Document Title	Specification of I2C Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1101

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
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.

Contents

1 Introduction and functional overview

The I2C Driver provides services for reading from and writing to devices connected via I2C busses. 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.

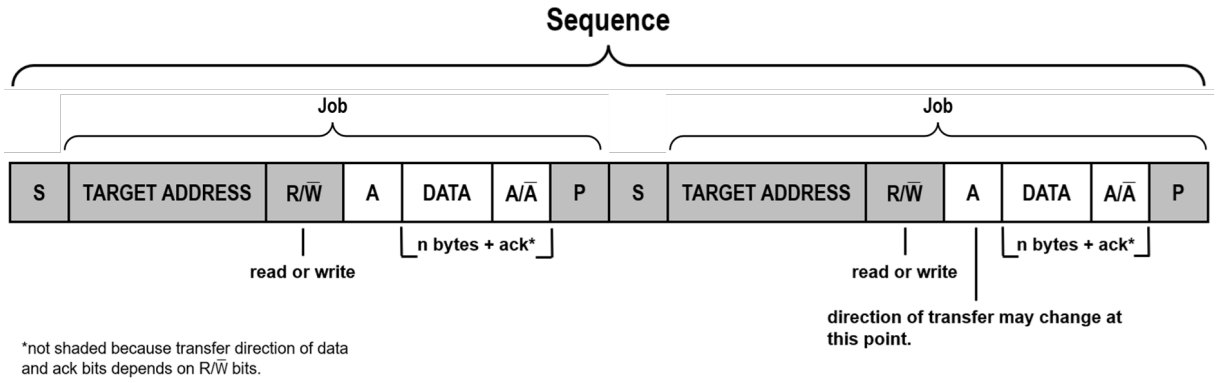

Figure 1.1: Terminology for Sequence and Job

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.

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)
EB	Externally buffered Jobs. Buffers containing data to transfer are outside the I2C Driver.
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.
Sequence	A Sequence is a number of consecutive Jobs to transmit.

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_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_82002]
[CP_RS_I2C_00005]	I2C Driver 7-bit and 10-bit Bus Addressing Modes	[CP_SWS_I2C_82002]
[CP_RS_I2C_00006]	I2C Driver Queuing Mechanism for Sequences	[CP_SWS_I2C_00823] [CP_SWS_I2C_80701] [CP_SWS_I2C_82304] [CP_SWS_I2C_82308]
[CP_RS_I2C_00007]	I2C Driver Support for Asynchronous, Interrupt Driven Transmit/ Read Operations	[CP_SWS_I2C_00310] [CP_SWS_I2C_00823] [CP_SWS_I2C_00828] [CP_SWS_I2C_00832] [CP_SWS_I2C_80701] [CP_SWS_I2C_80702] [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_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_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_00705] [CP_SWS_I2C_00828]
[CP_RS_I2C_00015]	I2C Target device support	[CP_SWS_I2C_00835] [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_82806]
[CP_RS_I2C_00017]	Independent Treatment of Each Hardware Unit	[CP_SWS_I2C_00823] [CP_SWS_I2C_00824] [CP_SWS_I2C_80802] [CP_SWS_I2C_80804] [CP_SWS_I2C_80805] [CP_SWS_I2C_82002] [CP_SWS_I2C_82303] [CP_SWS_I2C_82304] [CP_SWS_I2C_82308] [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_82601]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[CP_SWS_I2C_00821]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[CP_SWS_I2C_00820]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[CP_SWS_I2C_82002]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[CP_SWS_I2C_00820]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[CP_SWS_I2C_80803] [CP_SWS_I2C_80806] [CP_SWS_I2C_82307] [CP_SWS_I2C_82308] [CP_SWS_I2C_82407]

△

Requirement	Description	Satisfied by
[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 Error Classification

Section "Error Handling" of the document "General Specification of Basic Software Modules" [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.

7.2.1 Development Errors

[CP_SWS_I2C_00700] Definiton of development errors in module I2C

Status: DRAFT

⌈

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	I2C_E_UNINIT	0x03
API is called under wrong condition	I2C_E_WRONG_CONDITION	0x04

⌋

7.2.2 Runtime Errors

[CP_SWS_I2C_00701] Definiton of runtime errors in module I2C

Status: DRAFT

⌈

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 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
Error is reported if the SCL line is stuck low	I2C_E_BUS_FAILURE	0x03
The function I2C_StartListening is called while the Target listening mode is set to always listening	I2C_E_WRONG_MODE	0x04

⌋

[CP_SWS_I2C_00702] Error `I2C_E_FIFO_HANDLING`

Status: DRAFT
Upstream requirements: CP_RS_I2C_00012

⌈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`

Status: DRAFT
Upstream requirements: CP_RS_I2C_00012

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

[CP_SWS_I2C_00704] `I2C_E_ARBITRATION_FAILURE`

Status: DRAFT
Upstream requirements: CP_RS_I2C_00012

⌈A loss in arbitration 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.

