SCOPE OF APPLICATION All Project/Engineering	HYUNDAI AUTRON	SHT/SHTS 1 / 51
Responsibility: Body & Chassis R&D Center	AUTRON Standard Process	Doc. No. CSSWP150
TITLE : Safety Manual for E2E		

Document Change History				
Date (YYYY-MM-DD)	Ver.	Editor	Chap	Description(Change before -> after)
2016-12-23	1.1.4	Seongmin Kim	All	• Initial Creation
				•
				•
				•

1st Edition Date : 2016-12-23	File Name: E2E_Safety_Manual.docx	Creation Seongmin Kim 2016-12-23	Check JongHyun Baek 2016-12-23	Approval Sangil Lee 2016-12-23
Document Management System SVN				

TABLE OF CONTENTS

LIST OF TABLES

LIST OF Figures

1. Introduction

1.1. Purpose

This document is a safety manual with respect to the AUTOSAR E2E module in an AUTOSAR Platform. The purpose of this document is to specify the necessary information related to the AUTOSAR E2E module which is provided to an integrator based on ISO 26262-10 [N5], A.3.10, in order to enable the integration of the AUTOSAR E2E module into the AUTOSAR platform later on.

1.2. Scope

This document is only limited to the safety manual of the AUTOSAR E2E module developed by Hyundai Autron Co., Ltd. (hereinafter called the "Autron") as a SEooC.

1.3. References

1.3.1. Applicable Standards

Table 1.1 below shows the list of standards applicable to this document.

Table 1.1 List of applicable standards

Ref. No.	Title	Published Year
[N1]	ISO 26262-4, Road vehicles – Functional safety – Part 4: Product development at the system level	2011
[N2]	ISO 26262-6, Road vehicles – Functional safety – Part 6: Product development at the software level	2011
[N3]	ISO 26262-8, Road vehicles – Functional safety – Part 8: Supporting processes	2011
[N4]	ISO 26262-9, Road vehicles – Functional safety – Part 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analysis	2011
[N5]	ISO 26262-10, Road vehicles – Functional safety – Part 10: Guideline on ISO 26262	2012

1.3.2. Input Documents

Table 1.2 below shows the list of input documents referred to this document.

Table 1.2 List of input documents

Ref. No.	Doc. No.	Document Name	Ver. No.

Ref. No.	Doc. No.	Document Name	Ver. No.
[D1]	CSSWP109	Software Requirements Specification for E2E (E2E_SRS.docx)	1.1.4
[D2]	CSSWP111	Verification review of the Software Requirement_Check-list&Result for E2E (E2E_SRS_VR.xlsx)	1.1.4
[D3]	CSSWP114	Software Architectural Design Specification for E2E (E2E_SAD.docx)	1.1.4
[D4]	CSSWP123	Verification review of the Software Architecture_Check-list&Result for E2E (E2E_SAD_VR.xlsx)	1.1.4
[D5]	CSSWP117	Safety Analysis Report for E2E (E2E_FMEA_SAR.xls)	1.1.4
[D6]	CSSWP120	Dependent Failures Analysis Report for E2E (E2E_DFAR.docx)	1.1.4
[D7]	CSSWP172	Verification review of the Software Safety Analysis_Check-list&Result for E2E (E2E_SSA_VR.xlsx)	1.1.4
[D8]	CSSWP126	Software Unit Design Specification for E2E (E2E_SUD.docx)	1.1.4
[D9]	CSSWP129	Verification review of the Software Unit Design_Check-list&Result for E2E (E2E_SUD_VR.xlsx)	1.1.4
[D10]	CSSWP132	Software Unit Test Specification for E2E (E2E_SUTS.xlsx)	1.1.4
[D11]	CSSWP135	Software Unit Test Report for E2E (E2E_SUTR.xlsx)	1.1.4
[D12]	CSSWP138	Software Integration Test Specification for E2E (E2E_SITS.xlsx)	1.1.4
[D13]	CSSWP141	Software Integration Test Report for E2E (AUTRON_AUTOSAR_E2E_ESTS.doc)	1.1.4
[D14]	CSSWP144	Software Requirement Test Specification for E2E (E2E_SRTS.xlsx)	1.1.4
[D15]	CSSWP177	Inspection review of the Software Requirement Test_Check-list&Result for E2E (E2E_SRTS_IR.xlsx)	1.1.4

Ref. No.	Doc. No.	Document Name	Ver. No.
[D16]	CSSWP147	Software Requirement Test Report for E2E (E2E_SRTR.xlsx)	1.1.4
[D17]	CSSWP175	AUTRON_AUTOSAR_E2E_Traceability (AUTRON_AUTOSAR_E2E_Traceability.xls)	1.1.4
[D18]	N/A	AUTOSAR Technical Safety Concept Status Report (AUTOSAR_TR_SafetyConceptStatusReport.pdf)	1.1.0 (AUTOSAR 4.0.2)
[D19]	N/A	Specification of SW-C End-to-End Communication Protection Library (AUTOSAR_SWS_E2ELibrary.pdf)	2.0.0 (AUTOSAR 4.0.3)
[D20]	N/A	Specification of CRC Routines (AUTOSAR_SWS_CRCLibrary.pdf)	5.0.0 (AUTOSAR 4.0.3)

1.4. Abbreviations, Acronyms and Terms

1.4.1. Definition of Abbreviations and Acronyms

Table 1.3 below shows the definition of abbreviations and acronyms used in this document.

Table 1.3 Definition of abbreviations and acronyms

Abbreviation / Acronym	Definition
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
AUTOSAR	AUTomotive Open System ARchitecture
BSW	Basic SoftWare
CRC	Cyclic Redundancy Check
ECU	Electronic Control Unit
E2E	End-to-End
MCAL	Microcontroller Abstraction Layer
MCU	MicroController Unit
NA	Not Applicable
RAM	Random Access Memory
ROM	Read Only Memory
RTE	Runtime Environment

Abbreviation / Acronym	Definition
SRS	Software Requirements Specification
SWS	SoftWare Specification
SEooC	Safety Element out of Context

1.4.2. Description of Terms

Table 1.4 below shows the description of terms used in this document.

Table 1.4 Description of terms

Term	Description
Integrator	The person, department or organization that integrates software is called the integrator.
P2CONST	Pointer to a Constant
P2VAR	Pointer to a Variable
µC	Micro-controller

1.4.3. ID Notation of Integration Requirement

The unique identifications of integration requirements, which are required to enable the integration of the AUTOSAR E2E module into the AUTOSAR platform later on, are indicated as '[E2E_IR_#xx], x = integer in this document. The list of the whole integration requirements specified in this document is shown in Annex A.

2. Safety Lifecycle

Based on the safety lifecycle specified in ISO 26262-2, Clause 5.2.1 and guidelines in ISO 26262-10, Clause 9.2.4, the AUTOSAR E2E module has been developed as a SEooC, in compliance with ISO 26262-2, ISO 26262-6 ISO 26262-8 (partially) and ISO 26262-9 (partially). Figure 2.1 below shows the ISO 26262 safety lifiecycle.

