Elektrobit

EB tresos® E2E Profile 11 documentation

product release 8.8.7

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview of EB tresos E2E Profile 11 documentation

Welcome to the EB tresos E2E Profile 11 (E2EP11) product documentation.

This document provides:

- Chapter 2, “E2EP11 release notes“: release notes for the E2EP11 modules
- Chapter 3, “E2EP11 user's guide“: containing background information and instructions
- Chapter 4, “E2EP11 module references“: information about configuration parameters and the application programming interface

2. E2EP11 release notes

2.1. Overview

This chapter provides the E2EP11 product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

2.2. Scope of the release

2.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 29.2.0 b220916-0321

2.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this E2EP11 release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
E2EP11	4.3.0 []	4.3.0 [0000]	1.0.1	Elektrobit Automotive GmbH

Table 2.1. Hardware-Independent Modules specified by the AUTOSAR standard

2.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
SCrc	2.0.12	Elektrobit Automotive GmbH

Table 2.2. Modules not specified by the AUTOSAR standard

2.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

2.3. Module release notes

2.3.1. E2EP11 module release notes

- AUTOSAR R4.3 Rev 0
- AUTOSAR SWS document version: 4.3.0
- Module version: 1.0.1.B632837
- Supplier: Elektrobit Automotive GmbH

2.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 1.0.1
2022-10-12

- Internal module improvement. This module version update does not affect module functionality

Module version 1.0.0
2022-01-28

[1] `$TRESOS_BASE` is the location at which you installed EB tresos Studio.

- Initial release

2.3.1.2. New features

- No new features have been added since the last release.

2.3.1.3. Elektrobit-specific enhancements

This chapter lists the enhancements provided by the module.

- [ASCE2E-898] Support for Profile 11 variant B

  Description:

  Provision of the new Profile variant 11B which uses a 16-bit DataId where depending on parity of the counter (alternating dataID configuration) the low or the high byte is included. For even counter values the low byte is included and for odd counter values the high byte is included.

  Requirements:

  SWS_E2E_00513, SWS_E2E_00565

2.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- The E2E library uses the Safety CRC module `SCrc` instead of the AUTOSAR `Crc` module

  Description:

  Instead of the standard AUTOSAR `Crc` module, a Safety CRC module named `SCrc` is used. That is, instead of `Crc_CalculateCRC8()`, the function `SCrc_CalculateCRC8()` of the `SCrc` module is called.

  Rationale:

  The implementation of the `SCrc` library complies with the requirements for the development of safety-related software for the automotive domain.

  Requirements:

  SWS_E2E_00508

- Achieving bus compatibility with Profile 01

Description:

Requirement EB_E2EP11020513 incorporates requirement PRS_E2E_00513 of E2E Protocol Specification Release R21-11. PRS_E2E_00513 adds a XOR operation of `ComputedCRC` with 0xFF before finally returning the `ComputedCRC` to achieve bus compatibility with Profile 01.

For more information, see https://jira.autosar.org/browse/AR-90756.

Rationale:

Add step `ComputedCRC = ComputedCRC ^ 0xFF` to the end of the "Compute CRC" figure.

Requirements:

SWS_E2E_00513

2.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

For this module no constraints and limitations are known.

2.3.1.6. Open-source software

E2EP11 does not use open-source software.

2.3.2. SCrc module release notes

- Module version: 2.0.12.B632837
- Supplier: Elektrobit Automotive GmbH

2.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 2.0.12
2022-05-11

- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.11
2021-03-05

- Updated preprocessor include guards to be PC-lint compatible
- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.10
2020-06-19

- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.9
2019-10-11

- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.8
2019-06-14

- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.7
2019-02-15

- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.6
2018-10-26

- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.5
2018-02-01

- Internal module improvement. This module version update does not affect module functionality

Module version 2.0.4
2017-09-22

- Switch from MISRA-C:2004 to MISRA-C:2012

Module version 2.0.3
2015-06-19

- Added CRC routines required for new E2E Profiles 4, 5, 6

Module version 2.0.2
2013-11-29

- Non-functional code improvements (Inspection findings, Consistent use of static qualifier).
- ASCSCRC-74 Fixed known issue: The SCrc module used a compiler abstraction macro of the Crc module.

