock prescaler used for Timer Decrement.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	DIV_1
literals	['DIV_1', 'DIV_16', 'DIV_256']

4.62 Parameter I2cBaudRate

The FlexIO baud rate is calculated as:

'I2cFlexIOModuleConfiguration/I2cClockRef' / (2 * (I2cFlexIOCompareValue + 9)), when PRESCALER = DIV_1

('I2cFlexIOModuleConfiguration/I2cClockRef' / 16) / (2 * (I2cFlexIOCompareValue + 1)), when PRESCALER = DIV_16

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0.0
max	6000000.0
min	0.0

4.63 Reference I2cDmaTxChannelRef

Reference to the DMA TX channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP

Property	Value
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Mcl/MclConfig/dmaLogicChannel_Type

4.64 Reference I2cDmaRxChannelRef

Reference to the FLEXIO logic channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Mcl/MclConfig/dmaLogicChannel_Type

4.65 Reference I2cTimerDmaRef

When I2cDmaOptimize is enabled, this additional DMA channel is used to reduce CPU load for transmissions with size larger than 13 bytes of data. The DMA channel is used to replace functionality of Flexio_I2c_Ip_TimerEventHandler. Instead of reconfiguring the timer value from interrupt, after each set of 14 bytes of data transmitted, the driver will use this additional DMA channel.For more details refer to driver user manual.

Property	Value
type	ECUC-REFERENCE-DEF

Property	Value
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Mcl/MclConfig/dmaLogicChannel_Type

4.66 Reference SclFlexioRef

Reference to the FLEXIO logic channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels

4.67 Reference SdaFlexioRef

Reference to the DMA RX channel, which is set in the Mcl driver configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels

4.68 Container I2cDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.69 Reference I2C_E_TIMEOUT_FAILURE

Reference to the DemEventParameter which shall be issued when the error "Timeout caused by hardware error" has occured.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	true
destination	/AUTOSAR/EcucDefs/Dem/DemConfigSet/DemEventParameter

4.70 Container CommonPublishedInformation

Common container, aggregated by all modules.

It contains published information about vendor and versions.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.71 Parameter ArReleaseMajorVersion

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.72 Parameter ArReleaseMinorVersion

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

4.73 Parameter ArReleaseRevisionVersion

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION

Property	Value
defaultValue	0
max	0
min	0

4.74 Parameter ModuleId

Module ID of this module from Module List.

Note: Implementation Specific Parameter

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	255
max	255
min	255

4.75 Parameter SwMajorVersion

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

Note: Implementation Specific Parameter

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	3
max	3
min	3

4.76 Parameter SwMinorVersion

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Note: Implementation Specific Parameter

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.77 Parameter SwPatchVersion

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

Note: Implementation Specific Parameter

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.78 Parameter VendorApiInfix

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name.

This parameter is used to specify the vendor specific name. In total, the Implementation specific name is generated as follows:

<ModuleName>_>VendorId>_<VendorApiInfix>.

E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name

Can_Write defined in the SWS will translate to Can_123_v11r456Write.

This parameter is mandatory for all modules with upper multiplicity >

1. It shall not be used for modules with upper multiplicity =1.

Note: Implementation Specific Parameter

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.79 Parameter VendorId

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

Note: Implementation Specific Parameter

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Chapter 6

Module Documentation

6.1 Lpi2c Driver

6.1.1 Detailed Description

6.1.1.1 General information

The I2C module provides a simple and efficient method of data exchange between a chip and other devices, such as microcontrollers, EEPROM, real-time clock devices, analog-to-digital converters and LCDs.

The advantages of the I2C bus is that it minimizes interconnections between devices, allows the connection of additional devices to the bus, includes collision detection and arbitration that prevent data corruption and it doesn't require an external address decoder.

6.1.1.2 Features

- Interrupt based
- Master or slave operation
- Provides blocking and non-blocking transmit and receive functions
- 7-bit or 10-bit addressing
- Configurable baud rate
- Provides support for all operating modes supported by the hardware
  - Standard-mode (Sm): bidirectional data transfers up to 100 kbit/s
  - Fast-mode (Fm): bidirectional data transfers up to 400 kbit/s

6.1.1.2.1 Master Mode Master Mode provides functions for transmitting or receiving data to/from any I2C slave. Slave address and baud rate are provided at initialization time through the master configuration structure, but they can be changed at runtime by using LPI2C_Ip_MasterSetBaudRate() or LPI2C_Ip_MasterSetSlaveAddr().

To send or receive data to/from the currently configured slave address, use functions Lpi2c_Ip_MasterSendData() or Lpi2c_Ip_MasterReceiveData() (or their blocking counterparts). Parameter `sendStop` can be used to chain multiple transfers with repeated START condition between them, for example when sending a command and then immediately receiving a response. The application should ensure that any send or receive transfer with `sendStop` set to `false` is followed by another transfer, otherwise the LPI2C master will hold the SCL line low indefinitely and block the I2C bus. The last transfer from a chain should always have `sendStop` set to `true`.

Blocking operations will return only when the transfer is completed, either successfully or with error. Non-blocking operations will initiate the transfer and return STATUS_SUCCESS, but the module is still busy with the transfer and another transfer can't be initiated until the current transfer is complete. The application can check the status of the current transfer by calling Lpi2c_Ip_MasterGetTransferStatus(). If the transfer is completed, the functions will return either STATUS_SUCCESS or an error code, depending on the outcome of the last transfer.

6.1.1.2.2 Slave Mode Slave Mode provides functions for transmitting or receiving data to/from any I2C master. The slave is always in listening mode. To check the status Lpi2c_Ip_SlaveGetTransferStatus should be called.

Data Structures

- struct Lpi2c_Ip_BaudRateType
  Baud rate structure. *More...*
- struct Lpi2c_Ip_MasterStateType
  Master internal context structure. *More...*
- struct Lpi2c_Ip_MasterConfigType
  Master configuration structure. *More...*
- struct Lpi2c_Ip_SlaveStateType
  Slave internal context structure. *More...*
- struct Lpi2c_Ip_SlaveConfigType
  Slave configuration structure. *More...*

Macros

- #define LPI2C_IP_MASTER_DATA_MATCH_INT
- #define LPI2C_IP_MASTER_PIN_LOW_TIMEOUT_INT
- #define LPI2C_IP_MASTER_FIFO_ERROR_INT
- #define LPI2C_IP_MASTER_ARBITRATION_LOST_INT
- #define LPI2C_IP_MASTER_NACK_DETECT_INT
- #define LPI2C_IP_MASTER_STOP_DETECT_INT
- #define LPI2C_IP_MASTER_END_PACKET_INT
- #define LPI2C_IP_MASTER_RECEIVE_DATA_INT
- #define LPI2C_IP_MASTER_TRANSMIT_DATA_INT
- #define LPI2C_IP_SLAVE_SMBUS_ALERT_RESPONSE_INT
- #define LPI2C_IP_SLAVE_GENERAL_CALL_INT
- #define LPI2C_IP_SLAVE_ADDRESS_MATCH_1_INT

- #define LPI2C_IP_SLAVE_ADDRESS_MATCH_0_INT
- #define LPI2C_IP_SLAVE_FIFO_ERROR_INT
- #define LPI2C_IP_SLAVE_BIT_ERROR_INT
- #define LPI2C_IP_SLAVE_STOP_DETECT_INT
- #define LPI2C_IP_SLAVE_REPEATED_START_INT
- #define LPI2C_IP_SLAVE_TRANSMIT_ACK_INT
- #define LPI2C_IP_SLAVE_ADDRESS_VALID_INT
- #define LPI2C_IP_SLAVE_RECEIVE_DATA_INT
- #define LPI2C_IP_SLAVE_TRANSMIT_DATA_INT

Enum Reference

- enum Lpi2c_Ip_SlaveEventType

  Define the enum of the events which can trigger I2C slave callback.
- enum Lpi2c_Ip_MasterEventType

  Define the enum of the events which can trigger I2C master callback.
- enum Lpi2c_Ip_PinConfigType

  Pin Configuration selection.
- enum Lpi2c_Ip_NackConfigType

  Master NACK reaction configuration.
- enum Lpi2c_Ip_SlaveAddressConfigType

  Slave address configuration.
- enum Lpi2c_Ip_SlaveNackConfigType

  Slave NACK reaction configuration.
- enum Lpi2c_Ip_SlaveNackTransmitType

  Slave ACK transmission options.
- enum Lpi2c_Ip_ModeType

  I2C operating modes.
- enum Lpi2c_Ip_AsyncTransferType

  Type of LPI2C transfer (based on interrupts or DMA).
- enum Lpi2c_Ip_StatusType

  Type of LPI2C transfer (based on interrupts or DMA).
- enum Lpi2c_Ip_MasterPrescalerType

  Defines the example structure.
- enum Lpi2c_Ip_DirectionType

LPI2C Driver

- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterInit (uint8 Instance, const Lpi2c_Ip_MasterConfigType *ConfigPtr)

  Initialize the LPI2C master mode driver.
- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterDeinit (uint8 Instance)

  De-initialize the LPI2C master mode driver.
- void Lpi2c_Ip_MasterGetBaudRate (uint8 Instance, uint32 InputClock, uint32 *BaudRate)

  Get the currently configured baud rate.
- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterSetBaudRate (uint8 Instance, Lpi2c_Ip_ModeType OperatingMode, uint32 Baudrate, uint32 InputClock)

Set the baud rate for any subsequent I2C communication.

