Document Title	E2E Protocol Specification
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	849

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R19-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2019-11-28	R19-11	AUTOSAR Release Management	• Introduction of Constraints for Client-Server Communication. • Added E2E_PXXForward functionality to provide a mechanism for replicating received E2E Errors. • Incorporated new configuration options for switching between valid and invalid state of E2E-Statemachine. • Fixed interoperability issues between P01 and P11, P02 and P22. • Changed Document Status from Final to published.
2019-03-29	1.5.1	AUTOSAR Release Management	• clarification on choosing suitable maximum data lengths for E2E profiles.

2018-10-31	1.5.0	AUTOSAR Release Management	• Migrated all functional specifications from Classic Platform's SWS E2ELibrary into Foundation's E2E Protocol Specification • Moved all figures and tables out of specifications and added references to them • Fixed duplicate/missing figures in profiles 2 (Calculate DeltaCounter), 5 (Read CRC), 6 (Read Counter) and 11 (Read DataIDNibble). • Added protocol examples for each profile
2018-03-29	1.4.0	AUTOSAR Release Management	• No content changes
2017-12-08	1.3.0	AUTOSAR Release Management	• No content changes
2017-10-27	1.2.0	AUTOSAR Release Management	• Initial Release

Disclaimer

Table of Contents

1 Introduction and functional overview

The concept of E2E communication protection assumes that safety-related data exchange shall be protected at runtime against the effects of faults on the communication link (see Figure 1.1). Faults detected between a sender and a receiver using E2E communication protection include systematic software faults, such as faults that are introduced on the lower communication layers of sender or receiver, and random hardware faults introduced by the MCU hardware, communication peripherals, transceivers, communication lines or other communication infrastructure.

Examples for such faults are random HW faults (e.g. corrupt registers of a CAN transceiver), interference (e.g. due to EMC), and systematic faults of the lower communication layers (e.g. RTE, IOC, COM and network stacks).

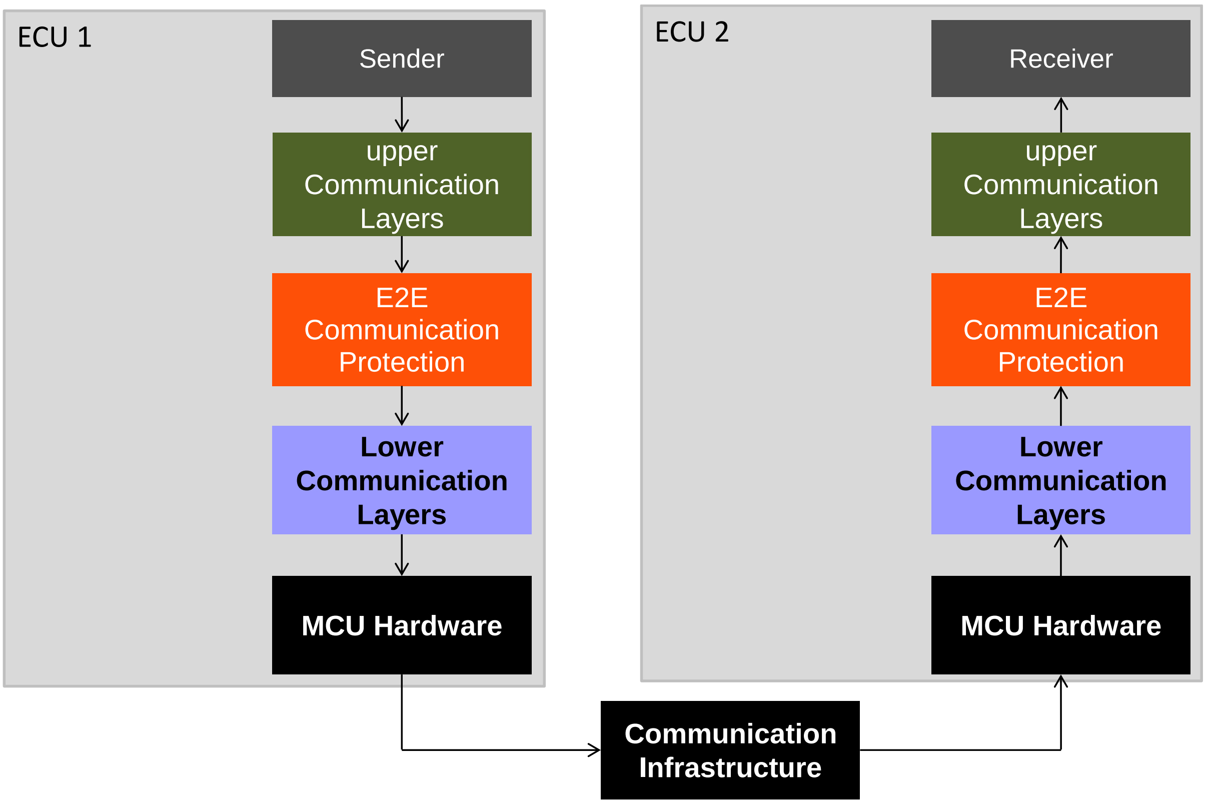

Figure 1.1: Overview of E2E communication protection between a sender and a receiver

By using E2E communication protection mechanisms, faults in lower software and hardware layers can be detected and handled at runtime. The E2E Supervision provides mechanisms for E2E communication protection, adequate for safety-related communication having requirements up to ASIL D.

The algorithms of protection mechanisms are implemented in the E2E Supervision. The callers of the E2E Supervision are responsible for the correct usage of the E2E Supervision, in particular for providing correct parameters the E2E Supervision routines.

The E2E communication protection allows the following:

1. It protects the safety-related data to be sent by adding control data,
2. It verifies the safety-related data received using this control data, and
3. It provides the check result to the receiver, which then has to handle it sufficiently.

To provide the appropriate solution addressing flexibility and standardization, AUTOSAR specifies a set of flexible E2E profiles that implement an appropriate combination of E2E communication protection mechanisms. Each specified E2E profile has a fixed set of mechanisms, as well as configuration options to configure the protocol header layout and status evaluation on the receiver side.

The E2E Supervision can be invoked from communication middleware e.g. from Adaptive Platform's ARA, Classic Platform's RTE. It can be also invoked in a non-standardized way from other software, e.g. non-volatile memory managers, local IPCs, or intra-ECU bus stacks.

Appropriate usage of the E2E Supervision to fulfill the specific safety requirements for communication depends on several aspects. The specified profiles are capable, to a high probability, of detecting a large variety of communication faults. However, the use of a specific E2E profile requires the user to demonstrate that the selected profile provides sufficient error detection capabilities for the considered use case (taking into account various contributing factors, such as hardware failure rates, bit error rates, number of nodes in the network, repetition rate of messages, the usage of a gateway, potential software faults on the communication channel), as well as appropriate reaction on detected faults (e.g. by revoking repeated messages, determining timed-out communication or reacting on corrupt messages by initiating a safety reaction).

This specification specifies also the functionality, API and the configuration of the CRC routines.

The following routines for CRC calculation are specified:

- CRC8: SAEJ1850
- CRC8H2F: CRC8 0x2F polynomial
- CRC16
- CRC32
- CRC32P4: CRC32 0x1F4ACFB13 polynomial
- CRC64: CRC-64-ECMA

For all routines (CRC8, CRC8H2F, CRC16, CRC32, CRC32P4 and CRC64), the following calculation methods are possible:

- Table based calculation: Fast execution, but larger code size (ROM table)
- Runtime calculation: Slower execution, but small code size (no ROM table)

- Hardware supported CRC calculation (device specific): Fast execution, less CPU time

All routines are re-entrant and can be used by multiple applications at the same time. Hardware supported CRC calculation may be supported by some devices in the future.

2 Acronyms and Abbreviations

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

Abbreviation / Acronym:	**Description:**
Data ID	An identifier that uniquely identifies the message / data element / data.
Repetition	The same message was received more than once
Loss	A message was not received
Delay	A message was received later than expected
Insertion	Unexpected information or an extra message was inserted
Masquerade	non-authentic information is accepted as authentic information by a receiver.
Incorrect addressing	information is accepted from an incorrect sender or by an incorrect receiver.
Corruption	A communication fault, which changes information
Asymmetric information	Receivers do receive different information from the same sender
Subset	Information from a sender received by only a subset of the receivers
Blocking	Blocking access to a communication channel

Table 2.1: table:acronyms

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_TR_Glossary

[2] Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary

[3] Specification of SW-C End-to-End Communication Protection Library
AUTOSAR_SWS_E2ELibrary

3.2 Related specification

1. SAE-J1850 8-bit CRC
2. CCITT-FALSE 16-bit CRC. Refer to:

   ITU-T Recommendation X.25 (1096) (Previously „CCITT Recommendation")
   SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATION
   Public data networks - Interfaces
   Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit

   Section 2.2.7.4 „Frame Check Sequence (FCS) field" and Appendix I „Examples of data link layer transmitted bit patterns by the DCE and the DTE"
   `http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.25-199610-I!!PDF-E&type=items`
3. IEEE 802.3 Ethernet 32-bit CRC
4. "32-Bit Cyclic Redundancy Codes for Internet Applications" [Koopman 2002]
5. Collection and evaluation of CRC polynomials by Philip Koopman, Carnegie Mellon University `https://users.ece.cmu.edu/~koopman/crc/`

4 Constraints and assumptions

4.1 Limitations

E2E communication protection is limited to periodic or semi-periodic data communication paradigm, where the receiver (subscriber) has an expectancy on the regular reception of data and in case of communication loss/timeout or error, it performs an error handling.

Data communication is called sender/receiver in Classic Platform, and it is called event communication in Adaptive Platform. Note that the word event is a bit confusing as a periodic communication is required.

This means, not all protection methods are supported for client-server (methods) as well as non-periodic data communication.

4.2 Applicability to car domains

The E2E supervision is applicable for the realization of safety-related automotive systems implemented by various SW-Cs distributed across different ECUs in a vehicle, interacting via communication links. The Supervision may also be used for intra-ECU communication (e.g. between memory partitions, processes, OSes/VMs in the same micorcontroller, between CPU cores or microcontrollers).

4.3 Background information concerning functional safety

This chapter provides some safety background information considered during the design of the E2E supervision, including the fault model for communication and definition of sources of faults.

4.3.1 Functional safety and communication

With respect to the exchange of information in safety-related systems, the mechanisms for the in-time detection of causes for faults, or effects of faults as listed below, can be used to design suitable safety concepts, e.g. to achieve freedom from interference between system elements sharing a common communication infrastructure (see ISO 26262-6:2011, annex D.2.4):

- repetition of information;
- loss of information;
- delay of information;
- insertion of information;
- masquerade or incorrect addressing of information;
- incorrect sequence of information;
- corruption of information;
- asymmetric information sent from a sender to multiple receivers;
- information from a sender received by only a subset of the receivers;

- blocking access to a communication channel.

4.3.2 Sources of faults in E2E communication

E2E communication protection aims to detect and mitigate the causes for or effects of communication faults arising from:

1. (systematic) software faults,
2. (random) hardware faults,
3. transient faults due to external influences.

These three sources are described in the sections below.

4.3.2.1 Software faults

Software like, communication stack modules and RTE, may contain faults, which are of a systematic nature.

Systematic faults may occur in any stage of the system's life cycle including specification, design, manufacturing, operation, and maintenance, and they will always appear when the circumstances (e.g. trigger conditions for the root-cause) are the same. The consequences of software faults can be failures of the communication, like interruption of sending of data, overrun of the receiver (e.g. buffer overflow), or underrun of the sender (e.g. buffer empty). To prevent (or to handle) resulting failures the appropriate technical measures to detect and handle such faults (e.g. program flow monitoring or E2E supervision) have to be considered.

4.3.2.2 Random hardware faults

A random hardware fault is typically the result of electrical overload, degradation, aging or exposure to external influences (e.g. environmental stress) of hardware parts. A random hardware fault cannot be avoided completely, but its probability can be evaluated and appropriate technical measures can be implemented (e.g. diagnostics).

4.3.2.3 External influences, environmental stress

This includes influences like EMI, ESD, humidity, corrosion, temperature or mechanical stress (e.g. vibration).

4.3.3 Communication faults

Relevant faults related to the exchange of information are listed in this section.

4.3.3.1 Repetition of information

A type of communication fault, where information is received more than once.

4.3.3.2 Loss of information

A type of communication fault, where information or parts of information are removed from a stream of transmitted information.

4.3.3.3 Delay of information

A type of communication fault, where information is received later than expected.

4.3.3.4 Insertion of information

A type of communication fault, where additional information is inserted into a stream of transmitted information.

4.3.3.5 Masquerading

A type of communication fault, where non-authentic information is accepted as authentic information by a receiver.

4.3.3.6 Incorrect addressing

A type of communication fault, where information is accepted from an incorrect sender or by an incorrect receiver.

4.3.3.7 Incorrect sequence of information

A type of communication fault, which modifies the sequence of the information in a stream of transmitted information.

4.3.3.8 Corruption of information

A type of communication fault, which changes information.

4.3.3.9 Asymmetric information sent from a sender to multiple receivers

A type of communication fault, where receivers do receive different information from the same sender.

4.3.3.10 Information from a sender received by only a subset of the receivers

A type of communication fault, where some receivers do not receive the information.

4.3.3.11 Blocking access to a communication channel

A type of communication fault, where the access to a communication channel is blocked.

5 Requirements Tracing

Requirement	Description	Satisfied by
[RS_E2E_08528]	E2E protocol shall provide different E2E profiles	[PRS_E2E_00012] [PRS_E2E_00075] [PRS_E2E_00076] [PRS_E2E_00085] [PRS_E2E_00117] [PRS_E2E_00118] [PRS_E2E_00119] [PRS_E2E_00120] [PRS_E2E_00121] [PRS_E2E_00122] [PRS_E2E_00123] [PRS_E2E_00124] [PRS_E2E_00125] [PRS_E2E_00126] [PRS_E2E_00127] [PRS_E2E_00128] [PRS_E2E_00129] [PRS_E2E_00130] [PRS_E2E_00132] [PRS_E2E_00133] [PRS_E2E_00134] [PRS_E2E_00135] [PRS_E2E_00136] [PRS_E2E_00137] [PRS_E2E_00138] [PRS_E2E_00139] [PRS_E2E_00140] [PRS_E2E_00141] [PRS_E2E_00142] [PRS_E2E_00143] [PRS_E2E_00145] [PRS_E2E_00146] [PRS_E2E_00147] [PRS_E2E_00148] [PRS_E2E_00149] [PRS_E2E_00150] [PRS_E2E_00151] [PRS_E2E_00163] [PRS_E2E_00169] [PRS_E2E_00190] [PRS_E2E_00195] [PRS_E2E_00196] [PRS_E2E_00217] [PRS_E2E_00221] [PRS_E2E_00227] [PRS_E2E_00228] [PRS_E2E_00298] [PRS_E2E_00299]

Requirement	Description	Satisfied by
		[PRS_E2E_00300] [PRS_E2E_00301] [PRS_E2E_00306] [PRS_E2E_00307] [PRS_E2E_00329] [PRS_E2E_00400] [PRS_E2E_00420] [PRS_E2E_00484] [PRS_E2E_00508] [PRS_E2E_00526] [PRS_E2E_00540] [PRS_E2E_00541] [PRS_E2E_00584] [PRS_E2E_00585] [PRS_E2E_00586] [PRS_E2E_00587] [PRS_E2E_00588] [PRS_E2E_00589] [PRS_E2E_00590] [PRS_E2E_00591] [PRS_E2E_00592] [PRS_E2E_00593] [PRS_E2E_00594] [PRS_E2E_00595] [PRS_E2E_00596] [PRS_E2E_00597] [PRS_E2E_00598] [PRS_E2E_00599] [PRS_E2E_00600] [PRS_E2E_00601] [PRS_E2E_00602] [PRS_E2E_00603] [PRS_E2E_00604] [PRS_E2E_00605] [PRS_E2E_00608] [PRS_E2E_00609] [PRS_E2E_00610] [PRS_E2E_00611] [PRS_E2E_00612] [PRS_E2E_00613] [PRS_E2E_00614] [PRS_E2E_UC_00051] [PRS_E2E_UC_00061] [PRS_E2E_UC_00237] [PRS_E2E_UC_00316] [PRS_E2E_UC_00351] [PRS_E2E_UC_00466] [PRS_E2E_USE_00235] [PRS_E2E_USE_00325]

Requirement	Description	Satisfied by
[RS_E2E_08529]	Each E2E profile shall use an appropriate subset of specific protection mechanisms	[PRS_E2E_00070] [PRS_E2E_00218] [PRS_E2E_00219] [PRS_E2E_00372] [PRS_E2E_00394] [PRS_E2E_00479] [PRS_E2E_00480] [PRS_E2E_00503] [PRS_E2E_00522]
[RS_E2E_08530]	Each E2E profile shall have a unique Profile ID, define precisely a set of mechanisms and its behavior in a semi-formal way	[PRS_E2E_00196] [PRS_E2E_00218] [PRS_E2E_00219] [PRS_E2E_00372] [PRS_E2E_00394] [PRS_E2E_00479] [PRS_E2E_00480] [PRS_E2E_00503] [PRS_E2E_00522]
[RS_E2E_08533]	CRC used in a E2E profile shall be different than the CRC used by the underlying physical communication protocol	[PRS_E2E_00070] [PRS_E2E_00218] [PRS_E2E_00219] [PRS_E2E_00372] [PRS_E2E_00394] [PRS_E2E_00479] [PRS_E2E_00480] [PRS_E2E_00503] [PRS_E2E_00522]
[RS_E2E_08534]	E2E Protocol shall provide E2E Check status to the application	[PRS_E2E_00318] [PRS_E2E_00319] [PRS_E2E_00320] [PRS_E2E_00322] [PRS_E2E_00323] [PRS_E2E_00324] [PRS_E2E_USE_00321]
[RS_E2E_08536]	Either SW-C or E2E Library shall compute the intermediate CRC over application data element	[PRS_E2E_00082] [PRS_E2E_00126] [PRS_E2E_00134] [PRS_E2E_00330] [PRS_E2E_00401] [PRS_E2E_00421] [PRS_E2E_00485] [PRS_E2E_00527] [PRS_E2E_00613]

Requirement	Description	Satisfied by
[RS_E2E_08539]	An E2E protection mechanism for inter-ECU communication of short to large data shall be provided	[PRS_E2E_00326] [PRS_E2E_00329] [PRS_E2E_00345] [PRS_E2E_00354] [PRS_E2E_00355] [PRS_E2E_00356] [PRS_E2E_00357] [PRS_E2E_00358] [PRS_E2E_00359] [PRS_E2E_00360] [PRS_E2E_00361] [PRS_E2E_00362] [PRS_E2E_00363] [PRS_E2E_00364] [PRS_E2E_00365] [PRS_E2E_00366] [PRS_E2E_00367] [PRS_E2E_00368] [PRS_E2E_00369] [PRS_E2E_00375] [PRS_E2E_00376] [PRS_E2E_00397] [PRS_E2E_00399] [PRS_E2E_00400] [PRS_E2E_00401] [PRS_E2E_00403] [PRS_E2E_00404] [PRS_E2E_00405] [PRS_E2E_00406] [PRS_E2E_00407] [PRS_E2E_00409] [PRS_E2E_00411] [PRS_E2E_00412] [PRS_E2E_00413] [PRS_E2E_00414] [PRS_E2E_00416] [PRS_E2E_00417] [PRS_E2E_00419] [PRS_E2E_00420] [PRS_E2E_00421] [PRS_E2E_00423] [PRS_E2E_00424] [PRS_E2E_00425] [PRS_E2E_00426] [PRS_E2E_00427] [PRS_E2E_00428] [PRS_E2E_00429] [PRS_E2E_00430]

Requirement	Description	Satisfied by
		[PRS_E2E_00431] [PRS_E2E_00432] [PRS_E2E_00433] [PRS_E2E_00434] [PRS_E2E_00436] [PRS_E2E_00466] [PRS_E2E_00467] [PRS_E2E_00469] [PRS_E2E_00470] [PRS_E2E_00478] [PRS_E2E_00481] [PRS_E2E_00482] [PRS_E2E_00483] [PRS_E2E_00484] [PRS_E2E_00486] [PRS_E2E_00487] [PRS_E2E_00489] [PRS_E2E_00490] [PRS_E2E_00491] [PRS_E2E_00492] [PRS_E2E_00493] [PRS_E2E_00494] [PRS_E2E_00495] [PRS_E2E_00496] [PRS_E2E_00497] [PRS_E2E_00498] [PRS_E2E_00499] [PRS_E2E_00500] [PRS_E2E_00501] [PRS_E2E_00504] [PRS_E2E_00505] [PRS_E2E_00506] [PRS_E2E_00508] [PRS_E2E_00509] [PRS_E2E_00510] [PRS_E2E_00511] [PRS_E2E_00512] [PRS_E2E_00513] [PRS_E2E_00514] [PRS_E2E_00515] [PRS_E2E_00516] [PRS_E2E_00517] [PRS_E2E_00518] [PRS_E2E_00519] [PRS_E2E_00521] [PRS_E2E_00523] [PRS_E2E_00524] [PRS_E2E_00525]

Requirement	Description	Satisfied by
		[PRS_E2E_00526] [PRS_E2E_00527] [PRS_E2E_00528] [PRS_E2E_00529] [PRS_E2E_00530] [PRS_E2E_00531] [PRS_E2E_00532] [PRS_E2E_00533] [PRS_E2E_00534] [PRS_E2E_00535] [PRS_E2E_00536] [PRS_E2E_00537] [PRS_E2E_00539] [PRS_E2E_00582] [PRS_E2E_00583] [PRS_E2E_00607] [PRS_E2E_00615] [PRS_E2E_00616] [PRS_E2E_00617] [PRS_E2E_00618] [PRS_E2E_00619] [PRS_E2E_00620] [PRS_E2E_00621] [PRS_E2E_00632] [PRS_E2E_00639] [PRS_E2E_0507] [PRS_E2E_UC_00236] [PRS_E2E_UC_00327] [PRS_E2E_UC_00463] [PRS_E2E_UC_00464]
[RS_E2E_08540]	E2E protocol shall support protected periodic/mixed periodic communication	[PRS_E2E_USE_00236] [PRS_E2E_USE_00237]
[RS_E2E_08541]	E2E protocol shall support protected non-periodic communication	[PRS_E2E_USE_00606]
[RS_E2E_08639]	No description	[PRS_E2E_00622] [PRS_E2E_00623] [PRS_E2E_00624] [PRS_E2E_00625]
[RS_E2E_08739]	No description	[PRS_E2E_00626] [PRS_E2E_00627] [PRS_E2E_00628] [PRS_E2E_00629] [PRS_E2E_00630] [PRS_E2E_00631] [PRS_E2E_00633] [PRS_E2E_00634] [PRS_E2E_00635] [PRS_E2E_00636] [PRS_E2E_00637] [PRS_E2E_00638]

6 Functional specification

This chapter contains the specification of the internal functional behavior of the E2E supervision. For general introduction of the E2E supervision, see first chapter 1.

6.1 Overview of communication protection

An important aspect of a communication protection mechanism is its standardization and its flexibility for different purposes. This is resolved by having a set of E2E Profiles, that define a combination of protection mechanisms, a message format, and a set of configuration parameters.

Moreover, some E2E Profiles have standard E2E variants. An E2E variant is simply a set of configuration options to be used with a given E2E Profile. For example, in E2E Profile 1, the positions of CRC and counter are configurable. The E2E variant 1A requires that CRC starts at bit 0 and counter starts at bit 8.

E2E communication protection works as follows:

- Sender: addition of control fields like CRC or counter to the transmitted data;
- Receiver: evaluation of the control fields from the received data, calculation of control fields (e.g. CRC calculation on the received data), comparison of calculated control fields with an expected/received content.

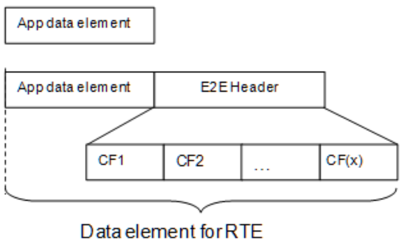

Figure 6.1: Safety protocol concept (with exemplary location of the E2E header)

Each E2E Profile has a specific set of control fields with a specific functional behavior and with specific properties for the detection of communication faults.

6.2 Overview of E2E Profiles

The E2E Profiles provide a consistent set of data protection mechanisms, designed to protecting against the faults considered in the fault model.

Each E2E Profile provides an alternative way to protect the communication, by means of different algorithms. However, E2E Profile have similar interfaces and behavior.

[PRS_E2E_00221] ⌈Each E2E Profile shall use a subset of the following data protection mechanisms:

1. A CRC, provided by CRC Supervision;
2. A Sequence Counter incremented at every transmission request, the value is checked at receiver side for correct incrementation;
3. An Alive Counter incremented at every transmission request, the value checked at the receiver side if it changes at all, but correct incrementation is not checked;
4. A specific ID for every port data element sent over a port or a specific ID for every message-group (global to system, where the system may contain potentially several ECUs);
5. Timeout detection:
   (a) Receiver communication timeout.
   (b) Sender acknowledgement timeout.

Depending on the used communication and network stack, appropriate subsets of these mechanisms are defined as E2E communication profiles.

⌋*(RS_E2E_08528)*

Some of the above mechanisms are implemented in RTE, COM, and/or communication stacks. However, to reduce or avoid an allocation of safety requirements to these modules, they are not considered: E2E Supervision provides all mechanisms internally (only with usage of CRC Supervision).

The E2E Profiles can be used for both inter and intra ECU communication. The E2E Profiles were specified for specific communication infrastructure, such as CAN, CAN FD, FlexRay, LIN, Ethernet.

Depending on the system, the user selects which E2E Profile is to be used, from the E2E Profiles provided by E2E Supervision.

[PRS_E2E_00217] ⌈The implementation of the E2E Supervision shall provide at least one of the E2E Profiles.

⌋*(RS_E2E_08528)*

6.2.1 Error detection

[PRS_E2E_00012] ⌈The internal Supervision mechanisms error detection and reporting shall be implemented according to the pre-defined E2E Profiles specified in this document.⌋*(RS_E2E_08528)*

6.3 Specification of E2E Profile 1 (Only for CP)

[PRS_E2E_00218] ⌈Profile 1 shall provide the following mechanisms: Counter, Timeout monitoring, Data ID, CRC (see Table 6.1).⌋*(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Mechanism	Description
Counter	4bit (explicitly sent) representing numbers from 0 to 14 incremented on every send request. Both Alive Counter and Sequence Counter mechanisms are provided by E2E Profile 1, evaluating the same 4 bits.
Timeout monitoring	Timeout is determined by E2E Supervision by means of evaluation of the Counter, by a nonblocking read at the receiver. Timeout is reported by E2E Supervision to the caller by means of the status flags in E2E_P01CheckStatusType.
Data ID	16 bit, unique number, included in the CRC calculation. For dataIdMode equal to 0, 1 or 2, the Data ID is not transmitted, but included in the CRC computation (implicit transmission). For dataIdMode equal to 3: • the high nibble of high byte of DataID is not used (it is 0x0), as the DataID is limited to 12 bits, • the low nibble of high byte of DataID is transmitted explicitly and covered by CRC calculation when computing the CRC over Data. • the low byte is not transmitted, but it is included in the CRC computation as start value (implicit transmission, like for dataIDMode equal to 0, 1 or 2) .
CRC	CRC-8-SAE J1850 - 0x1D (x8 + x4 + x3 + x2 + 1), but with different start and XOR values (both start value and XOR value are 0x00). This CRC is provided by CRC Supervision. Starting with AUTOSAR R4.0, the SAE8 CRC function of the CRC Supervision uses 0xFF as start value and XOR value. To compensate a different behavior of the CRC Supervision, the E2E Supervision applies additional XOR 0xFF operations starting with R4.0, to come up with 0x00 as start value and XOR value. Note: This CRC polynomial is different from the CRC-polynomials used by FlexRay, CAN and LIN.

Table 6.1: E2E Profile 1 mechanisms

The E2E mechanisms can detect the following faults or effects of faults:

E2E Mechanism	Detected communication faults
Counter	Repetition, Loss, insertion, incorrect sequence, blocking

Transmission on a regular basis and timeout monitoring using E2E-Supervision [1]	Loss, delay, blocking
Data ID + CRC	Masquerade and incorrect addressing, insertion
CRC	Corruption, Asymmetric information [2]

Table 6.2: Detectable communication faults using Profile 1

[PRS_E2E_00070] ⌈

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 equal to 0x00.

⌋*(RS_E2E_08529, RS_E2E_08533)*

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[2].

6.3.1 Data Layout

In the E2E Profile 1, the layout is in general free to be defined by the user, as long as the basic limitations of signal alignment are followed:

- signals that have length < 8 bits should be allocated to one byte of an I-PDU, i.e. they should not span over two bytes.
- signals that have length >= 8 bits should start or finish at the byte limit of an message.

However, predefined E2E Profile 1 variants define specific data layouts regarding the protocol data fields, see subsection 6.3.6.

6.3.2 Counter

In E2E Profile 1, the counter is initialized, incremented, reset and checked by E2E profile.

[PRS_E2E_00075] ⌈In E2E Profile 1, on the sender side, for the first transmission request of a data element the counter shall be initialized with 0 and shall be incremented by 1 for every subsequent send request (from sender SW-C). When the counter reaches the value 14 (0xE), then it shall restart with 0 for the next send request (i.e. value 0xF shall be skipped). All these actions shall be executed by E2E Supervision.

⌋*(RS_E2E_08528)*

[1] Implementation by sender and receiver, which are using E2E-Supervision
[2] for a set of data protected by same CRC

[PRS_E2E_00076] ⌈In E2E Profile 1, on the receiver side, by evaluating the counter of received data against the counter of previously received data, the following shall be detected by the E2E Supervision: (1) no new data has arrived since last invocation of E2E Supervision check function, (2) no new data has arrived since receiver start, (3) the data is repeated (4) counter is incremented by one (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).

⌋*(RS_E2E_08528)*

Case 3 corresponds to the failed alive counter check, and case 6 correspond to failed sequence counter check.

The above requirements are specified in more details by the UML diagrams in the following document sections.

6.3.3 Data ID

The unique Data IDs are to verify the identity of each transmitted safety-related data element.

[PRS_E2E_00163] ⌈There shall be following four inclusion modes for the two-byte Data ID into the calculation of the one-byte CRC:

1. E2E_P01_DATAID_BOTH: both two bytes (double ID configuration) are included in the CRC, first low byte and then high byte (see variant 1A - PRS_E2EProtocol_00227) or
2. E2E_P01_DATAID_ALT: depending on parity of the counter (alternating ID configuration) the high and the low byte is included (see variant 1B - PRS_E2EProtocol_00228). For even counter values the low byte is included and for odd counter values the high byte is included.
3. E2E_P01_DATAID_LOW: only the low byte is included and high byte is never used. This equals to the situation if the Data IDs (in a given application) are only 8 bits.
4. E2E_P01_DATAID_NIBBLE:
    - the high nibble of high byte of DataID is not used (it is 0x0), as the DataID is limited to 12 bits,
    - the low nibble of high byte of DataID is transmitted explicitly and covered by CRC calculation when computing the CRC over Data.
    - the low byte is not transmitted, but it is included in the CRC computation as start value (implicit transmission, like for the inclusion modes _BOTH, _ALT and _LOW)

⌋*(RS_E2E_08528)*

[PRS_E2E_00085] ⌈In E2E Profile 1, with E2E_P01DataIDMode equal to E2E_P01_DATAID_BOTH or E2E_P01_DATAID_ALT the length of the Data ID shall be 16 bits (i.e. 2 byte).⌋*(RS_E2E_08528)*

[PRS_E2E_00169] ⌈In E2E Profile 1, with E2E_P01DataIDMode equal to E2E_P01_DATAID_LOW, the high byte of Data ID shall be set to 0x00.⌋*(RS_E2E_08528)*

The above requirement means that when high byte of Data ID is unused, it is set to 0x00.

[PRS_E2E_00306] ⌈In E2E Profile 1, with E2E_P01DataIDMode equal to E2E_P01_DATAID_NIBBLE, the high nibble of the high byte shall be 0x0.⌋*(RS_E2E_08528)*

The above requirement means that the address space with E2E_P01_DATAID_NIBBLE is limited to 12 bits.

In case of usage of E2E Supervision for protecting data elements, due to multiplicity of communication (1:1 or 1:N), a receiver of a data element receives it only from one sender. In case of usage of E2E Supervision for protecting messages, because each message has a unique Data ID, the receiver COM of a message receives it only from one sender COM. As a result (regardless if the protection is at data element level or at messages), the receiver expects data with only one Data ID. The receiver uses the expected Data ID to calculate the CRC. If CRC matches, it means that the Data ID used by the sender and expected Data ID used by the receiver are the same.

6.3.4 CRC calculation

E2E Profile 1 uses CRC-8-SAE J1850, but using different start and XOR values. This checksum is already provided by AUTOSAR CRC Supervision, which typically is quite efficient and may use hardware support.

[PRS_E2E_00190] ⌈E2E Profile 1 shall use the Crc_CalculateCRC8 () function of the SWS CRC Supervision for calculating CRC checksums.⌋*(RS_E2E_08528)*

Note: The CRC used by E2E Profile 1 is different than the CRCs used by FlexRay and CAN and is provided by different software modules (FlexRay and CAN CRCs are provided by hardware support in Communication Controllers, not by CRC Supervision).

The CRC calculation is illustrated by the following two examples.

For standard variant 1A:

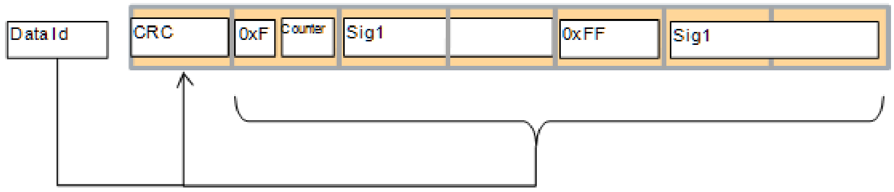

Figure 6.2: E2E Profile 1 variant 1A CRC calculation example

For standard variant 1C:

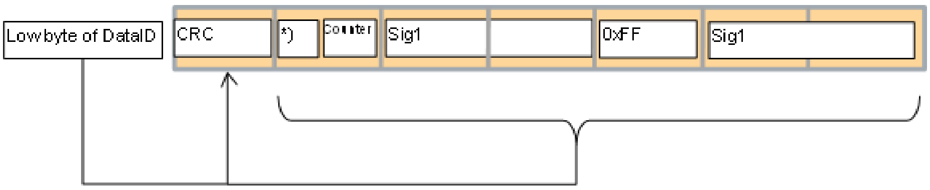

Figure 6.3: E2E Profile 1 variant 1C CRC calculation example

The Data ID can be encoded in CRC in different ways, see [PRS_E2E_00163].

[PRS_E2E_00082] ⌈In E2E Profile 1, the CRC is calculated over:

1. First over the one or two bytes of the Data ID (depending on Data ID configuration), and
2. then over all transmitted bytes of a safety-related complex data element/signal group (except the CRC byte).

⌋*(RS_E2E_08536)*

6.3.5 Timeout detection

The previously mentioned mechanisms (CRC, counter, Data ID) enable to check the validity of received data element, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for Data Elements or respectively signal groups, but instead if the receiver reads the currently available data (i.e. checks if new data is available). Then, by means of the counter, the receiver can detect loss of communication and timeouts.

The attribute State->Status = E2E_P01STATUS_REPEATED means that there is a repetition (caused either by communication loss, delay or duplication of the previous message). The receiver uses State->Status for detecting communication timeouts.

6.3.6 E2E Profile 1 variants

The E2E Profile 1 has recommended variants. The variants are specific configurations of E2E Profile.

[PRS_E2E_00227] ⌈The E2E Profile variant 1A is defined as follows:

1. CRC is the 0th byte in the signal group (i.e. starts with bit offset 0)
2. Alive counter is located in lowest 4 bits of 1st byte (i.e. starts with bit offset 8)
3. E2E_P01DataIDMode = E2E_P01_DATAID_BOTH
4. SignalIPdu.unusedBitPattern = 0xFF.

⌋*(RS_E2E_08528)*

[PRS_E2E_00228] ⌈The E2E Profile variant 1B is defined as follows:

1. CRC is the 0th byte in the signal group (i.e. starts with bit offset 0)
2. Alive counter is located in lowest 4 bits of 1st byte (i.e. starts with bit offset 8)
3. E2E_P01DataIDMode = E2E_P01_DATAID_ALT
4. SignalIPdu.unusedBitPattern = 0xFF.

⌋*(RS_E2E_08528)*

Below is an example compliant to 1A/1B:

Figure 6.4: E2E Profile 1 example layout (two signal groups protected by E2E in one message)

[PRS_E2E_00307] ⌈The E2E Profile variant 1C is defined as follows:

1. CRC is the 0th byte in the signal group (i.e. starts with bit offset 0)
2. Alive counter is located in lowest 4 bits of 1st byte (i.e. starts with bit offset 8)
3. The Data ID nibble is located in the highest 4 bits of 1st byte (i.e. starts with bit offset 12)
4. E2E_P01DataIDMode = E2E_P01_DATAID_NIBBLE
5. SignalIPdu.unusedBitPattern = 0xFF.