[CP_SWS_I2C_00705] `I2C_E_BUS_FAILURE`

Status: DRAFT
Upstream requirements: CP_RS_I2C_00012

⌈If the SCL line is stuck low or high, it shall be reported as Runtime Error `I2C_E_BUS_FAILURE`.⌋

7.2.3 Production Errors

There are no production errors.

7.2.4 Extended Production Errors

There are no extended 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

Status: DRAFT

⌈

Module	*Header File*	*Imported Type*
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋

8.2 Type definitions

8.2.1 I2C_ConfigType

[CP_SWS_I2C_00801] Definition of datatype I2C_ConfigType

Status: DRAFT

⌈

Name	I2C_ConfigType (draft)	
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. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

8.2.2 I2C_AddressType

[CP_SWS_I2C_00803] Definition of datatype I2C_AddressType

Status: DRAFT

⌈

Name	I2C_AddressType (draft)
Kind	Type
Derived from	uint16
Description	– **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.2.3 I2C_DataType

[CP_SWS_I2C_00804] Definition of datatype I2C_DataType

Status: DRAFT

⌈

Name	I2C_DataType (draft)
Kind	Type
Derived from	uint8
Description	This type defines the data to be transmitted using the I2C Driver. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.2.4 I2C_DataPtrType

[CP_SWS_I2C_00805] Definition of datatype I2C_DataPtrType

Status: DRAFT

⌈

Name	I2C_DataPtrType (draft)
Kind	Pointer
Type	uint8*
Description	Definition for the pointer type for general buffer handling. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.2.5 I2C_DataConstPtrType

[CP_SWS_I2C_00806] Definition of datatype I2C_DataConstPtrType

Status: DRAFT

⌈

Name	I2C_DataConstPtrType (draft)
Kind	Const Pointer
Type	const uint8*
Description	Definition for the pointer type for TX buffer handling. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.2.6 I2C_SequenceResultType

[CP_SWS_I2C_00807] Definition of datatype I2C_SequenceResultType

Status: DRAFT

⌈

Name	I2C_SequenceResultType (draft)		
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. **Tags:** atp.Status=draft		
Available via	I2c.h		

⌋

8.2.7 I2C_HwUnitType

[CP_SWS_I2C_00808] Definition of datatype I2C_HwUnitType

Status: DRAFT

⌈

Name	I2C_HwUnitType (draft)
Kind	Type
Derived from	uint8
Description	Specifies the identification (ID) for a I2C Hardware microcontroller peripheral (unit). **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.2.8 I2C_JobType

[CP_SWS_I2C_00809] Definition of datatype I2C_JobType

Status: DRAFT

⌈

Name	I2C_JobType (draft)
Kind	Type
Derived from	uint8
Description	This is the type for a Job identifier. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.2.9 I2C_SequenceType

[CP_SWS_I2C_00810] Definition of datatype I2C_SequenceType

Status: DRAFT

⌈

Name	I2C_SequenceType (draft)
Kind	Type
Derived from	uint8
Description	This is the type for a Sequence identifier. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.2.10 I2C_NumberOfDataType

[CP_SWS_I2C_00811] Definition of datatype I2C_NumberOfDataType

Status: DRAFT

⌈

Name	I2C_NumberOfDataType (draft)
Kind	Type
Derived from	uint16
Description	Type to define the number of data elements to be sent and / or received during a transmission. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

8.3 Function definitions

8.3.1 I2C_Init

[CP_SWS_I2C_00820] Definition of API function I2C_Init

Status: DRAFT
Upstream requirements: SRS_BSW_00358, SRS_BSW_00414

⌈

Service Name	I2C_Init (draft)	

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. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

[CP_SWS_I2C_82002] Initialzation

Status: DRAFT
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 `I2cHwUnitBaseAddress` and configure the `I2cBaudRate` accordingly.
- set the Sequences result to I2C_SEQ_OK for each `I2cSequence`.

⌋

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

8.3.2 I2C_DeInit

[CP_SWS_I2C_00821] Definition of API function I2C_DeInit

Status: DRAFT
Upstream requirements: SRS_BSW_00336

⌈

Service Name	I2C_DeInit (draft)
Syntax	`void I2C_DeInit ( void )`
Service ID [hex]	0x1
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. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

[CP_SWS_I2C_82105] Deinitialization

Status: DRAFT
Upstream requirements: SRS_SPAL_12057

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

[CP_SWS_I2C_82108] Development Error Detection

Status: DRAFT
Upstream requirements: SRS_SPAL_12057

⌈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 initialzed.⌋

8.3.3 I2C_SetupEB

[CP_SWS_I2C_00822] Definition of API function I2C_SetupEB

Status: DRAFT
Upstream requirements: CP_RS_I2C_00002

⌈

Service Name	I2C_SetupEB (draft)	
Syntax	`Std_ReturnType I2C_SetupEB ( I2C_JobType JobId, I2C_AddressType NodeAddress, I2C_DataConstPtrType* TxDataBufferPtr, I2C_DataPtrType* RxDataBufferPtr, I2C_NumberOfDataType Length )`	
Service ID [hex]	0x2	
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.
Description	Service to setup the buffers and the length of data for the EB I2C Driver Job specified. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