Module version 2.0.1
2013-02-15

- Provide a Basic Software Module Description that specifies the Memory Mappings.

Module version 2.0.0
2012-06-22

- Update to ASR 4.0 Rev3 (New parameter isFirstCall in CalculateCRC APIs).

Module version 1.0.8
2011-09-16

- Non-functional code improvements.

Module version 1.0.7
2011-05-20

- Usage of unix line endings.

Module version 1.0.6
2011-04-20

- Correction of common published information.
- Correction of compiler abstraction.

Module version 1.0.5
2011-03-18

- Macro definitions for switching between Crc implementation of SCrc module or external Crc module.

Module version 1.0.4
2011-02-28

- The GetVersionInfoApi() removed from the SCrc module.

Module version 1.0.3
2011-02-14

- Elimination of misra violations.

Module version 1.0.2
2010-12-23

- Table-based 8-bit Crc routine according to SAE J1850 CRC.

Module version 1.0.1
2010-11-30

- Removed unintended header file in Crc.c.

Module version 1.0.0
2010-11-12

- Table-based 8-bit Crc routine with generator polynomial 0x2F.

2.3.2.2. New features

- No new features have been added since the last release.

2.3.2.3. Elektrobit-specific enhancements

This module is not part of the AUTOSAR specification.

2.3.2.4. Deviations

This module is not part of the AUTOSAR specification.

2.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- For this module no limitations are known.

2.3.2.6. Open-source software

Open-source software information is not available for this module.

3. E2EP11 user's guide

3.1. Overview

This user's guide describes the `E2EP11` module. From this user's guide you learn the basic functionality of the `E2EP11`. You also learn which related modules are necessary to configure the `E2EP11` module. The `E2EP11` module reference provides further information on how to configure the `E2EP11` itself.

Note that this user's guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the `E2EP11`. The information provided here helps you to integrate the `E2EP11` in your AUTOSAR project.

- Section 3.2, “Background information” provides an overview of the basic functionality of the `E2EP11`.
- Section 3.3, “Configuring E2EP11” provides information on related modules that are needed in order to configure the `E2EP11`.
- Section 3.4, “E2EP11 integration notes” provides notes for the integration of the `E2EP11` module into your project.
- For details on how to configure the `E2EP11` itself, see the parameter descriptions provided in the `E2EP11` module reference Chapter 4, “E2EP11 module references“, which is provided together with the dependent modules `E2E` and `SCrc`.

3.2. Background information

The general concept of end-to-end communication protection is described in the EB tresos E2E Protection Transformer documentation of the `E2EXf` module, based on the AUTOSAR E2E transformer specified in [2].

3.2.1. Functional overview

3.2.1.1. Safety mechanisms

This profile is based on E2E Profile 11 specified by AUTOSAR, see [1]. It is called from the virtual functional bus generated by the `Rte` module together with a previously called serializing transformer, e.g. `ComXf` to add protection information to the serialized data stream for the following communication paradigms:

- Non-blocking queued sender-receiver communication

`E2EP11` provides APIs to add protection information at the sender to the result of a serializing transformer, e.g. `ComXf`. It also provides APIs to cyclically check for communication errors by using this information at the receiver. Its API functions are called by the `E2EXf` module.