- void Lpi2c_Ip_MasterSetSlaveAddr (uint8 Instance, const uint16 Address, const boolean Is10bitAddr)

  Set the slave address for any subsequent I2C communication.
- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterSendData (uint8 Instance, uint8 *TxBuff, uint32 TxSize, boolean SendStop)

  Perform a non-blocking send transaction on the I2C bus.
- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterSendDataBlocking (uint8 Instance, uint8 *TxBuff, uint32 TxSize, boolean SendStop, uint32 Timeout)

  Perform a blocking send transaction on the I2C bus.
- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterReceiveData (uint8 Instance, uint8 *RxBuff, uint32 RxSize, boolean SendStop)

  Perform a non-blocking receive transaction on the I2C bus.
- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterReceiveDataBlocking (uint8 Instance, uint8 *RxBuff, uint32 RxSize, boolean SendStop, uint32 Timeout)

  Perform a blocking receive transaction on the I2C bus.
- Lpi2c_Ip_StatusType Lpi2c_Ip_MasterGetTransferStatus (uint8 Instance, uint32 *BytesRemaining)

  Return the current status of the I2C master transfer.
- void Lpi2c_Ip_MasterIRQHandler (uint8 Instance)

  Handle master operation when I2C interrupt occurs.
- Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveInit (uint8 Instance, const Lpi2c_Ip_SlaveConfigType *ConfigPtr)

  Initialize the I2C slave mode driver.
- Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveDeinit (uint8 Instance)

  De-initialize the I2C slave mode driver.
- Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveSetBuffer (uint8 Instance, uint8 *DataBuff, uint32 DataSize)

  Provide a buffer for transmitting data.
- Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveGetTransferStatus (uint8 Instance, uint32 *BytesRemaining)

  Return the current status of the I2C slave transfer.
- void Lpi2c_Ip_SlaveIRQHandler (uint8 Instance)

  Handle slave operation when I2C interrupt occurs.
- void Lpi2c_Ip_ModuleIRQHandler (uint8 Instance)

  Handler for both slave and master operation when I2C interrupt occurs.
- void Lpi2c_Ip_SetMasterCallback (uint8 Instance, Lpi2c_Ip_MasterCallbackType MasterCallback)

  Sets the master callback.
- void Lpi2c_Ip_SetSlaveCallback (uint8 Instance, Lpi2c_Ip_SlaveCallbackType SlaveCallback)

  Sets the slave callback.
- void Lpi2c_Ip_StartListening (uint8 Instance)

  Start listening.
- void Lpi2c_Ip_SetMasterHighSpeedMode (uint8 Instance, boolean HighSpeedEnabled)

  Set high speed mode for master.
- #define I2C_START_SEC_CODE
- #define **I2C_STOP_SEC_CODE**

Configuration

- void Lpi2c_Ip_Init (LPI2C_Type *BaseAddr)

  Initializes the LPI2C module to a known state.
- #define **I2C_STOP_SEC_CODE**

6.1.2 Data Structure Documentation

6.1.2.1 struct Lpi2c_Ip_BaudRateType

Baud rate structure.

This structure is used for setting or getting the baud rate.

Definition at line 208 of file Lpi2c_Ip_Types.h.

6.1.2.2 struct Lpi2c_Ip_MasterStateType

Master internal context structure.

This structure is used by the master-mode driver for its internal logic. It must be provided by the application through the LPI2C_DRV_MasterInit() function, then it cannot be freed until the driver is de-initialized using LPI2C_DRV_MasterDeinit(). The application should make no assumptions about the content of this structure.

Definition at line 262 of file Lpi2c_Ip_Types.h.

6.1.2.3 struct Lpi2c_Ip_MasterConfigType

Master configuration structure.

This structure is used to provide configuration parameters for the LPI2C master at initialization time.

Definition at line 294 of file Lpi2c_Ip_Types.h.

Data Fields

- uint16 SlaveAddress
- boolean Is10bitAddr
- Lpi2c_Ip_ModeType OperatingMode
- const Lpi2c_Ip_BaudRateType * BaudrateParams
- uint32 PinLowTimeout
- uint32 BusIdleTimeout
- uint32 GlitchFilterSDA
- uint32 GlitchFilterSCL
- uint8 MasterCode
- Lpi2c_Ip_AsyncTransferType TransferType
- uint32 DmaTxChannel
- uint32 DmaRxChannel
- Lpi2c_Ip_MasterCallbackType MasterCallback
- uint8 CallbackParam
- uint8 MasterStateIdx

6.1.2.3.1 Field Documentation

6.1.2.3.1.1 SlaveAddress `uint16 SlaveAddress`

Slave address, 7-bit or 10-bit

Definition at line 296 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.2 Is10bitAddr `boolean Is10bitAddr`

Selects 7-bit or 10-bit slave address

Definition at line 297 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.3 OperatingMode `Lpi2c_Ip_ModeType OperatingMode`

I2C Operating mode

Definition at line 298 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.4 BaudrateParams `const Lpi2c_Ip_BaudRateType* BaudrateParams`

Baud rate in Hz

Definition at line 299 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.5 PinLowTimeout `uint32 PinLowTimeout`

Pin Low Timeout

Definition at line 300 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.6 BusIdleTimeout `uint32 BusIdleTimeout`

Bus Idle Timeout

Definition at line 301 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.7 GlitchFilterSDA `uint32 GlitchFilterSDA`

SDA glitch filter

Definition at line 302 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.8 GlitchFilterSCL `uint32 GlitchFilterSCL`

SCL glitch filter

Definition at line 303 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.9 MasterCode `uint8 MasterCode`

Master code for High-speed mode. Valid range: 0-7. Unused in other operating modes

Definition at line 304 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.10 TransferType `Lpi2c_Ip_AsyncTransferType TransferType`

Type of LPI2C transfer

Definition at line 305 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.11 DmaTxChannel `uint32 DmaTxChannel`

Channel number for DMA Tx channel. If DMA mode isn't used this field will be ignored.

Definition at line 306 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.12 DmaRxChannel `uint32 DmaRxChannel`

Channel number for DMA Rx channel. If DMA mode isn't used this field will be ignored.

Definition at line 307 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.13 MasterCallback `Lpi2c_Ip_MasterCallbackType MasterCallback`

Master callback function. Note that this function will be called from the interrupt service routine at the end of a transfer, so its execution time should be as small as possible. It can be NULL if you want to check manually the status of the transfer.

Definition at line 308 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.14 CallbackParam `uint8 CallbackParam`

Parameter for the master callback function

Definition at line 312 of file Lpi2c_Ip_Types.h.

6.1.2.3.1.15 MasterStateIdx `uint8 MasterStateIdx`

Master State index

Definition at line 313 of file Lpi2c_Ip_Types.h.

6.1.2.4 struct Lpi2c_Ip_SlaveStateType

Slave internal context structure.

This structure is used by the slave-mode driver for its internal logic. It must be provided by the application through the LPI2C_DRV_SlaveInit() function, then it cannot be freed until the driver is de-initialized using LPI2C_DR↩V_SlaveDeinit(). The application should make no assumptions about the content of this structure.

Definition at line 324 of file Lpi2c_Ip_Types.h.

6.1.2.5 struct Lpi2c_Ip_SlaveConfigType

Slave configuration structure.

This structure is used to provide configuration parameters for the LPI2C slave at initialization time.

Definition at line 352 of file Lpi2c_Ip_Types.h.

Data Fields

- uint16 SlaveAddress
- boolean Is10bitAddr
- boolean SlaveListening
- Lpi2c_Ip_ModeType OperatingMode
- Lpi2c_Ip_AsyncTransferType TransferType
- uint32 GlitchFilterSDA
- uint32 GlitchFilterSCL
- uint32 DmaTxChannel
- uint32 DmaRxChannel
- Lpi2c_Ip_SlaveCallbackType SlaveCallback
- uint8 CallbackParam
- uint8 SlaveStateIdx

6.1.2.5.1 Field Documentation

6.1.2.5.1.1 SlaveAddress `uint16 SlaveAddress`

Slave address, 7-bit or 10-bit

Definition at line 354 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.2 Is10bitAddr `boolean Is10bitAddr`

Selects 7-bit or 10-bit slave address

Definition at line 355 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.3 SlaveListening `boolean SlaveListening`

Specifies if slave is in listening mode

Definition at line 356 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.4 OperatingMode `Lpi2c_Ip_ModeType OperatingMode`

I2C Operating mode

Definition at line 357 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.5 TransferType Lpi2c_Ip_AsyncTransferType TransferType

Type of LPI2C transfer

Definition at line 358 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.6 GlitchFilterSDA uint32 GlitchFilterSDA

SDA glitch filter

Definition at line 359 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.7 GlitchFilterSCL uint32 GlitchFilterSCL

SCL glitch filter

Definition at line 360 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.8 DmaTxChannel uint32 DmaTxChannel

Channel number for DMA tx channel. If DMA mode isn't used this field will be ignored.

Definition at line 361 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.9 DmaRxChannel uint32 DmaRxChannel

Channel number for DMA rx channel. If DMA mode isn't used this field will be ignored.

Definition at line 362 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.10 SlaveCallback Lpi2c_Ip_SlaveCallbackType SlaveCallback

Slave callback function. Note that this function will be called from the interrupt service routine, so its execution time should be as small as possible. It can be NULL if the slave is not in listening mode (slaveListening = false)

Definition at line 363 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.11 CallbackParam uint8 CallbackParam

Parameter for the slave callback function

Definition at line 368 of file Lpi2c_Ip_Types.h.