⌋*(RS_E2E_08528)*

6.3.7 E2E_P01Protect

[PRS_E2E_00195] ⌈The function E2E_P01Protect() shall:

1. write the Counter in Data,
2. write DataID nibble in Data, if E2E_P01_DATAID_NIBBLE configuration is used
3. compute the CRC over DataID and Data
4. write CRC in Data
5. increment the Counter (which will be used in the next invocation of E2E_P01Protect()),as specified by Figure 6.5 and Figure 6.6

⌋(*RS_E2E_08528*)

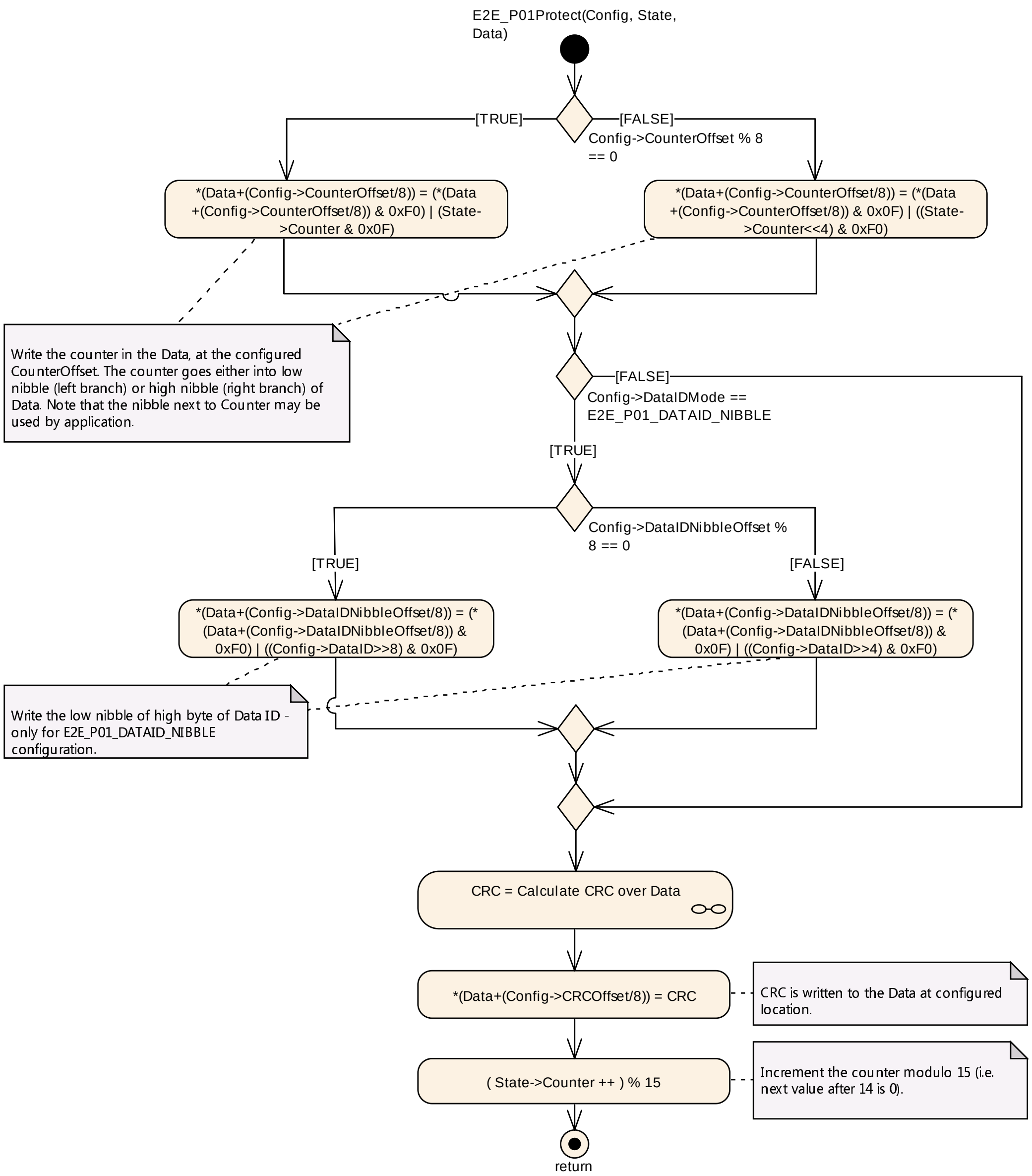

Figure 6.5: E2E_P01Protect()

6.3.8 Calculate CRC

The diagram of the function E2E_P01Protect() (see above chapter), E2E_P01Forward() and E2E_P01Check() (see below chapters) have a sub-diagram specifying the calculation of CRC:

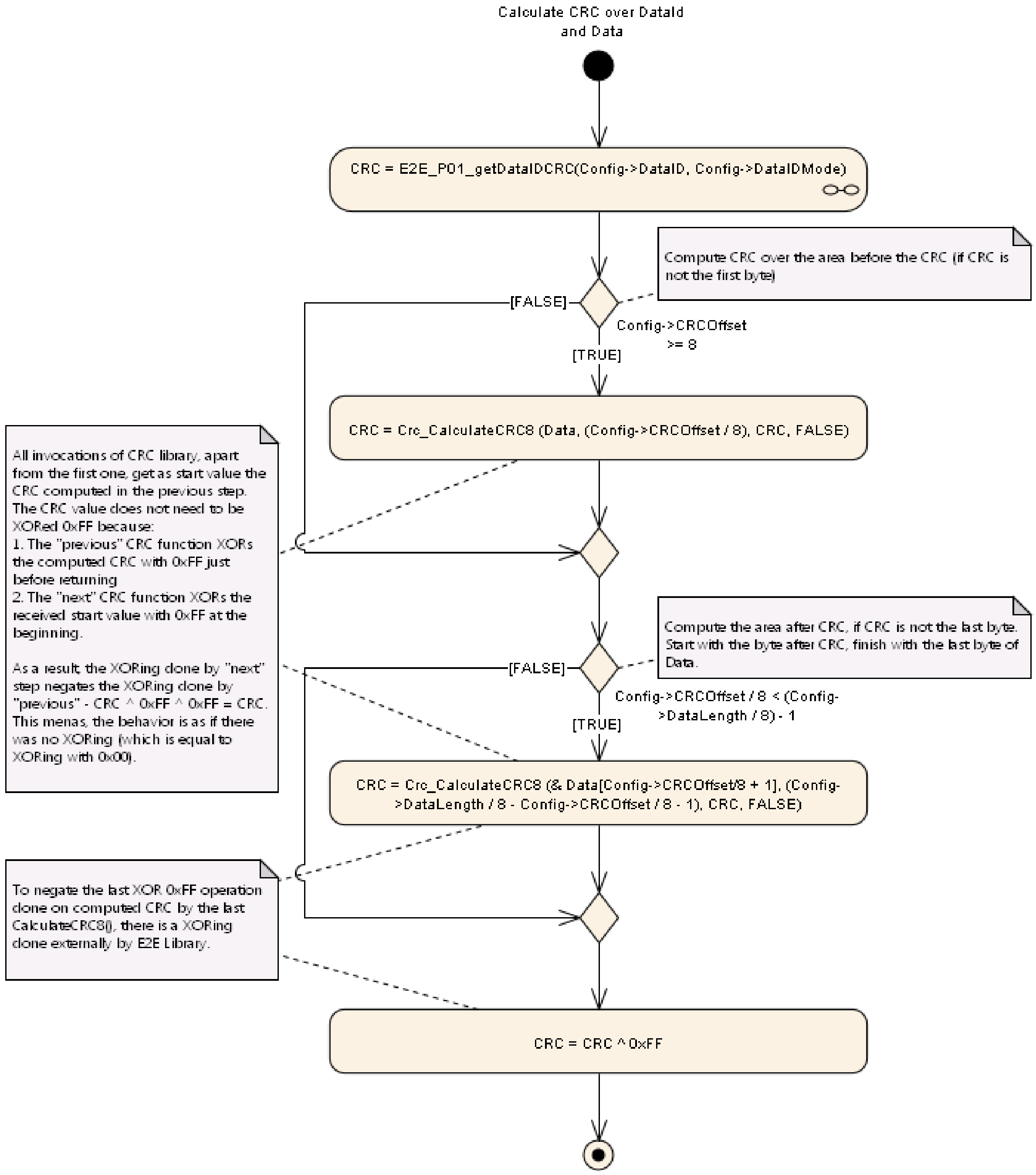

Figure 6.6: Subdiagram „Calculate CRC over Data ID and Data", used by E2E_P01Protect(), E2E_P01Forward() and E2E_P01Check()

The diagram of the function "Calculate CRC over Data ID and Data" has a sub-diagram specifying the calculation of DataID CRC, which is shown by Figure 6.7.

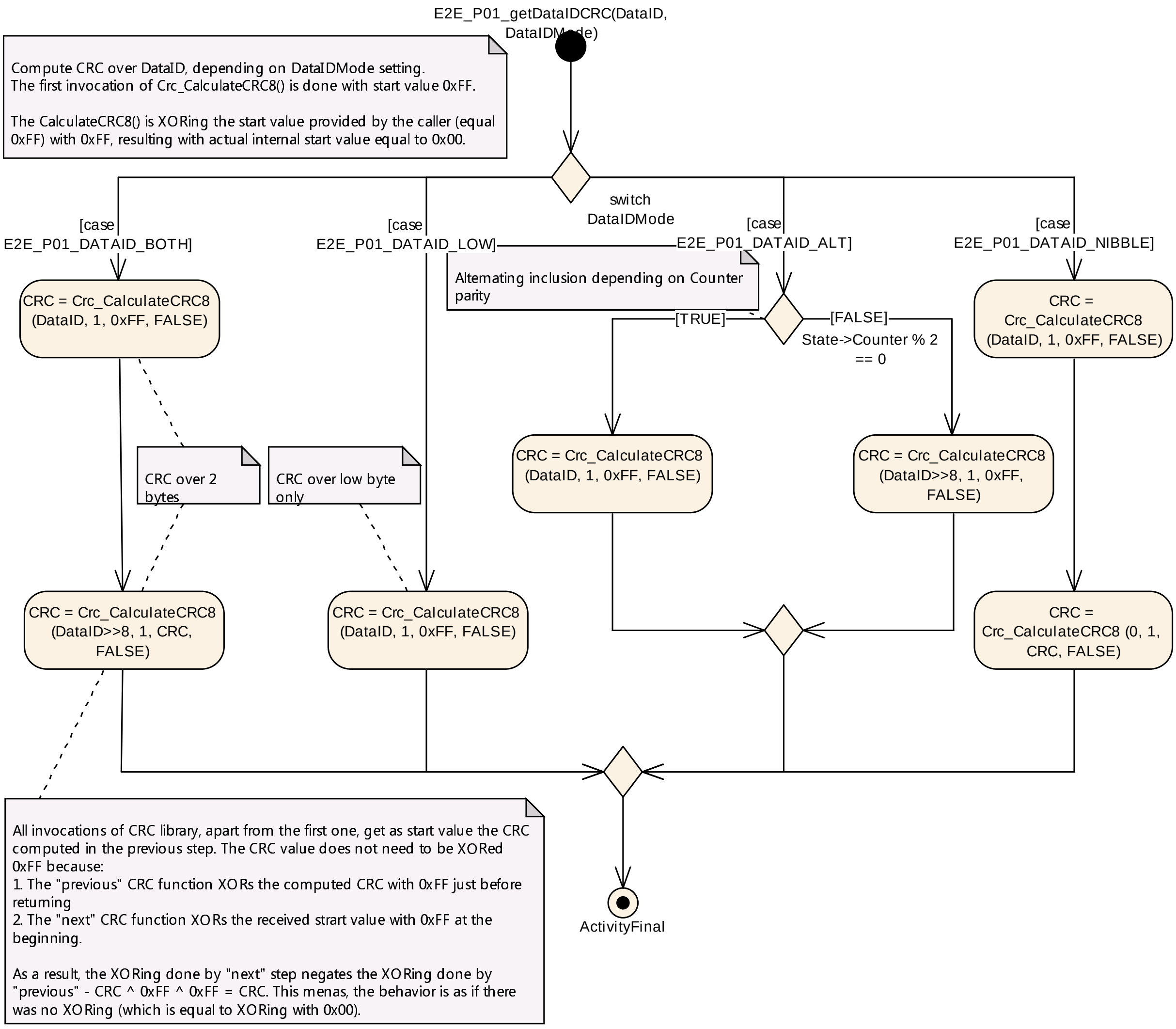

Figure 6.7: Subdiagram "getDataIDCRC", used by E2E_P01Protect() and E2E_P01Check()

It is important to note that the function Crc_CalculateCRC8 of CRC Supervision / CRC routines have changed is functionality since R4.0, i.e. it is different in R3.2 and >=R4.0:

1. There is an additional parameter Crc_IsFirstCall
2. The function has different start value and different XOR values (changed from 0x00 to OxFF).

This results with a different value of computed CRC of a given buffer.

To have the same results of the functions E2E_P01Protect() and E2E_P02Check() in >=R4.0 and R3.2, while using differently functioning CRC Supervision, E2E „compensates" different behavior of the CRC Supervision. This results with different invocation of the CRC Supervision by E2E Supervision Figure 6.6 in >=R4.0 and R3.2. This means Figure 6.6 is different in >=R4.0 and R3.2.

6.3.9 E2E_P01Forward

[PRS_E2E_00608] Draft ⌈The function E2E_P01Forward() shall calculate the e2e header data based on the current value of the IN parameter ForwardStatus.⌋*(RS_-E2E_08528)*

The E2E_P01Forward() has additional requirements to the E2E_P01Protect() since it shall be used to reconstruct an E2E-State on an outgoing message.

[PRS_E2E_00609] Draft ⌈If ForwardStatus equals to E2E_P_OK the function E2E_P01Forward() shall:

1. write the Counter in Data
2. write DataID nibble in Data, if E2E_P01_DATAID_NIBBLE configuration is used
3. compute the CRC over DataID and Data
4. write CRC in Data
5. increment the Counter (which will be used in the next invocation of E2E_P01Forward()), as specified by Figure 6.8 and Figure 6.6

⌋*(RS_E2E_08528)*

[PRS_E2E_00610] Draft ⌈If ForwardStatus equals to E2E_P_REPEATED the function E2E_P01Forward() shall :

1. decrement the Counter
2. write Counter in Data
3. write DataID nibble in Data, if E2E_P01_DATAID_NIBBLE configuration is used
4. compute the CRC over DataID and Data
5. write CRC in Data
6. increment the Counter (which will be used in the next invocation of E2E_P01Forward()), as specified by Figure 6.8 and Figure 6.6

⌋*(RS_E2E_08528)*

[PRS_E2E_00611] Draft ⌈If ForwardStatus equals to E2E_P_WRONGSEQUENCE the function E2E_P01Forward() shall use counter + MaxDeltaCounterInit :

1. calculate Counter = Counter + MaxDeltaCounterInit
2. write the Counter in Data
3. write DataID nibble in Data, if E2E_P01_DATAID_NIBBLE configuration is used
4. compute the CRC over DataID and Data
5. write CRC in Data

6. increment the Counter (which will be used in the next invocation of E2E_P01Forward()), as specified by Figure 6.8 and Figure 6.6

⌋(*RS_E2E_08528*)

[PRS_E2E_00612] Draft ⌈If ForwardStatus equals to E2E_P_ERROR the function E2E_P01Forward() shall use DataID + 1:

1. DataID = DataID+1
2. write the Counter in Data
3. write DataID nibble in Data, if E2E_P01_DATAID_NIBBLE configuration is used
4. compute the CRC over DataID and Data
5. write CRC in Data
6. increment the Counter (which will be used in the next invocation of E2E_P01Forward()), as specified by Figure 6.8 and Figure 6.6

⌋(*RS_E2E_08528*)

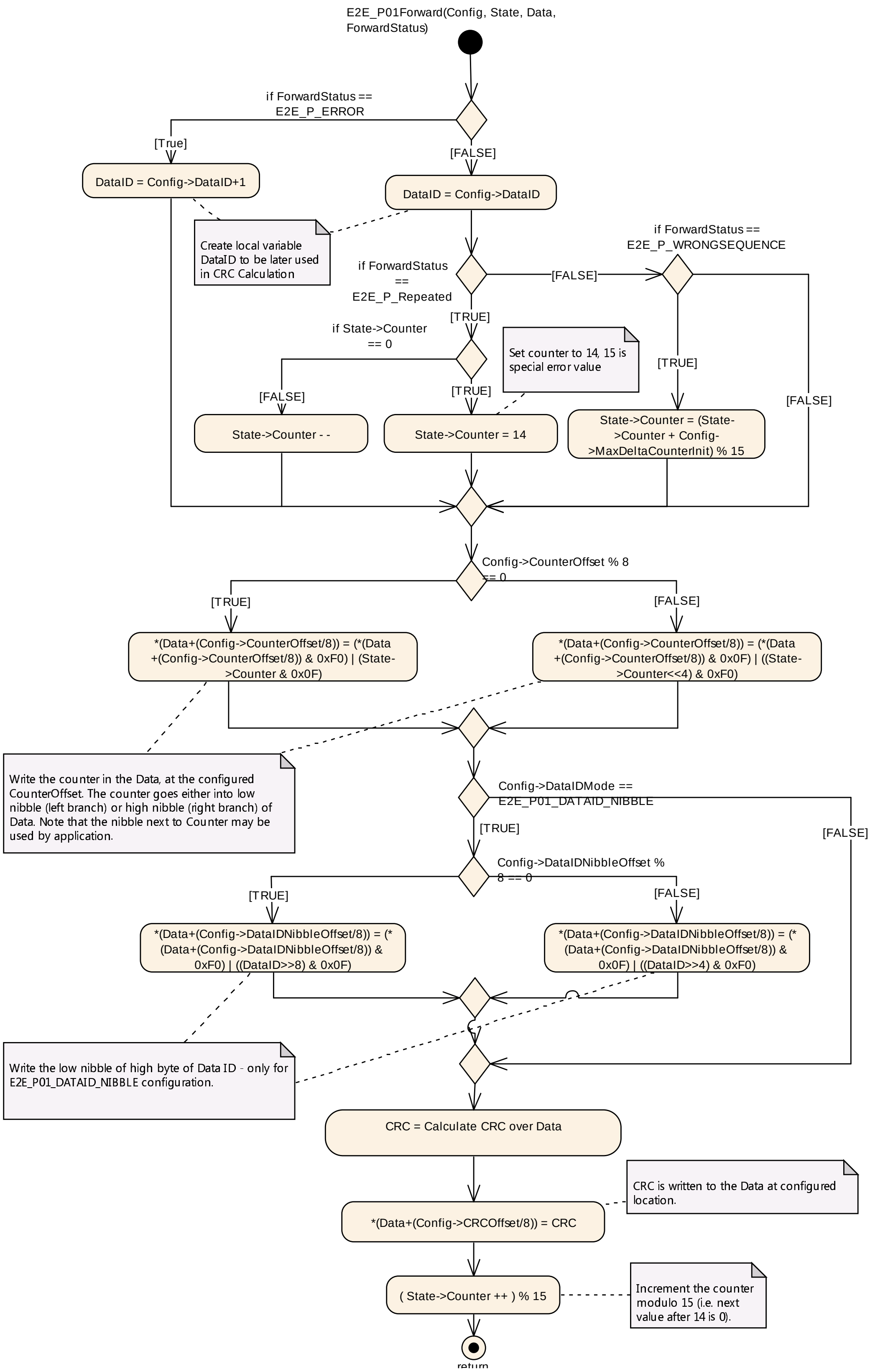

Figure 6.8: E2E_P01Forward()

6.3.10 E2E_P01Check

[PRS_E2E_00196] ⌈The function E2E_P01Check shall

1. Check the CRC
2. Check the Data ID nibble, i.e. compare the expected value with the received value (for E2E_P01_DATAID_NIBBLE configuration only)
3. Check the Counter,
4. determine the check Status,as specified by Figure 6.9 and Figure 6.6.

⌋*(RS_E2E_08528, RS_E2E_08530)*

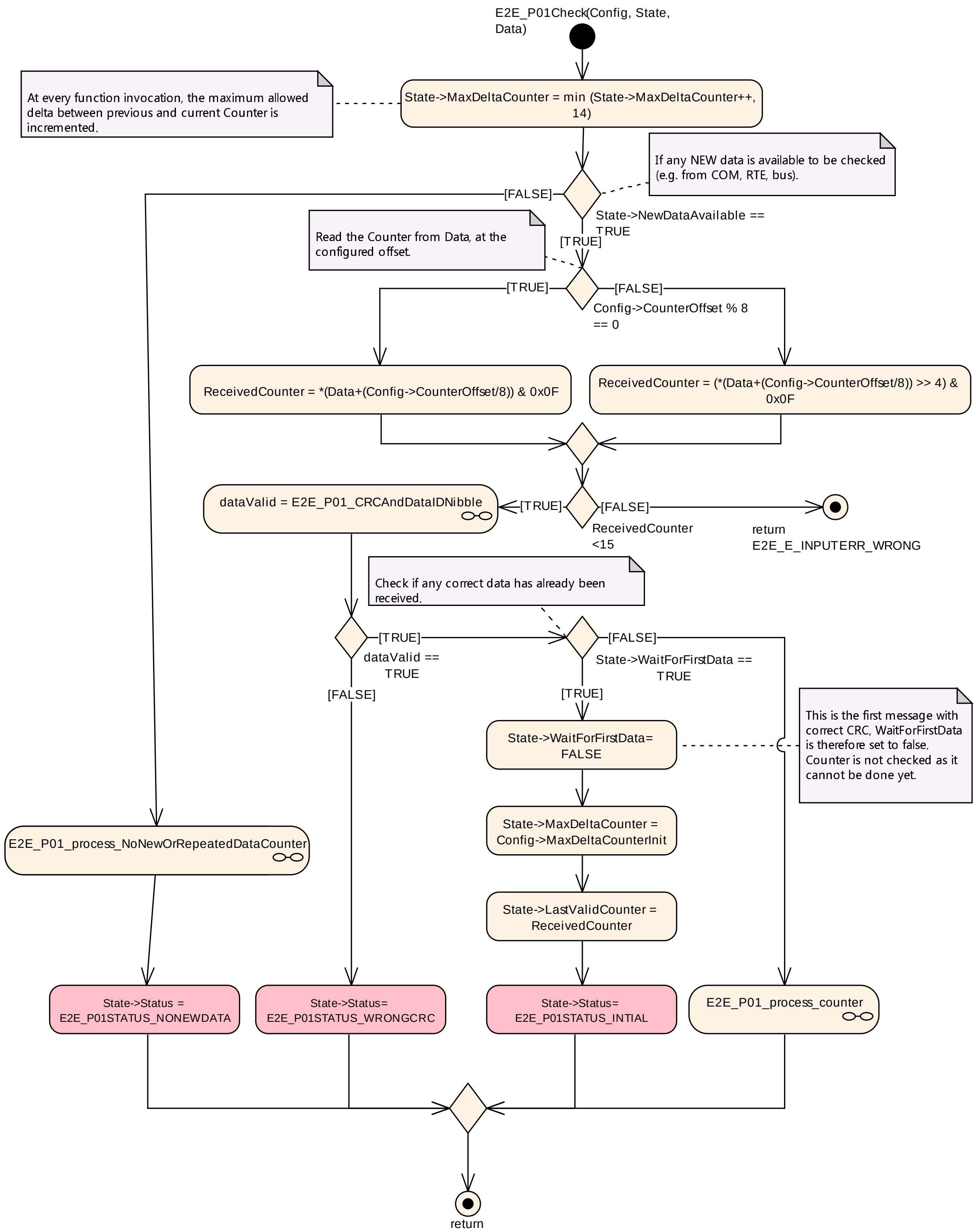

Figure 6.9: E2E_P01Check()

The diagram of the function E2E_P01Check() has a sub-diagram E2E_P01_CRCAndDataIDNibble specifying the calculation of CRC and comparing it with the received CRC, which is shown by Figure 6.6. The subroutines of Figure 6.10 are described in Figure 6.6 and Figure 6.7

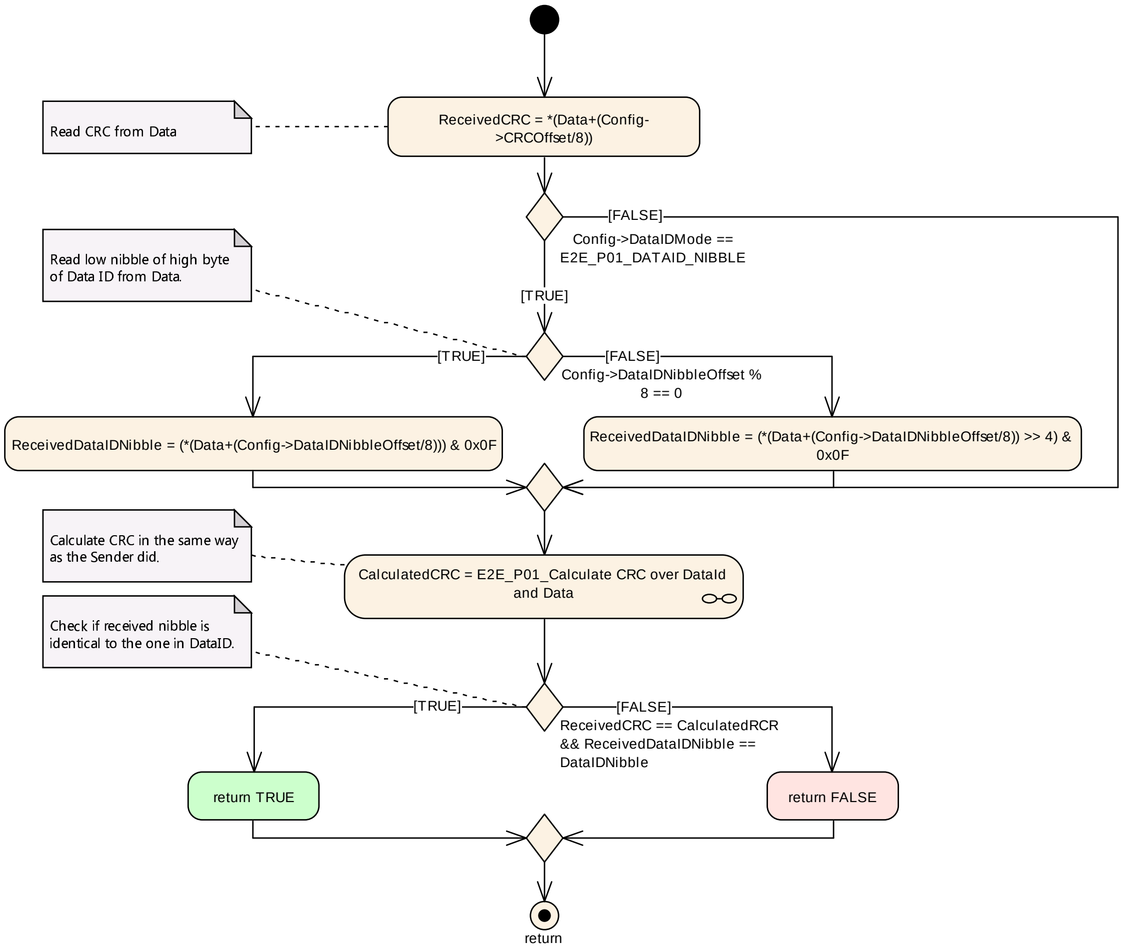

Figure 6.10: E2E Profile Check step "E2E_P01_CRCAndDataIDNibble"

The diagram of the function E2E_P01Check() has a sub-diagram E2E_P01_process_NoNewOrRepeatedDataCounter specifying the evaluation of the different counter states, which is shown in Figure 6.11.

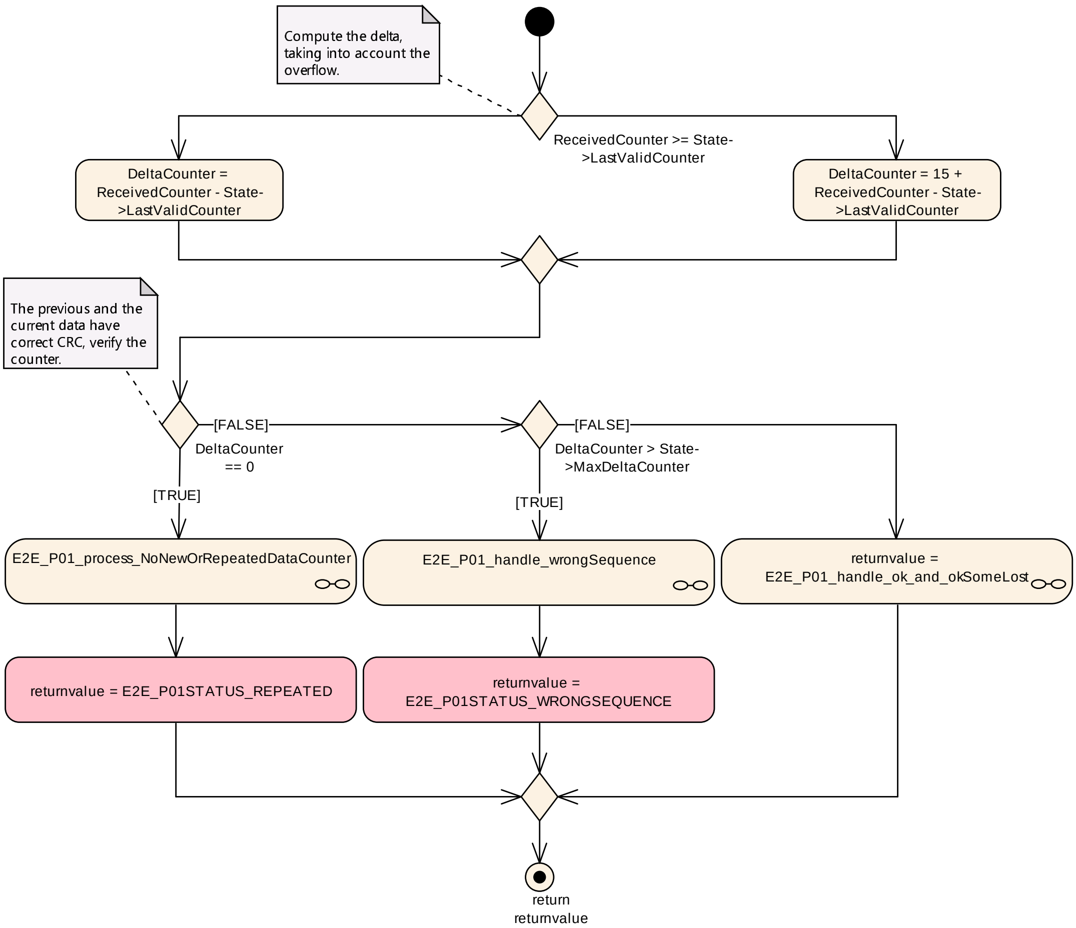

Figure 6.11: E2E Profile Check step "E2E_P01_process_counter"

The diagram of the function E2E_P01Check() and "E2E_P01_process_counter" have a sub-diagram E2E_P01_process_NoNewOrRepeatedDataCounter specifying the handling of receiving a repeated message and receiving no message, which is shown in Figure 6.12.

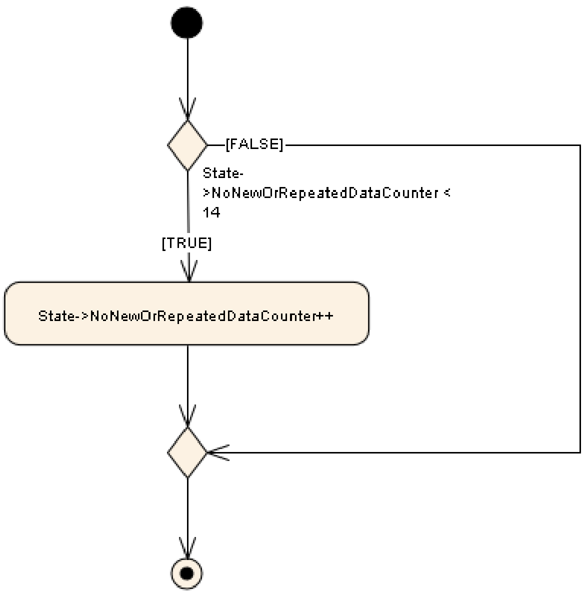

Figure 6.12: E2E Profile Check step "E2E_P01_process_NoNewOrRepeatedDataCounter"

The diagram of the step "E2E_P01_process_counter" has a sub-diagram "E2E_P01_handle_wrongSequence" specifying the handling of receiving a message where the counter exceeded the maximum between two messages, which is shown in Figure 6.13.

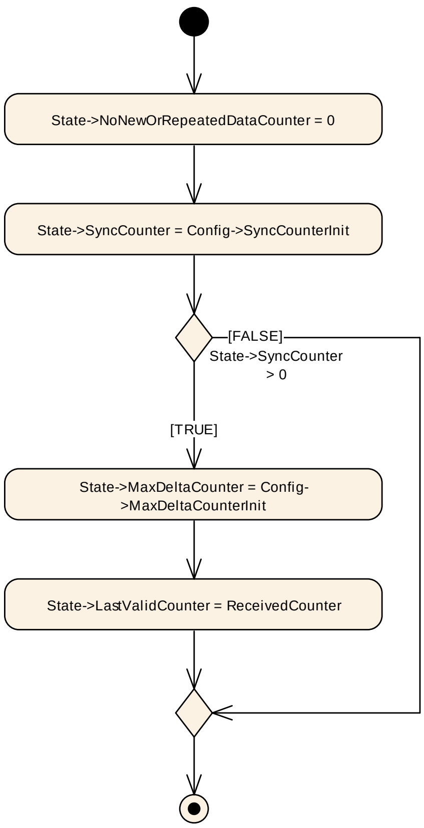

Figure 6.13: E2E Profile Check step "E2E_P01_handle_wrongSequence"

The diagram of the step ”E2E_P01_process_counter” has a sub-diagram ”E2E_P01_handle_ok_and_okSomeLost” specifying the handling of receiving a message of valid messages where the no fault was detected, some messages where lost but this particular is valid or the the profile is synchronizing the counter, which is shown in Figure 6.14.

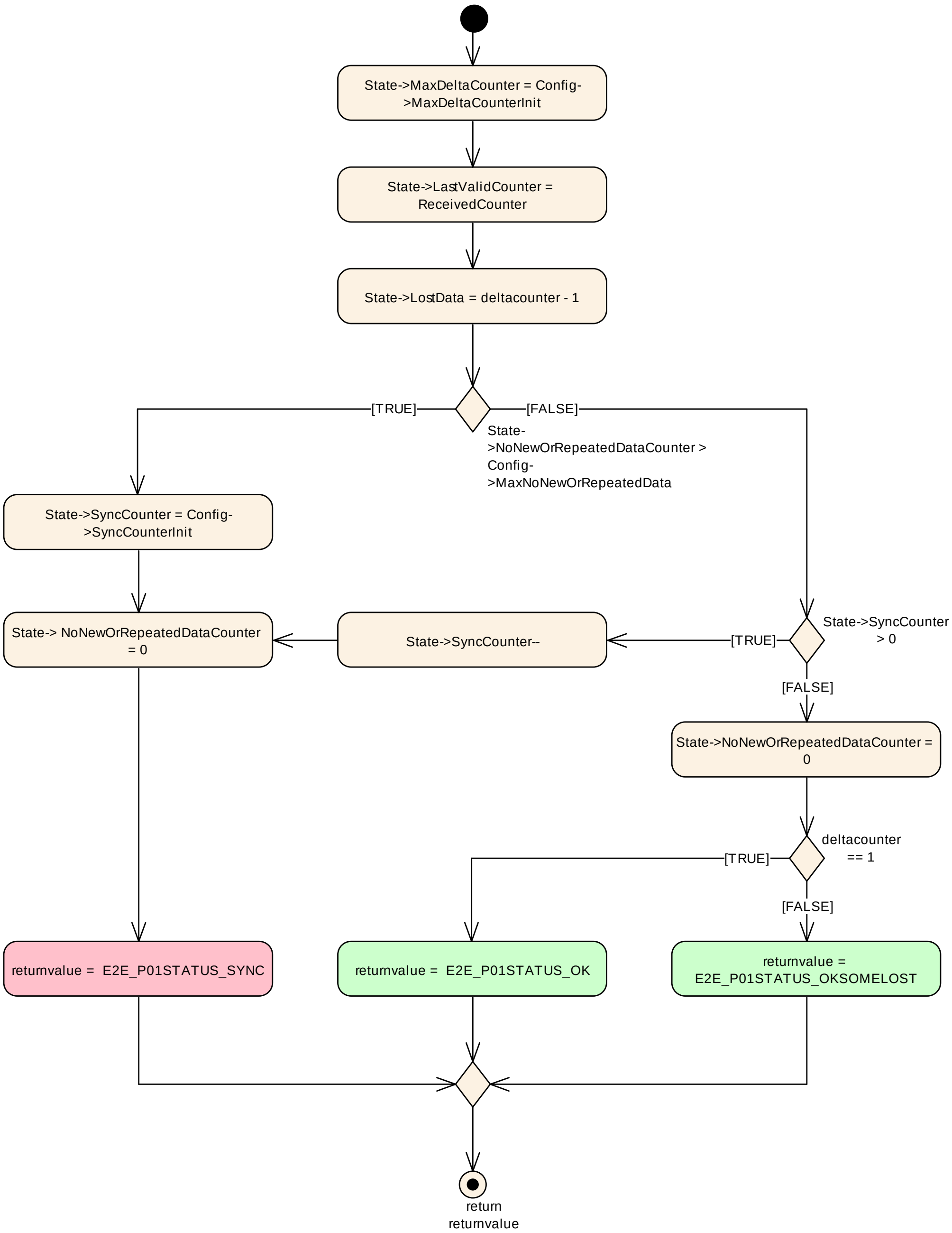

Figure 6.14: E2E Profile Check step ”E2E_P01_handle_ok_and_okSomeLost”

6.3.10.1 Profile 1 Check Status Enumeration

[PRS_E2E_00588] ⌈The E2E_P01Check function shall set State->Status to one of the following enumeration values (see Table 6.3).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_P01STATUS_OK	OK	The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by 1 with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that no Data has been lost since the last correct data reception.
E2E_P01STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed
E2E_P01STATUS_WRONGCRC	Error	The data has been received according to communication medium, but 1. the CRC is incorrect (applicable for all E2E Profile 1 configurations) or 2. the low nibble of the high byte of Data ID is incorrect (applicable only for E2E Profile 1 with E2E_P01DataIDMode = E2E_P01_DATAID_NIBBLE). The two above errors can be a result of corruption, incorrect addressing or masquerade.
E2E_P01STATUS_SYNC	Not Valid	The new data has been received after detection of an unexpected behavior of counter. The data has a correct CRC and a counter within the expected range with respect to the most recent Data received, but the determined continuity check for the counter is not finalized yet.
E2E_P01STATUS_INITIAL	Initial	The new data has been received according to communication medium, the CRC is correct, but this is the first Data since the receiver's initialization or reinitialization, so the Counter cannot be verified yet.
E2E_P01STATUS_REPEATED	Error	The new data has been received according to communication medium, the CRC is correct, but the Counter is identical to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST.

E2E_P01STATUS_OKSOMELOST	OK	The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by DeltaCounter (1 < DeltaCounter = MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that some Data in the sequence have been probably lost since the last correct/initial reception, but this is within the configured tolerance range.
E2E_P01STATUS_WRONGSEQUENCE	Error	The new data has been received according to communication medium, the CRC is correct, but the Counter Delta is too big (DeltaCounter > MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that too many Data in the sequence have been probably lost since the last correct/initial reception.

Table 6.3: E2E Profile 1 Check Status Enumeration

6.3.11 E2E Profile 1 Protocol Examples

The default configuration assumed for the following examples, if not otherwise stated to be different:

E2E_P01ConfigType field	Value
CounterOffset	8
CRCOffset	0
DataID	0x123
DataIDNibbleOffset	12
DataIDMode	E2E_P01_DATAID_BOTH
DataLength	64
MaxDeltaCounterInit	1
MaxNoNewOrRepeatedData	15
SyncCounterInit	0

Table 6.4: E2E Profile 1 protocol example configuration

E2E_P01ProtectStateType field	Value
Counter	0

Table 6.5: E2E Profile 1 example state initialization

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 0:

Byte							
0	1	2	3	4	5	6	7
0xcc	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.6: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_BOTH, counter 0

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 1:

Byte							
0	1	2	3	4	5	6	7
0x91	0x01	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.7: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_BOTH, counter 1

6.3.11.1 DataIDMode set to E2E_P01_DATAID_ALT

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 0:

Byte							
0	1	2	3	4	5	6	7
0x5f	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.8: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_ALT, counter 0

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 1:

Byte							
0	1	2	3	4	5	6	7
0x93	0x01	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.9: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_ALT, counter 1

6.3.11.2 DataIDMode set to E2E_P01_DATAID_LOW

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 0:

Byte							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0x5f	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.10: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_LOW, counter 0

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 1:

Byte							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0x02	0x01	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.11: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_LOW, counter 1

6.3.11.3 DataIDMode set to E2E_P01_DATAID_NIBBLE

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 0:

Byte							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0x2a	0x10	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.12: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_NIBBLE, counter 0

Result data of E2E_P01Protect() with data equals all zeros (0x00), counter = 1:

Byte							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0x77	0x11	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.13: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_NIBBLE, counter 1

6.4 Specification of E2E Profile 2 (only for CP)

[PRS_E2E_00219] ⌈Profile 2 shall provide the following mechanisms: Sequence Number (Counter), Message Key used for CRC calculation (Data ID), Data ID + CRC, Safety Code (CRC) (see Table 6.14).⌋*(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Mechanism	**Description**

Sequence Number (Counter)	4bit (explicitly sent) representing numbers from 0 to 15 incremented by 1 on every send request (Bit 0:3 of Data 1) at sender side. The counter is incremented on every call of the E2E_P02Protect() function, i.e. on every transmission request of the SW-C
Message Key used for CRC calculation (Data ID)	8 bit (not explicitly sent) The specific Data ID used to calculate the CRC depends on the value of the Counter and is an element of an pre-defined set of Data IDs (value of the counter as index to select the particular Data ID used for the protection). For every Data element, the List of Data IDs depending on each value of the counter is unique.
Data ID + CRC	Masquerade and incorrect addressing, insertion
Safety Code(CRC)	8 bit explicitly sent (Data[0]) Polynomial: 0x2F (x8 + x5 + x3 + x2 + x + 1) Start value: 0xFF Final XOR-value: 0xFF Note: This CRC polynomial is different from the CRC-polynomials used by FlexRay and CAN.