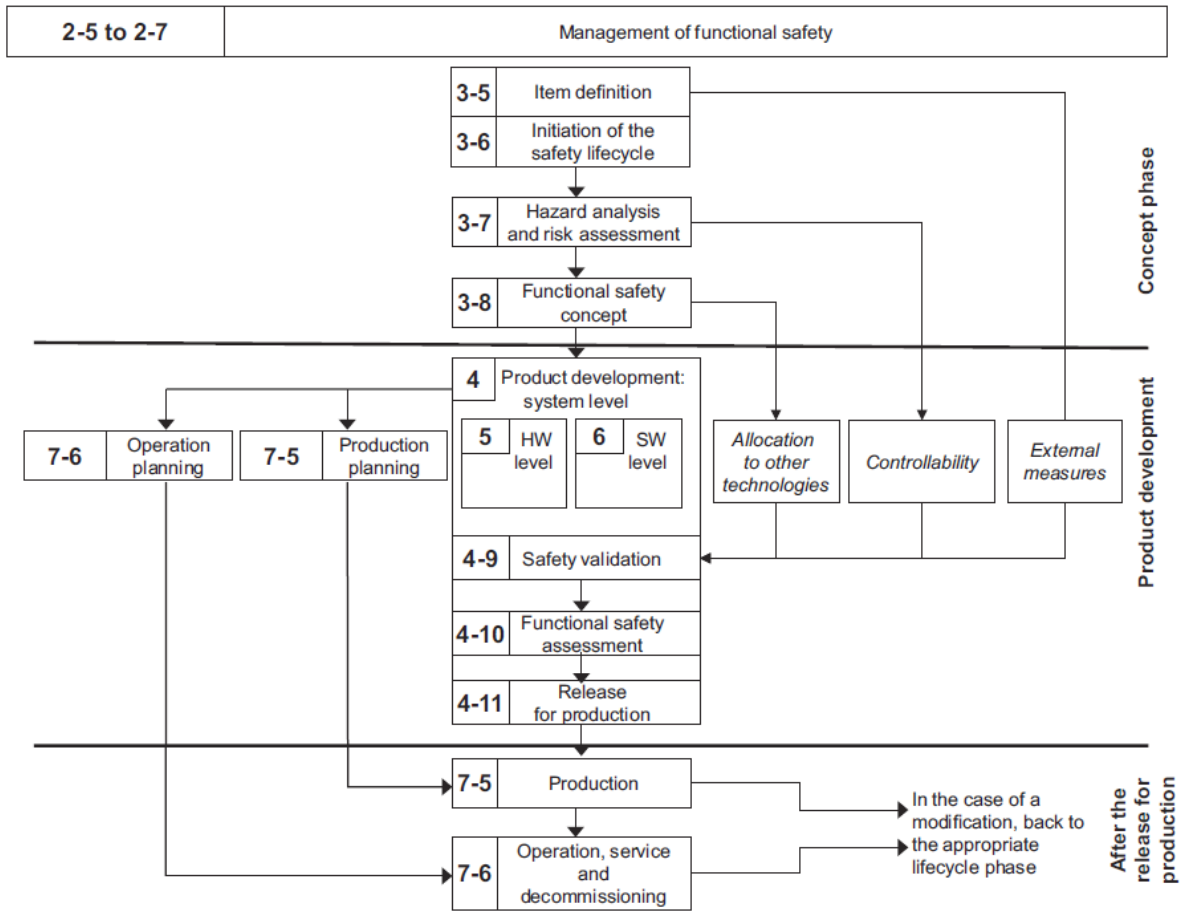

Figure 2.1 ISO 26262 Safety Lifecycle

2.1. Tailored Safety Lifecycle

Autron has tailored the safety lifecycle of ISO 26262:2011 to match the needs of a safety element out of context (SEooC). The tailoring activity is performed to ensure developing the AUTOSAR E2E module to meet the ISO 26262 requirements and supplying it to the customer. The AUTOSAR E2E module covered by this project is software components developed as safety elements out of context. So, Autron's safety activities are mainly focused on management of functional safety and software component development. Detail Elements of the tailored safety lifecycle are described Table 2.1 below.

Table 2.1 Tailoring result of Safety life cycle

Sub-process	Activity	O/X	Tailoring reason	Responsibility

Management of functional safety	Overall safety management	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Safety management during the concept phase and the product development	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Safety management after the item 's release for production	X	This is not our project scope because we do not consider safety management after the item's release for production.	-
Concept phase	Assumed Item Definition	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Initiation of the safety lifecycle	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Hazard analysis and risk assessment	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Functional safety concept	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-

Development at the system level	Specification of the technical safety requirements	△	It is not our project scope to implement the system design. because it will be performed according to SEooC Process for developing SW component But, for SEooC, we will make hardware-Software Interface specification and consider AUTOSAR_TR_SafetyConceptStatusReport from AUTOSAR for E2E.	Standard Platform Team
	System design	X	It is not our project scope to implement the system design. because it will be performed according to SEooC Process for developing SW component	-
	Item integration and testing	X	It is not our project scope to implement the system design. because it will be performed according to SEooC Process for developing SW component	-
	Safety validation	X	It is not our project scope to implement the system design. because it will be performed according to SEooC Process for developing SW component	-
	Functional safety assessment	X	It is not our project scope to implement the system design. because it will be performed according to SEooC Process for developing SW component	-
	Release for production	X	It is not our project scope to implement the system design. because it will be performed according to SEooC Process for developing SW component	-
Development at the software level	Specification of software safety requirements	O	-	Standard Platform Team
	Software architecture design	O	-	Standard Platform Team

	Software unit design and implementation	O	-	Standard Platform Team
	Software unit testing	O	-	Standard Platform Team
	Software integration and testing	O	-	Standard Platform Team
	Verification of software safety requirements	O	-	Standard Platform Team
Development at the hardware level	Specification of hardware safety requirements	X	It is not our project scope. because it will be performed according to SEooC Process for developing SW component	-
	Hardware design	X	It is not our project scope. because it will be performed according to SEooC Process for developing SW component	-
	Evaluation of the hardware architectural metrics	X	It is not our project scope. because it will be performed according to SEooC Process for developing SW component	-
	Evaluation of the safety goal violations due to random hardware failures	X	It is not our project scope. because it will be performed according to SEooC Process for developing SW component	-
	Hardware integration and testing	X	It is not our project scope. because it will be performed according to SEooC Process for developing SW component	-

Production and operation	Production	X	It is not our project scope. because it will be performed according to SEooC Process for developing SW component	-
	Operation, service and decommissioning	X	It is not our project scope. because it will be performed according to SEooC Process for developing SW component	-
Supporting Process	Interfaces within distributed developments	X	It is not our project scope because all software will be developed without the distributed development.	
	Specification and management of safety requirements	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Configuration management	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Change management	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Verification	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Documentation	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-
	Confidence in the use of software tools	X	It is not our project scope to implement Concept phase. because it will be performed according to SEooC Process for developing SW component	-

	Qualification of software components	X	It is not our project scope. Because our project doesn't use the other software components.	-
	Qualification of hardware components	X	It is not our project scope. Because our project will only consider software development. so, Qualification of hardware componets will not be considered.	-
	Proven in use argument	X	It is not our project scope. Because that we don't consider Proven in use argument.	-
ASIL-oriented and safety-oriented analyses	Requirements decomposition with respect to ASIL tailoring	X	It is not our project scope. Because E2E modules will be followed Technical safety concept report in AUTOSAR.	-
	Criteria for coexistence of elements	X	It is not our project scope. Because E2E modules will be followed Technical safety concept report in AUTOSAR.	-
	Analysis of dependent failures	O	-	Standard Platform Team
	Safety analyses	O	-	Standard Platform Team

2.2. Producted Work Products

Autron has performed activities as tailored the safety lifecycles chapter 2.1. Autron's safety activities are mainly focused on management of functional safety and software component development. Detail Elements of the product are described Table 2.2 below.

Table 2.2 Work Products of each Sub-pahse and Activity

Sub-phase	Activity	Work Product

Development at the software level	Initiation of product development at the software level	1)Tool application guidelines
	Specification of software safety requirements	1) Software Requirement Specification for E2E [D1] 2) Verification review of the Software Requirement_Check-list&Result for E2E [D2]
	Software architecture design	1) Software Architectural Design Specification for E2E [D3] 2) Verification review of the Software Architecture_Check-list&Result for E2E [D4]
	Software unit design and implementation	1) Software Unit Design Specification for E2E [D8] 2) Verification review of the Software Unit Design_Check-list&Result for E2E [D9]
	Software unit testing	1) Software Unit Test Specification for E2E [D10] 2) Software Unit Test Report for E2E [D11]
	Software integration and testing	1) Software Integration Test Specification for E2E [D12] 2) Software Integration Test Report for E2E [D13]
	Verification of software safety requirements	1) Inspection review of the Software Requirement Test_Check-list&Result for E2E [D15] 2) Software Requirement Test Report for E2E [D16]
ASIL-oriented and safety-oriented analyses	Analysis of dependent failures	Dependent Failure Analysis for E2E [D6]
	Safety analyses	Safety Analysis Report for E2E [D8] Verification review of the Software Safety Analysis_Check-list&Result for E2E [D7]

3. Assumptions of Use

This chapter describes assumptions of use of the AUTOSAR E2E module with respect to its intended use, including main functions, safety requirements and safe states with respect to a failure which could lead to a violation of a safety requirement, external interfaces and so on.

3.1. Main Safety Functions

It was assumed that the AUTOSAR E2E module carries out 2 main safety functions in order to be developed as a SEooC on the basis of the AUSTOSAR Specification. Table 3.1 below shows the main safety functions carried out by the AUTOSAR E2E module, including the ASIL in accordance with ISO 26262 Standard requirements.

Table 3.1 Main Safety Functions of the AUTOSAR E2E module

No.	Main safety function	Functional description	Determined ASIL
1	SW-C end-to-end communication protection	The AUTOSAR E2E module provides mechanisms of end-to-end protection in the communication between SW-Cs. At sender side, control fields like CRC or counter to the transmitted data are added. At receiver side, the control fields from the received data are evaluated by calculation of control fields (e.g. CRC calculation on the received data) and comparison of calculated control fields with an expected/received content.	ASIL D

Figure 3.1 below shows the basic concept of "SW-C end-to-end communication protection" functionality. The AUTOSAR E2E module on sender side will add control fields at configured position in data. The AUTOSAR E2E module on receiver side will evaluate the control fields by calculation control fields and comparison of calculated control fields with an expected/received content.

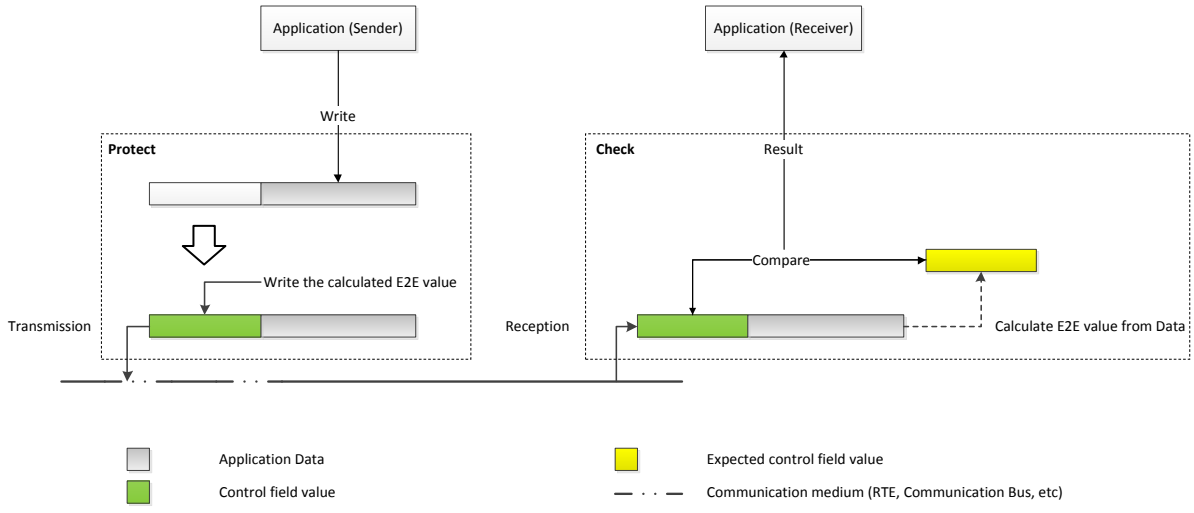

Figure 3.1 AUTOSAR E2E module's end-to-end communication protection

For details of the non-safety functions, please refer to chapter 4 in the "Software Requirement Specification for E2E" [D1].