6.1.2.5.1.12 SlaveStateIdx uint8 SlaveStateIdx

Slave State index

Definition at line 369 of file Lpi2c_Ip_Types.h.

6.1.3 Macro Definition Documentation

6.1.3.1 I2C_START_SEC_CODE

```
#define I2C_START_SEC_CODE
```

Note

put all I2C code into defined section

Definition at line 146 of file Lpi2c_Ip.h.

6.1.3.2 LPI2C_IP_MASTER_DATA_MATCH_INT

```
#define LPI2C_IP_MASTER_DATA_MATCH_INT
```

LPI2C master interrupts Data Match Interrupt

Definition at line 84 of file Lpi2c_Ip_HwAccess.h.

6.1.3.3 LPI2C_IP_MASTER_PIN_LOW_TIMEOUT_INT

```
#define LPI2C_IP_MASTER_PIN_LOW_TIMEOUT_INT
```

Pin Low Timeout Interrupt

Definition at line 85 of file Lpi2c_Ip_HwAccess.h.

6.1.3.4 LPI2C_IP_MASTER_FIFO_ERROR_INT

```
#define LPI2C_IP_MASTER_FIFO_ERROR_INT
```

FIFO Error Interrupt

Definition at line 86 of file Lpi2c_Ip_HwAccess.h.

6.1.3.5 LPI2C_IP_MASTER_ARBITRATION_LOST_INT

```
#define LPI2C_IP_MASTER_ARBITRATION_LOST_INT
```

Arbitration Lost Interrupt

Definition at line 87 of file Lpi2c_Ip_HwAccess.h.

6.1.3.6 LPI2C_IP_MASTER_NACK_DETECT_INT

```
#define LPI2C_IP_MASTER_NACK_DETECT_INT
```

NACK Detect Interrupt

Definition at line 88 of file Lpi2c_Ip_HwAccess.h.

6.1.3.7 LPI2C_IP_MASTER_STOP_DETECT_INT

```
#define LPI2C_IP_MASTER_STOP_DETECT_INT
```

STOP Detect Interrupt

Definition at line 89 of file Lpi2c_Ip_HwAccess.h.

6.1.3.8 LPI2C_IP_MASTER_END_PACKET_INT

```
#define LPI2C_IP_MASTER_END_PACKET_INT
```

End Packet Interrupt

Definition at line 90 of file Lpi2c_Ip_HwAccess.h.

6.1.3.9 LPI2C_IP_MASTER_RECEIVE_DATA_INT

```
#define LPI2C_IP_MASTER_RECEIVE_DATA_INT
```

Receive Data Interrupt

Definition at line 91 of file Lpi2c_Ip_HwAccess.h.

6.1.3.10 LPI2C_IP_MASTER_TRANSMIT_DATA_INT

```
#define LPI2C_IP_MASTER_TRANSMIT_DATA_INT
```

Transmit Data Interrupt

Definition at line 92 of file Lpi2c_Ip_HwAccess.h.

6.1.3.11 LPI2C_IP_SLAVE_SMBUS_ALERT_RESPONSE_INT

```
#define LPI2C_IP_SLAVE_SMBUS_ALERT_RESPONSE_INT
```

LPI2C slave interrupts SMBus Alert Response Interrupt

Definition at line 97 of file Lpi2c_Ip_HwAccess.h.

6.1.3.12 LPI2C_IP_SLAVE_GENERAL_CALL_INT

```
#define LPI2C_IP_SLAVE_GENERAL_CALL_INT
```

General Call Interrupt

Definition at line 98 of file Lpi2c_Ip_HwAccess.h.

6.1.3.13 LPI2C_IP_SLAVE_ADDRESS_MATCH_1_INT

```
#define LPI2C_IP_SLAVE_ADDRESS_MATCH_1_INT
```

Address Match 1 Interrupt

Definition at line 99 of file Lpi2c_Ip_HwAccess.h.

6.1.3.14 LPI2C_IP_SLAVE_ADDRESS_MATCH_0_INT

```
#define LPI2C_IP_SLAVE_ADDRESS_MATCH_0_INT
```

Address Match 0 Interrupt

Definition at line 100 of file Lpi2c_Ip_HwAccess.h.

6.1.3.15 LPI2C_IP_SLAVE_FIFO_ERROR_INT

```
#define LPI2C_IP_SLAVE_FIFO_ERROR_INT
```

FIFO Error Interrupt

Definition at line 101 of file Lpi2c_Ip_HwAccess.h.

6.1.3.16 LPI2C_IP_SLAVE_BIT_ERROR_INT

```
#define LPI2C_IP_SLAVE_BIT_ERROR_INT
```

Bit Error Interrupt

Definition at line 102 of file Lpi2c_Ip_HwAccess.h.

6.1.3.17 LPI2C_IP_SLAVE_STOP_DETECT_INT

```
#define LPI2C_IP_SLAVE_STOP_DETECT_INT
```

STOP Detect Interrupt

Definition at line 103 of file Lpi2c_Ip_HwAccess.h.

6.1.3.18 LPI2C_IP_SLAVE_REPEATED_START_INT

```
#define LPI2C_IP_SLAVE_REPEATED_START_INT
```

Repeated Start Interrupt

Definition at line 104 of file Lpi2c_Ip_HwAccess.h.

6.1.3.19 LPI2C_IP_SLAVE_TRANSMIT_ACK_INT

```
#define LPI2C_IP_SLAVE_TRANSMIT_ACK_INT
```

Transmit ACK Interrupt

Definition at line 105 of file Lpi2c_Ip_HwAccess.h.

6.1.3.20 LPI2C_IP_SLAVE_ADDRESS_VALID_INT

```
#define LPI2C_IP_SLAVE_ADDRESS_VALID_INT
```

Address Valid Interrupt

Definition at line 106 of file Lpi2c_Ip_HwAccess.h.

6.1.3.21 LPI2C_IP_SLAVE_RECEIVE_DATA_INT

```
#define LPI2C_IP_SLAVE_RECEIVE_DATA_INT
```

Receive Data Interrupt

Definition at line 107 of file Lpi2c_Ip_HwAccess.h.

6.1.3.22 LPI2C_IP_SLAVE_TRANSMIT_DATA_INT

```
#define LPI2C_IP_SLAVE_TRANSMIT_DATA_INT
```

Transmit Data Interrupt

Definition at line 108 of file Lpi2c_Ip_HwAccess.h.

6.1.4 Enum Reference

6.1.4.1 Lpi2c_Ip_SlaveEventType

```
enum Lpi2c_Ip_SlaveEventType
```

Define the enum of the events which can trigger I2C slave callback.
This enum should include the events for all platforms implements Lpi2c_Ip_SlaveEventType_enum
Definition at line 79 of file Lpi2c_Ip_Callbacks.h.

6.1.4.2 Lpi2c_Ip_MasterEventType

```
enum Lpi2c_Ip_MasterEventType
```

Define the enum of the events which can trigger I2C master callback.
This enum should include the events for all platforms implements Lpi2c_Ip_MasterEventType_enum
Definition at line 98 of file Lpi2c_Ip_Callbacks.h.

6.1.4.3 Lpi2c_Ip_PinConfigType

```
enum Lpi2c_Ip_PinConfigType
```

Pin Configuration selection.

Enumerator

LPI2C_CFG_2PIN_OPEN_DRAIN	2-pin open drain mode
LPI2C_CFG_2PIN_OUTPUT_ONLY	2-pin output only mode (ultra-fast mode)
LPI2C_CFG_2PIN_PUSH_PULL	2-pin push-pull mode
LPI2C_CFG_4PIN_PUSH_PULL	4-pin push-pull mode
LPI2C_CFG_2PIN_OPEN_DRAIN_SLAVE	2-pin open drain mode with separate LPI2C slave
LPI2C_CFG_2PIN_OUTPUT_ONLY_SLAVE	2-pin output only mode (ultra-fast mode) with separate LPI2C slave
LPI2C_CFG_2PIN_PUSH_PULL_SLAVE	2-pin push-pull mode with separate LPI2C slave
LPI2C_CFG_4PIN_PUSH_PULL_INVERTED	4-pin push-pull mode (inverted outputs)

Definition at line 114 of file Lpi2c_Ip_HwAccess.h.

6.1.4.4 Lpi2c_Ip_NackConfigType

enum Lpi2c_Ip_NackConfigType

Master NACK reaction configuration.

Enumerator

LPI2C_NACK_RECEIVE	Receive ACK and NACK normally
LPI2C_NACK_IGNORE	Treat a received NACK as if it was an ACK

Definition at line 128 of file Lpi2c_Ip_HwAccess.h.

6.1.4.5 Lpi2c_Ip_SlaveAddressConfigType

enum Lpi2c_Ip_SlaveAddressConfigType

Slave address configuration.

Enumerator

LPI2C_SLAVE_ADDR_MATCH_0_7BIT	Address match 0 (7-bit)
LPI2C_SLAVE_ADDR_MATCH_0_10BIT	Address match 0 (10-bit)
LPI2C_SLAVE_ADDR_MATCH_0_7BIT_OR_↩ 1_7BIT	Address match 0 (7-bit) or Address match 1 (7-bit)
LPI2C_SLAVE_ADDR_MATCH_0_10BIT_OR↩ _1_10BIT	Address match 0 (10-bit) or Address match 1 (10-bit)

Enumerator

LPI2C_SLAVE_ADDR_MATCH_0_7BIT_OR_1_10BIT	Address match 0 (7-bit) or Address match 1 (10-bit)
LPI2C_SLAVE_ADDR_MATCH_0_10BIT_OR_1_7BIT	Address match 0 (10-bit) or Address match 1 (7-bit)
LPI2C_SLAVE_ADDR_MATCH_RANGE_7BIT	From Address match 0 (7-bit) to Address match 1 (7-bit)
LPI2C_SLAVE_ADDR_MATCH_RANGE_10BIT	From Address match 0 (10-bit) to Address match 1 (10-bit)

Definition at line 136 of file Lpi2c_Ip_HwAccess.h.