Table 6.14: E2E Profile 2 mechanisms

The mechanisms provided by Profile 2 enable the detection of the relevant failure modes except message delay (for details see the table in Table 6.14):

Since this profile is implemented in a Supervision, the Supervision's E2E_P02Check() function itself cannot ensure to be called in a periodic manner. Thus, a required protection mechanism against undetected message delay (e.g. Timeout) must be implemented in the caller.

The E2E mechanisms can detect the following faults or effects of faults:

E2E Mechanism	Detected communication faults
Counter	Repetition, Loss, insertion, incorrect sequence, blocking
Transmission on a regular bases and timeout monitoring using E2E-Library [3]	Loss, delay, blocking
Data ID + CRC	Masquerade and incorrect addressing, insertion
CRC	Corruption, Asymmetric information [4]

Table 6.15: Detectable communication faults using Profile 2

[PRS_E2E_00117] ⌈E2E Profile 2 shall use the Crc_CalculateCRC8H2F() function of the SWS CRC Supervision for calculating CRC checksums.⌋*(RS_E2E_08528)*

[3]Implementation by sender and receiver
[4]for a set of data protected by same CRC

[PRS_E2E_00118] ⌈E2E Profile 2 shall use 0xFF as the start value CRC_StartValue8 for CRC calculation.⌋*(RS_E2E_08528)*

[PRS_E2E_00119] ⌈In E2E Profile 2, the specific Data ID used to calculate a specific CRC shall be of length 8 bit.⌋*(RS_E2E_08528)*

[PRS_E2E_00120] ⌈In E2E Profile 2, the specific Data ID used for CRC calculation shall be selected from a pre-defined DataIDList[16] using the value of the Counter as an index.⌋*(RS_E2E_08528)*

Each data, which is protected by a CRC owns a dedicated DataIDList which is deposited on the sender site and all the receiver sites.

The pre-defined DataIDList[16] is generated offline. In general, there are several factors influencing the contents of DataIDList, e.g:

1. length of the protected data
2. number of protected data elements
3. number of cycles within a masquerading fault has to be detected
4. number of senders and receivers
5. characteristics of the CRC polynomial.

Due to the limited length of the 8bit polynomial, a masquerading fault cannot be detected in a specific cycle when evaluating a received CRC value. Due to the adequate Data IDs in the DataIDList, a masquerading fault can be detected in one of the successive communication cycles.

Due to the underlying rules for the DataIDList, the system design of the application has to take into account that a masquerading fault is detected not until evaluating a certain number of communication cycles.

[PRS_E2E_00121] ⌈In E2E Profile 2, the layout of the data buffer (Data) shall be as depicted in Figure 6.15, with a maximum length of 256 bytes (i.e. N=255)⌋*(RS_E2E_08528)*

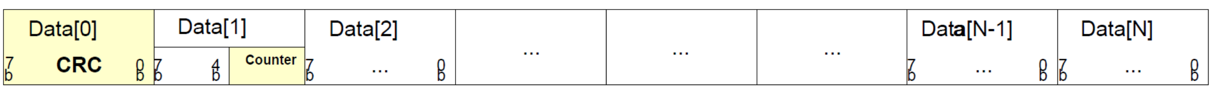

Figure 6.15: E2E Profile 2 data buffer layout

[PRS_E2E_00122] ⌈In E2E Profile 2, the CRC shall be Data[0].⌋*(RS_E2E_08528)*

[PRS_E2E_00123] ⌈In E2E Profile 2, the Counter shall be the low nibble (Bit 0...Bit 3) of Data[1].⌋*(RS_E2E_08528)*

[PRS_E2E_00124] ⌈In E2E Profile 2, both the E2E_P02Protect() and the E2E_P02Forward() function shall not modify any bit of Data except the bits representing the CRC and the Counter.⌋*(RS_E2E_08528)*

[PRS_E2E_00125] ⌈In E2E Profile 2, the E2E_P02Check() function shall not modify any bit in Data.⌋*(RS_E2E_08528)*

6.4.1 E2E_P02Protect

The E2E_P02Protect() function of E2E Profile 2 is called by a SW-C in order to protect its application data against the failure modes as shown in table in Table 6.14. E2E_P02Protect() therefore calculates the Counter and the CRC and puts it into the data buffer (Data). A flow chart with the visual description of the function E2E_P02Protect() is depicted in Figure 6.16 and Figure 6.17.

[PRS_E2E_00126] ⌈In E2E Profile 2, the E2E_P02Protect() function shall perform the activities as specified in Figure 6.16 and Figure 6.17.⌋*(RS_E2E_08528, RS_E2E_-08536)*

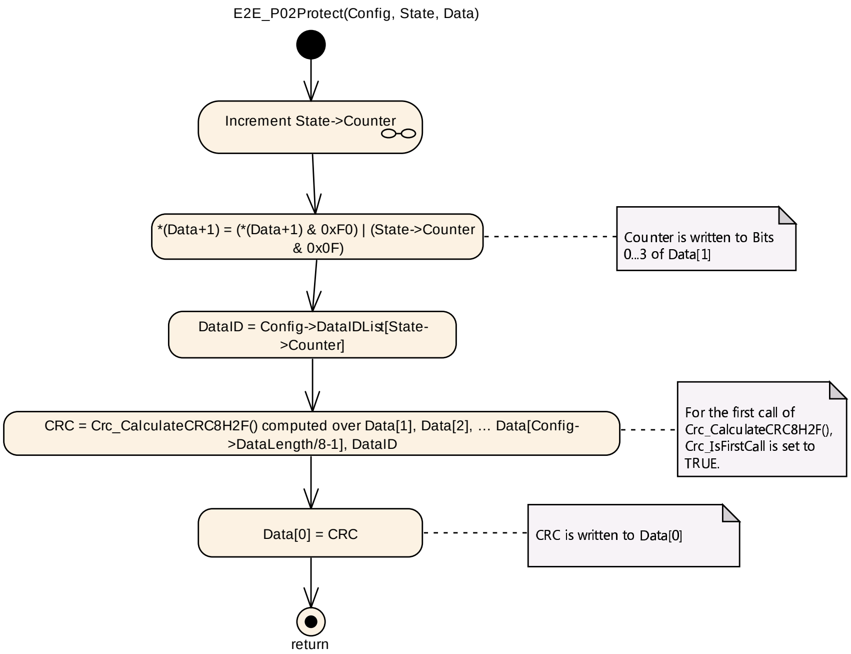

Figure 6.16: E2E_P02Protect()

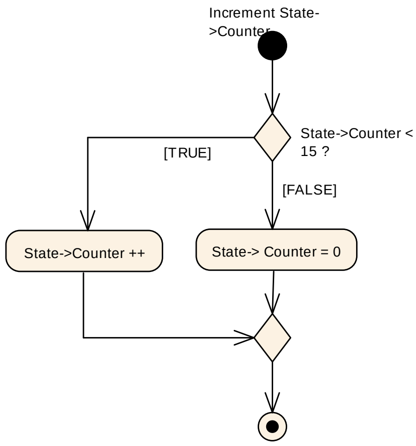

Figure 6.17: Increment Counter

[PRS_E2E_00127] ⌈In E2E Profile 2, the E2E_P02Protect() function shall increment the Counter of the state (E2E_P02ProtectStateType) by 1 on every transmission request from the sending SW-C, i.e. on every call of E2E_P02Protect().⌋*(RS_E2E_08528)*

[PRS_E2E_00128] ⌈In E2E Profile 2, the range of the value of the Counter shall be [0...15].⌋*(RS_E2E_08528)*

[PRS_E2E_00129] ⌈When the Counter has reached its upper bound of 15 (0xF), it shall restart at 0 for the next call of the E2E_P02Protect() from the sending SW-C.⌋*(RS_E2E_08528)*

[PRS_E2E_00130] ⌈In E2E Profile 2, the E2E_P02Protect() function shall update the Counter (i.e. low nibble (Bit 0...Bit 3) of Data byte 1) in the data buffer (Data) after incrementing the Counter.⌋*(RS_E2E_08528)*

The specific Data ID used for this send request is then determined from a DataIDList[] depending on the value of the Counter (Counter is used as an index to select the Data ID from DataIDList[]). The DataIDList[] is defined in E2E_P02ConfigType.

[PRS_E2E_00132] ⌈In E2E Profile 2, after determining the specific Data ID, the E2E_P02Protect() and E2E_P02Forward() functions shall calculate the CRC over Data[1], Data[2], ... Data[Config->DataLength/8-1] of the data buffer (Data) extended with the Data ID.⌋*(RS_E2E_08528)*

[PRS_E2E_00133] ⌈In E2E Profile 2, the E2E_P02Protect() and E2E_P02Forward() functions shall update the CRC (i.e. Data[0]) in the data buffer (Data) after computing the CRC.⌋*(RS_E2E_08528)*

The specific Data ID itself is not transmitted on the bus. It is just a virtual message key used for the CRC calculation.

6.4.2 E2E_P02Forward

The E2E_P02Forward() function of E2E Profile 2 is called by a SW-C in order to protect its application data and forward an received E2E-Status for use cases like translation of signal based to service oriented communication. If the received E2E status equals E2E_P_OK the behavior of the function shall be the same like E2E_P02Protect(). A flow chart with the visual description of the function E2E_P02Forward() is depicted in Figure 6.18 and Figure 6.19.

[PRS_E2E_00613] Draft ⌈In E2E Profile 2, the E2E_P02Forward() function shall perform the activities as specified in Figure 6.18 and Figure 6.17.⌋*(RS_E2E_08528, RS_-E2E_08536)*

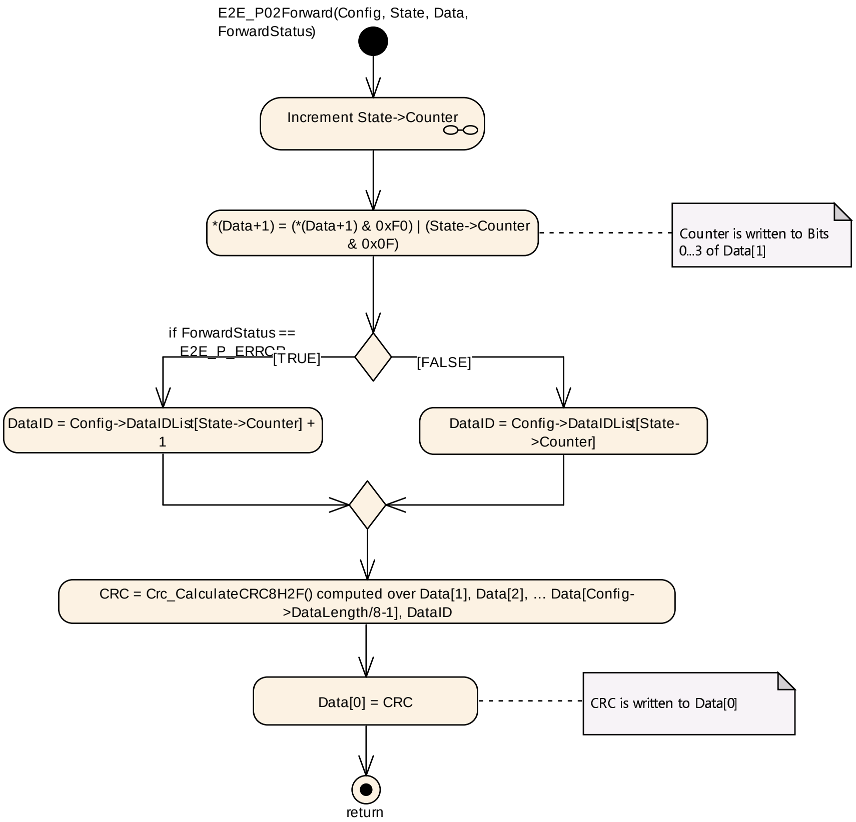

Figure 6.18: E2E_P02Forward()

[PRS_E2E_00614] Draft ⌈In E2E Profile 2, the E2E_P02Forward() function shall increment the Counter according to Figure 6.19.⌋*(RS_E2E_08528)*

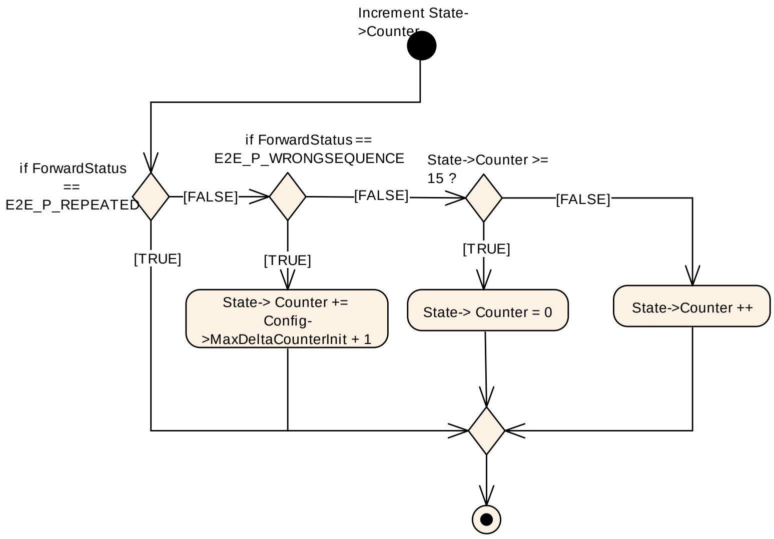

Figure 6.19: Increment Counter

6.4.3 E2E_P02Check

The E2E_P02Check() function is used as an error detection mechanism by a caller in order to check if the received data is correct with respect to the failure modes mentioned in the profile summary.

A flow chart with the visual description of the function E2E_P02Check() is depicted in Figure 6.20, Figure 6.21 and Figure 6.22.

[PRS_E2E_00134] ⌈In E2E Profile 2, the E2E_P02Check() function shall perform the activities as specified in Figure 6.20, Figure 6.21 and Figure 6.22.⌋*(RS_E2E_08528, RS_E2E_08536)*

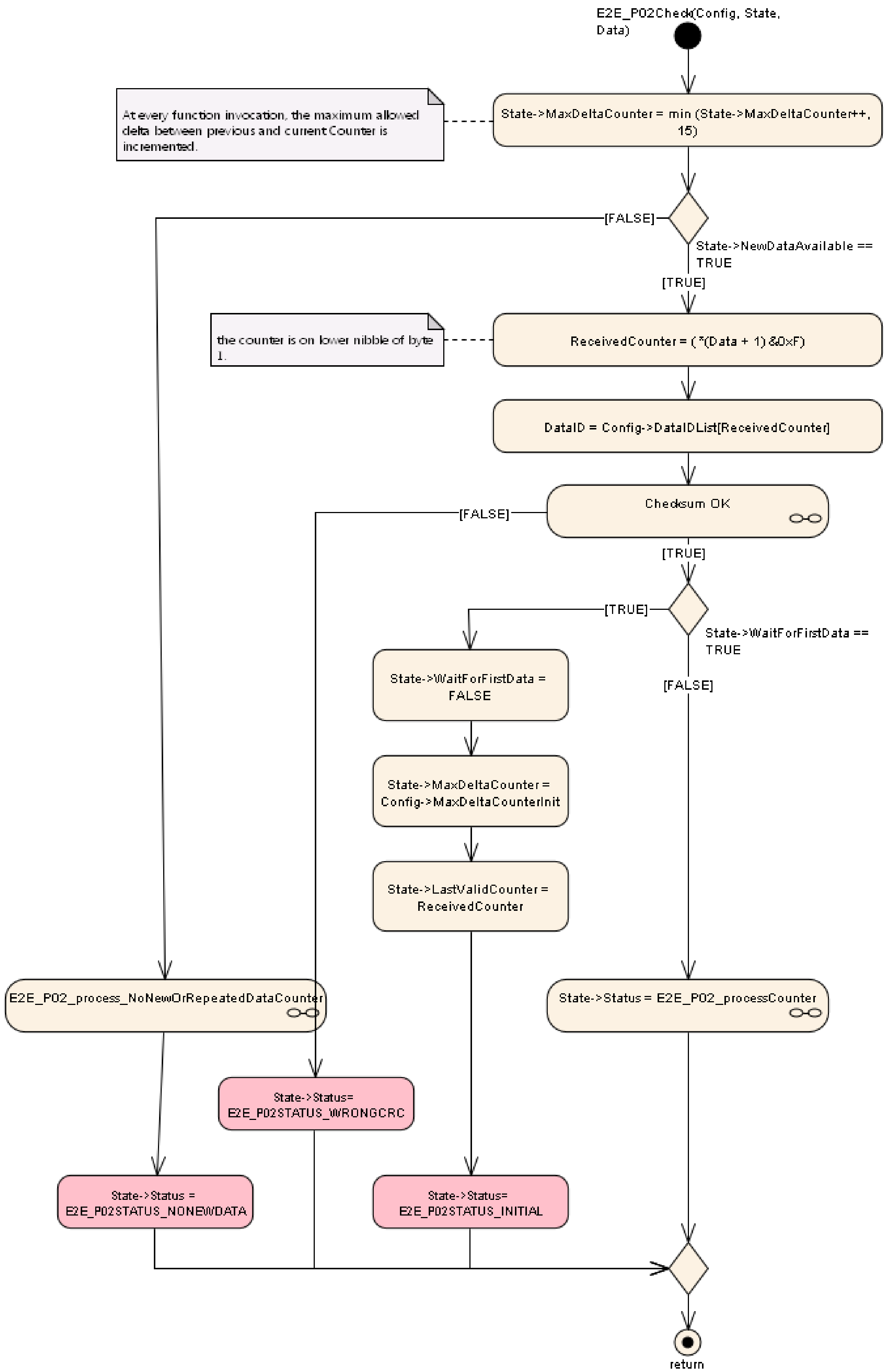

Figure 6.20: E2E_P02Check

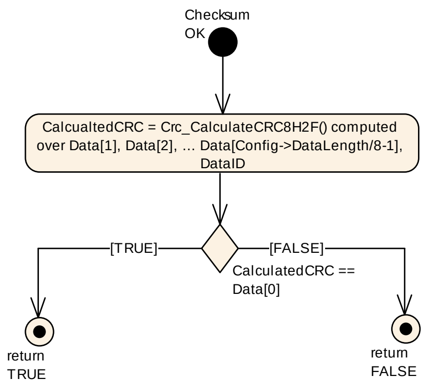

Figure 6.21: Checksum OK

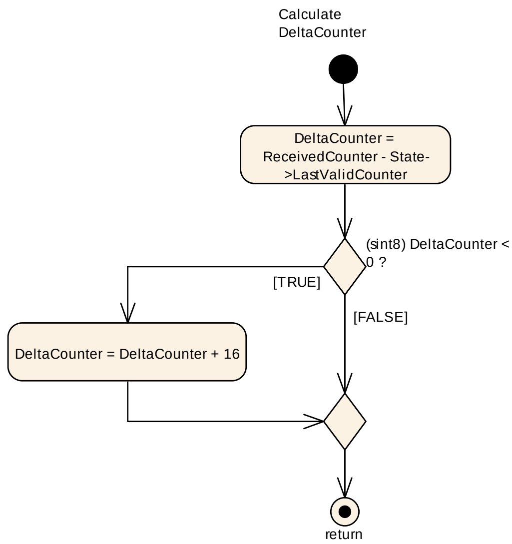

Figure 6.22: Calculate Delta Counter

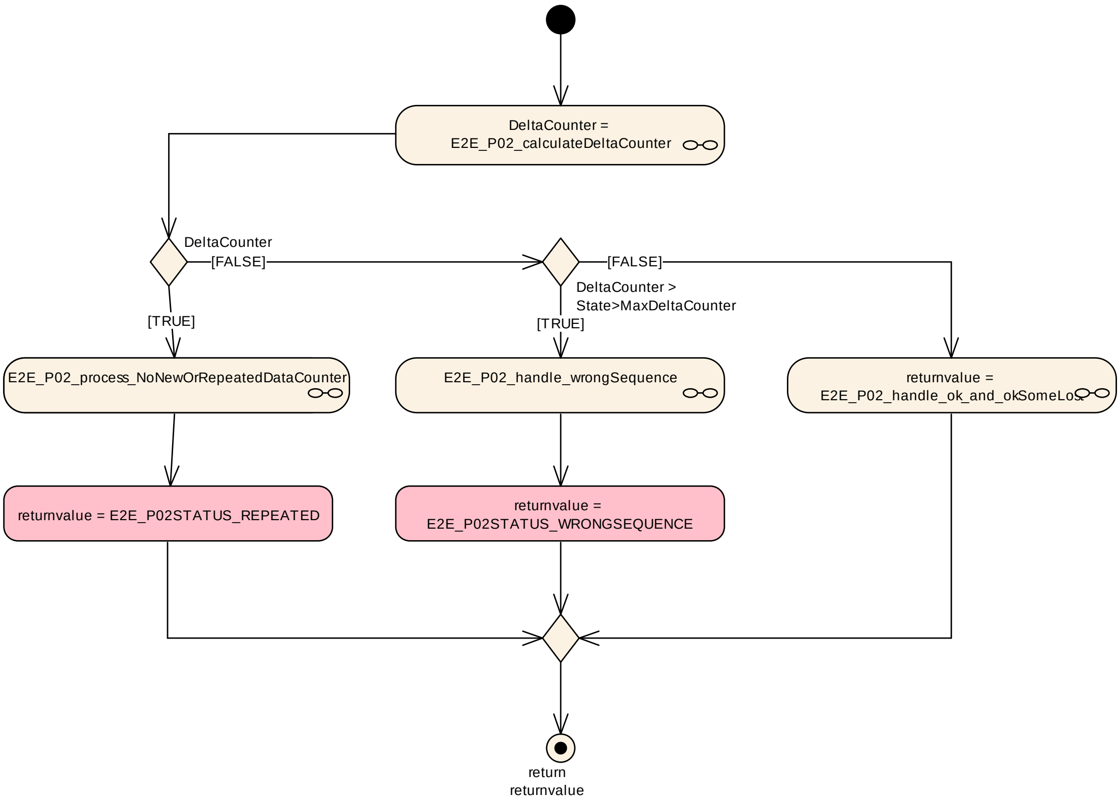

Figure 6.23: E2E Profile Check step "E2E_P02_process_counter"

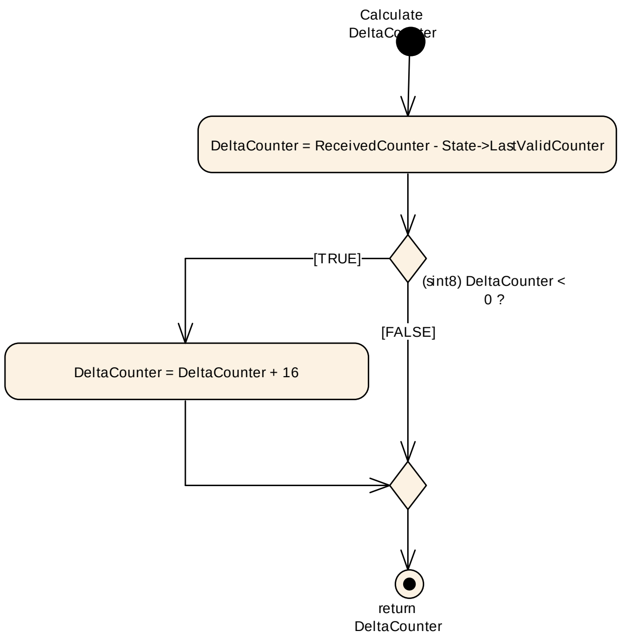

Figure 6.24: E2E Profile Check step "E2E_P02_process_counter"

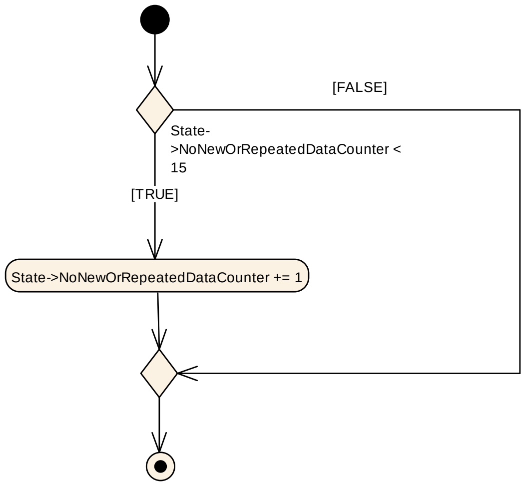

Figure 6.25: E2E Profile Check step "E2E_P02_process_NoNewOrRepeatedDataCounter"

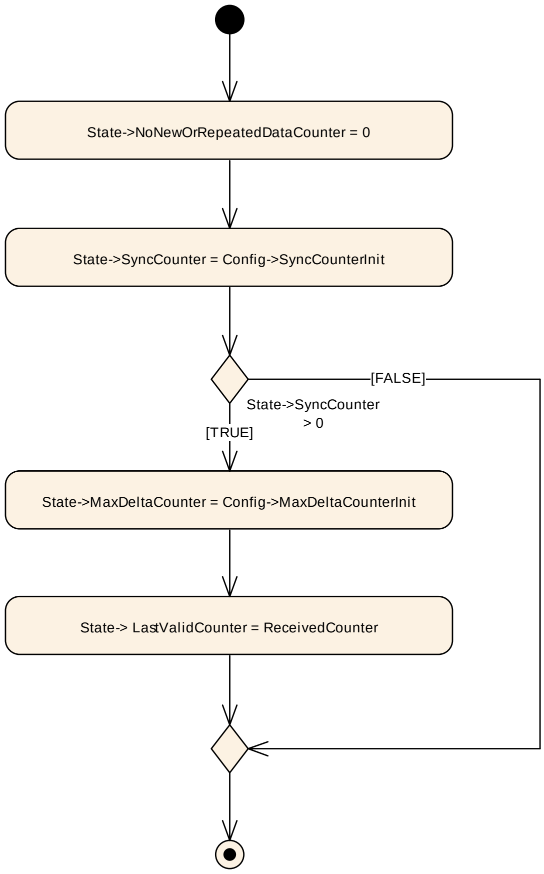

Figure 6.26: E2E Profile Check step "E2E_P02_handle_wrongSequence"

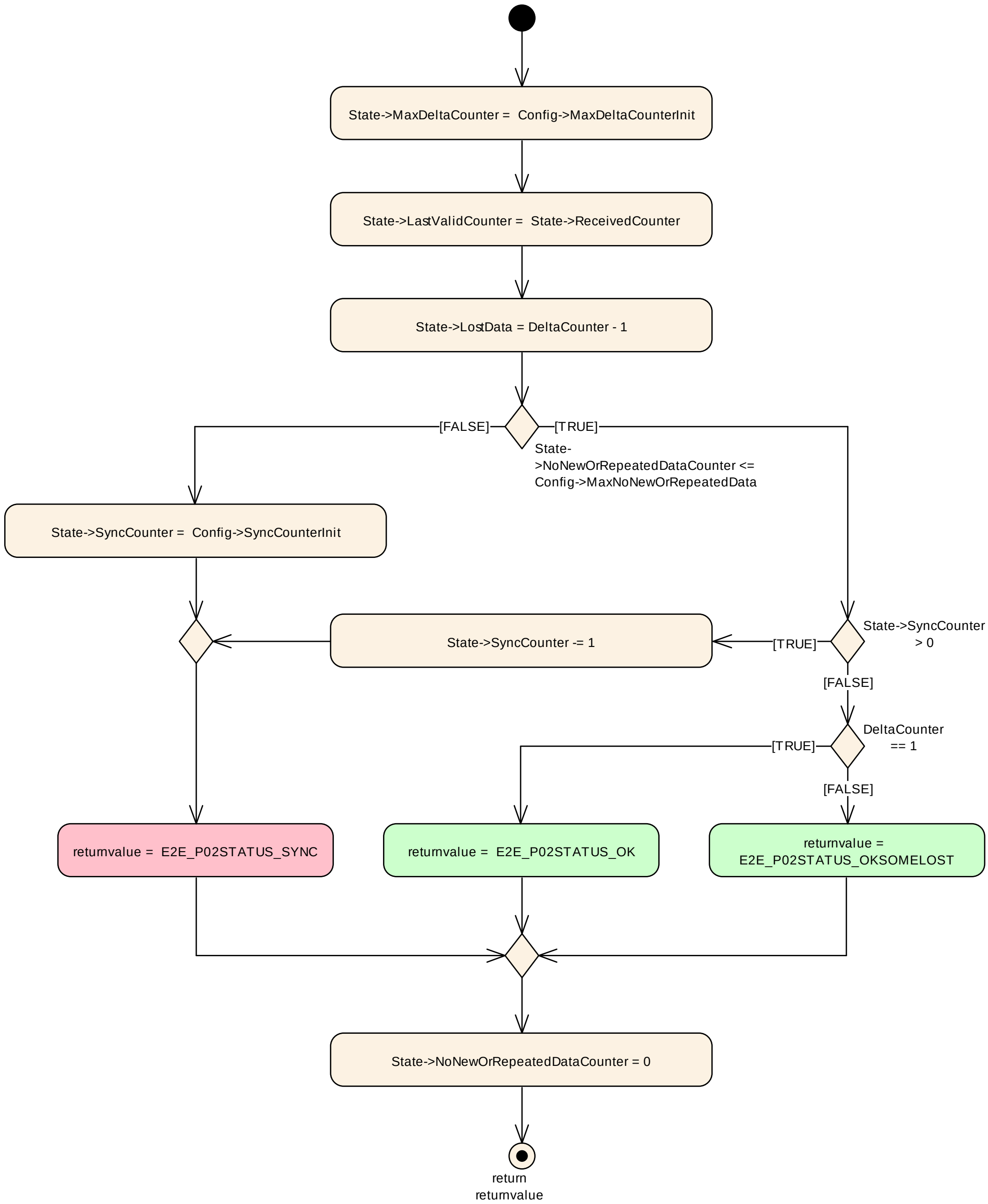

Figure 6.27: E2E Profile Check step "E2E_P02_handle_ok_and_okSomeLost"

First, the E2E_P02Check() function increments the value MaxDeltaCounter. MaxDeltaCounter specifies the maximum allowed difference between two Counter values of two consecutively received valid messages. Note: MaxDeltaCounter is used in order to perform a plausibility check for the failure mode re-sequencing. If the flag NewDataAvailable is set, the E2E_P02Check() function continues with the evaluation of the CRC. Otherwise, it returns with Status set to E2E_P02STATUS_NONEWDATA. To evaluate the correctness of the CRC, the following actions are performed:

- The specific Data ID is determined using the value of the Counter as provided in Data.

- Then the CRC is calculated over Data payload extended with the Data ID as last Byte: CalculatedCRC = Crc_CalculateCRC8H2F() calculated over Data[1], Data[2], ... Data[Config->DataLength/8-1], Data ID
- Finally, the check for correctness of the received Data is performed by comparing CalculatedCRC with the value of CRC stored in Data.

In case CRC in Data and CalculatedCRC do not match, the E2E_P02Check() function returns with Status E2E_P02STATUS_WRONGCRC, otherwise it continues with further evaluation steps.

The flag WaitForFirstData specifies if the SW-C expects the first message after startup or after a timeout error. This flag should be set by the SW-C if the SW-C expects the first message e.g. after startup or after reinitialization due to error handling. This flag is allowed to be reset by the E2E_P02Check() function only. The reception of the first message is a special event because no plausibility checks against previously received messages is performed.

If the flag WaitForFirstData is set by the SW-C, E2E_P02Check() does not evaluate the Counter of Data and returns with Status E2E_P02STATUS_INITIAL. However, if the flag WaitForFirstData is reset (the SW-C does not expect the first message) the E2E_P02Check() function evaluates the value of the Counter in Data.

For messages with a received Counter value within a valid range, the E2E_P02Check() function returns either with E2E_P02STATUS_OK or E2E_P02STATUS_OKSOMELOST. In LostData, the number of missing messages since the most recently received valid message is provided to the SW-C.

For messages with a received Counter value outside of a valid range, E2E_P02Check() returns with one of the following states: E2E_P02STATUS_WRONGSEQUENCE or E2E_P02STATUS_REPEATED.

[PRS_E2E_00135] ⌈In E2E Profile 2, the local variable DeltaCounter shall be calculated by subtracting LastValidCounter from Counter in Data, considering an overflow due to the range of values [0...15].⌋*(RS_E2E_08528)*

Details on the calculation of DeltaCounter are depicted in Figure 7-12.

[PRS_E2E_00136] ⌈In E2E Profile 2, MaxDeltaCounter shall specify the maximum allowed difference between two Counter values of two consecutively received valid messages.⌋*(RS_E2E_08528)*

[PRS_E2E_00137] ⌈In E2E Profile 2, MaxDeltaCounter shall be incremented by 1 every time the E2E_P02Check() function is called, up to the maximum value of 15 (0xF).⌋*(RS_E2E_08528)*

[PRS_E2E_00138] ⌈In E2E Profile 2, the E2E_P02Check() function shall set Status to E2E_P02STATUS_NONEWDATA if the attribute NewDataAvailable is FALSE.⌋*(RS_E2E_08528)*

[PRS_E2E_00139] ⌈In E2E Profile 2, the E2E_P02Check() function shall determine the specific Data ID from DataIDList using the Counter of the received Data as index.⌋ *(RS_E2E_08528)*

[PRS_E2E_00140] ⌈In E2E Profile 2, the E2E_P02Check() function shall calculate CalculatedCRC over Data[1], Data[2], ... Data[Config->DataLength/8-1] of the data buffer (Data) extended with the determined Data ID.⌋ *(RS_E2E_08528)*

[PRS_E2E_00141] ⌈In E2E Profile 2, the E2E_P02Check() function shall set Status to E2E_P02STATUS_WRONGCRC if the calculated CalculatedCRC value differs from the value of the CRC in Data.

⌋ *(RS_E2E_08528)*

[PRS_E2E_00142] ⌈In E2E Profile 2, the E2E_P02Check() function shall set Status to E2E_P02STATUS_INITIAL if the flag WaitForFirstData is TRUE.⌋ *(RS_E2E_08528)*

[PRS_E2E_00143] ⌈In E2E Profile 2, the E2E_P02Check() function shall clear the flag WaitForFirstData if it returns with Status E2E_P02STATUS_INITIAL.⌋ *(RS_E2E_08528)*

For the first message after start up no plausibility check of the Counter is possible. Thus, at least a minimum number of messages need to be received in order to perform a check of the Counter values and in order to guarantee that at least one correct message was received.

[PRS_E2E_00145] ⌈The E2E_P02Check() function shall

- set Status to E2E_P02STATUS_WRONGSEQUENCE; and
- re-initialize SyncCounter with SyncCounterInit

if the calculated value of DeltaCounter exceeds the value of MaxDeltaCounter.⌋ *(RS_E2E_08528)*

[PRS_E2E_00146] ⌈The E2E_P02Check() function shall set Status to E2E_P02STATUS_REPEATED if the calculated DeltaCounter equals 0.⌋ *(RS_E2E_08528)*

[PRS_E2E_00147] ⌈The E2E_P02Check() function shall set Status to E2E_P02STATUS_OK if the following conditions are true:

- the calculated DeltaCounter equals 1; and
- the value of the NoNewOrRepeatedDataCounter is less than or equal to MaxNoNewOrRepeatedData (i.e. State –> NoNewOrRepeatedDataCounter <= Config –> MaxNoNewOrRepeatedData); and
- the SyncCounter equals 0.

⌋ *(RS_E2E_08528)*

[PRS_E2E_00298] ⌈The E2E_P02Check() function shall

- re-initialize SyncCounter with SyncCounterInit; and
- set Status to E2E_P02STATUS_SYNC; if the following conditions are true:
- the calculated DeltaCounter is within the parameters of 1 and MaxDeltaCounter (i.e. 1 =/< DeltaCounter =/< MaxDeltaCounter); and
- the value of the NoNewOrRepeatedDataCounter exceeds MaxNoNewOrRepeatedData. (i.e. State NoNewOrRepeatedDataCounter > Config MaxNoNewOrRepeatedData)

⌋*(RS_E2E_08528)*

[PRS_E2E_00299] ⌈The E2E_P02Check() function shall

- decrement SyncCounter by 1; and
- set Status to E2E_P02STATUS_SYNC if the following conditions are true:
- the calculated DeltaCounter is within the parameters of 1 and MaxDeltaCounter (i.e. 1 =/< DeltaCounter =/< MaxDeltaCounter); and
- the value of the NoNewOrRepeatedDataCounter is less than or equal to MaxNoNewOrRepeatedData (i.e. State NoNewOrRepeatedDataCounter =/< Config MaxNoNewOrRepeatedData); and
- the SyncCounter exceeds 0.

⌋*(RS_E2E_08528)*

[PRS_E2E_00148] ⌈The E2E_P02Check() function shall set Status to E2E_P02STATUS_OKSOMELOST if the following conditions are true:

- the calculated DeltaCounter is greater-than 1 but less-than or equal to MaxDeltaCounter (i.e. 1 < DeltaCounter =/< MaxDeltaCounter); and
- the NoNewOrRepeatedDataCounter is less than or equal to MaxNoNewOrRepeatedData (i.e. State NoNewOrRepeatedDataCounter =/< Config MaxNoNewOrRepeatedData); and
- the SyncCounter equals 0.

⌋*(RS_E2E_08528)*

[PRS_E2E_00149] ⌈The E2E_P02Check() function shall set the value LostData to (DeltaCounter - 1) if the calculated DeltaCounter is greater-than 1 but less-than or equal to MaxDeltaCounter.⌋*(RS_E2E_08528)*

[PRS_E2E_00150] ⌈The E2E_P02Check() function shall r-initialize MaxDeltaCounter with MaxDeltaCounterInit if it returns one of the following Status:

- E2E_P02STATUS_OK; or
- E2E_P02STATUS_OKSOMELOST; or
- E2E_P02STATUS_INITIAL; or

- E2E_P02STATUS_SYNC; or
- E2E_P02STATUS_WRONGSEQUENCE on condition that SyncCounter exceeds 0 (i.e. SyncCounter > 0).

⌋*(RS_E2E_08528)*

[PRS_E2E_00151] ⌈The E2E_P02Check() function shall set LastValidCounter to Counter of Data if it returns one of the following Status:

- E2E_P02STATUS_OK; or
- E2E_P02STATUS_OKSOMELOST; or
- E2E_P02STATUS_INITIAL; or
- E2E_P02STATUS_SYNC; or
- E2E_P02STATUS_WRONGSEQUENCE on condition that SyncCounter exceeds 0 (i.e. SyncCounter > 0).