[E2E_IR_#01] The safety and non-safety functions are assumed by Autron's definition in the Software Requirements Specification for E2E [D1] and analyzed by Dependent Failures Analysis Report for E2E [D6]. An integrator shall analyze that the violation of non-safety requirements does not lead to dangerous situation at the application level.

3.2. Software Requirements

The safety-related software requirements are quoted from AUTOSAR specifications and the safety parts are identified from the published "Safety Features" in AUTOSAR Technical Safety Concept Status Report [D18]. And Autron identifies safety requirements identified from "Safety Features".
Table 3.2 below shows the safety-related software requirements of the AUTOSAR E2E module identified in the "Software Requirement Specification for E2E" [D1]. The ASIL for each safety-related software requirements inherits and is assigned to match the function of each requirement from the determined ASIL of main safety functions in Table 3.2.

Table 3.2 Software safety requirements of the AUTOSAR E2E module

Requirement ID	Requirement	Assigned ASIL	Relevant Main function
E2E-SRS-SFUN-REQ-01	The implementation of the E2E Library shall comply with the requirements for the development of safety-related software for the automotive domain. The ASIL assigned to the requirements implemented by the E2E library depends on the safety concept of a particular system. Depending on that application, the E2E Library at least may need to comply with an ASIL A, B, C or D development process. Therefore it may be most efficient to develop the library according to the highest ASIL, which enables to use the same library for lower ASILs as well.	ASIL D	SW-C end-to-end communication protection

Requirement ID	Requirement	Assigned ASIL	Relevant Main function
E2E-SRS-SFUN-REQ-02	The implementation of the E2E Library shall provide at least one of the E2E Profiles, i.e. E2E Profile 1 or E2E Profile 2.	ASIL D	SW-C end-to-end communication protection
E2E-SRS-SFUN-REQ-03	E2E Profile 1 shall use the polynomial of CRC-8-SAE J1850, i.e. the polynomial 0x1D (x8 + x4 + x3 + x2 + 1), but with start value and XOR value shall be 0x00.	ASIL D	SW-C end-to-end communication protection
E2E-SRS-SFUN-REQ-04	E2E Profile 2 shall use the Crc_CalculateCRC8H2F() function of the SWS CRC Library for calculating CRC checksums.	ASIL D	SW-C end-to-end communication protection

All the AUTOSAR E2E specifications except the published "Safety Features" are defined as non-safety requirement. For details of the non-safety functions, please refer to chapter 4 in the "Software Requirement Specification for E2E" [D1].

3.3. Safe State

In case of a failure which could lead to a violation of a safety requirement, the AUTOSAR E2E module has been designed so that the failure is detected and controlled in order to achieve and maintain a safe state. Table 3.3 below show safe states of the AUTOSAR E2E module to be achieved and retained in case of a failure which could lead to a violation of a safety requirement.

Table 3.3 The Safe State of E2E

Safety Requirement ID	Safe State	Description
E2E-SRS-SFUN-REQ-01	Indicating the detected status to users	The AUTOSAR E2E module indicates the detected status to the user to handle the detected failure modes.
E2E-SRS-SFUN-REQ-02		
E2E-SRS-SFUN-REQ-03		
E2E-SRS-SFUN-REQ-04		

[E2E_IR_#02] Since the AUTOSAR E2E library is a library which is invoked by function call of application software. This means that the safe state of the AUTOSAR E2E module with respect to a

failure which is detected by external watchdog-related modules and External Watchdog depends on the safe state of application software. Therefore, the safe state of application software shall be defined properly by the integrator that integrates the AUTOSAR OS or the whole AUTOSAR Platform, in order to achieve and maintain a safe state on the AUTOSAR OS level or on the whole AUTOSAR Platform level.

3.4. External Interfaces

To realize the safety requirements defined for safety-related main functions, including non-safety requirements defined for the non-safety-related main function, and the safe states mentioned in sections 3.2 and 3.3, the AUTOSAR E2E module has the overall external interfaces as shown in Figure 3.2 below.

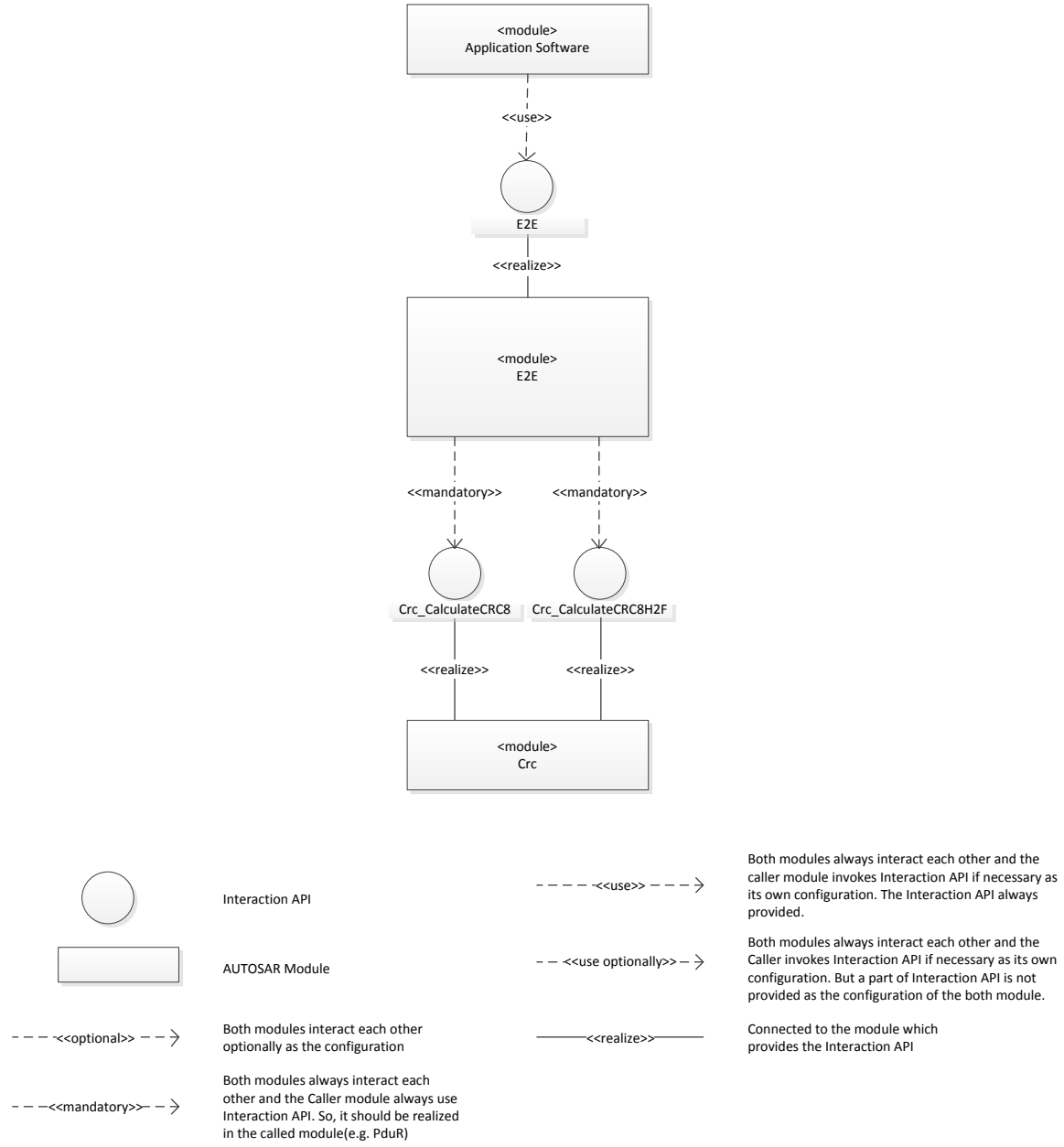

Figure 3.2 Overall external interfaces of the AUTOSAR E2E module

3.4.1. Software Interface

Table 3.4 Module Description of Software Interface

Module name	Module Description
Application Software	The application software uses the capabilities of the AUTOSAR E2E

	module to protect and check data. In AUTOSAR, the application software is the higher layer above the AUTOSAR E2E module.
Crc	Crc module provides functions for CRC calculation