6.1.4.6 Lpi2c_Ip_SlaveNackConfigType

enum Lpi2c_Ip_SlaveNackConfigType

Slave NACK reaction configuration.

Enumerator

LPI2C_SLAVE_NACK_END_TRANSFER	Slave will end transfer when NACK detected
LPI2C_SLAVE_NACK_CONTINUE_TRANSFER	Slave will not end transfer when NACK detected

Definition at line 150 of file Lpi2c_Ip_HwAccess.h.

6.1.4.7 Lpi2c_Ip_SlaveNackTransmitType

enum Lpi2c_Ip_SlaveNackTransmitType

Slave ACK transmission options.

Enumerator

LPI2C_SLAVE_TRANSMIT_ACK	Transmit ACK for received word
LPI2C_SLAVE_TRANSMIT_NACK	Transmit NACK for received word

Definition at line 158 of file Lpi2c_Ip_HwAccess.h.

6.1.4.8 Lpi2c_Ip_ModeType

enum Lpi2c_Ip_ModeType

I2C operating modes.

Enumerator

LPI2C_STANDARD_MODE	Standard-mode (Sm), bidirectional data transfers up to 100 kbit/s
LPI2C_FAST_MODE	Fast-mode (Fm), bidirectional data transfers up to 400 kbit/s
LPI2C_FASTPLUS_MODE	Fast-mode Plus (Fm+), bidirectional data transfers up to 1 Mbit/s
LPI2C_HIGHSPEED_MODE	High-speed Mode (Hs-mode), bidirectional data transfers up to 3.4 Mbit/s

Definition at line 138 of file Lpi2c_Ip_Types.h.

6.1.4.9 Lpi2c_Ip_AsyncTransferType

enum Lpi2c_Ip_AsyncTransferType

Type of LPI2C transfer (based on interrupts or DMA).

Enumerator

LPI2C_USING_DMA	The driver will use DMA to perform I2C transfer
LPI2C_USING_INTERRUPTS	The driver will use interrupts to perform I2C transfer

Definition at line 152 of file Lpi2c_Ip_Types.h.

6.1.4.10 Lpi2c_Ip_StatusType

enum Lpi2c_Ip_StatusType

Type of LPI2C transfer (based on interrupts or DMA).

Enumerator

LPI2C_IP_SUCCESS_STATUS	I2C specific error codes
LPI2C_IP_RECEIVED_NACK_STATUS	NACK signal received
LPI2C_IP_TX_UNDERRUN_STATUS	TX underrun error
LPI2C_IP_RX_OVERRUN_STATUS	RX overrun error
LPI2C_IP_ARBITRATION_LOST_STATUS	Arbitration lost
LPI2C_IP_ABORTED_STATUS	A transfer was aborted
LPI2C_IP_BUS_BUSY_STATUS	I2C bus is busy, cannot start transfer

Definition at line 160 of file Lpi2c_Ip_Types.h.

6.1.4.11 Lpi2c_Ip_MasterPrescalerType

enum Lpi2c_Ip_MasterPrescalerType

Defines the example structure.

This structure is used as an example.

LPI2C master prescaler options

Enumerator

LPI2C_MASTER_PRESC_DIV_1	Divide by 1
LPI2C_MASTER_PRESC_DIV_2	Divide by 2
LPI2C_MASTER_PRESC_DIV_4	Divide by 4
LPI2C_MASTER_PRESC_DIV_8	Divide by 8
LPI2C_MASTER_PRESC_DIV_16	Divide by 16
LPI2C_MASTER_PRESC_DIV_32	Divide by 32
LPI2C_MASTER_PRESC_DIV_64	Divide by 64
LPI2C_MASTER_PRESC_DIV_128	Divide by 128

Definition at line 189 of file Lpi2c_Ip_Types.h.

6.1.4.12 Lpi2c_Ip_DirectionType

enum Lpi2c_Ip_DirectionType

Enumerator

LPI2C_IP_SEND	Send operation
LPI2C_IP_RECEIVE	Receive operation

Definition at line 247 of file Lpi2c_Ip_Types.h.

6.1.5 Function Reference

6.1.5.1 Lpi2c_Ip_MasterInit()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterInit (
            uint8 Instance,
            const Lpi2c_Ip_MasterConfigType * ConfigPtr )
```

Initialize the LPI2C master mode driver.

This function initializes the LPI2C driver in master mode.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*ConfigPtr*	Pointer to the LPI2C master user configuration structure. The function reads configuration data from this structure and initializes the driver accordingly. The application may free this structure after the function returns.

Returns

Error or success status returned by API

6.1.5.2 Lpi2c_Ip_MasterDeinit()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterDeinit (
            uint8 Instance )
```

De-initialize the LPI2C master mode driver.

This function de-initializes the LPI2C driver in master mode. The driver can't be used again until reinitialized. The context structure is no longer needed by the driver and can be freed after calling this function.

Parameters

in	*Instance*	LPI2C peripheral instance number

Returns

Error or success status returned by API

6.1.5.3 Lpi2c_Ip_MasterGetBaudRate()

```
void Lpi2c_Ip_MasterGetBaudRate (
            uint8 Instance,
            uint32 InputClock,
            uint32 * BaudRate )
```

Get the currently configured baud rate.

This function returns the currently configured baud rate.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*InputClock*	input clock in Hz
out	*BaudRate*	structure that contains the current baud rate in hertz and the baud rate in hertz for High-speed mode (unused in other modes, can be NULL)

6.1.5.4 Lpi2c_Ip_MasterSetBaudRate()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterSetBaudRate (
            uint8 Instance,
            Lpi2c_Ip_ModeType OperatingMode,
            uint32 Baudrate,
            uint32 InputClock )
```

Set the baud rate for any subsequent I2C communication.

This function sets the baud rate (SCL frequency) for the I2C master. It can also change the operating mode. Note that due to module limitation not any baud rate can be achieved. The driver will set a baud rate as close as possible to the requested baud rate, but there may still be substantial differences, for example if requesting a high baud rate while using a low-frequency protocol clock for the LPI2C module. The application should call Lpi2c_Ip_MasterGetBaudRate() after Lpi2c_Ip_MasterSetBaudRate() to check what baud rate was actually set.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*OperatingMode*	I2C operating mode
in	*BaudRate*	structure that contains the baud rate in hertz to use by current slave device and also the baud rate in hertz for High-speed mode (unused in other modes)
in	*InputClock*	input clock in Hz

Returns

Error or success status returned by API

6.1.5.5 Lpi2c_Ip_MasterSetSlaveAddr()

```
void Lpi2c_Ip_MasterSetSlaveAddr (
             uint8 Instance,
             const uint16 Address,
             const boolean Is10bitAddr )
```

Set the slave address for any subsequent I2C communication.

This function sets the slave address which will be used for any future transfer initiated by the LPI2C master.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*Address*	slave address, 7-bit or 10-bit
in	*Is10bitAddr*	specifies if provided address is 10-bit

6.1.5.6 Lpi2c_Ip_MasterSendData()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterSendData (
             uint8 Instance,
             uint8 * TxBuff,
             uint32 TxSize,
             boolean SendStop )
```

Perform a non-blocking send transaction on the I2C bus.

This function starts the transmission of a block of data to the currently configured slave address and returns immediately. The rest of the transmission is handled by the interrupt service routine. Use Lpi2c_Ip_MasterGetSendStatus() to check the progress of the transmission.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*TxBuff*	pointer to the data to be transferred
in	*TxSize*	length in bytes of the data to be transferred
in	*SendStop*	specifies whether or not to generate stop condition after the transmission

Returns

Error or success status returned by API

6.1.5.7 Lpi2c_Ip_MasterSendDataBlocking()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterSendDataBlocking (
            uint8 Instance,
            uint8 * TxBuff,
            uint32 TxSize,
            boolean SendStop,
            uint32 Timeout )
```

Perform a blocking send transaction on the I2C bus.

This function sends a block of data to the currently configured slave Address, and only returns when the transmission is complete.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*TxBuff*	pointer to the data to be transferred
in	*TxSize*	length in bytes of the data to be transferred
in	*SendStop*	specifies whether or not to generate stop condition after the transmission
in	*Timeout*	Timeout for the transfer in milliseconds

Returns

Error or success status returned by API

6.1.5.8 Lpi2c_Ip_MasterReceiveData()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterReceiveData (
            uint8 Instance,
            uint8 * RxBuff,
            uint32 RxSize,
            boolean SendStop )
```

Perform a non-blocking receive transaction on the I2C bus.

This function starts the reception of a block of data from the currently configured slave address and returns immediately. The rest of the reception is handled by the interrupt service routine. Use Lpi2c_Ip_MasterGetReceiveStatus() to check the progress of the reception.