⌋*(RS_E2E_08528)*

[PRS_E2E_00300] ⌈The E2E_P02Check() function shall reset the NoNewOrRepeatedDataCounter to 0 if it returns one of the following status:

- E2E_P02STATUS_OK; or
- E2E_P02STATUS_OKSOMELOST; or
- E2E_P02STATUS_SYNC; or
- E2E_P02STATUS_WRONGSEQUENCE

⌋*(RS_E2E_08528)*

[PRS_E2E_00301] ⌈The E2E_P02Check() function shall increment NoNewOrRepeatedDataCounter by 1 if it returns the Status E2E_P02STATUS_NONEWDATA or E2E_P02STATUS_REPEATED up to the maximum value of Counter (i.e. 15 or 0xF).⌋*(RS_E2E_08528)*

6.4.3.1 Profile 2 Check Status Enumeration

[PRS_E2E_00589] ⌈The E2E_P02Check function shall set State->Status to one of the following enumeration values (see Table 6.16).⌋*(RS_E2E_08528)*

Name	State Type	Description

E2E_P02STATUS_OK	OK	The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by 1 with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that no Data has been lost since the last correct data reception.
E2E_P02STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed.
E2E_P02STATUS_WRONGCRC	Error	The data has been received according to communication medium, but the CRC is incorrect.
E2E_P02STATUS_SYNC	Not Valid	The new data has been received after detection of an unexpected behavior of counter. The data has a correct CRC and a counter within the expected range with respect to the most recent Data received, but the determined continuity check for the counter is not finalized yet.
E2E_P02STATUS_INITIAL	Initial	The new data has been received according to communication medium, the CRC is correct, but this is the first Data since the receiver's initialization or reinitialization, so the Counter cannot be verified yet.
E2E_P02STATUS_REPEATED	Error	The new data has been received according to communication medium, the CRC is correct, but the Counter is identical to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST.
E2E_P02STATUS_OKSOMELOST	OK	The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by DeltaCounter (1 < DeltaCounter =MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that some Data in the sequence have been probably lost since the last correct/initial reception, but this is within the configured tolerance range.

E2E_P02STATUS_WRONGSEQUENCE	Error	The new data has been received according to communication medium, the CRC is correct, but the Counter Delta is too big (DeltaCounter > MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that too many Data in the sequence have been probably lost since the last correct/initial reception.

Table 6.16: E2E Profile 2 Check Status Enumeration

6.4.4 E2E Profile 2 Protocol Examples

E2E_P02ConfigType field	Value
DataLength	64
DataIDList	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0x10
MaxDeltaCounterInit	1
MaxNoNewOrRepeatedData	15
SyncCounterInit	0
Offset	0

Table 6.17: E2E Profile 2 protocol example configuration

E2E_P02ProtectStateType field	Value
Counter	0

Table 6.18: E2E Profile 2 example state initialization

Result data of E2E_P02Protect() with data equals all zeros (0x00), counter starting with 1 (note: first used counter is 1, although counter field is initialized with 0, as counter is incremented before usage):

Counter	DataID	Byte							
		0	1	2	3	4	5	6	7
1	0x02	0x1b	0x01	0x00	0x00	0x00	0x00	0x00	0x00
2	0x03	0x98	0x02	0x00	0x00	0x00	0x00	0x00	0x00
3	0x04	0x31	0x03	0x00	0x00	0x00	0x00	0x00	0x00
4	0x05	0x0d	0x04	0x00	0x00	0x00	0x00	0x00	0x00
5	0x06	0x18	0x05	0x00	0x00	0x00	0x00	0x00	0x00
6	0x07	0x9b	0x06	0x00	0x00	0x00	0x00	0x00	0x00
7	0x08	0x65	0x07	0x00	0x00	0x00	0x00	0x00	0x00
8	0x09	0x08	0x08	0x00	0x00	0x00	0x00	0x00	0x00
9	0x0a	0x1d	0x09	0x00	0x00	0x00	0x00	0x00	0x00
10	0x0b	0x9e	0x0a	0x00	0x00	0x00	0x00	0x00	0x00
11	0x0c	0x37	0x0b	0x00	0x00	0x00	0x00	0x00	0x00
12	0x0d	0x0b	0x0c	0x00	0x00	0x00	0x00	0x00	0x00
13	0x0e	0x1e	0x0d	0x00	0x00	0x00	0x00	0x00	0x00
14	0x0f	0x9d	0x0e	0x00	0x00	0x00	0x00	0x00	0x00
15	0x10	0xcd	0x0f	0x00	0x00	0x00	0x00	0x00	0x00
0	0x01	0x0e	0x00	0x00	0x00	0x00	0x00	0x00	0x00
		CRC	*4 bit Data + 4 bit Counter*	*Data*					

Table 6.19: E2E Profile 2 example protect result

6.5 Specification of E2E Profile 4

[PRS_E2E_00372] ⌈Profile 4 shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID (see Table 6.20).⌋*(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Control field	Description
Length	16 bits, to support dynamic-size data.
Counter	16-bits.
CRC	32 bits, polynomial in normal form 0x1F4ACFB13, provided by CRC library. Note: This CRC polynomial is different from the CRC-polynomials used by FlexRay, CAN and LIN and TCPIP.
Data ID	32-bits, unique system-wide.

Table 6.20: E2E Profile 4 mechanisms

The E2E mechanisms can detect the following faults or effects of faults:

Fault	Main safety mechanisms
Repetition of information	Counter
Loss of information	Counter
Delay of information	Counter
Insertion of information	Data ID
Masquerading	Data ID, CRC
Incorrect addressing	Data ID
Incorrect sequence of information	Counter
Corruption of information	CRC
Asymmetric information sent from a sender to multiple receivers	CRC (to detect corruption at any of receivers)
Information from a sender received by only a subset of the receivers	Counter (loss on specific receivers)
Blocking access to a communication channel	Counter (loss or timeout)

Table 6.21: Detectable communication faults using Profile 4

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[2].

6.5.1 Data Layout

6.5.1.1 User data layout

In the E2E Profile 4, the user data layout (of the data to be protected) is not constrained by E2E Profile 4 - there is only a requirement that the length of data to be protected is multiple of 1 byte.

6.5.1.2 Header layout

The header of the E2E Profile 4 has one fixed layout, as follows:

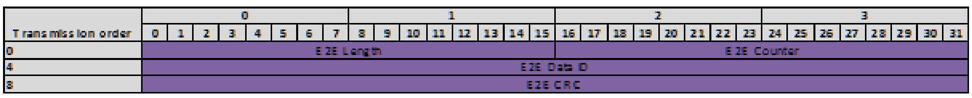

Figure 6.28: E2E Profile 4 Header

The bit numbering shown above represents the order in which bits are transmitted. The E2E header fields (e.g. E2E Counter) are encoded as:

1. Big Endian (most significant byte fist) - imposed by profile
2. LSB Fist (least significant bit within byte first) - imposed by TCPIP bus

For example, the 16 bits of the E2E counter are transmitted in the following order (higher number meaning higher significance): 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7.

The header can be placed at a specific location in the protected data, by configuring the offset of the entire E2E header.

6.5.2 Counter

In E2E Profile 4, the counter is initialized, incremented, reset and checked by E2E profile. The counter is not manipulated or used by the caller of the E2E Supervision.

[PRS_E2E_00478] ⌈In E2E Profile 4, on the sender side, for the first transmission request of a data element the counter shall be initialized with 0 and shall be incremented by 1 for every subsequent send request. When the counter reaches the maximum value (0xFF'FF), then it shall restart with 0 for the next send request.⌋*(RS_E2E_08539)*

Note: This specification was previously falsely identified as PRS_E2EProtocol_00324.

Note that the counter value 0xFF'FF is not reseved as a special invalid value, but it is used as a normal counter value.

In E2E Profile 4, on the receiver side, by evaluating the counter of received data against the counter of previously received data, the following is detected:

1. Repetition:

   a. no new data has arrived since last invocation of E2E Supervision check function, b. the data is repeated

2. OK: a. counter is incremented by one (i.e. no data lost), b. counter is incremented more than by one, but still within allowed limits (i.e. some data lost),

3. Wrong sequence: a. counter is incremented more than allowed (i.e. too many data lost).

Case 1 corresponds to the failed alive counter check, and case 3 correspond to failed sequence counter check.

The above requirements are specified in more details by the UML diagrams in the following document sections.

6.5.3 Data ID

The unique Data IDs are to verify the identity of each transmitted safety-related data element.

[PRS_E2E_00326] ⌈In the E2E Profile 4, the Data ID shall be explicitly transmitted, i.e. it shall be the part of the transmitted E2E header.⌋*(RS_E2E_08539)*

[PRS_E2E_UC_00327] ⌈In the E2E profile 4, the Data IDs shall be globally unique within the network of communicating system (made of several ECUs each sending different data).⌋*(RS_E2E_08539)*

In case of usage of E2E Supervision for protecting data elements (i.e invocation from RTE), due to multiplicity of communication (1:1 or 1:N), a consumer of a data element expects only a specific data element, which is checked by E2E Supervision using Data ID.

In case of usage of E2E Supervision for protecting messages (i.e. invocation from COM), the receiver COM expects at a reception only a specific message, which is checked by E2E Supervision using Data ID.

6.5.4 Length

The Length field is introduced to support variable-size length - the Data [] array storing the serialized data can potentially have a different length in each cycle. The Length includes user data + E2E Header (CRC + Counter + Length + DataID).

6.5.5 CRC

E2E Profile 4 uses a 32-bit CRC, to ensure a high detection rate and high Hamming Distance.

[PRS_E2E_00329] ⌈E2E Profile 4 shall use the Crc_CalculateCRC32P4 () function of the SWS CRC Supervision for calculating the CRC.⌋*(RS_E2E_08528, RS_E2E_08539)*

Note: The CRC used by E2E Profile 4 is different from the CRCs used by FlexRay, CAN and TCP/IP. It is also provided by different software modules (FlexRay, CAN and TCP/IP stack CRCs/checksums are provided by hardware support in Communication Controllers or by communication stack software, but not by CRC Supervision).

[PRS_E2E_00330] ⌈In E2E Profile 4, the CRC shall be calculated over the entire E2E header (excluding the CRC bytes) and over the user data.⌋*(RS_E2E_08536)*

6.5.6 Timeout detection

The previously mentioned mechanisms (CRC, Counter, Data ID, Length) enable to check the validity of received data element, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for Data Elements or respectively messages, but instead if the receiver reads the currently available data (i.e. checks if new data is available). Then, by means of the counter, the receiver can detect loss of communication and timeouts.

6.5.7 E2E Profile 4 variants

[PRS_E2E_00584] ⌈The E2E Profile variant 4A is defined as follows:

1. The CRC is the 64th bit in the signal group
2. The max Delta Counter is 1

⌋*(RS_E2E_08528)*

[PRS_E2E_00585] ⌈The E2E Profile variant 4B is defined as follows:

1. The CRC is the 64th bit in the signal group
2. The max Delta Counter is 2

⌋*(RS_E2E_08528)*

6.5.8 E2E_P04Protect

The function E2E_P04Protect() performs the steps as specified by the following eight diagrams in this section.

[PRS_E2E_00362] ⌈The function E2E_P04Protect() shall have the overall behavior as shown in Figure 6.29.⌋*(RS_E2E_08539)*

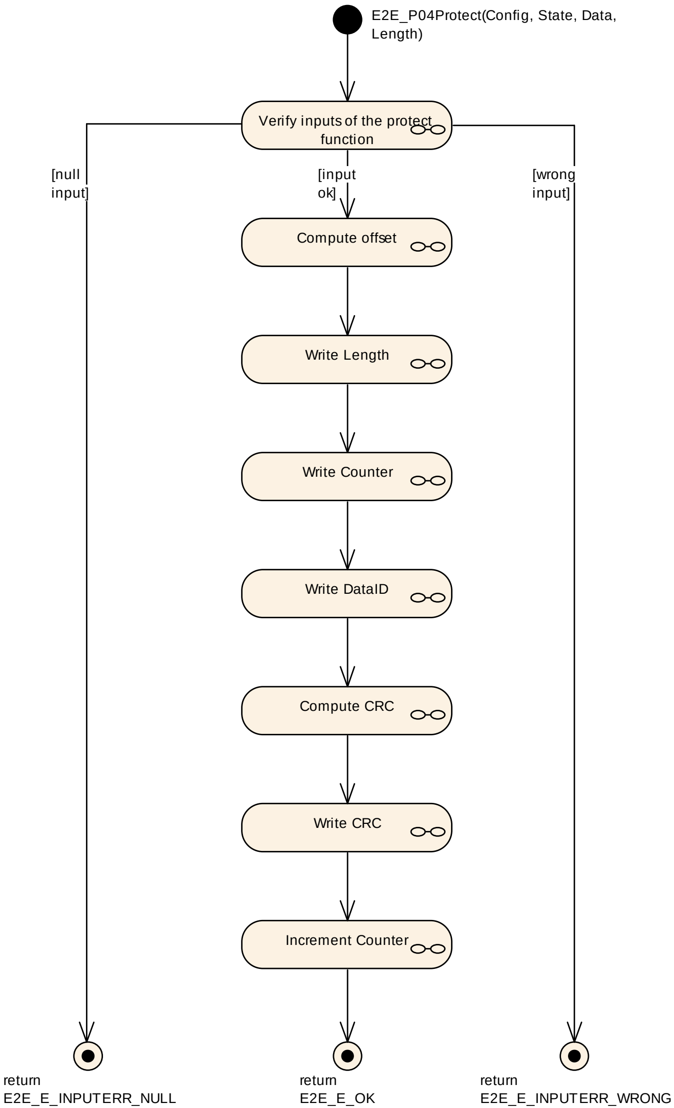

Figure 6.29: E2E Profile 4 Protect

[PRS_E2E_00363] ⌈The step "Verify inputs of the protect function" in E2E_P04Protect() shall behave as shown in Figure 6.30.⌋*(RS_E2E_08539)*

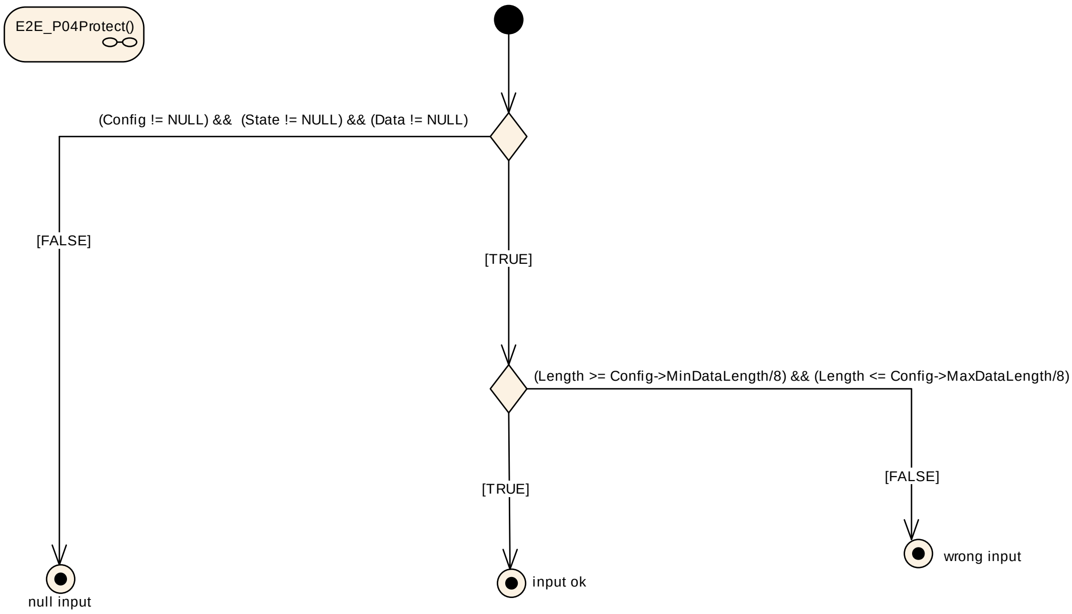

Figure 6.30: E2E Profile 4 Protect step "Verify inputs of the protect function"

[PRS_E2E_00376] ⌈The step "Compute offset" in E2E_P04Protect(), E2E_P04Forward() and E2E_P04Check() shall behave as shown in Figure 6.31.⌋*(RS_E2E_08539)*

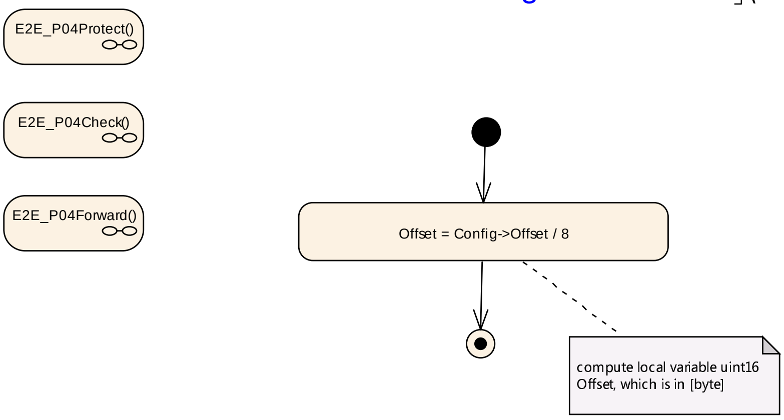

Figure 6.31: E2E Profile 4 Protect step "Compute offset"

[PRS_E2E_00364] ⌈The step "Write Length" in E2E_P04Protect() and E2E_P04Forward() shall behave as shown in Figure 6.32.⌋*(RS_E2E_08539)*

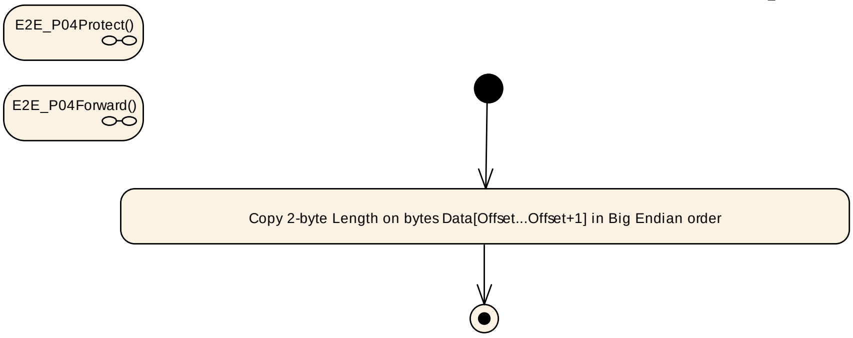

Figure 6.32: E2E Profile 4 Protect step "Write Length"

[PRS_E2E_00365] ⌈The step "Write Counter" in E2E_P04Protect() shall behave as shown in Figure 6.33.⌋*(RS_E2E_08539)*

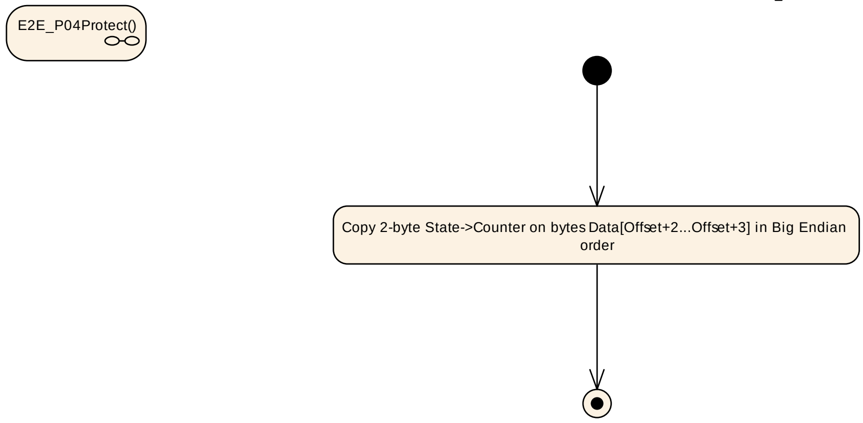

Figure 6.33: E2E Profile 4 Protect step "Write Counter"

[PRS_E2E_00366] ⌈The step "Write DataID" in E2E_P04Protect() shall behave as shown in Figure 6.34.⌋*(RS_E2E_08539)*

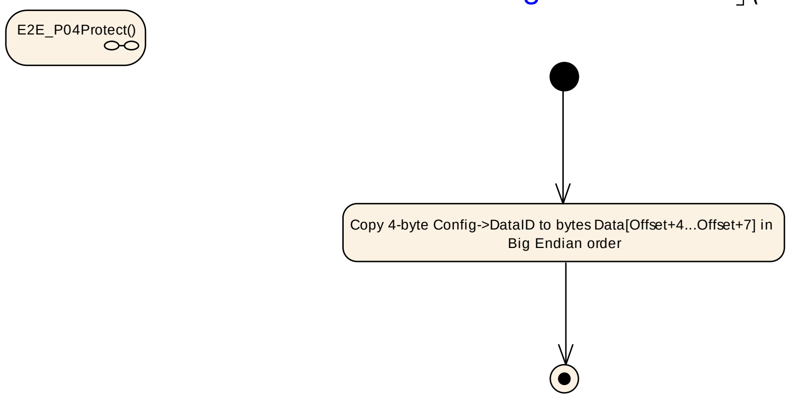

Figure 6.34: E2E Profile 4 Protect step "Write DataID"

[PRS_E2E_00367] ⌈The step "Compute CRC" in E2E_P04Protect(), E2E_P04Forward() and in E2E_P04Check()

shall behave as shown in Figure 6.35.⌋(RS_E2E_08539)

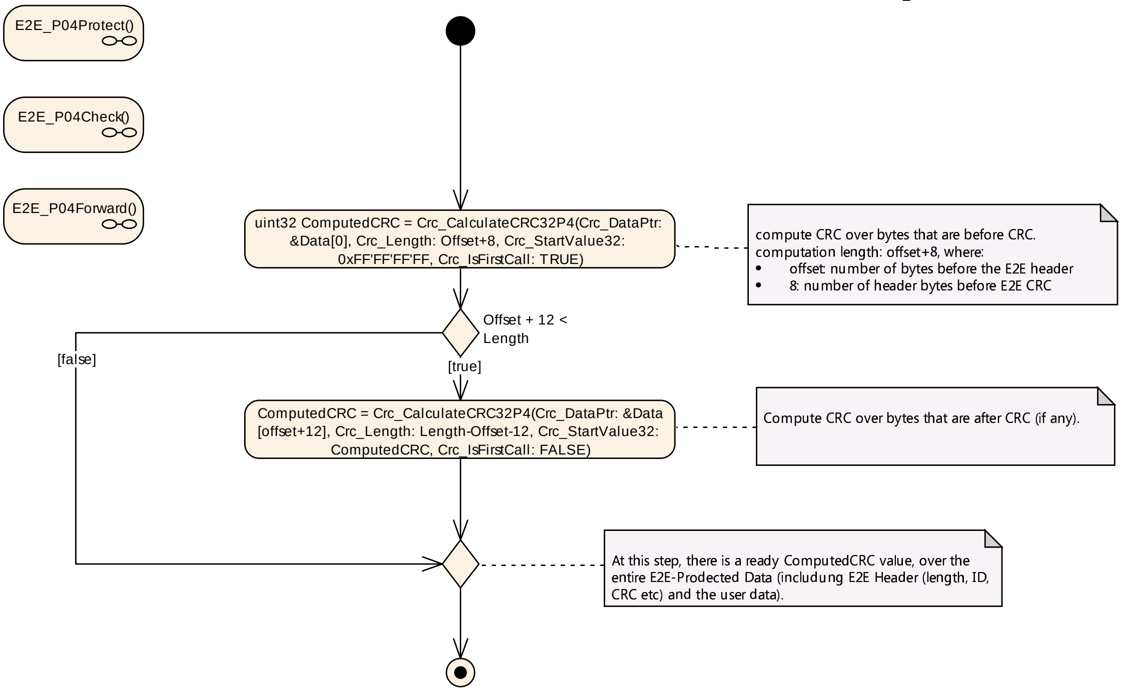

Figure 6.35: E2E Profile 4 Protect and Check step "Compute CRC"

[PRS_E2E_00368] ⌈The step "Write CRC"in E2E_P04Protect() and E2E_P04Forward() shall behave as shown in Figure 6.36.⌋(RS_E2E_08539)

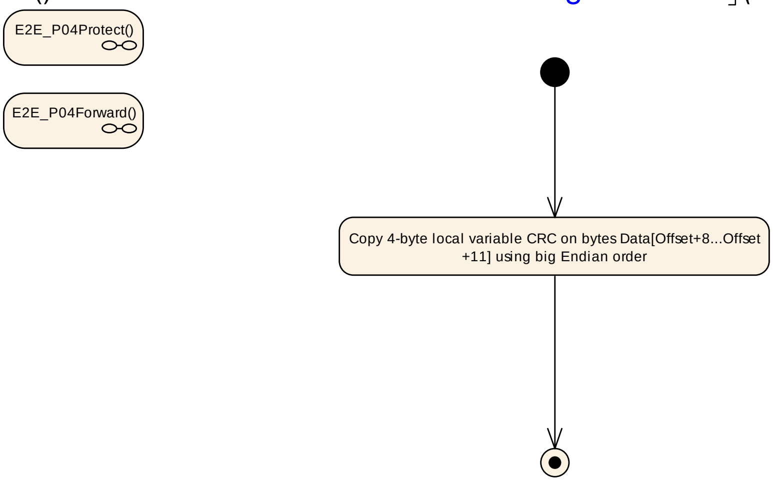

Figure 6.36: E2E Profile 4 Protect step "Write CRC"

[PRS_E2E_00369] ⌈The step "Increment Counter" in E2E_P04Protect() and E2E_P04Forward() shall behave as shown in Figure 6.37.⌋*(RS_E2E_08539)*

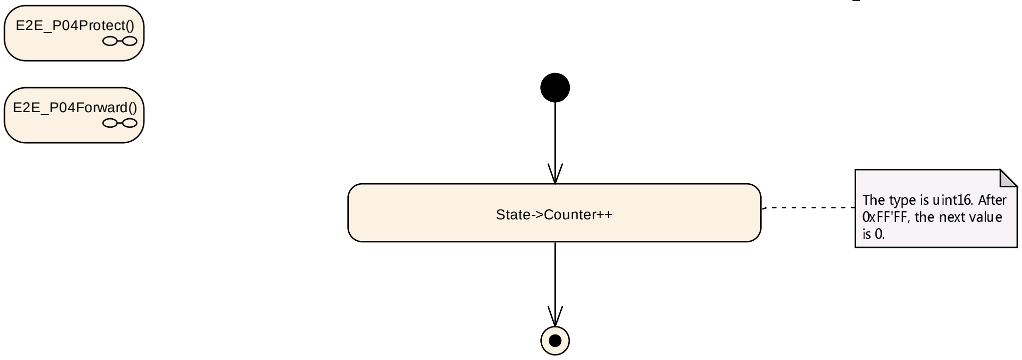

Figure 6.37: E2E Profile 4 Protect step "Increment Counter"

6.5.9 E2E_P04Forward

The E2E_P04Forward() function of E2E Profile 4 is called by a SW-C in order to protect its application data and forward an received E2E-Status for use cases like translation of signal based to service oriented communication. If the received E2E status equals E2E_P_OK the behavior of the function shall be the same like E2E_P04Protect(). The function E2E_P04Forward() performs the steps as specified by the following four diagrams in this section.

[PRS_E2E_00615] Draft ⌈The function E2E_P04Forward() shall have the overall behavior as shown in Figure 6.38.⌋*(RS_E2E_08539)*

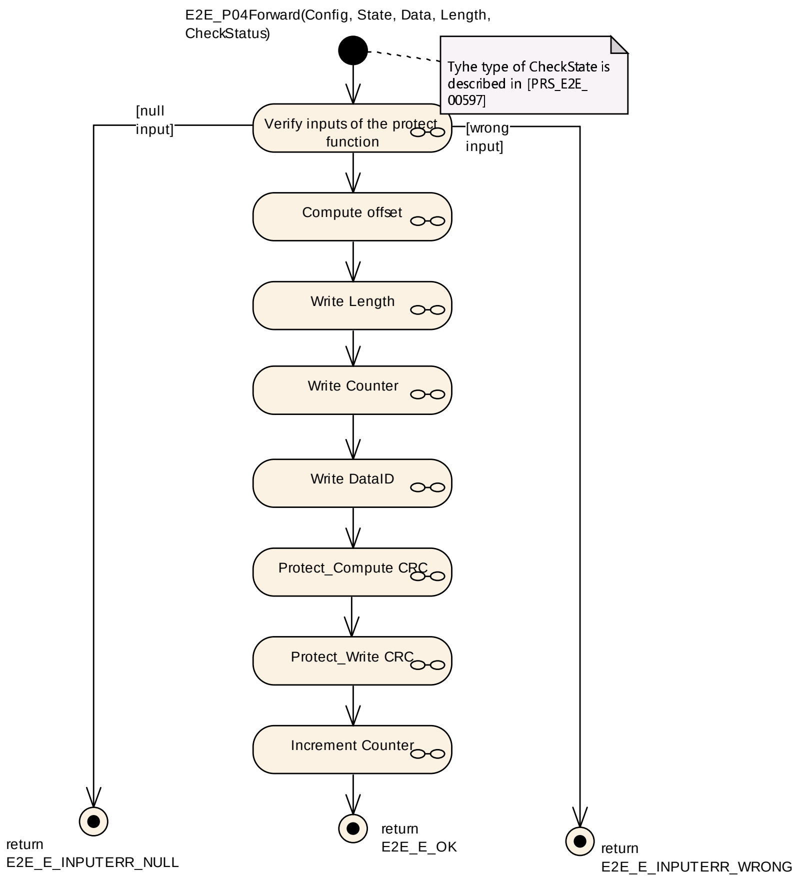

Figure 6.38: E2E Profile 4 Forward

Following steps are described in Section in Section 6.5.8

- ”Compute Offset” see [PRS_E2E_00376]
- ”Write Length” see [PRS_E2E_00364]
- ”Compute CRC” see [PRS_E2E_00367]
- ”Write CRC” see [PRS_E2E_00368]
- ”Increment Counter” see [PRS_E2E_00369]

[PRS_E2E_00616] Draft ⌈The step ”Verify inputs of the forward function” in E2E_P04Forward() shall behave as shown in Figure 6.39.⌋*(RS_E2E_08539)*

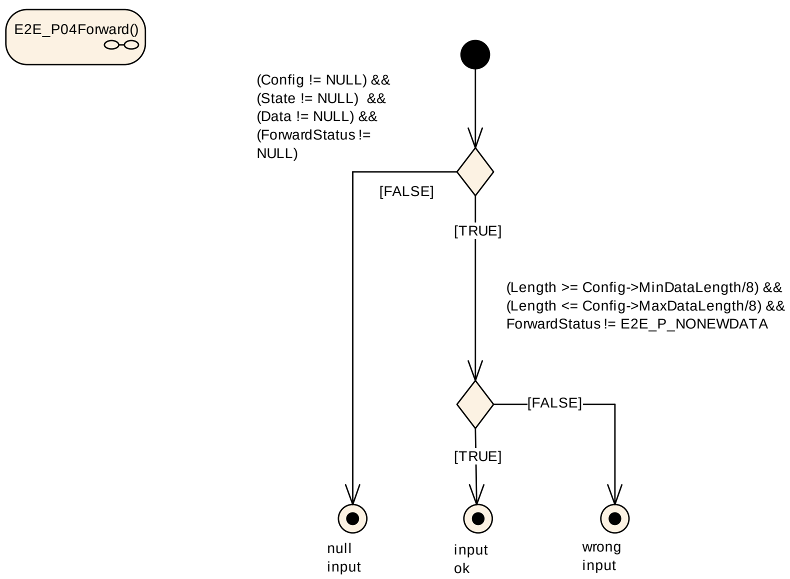

Figure 6.39: E2E Profile 4 Forward step "Verify inputs of the forward function"

[PRS_E2E_00617] Draft ⌈The step "Write Counter" in E2E_P04Forward() shall behave as shown in Figure 6.40.⌋ *(RS_E2E_08539)*

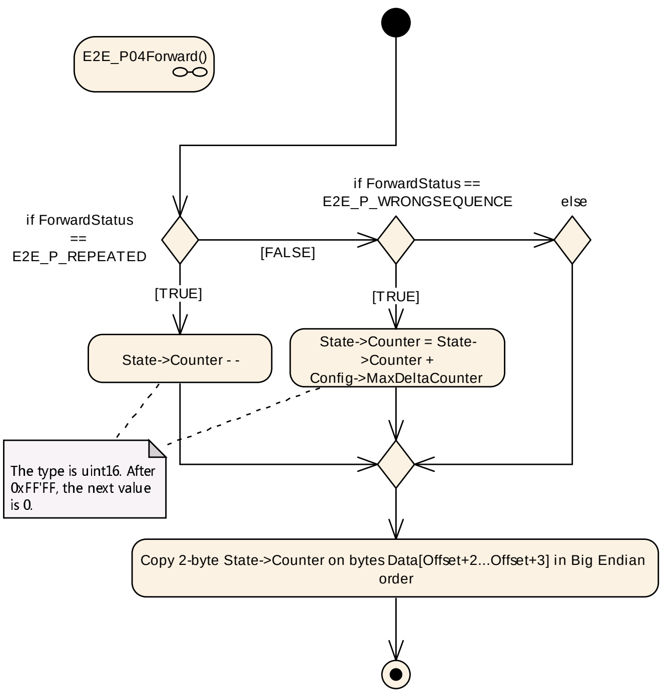

Figure 6.40: E2E Profile 4 Forward step "Write Counter"

[PRS_E2E_00618] **Draft** ⌈The step "Write DataID" in E2E_P04Forward() shall behave as shown in Figure 6.41.⌋*(RS_E2E_08539)*

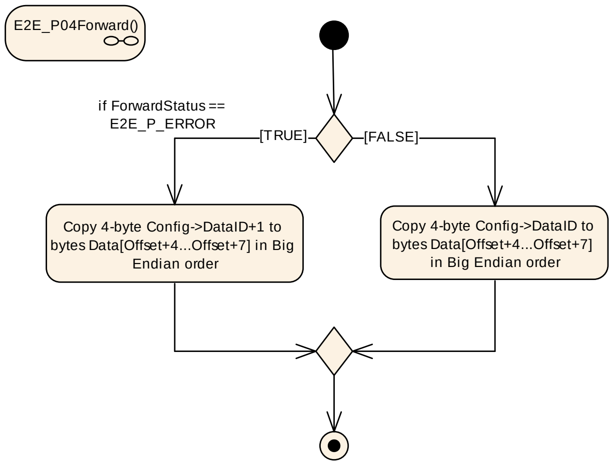

Figure 6.41: E2E Profile 4 Forward step "Write DataID"

6.5.10 E2E_P04Check

The function E2E_P04Check performs the actions as as specified by the following seven diagrams in this section and according to diagram PRS_E2EProtocol_00367.