The following E2E APIs are provided for interface of other modules

Table 3.5 Interface Function Description

No	E2E API	Description	Called module
1	Std_ReturnType E2E_P01Protect(E2E_P01ConfigType* Config, E2E_P01SenderStateType* State, uint8* Data)	This service protects the array/buffer to be transmitted using the E2E profile 1. This includes checksum calculation, handling of counter and Data ID.	Application Software
2	Std_ReturnType E2E_P01Check(E2E_P01ConfigType* Config, E2E_P01ReceiverStateType* State, uint8* Data)	This service checks the Data received using the E2E profile 1. This includes CRC calculation, handling of Counter and Data ID.	Application Software
3	Std_ReturnType E2E_P01Protect(E2E_P02ConfigType* Config, E2E_P02SenderStateType* State, uint8* Data)	This service protects the array/buffer to be transmitted using the E2E profile 2. This includes checksum calculation, handling of sequence counter and Data ID.	Application Software
4	Std_ReturnType E2E_P02Check(E2E_P02ConfigType* Config, E2E_P02ReceiverStateType* State, uint8* Data)	This service checks the array/buffer using the E2E profile 2. This includes checksum calculation, handling of sequence counter and Data ID.	Application Software
5	Std_ReturnType E2E_CRC8u8(uint8 Data, uint8 StartValue)	This service is the utility function for computing CRC over uint8 data transmited with E2E Protocol, as in E2E Profile 1.	Application Software
6	Std_ReturnType E2E_CRC8u16(uint16 Data, uint8 StartValue)	This service is the utility function for computing CRC over uint16 data transmited with E2E Protocol, as in E2E Profile 1.	Application Software
7	Std_ReturnType E2E_CRC8u32(uint32 Data,	This service is the utility function for computing CRC over uint32	Application Software

	uint8 StartValue)	data transmited with E2E Protocol, as in E2E Profile 1.	
8	Std_ReturnType E2E_CRC8u8Array(uint16 Length, uint8* Data, uint8 StartValue)	This service is the utility function for computing CRC over an array of uint8 transmited with E2E Protocol, as in E2E Profile 1.	Application Software
9	Std_ReturnType E2E_CRC8u16Array(uint16 Length, uint16* Data, uint8 StartValue)	This service is the utility function for computing CRC over an array of uint16 transmited with E2E Protocol, as in E2E Profile 1.	Application Software
10	Std_ReturnType E2E_CRC8u32Array(uint16 Length, uint32* Data, uint8 StartValue)	This service is the utility function for computing CRC over an array of uint32 transmited with E2E Protocol, as in E2E Profile 1.	Application Software
11	Std_ReturnType E2E_CRC8H2Fu8(uint8 Data, uint8 StartValue)	This service is the utility function for computing CRC over uint8 data transmited with E2E Protocol, as in E2E Profile 2.	Application Software
12	Std_ReturnType E2E_CRC8H2Fu16(uint16 Data, uint8 StartValue)	This service is the utility function for computing CRC over uint16 data transmited with E2E Protocol, as in E2E Profile 2.	Application Software
13	Std_ReturnType E2E_CRC8H2Fu32(uint32 Data, uint8 StartValue)	This service is the utility function for computing CRC over uint32 data transmited with E2E Protocol, as in E2E Profile 2.	Application Software
14	Std_ReturnType E2E_CRC8H2Fu8Array(uint16 Length, uint8* Data, uint8 StartValue)	This service is the utility function for computing CRC over an array of uint8 transmited with E2E Protocol, as in E2E Profile 2.	Application Software
15	Std_ReturnType E2E_CRC8uH2F16Array(uint16 Length, uint16* Data, uint8 StartValue)	This service is the utility function for computing CRC over an array of uint16 transmited with E2E Protocol, as in E2E Profile 2.	Application Software
16	Std_ReturnType E2E_CRC8H2Fu32Array(	This service is the utility function	Application

	uint16 Length, uint32* Data, uint8 StartValue)	for computing CRC over an array of uint32 transmited with E2E Protocol, as in E2E Profile 2.	Software
17	uint8 E2E_UpdateCounter(uint8 Counter)	This service increments the counter provided by the parameter, and returns it by return value.	Application Software
18	void E2E_GetVersionInfo(Std_VersionInfoType* VersionInfo)	This service returns the version information of this module.	Application Software

3.4.2. Hardware Interface

The E2E module is separate from hardware interface. The related lower module will interface with hardware through MCAL layer.

3.5. Configuration

E2E Library, like all AUTOSAR libraries, has no configuration options. All the information needed for execution of Library functions is passed at runtime by function parameters. For the functions E2E_PXXProtect() and E2E_PXXCheck(), one of the parameters is Config, which contains the options for the protection of Data.

3.6. Memory and Time Constraints of the AUTOSAR E2E module

3.6.1. Memory Constraint

The ROM area is needed at least 1,038 Bytes available space for the AUTOSAR E2E module. The AUTOSAR E2E module does not need any RAM areas.

3.6.2. Time Constraint

The AUTOSAR E2E module has no time constraints. Carried out performance is influenced by the MCU. However, the related time constraint in technical point of view as explained below should be considered when integrating is conducted.

4. Software Architecture

This chapter describes the overall software structure and software hierarchy, the results of software safety analysis and dependent failure analysis carried out at the software architectural level and the safety architecture designed on the basis of the results of these analyses with respect to the AUTOSAR E2E module.

4.1. Overall Software Structure

In AUTOSAR, libraries are a collection of functions for related purposes. Libraries have the characteristics as follows:

- Can be called by BSW modules (including the RTE), SW-Cs, libraries or integration code.
- Run in the context of the caller in the same protection environment
- Can only call libraries
- Are re-entrant
- Do not have internal states
- Do not require any initialization
- Are synchronous, i.e. they do not have wait points.

The AUTOSAR E2E module is a library in AUTOSAR, which provides functions to protect and check the safety-related data. Application software uses the AUTOSAR E2E module by function call in its context in its protection environment and APIs provided by the AUTOSAR E2E module are re-entrant. This means that the APIs called by the application software are parts of application software in the view of "component" when application software uses the AUTOSAR E2E module.

The AUTOSAR E2E module has consisted of 5 software components at the highest level in the software hierarchy (see chapter 4.2) to realize the software requirements (see chapter 3.2) identified based on the main safety functions (see chapter 3.1). The ASIL assigned to each software component is based on the highest ASIL of ASILs assigned to the corresponding software requirements. Table 4.1 below shows these software components, including their ASIL and corresponding software requirement IDs and main functions.

Table 4.1 Software components at the highest level and their ASIL assigned

Highest Software component ID	Sub software component ID	Assigned ASIL	Relevant Software requirement ID	Relevant Main function
E2E Profile 1 (E2E-SA-01)	E2E Profile 1 Protect (E2E-SA-01-01) E2E Profile 1 Check (E2E-SA-01-02) E2E Profile 1 CRC Calculation (E2E-SA-01-03)	ASIL D	E2E-SRS-SFUN-REQ-01	SW-C end-to-end communication protection
			E2E-SRS-SFUN-REQ-02	
			E2E-SRS-SFUN-REQ-03	
E2E Profile 2 (E2E-SA-02)	E2E Profile 2 Protect (E2E-SA-02-01) E2E Profile 2 Check (E2E-SA-02-02)	ASIL D	E2E-SRS-SFUN-REQ-01	SW-C end-to-end communication protection
			E2E-SRS-SFUN-REQ-02	
			E2E-SRS-SFUN-REQ-04	

Table 4.2 below shows the functional descriptions of these highest software components mentioned in Table 4.1 above.

Table 4.2 Functional descriptions of software components at the highest level

Software component ID	Software component	Functional description
E2E-SA-01	E2E Profile 1	The **Profile 1** software component provides functionality to protect (E2E_P01Protect()) and check (E2E_P01Check()) safety-related data based on the polynomial CRC-8-SAE J1850.
E2E-SA-02	E2E Profile 2	The **Profile 2** software component provides functionality to protect (E2E_P02Protect()) and check (E2E_P02Check()) safety-related data based on the polynomial 0x2F.

Table 4.3 below shows the functional descriptions of these software components mentioned in Table 4.1 above.

Table 4.3 Functional descriptions of software components

Software component ID	Software component	Functional description
E2E-SA-01-01	E2E Profile 1 Protect	"E2E Profile 1 Protect" software component writes the Counter and CRC in Data and then it increments the Counter based on Profile 1.
E2E-SA-01-02	E2E Profile 1 Check	"E2E Profile 1 Check" software component checks the Counter and CRC of the received Data and determine the check Status based on Profile 1.
E2E-SA-01-03	E2E Profile 1 CRC Calculation	"E2E Profile 1 CRC calculation" software component computes the CRC over DataID and Data.
E2E-SA-02-01	E2E Profile 2 Protect	"E2E Profile 1 Protect" software component writes the Counter and CRC in Data and then it increments the Counter based on Profile 2.
E2E-SA-02-02	E2E Profile 2 Check	"E2E Profile 2 Check" software component checks the Counter and CRC of the received Data and determine the check Status based on Profile 2.

Figure 4.1 below shows the overall software structure of the AUTOSAR E2E module, considering the software components mentioned in Table 4.2 above including software component ID up to Level 3. The same color means that they are the same level software component in hierarchy.