[CP_SWS_I2C_00101] Development Error Detection

Status: DRAFT
Upstream requirements: CP_RS_I2C_00002

⌈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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00002

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

[CP_SWS_I2C_00105] Buffer Pointer Setup EB - `Target` mode

Status: DRAFT
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_StartListening`) shall be a read operation on the I2C bus.⌋

[CP_SWS_I2C_00106] Buffer Pointer Setup EB - `Target` mode

Status: DRAFT
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_StartListening`) shall be a write operation on the I2C bus.⌋

[CP_SWS_I2C_00103] Node Address override Setup EB

Status: DRAFT
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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00002

⌈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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00017, CP_RS_I2C_00006, CP_RS_I2C_00007

⌈

Service Name	I2C_AsyncTransmit (draft)	
Syntax	`Std_ReturnType I2C_AsyncTransmit ( I2C_SequenceType SequenceId )`	
Service ID [hex]	0x3	
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. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

[CP_SWS_I2C_82303] No transmission is ongoing

Status: DRAFT
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

Status: DRAFT
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 in FIFO the request, set the state I2C_SEQ_QUEUED, and return `E_OK`.⌋

[CP_SWS_I2C_82305] The same transmission is ongoing

Status: DRAFT
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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00007, SRS_SPAL_00157

⌈If the `I2cSequence` consisting of multiple Jobs (`I2cAssignedJob`), the function `I2C_AsyncTransmit` shall transmit from the first Job up to the last Job in the Sequence. After the last job is executed or if any error occured, the I2C driver shall set the state to

- I2C_SEQ_FAILED, if any failure occured
- I2C_SEQ_NACK, if a NACK message was recived
- 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

Status: DRAFT
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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00007

⌈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 a 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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00007

⌈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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00017, CP_RS_I2C_00008

⌈

Service Name	I2C_SyncTransmit (draft)	
Syntax	`Std_ReturnType I2C_SyncTransmit ( I2C_SequenceType SequenceId )`	
Service ID [hex]	0x4	
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. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

[CP_SWS_I2C_82403] No transmission is ongoing

Status: DRAFT
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

Status: DRAFT
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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00008, SRS_SPAL_00157

⌈If the I2cSequence consisting of multiple Jobs (I2cAssignedJob), the function I2C_SyncTransmit shall transmit from the first Job up to the last Job in the Sequence. After the last job is executed, I2C driver shall return E_OK.⌋

[CP_SWS_I2C_82409] Development Error Detection

Status: DRAFT
Upstream requirements: CP_RS_I2C_00008

⌈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 a 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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00008

⌈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_GetVersionInfo

[CP_SWS_I2C_00827] Definition of API function I2C_GetVersionInfo

Status: DRAFT

⌈

Service Name	I2C_GetVersionInfo (draft)	

Syntax	`void I2C_GetVersionInfo (`
	`  Std_VersionInfoType* VersionInfo`
	`)`	

Service ID [hex]	0x7	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	VersionInfo	Pointer to where to store the version information of this module.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service returns the version information of this module. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

[CP_SWS_I2C_82601] Development Error Detection

Status: DRAFT
Upstream requirements: SRS_BSW_00323

⌈If development error detection is enabled (`I2cDevErrorDetect` == True), the function `I2C_GetVersionInfo` shall raise the development error `I2C_E_PARAM_POINTER` if the parameter `VersionInfo` is a NULL pointer.⌋

8.3.7 I2C_GetSequenceResult

[CP_SWS_I2C_00828] Definition of API function I2C_GetSequenceResult

Status: DRAFT
Upstream requirements: CP_RS_I2C_00007, CP_RS_I2C_00012

⌈

Service Name	I2C_GetSequenceResult (draft)

Syntax	`I2C_SequenceResultType I2C_GetSequenceResult (`
	`  I2C_SequenceType SequenceId`
	`)`

Service ID [hex]	0x9

▽

△

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. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

[CP_SWS_I2C_80701] Sequence Result

Status: DRAFT
Upstream requirements: CP_RS_I2C_00006, CP_RS_I2C_00007

⌈`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 curent 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

Status: DRAFT
Upstream requirements: CP_RS_I2C_00007

⌈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`).⌋

8.3.8 I2C_StartListening

[CP_SWS_I2C_00835] Definition of API function I2C_StartListening

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015

⌈

Service Name	I2C_StartListening (draft)	
Syntax	`Std_ReturnType I2C_StartListening ( I2C_SequenceType SequenceId )`	
Service ID [hex]	0x0A	
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	Makes a target channel available for processing requests (addressing). When called, the target channel becomes available for starting incoming or outgoing transfers. **Tags:** atp.Status=draft	
Available via	I2c.h	