[PRS_E2E_00355] ⌈The function E2E_P04Check() shall have the overall behavior as shown in Figure 6.42.⌋*(RS_E2E_08539)*

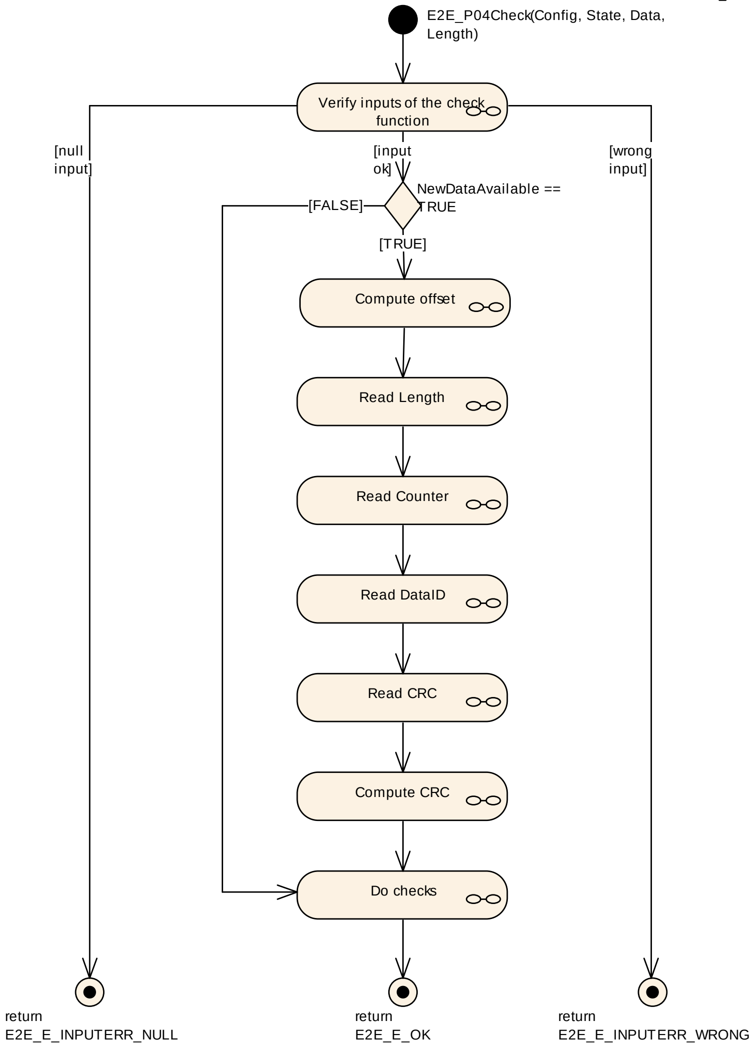

Figure 6.42: E2E Profile Check

[PRS_E2E_00356] ⌈The step "Verify inputs of the check function" in E2E_P04Check() shall behave as shown in Figure 6.43.⌋*(RS_E2E_08539)*

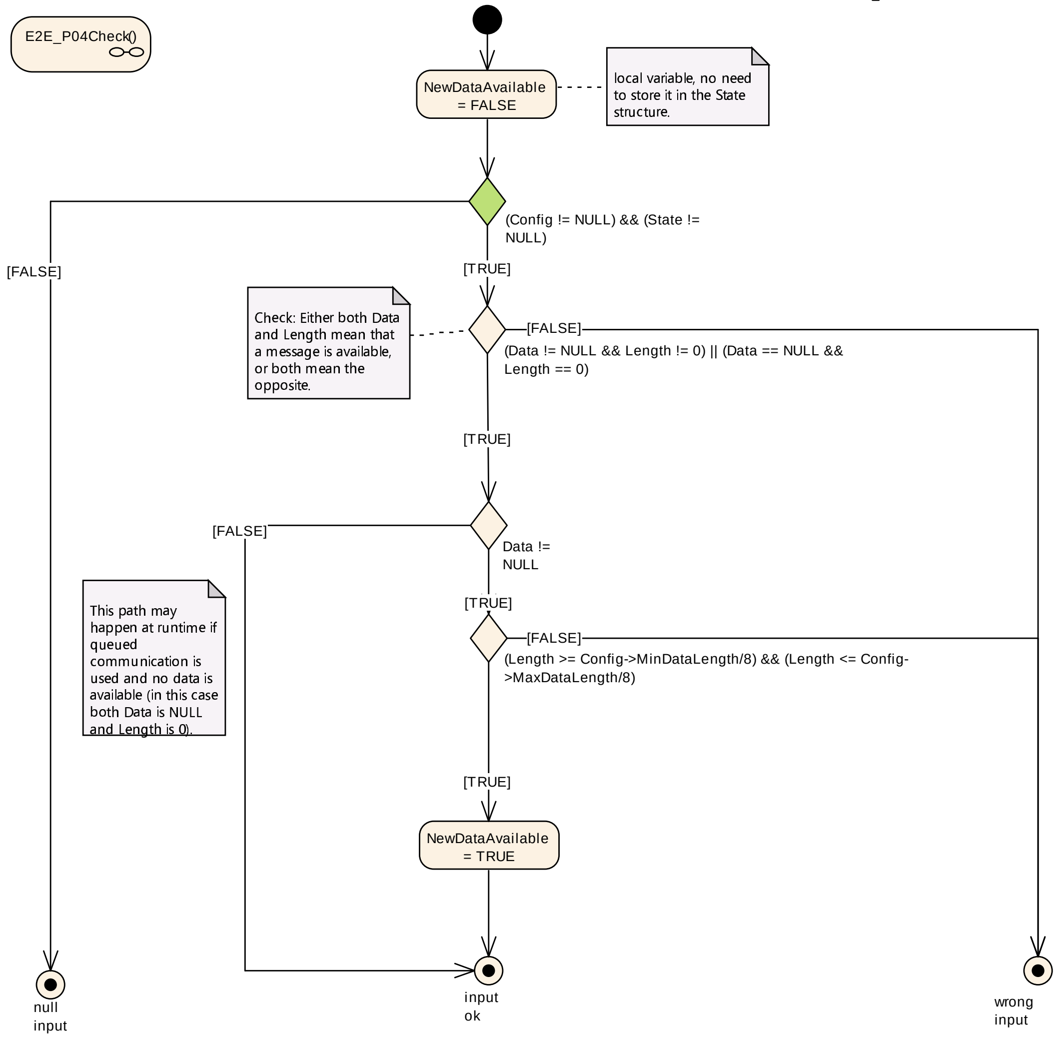

Figure 6.43: E2E Profile Check step 'Verify inputs of the check function"

[PRS_E2E_00357] ⌈The step "Read Length" in E2E_P04Check() shall behave as shown in Figure 6.44.⌋*(RS_E2E_08539)*

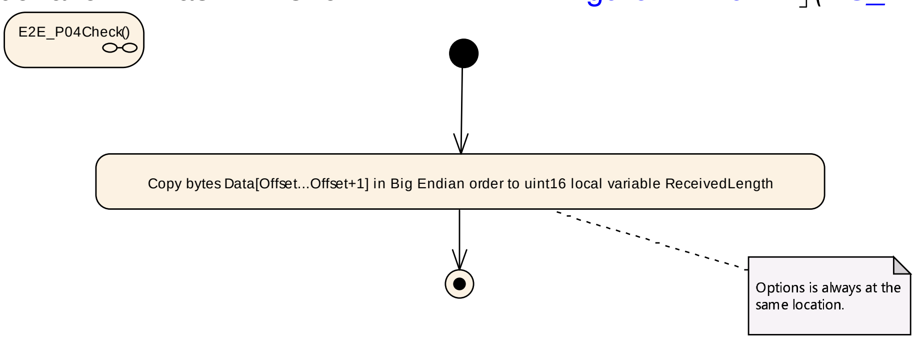

Figure 6.44: E2E Profile Check step "Read Length"

[PRS_E2E_00358] ⌈The step "Read Counter" in E2E_P04Check() shall behave as shown in Figure 6.45.⌋*(RS_E2E_08539)*

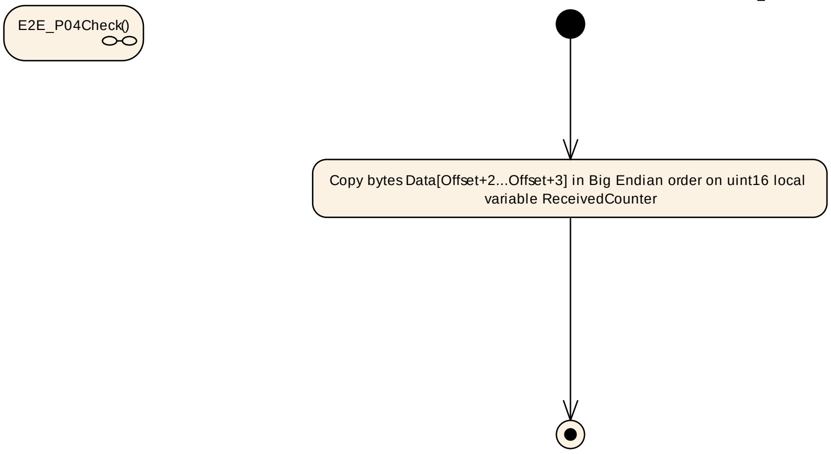

Figure 6.45: E2E Profile Check step "Read Counter"

[PRS_E2E_00359] ⌈The step "Read DataID" in E2E_P04Check() shall behave as shown in Figure 6.46.⌋*(RS_E2E_08539)*

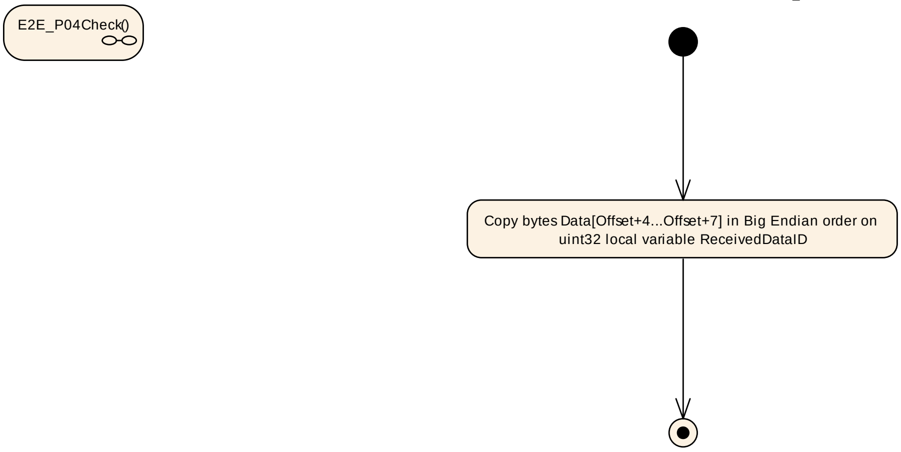

Figure 6.46: E2E Profile Check step "Read DataID"

[PRS_E2E_00360] ⌈The step ”Read CRC” in E2E_P04Check() shall behave as shown in Figure 6.47.⌋ *(RS_E2E_08539)*

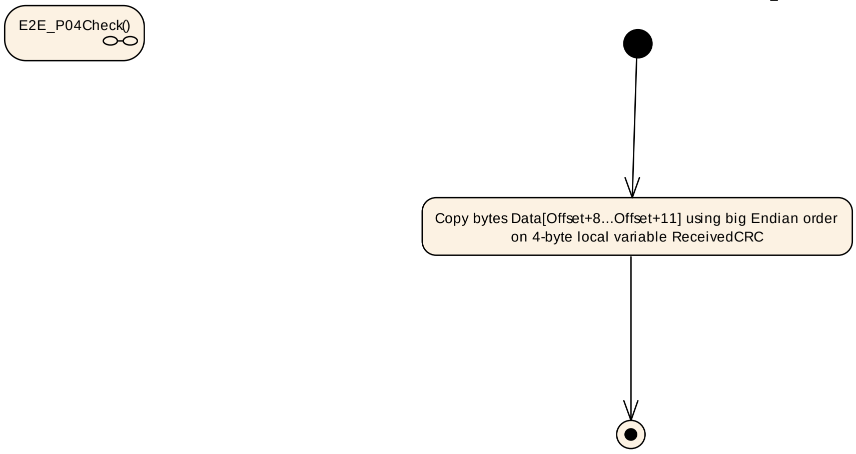

Figure 6.47: E2E Profile Check step ”Read CRC”

[PRS_E2E_00361] ⌈The step ”Do Checks” in E2E_P04Check() shall behave as shown in Figure 6.48.⌋ *(RS_E2E_08539)*

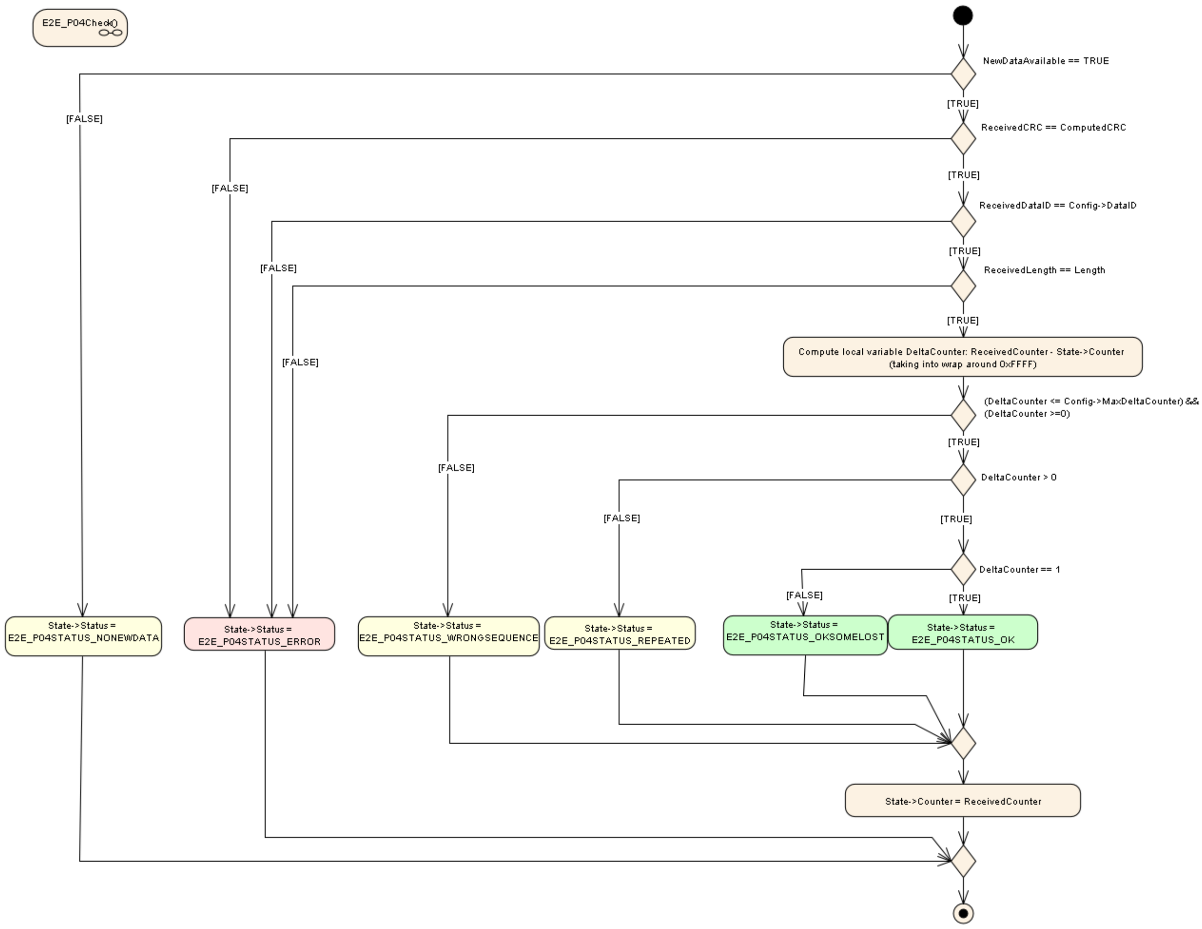

Figure 6.48: E2E Profile 4 Check step ”Do Checks”

6.5.10.1 Profile 4 Check Status Enumeration

[PRS_E2E_00590] ⌈The step "Do Checks" in E2E_P04Check shall set State->Status to one of the following enumeration values (see Table 6.22).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_P04STATUS_OK	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented by 1).
E2E_P04STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed. This may be considered similar to E2E_P04STATUS_REPEATED.
E2E_P04STATUS_ERROR	Error	Error not related to counters occurred (e.g. wrong crc, wrong length, wrong options, wrong Data ID).
E2E_P04STATUS_REPEATED	Error	The checks of the Data in this cycle were successful, with the exception of the repetition.
E2E_P04STATUS_OKSOMELOST	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented within the allowed configured delta).
E2E_P04STATUS_WRONGSEQUENCE	Error	The checks of the Data in this cycle were successful, with the exception of counter jump, which changed more than the allowed delta

Table 6.22: E2E Profile 4 Check Status Enumeration

6.5.11 E2E Profile 4 Protocol Examples

The default configuration assumed for the following examples, if not otherwise stated to be different:

E2E_P04ConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000
MinDataLength	96
MaxDataLength	32768
MaxDeltaCounter	1

Table 6.23: E2E Profile 4 protocol example configuration

E2E_P04ProtectStateType field	Value
Counter	0

Table 6.24: E2E Profile 4 example state initialization

Result data of E2E_P04Protect() with short data length (length 16 bytes, means 4 actual data bytes), offset = 0, counter = 0:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x00	0x10	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Length*		*Counter*		*DataID*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x34	0xea	0x4b	0xff	0x00	0x00	0x00	0x00
Field	*CRC*				*Data*			

Table 6.25: E2E Profile 4 example short

Result data of E2E_P04Protect() with minimum data length (4 data bytes), offset = 64 (as with SOME/IP header use case), datalength = 24, counter = 0:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x00	0x18	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Length*		*Counter*		*DataID*			
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0xe2	0x4d	0x10	0xfa	0x00	0x00	0x00	0x00
Field	*CRC*				*Data*			

Table 6.26: E2E Profile 4 example short with SOME/IP use case

6.6 Specification of E2E Profile 5

[PRS_E2E_00394] ⌈Profile 5 shall provide the following control fields, transmitted at runtime together with the protected data: Counter, CRC, Data ID (see Table 6.27).⌋ *(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Control field	Description
Counter	8 bits. (explicitly sent)
CRC	16 bits, polynomial in normal form 0x1021 (Autosar notation), provided by CRC library. (explicitly sent)
Data ID	16 bits, unique system-wide. (implicitly sent)E2E

Table 6.27: E2E Profile 5 mechanisms

The E2E mechanisms can detect the following faults or effects of faults:

Fault	Main safety mechanisms
Repetition of information	Counter
Loss of information	Counter
Delay of information	Counter
Masquerading	Data ID, CRC
Incorrect addressing	Data ID
Incorrect sequence of information	Counter
Corruption of information	CRC
Asymmetric information sent from a sender to multiple receivers	CRC (to detect corruption at any of receivers)
Information from a sender received by only a subset of the receivers	Counter (loss on specific receivers)
Blocking access to a communication channel	Counter (loss or timeout)

Table 6.28: Detectable communication faults using Profile 5

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[2].

6.6.1 Data Layout

6.6.1.1 User data layout

In the E2E Profile 5, the user data layout (of the data to be protected) is not constrained by E2E Profile 5 - there is only a requirement, that the length of data to be protected is multiple of 1 byte.

6.6.1.2 Header layout

The header of the E2E Profile 5 has one fixed layout, as follows:

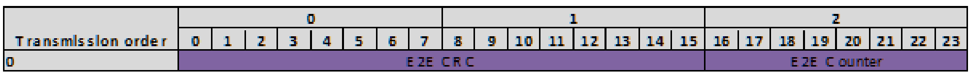

Figure 6.49: E2E Profile 5 header

The bit numbering shown above represents the order in which bits are transmitted. The E2E header fields (e.g. CRC) are encoded like in CAN and FlexRay, i.e.:

1. Little Endian (least significant byte fist) applicable for both implicit and explicit header fields - imposed by profile
2. MSB Fist (most significant bit within byte first) - imposed by FlexrayCAN bus.

6.6.2 Counter

In E2E Profile 5, the counter is initialized, incremented, reset and checked by E2E profile. The counter is not manipulated or used by the caller of the E2E Supervision.

[PRS_E2E_00397] ⌈In E2E Profile 5, on the sender side, for the first transmission request of a data element the counter shall be initialized with 0 and shall be incremented by 1 for every subsequent send request. When the counter reaches the maximum value (0xFF), then it shall restart with 0 for the next send request.⌋*(RS_E2E_08539)*

Note that the counter value 0xFF is not reserved as a special invalid value, but it is used as a normal counter value.

In E2E Profile 5, on the receiver side, by evaluating the counter of received data against the counter of previously received data, the following is detected:

1. Repetition:

   a. no new data has arrived since last invocation of E2E Supervision check function,

   b. the data is repeated

2. OK:

   a. counter is incremented by one (i.e. no data lost),

   b. counter is incremented more than by one, but still within allowed limits (i.e. some data lost),

3. Error: a. counter is incremented more than allowed (i.e. too many data lost).

Case 1 corresponds to the failed alive counter check, and case 3 correspond to failed sequence counter check.

The above requirements are specified in more details by the UML diagrams in the following document sections.

6.6.3 Data ID

The unique Data IDs are to verify the identity of each transmitted safety-related data element.

[PRS_E2E_00399] ⌈In the E2E Profile 5, the Data ID shall be implicitly transmitted, by adding the Data ID after the user data in the CRC calculation.⌋*(RS_E2E_08539)*

The Data ID is not a part of the transmitted E2E header (similar to Profile 2 and 6).

[PRS_E2E_UC_00463] ⌈In the E2E profile 5, the Data IDs shall be globally unique within the network of communicating system (made of several ECUs each sending different data).⌋*(RS_E2E_08539)*

In case of usage of E2E Supervision for protecting data elements (i.e invocation from RTE), due to multiplicity of communication (1:1 or 1:N), a consumer of a data element expects only a specific data element, which is checked by E2E Supervision using Data ID.

In case of usage of E2E Supervision for protecting messages (i.e. invocation from COM), the receiver COM expects at a reception only a specific message, which is checked by E2E Supervision using Data ID.

6.6.4 Length

In Profile 5 there is no explicit transmission of the length.

6.6.5 CRC

E2E Profile 5 uses a 16-bit CRC, to ensure a sufficient detection rate and sufficient Hamming Distance.

[PRS_E2E_00400] ⌈E2E Profile 5 shall use the Crc_CalculateCRC16() function of the SWS CRC Supervision for calculating the CRC (Polynomial: 0x1021; Autosar notation).⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_00401] ⌈In E2E Profile 5, the CRC shall be calculated over the entire E2E header (excluding the CRC bytes), including the user data extended at the end with the Data ID.⌋*(RS_E2E_08539, RS_E2E_08536)*

6.6.6 Timeout detection

The previously mentioned mechanisms (for Profile 5: CRC, Counter, Data ID) enable to check the validity of received data element, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for Data Elements or respectively messages, but instead if the receiver reads the currently available data (i.e. checks if new data is available). Then, by means of the counter, the receiver can detect loss of communication and timeouts.

The attribute State->NewDataAvailable == FALSE means that the transmission medium (e.g RTE) reports that no new data element is available at the transmission medium. The attribute State->Status = E2E_P05STATUS_REPEATED means that 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 an unavailability of valid data that was updated since the previous cycle.

6.6.7 E2E_P05Protect

The function E2E_P05Protect() performs the steps as specified by the following six diagrams in this section.

[PRS_E2E_00403] ⌈The function E2E_P05Protect() shall have the overall behavior as shown in Figure 6.50.⌋*(RS_E2E_08539)*

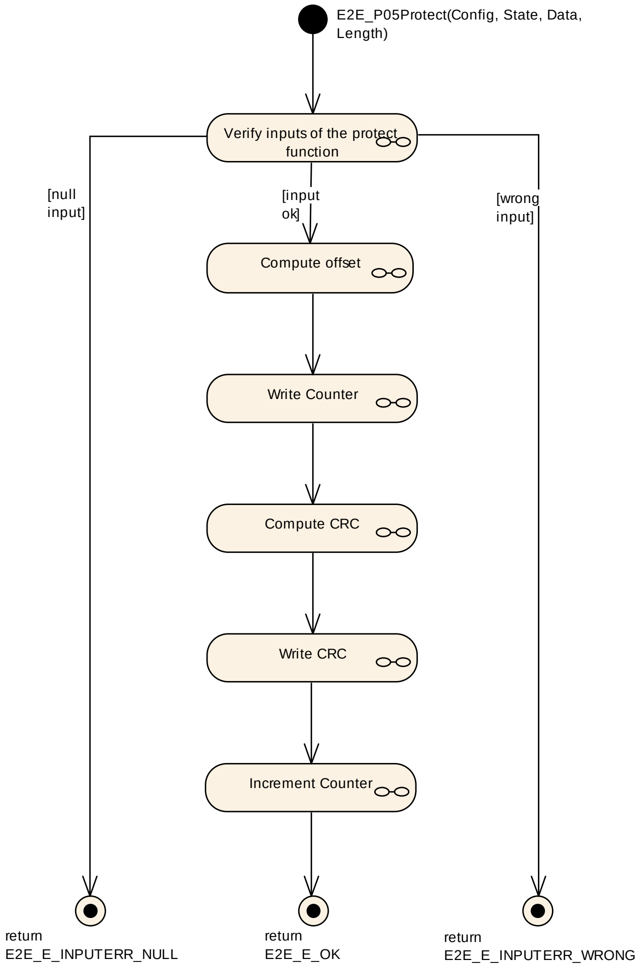

Figure 6.50: E2E Profile 5 Protect

[PRS_E2E_00404] ⌈The step "Verify inputs of the protect function" in E2E_P05Protect() shall behave as shown in Figure 6.51.⌋*(RS_E2E_08539)*

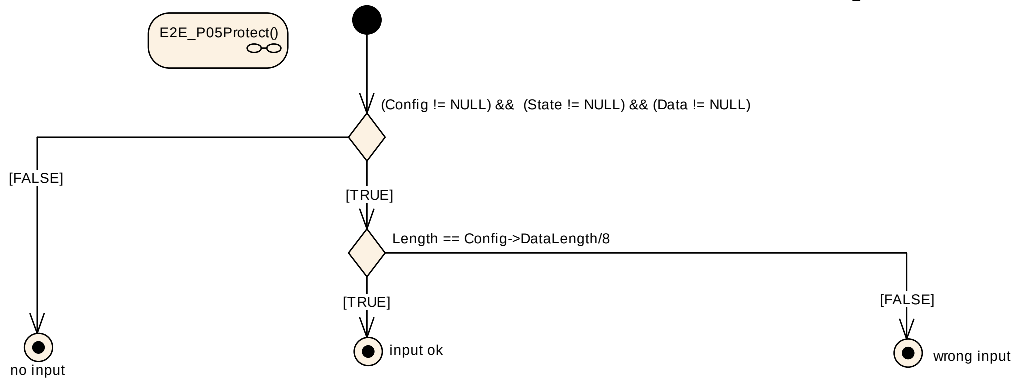

Figure 6.51: E2E Profile 5 Protect step "Verify inputs of the protect function"

[PRS_E2E_00469] ⌈The step "Compute offset" in E2E_P05Protect(), E2E_P05Forward() and E2E_P05Check() shall behave as shown in Figure 6.52.⌋*(RS_E2E_08539)*

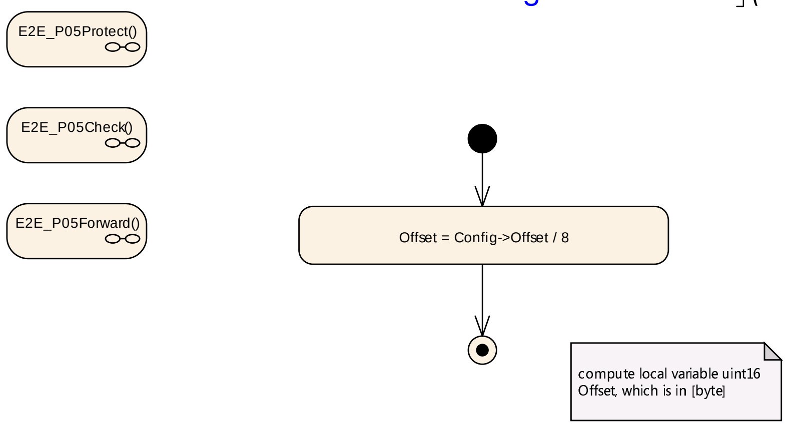

Figure 6.52: E2E Profile 5 Protect step "Compute offset"

[PRS_E2E_00405] ⌈The step "Write Counter" in E2E_P05Protect() shall behave as shown in Figure 6.53.⌋*(RS_E2E_08539)*

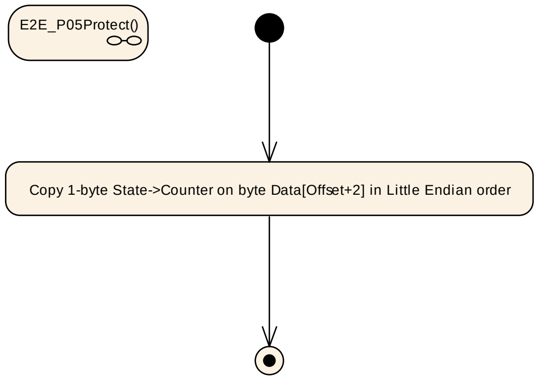

Figure 6.53: E2E Profile 5 Protect step "Write Counter"

[PRS_E2E_00406] ⌈The step "Compute CRC" in E2E_P05Protect() and in E2E_P05Check shall behave as shown in Figure 6.54.⌋*(RS_E2E_08539)*

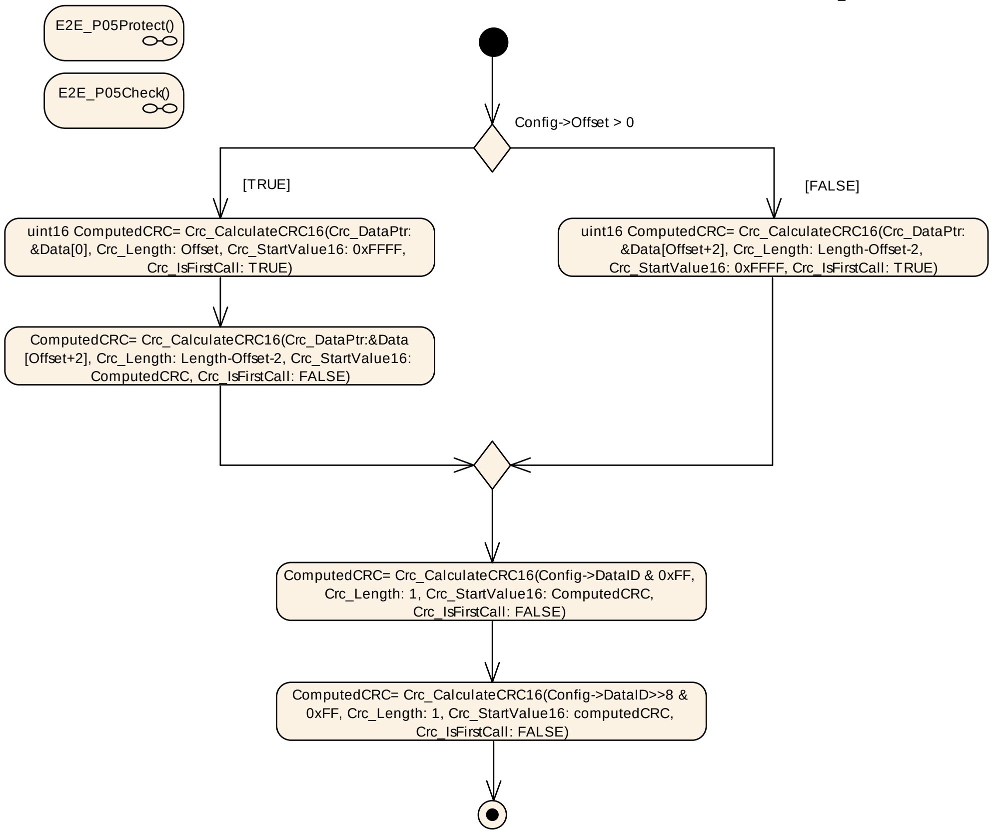

Figure 6.54: E2E Profile 5 Protect and Check step "Compute CRC"

[PRS_E2E_00407] ⌈The step "Write CRC" in E2E_P05Protect() and E2E_P05Forward() shall behave as shown in Figure 6.55.⌋*(RS_E2E_08539)*

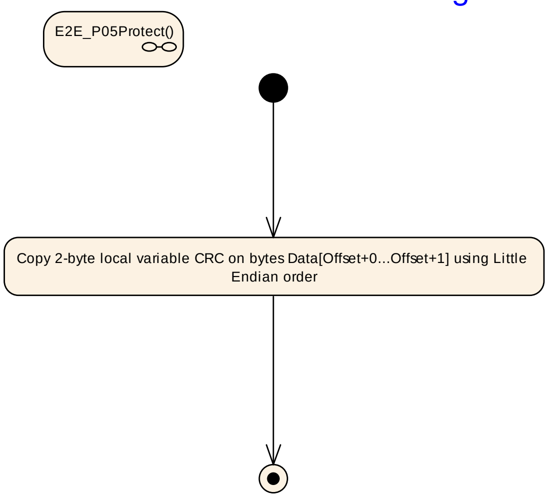

Figure 6.55: E2E Profile 5 Protect step "Write CRC"

[PRS_E2E_00409] ⌈The step "Increment Counter" in E2E_P05Protect() and E2E_P05Forward() shall behave as shown in Figure 6.56.⌋*(RS_E2E_08539)*

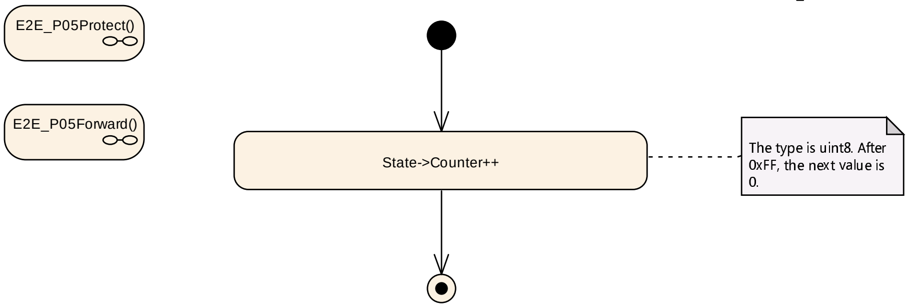

Figure 6.56: E2E Profile 5 Protect step "Increment Counter"

6.6.8 E2E_P05Forward

The E2E_P05Forward() function of E2E Profile 5 is called by a SW-C in order to protect its application data and forward an received E2E-Status for use cases like translation of signal based to service oriented communication. If the received E2E status equals E2E_P_OK the behavior of the function shall be the same like E2E_P05Protect(). The function E2E_P05Forward() performs the steps as specified by the following four diagrams in this section.

[PRS_E2E_00639] Draft ⌈The function E2E_P05Forward() shall have the overall behavior as shown in Figure 6.57.⌋*(RS_E2E_08539)*

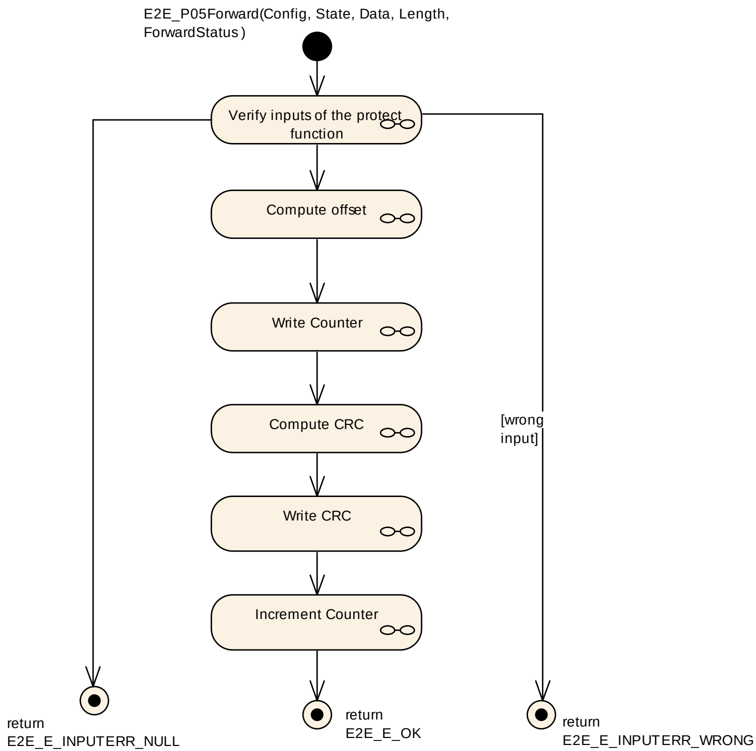

Figure 6.57: E2E Profile 5 Forward

Following steps are described in Section in Section 6.6.7

- ”Compute Offset” see [PRS_E2E_00469]
- ”Write CRC” see [PRS_E2E_00407]
- ”Increment Counter” see [PRS_E2E_00409]

[PRS_E2E_00619] Draft ⌈The step ”Verify inputs of the forward function” in E2E_P05Forward() shall behave as shown in Figure 6.58.⌋*(RS_E2E_08539)*

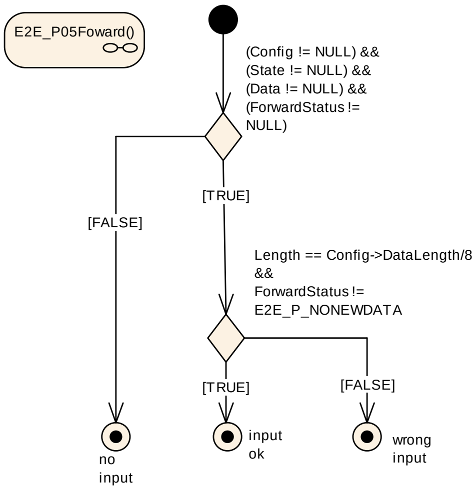

Figure 6.58: E2E Profile 5 Forward step ”Verify inputs of the forward function”

[PRS_E2E_00620] Draft ⌈The step ”Write Counter” in E2E_P05Forward() shall behave as shown in Figure 6.59.⌋*(RS_E2E_08539)*

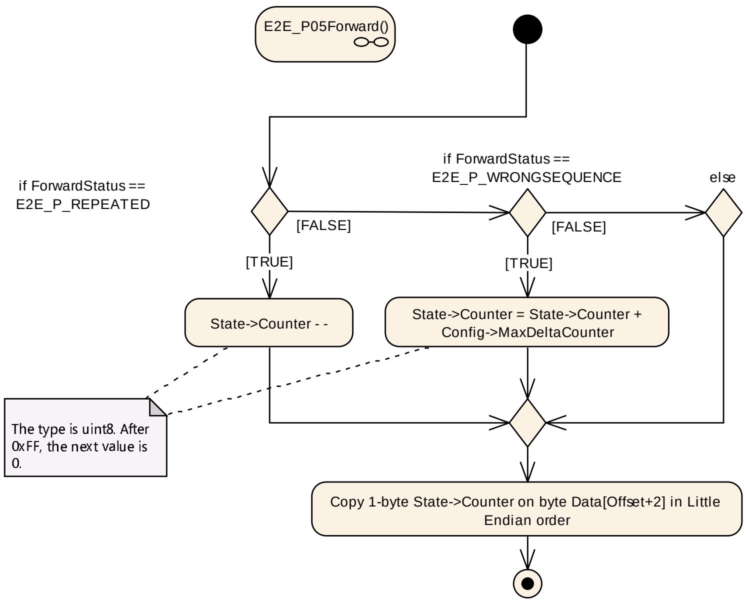

Figure 6.59: E2E Profile 5 Forward step ”Write Counter”

[PRS_E2E_00621] **Draft** ⌈The step "Compute CRC" in E2E_P05Forward() shall behave as shown in Figure 6.60.⌋*(RS_E2E_08539)*

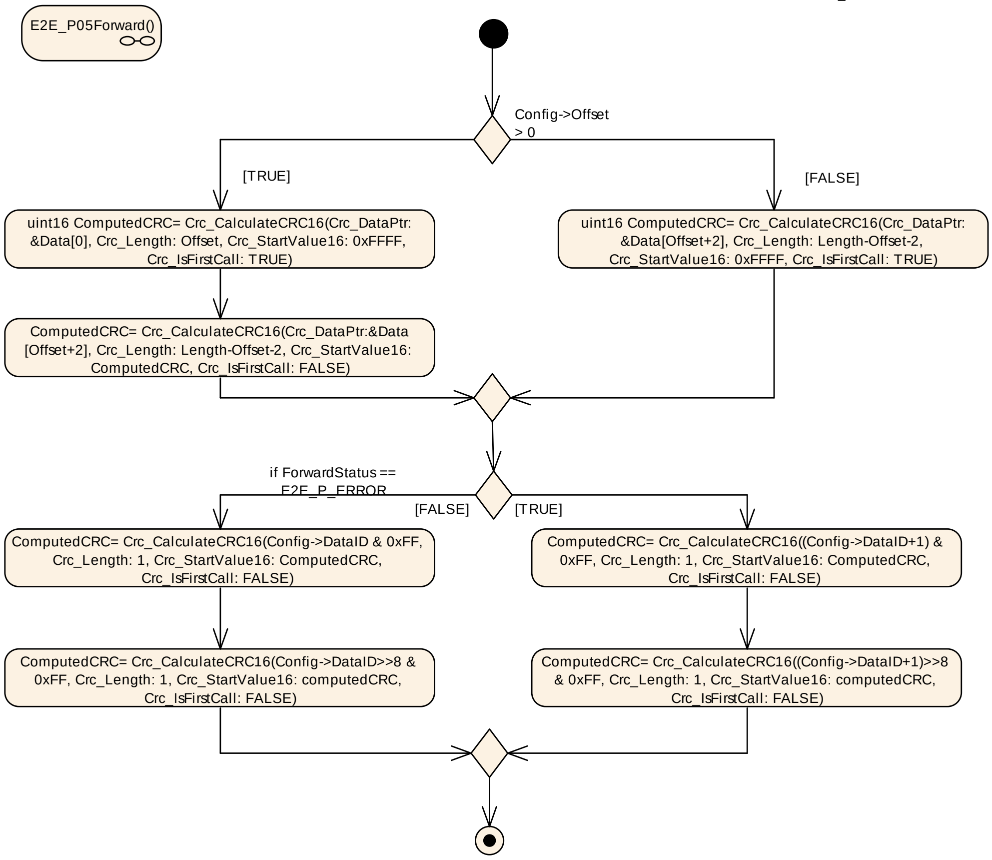

Figure 6.60: E2E Profile 5 Forward step "Compute CRC"

6.6.9 E2E_P05Check

The function E2E_P05Check performs the actions as specified by the following six diagrams in this section.

