Document Title	Specification of I2C Driver
Document Category	Software specification (SWS)
Document UID	1101
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Specification items and constraints promoted from draft to valid • API I2C_GetVersionInfo and configuration parameter I2CVersionInfoApi removed • Data type I2C_DataPtrType removed; I2C_SetupEB uses uint8 pointers • Editorial fixes
2025-11-27	R25-11	AUTOSAR Release Management	• Use case "combined sequence with repeated start" clarified • Spelling I2C harmonized • Bugfixes
2024-11-27	R24-11	AUTOSAR Release Management	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 5
2 Acronyms and Abbreviations 6
3 Related documentation 7
3.1 Input documents . . . 7
3.2 Related standards and norms . . . 7
3.3 Related specification . . . 7
4 Constraints and assumptions 8
4.1 Limitations . . . 8
4.2 Applicability to car domains . . . 8
5 Dependencies to other modules 9
6 Requirements Tracing 10
7 Functional specification 12
7.1 Background & Rationale . . . 12
7.2 Common design requirements . . . 12
7.3 Error Classification . . . 13
7.3.1 Development Errors . . . 13
7.3.2 Runtime Errors . . . 14
7.3.3 Production Errors . . . 14
8 API specification 15
8.1 Imported types . . . 15
8.2 Type definitions . . . 15
8.2.1 I2C_AddressType . . . 15
8.2.2 I2C_ChannelStatusType . . . 15
8.2.3 I2C_ChannelType . . . 16
8.2.4 I2C_ConfigType . . . 16
8.2.5 I2C_JobType . . . 17
8.2.6 I2C_NumberOfDataType . . . 17
8.2.7 I2C_SequenceType . . . 17
8.2.8 I2C_SequenceResultType . . . 18
8.3 Function definitions . . . 18
8.3.1 I2C_Init . . . 18
8.3.2 I2C_DeInit . . . 19
8.3.3 I2C_SetupEB . . . 20
8.3.4 I2C_AsyncTransmit . . . 22
8.3.5 I2C_SyncTransmit . . . 25
8.3.6 I2C_GetSequenceResult . . . 26
8.3.7 I2C_GetChannelStatus . . . 27
8.4 Scheduled functions . . . 28

8.4.1 I2C_MainFunction . . . 28
8.5 Expected interfaces . . . 28
8.5.1 Mandatory interfaces . . . 28
8.5.2 Optional interfaces . . . 29
8.5.3 Configurable interfaces . . . 29
8.5.3.1 I2C_SeqEndNotification . . . 29
9 Sequence diagrams 31
9.1 SetupEB/ AsyncTransmit . . . 31
9.2 SetupEB/ SyncTransmit . . . 31
10 Configuration specification 33
10.1 How to read this chapter . . . 33
10.2 Containers and configuration parameters . . . 33
10.2.1 I2C . . . 33
10.2.2 I2CGeneral . . . 34
10.2.3 I2CConfigSet . . . 36
10.2.4 I2CChannel . . . 36
10.2.5 I2CJob . . . 38
10.2.6 I2CSequence . . . 40
10.3 Published Information . . . 41
A Not applicable requirements 42
B Change history of AUTOSAR traceable items 44
B.1 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 44
B.1.1 Added Specification Items in R24-11 . . . 44
B.1.2 Changed Specification Items in R24-11 . . . 44
B.1.3 Deleted Specification Items in R24-11 . . . 44
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 45
B.2.1 Added Specification Items in R25-11 . . . 45
B.2.2 Changed Specification Items in R25-11 . . . 46
B.2.3 Deleted Specification Items in R25-11 . . . 48
B.2.4 Added Constraints in R25-11 . . . 49
B.2.5 Changed Constraints in R25-11 . . . 49
B.2.6 Deleted Constraints in R25-11 . . . 49
B.3 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 49
B.3.1 Added Specification Items in R26-11 . . . 49
B.3.2 Changed Specification Items in R26-11 . . . 50
B.3.3 Deleted Specification Items in R26-11 . . . 51
B.3.4 Added Constraints in R26-11 . . . 51
B.3.5 Changed Constraints in R26-11 . . . 51
B.3.6 Deleted Constraints in R26-11 . . . 51

1 Introduction and functional overview

The I2C Driver provides services for reading from and writing to devices connected via I2C buses. It provides access to I2C communication to several Targets (e.g. EEPROM).

This specification describes the functionality, API and the configuration of the module I2C Driver.

The I2C Driver implements Controller mode and Target mode. 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. 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.

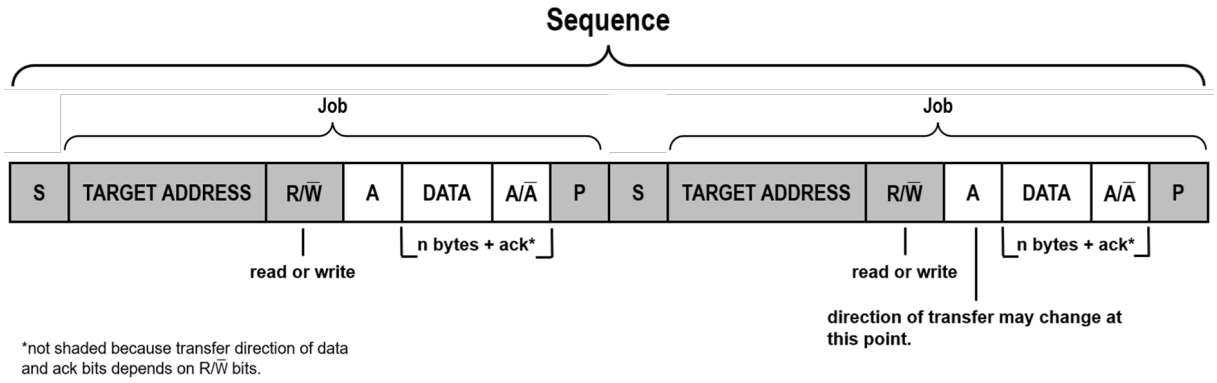

Figure 1.1: Terminology for Sequence and Job

To configure the I2C Driver these steps shall be followed:

- I2C Jobs shall be defined according to data usage provided by the user (EB).
- I2C Sequences consisting of multiple Jobs shall be defined in order to transmit data in a sorted way.

The driver supports multi master mode.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the I2C Driver module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
Controller	A device controlling other devices (Targets). Further it initiates a transfer, generates clock signals and terminates a transfer. Formerly known as "Master".
Rx	Reception (in the context of bus communication)
SCL	Serial Clock
SDA	Serial Data
Target	A device being addressed by a Controller device. Formerly known as "Slave".
TX	Transmission (in the context of bus communication)
Sequence	A Sequence is a number of consecutive Jobs to transmit.
Job	A Job is a software exchange medium for data that are defined with the same criteria: Config. Parameters, Number of Data elements with the same size and data pointers. Jobs can be shared between different Sequences.
Request Queue	A request queue queues only the transmission request without the data itself.
active request	An active request is an ongoing transmission.
pending request	A pending request is a queued transmission request.
EB	Externally buffered Jobs. Buffers containing data to transfer are outside the I2C Driver.

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Specification of MCU Driver
AUTOSAR_CP_SWS_MCUDriver

3.2 Related standards and norms

UM10204 - I2C-bus specification and user manual, NXP Semiconductors, 2021

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for I2C Driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for I2C Driver.

4 Constraints and assumptions

4.1 Limitations

No Limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

The I2C Driver module does not take care of setting the registers which configure the clock, prescaler(s) and PLL in its init function. This has to be done by the MCU module [3].

Note: I2C peripherals may depend on the system clock, prescaler(s) and PLL. Thus, any change of the system clock (e.g. PLL on / PLL off / clock dividers) may also affect the clock settings of the I2C hardware.

6 Requirements Tracing

Requirement	Description	Satisfied by
[CP_RS_I2C_00002]	I2C Driver Transmission Setup	[CP_SWS_I2C_00101] [CP_SWS_I2C_00102] [CP_SWS_I2C_00103] [CP_SWS_I2C_00104] [CP_SWS_I2C_00105] [CP_SWS_I2C_00106] [CP_SWS_I2C_00107] [CP_SWS_I2C_00808] [CP_SWS_I2C_00809] [CP_SWS_I2C_00810] [CP_SWS_I2C_00811] [CP_SWS_I2C_00822]
[CP_RS_I2C_00003]	I2C Driver Data Transfer Speeds	[CP_SWS_I2C_82002]
[CP_RS_I2C_00004]	I2C Driver Multi Controller Mode	[CP_SWS_I2C_00836] [CP_SWS_I2C_00837] [CP_SWS_I2C_00838] [CP_SWS_I2C_82002]
[CP_RS_I2C_00005]	I2C Driver 7-bit and 10-bit Bus Addressing Modes	[CP_SWS_I2C_00803] [CP_SWS_I2C_82002]
[CP_RS_I2C_00006]	I2C Driver Queuing Mechanism for Sequences	[CP_SWS_I2C_00810] [CP_SWS_I2C_00823] [CP_SWS_I2C_80701] [CP_SWS_I2C_82304] [CP_SWS_I2C_82306] [CP_SWS_I2C_82308] [CP_SWS_I2C_82311]
[CP_RS_I2C_00007]	I2C Driver Support for Asynchronous, Interrupt Driven Transmit/ Read Operations	[CP_SWS_I2C_00310] [CP_SWS_I2C_00807] [CP_SWS_I2C_00823] [CP_SWS_I2C_00828] [CP_SWS_I2C_00832] [CP_SWS_I2C_00841] [CP_SWS_I2C_80701] [CP_SWS_I2C_80702] [CP_SWS_I2C_82303] [CP_SWS_I2C_82304] [CP_SWS_I2C_82305] [CP_SWS_I2C_82306] [CP_SWS_I2C_82307] [CP_SWS_I2C_82308] [CP_SWS_I2C_82309] [CP_SWS_I2C_82311] [CP_SWS_I2C_91000] [CP_SWS_I2C_91001]
[CP_RS_I2C_00008]	I2C Driver Support for Synchronous, Non-interrupt Driven Transmit/ Read Operations	[CP_SWS_I2C_00410] [CP_SWS_I2C_00824] [CP_SWS_I2C_80902] [CP_SWS_I2C_82403] [CP_SWS_I2C_82404] [CP_SWS_I2C_82407] [CP_SWS_I2C_82409]
[CP_RS_I2C_00012]	I2C Driver Error Handling	[CP_SWS_I2C_00702] [CP_SWS_I2C_00703] [CP_SWS_I2C_00704] [CP_SWS_I2C_00807] [CP_SWS_I2C_00828] [CP_SWS_I2C_00839] [CP_SWS_I2C_00840]
[CP_RS_I2C_00015]	I2C Target device support	[CP_SWS_I2C_00835]
[CP_RS_I2C_00017]	Independent Treatment of Each Hardware Unit	[CP_SWS_I2C_00823] [CP_SWS_I2C_00824] [CP_SWS_I2C_00835] [CP_SWS_I2C_82002] [CP_SWS_I2C_82303] [CP_SWS_I2C_82304] [CP_SWS_I2C_82306] [CP_SWS_I2C_82308] [CP_SWS_I2C_82311] [CP_SWS_I2C_82403] [CP_SWS_I2C_82404]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[CP_SWS_I2C_82002]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[CP_SWS_I2C_00101] [CP_SWS_I2C_00104] [CP_SWS_I2C_00310] [CP_SWS_I2C_00410] [CP_SWS_I2C_00841] [CP_SWS_I2C_80702] [CP_SWS_I2C_82309] [CP_SWS_I2C_82409]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[CP_SWS_I2C_00821]
[SRS_BSW_00337]	Classification of development errors	[CP_SWS_I2C_00700]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[CP_SWS_I2C_00820]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[CP_SWS_I2C_80701]