⌋

[CP_SWS_I2C_80801] No listening is ongoing

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015

⌈When the function `I2C_StartListening` is called the I2C Driver shall take over the given parameter set by `I2C_SetupEB`, initiate a listening, set the sequence result to I2C_SEQ_PENDING and return `E_OK`.⌋

[CP_SWS_I2C_80802] Driver is in listening mode

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015, CP_RS_I2C_00017

⌈When the function `I2C_StartListening` is called and the driver is already in state listening (i.e. in state I2C_SEQ_PENDING) on the same `I2cChannel`, the I2C Driver shall return `E_NOT_OK`.⌋

[CP_SWS_I2C_80806] Message received

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015, SRS_SPAL_00157

⌈In case the received message is a read message, the I2C driver shall copy the data into the corresponding `RxDataBufferPtr`. In case of write message the I2C driver shall copy the data from the `TxDataBufferPtr` into the response message.⌋

[CP_SWS_I2C_80803] Last Message or any error received

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015, SRS_SPAL_00157

⌈If the last message of the sequence or any error is received after setting the driver into listening mode (i.e. call of I2C_StartListening), I2C driver shall set the Sequence state to

- I2C_SEQ_FAILED, if any failure occured
- I2C_SEQ_NACK, if a NACK message was recived
- I2C_SEQ_OK, if everything was successfully recived.

and afterwards invoke the sequence notification callback function I2C_SeqEndNotification.⌋

[CP_SWS_I2C_82806] Development Error Detection

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015

⌈If development error detection is enabled (I2cDevErrorDetect == True), the function I2C_StartListening shall raise the development error I2C_E_PARAM_SEQUENCE if the function I2C_SetupEB is a 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_80804] Development Error Detection

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015, CP_RS_I2C_00017

⌈If development error detection is enabled (I2cDevErrorDetect == True), the function I2C_StartListening shall raise the development error I2C_E_WRONG_CONDITION if I2cHwUnitMode is NOT set to I2C_HW_UNIT_MODE_TARGET.⌋

[CP_SWS_I2C_80805] I2C_E_WRONG_MODE

Status: DRAFT
Upstream requirements: CP_RS_I2C_00015, CP_RS_I2C_00017

⌈The function I2C_StartListening shall report the runtime error I2C_E_WRONG_MODE if I2cTargetListening is NOT set.⌋

8.4 Callback notifications

This chapter lists all functions provided by the I2C module to lower layer modules. The I2C Driver module belongs to the lowest layer of AUTOSAR Software Architecture hence this module specification has not identified any callback functions.

8.5 Scheduled functions

8.5.1 I2C_MainFunction

[CP_SWS_I2C_00834] Definition of scheduled function I2C_MainFunction

Status: DRAFT

⌈

Service Name	I2C_MainFunction (draft)

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. **Tags:** atp.Status=draft
Available via	I2c.h

⌋

[CP_SWS_I2C_80901] Permanent listening (`Target` Mode)

Status: DRAFT

⌈If `I2cTargetListening` is set, the I2C driver shall listen permanently for new message after the function `I2C_SetupEB` is called. For each received message the sequence notification callback function `I2C_SeqEndNotification` shall be invoked.⌋

8.6 Expected interfaces

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

8.6.1 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

Status: DRAFT

⌈

API Function	Header File	Description
There are no mandatory interfaces.		

⌋

8.6.2 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

Status: DRAFT

⌈

API Function	Header File	Description
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

⌋

8.6.3 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.

8.6.3.1 I2C_SeqEndNotification

[CP_SWS_I2C_00832] Definition of configurable interface (*I2C_SeqEndNotification)

Status: DRAFT
Upstream requirements: CP_RS_I2C_00007

⌈

Service Name	(*I2C_SeqEndNotification) (draft)	
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. **Tags:** atp.Status=draft	
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 Dependant". In the beginning the module is initialized and at the end again de-initialized.