Parameters

in	*Instance*	LPI2C peripheral instance number
out	*RxBuff*	pointer to the buffer where to store received data
in	*RxSize*	length in bytes of the data to be transferred
in	*SendStop*	specifies whether or not to generate stop condition after the reception

Returns

Error or success status returned by API

6.1.5.9 Lpi2c_Ip_MasterReceiveDataBlocking()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterReceiveDataBlocking (
            uint8 Instance,
            uint8 * RxBuff,
            uint32 RxSize,
            boolean SendStop,
            uint32 Timeout )
```

Perform a blocking receive transaction on the I2C bus.

This function receives a block of data from the currently configured slave Address, and only returns when the transmission is complete.

Parameters

in	*Instance*	LPI2C peripheral instance number
out	*RxBuff*	pointer to the buffer where to store received data
in	*RxSize*	length in bytes of the data to be transferred
in	*SendStop*	specifies whether or not to generate stop condition after the reception
in	*Timeout*	Timeout for the transfer in milliseconds

Returns

Error or success status returned by API

6.1.5.10 Lpi2c_Ip_MasterGetTransferStatus()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_MasterGetTransferStatus (
            uint8 Instance,
            uint32 * BytesRemaining )
```

Return the current status of the I2C master transfer.

This function can be called during a non-blocking transmission to check the status of the transfer.

Parameters

in	*Instance*	LPI2C peripheral instance number
out	*BytesRemaining*	the number of remaining bytes in the active I2C transfer

Returns

Error or success status returned by API

6.1.5.11 Lpi2c_Ip_MasterIRQHandler()

```
void Lpi2c_Ip_MasterIRQHandler (
            uint8 Instance )
```

Handle master operation when I2C interrupt occurs.

This is the interrupt service routine for the LPI2C master mode driver. It handles the rest of the transfer started by one of the send/receive functions.

Parameters

in	*Instance*	LPI2C peripheral instance number

6.1.5.12 Lpi2c_Ip_SlaveInit()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveInit (
            uint8 Instance,
            const Lpi2c_Ip_SlaveConfigType * ConfigPtr )
```

Initialize the I2C slave mode driver.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*ConfigPtr*	Pointer to the LPI2C slave user configuration structure. The function reads configuration data from this structure and initializes the driver accordingly. The application may free this structure after the function returns.

Returns

Error or success status returned by API

6.1.5.13 Lpi2c_Ip_SlaveDeinit()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveDeinit (
            uint8 Instance )
```

De-initialize the I2C slave mode driver.

This function de-initializes the LPI2C driver in slave mode. The driver can't be used again until reinitialized. The context structure is no longer needed by the driver and can be freed after calling this function.

Parameters

in	*Instance*	LPI2C peripheral instance number

Returns

Error or success status returned by API

6.1.5.14 Lpi2c_Ip_SlaveSetBuffer()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveSetBuffer (
            uint8 Instance,
            uint8 * DataBuff,
            uint32 DataSize )
```

Provide a buffer for transmitting data.

This function provides a buffer from which the LPI2C slave-mode driver can transmit data. It can be called for example from the user callback provided at initialization time, when the driver reports events LPI2C_SLAVE_E↩ VENT_TX_REQ or LPI2C_SLAVE_EVENT_TX_EMPTY.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*TxBuff*	pointer to the data to be transferred
in	*TxSize*	length in bytes of the data to be transferred

Returns

Error or success status returned by API

6.1.5.15 Lpi2c_Ip_SlaveGetTransferStatus()

```
Lpi2c_Ip_StatusType Lpi2c_Ip_SlaveGetTransferStatus (
            uint8 Instance,
            uint32 * BytesRemaining )
```

Return the current status of the I2C slave transfer.

This function can be called during a non-blocking transmission to check the status of the transfer.

Parameters

in	*Instance*	LPI2C peripheral instance number
in	*bytesRemaining[out]*	the number of remaining bytes in the active I2C transfer

Returns

Error or success status returned by API

6.1.5.16 Lpi2c_Ip_SlaveIRQHandler()

```
void Lpi2c_Ip_SlaveIRQHandler (
             uint8 Instance )
```

Handle slave operation when I2C interrupt occurs.

This is the interrupt service routine for the LPI2C slave mode driver. It handles any transfer initiated by an I2C master and notifies the application via the provided callback when relevant events occur.

Parameters

in	*Instance*	LPI2C peripheral instance number

Returns

void

6.1.5.17 Lpi2c_Ip_ModuleIRQHandler()

```
void Lpi2c_Ip_ModuleIRQHandler (
             uint8 Instance )
```

Handler for both slave and master operation when I2C interrupt occurs.

This is the interrupt service routine for the LPI2C slave and master mode driver. It handles any transfer initiated by an I2C master and notifies the application via the provided callback when relevant events occur.

Parameters

in	*Instance*	LPI2C peripheral instance number

Returns

void

6.1.5.18 Lpi2c_Ip_SetMasterCallback()

```
void Lpi2c_Ip_SetMasterCallback (
            uint8 Instance,
            Lpi2c_Ip_MasterCallbackType MasterCallback )
```

Sets the master callback.

This functions sets the master callback

Parameters

in	*u32Instance*	LPI2C peripheral instance number
in	*masterCallback*	master callback to be set

Returns

void

6.1.5.19 Lpi2c_Ip_SetSlaveCallback()

```
void Lpi2c_Ip_SetSlaveCallback (
            uint8 Instance,
            Lpi2c_Ip_SlaveCallbackType SlaveCallback )
```

Sets the slave callback.

This functions sets the slave callback

Parameters

in	*u32Instance*	LPI2C peripheral instance number
in	*slaveCallback*	slave callback to be set

Returns

void

6.1.5.20 Lpi2c_Ip_StartListening()

```
void Lpi2c_Ip_StartListening (
            uint8 Instance )
```

Start listening.

This is used to enable slave events

Parameters

in	*u32Instance*	LPI2C peripheral instance number

Returns

void

6.1.5.21 Lpi2c_Ip_SetMasterHighSpeedMode()

```
void Lpi2c_Ip_SetMasterHighSpeedMode (
            uint8 Instance,
            boolean HighSpeedEnabled )
```

Set high speed mode for master.

This function enables high speed mode for master

Parameters

in	*u32Instance*	LPI2C peripheral instance number
in	*bHighSpeedEnabled*	enables/disables master high speed mode

Returns

void

6.1.5.22 Lpi2c_Ip_Init()

```
void Lpi2c_Ip_Init (
            LPI2C_Type * BaseAddr )
```

Initializes the LPI2C module to a known state.

This function initializes all the registers of the LPI2C module to their reset value.

Parameters

in	*BaseAddr*	base address of the LPI2C module

6.2 Flexio_I2c Driver

6.2.1 Detailed Description

6.2.1.1 General information

The I2C module provides a simple and efficient method of data exchange between a chip and other devices, such as microcontrollers, EEPROM, real-time clock devices, analog-to-digital converters and LCDs.

The advantages of the I2C bus is that it minimizes interconnections between devices, allows the connection of additional devices to the bus, includes collision detection and arbitration that prevent data corruption and it doesn't require an external address decoder.

6.2.1.2 Features

- Interrupt based
- Master operation
- Provides blocking and non-blocking transmit and receive functions
- 7-bit
- Configurable baud rate

6.2.1.2.1 Master Mode Master Mode provides functions for transmitting or receiving data to/from any I2C slave. Slave address and baud rate are provided at initialization time through the master configuration structure, but they can be changed at runtime by using Flexio_I2c_Ip_MasterSetBaudRate() or Flexio_I2c_Ip_MasterSet↩SlaveAddr().

To send or receive data to/from the currently configured slave address, use functions Flexio_I2c_Ip_MasterSend↩Data() or Flexio_I2c_Ip_MasterReceiveData() (or their blocking counterparts). Parameter `sendStop` can be used to chain multiple transfers with repeated START condition between them, for example when sending a command and then immediately receiving a response. The application should ensure that any send or receive transfer with `sendStop` set to `false` is followed by another transfer, otherwise the Flexio I2c master will hold the SCL line low indefinitely and block the I2C bus. The last transfer from a chain should always have `sendStop` set to `true`.

Blocking operations will return only when the transfer is completed, either successfully or with error. Non-blocking operations will initiate the transfer and return STATUS_SUCCESS, but the module is still busy with the transfer and another transfer can't be initiated until the current transfer is complete. The application can check the status of the current transfer by calling Flexio_I2c_Ip_MasterGetStatus(). If the transfer is completed, the functions will return either STATUS_SUCCESS or an error code, depending on the outcome of the last transfer.

Data Structures

- struct Flexio_I2c_Ip_MasterStateType
  Master internal context structure. *More...*
- struct Flexio_I2c_Ip_MasterConfigType
  Master configuration structure. *More...*

Macros

- #define FEATURE_FLEXIO_MAX_CHANNEL_COUNT
- #define FLEXIO_I2C_IP_INSTANCE_COUNT

Enum Reference

- enum Flexio_I2c_Ip_MasterEventType

  Define the enum of the events which can trigger I2C master callback.
- enum Flexio_I2c_Ip_StatusType

  Type of FLEXIO I2C transfer status.
- enum Flexio_I2c_Ip_AsyncTransferType

  Type of FLEXIO I2C transfer (based on interrupts or DMA).
- enum Flexio_I2c_Ip_Timer_Decrement

  Driver type: timer decrement.
- enum Flexio_Ip_DriverType

  Driver type: interrupts/polling/DMA.

6.2.2 Data Structure Documentation

6.2.2.1 struct Flexio_I2c_Ip_MasterStateType

Master internal context structure.

This structure is used by the driver for its internal logic. It must be provided by the application through the FLEXIO_I2C_DRV_MasterInit() function, then it cannot be freed until the driver is de-initialized using FLEXIO_I2C_DRV_MasterDeinit(). The application should make no assumptions about the content of this structure.

Definition at line 161 of file Flexio_I2c_Ip_Types.h.

6.2.2.2 struct Flexio_I2c_Ip_MasterConfigType

Master configuration structure.

This structure is used to provide configuration parameters for the flexio_i2c master at initialization time.

Definition at line 208 of file Flexio_I2c_Ip_Types.h.

Data Fields

- uint16 SlaveAddress
- Flexio_I2c_Ip_AsyncTransferType I2cAsyncMethod
- uint32 BaudRate
- uint8 SdaPin
- uint8 SclPin
- Flexio_I2c_Ip_MasterCallbackType Callback
- uint8 CallbackParam
- uint8 ResourceIndex
- uint32 DmaRxChannel
- uint32 DmaTxChannel
- uint8 MasterStateIdx

6.2.2.2.1 Field Documentation

6.2.2.2.1.1 SlaveAddress `uint16 SlaveAddress`

Slave address, 7-bit

Definition at line 210 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.2 I2cAsyncMethod `Flexio_I2c_Ip_AsyncTransferType I2cAsyncMethod`

Driver type: interrupts/polling/DMA

Definition at line 211 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.3 BaudRate `uint32 BaudRate`

Baud rate in hertz

Definition at line 212 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.4 SdaPin `uint8 SdaPin`

Flexio pin to use as I2C SDA pin

Definition at line 215 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.5 SclPin uint8 SclPin

Flexio pin to use as I2C SCL pin

Definition at line 216 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.6 Callback Flexio_I2c_Ip_MasterCallbackType Callback

User callback function. Note that this function will be called from the interrupt service routine, so its execution time should be as small as possible. It can be NULL if it is not needed

Definition at line 217 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.7 CallbackParam uint8 CallbackParam

Parameter for the callback function

Definition at line 221 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.8 ResourceIndex uint8 ResourceIndex

Index of first used internal resource instance (shifter and timer)

Definition at line 223 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.9 DmaRxChannel uint32 DmaRxChannel

Rx DMA channel number. Only used in DMA mode

Definition at line 229 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.10 DmaTxChannel uint32 DmaTxChannel

Tx DMA channel number. Only used in DMA mode

Definition at line 230 of file Flexio_I2c_Ip_Types.h.

6.2.2.2.1.11 MasterStateIdx uint8 MasterStateIdx

Master state index

Definition at line 231 of file Flexio_I2c_Ip_Types.h.

6.2.3 Macro Definition Documentation

6.2.3.1 FEATURE_FLEXIO_MAX_CHANNEL_COUNT