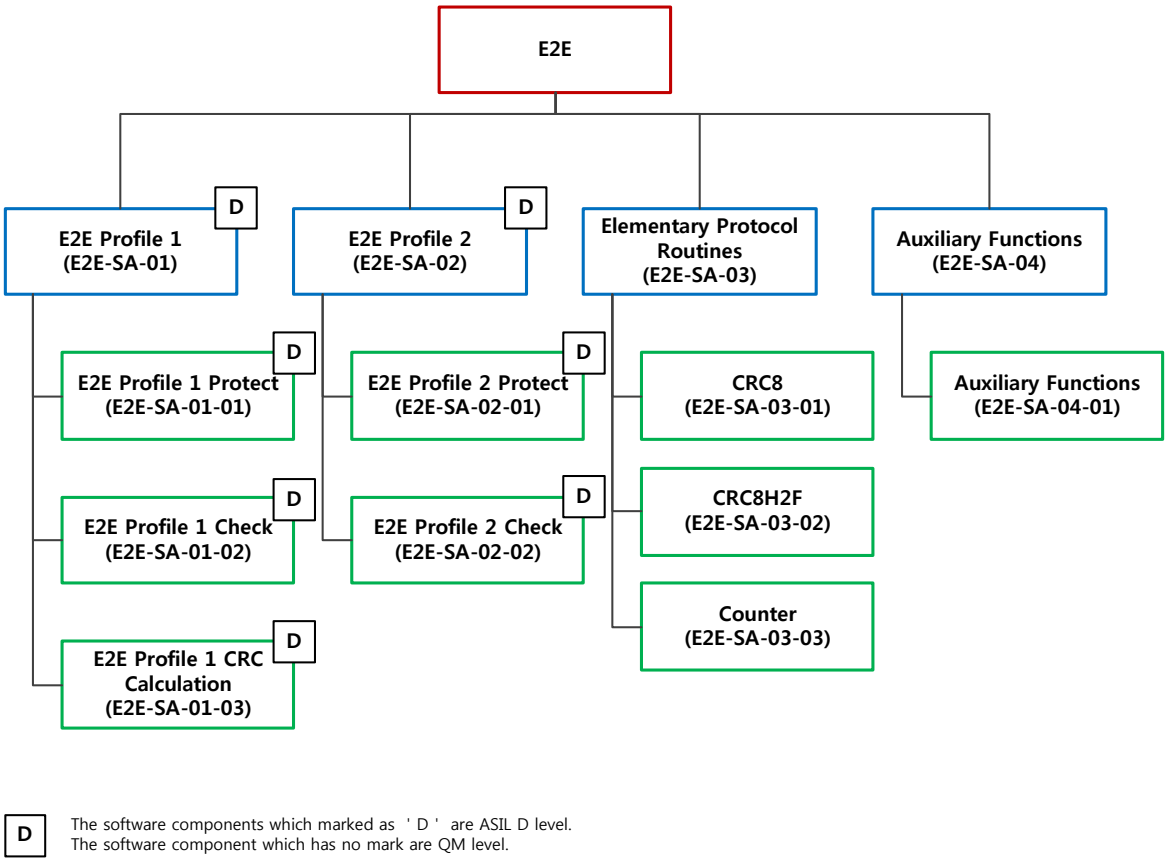

Figure 4.1 Overall software structure of the AUTOSAR E2E module

4.2. Overall Software Hierarchy

The software architectural design of the AUTOSAR E2E module has been developed down to the level where all software units (e.g. E2E Profile 1 Protect (E2E-SA-01-01), E2E Profile 2 Check (E2E-SA-02-02), etc. in Table 4.4) are identified, including the software components at the highest level mentioned in section 4.1, and the software unit design has been developed down to the level where all functions (e.g. E2E_P01Protect (E2E-SUD-01-01-01), E2E_P02Check (E2E-SUD-02-02-01), etc. in Table 4.4) are identified. Table 4.4 below shows the overall software hierarchy of the AUTOSAR E2E module produced during the software architectural design and unit design subphases.

Table 4.4 Software Hierarchy of the AUTOSAR E2E module

Target Software	Level1_SWC	Level2_SWC	Level3_SWC	SWU
E2E	E2E Profile 1 (E2E-SA-01)	E2E Profile 1 Protect (E2E-SA-01-01)		E2E_P01Protect (E2E-SUD-01-01-01)
		E2E Profile 1 Check (E2E-SA-01-02)		E2E_P01Check (E2E-SUD-01-02-01)
				E2E_P01CheckStatus (E2E-SUD-01-02-02)
		E2E Profile 1 CRC Calculation (E2E-SA-01-03)		E2E_P01CalculateCRC (E2E-SUD-01-03-01)
	E2E Profile 2 (E2E-SA-02)	E2E Profile 2 Protect (E2E-SA-02-01)		E2E_P02Protect (E2E-SUD-02-01-01)
		E2E Profile 2 Check (E2E-SA-02-02)		E2E_P02Check (E2E-SUD-02-02-01)
				E2E_P02CheckStatus (E2E-SUD-02-03-01)
	Elementary Protocol Routines (E2E-SA-03)	CRC8 (E2E-SA-03-01)		E2E_CRC8u8 (E2E-SUD-03-01-01)
				E2E_CRC8u16 (E2E-SUD-03-01-02)
				E2E_CRC8u32 (E2E-SUD-03-01-03)
				E2E_CRC8u8Array (E2E-SUD-03-01-04)
				E2E_CRC8u16Array (E2E-SUD-03-01-05)
				E2E_CRC8u32Array (E2E-SUD-03-01-06)
		CRC8H2F (E2E-SA-03-02)		E2E_CRC8H2Fu8 (E2E-SUD-03-02-01)
				E2E_CRC8H2Fu16 (E2E-SUD-03-02-02)
				E2E_CRC8H2Fu32 (E2E-SUD-03-02-03)
				E2E_CRC8H2Fu8Array (E2E-SUD-03-02-04)
				E2E_CRC8H2Fu16Array (E2E-SUD-03-02-05)
				E2E_CRC8H2Fu32Array (E2E-SUD-03-02-06)
		Counter (E2E-SA-03-03)		E2E_UpdateCounter (E2E-SUD-03-03-01)
	Auxiliary Functions (E2E-SA-04)	Auxiliary Functions (E2E-SA-04-01)		E2E_GetVersionInfo (E2E-SUD-04-01-01)

[E2E_IR_#03] The E2E Module separate from H/W interface. The related lower module will interface with H/W through MCAL layer. So, there is no function which interacts with external hardware in the AUTSOAR E2E module.

The below sub chapters describe the functions in AUTOSAR E2E module which interact with external AUTOSAR modules. The Service ID, Sync/Async and Reentrancy in description table are quoted from "Specification of SW-C End-to-End Communication Library" [D19]. And it also contains the information whether the function is safety or non-safety.

4.2.1. E2E Profile 1

Function Name	E2E_P01Protect
Syntax	FUNC(Std_ReturnType, E2E_CODE) E2E_P01Protect (P2VAR(E2E_P01ConfigType, AUTOMATIC, E2E_APPL_DATA) Config, P2VAR(E2E_P01SenderStateType, AUTOMATIC, E2E_APPL_DATA) State, P2VAR(uint8, AUTOMATIC, E2E_APPL_DATA) Data)
Service ID	0x01
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Config
Parameters (Inout)	State, Data
Parameters (Out)	None
Return Value	Std_ReturnType
Description	This service protects the array/buffer to be transmitted using the E2E profile 1. This includes checksum calculation, handling of counter and Data ID.
Safety Function	Yes

Function Name	E2E_P01Check
Syntax	FUNC(Std_ReturnType, E2E_CODE) E2E_P01Check (P2VAR(E2E_P01ConfigType, AUTOMATIC, E2E_APPL_DATA) Config, P2VAR(E2E_P01ReceiverStateType, AUTOMATIC, E2E_APPL_DATA) State, P2VAR(uint8, AUTOMATIC, E2E_APPL_DATA) Data)
Service ID	0x02
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Config, Data

Parameters (Inout)	State
Parameters (Out)	None
Return Value	Std_ReturnType
Description	This service checks the Data received using the E2E profile 1. This includes CRC calculation, handling of Counter and Data ID.
Safety Function	Yes

4.2.2. E2E Profile 2

Function Name	E2E_P02Protect
Syntax	FUNC(Std_ReturnType, E2E_CODE) E2E_P02Protect (P2VAR(E2E_P02ConfigType, AUTOMATIC, E2E_APPL_DATA) Config, P2VAR(E2E_P02SenderStateType, AUTOMATIC, E2E_APPL_DATA) State, P2VAR(uint8, AUTOMATIC, E2E_APPL_DATA) Data)
Service ID	0x03
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Config
Parameters (Inout)	State, Data
Parameters (Out)	None
Return Value	Std_ReturnType
Description	This service protects the array/buffer to be transmitted using the E2E profile 2. This includes checksum calculation, handling of sequence counter and Data ID.
Safety Function	Yes

Function Name	E2E_P02Check
Syntax	FUNC(Std_ReturnType, E2E_CODE) E2E_P02Check (P2VAR(E2E_P02ConfigType, AUTOMATIC, E2E_APPL_DATA) Config, P2VAR(E2E_P02ReceiverStateType, AUTOMATIC, E2E_APPL_DATA) State, P2VAR(uint8, AUTOMATIC, E2E_APPL_DATA) Data)
Service ID	0x04
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Config, Data
Parameters (Inout)	State
Parameters (Out)	None
Return Value	Std_ReturnType
Description	This service checks the array/buffer using the E2E profile 2. This includes

	checksum calculation, handling of sequence counter and Data ID.
Safety Function	Yes