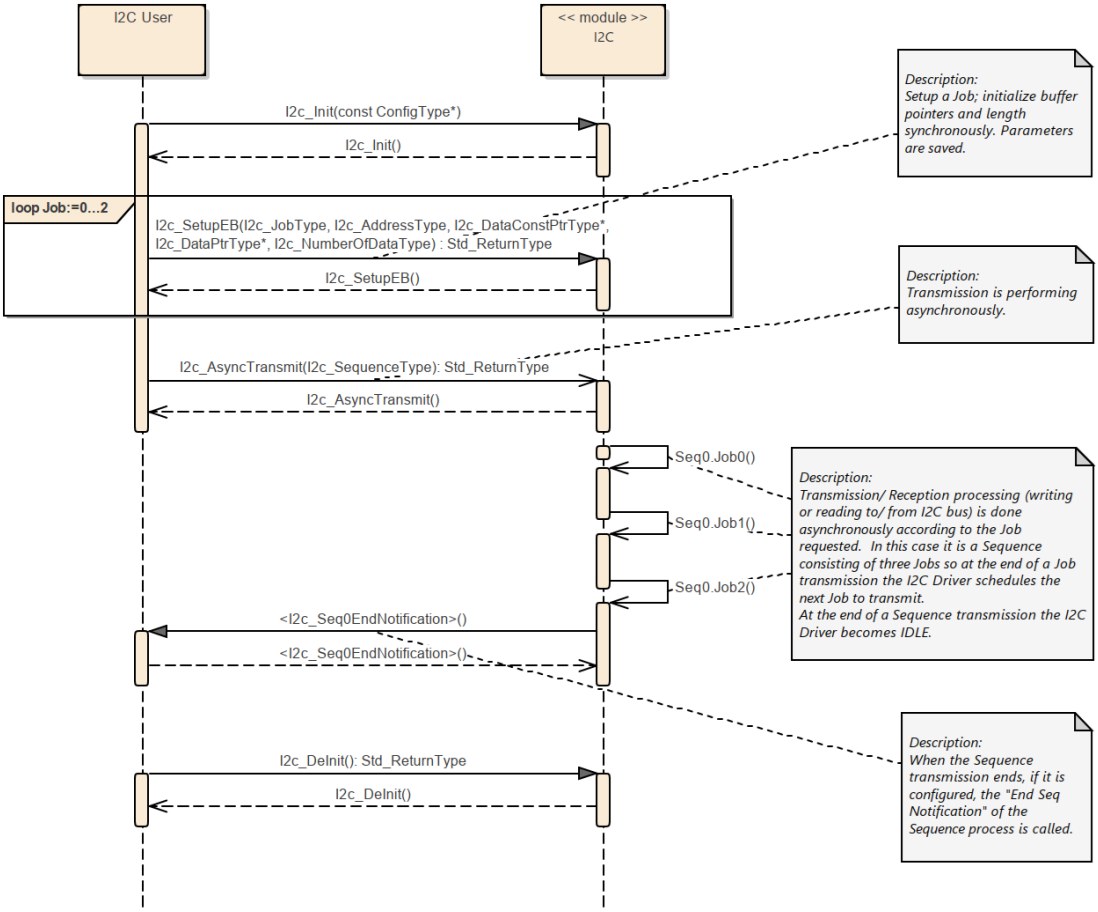

Figure 9.1: SetupEB/ AsyncTransmit

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 Dependant". In the beginning the module is initialized and at the end again de-initialized.