The `E2EP11` module uses the following safety mechanisms:

- **Cyclic redundancy check (CRC):** An 8-bit CRC is explicitly sent with polynomial in normal form `0x1D` with an initial value `0xFF` and a final XOR-value `0xFF`. The bit-offset of the CRC value within a transmitted signal group is configurable but must be byte-aligned.
- **Sequence counter/alive counter:** An 4-bit sequence number with a counter that represents numbers from 0 to 14 is explicitly sent and incremented at every transmission request. The bit-offset of the sequence counter/alive counter value within a transmitted signal group is configurable but must be aligned with respect to nibbles.
- **System-wide unique 16-bit data ID for every port data element sent over a port:** The following data ID inclusion modes can be configured:
  - **Both bytes** (dataIdMode=0): Both bytes of the 16-bit data ID are attached to the safety data for CRC calculation, but not explicitly sent.
  - **Alternating bytes** (dataIdMode=1): The low and the high byte are alternating attached to the safety data for CRC calculation depending on parity of the counter, but not explicitly sent. For even counter values the low byte is included and for odd counter values the high byte is included.
  - **Explicit transmission of data ID nibble** (dataIdMode=3): Both bytes of the 16-bit data ID are attached to the safety data for CRC calculation, but the low nibble of the high byte of the data ID is explicitly transmitted. Only 12 bits are used in this 16-bit data ID and the high nibble of the high byte is set to `0`. The bit-offset of the data ID nibble value within a transmitted signal group is configurable, but must be aligned with respect to nibbles. To be able to use this data ID inclusion mode together with the data ID inclusion mode **Both bytes**, the CRC is calculated over the low byte of the data ID and the high byte which is set to `0`. For more information on explicit transmission of data ID nibbles, see Figure 3.3, “Layout of the protected message including control data (CRC, SEQ) with explicit transmission of data ID nibble (`dataIdMode=3`)”.

Figure 3.1, “Layout of the protected message including control data (CRC, SEQ) with 2-byte data ID (`dataIdMode=0`)” shows the layout of the AUTOSAR E2E Profile 11 with a CRC offset of 0 bits and a sequence counter/alive counter offset of 8 bits for `dataIdMode=0`.

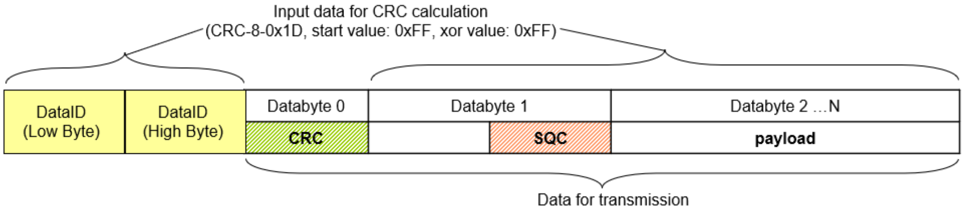

Figure 3.1. Layout of the protected message including control data (CRC, SEQ) with 2-byte data ID (`dataIdMode=0`)

Figure 3.2, "Layout of the protected message including control data (CRC, SEQ) with 2-byte data ID (dataIdMode=1)" shows the layout of the AUTOSAR E2E Profile 11 with a CRC offset of 0 bits and a sequence counter/alive counter offset of 8 bits for dataIdMode=1.

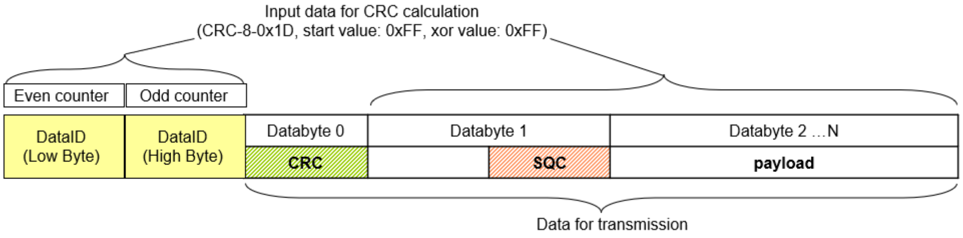

Figure 3.2. Layout of the protected message including control data (CRC, SEQ) with 2-byte data ID (dataIdMode=1)

Figure 3.3, "Layout of the protected message including control data (CRC, SEQ) with explicit transmission of data ID nibble (dataIdMode=3)" shows the layout of the AUTOSAR E2E Profile 11 with a CRC offset of 0 bits and a sequence counter/alive counter offset of 8 bits and a data ID nibble offset of 12 bits as used for dataIdMode=3.