```
#define FEATURE_FLEXIO_MAX_CHANNEL_COUNT
```

Number of Flexio i2c logical channel.

Definition at line 55 of file Flexio_I2c_Ip_Features.h.

6.2.3.2 FLEXIO_I2C_IP_INSTANCE_COUNT

```
#define FLEXIO_I2C_IP_INSTANCE_COUNT
```

Number of instances of the FLEXIO module.

Definition at line 58 of file Flexio_I2c_Ip_Features.h.

6.2.4 Enum Reference

6.2.4.1 Flexio_I2c_Ip_MasterEventType

```
enum Flexio_I2c_Ip_MasterEventType
```

Define the enum of the events which can trigger I2C master callback.

This enum should include the events for all platforms

Definition at line 78 of file Flexio_I2c_Ip_Callbacks.h.

6.2.4.2 Flexio_I2c_Ip_StatusType

```
enum Flexio_I2c_Ip_StatusType
```

Type of FLEXIO I2C transfer status.

Enumerator

FLEXIO_I2C_IP_SUCCESS_STATUS	I2C specific error codes
FLEXIO_I2C_IP_RECEIVED_NACK_STATUS	NACK signal received
FLEXIO_I2C_IP_TX_UNDERRUN_STATUS	TX underrun error
FLEXIO_I2C_IP_RX_OVERRUN_STATUS	RX overrun error
FLEXIO_I2C_IP_ARBITRATION_LOST_STATUS	Arbitration lost
FLEXIO_I2C_IP_ABORTED_STATUS	A transfer was aborted
FLEXIO_I2C_IP_BUS_BUSY_STATUS	I2C bus is busy, cannot start transfer

Definition at line 90 of file Flexio_I2c_Ip_Types.h.

6.2.4.3 Flexio_I2c_Ip_AsyncTransferType

enum Flexio_I2c_Ip_AsyncTransferType

Type of FLEXIO I2C transfer (based on interrupts or DMA).

Enumerator

FLEXIO_I2C_USING_DMA	The driver will use DMA to perform flexio I2C transfer
FLEXIO_I2C_USING_INTERRUPTS	The driver will use interrupts to perform flexio I2C transfer

Definition at line 109 of file Flexio_I2c_Ip_Types.h.

6.2.4.4 Flexio_I2c_Ip_Timer_Decrement

enum Flexio_I2c_Ip_Timer_Decrement

Driver type: timer decrement.

Enumerator

FLEXIO_TMR_DECREMENT_ON_FLEXIO_C↩LK_DIV_1	Decrement counter on FlexIO clock, Shift clock equals Timer output
FLEXIO_TMR_DECREMENT_ON_FLEXIO_C↩LK_DIV_16	Decrement counter on FlexIO clock divided by 16, Shift clock equals Timer output
FLEXIO_TMR_DECREMENT_ON_FLEXIO_C↩LK_DIV_256	Decrement counter on FlexIO clock divided by 256, Shift clock equals Timer output.

Definition at line 118 of file Flexio_I2c_Ip_Types.h.

6.2.4.5 Flexio_Ip_DriverType

enum Flexio_Ip_DriverType

Driver type: interrupts/polling/DMA.

Enumerator

FLEXIO_I2C_IP_DRIVER_TYPE_INTERRUPTS	Driver uses interrupts for data transfers
FLEXIO_I2C_IP_DRIVER_TYPE_POLLING	Driver is based on polling
FLEXIO_I2C_IP_DRIVER_TYPE_DMA	Driver uses DMA for data transfers

Definition at line 127 of file Flexio_I2c_Ip_Types.h.

6.3 I2c Driver

6.3.1 Detailed Description

Data Structures

- struct Lpi2c_Ipw_HwChannelConfigType

  The structure contains the hardware configuration for lpi2c module. More...
- struct I2c_Ipw_HwChannelConfigType

  The structure contains the hardware channel configuration type. More...
- struct I2c_HwUnitConfigType

  Structure that contains I2C Hw configuration. More...
- struct I2c_ConfigType

  This type contains initialization data. More...
- struct I2c_RequestType

  Definition for Request Buffer. This is the structure which is passed to I2c_SyncTransmit or I2c_AsyncTransmit function. This holds the necessary information required for the communication of I2C Hw with the Slave device. More...

Macros

- #define I2C_STOP_SEC_CONFIG_DATA_UNSPECIFIED

  Export Post-Build configurations.
- #define I2C_START_SEC_VAR_INIT_8

Types Reference

- typedef uint8 I2c_HwUnitType

  This gives the numeric ID (hardware number) of an I2C hw Unit.
- typedef uint8 I2c_PartCoreType

  This gives the numeric ID (paritition number) of an I2C hw Unit.
- typedef uint16 I2c_AddressType

  Type Address Value of Device and its register value.
- typedef uint8 I2c_DataType

  Type Data to be sent or received.

Enum Reference

- enum I2c_ApiFunctionIdType

  API functions service IDs.
- enum I2c_StatusType

  Definition for different state and errors of Operation Status.
- enum I2c_AsynchronousMethodType

  Asynchronous method used.
- enum I2c_MasterSlaveModeType

  Definition of the master/slave mode of an I2C hw unit.

Function Reference

- void I2c_Init (const I2c_ConfigType *Config)

  Initializes the I2C module.
- void I2c_DeInit (void)

  DeInitializes the I2C module.
- Std_ReturnType I2c_SyncTransmit (uint8 Channel, const I2c_RequestType *RequestPtr)

  Sends or receives an I2C message blocking.
- Std_ReturnType I2c_AsyncTransmit (uint8 Channel, const I2c_RequestType *RequestPtr)

  Starts an asynchronous transmission on the I2C bus.
- I2c_StatusType I2c_GetStatus (uint8 Channel)

  Gets the status of an I2C channel.
- Std_ReturnType I2c_PrepareSlaveBuffer (uint8 Channel, uint8 NumberOfBytes, I2c_DataType *DataBuffer)

  Prepare the RX or TX buffer for a slave channel.
- Std_ReturnType I2c_StartListening (uint8 Channel)

  Makes a slave channel available for processing requests (addressings).
- void I2c_Ipw_InitChannel (const uint8 Channel, const I2c_HwUnitConfigType *ConfigPtr)

  Initialize a I2c channel.
- void I2c_Ipw_DeInitChannel (const uint8 Channel, const I2c_HwUnitConfigType *ConfigPtr)

  De initialize a I2c channel.
- Std_ReturnType I2c_Ipw_SyncTransmit (uint8 Channel, const I2c_RequestType *Request, const I2c_HwUnitConfigType *HwConfigType)

  Sends or receives an I2c message from the slave.
- Std_ReturnType I2c_Ipw_AsyncTransmit (uint8 Channel, const I2c_RequestType *Request, const I2c_HwUnitConfigType *HwConfigType)

  Starts sending or receiving an I2c message from the slave.
- void I2c_Ipw_PrepareSlaveBuffer (uint8 Channel, uint8 NumberOfBytes, I2c_DataType *DataBuffer)

  Prepare the RX or TX buffer for a slave channel.
- I2c_StatusType I2c_Ipw_GetStatus (const uint8 Channel, const I2c_HwUnitConfigType *HwConfigType)

  Gets the status of an I2c channel.

Variables

- sint8 I2c_as8ChannelHardwareMap [I2C_HW_MAX_MODULES]
- const I2c_DemConfigType * I2c_apDemCfg [I2C_MAX_CORE_ID]

6.3.2 Data Structure Documentation

6.3.2.1 struct Lpi2c_Ipw_HwChannelConfigType

The structure contains the hardware configuration for lpi2c module.

Definition at line 142 of file I2c_Ipw_Types.h.

6.3.2.2 struct I2c_Ipw_HwChannelConfigType

The structure contains the hardware channel configuration type.

Definition at line 152 of file I2c_Ipw_Types.h.

6.3.2.3 struct I2c_HwUnitConfigType

Structure that contains I2C Hw configuration.

It contains the information specific to one I2C Hw unit

Definition at line 205 of file I2c_Types.h.

Data Fields

- const I2c_HwUnitType I2c_HwUnit
  Numeric instance value of I2C Hw Unit.
- const I2c_PartCoreType I2c_PartitionId
  Master/slave mode configuration of the I2C Hw Unit.
- const I2c_MasterSlaveModeType MasterSlaveConfig
  Hardware channel type.
- const I2c_HwChannelType I2c_Ipw_ChannelType