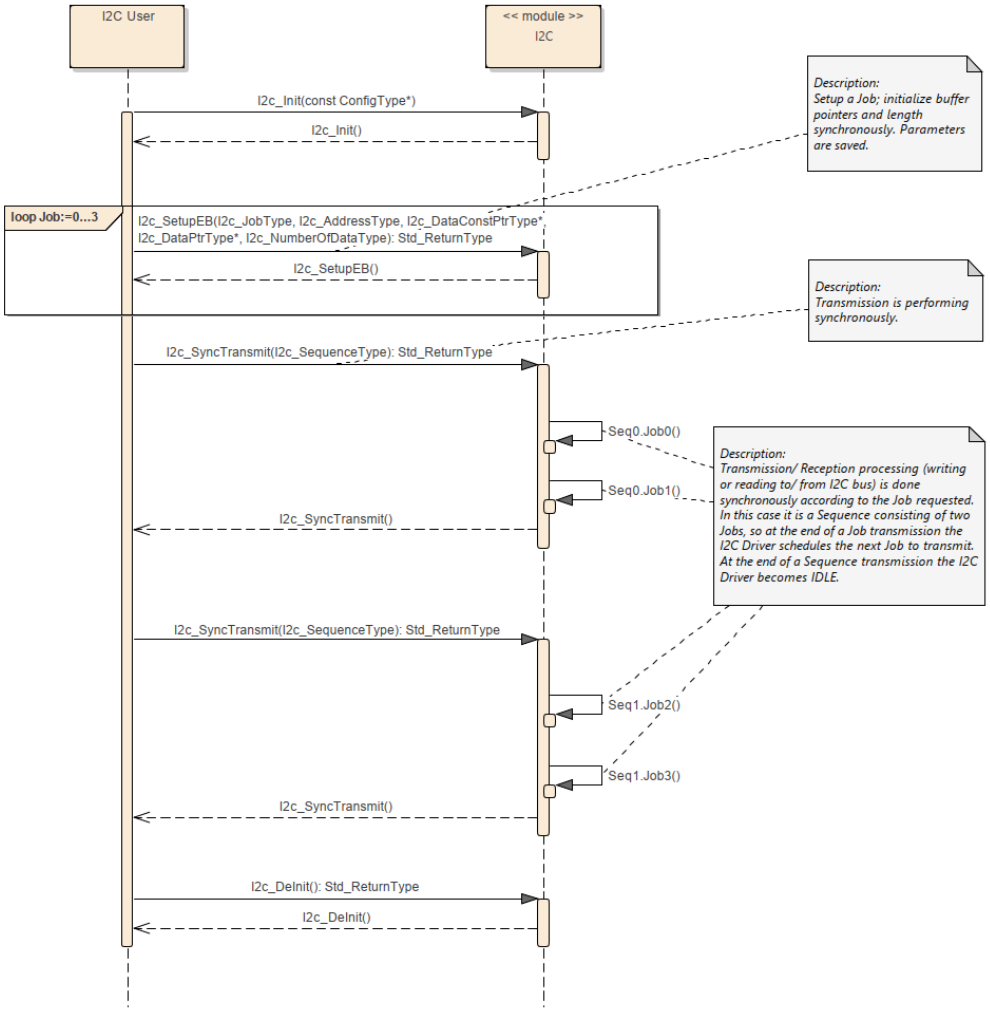

Figure 9.2: SetupEB/ SyncTransmit

9.3 SetupEB/ StartListening

The following sequence diagram shows an example of the process of `I2C_SetupEB`/`I2C_StartListening` calls for Sequence receptions composed of 3 Jobs each. Write or Read accesses are "User Dependant". In the beginning the module is initialized and at the end again de-initialized.

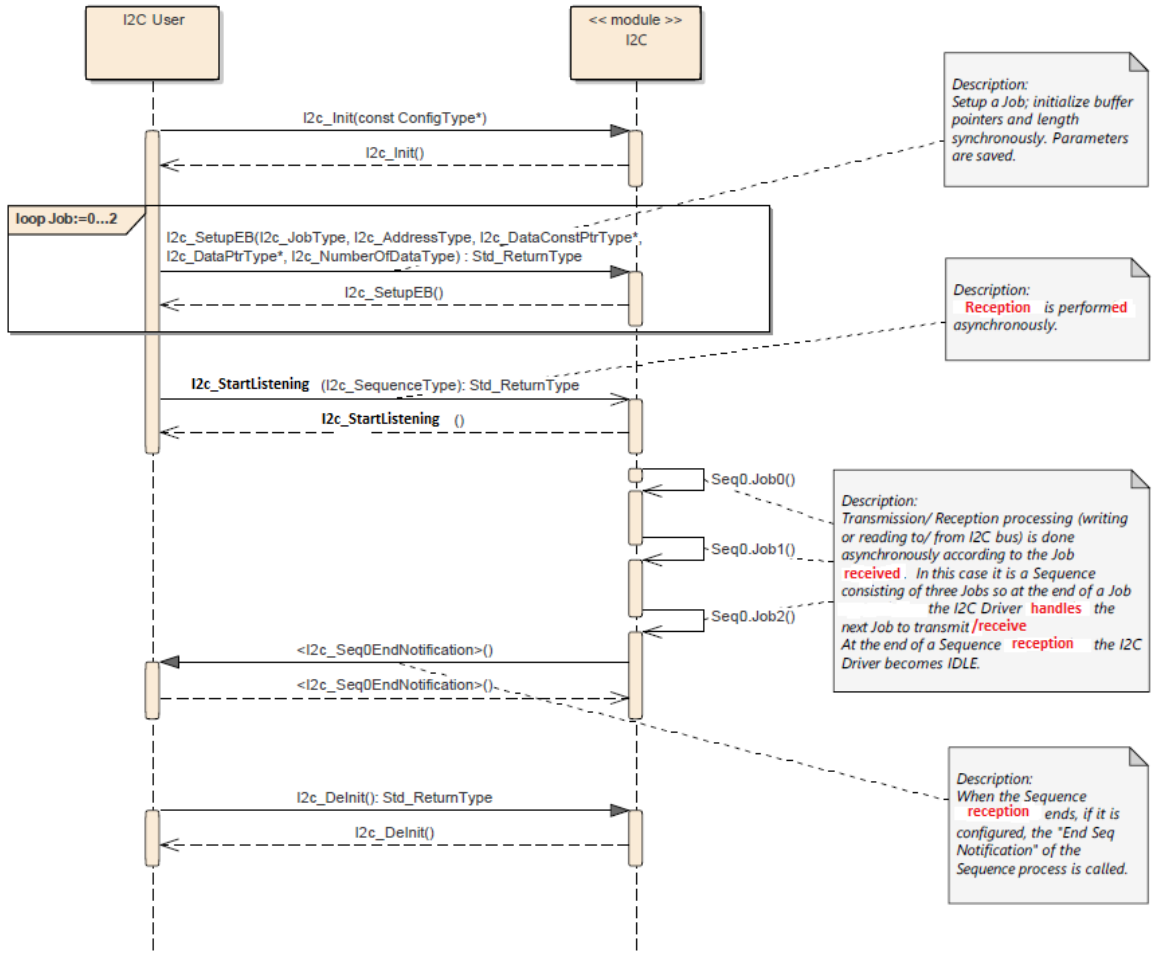

Figure 9.3: SetupEB/ StartListening

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 the chapter 10.1 “Introduction to configuration specification” in [2].