△

Requirement	Description	Satisfied by
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[CP_SWS_I2C_00830] [CP_SWS_I2C_00831] [CP_SWS_I2C_00833]
[SRS_BSW_00385]	List possible error notifications	[CP_SWS_I2C_00700] [CP_SWS_I2C_00701] [CP_SWS_I2C_00702] [CP_SWS_I2C_00703] [CP_SWS_I2C_00704]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[CP_SWS_I2C_00702] [CP_SWS_I2C_00703] [CP_SWS_I2C_00704]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[CP_SWS_I2C_82002]
[SRS_BSW_00406]	API handling in uninitialized state	[CP_SWS_I2C_82108]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[CP_SWS_I2C_00820]
[SRS_BSW_00424]	BSW module main processing functions shall not be allowed to enter a wait state	[CP_SWS_I2C_00834]
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[CP_SWS_I2C_00801]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[CP_SWS_I2C_00834]
[SRS_BSW_00452]	Classification of runtime errors	[CP_SWS_I2C_00701] [CP_SWS_I2C_00702] [CP_SWS_I2C_00703] [CP_SWS_I2C_00704]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[CP_SWS_I2C_82307] [CP_SWS_I2C_82308] [CP_SWS_I2C_82407]
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[CP_SWS_I2C_82002] [CP_SWS_I2C_82105] [CP_SWS_I2C_82108]
[SRS_SPAL_12125]	All driver modules shall only initialize the configured resources	[CP_SWS_I2C_82002]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Background & Rationale

The I2C Driver is widely used in the automotive industry. Due to the fact that there was no standard for I2C in AUTOSAR, each company developed its own I2C Driver. In consequence the individuality of their interfaces results in incompatibility with drivers from other manufacturers. For instance, drivers for external hardware that use an I2C Driver among themselves may not cooperate with the I2C Drivers of other companies upon delivery. This principle contradicts the AUTOSAR philosophy, which stands for manufacturer-independent components and aims to reduce the constant redevelopment of similar software components. For this reason, an AUTOSAR standard for I2C is beneficial.

7.2 Common design requirements

[CP_SWS_I2C_00836] Arbitration Detection and Handling in Multimaster I2C Mode

Upstream requirements: CP_RS_I2C_00004

⌈If `I2CMultiMaster` is set, the I2C driver shall implement arbitration detection and handling in compliance with the I2C specification for multimaster mode. During transmission, the master monitors the SDA line and detects if another master is transmitting a dominant bit (logic 0) while it attempts to transmit a recessive bit (logic 1). Upon losing arbitration, the master shall immediately cease transmission and release the bus.⌋

[CP_SWS_I2C_00837] Clock Line Monitoring and Synchronization

Upstream requirements: CP_RS_I2C_00004

⌈If `I2CMultiMaster` is set, the I2C driver shall support clock synchronization by monitoring the SCL line during transmission and respecting clock stretching by other masters or slaves. If the SCL line is held low by another device, the master shall pause its transmission and wait until the line is released (goes high) before proceeding with the next clock cycle.⌋

[CP_SWS_I2C_00838] Bus Idle Check Prior to Transmission

Upstream requirements: CP_RS_I2C_00004

⌈If `I2CMultiMaster` is set, the I2C driver shall verify that the I2C bus is idle before initiating any transmission. The bus is considered idle when both the SDA and SCL lines are in a high state. If the bus is busy (either line is low), the master shall wait until the bus becomes idle before proceeding.⌋

[CP_SWS_I2C_00839] Transmission Retry on Communication Failure

Upstream requirements: CP_RS_I2C_00012

⌈The I2C driver shall implement a retry mechanism for failed transmissions. Failures include arbitration loss, NACK responses, or timeouts. Upon detecting a failure, the driver shall abort the current transaction, wait for the bus to become idle, and retry the transmission up to a maximum number of attempts of `I2CMaximumRetry`.⌋

[CP_SWS_I2C_00840] Bus Recovery from Stuck Condition

Upstream requirements: CP_RS_I2C_00012

⌈The I2C driver shall implement a bus recovery procedure when the bus is detected to be stuck (e.g., SDA held low). The recovery shall include manually toggling the SCL line up to 9 times to release any stuck slave, followed by issuing a STOP condition and reinitializing the I2C peripheral.⌋

7.3 Error Classification

Chapter [2, General Specification of Basic Software Modules] 7.2 *"Error Handling"* [2] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.3.1 Development Errors

[CP_SWS_I2C_00700] Definition of development errors in module I2C

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385

⌈

Type of error	*Related error code*	*Error value*
API service called with wrong parameter	I2C_E_PARAM_JOB	0x00
API service called with wrong parameter	I2C_E_PARAM_SEQUENCE	0x01
API service called with an unexpected value for the pointer	I2C_E_PARAM_POINTER	0x02
API service used without module initialization (see [SWS_BSW_00243] for more details)	I2C_E_UNINIT	0x03
API is called under wrong condition	I2C_E_WRONG_CONDITION	0x04
API is called with unexpected HW unit ID	I2C_E_PARAM_UNIT	0x05

⌋

7.3.2 Runtime Errors

[CP_SWS_I2C_00701] Definition of runtime errors in module I2C

Upstream requirements: SRS_BSW_00452, SRS_BSW_00385

⌈

Type of error	*Related error code*	*Error value*
Error is reported if NACK was received	I2C_E_NACK_RECEIVED	0x00
Error is reported if the master loses arbitration. This usually happens if the SDA is stuck low or high or another master has won the arbitration procedure.	I2C_E_ARBITRATION_FAILURE	0x01
Error is reported in case of FIFO overflow	I2C_E_FIFO_HANDLING	0x02

⌋

[CP_SWS_I2C_00702] Error `I2C_E_FIFO_HANDLING`

Upstream requirements: CP_RS_I2C_00012, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00452

⌈In case the I2C hardware supports a FIFO, any FIFO related error shall be reported as Runtime Error as `I2C_E_FIFO_HANDLING`.⌋

[CP_SWS_I2C_00703] Error `I2C_E_NACK_RECEIVED`

Upstream requirements: CP_RS_I2C_00012, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00452

⌈A NACK error shall be reported as Runtime Error `I2C_E_NACK_RECEIVED`.⌋

[CP_SWS_I2C_00704] `I2C_E_ARBITRATION_FAILURE`

Upstream requirements: CP_RS_I2C_00012, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00452

⌈A loss of arbitration, or the SCL line being stuck low or high, shall be reported as Runtime Error `I2C_E_ARBITRATION_FAILURE`.⌋

Note: This usually happens if the SDA is stuck low or another `Controller` has won the arbitration procedure.