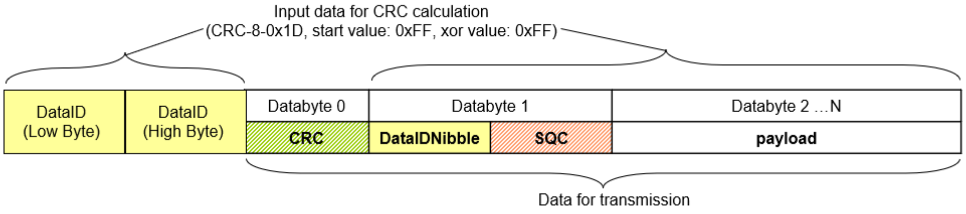

Figure 3.3. Layout of the protected message including control data (CRC, SEQ) with explicit transmission of data ID nibble (dataIdMode=3)

3.2.1.2. Failure modes and required safety mechanisms

The Table 3.1, "Failure modes detection matrix for E2E Profile 11" shows the failure modes and the required safety mechanisms of E2E Profile 11 with the different data ID variants for detection of the failure mode.

NOTE **Different data ID inclusion modes**

The different data ID inclusion modes only limits the applicable range of data IDs which can be used to detect masquerading.

An X specifies that the failure mode can be detected by the safety mechanism implemented in the E2E Profile.

An (X) specifies a safety mechanism which is only required to implement another safety mechanism.

An A specifies that the failure mode can be detected by a safety mechanism implemented in the data sink.

Failure mode/ safety mechanism	Sequence counter	CRC	Data ID	Timeout detection
Unintended message repetition	X			
Message loss	X			A
Insertion of message	X	(X)	X	
Resequencing	X			
Message corruption		X		
Delayed reception				A
Addressing faults	(X)	(X)	X	
Masquerading	(X)	(X)	X	

Table 3.1. Failure modes detection matrix for E2E Profile 11

3.3. Configuring E2EP11

To configure the `E2EP11` module, add the module to your project using EB tresos Studio. This module does not provide any configuration parameters except a common published information. You find this information in the module references section of this document. You also find these in the parameter description in EB tresos Studio.

To use the `E2EP11` module, you must configure additional modules as outlined below:

- The `E2EP11` module requires API functions and data types from the `E2E` library module. This module does not provide any configuration parameters.
- The `E2EP11` module requires API functions and data types from the `SCrc` library module. This module does not provide any configuration parameters.
- The `E2EP11` module provides API functions and data types required from the `E2ESM` library module and from the `E2EXf` module. For more information on the `E2ESM` and `E2EXf` modules, see [1] and [2]).

3.4. E2EP11 integration notes

You find general integration information in the EB tresos AutoCore Generic documentation.

In addition, you find module-specific information about exclusive areas, production errors and memory mapping in the module-specific integration notes. You find the module-specific integration notes in the module references chapter of this document. See Chapter 4, “E2EP11 module references“ sub-section `Integration notes` in each module.

4. E2EP11 module references

4.1. Overview

This chapter provides module references for the E2EP11 product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter E2EP11 user's guide.

4.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

4.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

4.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.

4.2. E2EP11

4.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

4.2.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
VendorApiInfix	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorApiInfix**
Multiplicity	1..1
Type	STRING_LABEL

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

4.2.1.2. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the E2EP11 can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

4.2.2. Application programming interface (API)

4.2.2.1. Type definitions

4.2.2.1.1. E2E_P11CheckStateType

Purpose	Definition of E2E Profile 11 receiver state type.	
Type	struct	
Members	`E2E_P11CheckStatusType Status`	Result of the verification of the Data, determined by the Check function.
	`uint8 Counter`	Counter of last valid received message.
Description	State of the sender for a Data protected with E2E Profile 11.	