[PRS_E2E_00411] ⌈The function E2E_P05Check() shall have the overall behavior as shown in Figure 6.61.⌋*(RS_E2E_08539)*

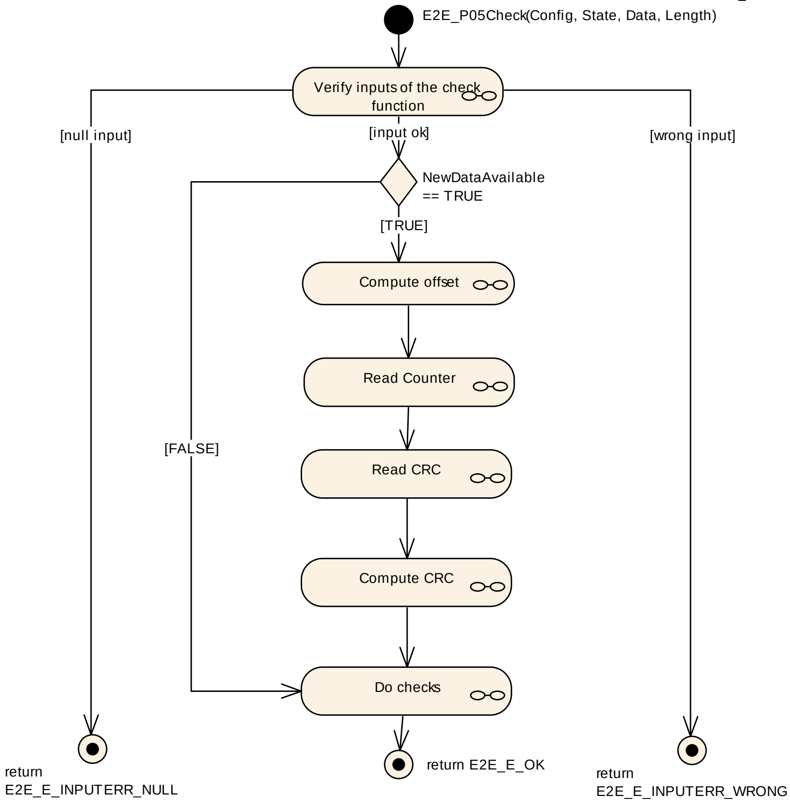

Figure 6.61: E2E Profile 5 Check

[PRS_E2E_00412] ⌈The step ”Verify inputs of the check function” in E2E_P05Check() shall behave as shown in Figure 6.62.⌋*(RS_E2E_08539)*

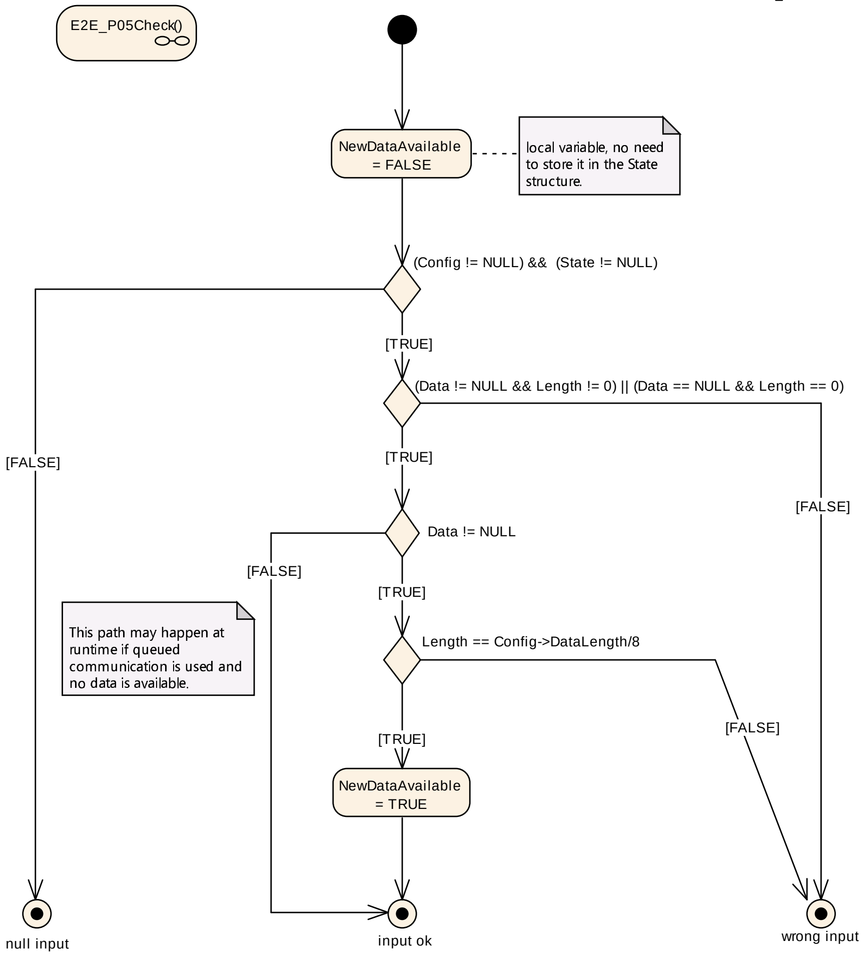

Figure 6.62

[PRS_E2E_00413] ⌈The step ”Read Counter” in E2E_P05Check() shall behave as shown in Figure 6.63.⌋*(RS_E2E_08539)*

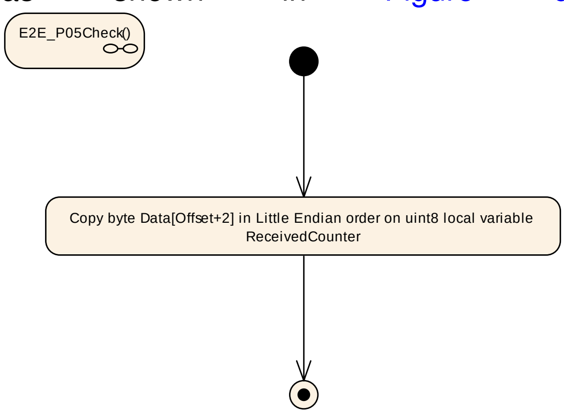

Figure 6.63: E2E Profile 5 Check step ”Read Counter”

[PRS_E2E_00414] ⌈The step ”Read CRC” in E2E_P05Check() shall behave as shown in Figure 6.64.⌋*(RS_E2E_08539)*

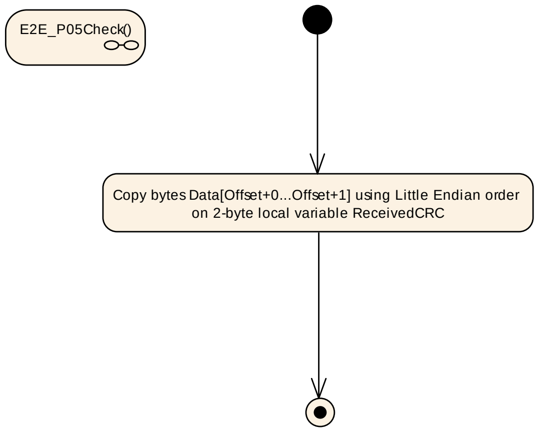

Figure 6.64: E2E Profile 5 Check step ”Read CRC”

[PRS_E2E_00416] ⌈The step ”Do Checks’ in E2E_P05Check() shall behave as shown in Figure 6.65.⌋*(RS_E2E_08539)*

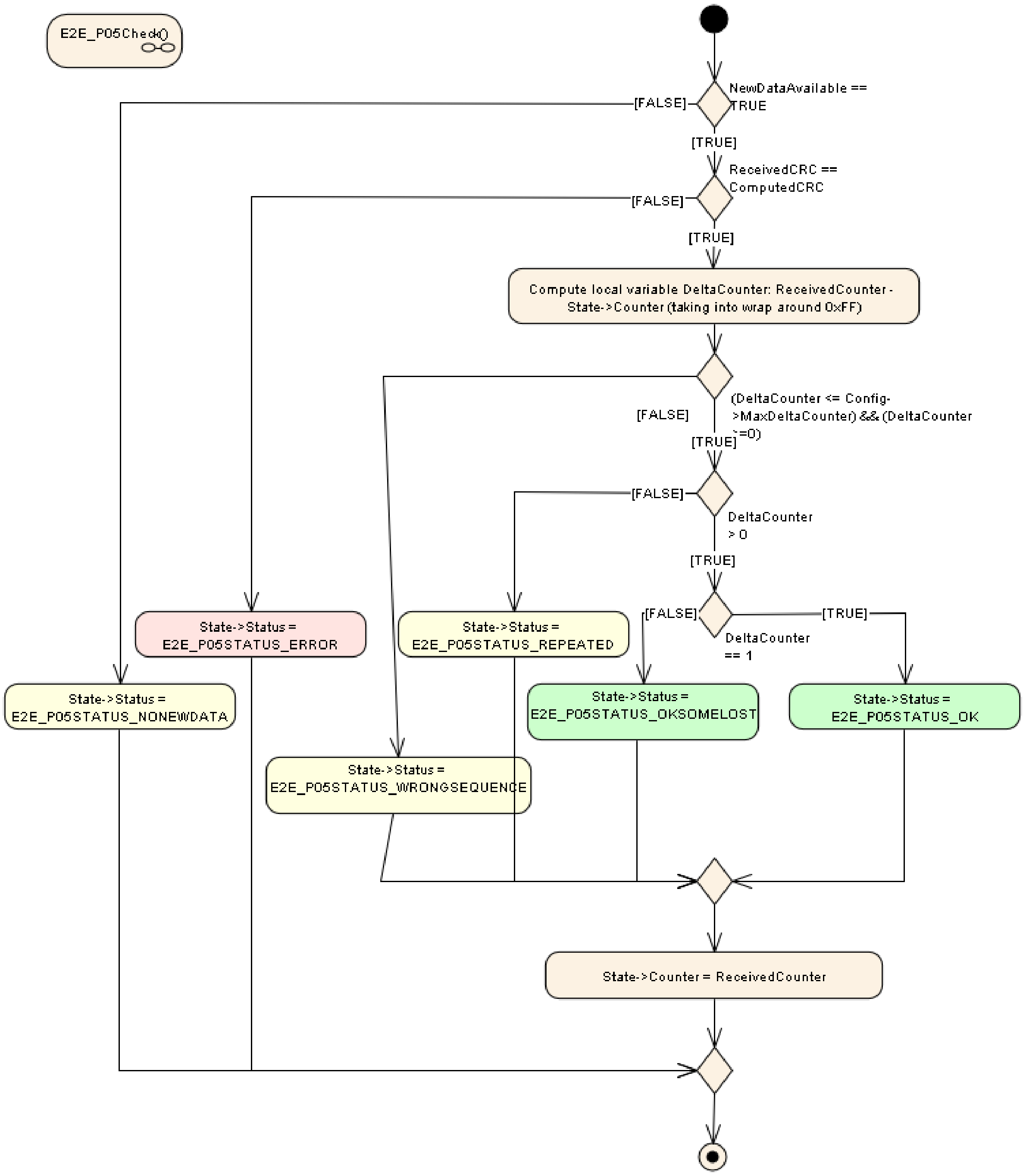

Figure 6.65: E2E Profile 5 Check step ”Do Checks”

6.6.9.1 Profile 5 Check Status Enumeration

[PRS_E2E_00591] ⌈The step ”Do Checks” in E2E_P05Check shall set State->Status to one of the following enumeration values (see Table 6.29).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_P05STATUS_OK	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented by 1).
E2E_P05STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed. This may be considered similar to E2E_P05STATUS_REPEATED.
E2E_P05STATUS_ERROR	Error	Error not related to counters occurred (e.g. wrong crc, wrong length, wrong options, wrong Data ID).
E2E_P05STATUS_REPEATED	Error	The checks of the Data in this cycle were successful, with the exception of the repetition.
E2E_P05STATUS_OKSOMELOST	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented within the allowed configured delta).
E2E_P05STATUS_WRONGSEQUENCE	Error	The checks of the Data in this cycle were successful, with the exception of counter jump, which changed more than the allowed delta

Table 6.29: E2E Profile 5 Check Status Enumeration

6.6.10 E2E Profile 5 Protocol Examples

The default configuration assumed for the following examples, if not otherwise stated to be different:

E2E_P05ConfigType field	Value
DataID	0x1234
Offset	0x0000
DataLength	24
MaxDeltaCounter	1

Table 6.30: E2E Profile 5 protocol example configuration

E2E_P05ProtectStateType field	Value
Counter	0

Table 6.31: E2E Profile 5 example state initialization

Result data of E2E_P05Protect() with short data length (length 8 bytes, with 5 actual data bytes), offset = 0, counter = 0:

Byte	0	1	2	3	4	5	6	7
Data	0x01c	0xca	0x00	0x00	0x00	0x00	0x00	0x00
Field	*CRC*		*Counter*	*Data*				

Table 6.32: E2E Profile 5 example short

Result data of E2E_P05Protect() with short data length (length 16 bytes, with 5 actual data bytes), offset = 64 (as with SOME/IP header use case), counter = 0:

Byte	0	1	2	3	4	5	6	7
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x28	0x91	0x00	0x00	0x00	0x00	0x00	0x00
Field	*CRC*		*Counter*	*Data*				

Table 6.33: E2E Profile 5 example short with SOME/IP use case

6.7 Specification of E2E Profile 6

[PRS_E2E_00479] ⌈Profile 6 shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID (see Table 6.34).⌋*(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Control field	Description
Length	16 bits, to support dynamic-size data. (explicitly sent)
Counter	8-bits. (explicitly sent)
CRC	16-bits, polynomial in normal form 0x1021 (Autosar notation), provided by CRC library. (explicitly sent)
Data ID	16-bits, unique system-wide. (implicitly sent)

Table 6.34: E2E Profile 6 mechanisms

The E2E mechanisms can detect the following faults or effects of faults:

Fault	Main safety mechanisms
Repetition of information	Counter
Loss of information	Counter
Delay of information	Counter
Insertion of information	Data ID
Masquerading	Data ID, CRC
Incorrect addressing	Data ID
Incorrect sequence of information	Counter

Corruption of information	CRC
Asymmetric information sent from a sender to multiple receivers	CRC (to detect corruption at any of receivers)
Information from a sender received by only a subset of the receivers	Counter (loss on specific receivers)
Blocking access to a communication channel	Counter (loss or timeout)

Table 6.35: Detectable communication faults using Profile 6

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[2].

6.7.1 Data Layout

6.7.1.1 User data layout

In the E2E Profile 6, the user data layout (of the data to be protected) is not constrained by E2E Profile 6 - there is only a requirement that the length of data to be protected is multiple of 1 byte.

6.7.1.2 Header layout

The header of the E2E Profile 6 has one fixed layout, as follows:

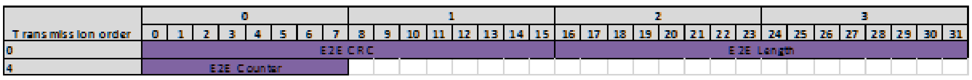

Figure 6.66: E2E Profile 6 header

The bit numbering shown above represents the order in which bits are transmitted. The E2E header fields (e.g. E2E Counter) are encoded as:

1. Big Endian (most significant byte fist), applicable for both implicit and explicit header fields - imposed by profile
2. LSB Fist (least significant bit within byte first) - imposed by TCP/IP bus

6.7.2 Counter

In E2E Profile 6, the counter is initialized, incremented, reset and checked by E2E profile. The counter is not manipulated or used by the caller of the E2E Supervision.

[PRS_E2E_00417] ⌈In E2E Profile 6, on the sender side, for the first transmission request of a data element the counter shall be initialized with 0 and shall be incremented by 1 for every subsequent send request. When the counter reaches the maximum value (0xFF), then it shall restart with 0 for the next send request.⌋*(RS_E2E_08539)*

Note that the counter value 0xFF is not reserved as a special invalid value, but it is used as a normal counter value.

In E2E Profile 6, on the receiver side, by evaluating the counter of received data against the counter of previously received data, the following is detected:

1. Repetition:

   a. no new data has arrived since last invocation of E2E Supervision check function,

   b. the data is repeated

2. OK:

   a. counter is incremented by one (i.e. no data lost),

   b. counter is incremented more than by one, but still within allowed limits (i.e. some data lost),

3. Error: a. counter is incremented more than allowed (i.e. too many data lost).

Case 1 corresponds to the failed alive counter check, and case 3 correspond to failed sequence counter check.

The above requirements are specified in more details by the UML diagrams in the following document sections.

6.7.3 Data ID

The unique Data IDs are to verify the identity of each transmitted safety-related data element.

[PRS_E2E_00419] ⌈In the E2E Profile 6, the Data ID shall be implicitly transmitted, by adding the Data ID after the user data in the CRC calculation.⌋*(RS_E2E_08539)*

The Data ID is not a part of the transmitted E2E header (similar to Profile 2 and 5).

[PRS_E2E_UC_00464] ⌈In the E2E profile 6, the Data IDs shall be globally unique within the network of communicating system (made of several ECUs each sending different data).⌋*(RS_E2E_08539)*

In case of usage of E2E Supervision for protecting data elements (i.e invocation from RTE), due to multiplicity of communication (1:1 or 1:N), a consumer of a data element expects only a specific data element, which is checked by E2E Supervision using Data ID.

In case of usage of E2E Supervision for protecting messages (i.e. invocation from COM), the receiver COM expects at a reception only a specific message, which is checked by E2E Supervision using Data ID.

6.7.4 Length

In Profile 6 the length field is introduced to support variable-size length - the Data [] array storing the serialized data can potentially have a different length in each cycle. In Profile 6 there is a explicit transmission of the length. The Length includes user data + E2E Header (CRC + Counter + Length).

6.7.5 CRC

E2E Profile 6 uses a 16-bit CRC, to ensure a sufficient detection rate and sufficient Hamming Distance.

[PRS_E2E_00420] ⌈E2E Profile 6 shall use the Crc_CalculateCRC16() function of the SWS CRC Supervision for calculating the CRC (Polynomial: 0x1021; Autosar notation).⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_00421] ⌈In E2E Profile 6, the CRC shall be calculated over the entire E2E header (excluding the CRC bytes), including the user data extended with the Data ID.⌋*(RS_E2E_08539, RS_E2E_08536)*

6.7.6 Timeout detection

The previously mentioned mechanisms (for Profile 6: CRC, Counter, Data ID, Length) enable to check the validity of received data element, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for Data Elements or respectively I-PDUs, but instead if the receiver reads the currently available data (i.e. checks if new data is available). Then, by means of the counter, the receiver can detect loss of communication and timeouts.

The attribute State->NewDataAvailable == FALSE means that the transmission medium (e.g RTE) reports that no new data element is available at the transmission medium. The attribute State->Status = E2E_P06STATUS_REPEATED means that 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 an unavailability of valid data that was updated since the previous cycle.

6.7.7 E2E_P06Protect

The function E2E_P06Protect() performs the steps as specified by the following seven diagrams in this section.

[PRS_E2E_00423] ⌈The function E2E_P06Protect() shall have the overall behavior as shown in Figure 6.67.⌋ *(RS_E2E_08539)*

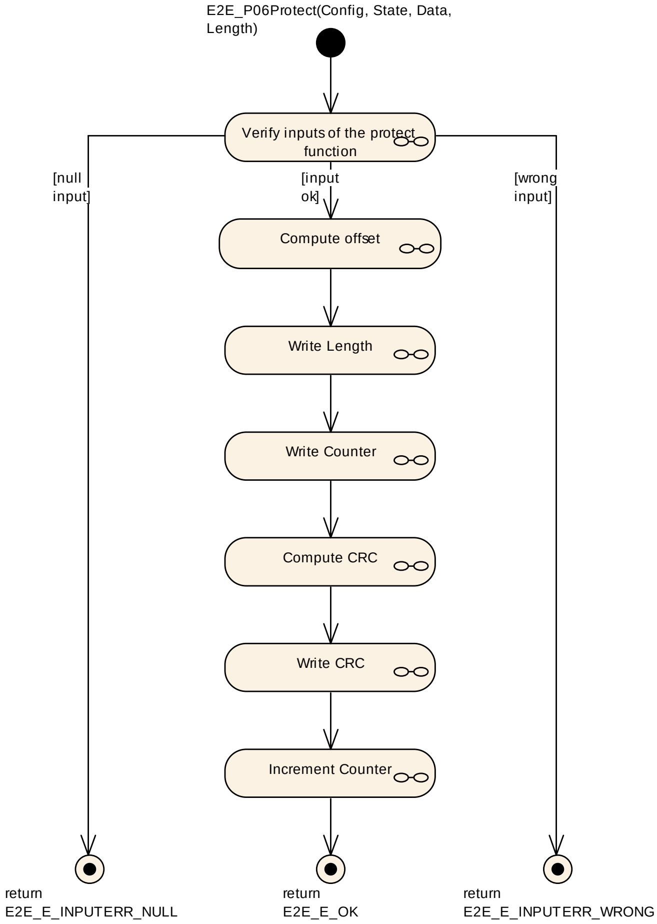

Figure 6.67: E2E Profile 6 Protect

[PRS_E2E_00424] ⌈The step "Verify inputs of the protect function" in E2E_P06Protect() shall behave as shown in Figure 6.68.⌋*(RS_E2E_08539)*

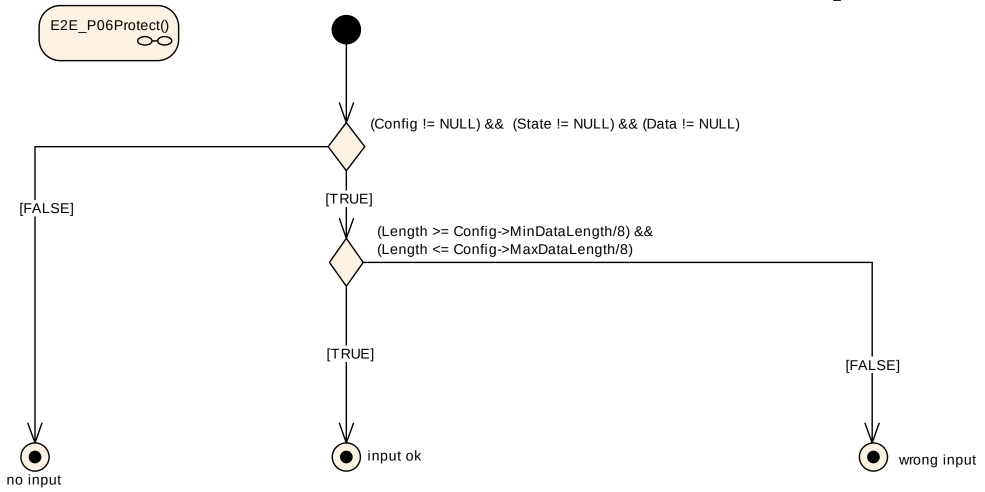

Figure 6.68: E2E Profile 6 Protect step "Verify inputs of the protect function"

[PRS_E2E_00470] ⌈The step "Compute offset" in E2E_P06Protect(), E2E_P06Forward() and E2E_P06Check() shall behave as shown in Figure 6.69.⌋*(RS_E2E_08539)*

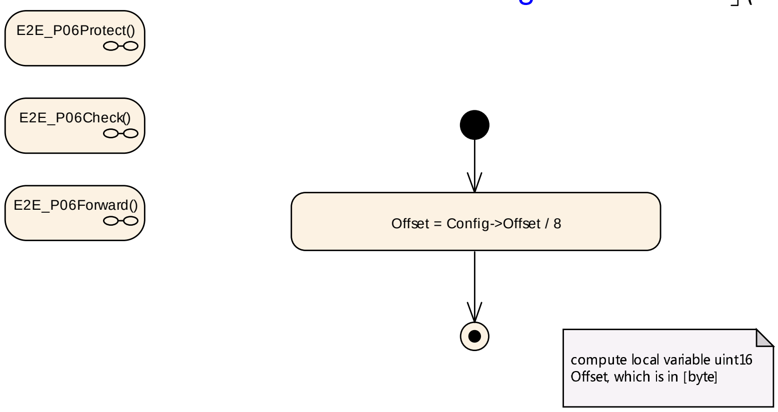

Figure 6.69: E2E Profile 6 Protect, Forward and Check step "Compute offset"

[PRS_E2E_00425] ⌈The step "Write Length" in E2E_P06Protect() and E2E_P06Forward() shall behave as shown in Figure 6.70.⌋*(RS_E2E_08539)*

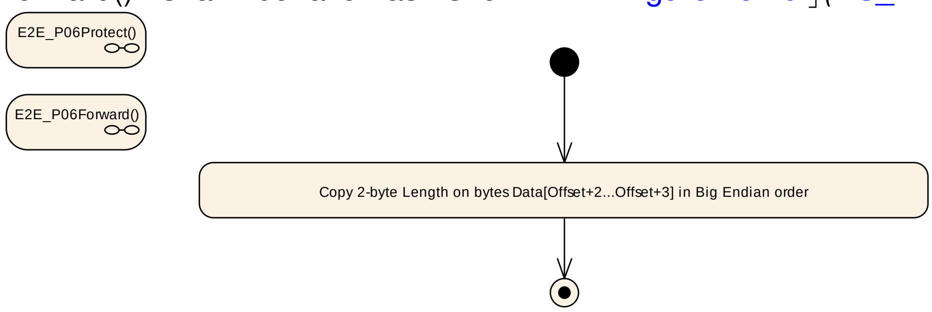

Figure 6.70: E2E Profile 6 Protect and Forward step "Write Length"

[PRS_E2E_00426] ⌈The step ”Write Counter” in E2E_P06Protect() shall behave as shown in Figure 6.71.⌋*(RS_E2E_08539)*

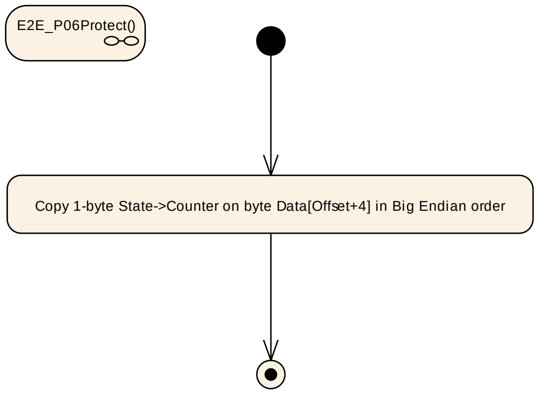

Figure 6.71: E2E Profile 6 Protect step ”Write Counter”

[PRS_E2E_00427] ⌈The step ”Compute CRC” in E2E_P06Protect() and E2E_P06Check() shall behave as shown in Figure 6.72.⌋*(RS_E2E_08539)*

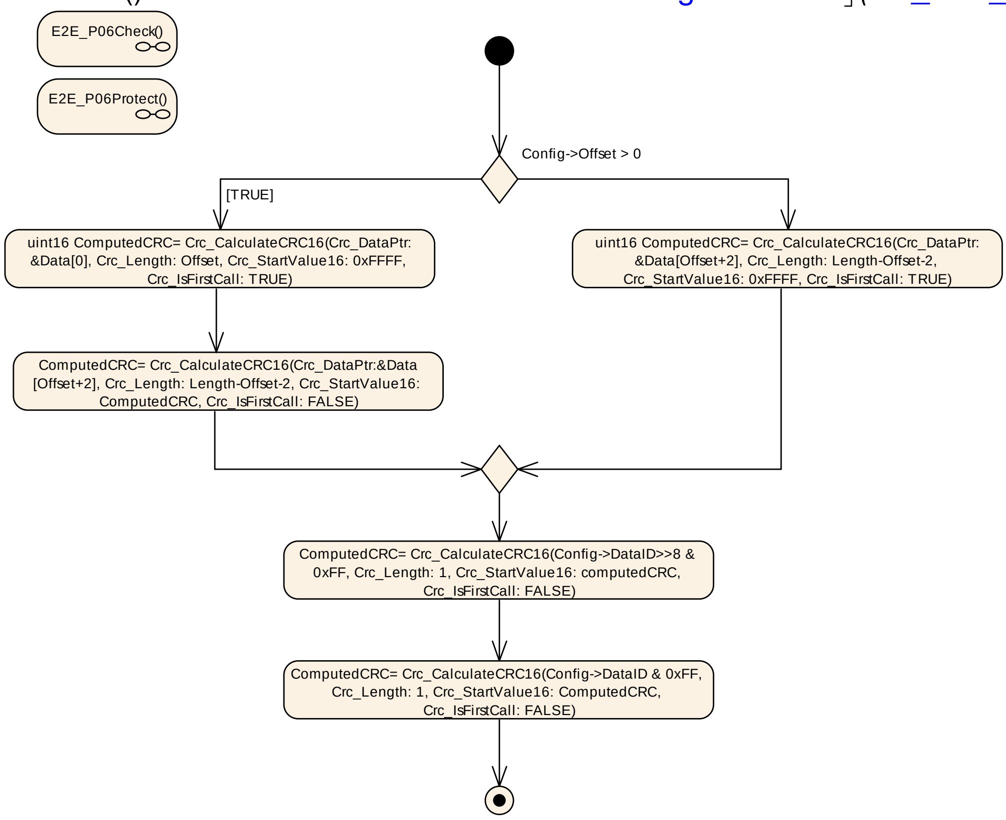

Figure 6.72: E2E Profile 6 Protect and Check step ”Compute CRC”

[PRS_E2E_00428] ⌈The step "Write CRC" in E2E_P06Protect() and E2E_P06Forward() shall behave as shown in Figure 6.73.⌋*(RS_E2E_08539)*

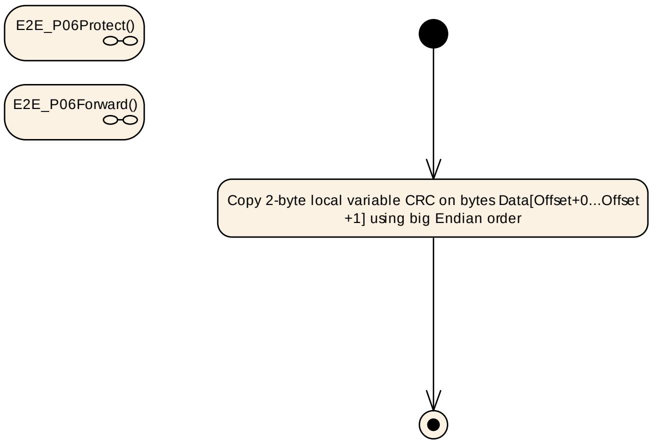

Figure 6.73: E2E Profile 6 Protect and Forward step "Write CRC"

[PRS_E2E_00429] ⌈The step "Increment Counter" in E2E_P06Protect() and E2E_P06Forward() shall behave as shown in Figure 6.74.⌋*(RS_E2E_08539)*

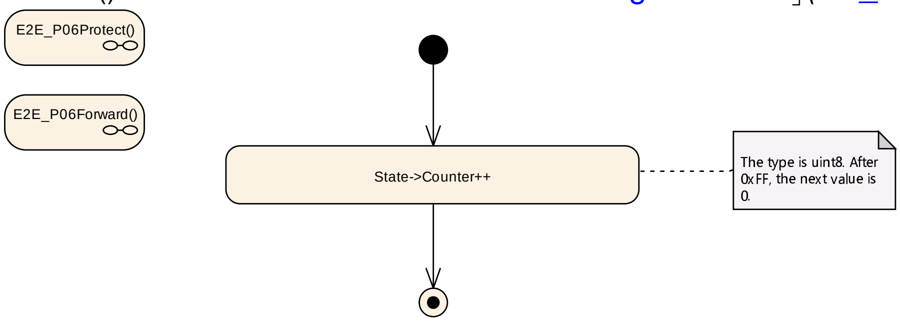

Figure 6.74: E2E Profile 6 Protect and Forward step "Increment Counter"

6.7.8 E2E_P06Forward

The E2E_P06Forward() function of E2E Profile 6 is called by a SW-C in order to protect its application data and forward an received E2E-Status for use cases like translation of signal based to service oriented communication. If the received E2E status equals E2E_P_OK the behavior of the function shall be the same like E2E_P06Protect(). The function E2E_P06Forward() performs the steps as specified by the following four diagrams in this section.

[PRS_E2E_00622] Draft ⌈The function E2E_P06Forward() shall have the overall behavior as shown in Figure 6.75.⌋*(RS_E2E_08639)*

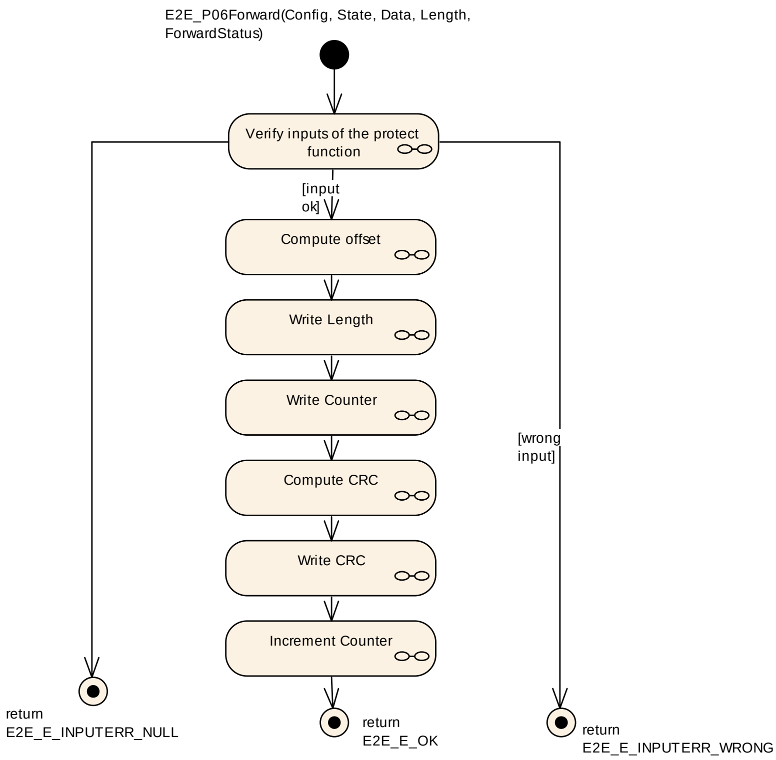

Figure 6.75: E2E Profile 6 Forward

Following steps are described in Section in Section 6.7.7

- ”Compute Offset” see [PRS_E2E_00470]
- ”Write Length” see [PRS_E2E_00425]
- ”Write CRC” see [PRS_E2E_00428]
- ”Increment Counter” see [PRS_E2E_00429]

[PRS_E2E_00623] Draft ⌈The step ”Verify inputs of the forward function” in E2E_P06Forward() shall behave as shown in Figure 6.76.⌋*(RS_E2E_08639)*

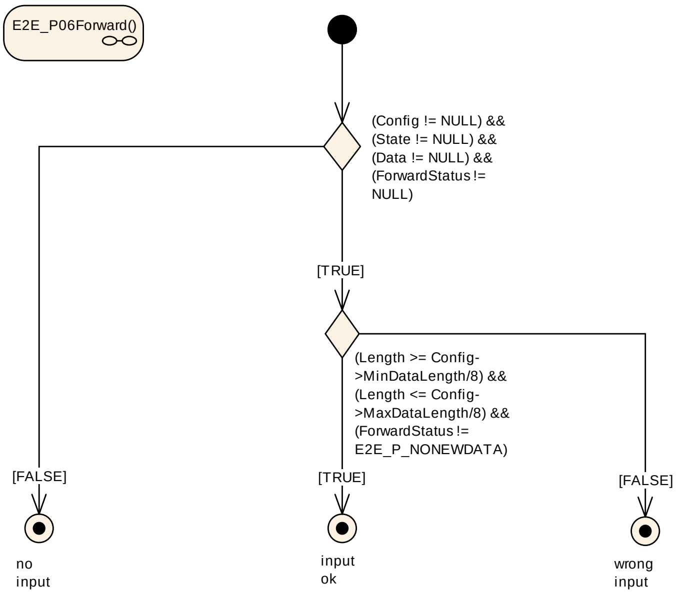

Figure 6.76: E2E Profile 6 Forward step "Verify inputs of the forward function"

[PRS_E2E_00624] Draft ⌈The step "Write Counter" in E2E_P06Forward() shall behave as shown in Figure 6.77.⌋ *(RS_E2E_08639)*

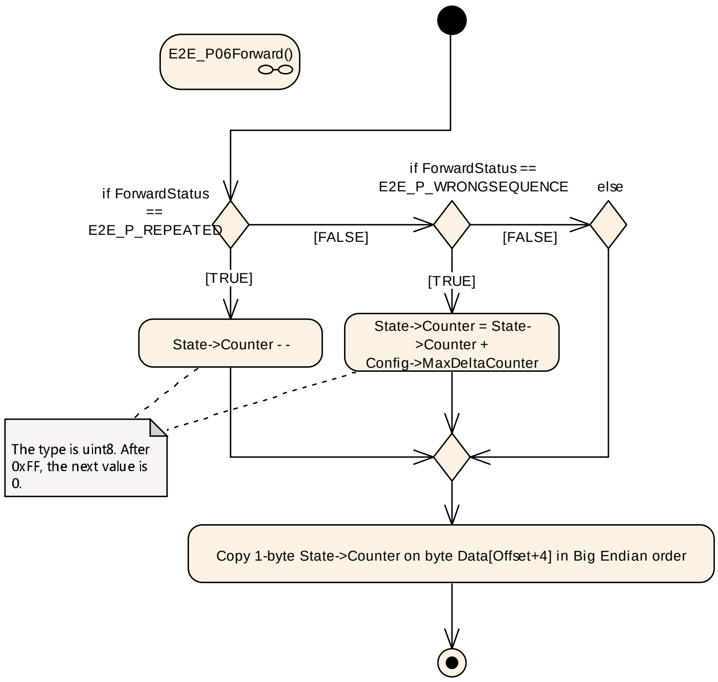

Figure 6.77: E2E Profile 6 Forward step "Write Counter"

[PRS_E2E_00625] **Draft** ⌈The step ”Compute CRC” in E2E_P06Forward() shall behave as shown in Figure 6.78.⌋*(RS_E2E_08639)*

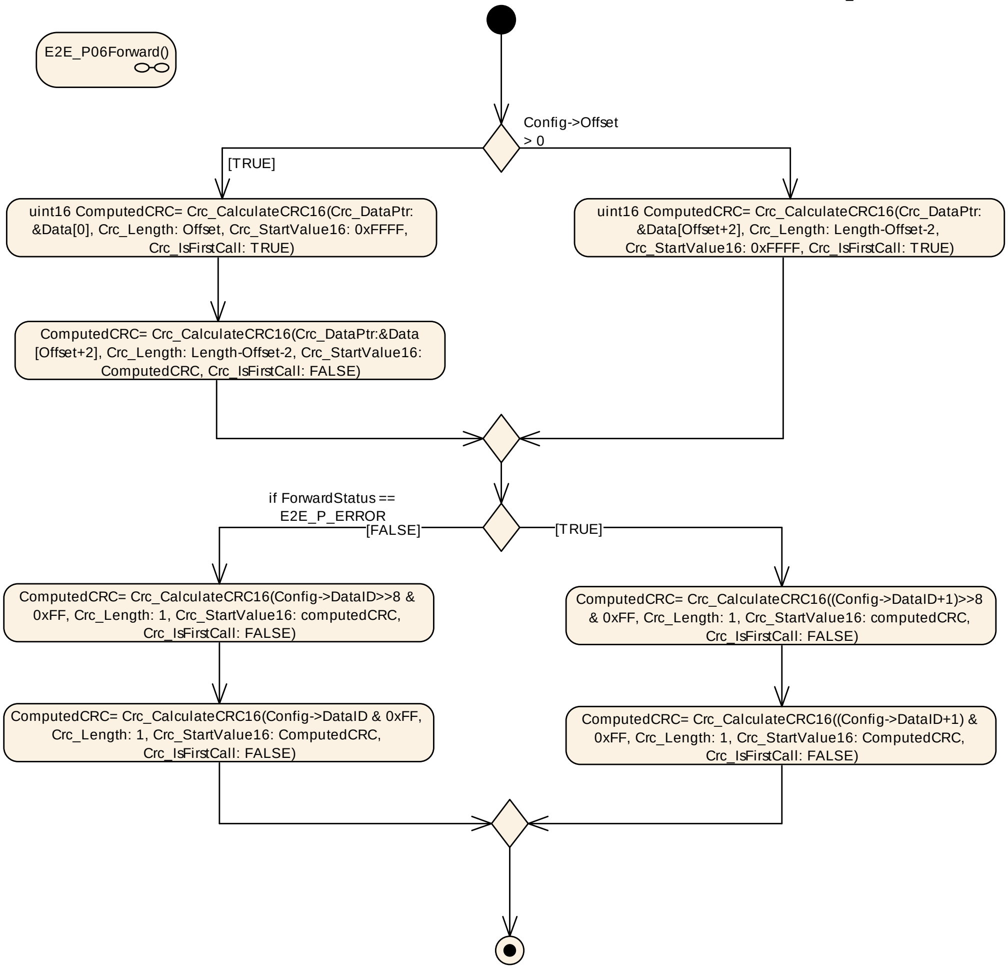

Figure 6.78: E2E Profile 6 Forward step ”Compute CRC”

6.7.9 E2E_P06Check

The function E2E_P06Check performs the actions as specified by the following seven diagrams in this section.