4.2.3. Elementary Protocol Routines

Function Name	E2E_CRC8u8
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u8 (VAR(uint8, E2E_VAR) E2E_Data, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x07
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over uint8 data transmited with E2E Protocol, as in E2E Profile 1.
Safety Function	No

Function Name	E2E_CRC8u16
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u16 (VAR(uint16, E2E_VAR) E2E_Data, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x08
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over uint16 data transmited with E2E Protocol, as in E2E Profile 1.
Safety Function	No

Function Name	E2E_CRC8u32
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u32 (VAR(uint32, E2E_VAR) E2E_Data, VAR(uint8, E2E_VAR) E2E_StartValue)

Service ID	0x09
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over uint32 data transmited with E2E Protocol, as in E2E Profile 1.
Safety Function	No

Function Name	E2E_CRC8u8Array
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u8Array (P2CONST(uint8, AUTOMATIC, E2E_APPL_CONST) E2E_DataPtr, VAR(uint32, E2E_VAR) E2E_ArrayLength, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x0A
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over an array of uint8 transmited with E2E Protocol, as in E2E Profile 1.
Safety Function	No

Function Name	E2E_CRC8u16Array
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u16Array (P2CONST(uint16, AUTOMATIC, E2E_APPL_CONST) E2E_DataPtr, VAR(uint32, E2E_VAR) E2E_ArrayLength, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x0B
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8

Description	This service is the utility function for computing CRC over an array of uint16 transmited with E2E Protocol, as in E2E Profile 1.
Safety Function	No

Function Name	E2E_CRC8u32Array
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u32Array (P2CONST(uint32, AUTOMATIC, E2E_APPL_CONST) E2E_DataPtr, VAR(uint32, E2E_VAR) E2E_ArrayLength, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x0C
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over an array of uint32 transmited with E2E Protocol, as in E2E Profile 1.
Safety Function	No

Function Name	E2E_CRC8u8H2F
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u8H2F (VAR(uint8, E2E_VAR) E2E_Data, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x0D
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over uint8 data transmited with E2E Protocol, as in E2E Profile 2.
Safety Function	No

Function Name	E2E_CRC8u16H2F
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u16H2F (VAR(uint16, E2E_VAR) E2E_Data, VAR(uint8, E2E_VAR) E2E_StartValue)

Service ID	0x0E
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over uint16 data transmited with E2E Protocol, as in E2E Profile 2.
Safety Function	No

Function Name	E2E_CRC8u32H2F
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u32H2F (VAR(uint32, E2E_VAR) E2E_Data, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x0F
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over uint32 data transmited with E2E Protocol, as in E2E Profile 2.
Safety Function	No

Function Name	E2E_CRC8u8H2FArray
Syntax	FUNC(uint10, E2E_CODE) E2E_CRC8u8H2FArray (P2CONST(uint8, AUTOMATIC, E2E_APPL_CONST) E2E_DataPtr, VAR(uint32, E2E_VAR) E2E_ArrayLength, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x10
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over an array of uint8

	transmited with E2E Protocol, as in E2E Profile 2.
Safety Function	No

Function Name	E2E_CRC8u16H2FArray
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u16H2FArray (P2CONST(uint16, AUTOMATIC, E2E_APPL_CONST) E2E_DataPtr, VAR(uint32, E2E_VAR) E2E_ArrayLength, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x11
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over an array of uint16 transmited with E2E Protocol, as in E2E Profile 2.
Safety Function	No

Function Name	E2E_CRC8u32H2FArray
Syntax	FUNC(uint8, E2E_CODE) E2E_CRC8u32H2FArray (P2CONST(uint32, AUTOMATIC, E2E_APPL_CONST) E2E_DataPtr, VAR(uint32, E2E_VAR) E2E_ArrayLength, VAR(uint8, E2E_VAR) E2E_StartValue)
Service ID	0x12
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Data, StartValue
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service is the utility function for computing CRC over an array of uint32 transmited with E2E Protocol, as in E2E Profile 2.
Safety Function	No

Function Name	E2E_UpdateCounter
Syntax	FUNC(uint8, E2E_CODE) E2E_UpdateCounter (VAR(uint8, E2E_VAR) Counter)
Service ID	0x13

Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	Counter
Parameters (Inout)	None
Parameters (Out)	None
Return Value	uint8
Description	This service increments the counter provided by the parameter, and returns it by return value.
Safety Function	No

4.2.4. Auxiliary Functions

Function Name	E2E_GetVersionInfo
Syntax	FUNC(void, E2E_CODE) E2E_GetVersionInfo (P2VAR(Std_VersionInfoType, AUTOMATIC, E2E_APPL_DATA) VersionInfo)
Service ID	0x14
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (In)	VersionInfo
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	This service returns the version information of this module.
Safety Function	No

4.3. Safety Architecture

This section describes the results of software safety analysis (i.e. qualitative FMEA) and dependent failure analysis conducted during the software architectural design subphase and the principle of the safety architectural design developed on the basis of the results of these analyses, with respect the AUTOSAR E2E module.

[E2E_IR_#04] An integrator shall implement some of safety mechanisms in order to complete the error detection and handling according to the results of safety analysis of AUTOSAR E2E module.

[E2E_IR_#05] An integrator shall consider APIs provided by the AUTOSAR E2E module during software safety analysis of application software which uses the AUTOSAR E2E module according to

the characteristics of libraries in the AUTOSAR architecture. From the software safety analysis, safety mechanisms can also be applied. (e.g. redundancy-based plausibility of safety-related data)

4.3.1. Results of Software Safety Analysis

This subsection describes software failure modes defined and safety mechanisms applied while the software safety analysis was carried out for the AUTOSAR E2E module.

4.3.1.1. Defined Failure Modes

10 failure modes were defined with respect to software components to be verified via the software safety analysis during the software architectural design subphase (as reference see EN 50159 and ISO 26262). Table 4.5 below shows the software failure modes defined for the software safety analysis with respect to the AUTOSAR E2E module.

Table 4.5 Failure modes defined for software safety analysis

No.	Failure mode	Description
1	Repetition	When the same message is received more than once.
2	Deletion	When the message or parts of it have been removed from the communication stream.
3	Insertion	When an additional message or parts of it have been inserted into the communication stream (see e.g. EN 50159-2 table C.2).
4	Incorrect sequence	When messages of a communication stream are received in an incorrect order (see e.g. EN 50159-2 table C.2).
5	Corruption	When the corruption data of a message or parts of it occurred (see e.g. EN 50159-2 table C.2).
6	Timing faults (delay)	When the timing constraints of a message are violated (e.g. the message is received too late).
7	Addressing faults	When a message is sent to the wrong destination, which then treats reception as correct.
8	Inconsistency	When communicating nodes have a different view of network status or of data being transferred.
9	Masquerading	When the design of a received message with non-authentic content appears authentic as just sent by the appropriate sender.

4.3.1.2. Applied Safety Mechanisms

The selected safety mechanisms based on ISO 26262-6, Clauses 7.4.14 were applied considering the defined failure modes (see Table 4.5) while the software safety analysis was carried out for the AUTOSAR E2E module. Table 4.6 below shows the safety mechanisms applied for detecting failure modes, including the failure modes which could be detected by them.

Table 4.6 Applied safety mechanisms for detecting failure modes

No.	Applied safety mechanism	Detectable failure modes
1	Counter	Repetition
		Deletion
		Insertion
		Incorrect sequence
		(Addressing faults)
		(Masquerading)
2	Timeout (detection and handling implemented by SW-C)	Deletion
		Delay
3	Data ID	Insertion
		Addressing faults
		Masquerading
4	CRC	Corruption
		(Insertion)
		(Addressing faults)
		(Masquerading)

In addition to safety mechanisms to detect the failure modes mentioned in Table 4.6, the selected safety mechanisms were applied to handle detected failure modes. Table 4.7 below shows the safety mechanisms applied for handling failure modes detected by the safety mechanisms mentioned in Table 4.6.

Table 4.7 Applied safety mechanisms for handling detected failure modes

No.	Applied safety mechanism	Detected failure modes	Description
1	User-defined error	'Repetition'	If the failure mode, which could lead

No.	Applied safety mechanism	Detected failure modes	Description
	handling	'Deletion' 'Insertion' 'Incorrect sequence' 'Corruption' 'Timing faults (delay)' 'Addressing faults' 'Inconsistency' 'Masquerading'	to a violation of a safety requirement, in the AUTOSAR E2E module occurs, the AUTOSAR E2E module reports to users and users are to handle the detected failure modes.

4.3.2. Results of Dependent Failure Analysis

As mentioned in section 4.1, the AUTOSAR E2E module consists of safety-related software components assigned as ASIL D and non-safety-related software components assigned as QM. In case of cascading failures between the safety-related software components, the safety mechanisms identified by carrying out the software safety analysis could cover these failures, but they are not sufficient to protect against cascading failures between safety-related software components and non-safety-related software components. Therefore, the dependent failure analysis was carried out during the software architectural design subphase, in order to verify whether or not the freedom from interference between safety-related software components and non-safety-related software components has been achieved. However, cascading failures between safety-related software components and non-safety-related software components do not occur since they do not interact.