4.2.2.1.2. E2E_P11CheckStatusType

Purpose	Definition of E2E Profile 11 receiver status type.	
Type	enum	
Constants	E2E_P11STATUS_OK	New data has been correctly received.
	E2E_P11STATUS_NONEWDATA	The Check function has been invoked but new Data is not available since the last call.
	E2E_P11STATUS_ERROR	The data has been received according to communication medium, but the CRC is incorrect or wrong length
	E2E_P11STATUS_REPEATED	New data has been correctly received, but the Counter is identical to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST.
	E2E_P11STATUS_OKSOMELOST	New data has been correctly received, but some data in the sequence have been probably lost.
	E2E_P11STATUS_WRONGSEQUENCE	The new data has been correctly received, but the Counter Delta is too big (DeltaCounter > MaxDeltaCounter)

Description	Result of the verification of the Data in E2E Profile 11, determined by the Check function.

4.2.2.1.3. E2E_P11ConfigType

Purpose	Configuration of transmitted Data for E2E Profile 11.	
Type	struct	
Members	`uint16 DataLength`	Length of the data in bits. The value shall be a multiple of 8 and shall be <= 240.
	`uint16 DataID`	A unique identifier for protection against masquerading.
	`uint8 MaxDeltaCounter`	Maximum allowed gap between two counter values of two consecutively received valid Data.
	`E2E_P11DataIDMode DataIDMode`	Inclusion mode of ID in CRC computation.
	`uint16 CRCOffset`	Bit offset of CRC (Least Significant Bit) from the beginning of the Byte-Array (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8. For example, offset 8 means that the CRC will take the byte 1, i.e. bits 8..15.
	`uint16 CounterOffset`	Bit offset of Counter (Least Significant Bit) from the beginning of the Byte-Array (bit numbering: bit 0 is the least important). The offset shall be a multiple of 4. For example, offset 8 means that the Counter will take the low nibble of the byte 1, i.e. bits 8..11.
	`uint16 DataIDNibbleOffset`	Bit offset of the low nibble (Least Significant Bit) of the high byte of Data ID from the beginning of the Byte-Array (bit numbering: bit 0 is the least important). The offset shall be a multiple of 4. For example, offset 8 means that the DataIDNibble will take the low nibble of the byte 1, i.e. bits 8..11.
Description	Configuration of transmitted Data (Data Element or I-PDU), for E2E Profile 11. For each transmitted Data, there is an instance of this typedef.	

4.2.2.1.4. E2E_P11DataIDMode

Purpose	Inclusion modes of Data ID for E2E Profile 11.	
Type	enum	
Constants	E2E_P11_DATAID_BOTH	Two bytes are included in the CRC .
	E2E_P11_DATAID_ALT	One of the two bytes is included, alternating high and low byte.
	E2E_P11_DATAID_NIBBLE	The low byte is included in the implicit CRC calculation, the low nibble of the high byte is explicitly transmitted along with the data, the high nibble of the high byte is not used.
Description	The Data ID is two bytes long in E2E Profile 11. There are three inclusion modes how the implicit two-byte Data ID is included in the one-byte CRC.	

4.2.2.1.5. E2E_P11ProtectStateType

Purpose	State of the sender for a Data protected with E2E Profile 11.	
Type	struct	
Members	`uint8 Counter`	Counter to be used for protecting the Data. The counter is incremented modulo 14

4.2.2.2. Macro constants

4.2.2.2.1. E2EP11_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

4.2.2.2.2. E2EP11_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	3U

4.2.2.2.3. E2EP11_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	0U

4.2.2.2.4. E2EP11_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	1U

4.2.2.2.5. E2EP11_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	0U

4.2.2.2.6. E2EP11_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	1U

4.2.2.2.7. E2EP11_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

4.2.2.3. Functions

4.2.2.3.1. E2E_P11Check

Purpose	Check the received Data using the E2E Profile P11.
Synopsis	`Std_ReturnType` **`E2E_P11Check`** `( const E2E_P11ConfigType * ConfigPtr , E2E_P11CheckStateType * StatePtr , const uint8 * DataPtr , uint16 Length );`
Service ID	0x38