7.3.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[CP_SWS_I2C_00833] Definition of imported datatypes of module I2C

Upstream requirements: SRS_BSW_00384

⌈

Module	Header File	Imported Type
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 I2C_AddressType

[CP_SWS_I2C_00803] Definition of datatype I2C_AddressType

Upstream requirements: CP_RS_I2C_00005

⌈

Name	I2C_AddressType		
Kind	Type		
Derived from	uint16		
Range	0..1023	–	See also `I2CDeviceAddress`
Description	Type to address a Target device which is accessed by the Controller. Values bigger than 0x7F (127) lead to Extended Addressing.		
Available via	I2c.h		

⌋

8.2.2 I2C_ChannelStatusType

[CP_SWS_I2C_91000] Definition of datatype I2C_ChannelStatusType

Upstream requirements: CP_RS_I2C_00007

⌈

Name	I2C_ChannelStatusType		
Kind	Enumeration		
Range	I2C_IDLE	0x01	–
	I2C_BUSY	0x02	–

▽

△

Description	Type to define the range of specific status of I2C HW peripheral.
Available via	I2c.h

⌋

8.2.3 I2C_ChannelType

[CP_SWS_I2C_00808] Definition of datatype I2C_ChannelType

Upstream requirements: CP_RS_I2C_00002

⌈

Name	I2C_ChannelType
Kind	Type
Derived from	uint8
Description	Specifies the identification (ID) for a I2C Hardware microcontroller peripheral (unit).
Available via	I2c.h

⌋

8.2.4 I2C_ConfigType

[CP_SWS_I2C_00801] Definition of datatype I2C_ConfigType

Upstream requirements: SRS_BSW_00438

⌈

Name	I2C_ConfigType	
Kind	Structure	
Elements	Implementation Specific	
	Type	–
	Comment	The contents of the initialization data structure are I2C specific.
Description	This type of external data structure shall contain the initialization data for the I2C Driver.	
Available via	I2c.h	

⌋

8.2.5 I2C_JobType

[CP_SWS_I2C_00809] Definition of datatype I2C_JobType

Upstream requirements: CP_RS_I2C_00002

⌈

Name	I2C_JobType
Kind	Type
Derived from	uint8
Description	This is the type for a Job identifier.
Available via	I2c.h

⌋

8.2.6 I2C_NumberOfDataType

[CP_SWS_I2C_00811] Definition of datatype I2C_NumberOfDataType

Upstream requirements: CP_RS_I2C_00002

⌈

Name	I2C_NumberOfDataType
Kind	Type
Derived from	uint16
Description	Type to define the number of data elements to be sent and / or received during a transmission.
Available via	I2c.h

⌋

8.2.7 I2C_SequenceType

[CP_SWS_I2C_00810] Definition of datatype I2C_SequenceType

Upstream requirements: CP_RS_I2C_00002, CP_RS_I2C_00006

⌈

Name	I2C_SequenceType
Kind	Type
Derived from	uint8
Description	This is the type for a Sequence identifier.
Available via	I2c.h

⌋

8.2.8 I2C_SequenceResultType

[CP_SWS_I2C_00807] Definition of datatype I2C_SequenceResultType

Upstream requirements: CP_RS_I2C_00007, CP_RS_I2C_00012

⌈

Name	I2C_SequenceResultType		
Kind	Enumeration		
Range	I2C_SEQ_OK	0x00	The last transmission of the Sequence has been finished successfully.
	I2C_SEQ_PENDING	0x01	The I2C Driver is performing an I2C Sequence. The meaning of this status is equal to I2C_BUSY.
	I2C_SEQ_QUEUED	0x02	An I2C Sequence is queued and waiting to be transmitted.
	I2C_SEQ_NACK	0x03	An I2C Sequence encountered a NACK signal.
	I2C_SEQ_FAILED	0x04	The last transmission of the Sequence has failed.
Description	This type defines a range of specific Sequences status for the I2C Driver.		
Available via	I2c.h		

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 I2C_Init

[CP_SWS_I2C_00820] Definition of API function I2C_Init

Upstream requirements: SRS_BSW_00358, SRS_BSW_00414

⌈

Service Name	I2C_Init	

Syntax	`void I2C_Init (`
	`  const I2C_ConfigType* ConfigPtr`
	`)`	

Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to I2C Driver configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service initializes the I2C Driver.	

▽

△

Available via	I2c.h

⌋

[CP_SWS_I2C_82002] Initialization

Upstream requirements: CP_RS_I2C_00003, CP_RS_I2C_00004, CP_RS_I2C_00005, CP_RS_I2C_00017, SRS_BSW_00405, SRS_BSW_00101, SRS_SPAL_12057, SRS_SPAL_12125

⌈The function `I2C_Init` shall

- initialize the I2C hardware for each `I2CChannel` using the `I2CChannelBaseAddress` and configure the `I2CBaudRate` accordingly.
- set the sequence result to I2C_SEQ_OK for each `I2CSequence`.

⌋

A re-initialization of an I2C Driver by executing the `I2C_Init` function requires a prior call of `I2C_DeInit`.

8.3.2 I2C_DeInit

[CP_SWS_I2C_00821] Definition of API function I2C_DeInit

Upstream requirements: SRS_BSW_00336

⌈

Service Name	I2C_DeInit

Syntax	`void I2C_DeInit (`
	`  void`
	`)`

Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This service de-initializes the I2C Driver.
Available via	I2c.h

⌋

[CP_SWS_I2C_82105] Deinitialization

Upstream requirements: SRS_SPAL_12057

⌈The function `I2C_DeInit` shall de-initialize the I2C peripheral(s) into a state such as Power On Reset.⌋

Note: This will include a stop of any ongoing transfers.

[CP_SWS_I2C_82108] Development Error Detection

Upstream requirements: SRS_SPAL_12057, SRS_BSW_00406

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_DeInit shall raise the development error I2C_E_UNINIT if the driver is not initialized.⌋

8.3.3 I2C_SetupEB

[CP_SWS_I2C_00822] Definition of API function I2C_SetupEB

Upstream requirements: CP_RS_I2C_00002

⌈

Service Name	I2C_SetupEB	

Syntax	`Std_ReturnType I2C_SetupEB (`
	`  I2C_JobType JobId,`
	`  I2C_AddressType NodeAddress,`
	`  const uint8* TxDataBufferPtr,`
	`  uint8* RxDataBufferPtr,`
	`  I2C_NumberOfDataType Length`
	`)`	

Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	JobId	Job to be used in the transmission.
	NodeAddress	Any other value then zero will override the statically configured node address.
	TxDataBufferPtr	Pointer to the TX transmission data location.
	RxDataBufferPtr	Pointer to the RX transmission data location.
	Length	Length (number of data elements) of the data to be transmitted from TxDataBufferPtr and/or received from RxDataBufferPtr.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Success. E_NOT_OK: Request rejected, due to implementation defined errors.
Description	API shall setup the buffers and the length of data for the EB I2C Driver Job specified.	
Available via	I2c.h	

⌋

[CP_SWS_I2C_00107] Return values

Upstream requirements: CP_RS_I2C_00002

⌈Return values of the function I2C_SetupEB are E_OK: Setup command has been accepted and E_NOT_OK: Setup command has not been accepted.⌋

[CP_SWS_I2C_00101] Development Error Detection

Upstream requirements: CP_RS_I2C_00002, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_SetupEB shall raise the development error I2C_E_PARAM_POINTER if:

- RxDataBufferPtr and TxDataBufferPtr are both NULL pointer
- RxDataBufferPtr and TxDataBufferPtr are both not NULL pointer

⌋

[CP_SWS_I2C_00102] Buffer Pointer Setup EB - Controller mode

Upstream requirements: CP_RS_I2C_00002

⌈When the function I2C_SetupEB is called with the parameter RxDataBuffer-Ptr being a NULL pointer, the according transmission(s) (triggered by I2C_Async-Transmit or I2C_SyncTransmit) shall be a write operation on the I2C bus; if TxDataBufferPtr is a NULL pointer, the according transmission shall be a read operation.⌋

[CP_SWS_I2C_00105] Buffer Pointer Setup EB - Target mode

Upstream requirements: CP_RS_I2C_00002

⌈When the function I2C_SetupEB is called with the parameter TxDataBufferPtr being a NULL pointer, the according listening(s) (triggered by I2C_AsyncTransmit or I2C_SyncTransmit) shall be a read operation on the I2C bus.⌋

[CP_SWS_I2C_00106] Buffer Pointer Setup EB - Target mode

Upstream requirements: CP_RS_I2C_00002

⌈When the function I2C_SetupEB is called with the parameter RxDataBufferPtr being a NULL pointer, the according listening(s) (triggered by I2C_AsyncTransmit or I2C_SyncTransmit) shall be a write operation on the I2C bus.⌋

[CP_SWS_I2C_00103] Node Address override Setup EB

Upstream requirements: CP_RS_I2C_00002

⌈When the function I2C_SetupEB is called with the parameter NodeAddress set to

- other than zero
  the node address given by the parameter NodeAddress
- zero
  the configured node address I2CDeviceAddress

shall be used in the according transmission(s) (triggered by I2C_AsyncTransmit or I2C_SyncTransmit).⌋

[CP_SWS_I2C_00104] Development Error Detection

Upstream requirements: CP_RS_I2C_00002, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_SetupEB shall raise the development error I2C_E_PARAM_JOB if JobId is invalid (i.e. is not configured in I2CJobId).⌋

8.3.4 I2C_AsyncTransmit

[CP_SWS_I2C_00823] Definition of API function I2C_AsyncTransmit

Upstream requirements: CP_RS_I2C_00017, CP_RS_I2C_00006, CP_RS_I2C_00007

⌈

Service Name	I2C_AsyncTransmit	
Syntax	`Std_ReturnType I2C_AsyncTransmit ( I2C_SequenceType SequenceId )`	
Service ID [hex]	0x03	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SequenceId	Sequence used for data exchange.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Success. E_NOT_OK: Request rejected.
Description	The service conducts an asynchronous data transmission according to the parameters provided to the I2C_SetupEB() service. The callback I2C_SeqEndNotification() is called, when the asynchronous operation has finished.	
Available via	I2c.h	

⌋

[CP_SWS_I2C_82303] No transmission is ongoing

Upstream requirements: CP_RS_I2C_00007, CP_RS_I2C_00017

⌈When the function I2C_AsyncTransmit is called and no other Sequence on the same I2CChannel is in state I2C_SEQ_PENDING, the I2C Driver shall take over the given parameter set by I2C_SetupEB, initiate a transmission of Length bytes, set the sequence result to I2C_SEQ_PENDING and return E_OK.⌋

[CP_SWS_I2C_82304] Another transmission is ongoing

Upstream requirements: CP_RS_I2C_00007, CP_RS_I2C_00017, CP_RS_I2C_00006

⌈When the function I2C_AsyncTransmit is called and any other Sequence on the same I2CChannel is in state I2C_SEQ_PENDING, the I2C Driver shall queue the request in FIFO order, set the state I2C_SEQ_QUEUED, and return E_OK.⌋

[CP_SWS_I2C_82306] Another transmission with a shared Job is ongoing

Upstream requirements: CP_RS_I2C_00007, CP_RS_I2C_00017, CP_RS_I2C_00006

⌈When the function I2C_AsyncTransmit is called and the requested Sequence shares on the same I2CChannel a Job (i.e. reference to the same I2CJob) in another Sequence which is in state I2C_SEQ_PENDING, the I2C Driver shall reject the request and return E_NOT_OK.⌋

[CP_SWS_I2C_82311] High priority transmission

Upstream requirements: CP_RS_I2C_00007, CP_RS_I2C_00017, CP_RS_I2C_00006

⌈When the function I2C_AsyncTransmit is called and another Sequence on the same I2CChannel is in state I2C_SEQ_PENDING and has a Job with a higher I2CJobPriority, the I2C Driver shall either:

- if I2CSeqInterruptible set to true, interrupt the current Sequence with the higher prior Sequence. After this Sequence is transmitted the interrupted Sequence shall be finished.
- if I2CSeqInterruptible set to false, transmit the higher-prior Sequence immediately after the current Sequence has finished.