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe 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

Status: DRAFT

⌈

Module Name	I2c
Description	Configuration of the I2c (Inter-Integrated Circuit) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
I2cConfigSet	1..*	The base for a multiple configuration set. **Tags:** atp.Status=draft
I2cGeneral	1	General configuration parameters of the I2c. **Tags:** atp.Status=draft

⌋

10.2.2 I2CGeneral

[ECUC_I2c_00002] Definition of EcucParamConfContainerDef I2cGeneral

Status: DRAFT

⌈

Container Name	I2cGeneral
Parent Container	I2c
Description	General configuration parameters of the I2c. **Tags:** atp.Status=draft
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
I2cDevErrorDetect	1	[ECUC_I2c_00004]
I2cVersionInfoApi	1	[ECUC_I2c_00005]

No Included Containers

⌋

[ECUC_I2c_00004] Definition of EcucBooleanParamDef I2cDevErrorDetect

Status: DRAFT

⌈

Parameter Name	I2cDevErrorDetect		
Parent Container	I2cGeneral		
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. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00005] Definition of EcucBooleanParamDef I2cVersionInfoApi

Status: DRAFT

⌈

Parameter Name	I2cVersionInfoApi		
Parent Container	I2cGeneral		
Description	This parameter enables/disables the function I2c_GetVersionInfo() to get major, minor and patch version information. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

10.2.3 I2cConfigSet

[ECUC_I2c_00003] Definition of EcucParamConfContainerDef I2cConfigSet

Status: DRAFT

⌈

Container Name	I2cConfigSet		
Parent Container	I2c		
Description	The base for a multiple configuration set. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
I2cChannel	1..*	The Hardware specific configuration parameters of the I2c. **Tags:** atp.Status=draft
I2cJob	1..*	Job specific configuration parameters of the I2c. **Tags:** atp.Status=draft
I2cSequence	1..*	Sequence specific configuration parameters of the I2c. **Tags:** atp.Status=draft

⌋

10.2.4 I2cChannel

[ECUC_I2c_00008] Definition of EcucParamConfContainerDef I2cChannel

Status: DRAFT

⌈

Container Name	I2cChannel		
Parent Container	I2cConfigSet		
Description	The Hardware specific configuration parameters of the I2c. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
I2cBaudRate	1	[ECUC_I2c_00014]
I2cHwUnitBaseAddress	1	[ECUC_I2c_00015]
I2cHwUnitMode	1	[ECUC_I2c_00016]
I2cTargetListening	1	[ECUC_I2c_00019]

No Included Containers

⌋

[ECUC_I2c_00014] Definition of EcucIntegerParamDef I2cBaudRate

Status: DRAFT

⌈

Parameter Name	I2cBaudRate		
Parent Container	I2cChannel		
Description	The baud rate of the bus in kbit/s. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	100		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00015] Definition of EcucIntegerParamDef I2cHwUnitBaseAddress

Status: DRAFT

⌈

Parameter Name	I2cHwUnitBaseAddress		
Parent Container	I2cChannel		
Description	The register address of the HW unit. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00016] Definition of EcucEnumerationParamDef I2cHwUnitMode

Status: DRAFT

⌈

Parameter Name	I2cHwUnitMode		
Parent Container	I2cChannel		
Description	Select whether the HW unit will be used in Controller or Target mode. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	I2C_HW_UNIT_MODE_CONTROLLER	HW unit will be used in Controller mode **Tags:** atp.Status=draft	
	I2C_HW_UNIT_MODE_TARGET	HW unit will be used in Target mode **Tags:** atp.Status=draft	
Default value	I2C_HW_UNIT_MODE_CONTROLLER		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00019] Definition of EcucBooleanParamDef I2cTargetListening

Status: DRAFT

⌈

Parameter Name	I2cTargetListening		
Parent Container	I2cChannel		
Description	This parameter specifies the Target mode, always listening or on demand only. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.5 I2cJob

[ECUC_I2c_00006] Definition of EcucParamConfContainerDef I2cJob

Status: DRAFT

⌈

Container Name	I2cJob		
Parent Container	I2cConfigSet		
Description	Job specific configuration parameters of the I2c. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
I2cDeviceAddress	1	[ECUC_I2c_00018]
I2cJobId	1	[ECUC_I2c_00017]

No Included Containers

⌋

[ECUC_I2c_00018] Definition of EcucIntegerParamDef I2cDeviceAddress

Status: DRAFT

⌈

Parameter Name	I2cDeviceAddress		
Parent Container	I2cJob		
Description	The address of a Target device which is accessed by the Controller. Values bigger than `0x7F (127)` lead to Extended Addressing. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1023		
Default value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00017] Definition of EcucIntegerParamDef I2cJobId