[PRS_E2E_00430] ⌈The function E2E_P06Check() shall have the overall behavior as shown in Figure 6.79.⌋*(RS_E2E_08539)*

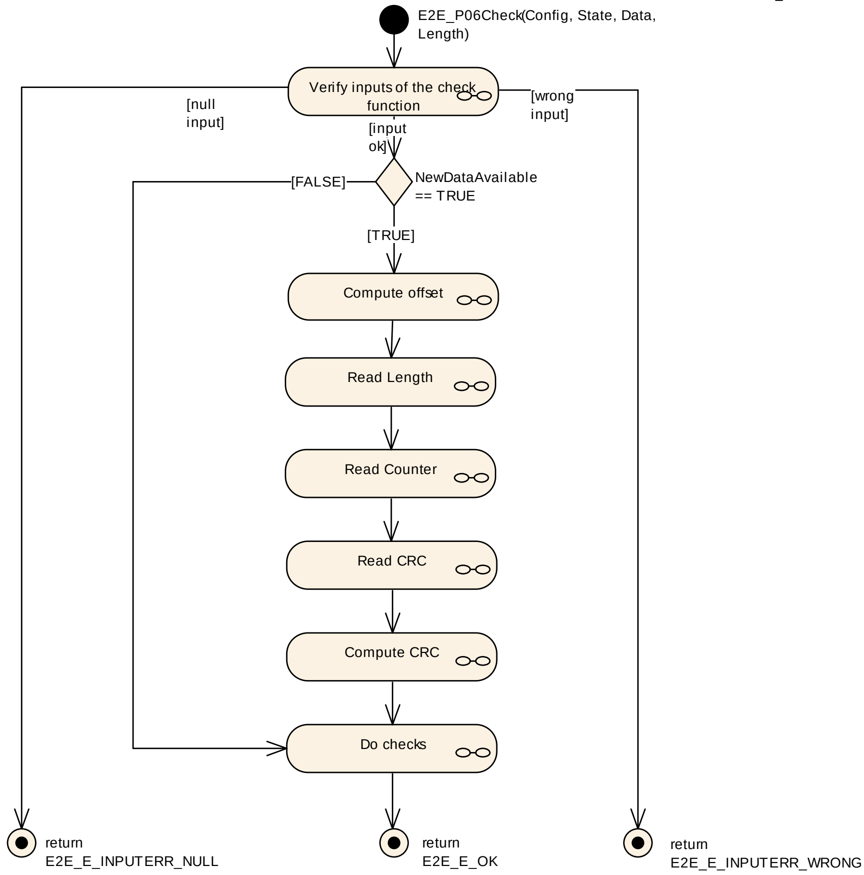

Figure 6.79: E2E Profile 6 Check

[PRS_E2E_00431] ⌈The step "Verify Inputs" in E2E_P06Check() shall behave as shown in Figure 6.80.⌋ *(RS_E2E_08539)*

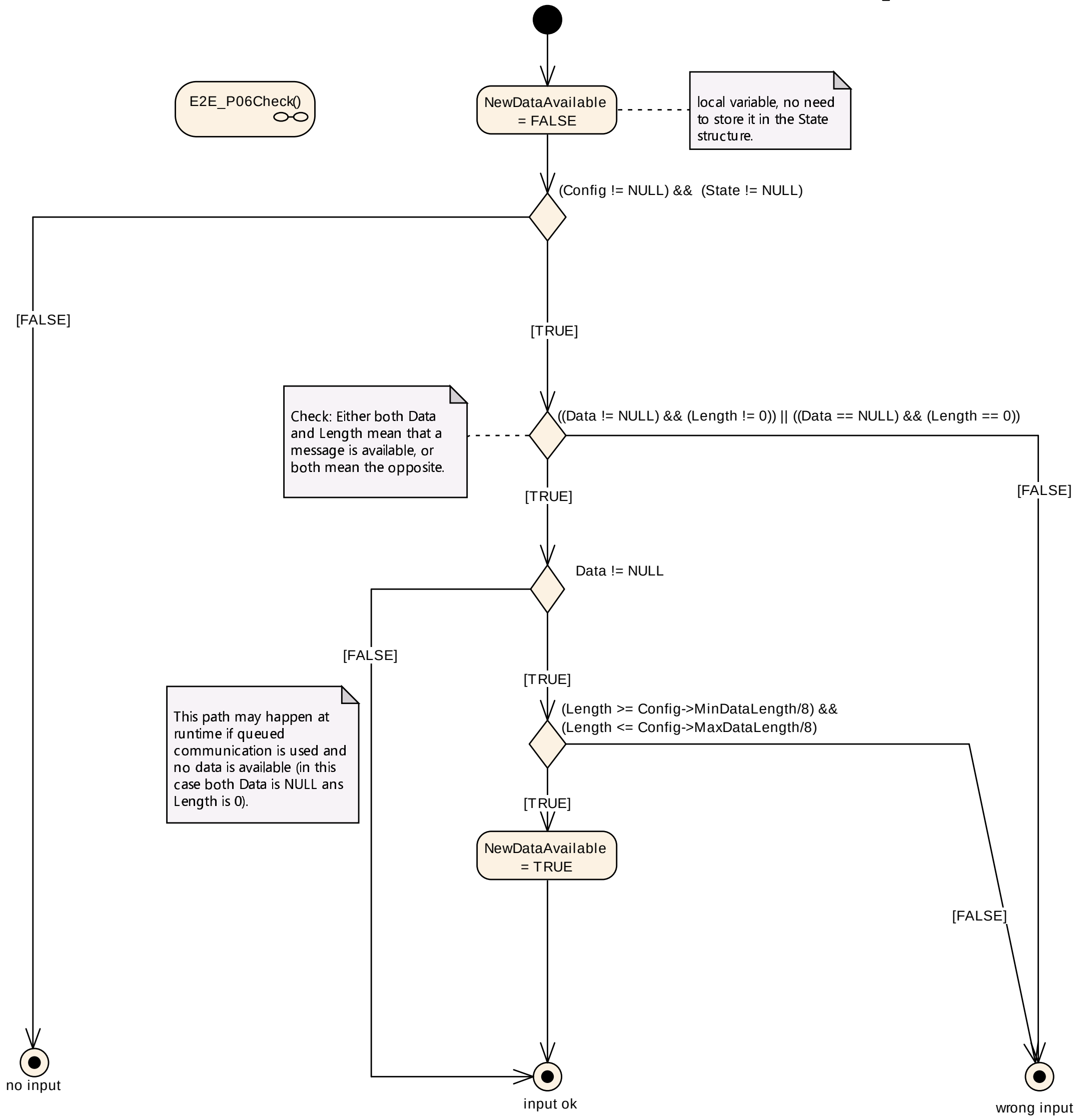

Figure 6.80: E2E Profile 6 Check step "Verify Inputs"

[PRS_E2E_00432] ⌈The step "Read Length" in E2E_P06Check() shall behave as shown in Figure 6.81.⌋ *(RS_E2E_08539)*

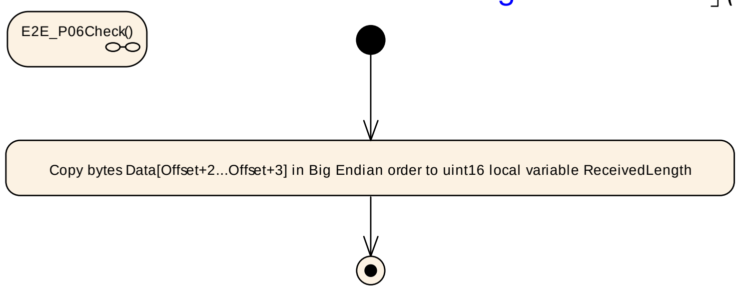

Figure 6.81: E2E Profile 6 Check step "Read Length"

[PRS_E2E_00433] ⌈The step ”Read Counter” in E2E_P06Check() shall behave as shown in Figure 6.82.⌋*(RS_E2E_08539)*

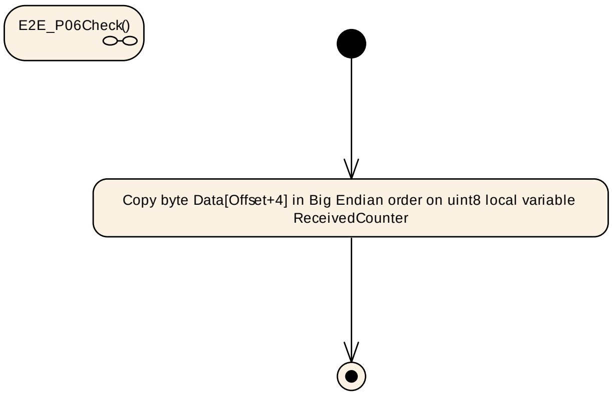

Figure 6.82: E2E Profile 6 Check step ”Read Counter”

[PRS_E2E_00434] ⌈The step ”Read CRC” in E2E_P06Check() shall behave as shown in Figure 6.83.⌋*(RS_E2E_08539)*

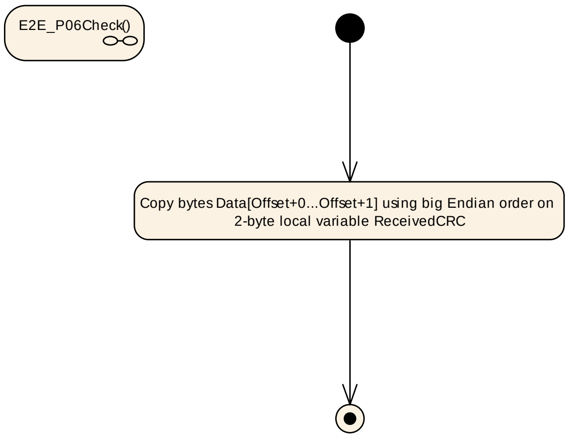

Figure 6.83: E2E Profile 6 Check step ”Read CRC”

[PRS_E2E_00436] ⌈The step "Do Checks" in E2E_P06Check() shall behave as shown in Figure 6.84.⌋*(RS_E2E_08539)*

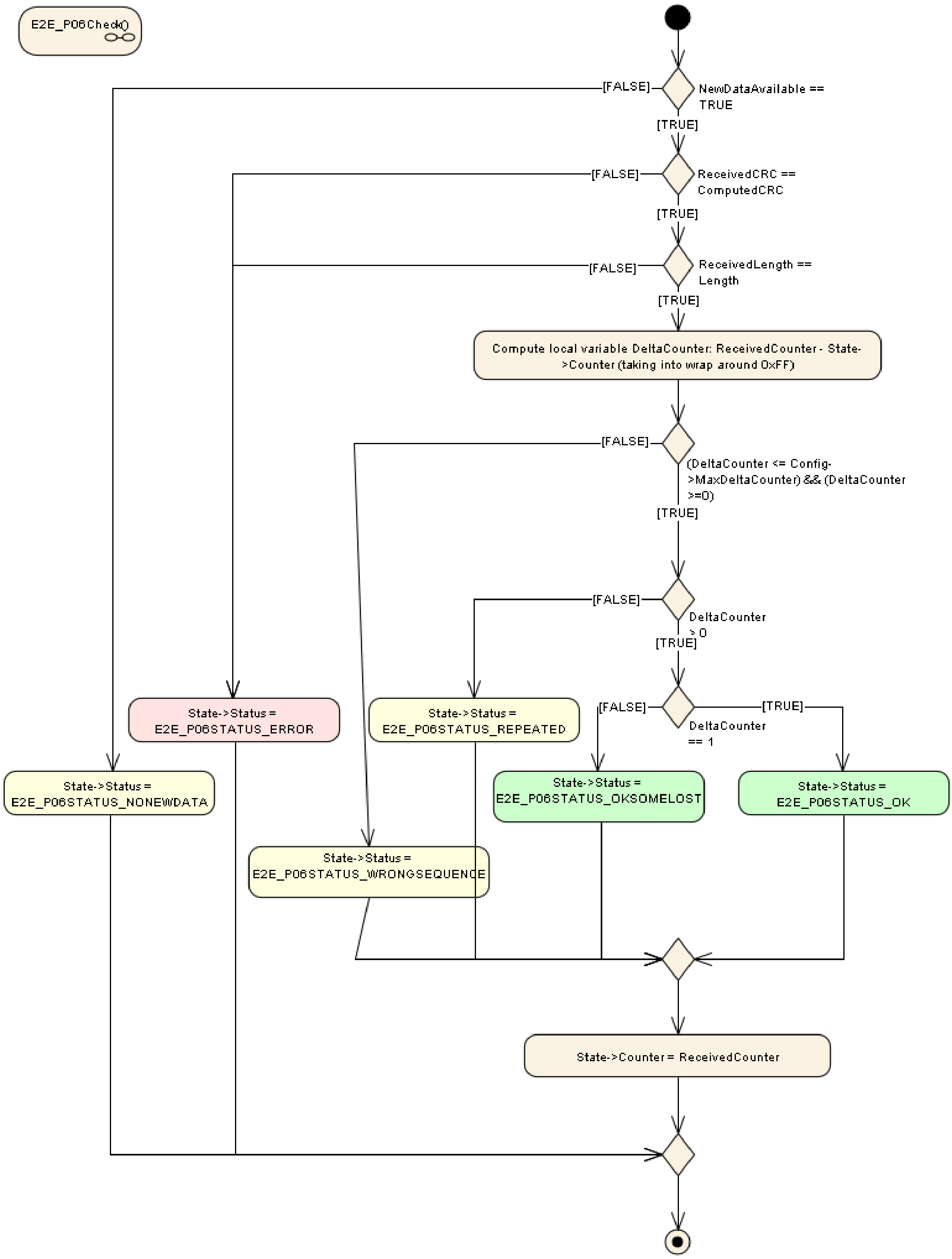

Figure 6.84: E2E Profile 6 Check step "Do Checks"

6.7.9.1 Profile 6 Check Status Enumeration

[PRS_E2E_00592] ⌈The step "Do Checks" in E2E_P06Check shall set State->Status to one of the following enumeration values (see Table 6.36).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_P06STATUS_OK	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented by 1).
E2E_P06STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed. This may be considered similar to E2E_P06STATUS_REPEATED.
E2E_P06STATUS_ERROR	Error	Error not related to counters occurred (e.g. wrong crc, wrong length, wrong options, wrong Data ID).
E2E_P06STATUS_REPEATED	Error	The checks of the Data in this cycle were successful, with the exception of the repetition.
E2E_P06STATUS_OKSOMELOST	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented within the allowed configured delta).
E2E_P06STATUS_WRONGSEQUENCE	Error	The checks of the Data in this cycle were successful, with the exception of counter jump, which changed more than the allowed delta

Table 6.36: E2E Profile 6 Check Status Enumeration

6.7.10 E2E Profile 6 Protocol Examples

The default configuration assumed for the following examples, if not otherwise stated to be different:

E2E_P06ConfigType field	Value
DataID	0x1234
Offset	0x0000
MinDataLength	40
MaxDataLength	32768
MaxDeltaCounter	1

Table 6.37: E2E Profile 6 protocol example configuration

E2E_P06ProtectStateType field	Value
Counter	0

Table 6.38: E2E Profile 6 example state initialization

Result data of E2E_P06Protect() with short data length (length 8 bytes, with 3 actual data bytes), offset = 0, counter = 0:

Byte	0	1	2	3	4	5	6	7
Data	0xb1	0x55	0x00	0x08	0x00	0x00	0x00	0x00
Field	*CRC*		*Length*		*Counter*	*Data*		

Table 6.39: E2E Profile 6 example short

Result data of E2E_P06Protect() with short data length (length 16 bytes, with 3 actual data bytes), offset = 64 (as with SOME/IP header use case), counter = 0:

Byte	0	1	2	3	4	5	6	7
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x4e	0xb7	0x00	0x10	0x00	0x00	0x00	0x00
Field	*CRC*		*Length*		*Counter*	*Data*		

Table 6.40: E2E Profile 6 example short with SOME/IP use case

6.8 Specification of E2E Profile 7

[PRS_E2E_00480] ⌈Profile 7 shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID (see Table 6.41).⌋*(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Control field	Description
Length	32 bits, to support dynamic-size data.
Counter	32 bits.
CRC	64 bits, polynomial in normal form 0x42F0E1EBA9EA3693, provided by CRC library. Note: This CRC polynomial is also known as “CRC-64 (ECMA)”.
Data ID	32 bits, unique system-wide.

Table 6.41: E2E Profile 7 mechanisms

The E2E mechanisms can detect the following faults or effects of faults:

Fault	Main safety mechanisms
Repetition of information	Counter
Loss of information	Counter
Delay of information	Counter
Insertion of information	Data ID, CRC
Masquerading	Data ID, CRC
Incorrect addressing	Data ID
Incorrect sequence of information	Counter
Corruption of information	CRC
Asymmetric information sent from a sender to multiple receivers	CRC (to detect corruption at any of receivers)
Information from a sender received by only a subset of the receivers	Counter (loss on specific receivers)
Blocking access to a communication channel	Counter (loss or timeout)

Table 6.42: Detectable communication faults using Profile 7

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[2].

6.8.1 Data Layout

6.8.1.1 User data layout

In the E2E Profile 7, the user data layout (of the data to be protected) is not constrained by E2E Profile 7 - there is only a requirement that the length of data to be protected is multiple of 1 byte.

6.8.1.2 Header layout

The header of the E2E Profile 7 has one fixed layout, as follows:

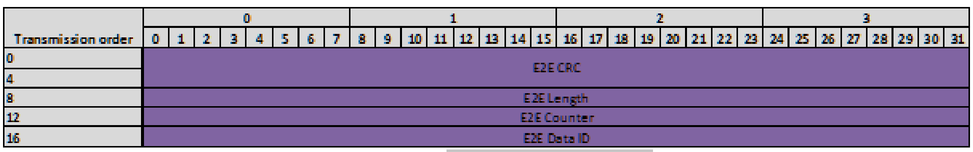

Figure 6.85: Profile 7 Header

The bit numbering shown above represents the order in which bits are transmitted. The E2E header fields (e.g. E2E Counter) are encoded as:

1. Big Endian (most significant byte fist) - imposed by profile

2. LSB Fist (least significant bit within byte first) - imposed by TCPIP bus

For example, the 32 bits of the E2E counter are transmitted in the following order (higher number meaning higher significance): 24 25 26 27 28 29 30 31 16 17 18 19 20 21 22 23 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7.

The header can be placed at a specific location in the protected data, by configuring the offset of the entire E2E header.

6.8.2 Counter

In E2E Profile 7, the counter is initialized, incremented, reset and checked by E2E profile. The counter is not manipulated or used by the caller of the E2E Supervision.

[PRS_E2E_00481] ⌈In E2E Profile 7, on the sender side, for the first transmission request of a data element the counter shall be initialized with 0 and shall be incremented by 1 for every subsequent send request. When the counter reaches the maximum value (0xFF'FF'FF'FF), then it shall restart with 0 for the next send request.⌋*(RS_E2E_08539)*

Note that the counter value 0xFF'FF'FF'FF is not reseved as a special invalid value, but it is used as a normal counter value.

In E2E Profile 7, on the receiver side, by evaluating the counter of received data against the counter of previously received data, the following is detected:

1. Repetition:

   a. no new data has arrived since last invocation of E2E Supervision check function, b. the data is repeated

2. OK: a. counter is incremented by one (i.e. no data lost), b. counter is incremented more than by one, but still within allowed limits (i.e. some data lost),

3. Wrong sequence: a. counter is incremented more than allowed (i.e. too many data lost).

Case 1 corresponds to the failed alive counter check, and case 3 correspond to failed sequence counter check.

The above requirements are specified in more details by the UML diagrams in the following document sections.

6.8.3 Data ID

The unique Data IDs are to verify the identity of each transmitted safety-related data element.

[PRS_E2E_00482] ⌈In the E2E Profile 7, the Data ID shall be explicitly transmitted, i.e. it shall be the part of the transmitted E2E header⌋*(RS_E2E_08539)*

There are currently no limitations on the values of Data ID - any values within the addres space of 32 bits are allowed.

[PRS_E2E_00483] ⌈In the E2E profile 7, the Data IDs shall be globally unique within the network of communicating system (made of several ECUs each sending different data).⌋*(RS_E2E_08539)*

In case of usage of E2E Supervision for protecting data elements (i.e invocation from RTE), due to multiplicity of communication (1:1 or 1:N), a consumer of a data element expects only a specific data element, which is checked by E2E Supervision using Data ID.

In case of usage of E2E Supervision for protecting messages (i.e. invocation from COM), the receiver COM expects at a reception only a specific message, which is checked by E2E Supervision using Data ID.

6.8.4 Length

The Length field is introduced to support variable-size length - the Data [] array storing the serialized data can potentially have a different length in each cycle. The Length includes user data + E2E Header (CRC + Counter + Length + DataID).

6.8.5 CRC

E2E Profile 7 uses a 64-bit CRC, to ensure a high detection rate and high Hamming Distance.

[PRS_E2E_00484] ⌈E2E Profile 7 shall use the Crc_CalculateCRC64 4 () function of the SWS CRC Supervision for calculating the CRC.⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_00485] ⌈In E2E Profile 7, the CRC shall be calculated over the entire E2E header (excluding the CRC bytes) and over the user data.⌋*(RS_E2E_08536)*

6.8.6 Timeout detection

The previously mentioned mechanisms (CRC, Counter, Data ID, Length) enable to check the validity of received data element, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for Data Elements or respectively messages, but instead if the receiver reads the currently available data (i.e. checks if new data is available). Then, by means of the counter, the receiver can detect loss of communication and timeouts.

6.8.7 E2E Profile 7 variants

[PRS_E2E_00586] ⌈The E2E Profile variant 7A is defined as follows:

1. The CRC is the 64th bit in the signal group
2. The max Delta Counter is 1

⌋*(RS_E2E_08528)*

[PRS_E2E_00587] ⌈The E2E Profile variant 7B is defined as follows:

1. The CRC is the 64th bit in the signal group
2. The max Delta Counter is 2

⌋*(RS_E2E_08528)*

6.8.8 E2E_P07Protect

The function E2E_P07Protect() performs the steps as specified by the following eight diagrams in this section.

[PRS_E2E_00486] ⌈The function E2E_P07Protect() shall have the overall behavior as shown in Figure 6.86.⌋*(RS_E2E_08539)*

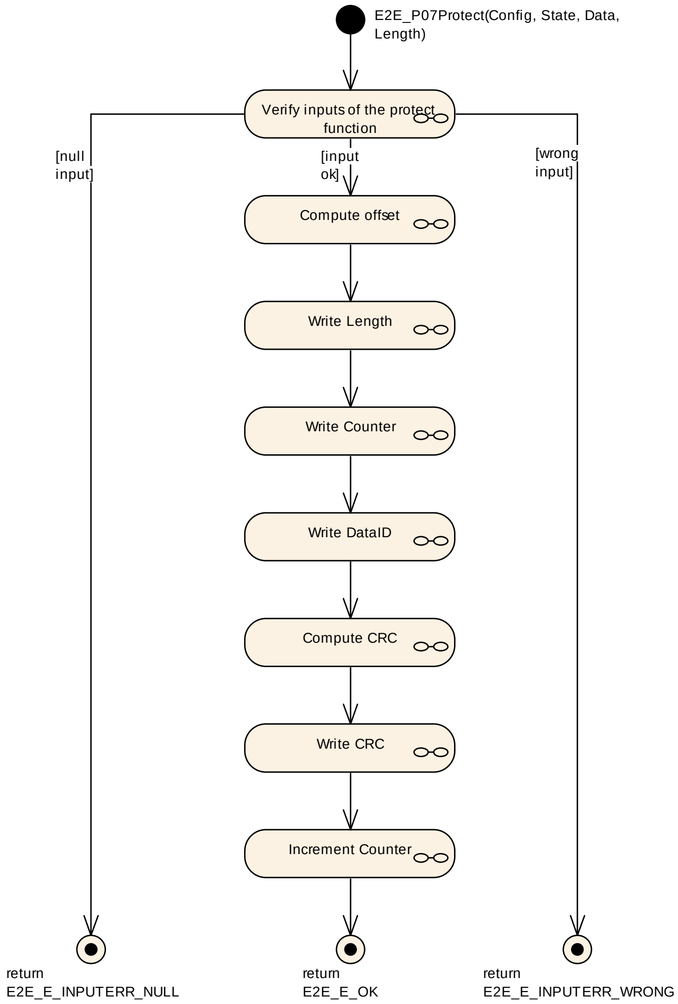

Figure 6.86: E2E Profile 7 Protect

[PRS_E2E_00487] ⌈The step "Verify inputs of the protect function" in E2E_P07Protect() shall behave as shown in Figure 6.87.⌋*(RS_E2E_08539)*

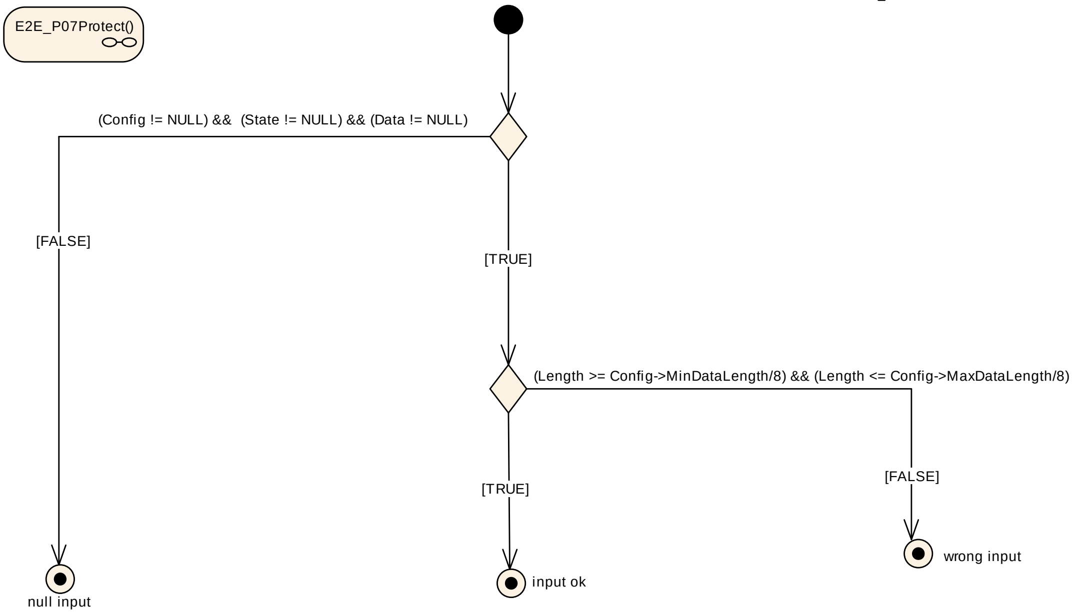

Figure 6.87: E2E Profile 7 Protect step "Verify inputs of the protect function"

[PRS_E2E_00488] ⌈The step "Compute offset" in E2E_P07Protect(), E2E_P07Forward() and E2E_P07Check() shall behave as shown in Figure 6.88.⌋*()*

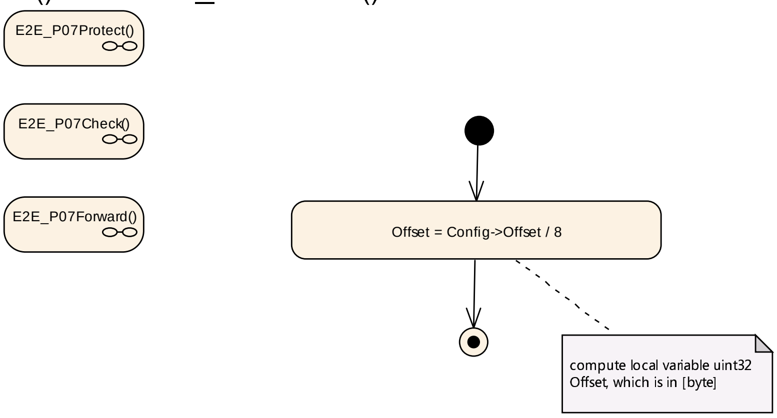

Figure 6.88: E2E Profile 7 Protect and Forward step "Compute offset"

[PRS_E2E_00489] ⌈The step "Write Length" in E2E_P07Protect() and E2E_P07Forward() shall behave as shown in Figure 6.89.⌋*(RS_E2E_08539)*

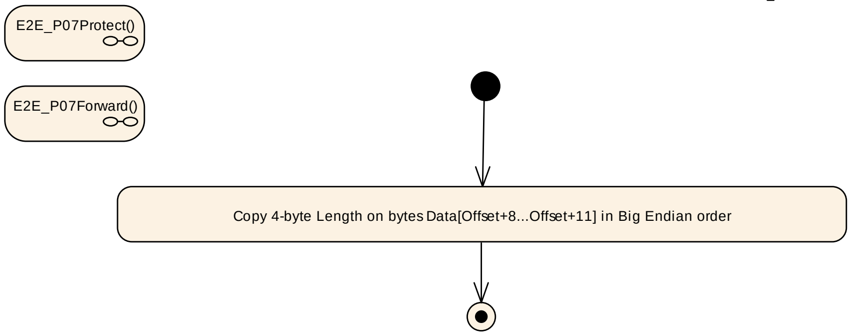

Figure 6.89: E2E Profile 7 Protect and Forward step "Write Length"

[PRS_E2E_00490] ⌈The step "Write Counter" in E2E_P07Protect() shall behave as shown in Figure 6.90.⌋*(RS_E2E_08539)*

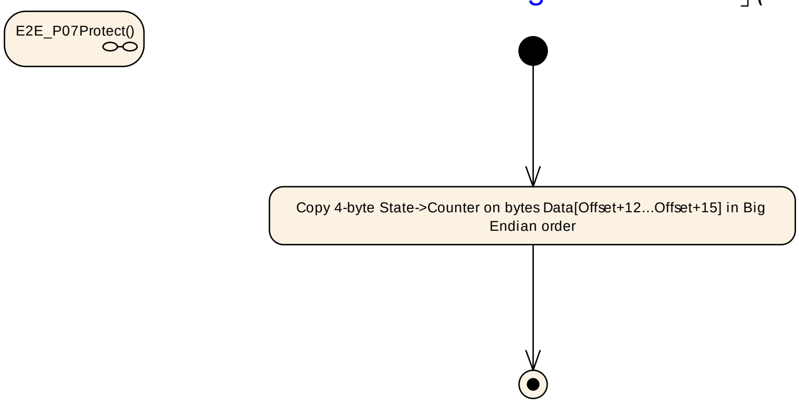

Figure 6.90: E2E Profile 7 Protect step "Write Counter"

[PRS_E2E_00491] ⌈The step "Write DataID"in E2E_P07Protect() shall behave as shown in Figure 6.91.⌋*(RS_E2E_08539)*

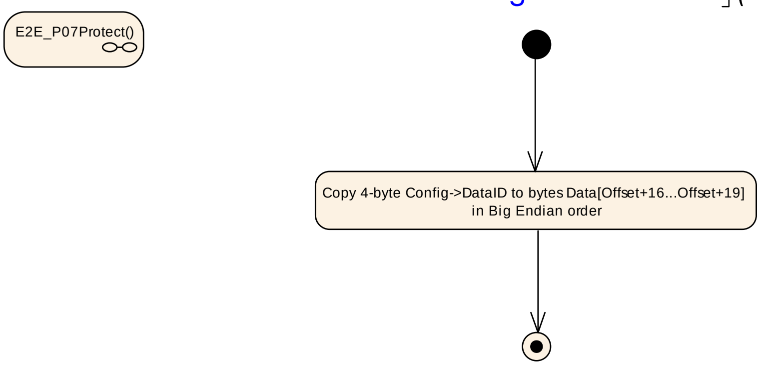

Figure 6.91: E2E Profile 7 Protect step "Write Data ID"

[PRS_E2E_00492] ⌈The step "Compute CRC" in E2E_P07Protect(), E2E_P07Forward() and in E2E_P07Check()

shall behave as shown in Figure 6.92.⌋(RS_E2E_08539)

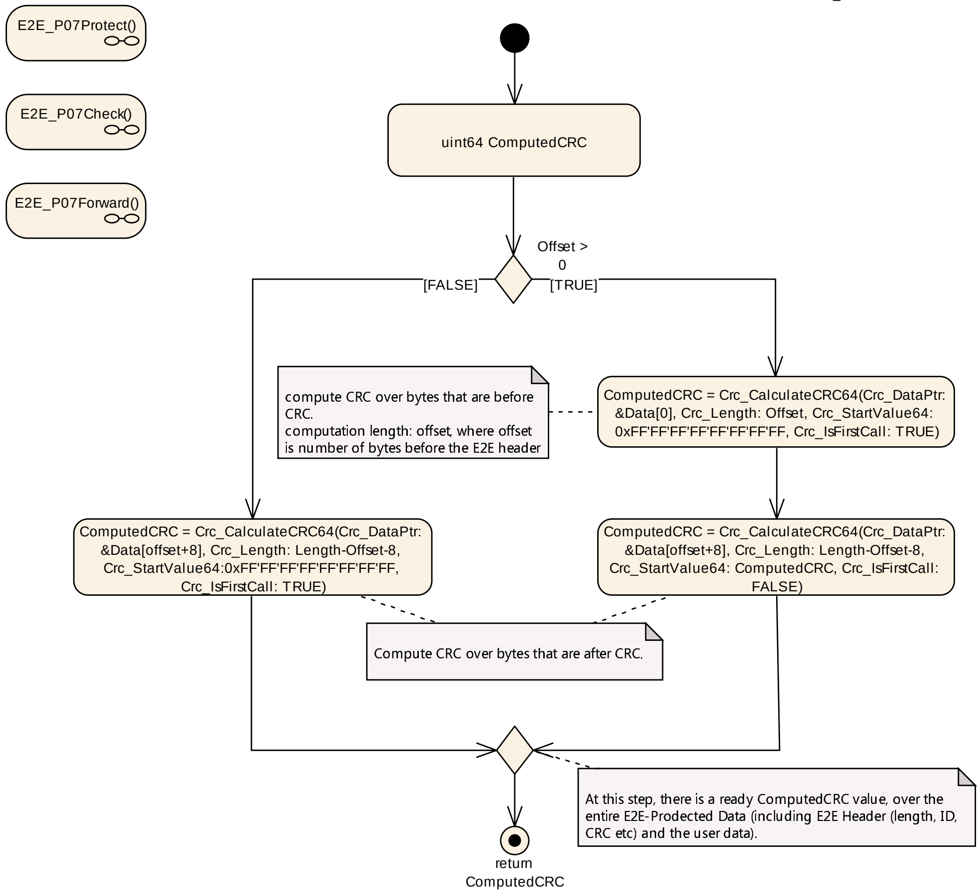

Figure 6.92: E2E Profile 7 Protect, Forward and Check step "ComputeCRC"

[PRS_E2E_00493] ⌈The step "Write CRC"in E2E_P07Protect() and E2E_P07Forward() shall behave as shown in Figure 6.93.⌋(RS_E2E_08539)

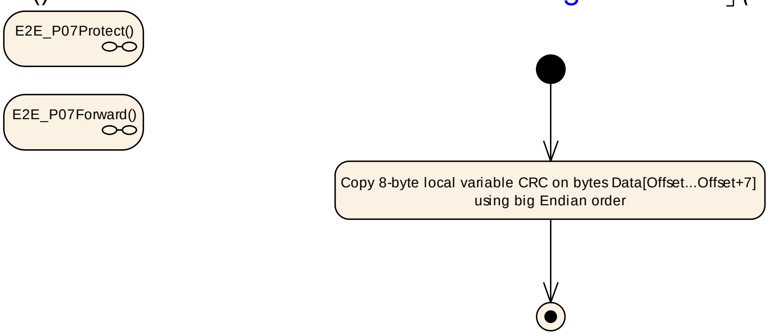

Figure 6.93: E2E Profile 7 Protect and Forward step "Write CRC"

[PRS_E2E_00494] ⌈The step ”Increment Counter” in E2E_P07Protect() and E2E_P07Forward() shall behave as shown in Figure 6.94.⌋*(RS_E2E_08539)*

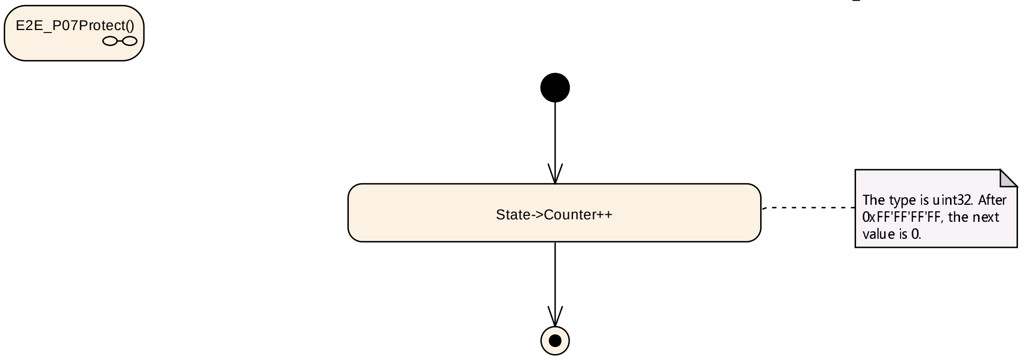

Figure 6.94: E2E Profile 7 Protect and Forward step ”Increment Counter”

6.8.9 E2E_P07Forward

The E2E_P07Forward() function of E2E Profile 7 is called by a SW-C in order to protect its application data and forward an received E2E-Status for use cases like translation of signal based to service oriented communication. If the received E2E status equals E2E_P_OK the behavior of the function shall be the same like E2E_P07Protect(). The function E2E_P07Forward() performs the steps as specified by the following four diagrams in this section.