Finally it shall return E_OK.

Note: A Job is atomic and cannot be interrupted.⌋

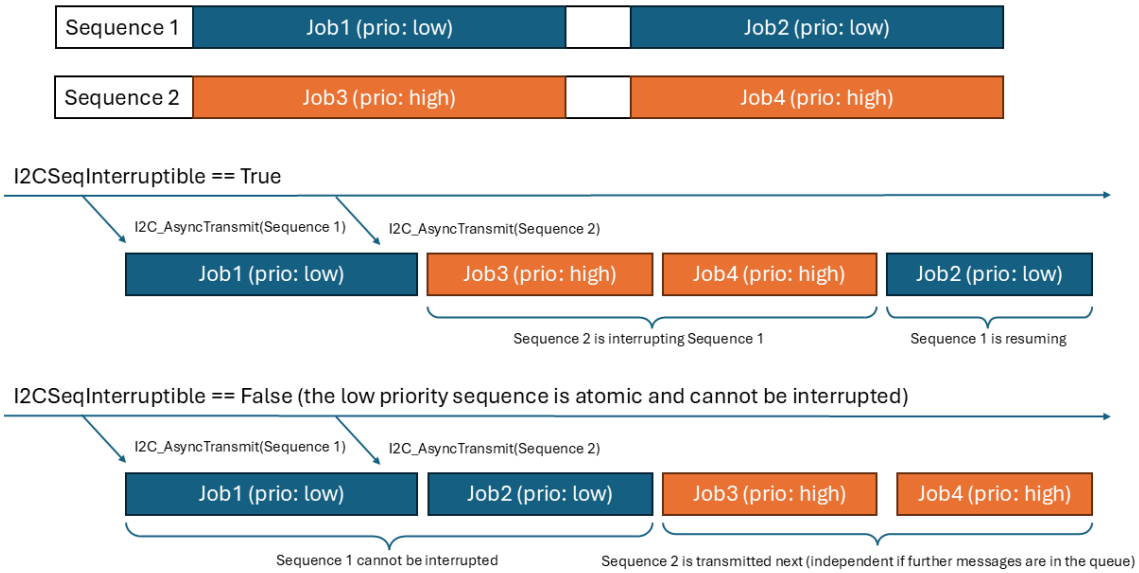

Figure 8.1: Interruptible Sequences

[CP_SWS_I2C_82305] The same transmission is ongoing

Upstream requirements: CP_RS_I2C_00007

⌈When the function I2C_AsyncTransmit is called and the requested Sequence is already in state I2C_SEQ_PENDING, the I2C Driver shall not take this new request into account. In that case, the function shall return with value E_NOT_OK.⌋

[CP_SWS_I2C_82307] Multiple Jobs

Upstream requirements: CP_RS_I2C_00007, SRS_SPAL_00157

⌈If the I2CSequence consists of multiple Jobs (I2CAssignedJob), the function I2C_AsyncTransmit shall transmit from the first Job up to the last Job in the Sequence. Further it shall be assured that the first Job is using the START condition (S) and all subsequent Jobs in the sequence the repeated START condition (Sr). After the last job is executed or if any error occurred, the I2C driver shall set the state to

- I2C_SEQ_FAILED, if any failure occurred
- I2C_SEQ_NACK, if a NACK was received
- I2C_SEQ_OK, if everything was successfully executed.

and afterwards invoke the sequence notification callback function I2C_SeqEndNotification.

⌋

[CP_SWS_I2C_82308] Continuation with Queued Elements

Upstream requirements: CP_RS_I2C_00007, SRS_SPAL_00157, CP_RS_I2C_00017, CP_RS_I2C_00006

⌈After the last job is executed and FIFO queue is not empty (i.e. any other Sequence on the same I2CChannel is in state I2C_SEQ_QUEUED), the I2C driver shall transmit the next FIFO element similar to [CP_SWS_I2C_82303] and remove this element from the queue.⌋

[CP_SWS_I2C_82309] Development Error Detection

Upstream requirements: CP_RS_I2C_00007, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_AsyncTransmit shall raise the development error I2C_E_PARAM_SEQUENCE if the function I2C_SetupEB is not called once in advance for all Jobs in the Sequence.⌋

Note: I2C_SetupEB does not need to be called every time before a call to I2C_AsyncTransmit, e.g. if the EB parameter have not changed a single call of I2C_SetupEB is enough.

[CP_SWS_I2C_00310] Development Error Detection

Upstream requirements: CP_RS_I2C_00007, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_AsyncTransmit shall raise the development error I2C_E_PARAM_SEQUENCE if SequenceId is invalid (i.e. is not configured in I2CSequenceId).⌋

8.3.5 I2C_SyncTransmit

[CP_SWS_I2C_00824] Definition of API function I2C_SyncTransmit

Upstream requirements: CP_RS_I2C_00017, CP_RS_I2C_00008

⌈

Service Name	I2C_SyncTransmit	
Syntax	`Std_ReturnType I2C_SyncTransmit ( I2C_SequenceType SequenceId )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SequenceId	Sequence used for data exchange.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Success. E_NOT_OK: Request rejected.
Description	This service sends or receives data using a blocking mechanism according to the parameters provided to the I2C_SetupEB() service.	
Available via	I2c.h	

⌋

[CP_SWS_I2C_82403] No transmission is ongoing

Upstream requirements: CP_RS_I2C_00008, CP_RS_I2C_00017

⌈When the function I2C_SyncTransmit is called and no other Sequence on the same I2CChannel is in state I2C_SEQ_PENDING, the I2C Driver shall take over the given parameter set by I2C_SetupEB, initiate a transmission of Length bytes, set the sequence result to I2C_SEQ_PENDING and return E_OK.⌋

[CP_SWS_I2C_82404] Another transmission is ongoing

Upstream requirements: CP_RS_I2C_00008, CP_RS_I2C_00017

⌈When the function I2C_SyncTransmit is called and any other asynchronous Sequence on the same I2CChannel is in state I2C_SEQ_PENDING, the I2C Driver shall reject the request and return E_NOT_OK.⌋

[CP_SWS_I2C_82407] Multiple Jobs

Upstream requirements: CP_RS_I2C_00008, SRS_SPAL_00157

⌈If the I2CSequence consists of multiple Jobs (I2CAssignedJob), the function I2C_SyncTransmit shall transmit from the first Job up to the last Job in the Sequence. Further it shall be assured that the first Job is using the START condition (S) and all subsequent Jobs in the sequence the repeated START condition (Sr). After the last job is executed, I2C driver shall return E_OK.⌋

[CP_SWS_I2C_82409] Development Error Detection

Upstream requirements: CP_RS_I2C_00008, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_SyncTransmit shall raise the development error I2C_E_PARAM_SEQUENCE if the function I2C_SetupEB is not called once in advance for all Jobs in the Sequence.⌋

Note: I2C_SetupEB does not need to be called every time before a call to I2C_SyncTransmit, e.g. if the EB parameter have not changed a single call of I2C_SetupEB is enough.

[CP_SWS_I2C_00410] Development Error Detection

Upstream requirements: CP_RS_I2C_00008, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_SyncTransmit shall raise the development error I2C_E_PARAM_SEQUENCE if SequenceId is invalid (i.e. is not configured in I2CSequenceId).⌋

8.3.6 I2C_GetSequenceResult

[CP_SWS_I2C_00828] Definition of API function I2C_GetSequenceResult

Upstream requirements: CP_RS_I2C_00007, CP_RS_I2C_00012

⌈

Service Name	I2C_GetSequenceResult	
Syntax	`I2C_SequenceResultType I2C_GetSequenceResult ( I2C_SequenceType SequenceId )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SequenceId	Sequence to query.
Parameters (inout)	None	
Parameters (out)	None	
Return value	I2C_SequenceResultType	Return the current status
Description	This service returns the current status of the given SequenceId.	
Available via	I2c.h	

⌋

[CP_SWS_I2C_80701] Sequence Result

Upstream requirements: CP_RS_I2C_00006, CP_RS_I2C_00007, SRS_BSW_00377

⌈I2C_GetSequenceResult function shall return

- I2C_SEQ_OK when the current transmission of the Sequence has been finished successfully.
- I2C_SEQ_PENDING when the I2C Driver is performing an I2C Sequence.

- I2C_SEQ_FAILED when the current transmission of the Sequence has failed.
- I2C_SEQ_NACK a NACK signal was encountered.
- I2C_SEQ_QUEUED when an I2C Sequence is queued and waiting to be transmitted.