4.3.3. Principle of Safety Architectural Design

4.3.3.1. Safety mechanisms for detecting failure modes

4.3.3.1.1. Counter

On the sender side, for the first transmission request of a data element the counter is initialized with 0 and is incremented by 1 for every subsequent send request (from sender SW-C). On the receiver side, by evaluating the counter received data against the counter of previously received data, the following could be detected:

(1) No new data has arrived since last invocation of E2E library check function
(2) No new data has arrived since receiver start
(3) The data is repeated.

(4) Counter is incremented by on (i.e. no data lost)
(5) Counter is incremented more than by one, but still within allowed limits (i.e. some data lost).
(6) Counter is incremented more than allowed (i.e. too many data lost).

4.3.3.1.2. Timeout

If the attribute NewDataAvailable of State is FALSE, the transmission medium (e.g RTE) reports that no new data element is available at the transmission medium. Then, by means of the counter, the receiver can detect loss of communication and timeouts. If the attribute Status of State is E2E_P01STATUS_REPEATED, the transmission medium (e.g. RTE) provided new valid data element, but this data element has the same counter as the previous valid data element. Both conditions represent a timeout.

4.3.3.1.3. Data ID

The unique Data IDs are to verify the identity of each transmitted safety-related data element. The Data ID is transmitted implicitly. This means that Data ID is not transmitted together with the data, but it is included in the CRC calculation.

4.3.3.1.4. CRC

On the sender side, CRC value is written in safety-related data and transmitted for every subsequent send request (from sender SW-C). When calculating CRC, Data IDs and all serialized signal (including empty areas, excluding CRC byte itself) are included. On the receiver side, by evaluating the CRC received data against the expected CRC value from the received data, it could be detected whether the data is corrupted or masqueraded.

4.3.3.2. Safety mechanisms for handling detected failure modes

4.3.3.2.1. User-defined error handling

If the failure mode, which could lead to a violation of a safety requirement, in the AUTOSAR E2E module occurs, the AUTOSAR E2E module reports to users and users are supposed to handle the detected failure modes.

4.3.3.3. Safety mechanisms for dependent failure

Since cascading failures between safety-related software components and non-safety-related

software components do not occur, no safety mechanisms for dependent failure exist.

5. Verification

This chapter describes the verification of reviews and tests. It contains the outputs of verification reviews and explanation about the strategy and method of verification tests.

5.1. Results of Verification Reviews

In the work products described in Section 2.2, the target work products that verification review should be performed are decided based on ISO 26262-2.
The Table 5.1 shows the target work products and results of verification reviews for the AUTOSAR E2E module.

Table 5.1 Result work products of verification reviews

No.	Target work product	Result of verification review
1	Software Requirements Specification for E2E [D1]	Verification review of the Software Requirement_Check-list&Result for E2E [D2]
2	Software Architectural Design Specification for E2E [D3]	Verification review of the Software Architecture_Check-list&Result for E2E [D4]
3	Safety Analysis Report for E2E [D5]	Verification review of the Software Safety Analysis_Check-list&Result for E2E [D7]
4	Dependent Failure Analysis for E2E [D6]	Verification review of the Software Safety Analysis_Check-list&Result for E2E [D7]
5	Software Unit Design Specification for E2E [D8]	Verification review of the Software Unit Design_Check-list&Result for E2E [D9]
6	Software Requirement Test Specification for E2E [D14]	Inspection review of the Software Requirement Test_Check-list&Result for E2E [D15]

5.2. Results of Verification Tests

The AUTOSAR E2E module has no dependency with H/W. But, the verification test of the AUTOSAR E2E module was performed on 'Aurix TC275TE MCU' target board.

[E2E_IR_#06] Any differences between the target board used by Autron during testing and the actual target of the integrator must be analysed by the integrator.

[E2E_IR_#07] As The AUTOSAR E2E module is developed as SEooC, the integration and integration testing on the all over software must be done by the integrator.

5.2.1. Software Unit Test

Each of functions which compose a software unit in the "Software Unit Design Specification for E2E" [D8] is tested by means of test cases created by applying test methods (see Table 5.2) and coverage metrics (see Table 5.4 and Table 5.5), and Pass/Fail criteria necessary to judge results of the software unit test. In addition, if one or more test results are 'Fail', the process of change management is conducted.

The test methods selected for the software unit test are shown in Table 5.2 below.

Table 5.2 Test methods for software unit test

No.	Method
1	Requirements-based test
2	Interface test
3	Fault injection test

Table 5.3 below shows the methods selected in order to derive the appropriate test cases for the selected test method in Table 5.2 above.

Table 5.3 Methods for deriving test cases

No.	Method
1	Analysis of requirements
2	Generation and analysis of equivalence classes
3	Analysis of boundary values
4	Error guessing

Table 5.4 below shows the test coverage metrics measured to evaluate the completeness of test cases derived from methods in Table 5.3 and to obtain confidence that there are no unintended functionalities of software components and embedded software, including criteria to be fulfilled.

Table 5.4 Measured coverage metrics and their criteria

No.	Coverage metrics	Coverage Criteria (%)

1	Branch coverage	100
2	MC/DC (Modified Condition/Decision Coverage)	100

Because the some part of integration test cases is conducted together during the unit test the test coverage metrics for Function and Call coverage in Table 5.5 for the test cases in the software unit test phase are measured to evaluate the completeness of test cases derived from methods in Table 5.3 and to obtain confidence that there are no unintended functionalities of software components and embedded software, including criteria to be fulfilled.

Table 5.5 Measured coverage metrics and their criteria for integration test conducted during unit test

No.	Coverage metrics	Coverage Criteria (%)
1	Function coverage	100%
2	Call coverage	100%

Below Figure 5.1 shows the environment of SW Unit Test.

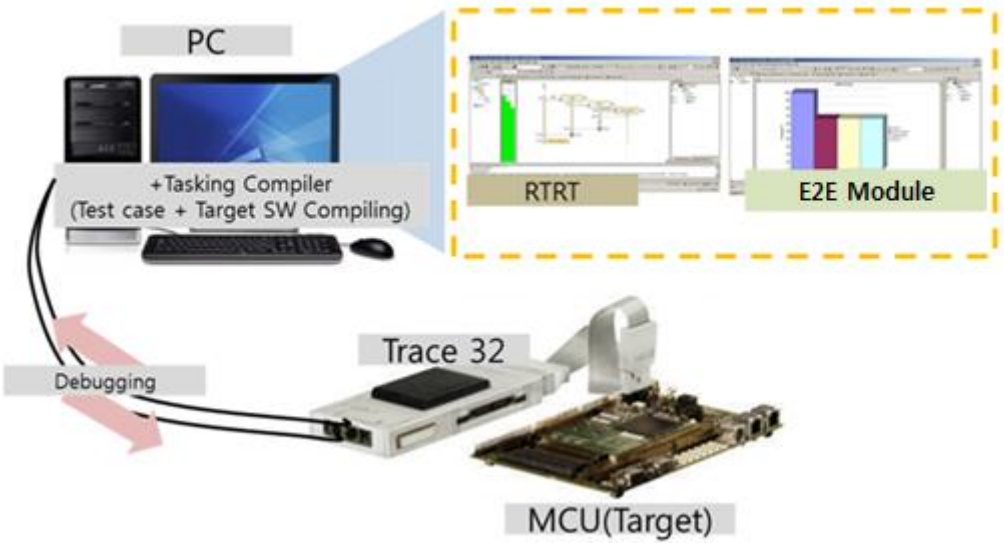

Figure 5.1 Environment of SW Unit Test

The environment of SW Unit Test is consists of a host PC, target MCU board and supporting tools that listed in Table 5.6.

Below Table 5.6 shows supporting tools of software unit test.

Table 5.6 Support Tool of Unit Testing

No	Category	Tool Name	Vendor	Version	Tool Confidence

					Level
1	Testing Tool	Rational Test RealTime	IBM	V8.0.0.4	TCL3
2	Compiler	TASKING VX-toolset for TriCore	Altium	V5.0r1	TCL2
3	Debugger	Trace32 PowerView for TriCore	Lauterbach	VAURIX	TCL1

Test procedures are as follows:

1. Tester creates RTRT test script based on "Software Unit Test Specification for E2E" [D10]
2. Run test script.
   A. Target software (i.e. AUTOSAR E2E module) and test codes generated from test script are compiled by compiler.
   B. Built binary file is loaded to target MCU board by debugger.
   C. Run tests.
3. Check result from test report that is generated by RTRT and write results to "Software Unit Test Report for E2E" [D11].

5.2.2. Software Integration Test

This Integration test use the sandwich testing approaching, based on the software hierarchy defined in the "Software Architectural Design Specification for E2E" [D3]. The sandwich testing is an approach to combine top down testing with bottom up testing.

The test methods selected for the software integration test are shown in Table 5.7 below.

Table 5.7 Test methods for software integration test

No.	Method
1	Requirements-based test
2	Interface test
3	Fault injection test

Table 5.8 below shows the methods selected in order to derive the appropriate test cases for the selected test method in Table 5.7 above.

Table 5.8 Methods for deriving test cases

No.	Method
1	Analysis of requirements

2	Generation and analysis of equivalence classes
3	Analysis of boundary values

Table 5.9 below shows the test coverage metrics measured to evaluate the completeness of test cases derived from methods in Table 5.7 and to obtain confidence that there are no unintended functionalities of software components and embedded software, including criteria to be fulfilled.