  Structure containing the hardware specific configuration for the channel.

6.3.2.3.1 Field Documentation

6.3.2.3.1.1 I2c_HwUnit `const I2c_HwUnitType I2c_HwUnit`

Numeric instance value of I2C Hw Unit.

<

Numeric Partition Id

Definition at line 208 of file I2c_Types.h.

6.3.2.3.1.2 I2c_PartitionId `const I2c_PartCoreType I2c_PartitionId`

Master/slave mode configuration of the I2C Hw Unit.

Definition at line 211 of file I2c_Types.h.

6.3.2.3.1.3 MasterSlaveConfig `const I2c_MasterSlaveModeType MasterSlaveConfig`

Hardware channel type.

Definition at line 214 of file I2c_Types.h.

6.3.2.3.1.4 I2c_Ipw_ChannelType `const I2c_HwChannelType I2c_Ipw_ChannelType`

Structure containing the hardware specific configuration for the channel.

Definition at line 217 of file I2c_Types.h.

6.3.2.4 struct I2c_ConfigType

This type contains initialization data.

This contains initialization data for the I2C driver. It shall contain:

- The number of I2C modules to be configured
- Dem error reporting configuration
- I2C dependent properties for used HW units

Definition at line 232 of file I2c_Types.h.

Data Fields

- const I2c_PartCoreType I2c_CoreId
  Numeric Partition Id.
- const I2c_DemConfigType * I2c_DemConfig
  Pointer to I2c hardware unit configuration.

6.3.2.4.1 Field Documentation

6.3.2.4.1.1 I2c_CoreId `const I2c_PartCoreType I2c_CoreId`

Numeric Partition Id.

<

DEM error reporting configuration.

Definition at line 235 of file I2c_Types.h.

6.3.2.4.1.2 I2c_DemConfig `const I2c_DemConfigType* I2c_DemConfig`

Pointer to I2c hardware unit configuration.

Definition at line 239 of file I2c_Types.h.

6.3.2.5 struct I2c_RequestType

Definition for Request Buffer. This is the structure which is passed to I2c_SyncTransmit or I2c_AsyncTransmit function. This holds the necessary information required for the communication of I2C Hw with the Slave device.

Definition at line 253 of file I2c_Types.h.

Data Fields

- I2c_AddressType SlaveAddress

  Slave Device Address.
- boolean BitsSlaveAddressSize

  If this is true the data will be sent with high speed enabled (if hardware support exists).
- boolean HighSpeedMode

  When this is true, NACK will be ignored during the address cycle.
- boolean ExpectNack

  When this is true, a repeated start (Sr) will be issued on the bus instead of a STOP at the end of the transfer.
- boolean RepeatedStart

  Buffer Size : The number of bytes for reading or writing.
- uint16 BufferSize

  Direction of the data. Can be either Send or Receive.
- I2c_DataDirectionType DataDirection

  Buffer to Store or to transmit Serial data.

6.3.2.5.1 Field Documentation

6.3.2.5.1.1 SlaveAddress `I2c_AddressType SlaveAddress`

Slave Device Address.

<

This is true when the slave address is 10 bits, when false the address is on 7 bits.

Definition at line 256 of file I2c_Types.h.

6.3.2.5.1.2 BitsSlaveAddressSize `boolean BitsSlaveAddressSize`

If this is true the data will be sent with high speed enabled (if hardware support exists).

Definition at line 259 of file I2c_Types.h.

6.3.2.5.1.3 HighSpeedMode `boolean HighSpeedMode`

When this is true, NACK will be ignored during the address cycle.

Definition at line 262 of file I2c_Types.h.

6.3.2.5.1.4 ExpectNack `boolean ExpectNack`

When this is true, a repeated start (Sr) will be issued on the bus instead of a STOP at the end of the transfer.

Definition at line 265 of file I2c_Types.h.

6.3.2.5.1.5 RepeatedStart `boolean RepeatedStart`

Buffer Size : The number of bytes for reading or writing.

Definition at line 268 of file I2c_Types.h.

6.3.2.5.1.6 BufferSize `uint16 BufferSize`

Direction of the data. Can be either Send or Receive.

Definition at line 271 of file I2c_Types.h.

6.3.2.5.1.7 DataDirection `I2c_DataDirectionType DataDirection`

Buffer to Store or to transmit Serial data.

Definition at line 274 of file I2c_Types.h.

6.3.3 Macro Definition Documentation

6.3.3.1 I2C_STOP_SEC_CONFIG_DATA_UNSPECIFIED

```
#define I2C_STOP_SEC_CONFIG_DATA_UNSPECIFIED
```

Export Post-Build configurations.

I2c_h_REF_1 MISRA 2012 Required Rule 19.15, Repeated include file I2c_h_REF_3 MISRA 2012 Advisory Rule 19.1, only preprocessor statements and comments before '#include'

Definition at line 240 of file CDD_I2c.h.

6.3.3.2 I2C_START_SEC_VAR_INIT_8

```
#define I2C_START_SEC_VAR_INIT_8
```

I2c_h_REF_1 MISRA 2012 Required Rule 19.15, Repeated include file I2c_h_REF_3 MISRA 2012 Advisory Rule 19.1, only preprocessor statements and comments before '#include'

Definition at line 249 of file CDD_I2c.h.

6.3.4 Types Reference

6.3.4.1 I2c_HwUnitType

```
typedef uint8 I2c_HwUnitType
```

This gives the numeric ID (hardware number) of an I2C hw Unit.

Definition at line 175 of file I2c_Types.h.

6.3.4.2 I2c_PartCoreType

```
typedef uint8 I2c_PartCoreType
```

This gives the numeric ID (paritition number) of an I2C hw Unit.

Definition at line 181 of file I2c_Types.h.

6.3.4.3 I2c_AddressType

```
typedef uint16 I2c_AddressType
```

Type Address Value of Device and its register value.

Definition at line 189 of file I2c_Types.h.

6.3.4.4 I2c_DataType

```
typedef uint8 I2c_DataType
```

Type Data to be sent or received.

Definition at line 197 of file I2c_Types.h.

6.3.5 Enum Reference

6.3.5.1 I2c_ApiFunctionIdType

```
enum I2c_ApiFunctionIdType
```

API functions service IDs.

Service IDs of the I2C API.

Enumerator

I2C_INIT_ID	I2c_Init() ID.
I2C_DEINIT_ID	I2c_DeInit() ID.
I2C_SYNCTRANSMIT_ID	I2c_SyncTransmit() ID.
I2C_ASYNCTRANSMIT_ID	I2c_AsyncTransmit() ID.
I2C_GETSTATUS_ID	I2c_GetStatus() ID.
I2C_PREPARESLAVEBUFFER_ID	I2c_PrepareSlaveBuffer() ID.
I2C_STARTLISTENING_ID	I2c_StartListening() ID.
I2C_GETVERSIONINFO_ID	I2c_GetVersionInfo() ID.

Definition at line 113 of file I2c_Types.h.

6.3.5.2 I2c_StatusType

enum I2c_StatusType

Definition for different state and errors of Operation Status.

Enumerator

I2C_CH_IDLE	Status Indication I2C channel is idle.
I2C_CH_SEND	Status Indication send operation is ongoing.
I2C_CH_RECEIVE	Status Indication receiving operation is ongoing.
I2C_CH_FINISHED	Status Indication operation is finished.
I2C_CH_ERROR_PRESENT	Status Indication an error is present.

Definition at line 132 of file I2c_Types.h.

6.3.5.3 I2c_AsynchronousMethodType

enum I2c_AsynchronousMethodType

Asynchronous method used.

Enumerator

I2C_INTERRUPT_MODE	Asynchronous Mechanism using interrupts.
I2C_DMA_MODE	Asynchronous Mechanism using DMA.

Definition at line 147 of file I2c_Types.h.

6.3.5.4 I2c_MasterSlaveModeType

enum I2c_MasterSlaveModeType

Definition of the master/slave mode of an I2C hw unit.

Definition at line 156 of file I2c_Types.h.

6.3.6 Function Reference

6.3.6.1 I2c_Init()