⌋

[CP_SWS_I2C_80702] Development Error Detection

Upstream requirements: CP_RS_I2C_00007, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_GetSequenceResult shall raise the development error I2C_E_PARAM_SEQUENCE if SequenceId is invalid (i.e. is not configured in I2CSequenceId).⌋

8.3.7 I2C_GetChannelStatus

[CP_SWS_I2C_91001] Definition of API function I2C_GetChannelStatus

Upstream requirements: CP_RS_I2C_00007

⌈

Service Name	I2C_GetChannelStatus	
Syntax	`I2C_ChannelStatusType I2C_GetChannelStatus ( I2C_ChannelType Channel )`	
Service ID [hex]	0x0B	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	I2C hardware microcontroller peripheral (unit) ID.
Parameters (inout)	None	
Parameters (out)	None	
Return value	I2C_ChannelStatusType	I2C_IDLE: 0x01 I2C_BUSY: 0x02
Description	This service returns the status of the specified I2C Hardware peripheral.	
Available via	I2c.h	

⌋

[CP_SWS_I2C_00835] I2C_GetChannelStatus

Upstream requirements: CP_RS_I2C_00015, CP_RS_I2C_00017

⌈I2C_GetChannelStatus shall return the status of specified I2C Hardware unit:

- I2C_BUSY, if given HW unit is busy transmitting a Sequence
- I2C_IDLE, if given HW unit is not transmitting any sequences

⌋

[CP_SWS_I2C_00841] Development Error Detection

Upstream requirements: CP_RS_I2C_00007, SRS_BSW_00323

⌈If development error detection is enabled (I2CDevErrorDetect == True), the function I2C_GetChannelStatus shall raise the development error I2C_E_PARAM_SEQUENCE if Channel is invalid (i.e. is not configured in I2CChannel).⌋

8.4 Scheduled functions

8.4.1 I2C_MainFunction

[CP_SWS_I2C_00834] Definition of scheduled function I2C_MainFunction

Upstream requirements: SRS_BSW_00424, SRS_BSW_00450

⌈

Service Name	I2C_MainFunction
Syntax	`void I2C_MainFunction ( void )`
Service ID [hex]	0x10
Description	Makes a target channel available for processing requests (addressing). When called, the target channel becomes available for starting incoming or outgoing transfers.
Available via	I2c.h

⌋

[CP_SWS_I2C_80902] Polling Mode

Upstream requirements: CP_RS_I2C_00008

⌈If I2CPollingMode is set to I2C_CHANNEL_MODE_POLLING, the I2C driver shall use the I2C_MainFunction to poll the hardware and not use interrupts.⌋

8.5 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.5.1 Mandatory interfaces

The I2C Driver module requires some interfaces to fulfill its core functionality.

[CP_SWS_I2C_00831] Definition of mandatory interfaces required by module I2C

Upstream requirements: SRS_BSW_00384

⌈

API Function	Header File	Description
There are no mandatory interfaces.		

⌋

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the I2C Driver module.

[CP_SWS_I2C_00830] Definition of optional interfaces requested by module I2C

Upstream requirements: SRS_BSW_00384

⌈

API Function	Header File	Description
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

⌋

8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

8.5.3.1 I2C_SeqEndNotification

[CP_SWS_I2C_00832] Definition of configurable interface (*I2C_SeqEndNotification)

Upstream requirements: CP_RS_I2C_00007

⌈

Service Name	(*I2C_SeqEndNotification)

Syntax	`void (*I2C_SeqEndNotification) (`
	`  I2C_SequenceType SequenceId,`
	`  I2C_SequenceResultType Result`
	`)`

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SequenceId	Sequence which is finished.
	Result	Status of currently executed sequence.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback routine provided by the user for each Sequence to notify the caller that a Sequence has been finished.	
Available via	I2c_Externals.h	

⌋

Note: This routine might be called on interrupt level, depending on the calling function.

9 Sequence diagrams

9.1 SetupEB/ AsyncTransmit

The following sequence diagram shows an example of the process of I2C_SetupEB/I2C_AsyncTransmit calls for one Sequence transmission composed of 3 Jobs. Write or Read accesses are "user dependent". In the beginning the module is initialized and at the end again de-initialized.

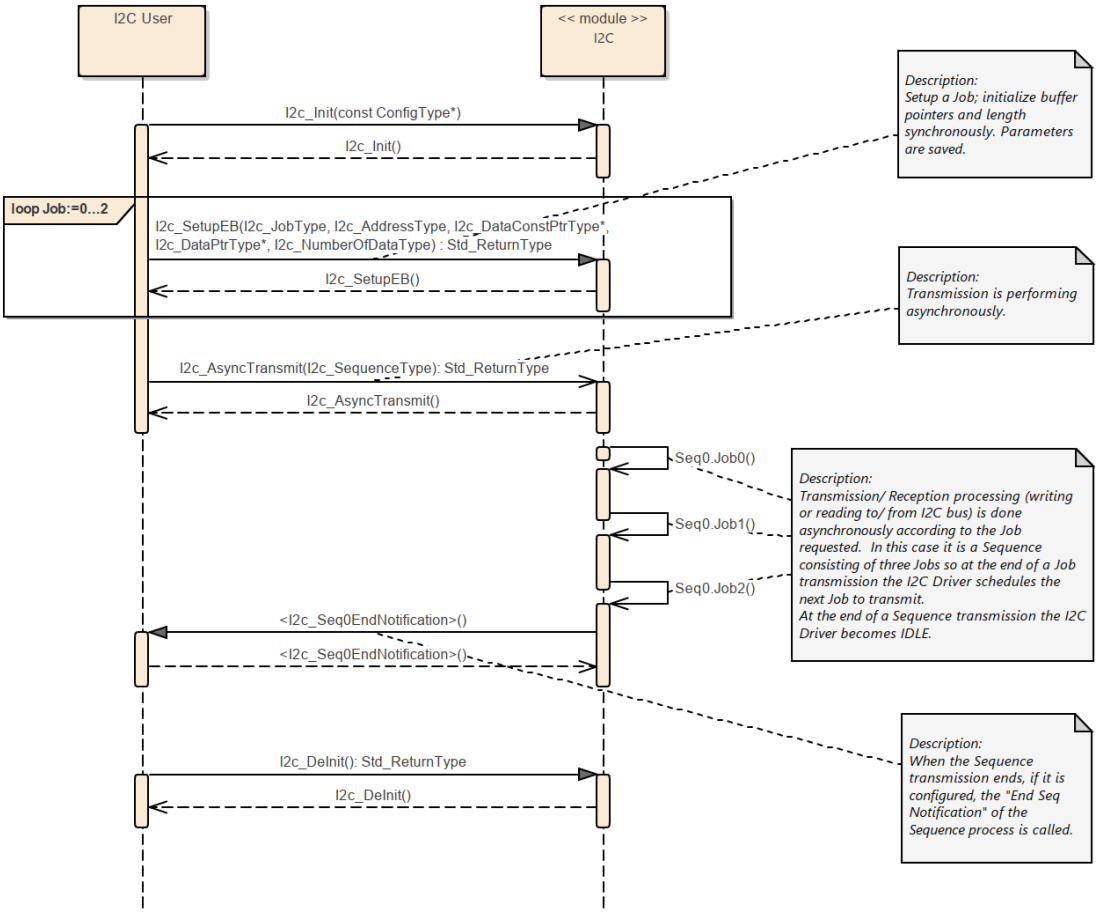

Figure 9.1: SetupEB/ AsyncTransmit

9.2 SetupEB/ SyncTransmit

The following sequence diagram shows an example of the process of I2C_SetupEB/I2C_SyncTransmit calls for two Sequence transmissions composed of 2 Jobs each. Write or Read accesses are "user dependent". In the beginning the module is initialized and at the end again de-initialized.

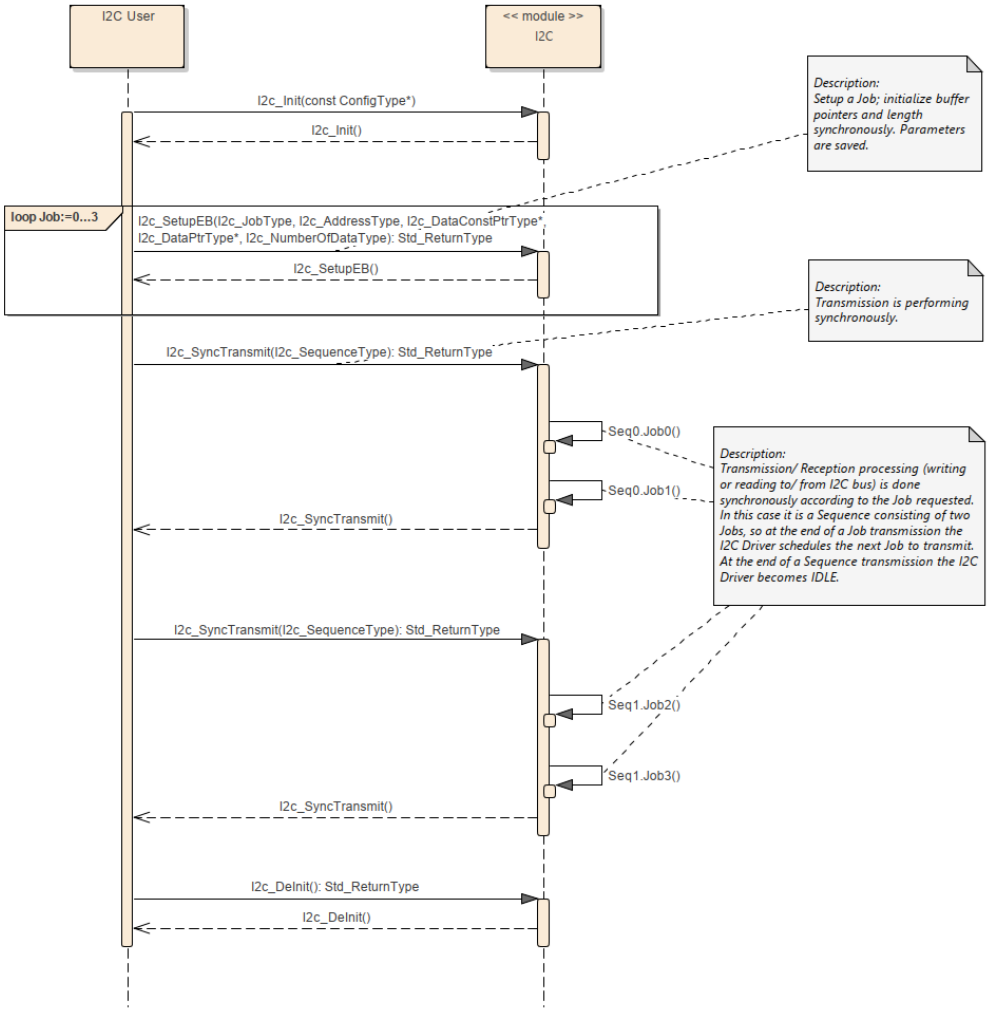

Figure 9.2: SetupEB/ SyncTransmit

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module I2C driver.