Sync/Async	Synchronous	
Reentrancy	Reentrant for different communication data / states	
Parameters (in)	`ConfigPtr`	Pointer to static configuration.
	`DataPtr`	Pointer to received Data.
	`Length`	Length of the data in bytes.
Parameters (in,out)	`StatePtr`	Pointer to port/data communication state.
Return Value	Function execution success status	
	`E2E_E_INPUTERR_NULL`	At least one pointer parameter is a NULL pointer.
	`E2E_E_INPUTERR_WRONG`	At least one input parameter is erroneous.
	`E2E_E_OK`	Function completed successfully.
Description	Checks the Data received using the E2E Profile P11. This includes CRC calculation, handling of Sequence Counter and Data ID.	

4.2.2.3.2. E2E_P11CheckInit

Purpose	Initializes the check state.	
Synopsis	Std_ReturnType **E2E_P11CheckInit** (E2E_P11CheckStateType * StatePtr);	
Service ID	0x39	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different states	
Parameters (out)	`StatePtr`	Pointer to port/data communication state.
Return Value	Function execution success status	
	`E2E_E_INPUTERR_NULL`	NULL pointer passed.
	`E2E_E_OK`	Function completed successfully.
Description	Initializes the state structure by setting: Counter = 0x0EU Status = E2E_P11STATUS_ERROR	

4.2.2.3.3. E2E_P11MapStatusToSM

Purpose	Maps the check status of Profile 11 to a generic check status.

Synopsis	`E2E_PCheckStatusType` **`E2E_P11MapStatusToSM`** `( Std_ReturnType CheckReturn , E2E_P11CheckStatusType Status );`	
Service ID	0x3a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different status types	
Parameters (in)	`CheckReturn`	Return value of the E2E_P11Check function.
	`Status`	Status determined by E2E_P11Check function.
Return Value	Profile-independent status of the reception on one single Data in one cycle.	
	`E2E_P_OK`	CheckReturn is E2E_E_OK and Status is E2E_P11STATUS_OK or E2E_P11STATUS_OKSOMELOST
	`E2E_P_ERROR`	CheckReturn is E2E_E_OK and Status is E2E_P11STATUS_ERROR or CheckReturn is different than E2E_E_OK
	`E2E_P_REPEATED`	CheckReturn is E2E_E_OK and Status is E2E_P11STATUS_REPEATED.
	`E2E_P_NONEWDATA`	CheckReturn is E2E_E_OK and Status is E2E_P11STATUS_NONEWDATA.
	`E2E_P_WRONGSEQUENCE`	CheckReturn is E2E_E_OK and Status is E2E_P11STATUS_WRONGSEQUENCE
Description	The function maps the check status of Profile 11 to a generic check status, which can be used by E2E state machine check function. The E2E Profile 11 delivers a more fine-granular status, but this is not relevant for the E2E state machine.	

4.2.2.3.4. E2E_P11Protect

Purpose	Protects the array/buffer to be transmitted using the E2E Profile P11.
Synopsis	`Std_ReturnType` **`E2E_P11Protect`** `( const E2E_P11ConfigType * ConfigPtr , E2E_P11ProtectStateType * StatePtr , uint8 * DataPtr , uint16 Length );`
Service ID	0x3b
Sync/Async	Synchronous
Reentrancy	Reentrant for different communication data / states

Parameters (in)	`ConfigPtr`	Pointer to static configuration.
	`Length`	Length of the data in bytes.
Parameters (in,out)	`StatePtr`	Pointer to port/data communication state.
	`DataPtr`	Pointer to Data to be protected.
Return Value	Function execution success status	
	`E2E_E_INPUTERR_NULL`	At least one pointer parameter is a NULL pointer.
	`E2E_E_INPUTERR_WRONG`	At least one input parameter is erroneous.
	`E2E_E_OK`	Function completed successfully.
Description	Protects the array/buffer to be transmitted using the E2E Profile P11. This includes checksum calculation, handling of counter and Data ID.	