```
void I2c_Init (
            const I2c_ConfigType * Config )
```

Initializes the I2C module.

This function performs software initialization of I2C driver:

- Maps logical channels to hardware channels
- Initializes all channels
- Sets driver state machine to I2C_INIT.

Parameters

in	*pConfig*	Pointer to I2C driver configuration set.

Returns

void

Note

Service ID: 0x00.

Synchronous, non re-entrant function.

6.3.6.2 I2c_DeInit()

```
void I2c_DeInit (
            void  )
```

DeInitializes the I2C module.

This function performs software de initialization of I2C modules to reset values. The service influences only the peripherals, which are allocated by static configuration and the runtime configuration set passed by the previous call of I2c_Init() The driver needs to be initialized before calling I2c_DeInit(). Otherwise, the function I2c_DeInit shall raise the development error I2C_E_UNINIT and leave the desired de initialization functionality without any action.

Parameters

in	*void*	

Returns

void

Note

Service ID: 0x01.

Synchronous, non re-entrant function.

6.3.6.3 I2c_SyncTransmit()

```
Std_ReturnType I2c_SyncTransmit (
            uint8 Channel,
            const I2c_RequestType * RequestPtr )
```

Sends or receives an I2C message blocking.

Sends the slave address and based on the direction of the message it sends or receives data by using a blocking mechanism.

Parameters

in	*u8Channel*	I2C channel to be addressed.
in	*pRequestPtr*	Pointer to data information to be used

Returns

Std_ReturnType.

Return values

E_NOT_OK	If the I2C Channel is not valid or I2C driver is not initialized or pRequestPtr is NULL or I2C Channel is in busy state.
E_OK	Otherwise.

Note

Service ID: 0x02.

Synchronous, non reentrant function.

6.3.6.4 I2c_AsyncTransmit()

```
Std_ReturnType I2c_AsyncTransmit (
            uint8 Channel,
            const I2c_RequestType * RequestPtr )
```

Starts an asynchronous transmission on the I2C bus.

Sends the slave address and enables the interrupts that will send or receive data depending on the direction of the message.

Parameters

in	*u8Channel*	I2C channel to be addressed.
in	*pRequestPtr*	Pointer to data information to be used

Returns

Std_ReturnType.

Return values

E_NOT_OK	If the I2C Channel is not valid or I2C driver is not initialized or pRequestPtr is NULL or I2C Channel is in busy state.
E_OK	Otherwise.

Note

Service ID: 0x03.

Synchronous, non reentrant function.

6.3.6.5 I2c_GetStatus()

```
I2c_StatusType I2c_GetStatus (
            uint8 Channel )
```

Gets the status of an I2C channel.

Gets the status of an I2C channel and checks for errors.

Parameters

in	*u8Channel*	I2C channel to be addressed.

Returns

I2C_StatusType.

Return values

I2C_CH_IDLE	If the I2C Channel is in default state.
I2C_CH_SEND	If the I2C Channel is busy sending data.
I2C_CH_RECEIVE	If the I2C Channel is busy receiving data.
I2C_CH_FINISHED	If the I2C Channel finished the last transmission (sending or receiving data) successfully with no errors.
I2C_CH_ERROR_PRESENT	If the I2C Channel encountered an error during the last transmission.

Note

Service ID: 0x04.

Synchronous, non re-entrant function.

6.3.6.6 I2c_PrepareSlaveBuffer()

```
Std_ReturnType I2c_PrepareSlaveBuffer (
            uint8 Channel,
            uint8 NumberOfBytes,
            I2c_DataType * DataBuffer )
```

Prepare the RX or TX buffer for a slave channel.

Prepares a RX or TX buffer that will be used to receive data or send data when requested by the master.

Parameters

in	*Channel*	I2C channel to be addressed.
in	*NumberOfBytes*	Maximum number of bytes to be sent or received.
in	*DataBuffer*	Pointer to data buffer

Returns

Std_ReturnType.

Return values

E_NOT_OK	If the I2C Channel is not valid or I2C driver is not initialized or DataBuffer is NULL or I2C Channel is a master channel.
E_OK	Otherwise.

Note

Service ID: 0x05.

Synchronous, non reentrant function.

6.3.6.7 I2c_StartListening()

```
Std_ReturnType I2c_StartListening (
            uint8 Channel )
```

Makes a slave channel available for processing requests (addressings).

When called, the slave channel becomes available for starting incoming or outgoing transfers.

Parameters

in	*u8Channel*	I2C channel to be addressed.

Returns

Std_ReturnType.

Return values

E_NOT_OK	If the I2C Channel is not valid or I2C driver is not initialized or I2C Channel is a master channel.
E_OK	Otherwise.

Note

Service ID: 0x06.

Synchronous, non reentrant function.

6.3.6.8 I2c_Ipw_InitChannel()

```
void I2c_Ipw_InitChannel (
            const uint8 Channel,
            const I2c_HwUnitConfigType * ConfigPtr )
```

Initialize a I2c channel.

This function initializes all hardware registers needed to start the I2c functionality on the selected channel.

Parameters

in	*Channel*	I2c channel to be initialized. ConfigPtr Configuration pointer containing hardware specific settings.

Returns

void.

6.3.6.9 I2c_Ipw_DeInitChannel()

```
void I2c_Ipw_DeInitChannel (
            const uint8 Channel,
            const I2c_HwUnitConfigType * ConfigPtr )
```

De initialize a I2c channel.

This function de initializes the hardware registers of an I2c channel

Parameters

in	*Channel*	I2c channel to be de initialized. eChannelType The type of the channel (LPI2C or FlexIO).

Returns

void.

6.3.6.10 I2c_Ipw_SyncTransmit()

```
Std_ReturnType I2c_Ipw_SyncTransmit (
            uint8 Channel,
            const I2c_RequestType * Request,
            const I2c_HwUnitConfigType * HwConfigType )
```

Sends or receives an I2c message from the slave.

Generate (repeated) START and send the address of the slave to initiate a transmission.

Parameters

in	*Channel*	I2c channel to be addressed.
in	*Request*	Pointer to the structure that contains the information necessary to begin the transmission: the address of the slave, high speed mode, expect NACK, number of bytes and the data buffer eChannelType The type of the channel (LPI2C or FlexIO).

Returns

Std_ReturnType.

Return values

E_NOT_OK	In case of a time out situation only.
E_OK	Otherwise.

6.3.6.11 I2c_Ipw_AsyncTransmit()

```
Std_ReturnType I2c_Ipw_AsyncTransmit (
            uint8 Channel,
            const I2c_RequestType * Request,
            const I2c_HwUnitConfigType * HwConfigType )
```

Starts sending or receiving an I2c message from the slave.

Generate (repeated) START and send the address of the slave to initiate a transmission.

Parameters

in	*Channel*	I2c channel to be addressed.
in	*Request*	Pointer to the structure that contains the information necessary to begin the transmission: the address of the slave, high speed mode, expect NACK, number of bytes and the data buffer HwConfigType Pointer to the configuration structure

Returns

Std_ReturnType.

Return values

E_NOT_OK	In case of a time out situation only.
E_OK	Otherwise.

6.3.6.12 I2c_Ipw_PrepareSlaveBuffer()

```
void I2c_Ipw_PrepareSlaveBuffer (
            uint8 Channel,
```

```
                uint8 NumberOfBytes,
                I2c_DataType * DataBuffer )
```

Prepare the RX or TX buffer for a slave channel.

Prepares a RX or TX buffer that will be used to receive data or send data when requested by the master.

Parameters

in	*Channel*	I2c channel to be addressed.
in	*NumberOfBytes*	Maximum number of bytes.
in	*DataBuffer*	Pointer to data buffer

Returns

void

6.3.6.13 I2c_Ipw_GetStatus()

```
I2c_StatusType I2c_Ipw_GetStatus (
                const uint8 Channel,
                const I2c_HwUnitConfigType * HwConfigType )
```

Gets the status of an I2c channel.

The function will check for error flags and return the status of a channel.

Parameters

in	*Channel*	I2c channel to be addressed. eChannelType The type of the channel (LPI2C or FlexIO).

Returns

I2c_StatusType.

Return values

I2C_CH_IDLE	In case the channel was just initialized and not request is pending.
I2C_CH_SEND	In case the channel is busy sending data.
I2C_CH_RECEIVE	In case the channel is busy receiving data.
I2C_CH_FINISHED	In case a transmission or reception of bytes has finished.
I2C_CH_ERROR_PRESENT	In case an error is present.

6.3.7 Variable Documentation

6.3.7.1 I2c_as8ChannelHardwareMap

```
sint8 I2c_as8ChannelHardwareMap[I2C_HW_MAX_MODULES]  [extern]
```

I2c_h_REF_1 MISRA 2012 Required Rule 19.15, Repeated include file I2c_h_REF_3 MISRA 2012 Advisory Rule 19.1, only preprocessor statements and comments before '#include' I2c_h_REF_4 This is incorrectly reported by the PCLint tool.

6.3.7.2 I2c_apDemCfg

```
const I2c_DemConfigType* I2c_apDemCfg[I2C_MAX_CORE_ID]  [extern]
```

I2c_h_REF_4 This is incorrectly reported by the PCLint tool.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C–5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C–Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2023 NXP B.V.