Chapter 10.3 specifies published information of the module I2C driver.

10.1 How to read this chapter

For details refer to [2] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in Chapter 7 and Chapter 8. Further hardware/ implementation specific parameters can be added if necessary.

10.2.1 I2C

[ECUC_I2C_00001] Definition of EcucModuleDef I2C ⌈

Module Name	I2C
Description	Configuration of the I2C (Inter-Integrated Circuit) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
I2CConfigSet	1..*	The base for a multiple configuration set.
I2CGeneral	1	General configuration parameters of the I2C.

⌋

[CP_SWS_I2C_CONSTR_00001] Partition mapping of I2C kernel ⌈The ECUC partitions referenced by `I2CKernelEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `I2CEcucPartitionRef`.⌋

[CP_SWS_I2C_CONSTR_00002] I2C availability on partitions ⌈If `I2CEcucPartitionRef` references one or more ECUC partitions, `I2CKernelEcuc-`

PartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

10.2.2 I2CGeneral

[ECUC_I2C_00002] Definition of EcucParamConfContainerDef I2CGeneral ⌈

Container Name	I2CGeneral
Parent Container	I2C
Upstream mapping defined	–
Description	General configuration parameters of the I2C.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
I2CDevErrorDetect	1	[ECUC_I2C_00004]
I2CMainFunctionPeriod	0..1	[ECUC_I2C_00023]
I2CMaximumRetry	1	[ECUC_I2C_00025]
I2CMultiMaster	1	[ECUC_I2C_00024]
I2CEcucPartitionRef	0..*	[ECUC_I2C_00021]
I2CKernelEcucPartitionRef	0..1	[ECUC_I2C_00022]

No Included Containers

⌋

[ECUC_I2C_00004] Definition of EcucBooleanParamDef I2CDevErrorDetect ⌈

Parameter Name	I2CDevErrorDetect
Parent Container	I2CGeneral
Upstream mapping defined	–
Description	This parameter switches the Development Error Detection and Notification ON or OFF. • True: Development error detection is enabled. • False: Development error detection is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_I2C_00023] Definition of EcucFloatParamDef I2CMainFunctionPeriod ⌈

Parameter Name	I2CMainFunctionPeriod	
Parent Container	I2CGeneral	
Upstream mapping defined	–	
Description	Period of the I2C main function	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[-INF .. INF]	
Default value	0.01	
Pre-Compile only	true	

⌋

[ECUC_I2C_00025] Definition of EcucIntegerParamDef I2CMaximumRetry ⌈

Parameter Name	I2CMaximumRetry	
Parent Container	I2CGeneral	
Upstream mapping defined	–	
Description	Maximum retries after a transmission fault	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 10	
Default value	3	
Pre-Compile only	true	

⌋

[ECUC_I2C_00024] Definition of EcucBooleanParamDef I2CMultiMaster ⌈

Parameter Name	I2CMultiMaster
Parent Container	I2CGeneral
Upstream mapping defined	–
Description	This parameter defines if the driver is multiple master capable.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_I2C_00021] Definition of EcucReferenceDef I2CEcucPartitionRef ⌈

Parameter Name	I2CEcucPartitionRef	
Parent Container	I2CGeneral	
Upstream mapping defined	–	
Description	Maps the I2C driver to zero or multiple ECUC partitions to make the driver API available in the according partition.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable

▽

△

Post-Build Value	-	Selectable

⌋

[ECUC_I2C_00022] Definition of EcucReferenceDef I2CKernelEcucPartitionRef ⌈

Parameter Name	I2CKernelEcucPartitionRef	
Parent Container	I2CGeneral	
Upstream mapping defined	–	
Description	Maps the I2C kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the I2C driver is mapped to.	
Multiplicity	0..1	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

10.2.3 I2CConfigSet

[ECUC_I2C_00003] Definition of EcucParamConfContainerDef I2CConfigSet ⌈

Container Name	I2CConfigSet
Parent Container	I2C
Upstream mapping defined	–
Description	The base for a multiple configuration set.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
I2CChannel	1..*	The Hardware specific configuration parameters of the I2C.
I2CJob	1..*	Job specific configuration parameters of the I2C.
I2CSequence	1..*	Sequence specific configuration parameters of the I2C.

⌋

10.2.4 I2CChannel

[ECUC_I2C_00008] Definition of EcucParamConfContainerDef I2CChannel ⌈

Container Name	I2CChannel
Parent Container	I2CConfigSet
Upstream mapping defined	–
Description	The Hardware specific configuration parameters of the I2C.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
I2CBaudRate	1	[ECUC_I2C_00014]
I2CChannelBaseAddress	1	[ECUC_I2C_00015]
I2cChannelMode	1	[ECUC_I2C_00016]
I2CPollingMode	1	[ECUC_I2C_00028]

No Included Containers

⌋

[ECUC_I2C_00014] Definition of EcucIntegerParamDef I2CBaudRate ⌈

Parameter Name	I2CBaudRate	
Parent Container	I2CChannel	
Upstream mapping defined	–	
Description	The baud rate of the bus in kbit/s.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	100	
Pre-Compile only	true	

⌋

[ECUC_I2C_00015] Definition of EcucIntegerParamDef I2CChannelBaseAddress ⌈

Parameter Name	I2CChannelBaseAddress	
Parent Container	I2CChannel	
Upstream mapping defined	–	
Description	The register address of the Channel.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_I2C_00016] Definition of EcucEnumerationParamDef I2cChannelMode ⌈

Parameter Name	I2cChannelMode	
Parent Container	I2CChannel	
Upstream mapping defined	–	
Description	Select whether the chanel will be used in Controller or Target mode.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	I2C_CHANNEL_MODE_CONTROLLER	Channel will be used in Controller mode
	I2C_CHANNEL_MODE_TARGET	Channel will be used in Target mode
Default value	I2C_CHANNEL_MODE_CONTROLLER	
Pre-Compile only	true	

⌋

[ECUC_I2C_00028] Definition of EcucEnumerationParamDef I2CPollingMode ⌈

Parameter Name	I2CPollingMode	
Parent Container	I2CChannel	
Upstream mapping defined	–	
Description	Select whether the Channel is used in polling or interrupt mode.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	I2C_CHANNEL_MODE_INTERRUPT	Channel is using interrupts
	I2C_CHANNEL_MODE_POLLING	Channel is polled by I2C_MainFunciton
Default value	I2C_CHANNEL_MODE_INTERRUPT	
Pre-Compile only	true	

⌋

10.2.5 I2CJob

[ECUC_I2C_00006] Definition of EcucParamConfContainerDef I2CJob ⌈

Container Name	I2CJob
Parent Container	I2CConfigSet
Upstream mapping defined	–
Description	Job specific configuration parameters of the I2C.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
I2CDeviceAddress	1	[ECUC_I2C_00018]
I2CJobId	1	[ECUC_I2C_00017]
I2CJobPriority	1	[ECUC_I2C_00027]

No Included Containers

⌋

[ECUC_I2C_00018] Definition of EcucIntegerParamDef I2CDeviceAddress ⌈

Parameter Name	I2CDeviceAddress	
Parent Container	I2CJob	
Upstream mapping defined	–	
Description	The address of a Target device which is accessed by the Controller. Values bigger than `0x7F (127)` lead to Extended Addressing.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 1023	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_I2C_00017] Definition of EcucIntegerParamDef I2CJobId ⌈

Parameter Name	I2CJobId	
Parent Container	I2CJob	
Upstream mapping defined	–	
Description	The identifier of a Job.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_I2C_00027] Definition of EcucIntegerParamDef I2CJobPriority ⌈

Parameter Name	I2CJobPriority	
Parent Container	I2CJob	
Upstream mapping defined	–	
Description	Defines the Priority of a job (0- lower prio , 3- higher prio)	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 3	
Default value	2	
Pre-Compile only	true	

⌋

10.2.6 I2CSequence

[ECUC_I2C_00007] Definition of EcucParamConfContainerDef I2CSequence ⌈

Container Name	I2CSequence
Parent Container	I2CConfigSet
Upstream mapping defined	–
Description	Sequence specific configuration parameters of the I2C.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
I2CEndNotification	1	[ECUC_I2C_00009]
I2CSeqInterruptible	1	[ECUC_I2C_00029]
I2CSequenceId	1	[ECUC_I2C_00011]
I2CAssignedChannel	1	[ECUC_I2C_00012]
I2CAssignedJob	1..*	[ECUC_I2C_00010]

No Included Containers

⌋

[ECUC_I2C_00009] Definition of EcucFunctionNameDef I2CEndNotification ⌈

Parameter Name	I2CEndNotification
Parent Container	I2CSequence
Upstream mapping defined	–
Description	The transmission end notification to inform the user that a transmission request has been serviced.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_I2C_00029] Definition of EcucBooleanParamDef I2CSeqInterruptible ⌈

Parameter Name	I2CSeqInterruptible
Parent Container	I2CSequence
Upstream mapping defined	–
Description	Defines if a sequence can be interrupted by a sequence including jobs with a higher priority
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false

▽

△

Pre-Compile only	true

⌋

[ECUC_I2C_00011] Definition of EcucIntegerParamDef I2CSequenceId ⌈

Parameter Name	I2CSequenceId
Parent Container	I2CSequence
Upstream mapping defined	–
Description	The Id of a I2C Sequence.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 254
Default value	0
Pre-Compile only	true

⌋

[ECUC_I2C_00012] Definition of EcucReferenceDef I2CAssignedChannel ⌈

Parameter Name	I2CAssignedChannel
Parent Container	I2CSequence
Upstream mapping defined	–
Description	References the bus which is assigned to the Job.
Multiplicity	1
Type	Reference to I2CChannel
Pre-Compile only	true

⌋

[ECUC_I2C_00010] Definition of EcucReferenceDef I2CAssignedJob ⌈

Parameter Name	I2CAssignedJob
Parent Container	I2CSequence
Upstream mapping defined	–
Description	Reference to a Job. **Attributes:** requiresIndex=true
Multiplicity	1..*
Type	Reference to I2CJob
Pre-Compile only	true

⌋

10.3 Published Information

For details refer to [2] Chapter 10.3 *“Published Information”*.