Status: DRAFT

⌈

Parameter Name	I2cJobId		
Parent Container	I2cJob		
Description	The identifier of a Job. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

10.2.6 I2cSequence

[ECUC_I2c_00007] Definition of EcucParamConfContainerDef I2cSequence

Status: DRAFT

⌈

Container Name	I2cSequence		
Parent Container	I2cConfigSet		
Description	Sequence specific configuration parameters of the I2c. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
I2cEndNotification	1	[ECUC_I2c_00009]
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

Status: DRAFT

⌈

Parameter Name	I2cEndNotification		
Parent Container	I2cSequence		
Description	The transmission end notification to inform the user that a transmission request has been serviced. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00011] Definition of EcucIntegerParamDef I2cSequenceId

Status: DRAFT

⌈

Parameter Name	I2cSequenceId		
Parent Container	I2cSequence		
Description	The Id of a I2c Sequence. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 254		
Default value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00012] Definition of EcucReferenceDef I2cAssignedChannel

Status: DRAFT

⌈

Parameter Name	I2cAssignedChannel		
Parent Container	I2cSequence		
Description	References the bus which is assigned to the Job. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to I2cChannel		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

[ECUC_I2c_00010] Definition of EcucReferenceDef I2cAssignedJob

Status: DRAFT

⌈

Parameter Name	I2cAssignedJob		
Parent Container	I2cSequence		
Description	Reference to a Job. **Tags:** atp.Status=draft **Attributes:** requiresIndex=true		
Multiplicity	1..*		
Type	Reference to I2cJob		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

⌋

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in [2].

A Not applicable requirements

[CP_SWS_I2C_NA_00999]

Status:	DRAFT
Upstream requirements:	SRS_SPAL_12267, SRS_SPAL_12463, SRS_SPAL_12068, SRS_SPAL_12069, SRS_SPAL_12169, SRS_SPAL_12064, SRS_SPAL_12067, SRS_SPAL_12077, SRS_SPAL_12078, SRS_SPAL_12092, SRS_SPAL_12265

⌈These requirements are not applicable to this specification.⌋

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

Number	Heading
[CP_SWS_I2C_00101]	Development Error Detection
[CP_SWS_I2C_00102]	Buffer Pointer Setup EB - `Controller` mode
[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]	Definiton of development errors in module I2C
[CP_SWS_I2C_00701]	Definiton of runtime errors in module I2C
[CP_SWS_I2C_00702]	Error `I2C_E_FIFO_HANDLING`
[CP_SWS_I2C_00703]	Error `I2C_E_NACK_RECEIVED`
[CP_SWS_I2C_00704]	`I2C_E_ARBITRATION_FAILURE`
[CP_SWS_I2C_00705]	`I2C_E_BUS_FAILURE`
[CP_SWS_I2C_00801]	Definition of datatype I2C_ConfigType
[CP_SWS_I2C_00803]	Definition of datatype I2C_AddressType
[CP_SWS_I2C_00804]	Definition of datatype I2C_DataType
[CP_SWS_I2C_00805]	Definition of datatype I2C_DataPtrType

▽

△

Number	Heading
[CP_SWS_I2C_-00806]	Definition of datatype I2C_DataConstPtrType
[CP_SWS_I2C_-00807]	Definition of datatype I2C_SequenceResultType
[CP_SWS_I2C_-00808]	Definition of datatype I2C_HwUnitType
[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_-00820]	Definition of API function I2C_Init
[CP_SWS_I2C_-00821]	Definition of API function I2C_DeInit
[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_-00827]	Definition of API function I2C_GetVersionInfo
[CP_SWS_I2C_-00828]	Definition of API function I2C_GetSequenceResult
[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_-00832]	Definition of configurable interface (*I2C_SeqEndNotification)
[CP_SWS_I2C_-00833]	Definition of imported datatypes of module I2C
[CP_SWS_I2C_-00834]	Definition of scheduled function I2C_MainFunction
[CP_SWS_I2C_-00835]	Definition of API function I2C_StartListening
[CP_SWS_I2C_-80701]	Sequence Result
[CP_SWS_I2C_-80702]	Development Error Detection
[CP_SWS_I2C_-80801]	No listening is ongoing

▽

△

Number	Heading
[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_-82002]	Initialzation
[CP_SWS_I2C_-82105]	Deinitialization
[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_-82305]	The same 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
[CP_SWS_I2C_-82601]	Development Error Detection
[CP_SWS_I2C_-82806]	Development Error Detection
[ECUC_I2c_00001]	Definition of EcucModuleDef I2c
[ECUC_I2c_00002]	Definition of EcucParamConfContainerDef I2cGeneral

▽

△

Number	Heading
[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
[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.1: Added Specification Items in R24-11

B.1.2 Changed Specification Items in R24-11

none

B.1.3 Deleted Specification Items in R24-11

none