Table 5.9 Measured coverage metrics and their criteria

No.	Coverage metrics	Coverage Criteria (%)
1	Function coverage	100
2	Call coverage	100

Below Figure 5.2 shows the environment of SW Integration Test.

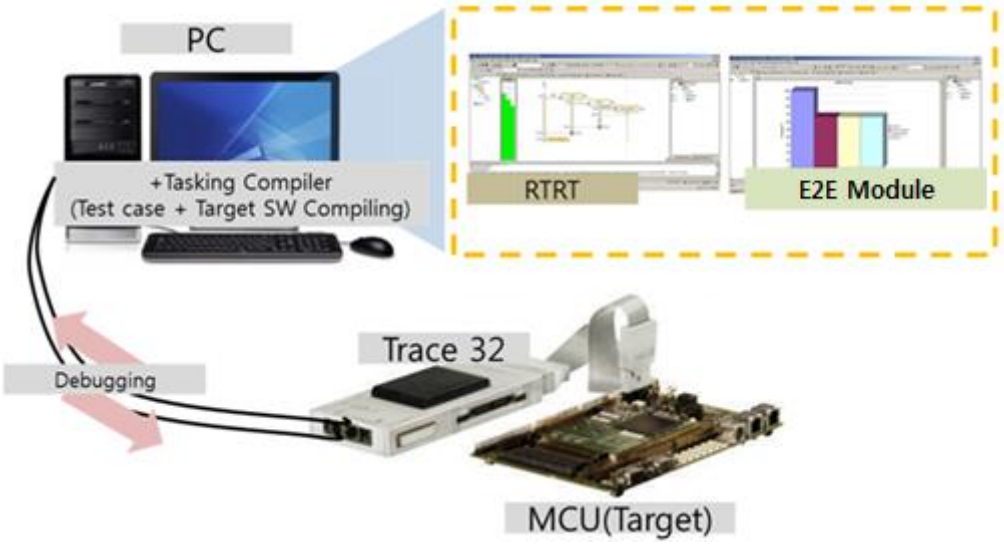

Figure 5.2 Environment of SW Integration Test

The environment of SW Integration Test is consists of a host PC, target MCU board and supporting tools that listed in Table 5.10.

Below Table 5.10 shows supporting tools of software integration test.

Table 5.10 Supporting Tool of Integration Test

No	Category	Tool Name	Vendor	Version	Tool Confidence Level

1	Testing Tool	Rational Test RealTime	IBM	V8.0.0.4	TCL3
2	Compiler	TASKING VX-toolset for TriCore	Altium	V5.0r1	TCL2
3	Debugger	Trace32 PowerView for TriCore	Lauterbach	VAURIX	TCL1

Integration test is performed on RTRT (Rational Test RealTime) which testing tool is described in Table 5.10. Test procedures are as follows:

1. Tester creates RTRT test script based on "Software Integration Test Specification for E2E" [D12].
2. Run test script.
   A. Target software (i.e. AUTOSAR E2E module) and test codes generated from test script are compiled by compiler.
   B. Built binary file is loaded to target MCU board by debugger.
   C. Run tests.
3. Check result from test report that is generated by RTRT and write results to "Software Integration Test Report for E2E" [D13].

5.2.3. Software Requirement Test

Functional test is carried out for all the requirements in the "Software Requirements Specification for E2E" [D1] by means of test cases created by applying the test method (see Table 5.11) and Pass/Fail criteria necessary to judge results of the functional test. In addition, if one or more test results are 'Fail', the process of change management is conducted.

Table 5.11 below shows the methods selected in order to derive the appropriate test cases for implementing the functional test.

Table 5.11 Methods for deriving test cases

No.	Method
1	Analysis of requirements

Below Figure 5.1 shows the environment of SW Requirement Test.

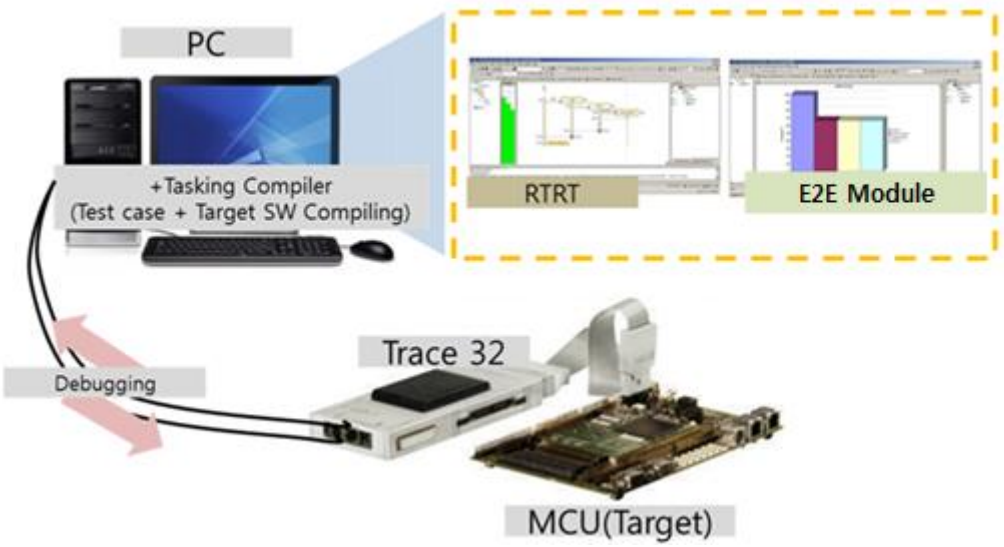

Figure 5.3 Environment of SW Requirement Test

The environment of SW Requirement Test is consists of a host PC, target MCU board and supporting tools that listed in Table 5.12.

Below Table 5.12 shows supporting tools of software requirement test.

Table 5.12 Supporting Tool of Requirement Test

No	Category	Tool Name	Vendor	Version	Tool Confidence Level
1	Compiler	TASKING VX-toolset for TriCore	Altium	V5.0r1	TCL2
2	Debugger	Trace32 PowerView for TriCore	Lauterbach	VAURIX	TCL1

[E2E_IR_#08] There is no compiled and built software but the source code delivered to the integrator. At Tasking Vx, it is only used to get built SW versions for unit- and integration tests. As the final software built as well as the SW qualification test of the integrator (assumed that unit-, integration and verification tests are run for the integration at the integrator's), the qualification of the compiler actually used for the productive software lies within the responsibility of the integrator.

Requirement test is performed on target MCU board with hand coded test program. Test procedures are as follows:

1. Tester creates test program based on "Software Requirement Test Specification for E2E" [D14].
2. Run test program.

A. Target software (i.e. AUTOSAR E2E module) and test codes are compiled by compiler.
B. Built binary file is loaded to target MCU board by debugger.
C. Run tests.

3. Check result from test log that is generated by debugger and write results to "Software Requirement Test Report for E2E" [D16].

Annex A Integration Requirements

Table A.0.1 below shows all integration requirements for AUTOSAR E2E module in this document. The contents of this table are linked to the descriptions of this document. The system designer or integrator should check all below integration requirement during integrating AUTOSAR platform which use the safety feature of the AUTOSAR E2E module.

Table A.0.1 List of integration requirements for AUTOSAR E2E module

Integration requirement ID	Integration Requirement
[E2E_IR_#01]	The safety and non-safety functions are assumed by Autron's definition in the Software Requirements Specification for E2E [D1] and analyzed by Dependent Failures Analysis Report for E2E [D6]. An integrator shall analyze that the violation of non-safety requirements does not lead to dangerous situation at the application level.
[E2E_IR_#02]	Since the AUTOSAR E2E library is a library which is invoked by function call of application software. This means that the safe state of the AUTOSAR E2E module with respect to a failure which is detected by external watchdog-related modules and External Watchdog depends on the safe state of application software. Therefore, the safe state of application software shall be defined properly by the integrator that integrates the AUTOSAR OS or the whole AUTOSAR Platform, in order to achieve and maintain a safe state on the AUTOSAR OS level or on the whole AUTOSAR Platform level.
[E2E_IR_#03]	The E2E Module separate from H/W interface. The related lower module will interface with H/W through MCAL layer. So, there is no function which interacts with external hardware in the AUTSOAR E2E module.
[E2E_IR_#04]	An integrator shall implement some of safety mechanisms in order to complete the error detection and handling according to the results of safety analysis of AUTOSAR E2E module.
[E2E_IR_#05]	An integrator shall consider APIs provided by the AUTOSAR E2E module during software safety analysis of application software which uses the AUTOSAR E2E module according to the characteristics of libraries in the AUTOSAR architecture. From the software safety analysis, safety mechanisms can also be applied. (e.g. redundancy-based plausibility of safety-related data)

Integration requirement ID	Integration Requirement
[E2E_IR_#06]	Any differences between the target board used by Autron during testing and the actual target of the integrator must be analysed by the integrator.
[E2E_IR_#07]	As The AUTOSAR E2E module is developed as SEooC, the integration and integration testing on the all over software must be done by the integrator.
[E2E_IR_#08]	There is no compiled and built software but the source code delivered to the integrator. At Tasking Vx, it is only used to get built SW versions for unit- and integration tests. As the final software built as well as the SW qualification test of the integrator (assumed that unit-, integration and verification tests are run for the integration at the integrator's), the qualification of the compiler actually used for the productive software lies within the responsibility of the integrator.