A Not applicable requirements

[CP_SWS_I2C_NA_00999] Requirements fulfilled by the General Specification of Basic Software Modules

Upstream requirements: SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00171, SRS_BSW_00339, SRS_BSW_00344, SRS_BSW_00345, SRS_BSW_00380, SRS_BSW_00396, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00402, SRS_BSW_00404, SRS_BSW_00409, SRS_BSW_00423, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00451, SRS_BSW_00458, SRS_BSW_00467, SRS_BSW_00470, SRS_BSW_00488, SRS_BSW_00489, SRS_BSW_00493, SRS_BSW_00496, SRS_BSW_00497

⌈These requirements are fulfilled by the General Specification of Basic Software Modules [2], which is a required specification for the I2C Driver (see Section 3.3). They are not specified again in this document.⌋

[CP_SWS_I2C_NA_00991] No production errors

Upstream requirements: SRS_BSW_00417, SRS_BSW_00422, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472

⌈The I2C Driver does not define production errors (see Section 7.3.3). Therefore these requirements are not applicable.⌋

[CP_SWS_I2C_NA_00992] No security events

Upstream requirements: SRS_BSW_00490, SRS_BSW_00491

⌈The I2C Driver does not define security events. Therefore these requirements are not applicable.⌋

[CP_SWS_I2C_NA_00993] No wake-up capability

Upstream requirements: SRS_BSW_00375, SRS_SPAL_12067, SRS_SPAL_12069, SRS_SPAL_12267

⌈The I2C Driver does not provide wake-up sources and does not handle wake-up events. Therefore these requirements are not applicable.⌋

[CP_SWS_I2C_NA_00994] No operation modes

Upstream requirements: SRS_SPAL_12064, SRS_SPAL_12169

⌈The I2C Driver does not provide different operation modes. Therefore these requirements are not applicable.⌋

[CP_SWS_I2C_NA_00995] Definition of configuration parameters

Upstream requirements: SRS_BSW_00383, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403

⌈These requirements address the definition and documentation of configuration parameters. They are fulfilled by the ECU configuration parameter definition of the I2C Driver

in the configuration specification of this document and are not specified by individual specification items.⌋

[CP_SWS_I2C_NA_00996] ECU integration and implementation

Upstream requirements: SRS_BSW_00416, SRS_BSW_00419, SRS_SPAL_12068, SRS_SPAL_12078, SRS_SPAL_12265, SRS_SPAL_12463

⌈These requirements address the integration of the ECU (initialization sequence, combination of register settings) or the implementation of the module (code efficiency, placement and constancy of configuration data). They are not specified by individual specification items of the I2C Driver.⌋

[CP_SWS_I2C_NA_00997] Requirements not relevant for the I2C Driver

Upstream requirements: SRS_BSW_00004, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00369, SRS_BSW_00425, SRS_BSW_00432, SRS_BSW_00461, SRS_BSW_00478, SRS_SPAL_12077, SRS_SPAL_12092

⌈These requirements are not applicable to the I2C Driver:

- SRS_BSW_00004: the I2C Driver does not import include files of other modules that require a version check.
- SRS_BSW_00168, SRS_BSW_00170: the I2C Driver is not tested via diagnostic services and is not an AUTOSAR software component.
- SRS_BSW_00369: the APIs of the I2C Driver return only E_OK or E_NOT_OK and no development error codes.
- SRS_BSW_00425: the requirement addresses the BSW module description template.
- SRS_BSW_00432, SRS_BSW_00478: the I2C Driver has one main function, which is used only for polling mode; its period is defined by the integration.
- SRS_BSW_00461: the I2C Driver is not called by a generic module.
- SRS_SPAL_12077: `I2C_SyncTransmit` is blocking by definition; all other APIs are non-blocking.
- SRS_SPAL_12092: no handler or manager is defined for the I2C Driver.

⌋

B Change history of AUTOSAR traceable items

B.1 Traceable item history of this document according to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

[CP_SWS_I2C_00101] [CP_SWS_I2C_00102] [CP_SWS_I2C_00103] [CP_SWS_I2C_00104] [CP_SWS_I2C_00105] [CP_SWS_I2C_00106] [CP_SWS_I2C_00310] [CP_SWS_I2C_00410] [CP_SWS_I2C_00700] [CP_SWS_I2C_00701] [CP_SWS_I2C_00702] [CP_SWS_I2C_00703] [CP_SWS_I2C_00704] [CP_SWS_I2C_00705] [CP_SWS_I2C_00801] [CP_SWS_I2C_00803] [CP_SWS_I2C_00804] [CP_SWS_I2C_00805] [CP_SWS_I2C_00806] [CP_SWS_I2C_00807] [CP_SWS_I2C_00808] [CP_SWS_I2C_00809] [CP_SWS_I2C_00810] [CP_SWS_I2C_00811] [CP_SWS_I2C_00820] [CP_SWS_I2C_00821] [CP_SWS_I2C_00822] [CP_SWS_I2C_00823] [CP_SWS_I2C_00824] [CP_SWS_I2C_00827] [CP_SWS_I2C_00828] [CP_SWS_I2C_00830] [CP_SWS_I2C_00831] [CP_SWS_I2C_00832] [CP_SWS_I2C_00833] [CP_SWS_I2C_00834] [CP_SWS_I2C_00835] [CP_SWS_I2C_80701] [CP_SWS_I2C_80702] [CP_SWS_I2C_80801] [CP_SWS_I2C_80802] [CP_SWS_I2C_80803] [CP_SWS_I2C_80804] [CP_SWS_I2C_80805] [CP_SWS_I2C_80806] [CP_SWS_I2C_80901] [CP_SWS_I2C_82002] [CP_SWS_I2C_82105] [CP_SWS_I2C_82108] [CP_SWS_I2C_82303] [CP_SWS_I2C_82304] [CP_SWS_I2C_82305] [CP_SWS_I2C_82307] [CP_SWS_I2C_82308] [CP_SWS_I2C_82309] [CP_SWS_I2C_82403] [CP_SWS_I2C_82404] [CP_SWS_I2C_82407] [CP_SWS_I2C_82409] [CP_SWS_I2C_82601] [CP_SWS_I2C_82806] [ECUC_I2c_00001] [ECUC_I2c_00002] [ECUC_I2c_00003] [ECUC_I2c_00004] [ECUC_I2c_00005] [ECUC_I2c_00006] [ECUC_I2c_00007] [ECUC_I2c_00008] [ECUC_I2c_00009] [ECUC_I2c_00010] [ECUC_I2c_00011] [ECUC_I2c_00012] [ECUC_I2c_00014] [ECUC_I2c_00015] [ECUC_I2c_00016] [ECUC_I2c_00017] [ECUC_I2c_00018] [ECUC_I2c_00019]

B.1.2 Changed Specification Items in R24-11

none

B.1.3 Deleted Specification Items in R24-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

Number	Heading
[CP_SWS_I2C_00107]	Return values
[CP_SWS_I2C_00836]	Arbitration Detection and Handling in Multimaster I2C Mode
[CP_SWS_I2C_00837]	Clock Line Monitoring and Synchronization)
[CP_SWS_I2C_00838]	Bus Idle Check Prior to Transmission
[CP_SWS_I2C_00839]	Transmission Retry on Communication Failure
[CP_SWS_I2C_00840]	Bus Recovery from Stuck Condition
[CP_SWS_I2C_00841]	Development Error Detection
[CP_SWS_I2C_80902]	Polling Mode
[CP_SWS_I2C_82306]	Another transmission with a shared Job is ongoing
[CP_SWS_I2C_82311]	High priority transmission
[CP_SWS_I2C_91000]	Definition of datatype I2C_ChannelStatusType
[CP_SWS_I2C_91001]	Definition of API function I2C_GetChannelStatus
[ECUC_I2C_00001]	Definition of EcucModuleDef I2C
[ECUC_I2C_00002]	Definition of EcucParamConfContainerDef I2CGeneral
[ECUC_I2C_00003]	Definition of EcucParamConfContainerDef I2CConfigSet
[ECUC_I2C_00004]	Definition of EcucBooleanParamDef I2CDevErrorDetect
[ECUC_I2C_00005]	Definition of EcucBooleanParamDef I2CVersionInfoApi
[ECUC_I2C_00006]	Definition of EcucParamConfContainerDef I2CJob
[ECUC_I2C_00007]	Definition of EcucParamConfContainerDef I2CSequence
[ECUC_I2C_00008]	Definition of EcucParamConfContainerDef I2CChannel
[ECUC_I2C_00009]	Definition of EcucFunctionNameDef I2CEndNotification
[ECUC_I2C_00010]	Definition of EcucReferenceDef I2CAssignedJob
[ECUC_I2C_00011]	Definition of EcucIntegerParamDef I2CSequenceId