4.2.2.3.5. E2E_P11ProtectInit

Purpose	Initializes the protection state.	
Synopsis	`Std_ReturnType` **`E2E_P11ProtectInit`** `( E2E_P11ProtectStateType * StatePtr );`	
Service ID	0x3c	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different states	
Parameters (out)	`StatePtr`	Pointer to port/data communication state.
Return Value	Function execution success status	
	`E2E_E_INPUTERR_NULL`	NULL pointer passed.
	`E2E_E_OK`	Function completed successfully.
Description	Initializes the state structure by setting the Counter to 0.	

4.2.3. Integration notes

4.2.3.1. Exclusive areas

Exclusive areas are not used by the `E2EP11` module.

4.2.3.2. Production errors

Production errors are not reported by the `E2EP11` module.

4.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE

4.2.3.4. Integration requirements

WARNING ⚠ **Integration requirements list is not exhaustive**

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the E2EP11 module.

4.3. SCrc

4.3.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information

Containers included		
		Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.
SCrcPublishedInformation	1..1	Additional published parameters not covered by CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, since they are published information.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile
Range	VariantPreCompile

4.3.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1

Parameters included	
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**

Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	12
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

4.3.1.2. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the SCrc can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	

4.3.1.3. SCrcPublishedInformation

4.3.2. Application programming interface (API)

4.3.2.1. Macro constants

4.3.2.1.1. SCRC_FUNCENABLED_SCRC_CRC16

Purpose	Defines whether or not SCrc_CalculateCRC16 shall map to SCrc_CalculateCRC16() of this SCrc module (STD_ON) or shall map to an external function Crc_CalculateCRC16() declared in Crc.h (STD_OFF).
Value	STD_ON

4.3.2.1.2. SCRC_FUNCENABLED_SCRC_CRC32P4

Purpose	Defines whether or not SCrc_CalculateCRC32P4 shall map to SCrc_CalculateCRC32P4() of this SCrc module (STD_ON) or shall map to an external function Crc_CalculateCRC32P4() declared in Crc.h (STD_OFF).
Value	STD_ON

4.3.2.1.3. SCRC_FUNCENABLED_SCRC_CRC8

Purpose	Defines whether or not SCrc_CalculateCRC8 shall map to SCrc_CalculateCRC8() of this SCrc module (STD_ON) or shall map to an external function Crc_CalculateCRC8() declared in Crc.h (STD_OFF).
Value	STD_ON

4.3.2.1.4. SCRC_FUNCENABLED_SCRC_CRC8H2F

Purpose	Defines whether or not SCrc_CalculateCRC8H2F shall map to SCrc_CalculateCRC8H2F() of this SCrc module (STD_ON) or shall map to an external function Crc_CalculateCRC8H2F() declared in Crc.h (STD_OFF).
Value	STD_ON

4.3.2.1.5. SCRC_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	2U

4.3.2.1.6. SCRC_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	0U

4.3.2.1.7. SCRC_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	12U

4.3.2.1.8. SCRC_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

4.3.2.2. Functions

4.3.2.2.1. SCrc_CalculateCRC16

Purpose	Calculation of CRC16.

Synopsis	uint16 **SCrc_CalculateCRC16** (const uint8 * SCrc_DataPtr , uint32 SCrc_Length , uint16 SCrc_StartValue16 , boolean SCrc_IsFirstCall);	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SCrc_DataPtr	Valid pointer to start address of data block
	SCrc_Length	Length of data block in bytes
	SCrc_StartValue16	Initial Value
	SCrc_IsFirstCall	TRUE: First call in a sequence or individual CRC calculation; start from initial value, ignore Crc_StartValue16. FALSE: Subsequent call in a call sequence; Crc_StartValue16 is interpreted to be the return value of the previous function call.
Return Value	calculated CRC16 value	
Description	This function performs the calculation of a 16-bit CRC value over the memory block referenced by SCrc_DataPtr of byte length SCrc_Length.	

4.3.2.2.2. SCrc_CalculateCRC32P4

Purpose	Calculation of CRC32P4.	
Synopsis	uint32 **SCrc_CalculateCRC32P4** (const uint8 * SCrc_DataPtr , uint32 SCrc_Length , uint32 SCrc_StartValue32P4 , boolean SCrc_IsFirstCall);	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SCrc_DataPtr	Valid pointer to start address of data block
	SCrc_Length	Length of data block in bytes
	SCrc_StartValue32P4	Initial Value
	SCrc_IsFirstCall	TRUE: First call in a sequence or individual CRC calculation; start from initial value, ignore Crc_StartValue8. FALSE: Subsequent call in a call sequence; Crc_StartValue32P4 is interpreted to be the return value of the previous function call.
Return Value	calculated CRC32 value	

Description	This function performs the calculation of a 32-bit CRC value for Ethernet over the memory block referenced by SCrc_DataPtr of byte length SCrc_Length.

4.3.2.2.3. SCrc_CalculateCRC8

Purpose	Calculation of CRC8.	
Synopsis	`uint8 SCrc_CalculateCRC8 ( const uint8 * SCrc_DataPtr , uint32 SCrc_Length , uint8 SCrc_StartValue8 , boolean SCrc_IsFirstCall );`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`SCrc_DataPtr`	Valid pointer to start address of data block
	`SCrc_Length`	Length of data block in bytes
	`SCrc_StartValue8`	Initial Value
	`SCrc_IsFirstCall`	TRUE: First call in a sequence or individual CRC calculation; start from initial value, ignore Crc_StartValue8. FALSE: Subsequent call in a call sequence; Crc_StartValue8 is interpreted to be the return value of the previous function call.
Return Value	calculated CRC8 value	
Description	This function performs the calculation of a 8-bit SAE J1850 CRC value over the memory block referenced by SCrc_DataPtr of byte length SCrc_Length.	

4.3.2.2.4. SCrc_CalculateCRC8H2F

Purpose	Calculation of CRC8 with the Polynomial 0x2F.	
Synopsis	`uint8 SCrc_CalculateCRC8H2F ( const uint8 * SCrc_DataPtr , uint32 SCrc_Length , uint8 SCrc_StartValue8H2F , boolean SCrc_IsFirstCall );`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`SCrc_DataPtr`	Valid pointer to start address of data block
	`SCrc_Length`	Length of data block in bytes

	`SCrc_StartValue8`	Initial Value
	`SCrc_IsFirstCall`	TRUE: First call in a sequence or individual CRC calculation; start from initial value, ignore Crc_StartValue8. FALSE: Subsequent call in a call sequence; Crc_StartValue8 is interpreted to be the return value of the previous function call.
Return Value	calculated CRC8 value	
Description	This function performs the calculation of a 8-bit CRC value with the Polynomial 0x2F over the memory block referenced by SCrc_DataPtr of byte length SCrc_Length.	

4.3.3. Integration notes

4.3.3.1. Exclusive areas

Exclusive areas are not used by the `SCrc` module.

4.3.3.2. Production errors

Production errors are not reported by the `SCrc` module.

4.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
CONST_8
CONST_16
CONST_32

4.3.3.4. Integration requirements

WARNING ⚠

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the SCrc module.

5. Bibliography

Bibliography

[1] *AUTOSAR Specification of SW-C End-to-End Communication Protection Library*, Issue AUTOSAR Release 4.3.0, Publisher: AUTOSAR

[2] *AUTOSAR Specification of Module E2E Transformer*, Issue AUTOSAR Release 4.3.0, Publisher: AUTOSAR