▽

△

Number	Heading
[ECUC_I2C_00012]	Definition of EcucReferenceDef I2CAssignedChannel
[ECUC_I2C_00014]	Definition of EcucIntegerParamDef I2CBaudRate
[ECUC_I2C_00015]	Definition of EcucIntegerParamDef I2CChannelBaseAddress
[ECUC_I2C_00016]	Definition of EcucEnumerationParamDef I2cChannelMode
[ECUC_I2C_00017]	Definition of EcucIntegerParamDef I2CJobId
[ECUC_I2C_00018]	Definition of EcucIntegerParamDef I2CDeviceAddress
[ECUC_I2C_00021]	Definition of EcucReferenceDef I2CEcucPartitionRef
[ECUC_I2C_00022]	Definition of EcucReferenceDef I2CKernelEcucPartitionRef
[ECUC_I2C_00023]	Definition of EcucFloatParamDef I2CMainFunctionPeriod
[ECUC_I2C_00024]	Definition of EcucBooleanParamDef I2CMultiMaster
[ECUC_I2C_00025]	Definition of EcucIntegerParamDef I2CMaximumRetry
[ECUC_I2C_00027]	Definition of EcucIntegerParamDef I2CJobPriority
[ECUC_I2C_00028]	Definition of EcucEnumerationParamDef I2CPollingMode
[ECUC_I2C_00029]	Definition of EcucBooleanParamDef I2CSeqInterruptible

Table B.1: Added Specification Items in R25-11

B.2.2 Changed Specification Items in R25-11

Number	Heading
[CP_SWS_I2C_00101]	Development Error Detection
[CP_SWS_I2C_00103]	Node Address override Setup EB
[CP_SWS_I2C_00104]	Development Error Detection
[CP_SWS_I2C_00105]	Buffer Pointer Setup EB - `Target` mode
[CP_SWS_I2C_00106]	Buffer Pointer Setup EB - `Target` mode
[CP_SWS_I2C_00310]	Development Error Detection
[CP_SWS_I2C_00410]	Development Error Detection
[CP_SWS_I2C_00700]	Definition of development errors in module I2C
[CP_SWS_I2C_00701]	Definition of runtime errors in module I2C

▽

△

Number	Heading
[CP_SWS_I2C_00704]	I2C_E_ARBITRATION_FAILURE
[CP_SWS_I2C_00803]	Definition of datatype I2C_AddressType
[CP_SWS_I2C_00808]	Definition of datatype I2C_ChannelType
[CP_SWS_I2C_00822]	Definition of API function I2C_SetupEB
[CP_SWS_I2C_00823]	Definition of API function I2C_AsyncTransmit
[CP_SWS_I2C_00824]	Definition of API function I2C_SyncTransmit
[CP_SWS_I2C_00830]	Definition of optional interfaces requested by module I2C
[CP_SWS_I2C_00833]	Definition of imported datatypes of module I2C
[CP_SWS_I2C_00835]	I2C_GetChannelStatus
[CP_SWS_I2C_80702]	Development Error Detection
[CP_SWS_I2C_82002]	Initialzation
[CP_SWS_I2C_82108]	Development Error Detection
[CP_SWS_I2C_82303]	No transmission is ongoing
[CP_SWS_I2C_82304]	Another transmission is ongoing
[CP_SWS_I2C_82307]	Multiple Jobs
[CP_SWS_I2C_82308]	Continuation with Queued Elements
[CP_SWS_I2C_82309]	Development Error Detection
[CP_SWS_I2C_82403]	No transmission is ongoing
[CP_SWS_I2C_82404]	Another transmission is ongoing
[CP_SWS_I2C_82407]	Multiple Jobs
[CP_SWS_I2C_82409]	Development Error Detection

▽

△

Number	Heading
[CP_SWS_I2C_82601]	Development Error Detection

Table B.2: Changed Specification Items in R25-11

B.2.3 Deleted Specification Items in R25-11

Number	Heading
[CP_SWS_I2C_00705]	`I2C_E_BUS_FAILURE`
[CP_SWS_I2C_00804]	Definition of datatype I2C_DataType
[CP_SWS_I2C_00806]	Definition of datatype I2C_DataConstPtrType
[CP_SWS_I2C_80801]	No listening is ongoing
[CP_SWS_I2C_80802]	Driver is in listening mode
[CP_SWS_I2C_80803]	Last Message or any error received
[CP_SWS_I2C_80804]	Development Error Detection
[CP_SWS_I2C_80805]	I2C_E_WRONG_MODE
[CP_SWS_I2C_80806]	Message received
[CP_SWS_I2C_80901]	Permanent listening (`Target` Mode)
[CP_SWS_I2C_82806]	Development Error Detection
[ECUC_I2c_00001]	Definition of EcucModuleDef I2c
[ECUC_I2c_00002]	Definition of EcucParamConfContainerDef I2cGeneral
[ECUC_I2c_00003]	Definition of EcucParamConfContainerDef I2cConfigSet
[ECUC_I2c_00004]	Definition of EcucBooleanParamDef I2cDevErrorDetect
[ECUC_I2c_00005]	Definition of EcucBooleanParamDef I2cVersionInfoApi
[ECUC_I2c_00006]	Definition of EcucParamConfContainerDef I2cJob
[ECUC_I2c_00007]	Definition of EcucParamConfContainerDef I2cSequence
[ECUC_I2c_00008]	Definition of EcucParamConfContainerDef I2cChannel
[ECUC_I2c_00009]	Definition of EcucFunctionNameDef I2cEndNotification
[ECUC_I2c_00010]	Definition of EcucReferenceDef I2cAssignedJob

▽

△

Number	Heading
[ECUC_I2c_00011]	Definition of EcucIntegerParamDef I2cSequenceId
[ECUC_I2c_00012]	Definition of EcucReferenceDef I2cAssignedChannel
[ECUC_I2c_00014]	Definition of EcucIntegerParamDef I2cBaudRate
[ECUC_I2c_00015]	Definition of EcucIntegerParamDef I2cHwUnitBaseAddress
[ECUC_I2c_00016]	Definition of EcucEnumerationParamDef I2cHwUnitMode
[ECUC_I2c_00017]	Definition of EcucIntegerParamDef I2cJobId
[ECUC_I2c_00018]	Definition of EcucIntegerParamDef I2cDeviceAddress
[ECUC_I2c_00019]	Definition of EcucBooleanParamDef I2cTargetListening

Table B.3: Deleted Specification Items in R25-11

B.2.4 Added Constraints in R25-11

Number	Heading
[CP_SWS_I2C_CONSTR_00001]	Partition mapping of I2C kernel
[CP_SWS_I2C_CONSTR_00002]	I2C availability on partitions

Table B.4: Added Constraints in R25-11

B.2.5 Changed Constraints in R25-11

none

B.2.6 Deleted Constraints in R25-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R26-11

B.3.1 Added Specification Items in R26-11

none

B.3.2 Changed Specification Items in R26-11

Number	Heading
[CP_SWS_I2C_00700]	Definition of development errors in module I2C
[CP_SWS_I2C_00701]	Definition of runtime errors in module I2C
[CP_SWS_I2C_00801]	Definition of datatype I2C_ConfigType
[CP_SWS_I2C_00803]	Definition of datatype I2C_AddressType
[CP_SWS_I2C_00807]	Definition of datatype I2C_SequenceResultType
[CP_SWS_I2C_00808]	Definition of datatype I2C_ChannelType
[CP_SWS_I2C_00809]	Definition of datatype I2C_JobType
[CP_SWS_I2C_00810]	Definition of datatype I2C_SequenceType
[CP_SWS_I2C_00811]	Definition of datatype I2C_NumberOfDataType
[CP_SWS_I2C_00822]	Definition of API function I2C_SetupEB
[CP_SWS_I2C_00830]	Definition of optional interfaces requested by module I2C
[CP_SWS_I2C_00831]	Definition of mandatory interfaces required by module I2C
[CP_SWS_I2C_00833]	Definition of imported datatypes of module I2C
[CP_SWS_I2C_00834]	Definition of scheduled function I2C_MainFunction
[ECUC_I2C_00001]	Definition of EcucModuleDef I2C
[ECUC_I2C_00002]	Definition of EcucParamConfContainerDef I2CGeneral
[ECUC_I2C_00003]	Definition of EcucParamConfContainerDef I2CConfigSet
[ECUC_I2C_00004]	Definition of EcucBooleanParamDef I2CDevErrorDetect
[ECUC_I2C_00006]	Definition of EcucParamConfContainerDef I2CJob
[ECUC_I2C_00007]	Definition of EcucParamConfContainerDef I2CSequence
[ECUC_I2C_00008]	Definition of EcucParamConfContainerDef I2CChannel
[ECUC_I2C_00009]	Definition of EcucFunctionNameDef I2CEndNotification
[ECUC_I2C_00010]	Definition of EcucReferenceDef I2CAssignedJob
[ECUC_I2C_00011]	Definition of EcucIntegerParamDef I2CSequenceId
[ECUC_I2C_00012]	Definition of EcucReferenceDef I2CAssignedChannel

▽

△

Number	Heading
[ECUC_I2C_00014]	Definition of EcucIntegerParamDef I2CBaudRate
[ECUC_I2C_00015]	Definition of EcucIntegerParamDef I2CChannelBaseAddress
[ECUC_I2C_00016]	Definition of EcucEnumerationParamDef I2cChannelMode
[ECUC_I2C_00017]	Definition of EcucIntegerParamDef I2CJobId
[ECUC_I2C_00018]	Definition of EcucIntegerParamDef I2CDeviceAddress
[ECUC_I2C_00023]	Definition of EcucFloatParamDef I2CMainFunctionPeriod

Table B.5: Changed Specification Items in R26-11

B.3.3 Deleted Specification Items in R26-11

Number	Heading
[CP_SWS_I2C_00805]	Definition of datatype I2C_DataPtrType
[CP_SWS_I2C_00827]	Definition of API function I2C_GetVersionInfo
[CP_SWS_I2C_82601]	Development Error Detection
[ECUC_I2C_00005]	Definition of EcucBooleanParamDef I2CVersionInfoApi

Table B.6: Deleted Specification Items in R26-11

B.3.4 Added Constraints in R26-11

none

B.3.5 Changed Constraints in R26-11

none

B.3.6 Deleted Constraints in R26-11

none