[PRS_E2E_00626] Draft ⌈The function E2E_P07Forward() shall have the overall behavior as shown in Figure 6.95.⌋*(RS_E2E_08739)*

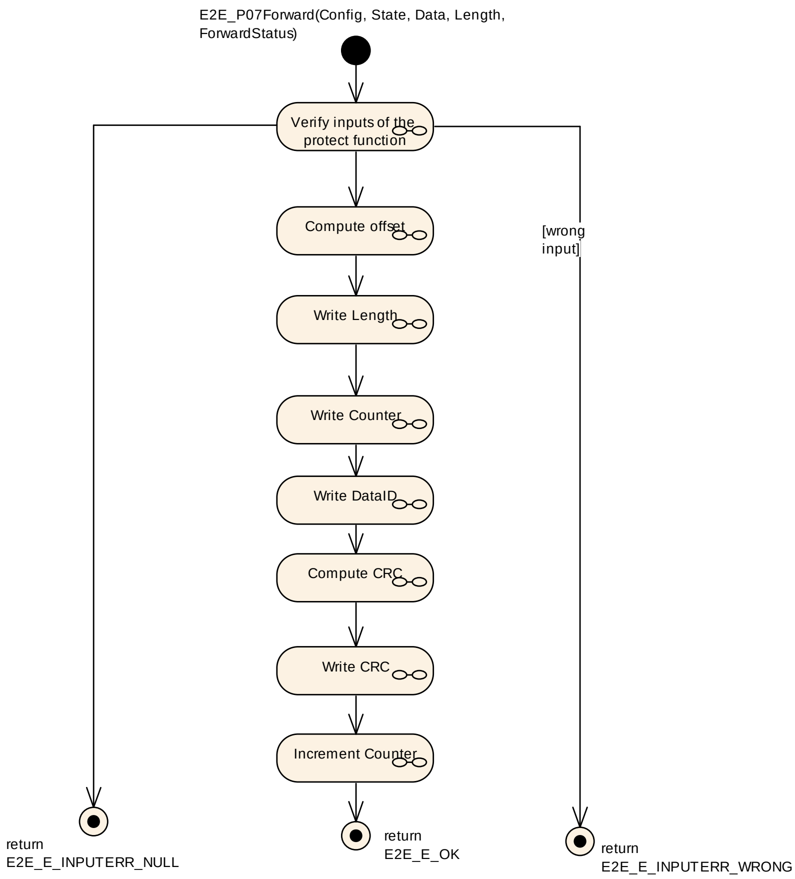

Figure 6.95: E2E Profile 7 Forward

Following steps are described in Section in Section 6.8.8

- ”Compute Offset” see [PRS_E2E_00488]
- ”Write Length” see [PRS_E2E_00489]
- ”Compute CRC” see [PRS_E2E_00492]
- ”Write CRC” see [PRS_E2E_00493]
- ”Increment Counter” see [PRS_E2E_00494]

[PRS_E2E_00627] Draft ⌈The step ”Verify inputs of the forward function” in E2E_P07Forward() shall behave as shown in Figure 6.96.⌋*(RS_E2E_08739)*

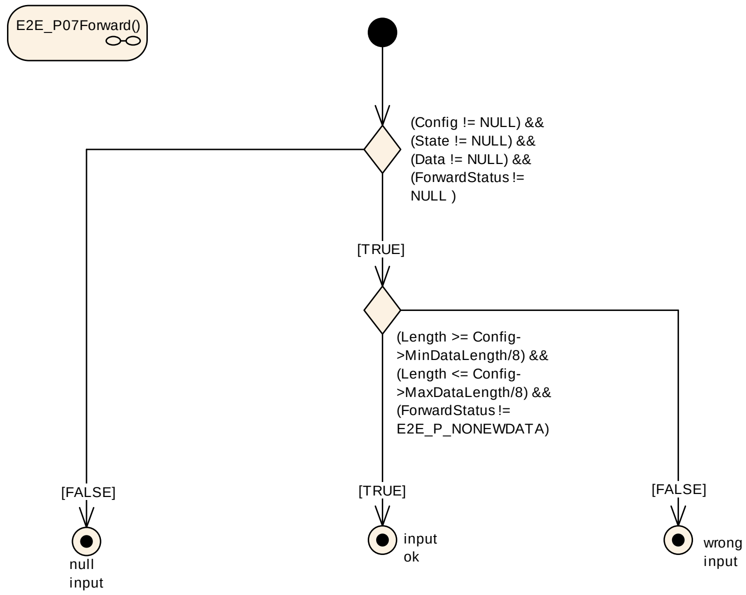

Figure 6.96: E2E Profile 7 Forward step "Verify inputs of the forward function"

[PRS_E2E_00628] Draft ⌈The step "Write Counter" in E2E_P07Forward() shall behave as shown in Figure 6.97.⌋*(RS_E2E_08739)*

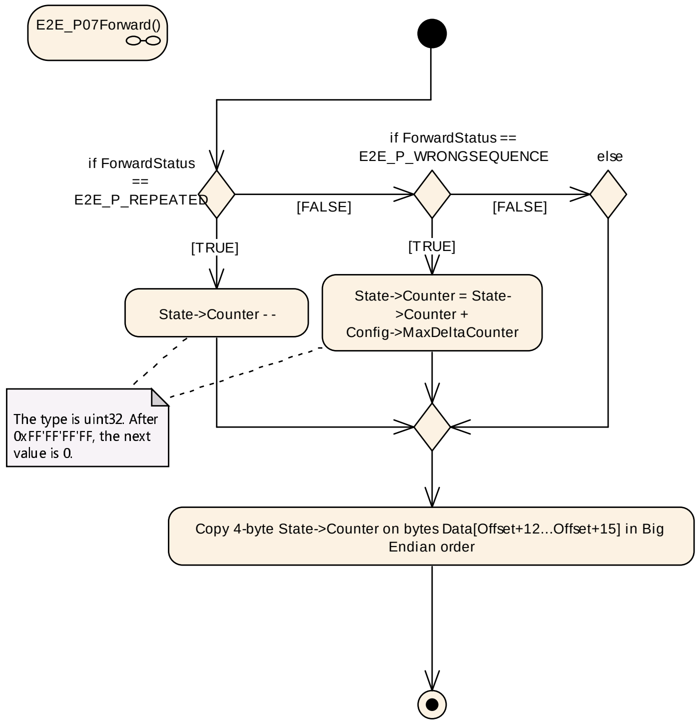

Figure 6.97: E2E Profile 7 Forward step "Write Counter"

[PRS_E2E_00629] **Draft** ⌈The step ”Write DataID” in E2E_P07Forward() shall behave as shown in Figure 6.98.⌋*(RS_E2E_08739)*

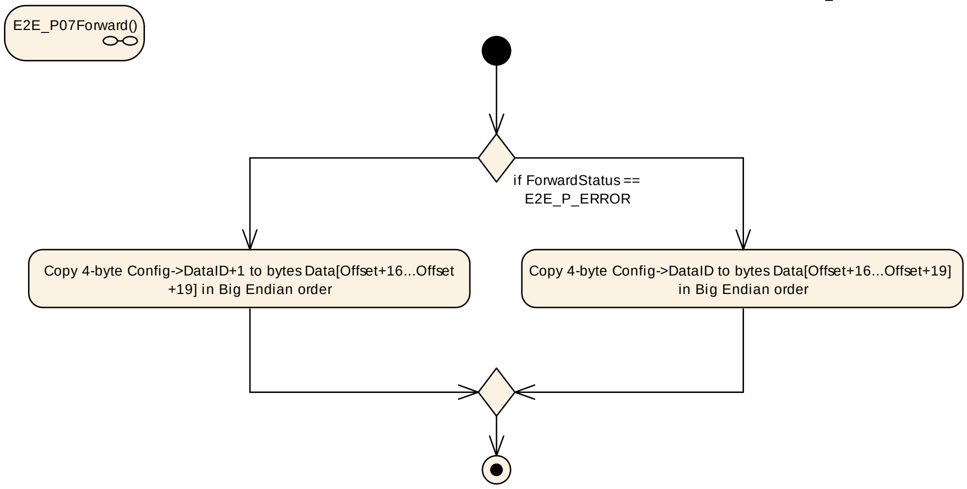

Figure 6.98: E2E Profile 7 Forward step ”Write DataID”

6.8.10 E2E_P07Check

The function E2E_P07Check performs the actions as as specified by the following seven diagrams in this section and according to diagram PRS_E2EProtocol_00492.

[PRS_E2E_00495] ⌈The function E2E_P07Check() shall have the overall behavior as shown in Figure 6.99.⌋*(RS_E2E_08539)*

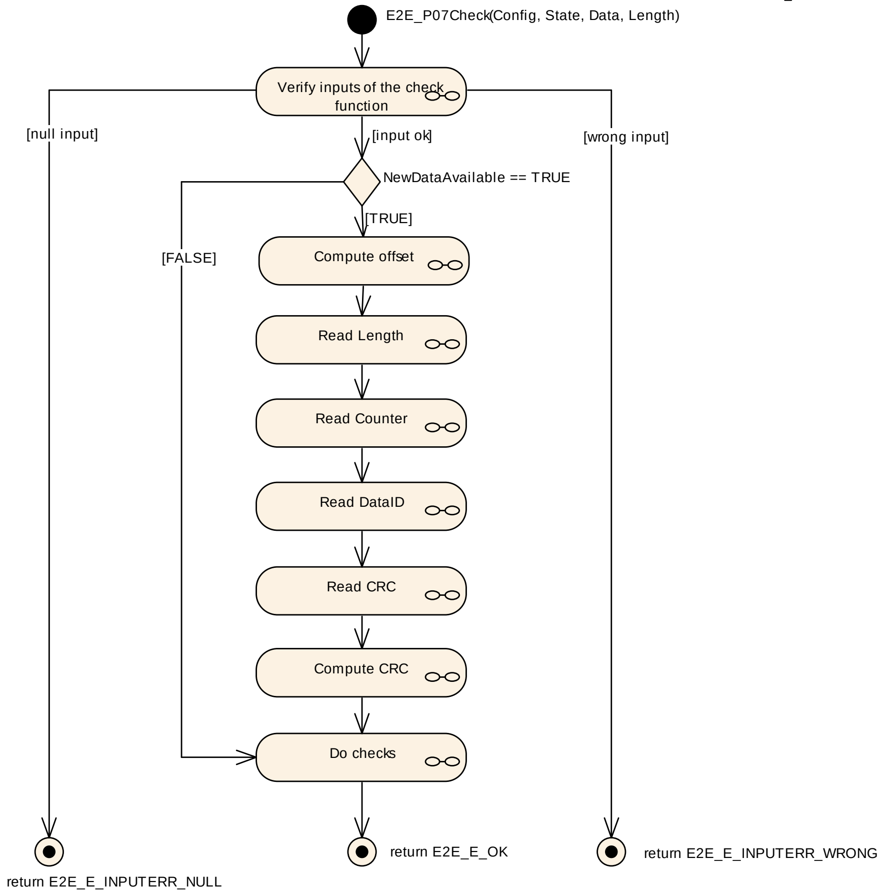

Figure 6.99: E2E Profile 7 Check

[PRS_E2E_00496] ⌈The step "Verify inputs of the check function" in E2E_P07Check() shall behave as shown in Figure 6.100.⌋*(RS_E2E_08539)*

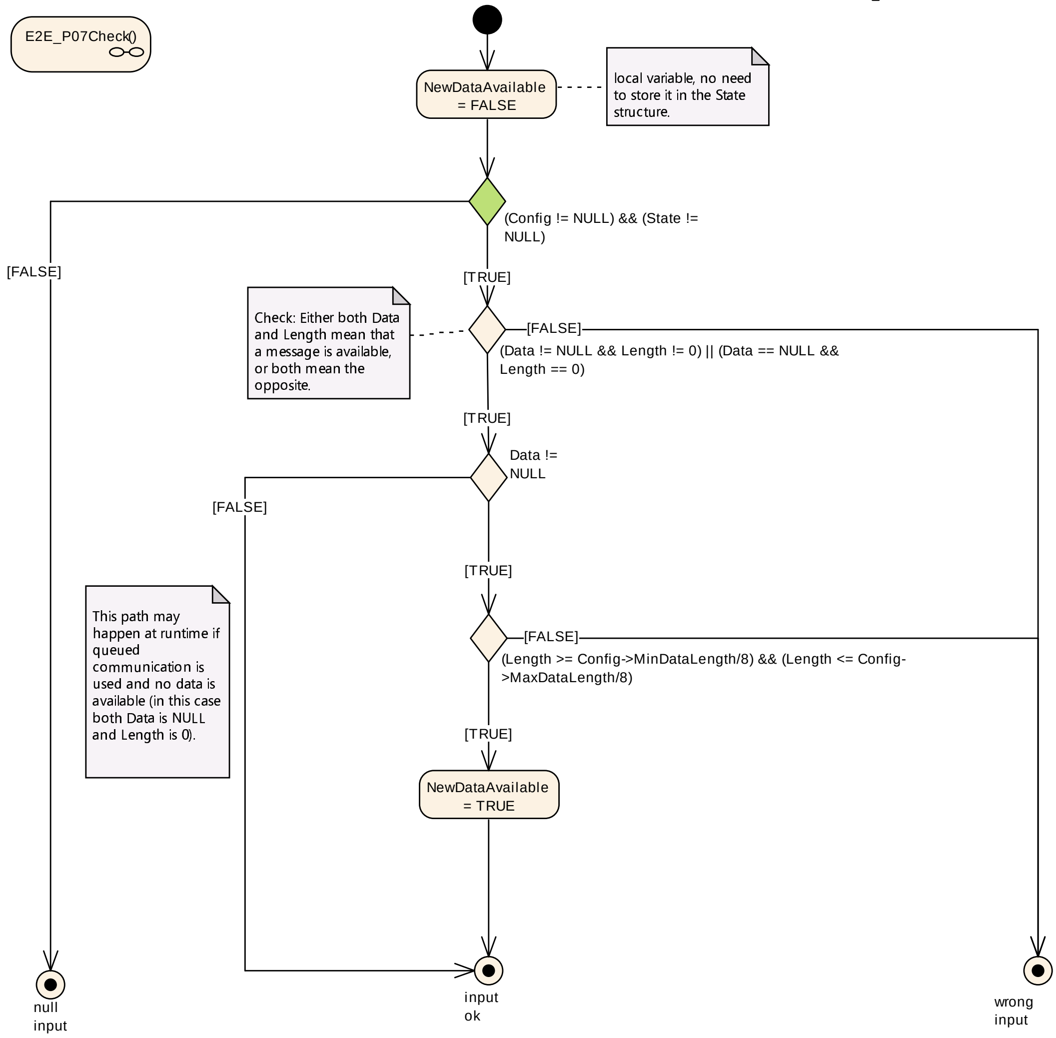

Figure 6.100: E2E Profile 7 Check step "Verify inputs of the check function"

[PRS_E2E_00497] ⌈The step "Read Length" in E2E_P07Check() shall behave as shown in Figure 6.101.⌋*(RS_E2E_08539)*

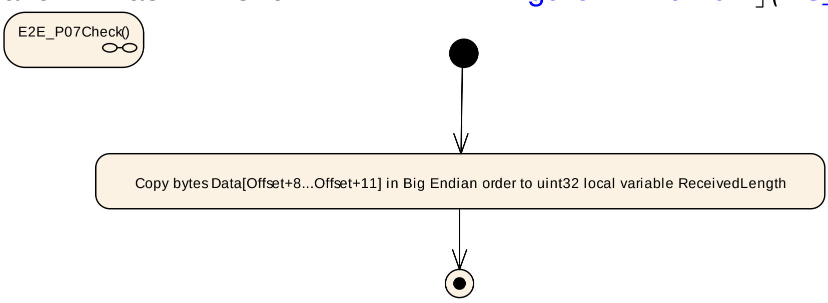

Figure 6.101: E2E Profile 7 Check step "Read Length"

[PRS_E2E_00498] ⌈The step "Read Counter" in E2E_P07Check() shall behave as shown in Figure 6.102.⌋ *(RS_E2E_08539)*

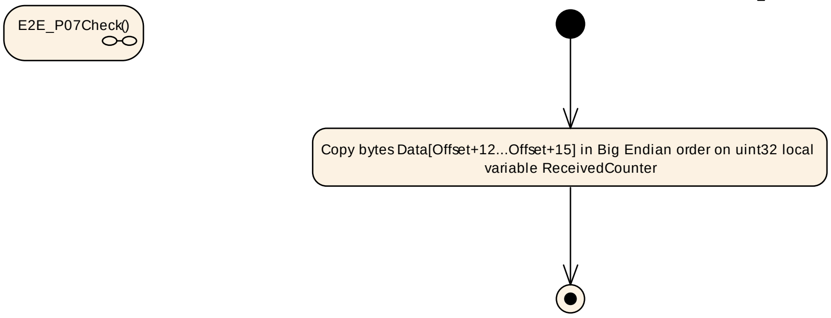

Figure 6.102: E2E Profile 7 Check step "Read Counter"

[PRS_E2E_00499] ⌈The step "Read DataID" in E2E_P07Check() shall behave as shown in Figure 6.103.⌋ *(RS_E2E_08539)*

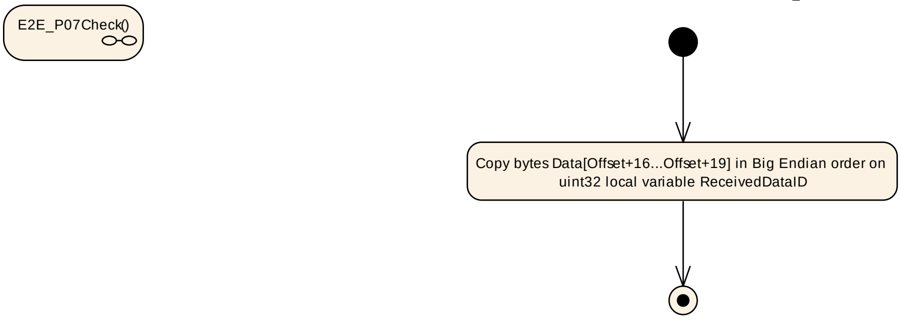

Figure 6.103: E2E Profile 7 Check step "Read DataID"

[PRS_E2E_00500] ⌈The step "Read CRC" in E2E_P07Check() shall behave as shown in Figure 6.104.⌋ *(RS_E2E_08539)*

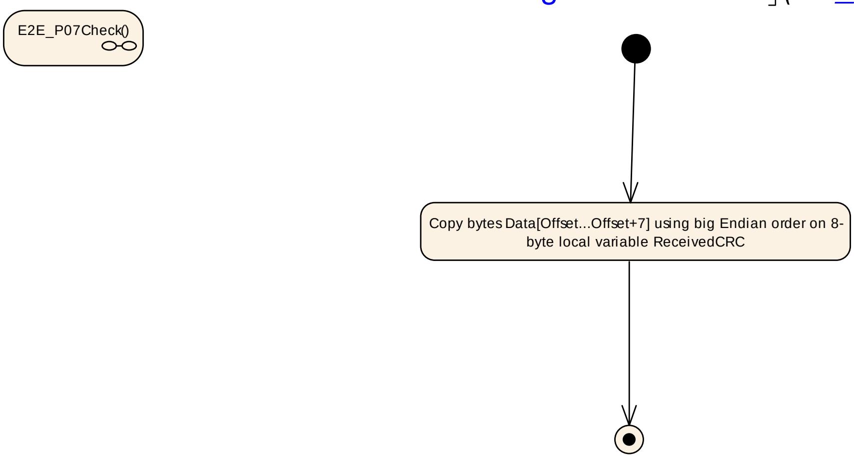

Figure 6.104: E2E Profile 7 Check step "Read CRC"

[PRS_E2E_00501] ⌈The step ”Do Checks” in E2E_P07Check() shall behave as shown in Figure 6.105.⌋*(RS_E2E_08539)*

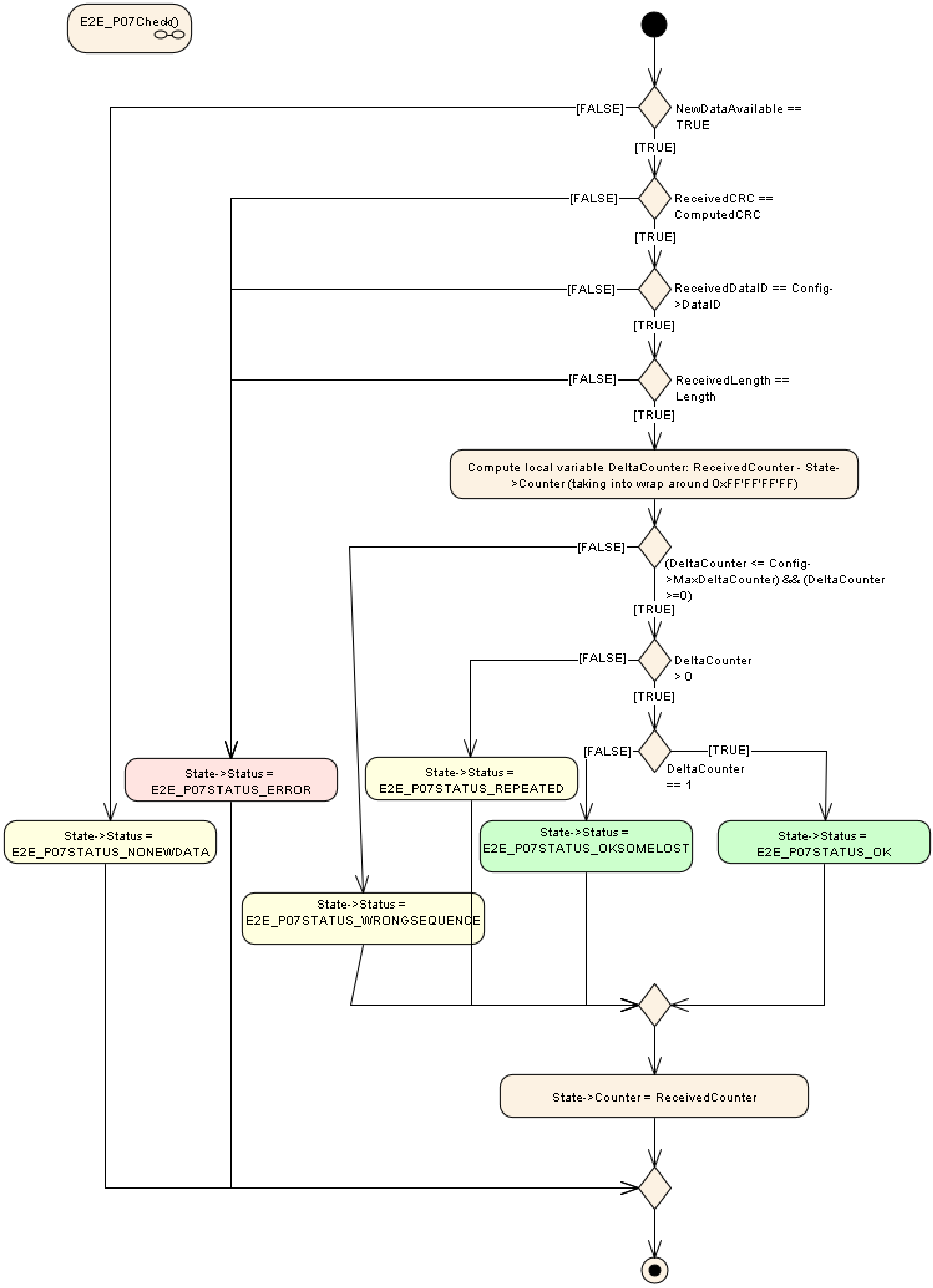

Figure 6.105: E2E Profile 7 Check step ”Do Checks”

6.8.10.1 Profile 7 Check Status Enumeration

[PRS_E2E_00593] ⌈The step "Do Checks" in E2E_P07Check shall set State->Status to one of the following enumeration values (see Table 6.43).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_P07STATUS_OK	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented by 1).
E2E_P07STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed. This may be considered similar to E2E_P07STATUS_REPEATED.
E2E_P07STATUS_ERROR	Error	Error not related to counters occurred (e.g. wrong crc, wrong length, wrong options, wrong Data ID).
E2E_P07STATUS_REPEATED	Error	The checks of the Data in this cycle were successful, with the exception of the repetition.
E2E_P07STATUS_OKSOMELOST	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented within the allowed configured delta).
E2E_P07STATUS_WRONGSEQUENCE	Error	The checks of the Data in this cycle were successful, with the exception of counter jump, which changed more than the allowed delta

Table 6.43: E2E Profile 7 Check Status Enumeration

6.8.11 E2E Profile 7 Protocol Examples

The default configuration assumed for the following examples, if not otherwise stated to be different:

E2E_P07ConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000
MinDataLength	160
MaxDataLength	32768
MaxDeltaCounter	1

Table 6.44: E2E Profile 7 protocol example configuration

E2E_P07ProtectStateType field	Value
Counter	0

Table 6.45: E2E Profile 7 example state initialization

Result data of E2E_P07Protect() with short data length (length 24 bytes, means 4 actual data bytes), offset = 0, counter = 0:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x1f	0xb2	0xe7	0x37	0xfc	0xed	0xbc	0xd9
Field	*CRC*							
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x00	0x00	0x00	0x18	0x00	0x00	0x00	0x00
Field	*Length*				*Counter*			
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0x0a	0x0b	0x0c	0x0d	0x00	0x00	0x00	0x00
Field	*DataID*				*Data*			

Table 6.46: E2E Profile 7 example short

Result data of E2E_P07Protect() with short data length (length 32, means 4 actual data bytes), offset = 64 (as with SOME/IP header use case), counter = 0:

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x17	0xf7	0xc8	0x17	0x32	0x38	0x65	0xa8
Field	*CRC*							
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0x00	0x00	0x00	0x20	0x00	0x00	0x00	0x00
Field	*Length*				*Counter*			
Byte	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Data	0x0a	0x0b	0x0c	0x0d	0x00	0x00	0x00	0x00
Field	*DataID*				*Data*			

Table 6.47: E2E Profile 7 example short with SOME/IP use case

6.9 Specification of E2E Profile 11

Profile 11 is bus-compatible to profile 1, but provides "new" profile behavior similar to profiles 4 to 7 on receiver side. Moreover, some legacy DataIDModes that are by now obsolete are omitted.

[PRS_E2E_00503] ⌈Profile 11 shall provide the following control fields, transmitted at runtime together with the protected data: Counter, CRC, Data ID (see Table 6.48).⌋ *(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Control field	Description
Counter	4 bits. (explicitly sent)
CRC	8 bits, CRC-8-SAE J1850, provided by CRC library. (explicitly sent)
Data ID	16 bits or 12 bit, unique system-wide. (either implicitly sent (16 bits) or partly explicitly sent (12 bits; 4 bits explicitly and 8 bits implicitly sent))

Table 6.48: E2E Profile 11 mechanisms

The E2E mechanisms can detect the following faults or effects of faults:

Fault	Main safety mechanisms
Repetition of information	Counter
Loss of information	Counter
Delay of information	Counter
Insertion of information	Data ID
Masquerading	Data ID, CRC
Incorrect addressing	Data ID
Incorrect sequence of information	Counter
Corruption of information	CRC
Asymmetric information sent from a sender to multiple receivers	CRC (to detect corruption at any of receivers)
Information from a sender received by only a subset of receivers and the receivers	Counter (loss on specific receivers)
Blocking access to a communication channel	Counter (loss or timeout)

Table 6.49: Detectable communication faults using Profile 11

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[2].

6.9.1 Data Layout

6.9.1.1 User data layout

In the E2E Profile 11, the user data layout (of the data to be protected) is not constrained by E2E Profile 11 - there is only a requirement, that the length of data to be protected is multiple of 1 byte.

6.9.1.2 Header layout

Profile 11 is backward compatible to the bus-layout of profile 1. This means that while all the header fields are configurable, the profile variants of profile 1 are also applicable. Namely, profile 1 variant 1A and variant 1C.

Byte Order	0								1							
Transmission Order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit Order	7	6	5	4	3	2	1	0	15	14	12	12	11	10	9	8
	E2E CRC								DataIDNibble				Counter			

Figure 6.106: E2E Profile 11 header

The figure above shows Profile 11 variant 11C where the configuration is given as: The E2E header fields (e.g. CRC) are encoded like in CAN and FlexRay, i.e.:

1. CRCOffset = 0
2. CounterOffset = 8 by FlexrayCAN bus.
3. DataIDNibbleOffset = 12

For Profile 11 Variant 11A, DataIDNibble is not used. Instead, user data can be placed there.

[PRS_E2E_00540] ⌈The E2E Profile variant 11A is defined as follows:

1. CRC is the 0th byte in the signal group (i.e. starts with bit offset 0)
2. Alive counter is located in lowest 4 bits of 1st byte (i.e. starts with bit offset 8)
3. E2E_P11DataIDMode = E2E_P11_DATAID_BOTH
4. SignalIPdu.unusedBitPattern = 0xFF.

⌋*(RS_E2E_08528)*

Below is an example compliant to 11A:

Byte Order	0								1							
Transmission Order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit Order	7	6	5	4	3	2	1	0	15	14	12	12	11	10	9	8
0	E2E CRC												Counter			

Figure 6.107: E2E Profile 11 Variant A

[PRS_E2E_00541] ⌈The E2E Profile variant 11C is defined as follows:

1. CRC is the 0th byte in the signal group (i.e. starts with bit offset 0)
2. Alive counter is located in lowest 4 bits of 1st byte (i.e. starts with bit offset 8)
3. The Data ID nibble is located in the highest 4 bits of 1st byte (i.e. starts with bit offset 12)

4. E2E_P11DataIDMode = E2E_P11_DATAID_NIBBLE
5. SignalIPdu.unusedBitPattern = 0xFF

⌋(*RS_E2E_08528*)

E2E Profile variants 11A and 11C relate to the recommended Configuration of E2E Profile 11 configuration settings 11A and 11C in system template (system template is more specific).

The transmission order shown above represents the order in which bits are transmitted. For comparability to the figures of profile 1, also the bit order is given. The E2E header fields (e.g. CRC) are encoded like in CAN and FlexRay, i.e.:

1. Little Endian (least significant byte fist) applicable for both implicit and explicit header fields - imposed by profile
2. MSB Fist (most significant bit within byte first) - imposed by Flexray/CAN bus.

6.9.2 Counter

In E2E Profile 11, the counter is initialized, incremented, reset and checked by E2E profile. The counter is not manipulated or used by the caller of the E2E Supervision.

[PRS_E2E_00504] ⌈In E2E Profile 11, on the sender side, for the first transmission request of a data element the counter shall be initialized with 0 and shall be incremented by 1 for every subsequent send request. When the counter reaches the maximum value (0x0E), then it shall restart with 0 for the next send request.⌋(*RS_E2E_08539*)

Note that the counter value 0x0F is reserved as a special invalid value, and must never be used by the E2E profile 11.

In E2E Profile 11, on the receiver side, by evaluating the counter of received data against the counter of previously received data, the following is detected:

1. Repetition:

   a. no new data has arrived since last invocation of E2E Supervision check function,

   b. the data is repeated

2. OK:

   a. counter is incremented by one (i.e. no data lost),

   b. counter is incremented more than by one, but still within allowed limits (i.e. some data lost),

3. Error: a. counter is incremented more than allowed (i.e. too many data lost).

Case 1 corresponds to the failed alive counter check, and case 3 correspond to failed sequence counter check.

The above requirements are specified in more details by the UML diagrams in the following document sections.

6.9.3 Data ID

The unique Data IDs are to verify the identity of each transmitted safety-related data element.

[PRS_E2E_00583] ⌈The following two Data ID modes shall be supported:

1. E2E_P11_DATAID_BOTH: both bytes of the 16 bit Data ID are used in the CRC calculation: first the low byte and then the high byte.
2. E2E_P11_DATAID_NIBBLE:

   the high nibble of high byte of DataID is not used (it is 0x0), as the DataID is limited to 12 bits,

   the low nibble of high byte of DataID is transmitted explicitly and covered by CRC calculation when computing the CRC over Data.

   the low byte is not transmitted, but it is included in the CRC computation as start value.

⌋*(RS_E2E_08539)*

[PRS_E2E_0507] ⌈In the E2E profile 11, the Data IDs shall be globally unique within the network of communicating system (made of several ECUs each sending different data).⌋*(RS_E2E_08539)*

In case of usage of E2E Supervision for protecting data elements (i.e invocation from RTE), due to multiplicity of communication (1:1 or 1:N), a consumer of a data element expects only a specific data element, which is checked by E2E Supervision using Data ID.

In case of usage of E2E Supervision for protecting messages (i.e. invocation from COM), the receiver COM expects at a reception only a specific message, which is checked by E2E Supervision using Data ID.

6.9.4 Length

In Profile 11 there is no explicit transmission of the length.

6.9.5 CRC

E2E Profile 11 uses a 8-bit CRC, to ensure a sufficient detection rate and sufficient Hamming Distance.

[PRS_E2E_00508] ⌈E2E Profile 11 shall use the Crc_CalculateCRC8 function of the SWS CRC Supervision for calculating the CRC (CRC-8-SAE J1850).⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_00505] ⌈In the E2E Profile 11 with DataIDMode set to E2E_P11_DATAID_BOTH, the Data ID shall be implicitly transmitted, by adding first the Data ID low byte, then the Data ID high byte before the user data in the CRC calculation⌋*(RS_E2E_08539)*

[PRS_E2E_00506] ⌈In E2E Profile 11 with DataIDMode set to E2E_P11_DATAID_NIBBLE, the lower nibble of the high byte of the DataID shall be placed in the transmitted data at bit position DataIDNibbleOffset, and the CRC calculation shall be done by first calculating over the low byte of the Data ID, then a 0-byte, and then the user data.⌋*(RS_E2E_08539)*

Note: the byte containing the CRC is always omitted from the CRC calculation.

6.9.6 Timeout detection

The previously mentioned mechanisms (for Profile 11: CRC, Counter, Data ID) enable to check the validity of received data element, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for Data Elements or respectively messages, but instead if the receiver reads the currently available data (i.e. checks if new data is available). Then, by means of the counter, the receiver can detect loss of communication and timeouts.

The attribute State->NewDataAvailable == E2E_P11STATUS_NONEWDATA means that the transmission medium (e.g RTE) reports that no new data element is available at the transmission medium. The attribute State->Status = E2E_P11STATUS_REPEATED means that 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 an unavailability of valid data that was updated since the previous cycle.

6.9.7 E2E_P11Protect

The function E2E_P11Protect() performs the steps as specified by the following six diagrams in this section.

[PRS_E2E_00509] ⌈The function E2E_P11Protect() shall have the overall behavior as shown in Figure 6.108.⌋*(RS_E2E_08539)*

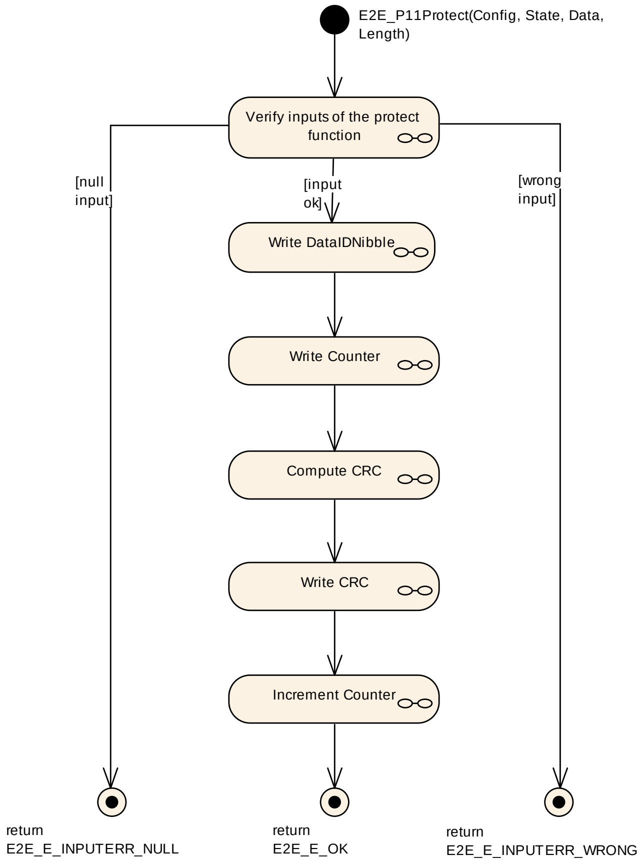

Figure 6.108: E2E Profile 11 Protect

[PRS_E2E_00510] ⌈The step "Verify inputs of the protect function" in E2E_P11Protect() shall behave as shown in Figure 6.109.⌋*(RS_E2E_08539)*

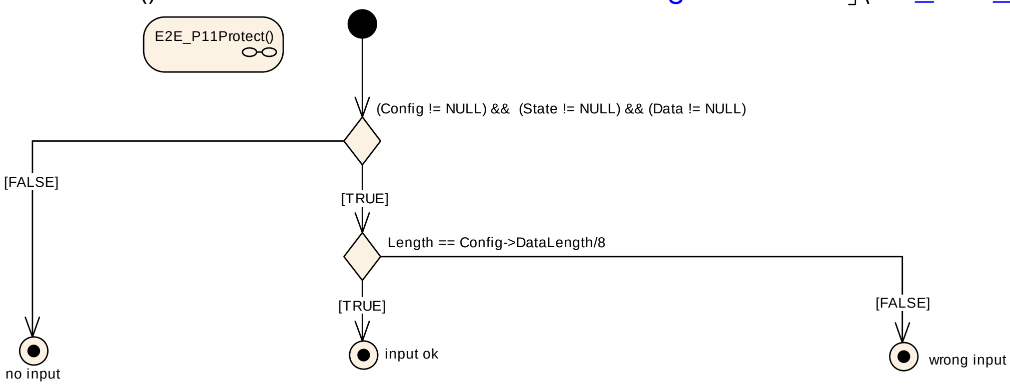

Figure 6.109: E2E Profile 11 Protect step "Verify inputs of the protect function"

[PRS_E2E_00511] ⌈The step „Write DataIDNibble" in E2E_P11Protect() shall behave as shown in Figure 6.110.⌋*(RS_E2E_08539)*

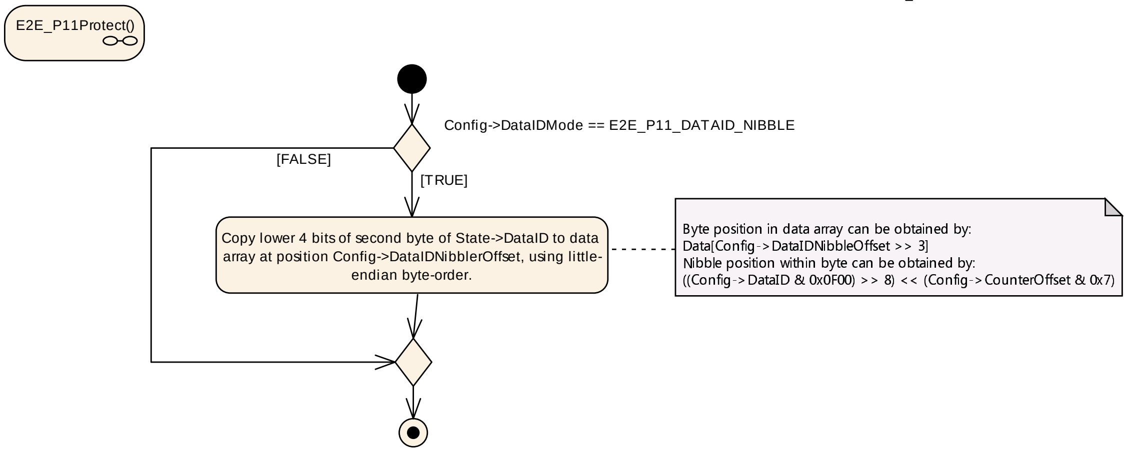

Figure 6.110: E2E Profile 11 Protect step "Write DataIDNibble"

[PRS_E2E_00512] ⌈The step "Write Counter" in E2E_P11Protect() shall behave as shown in Figure 6.111.⌋*(RS_E2E_08539)*

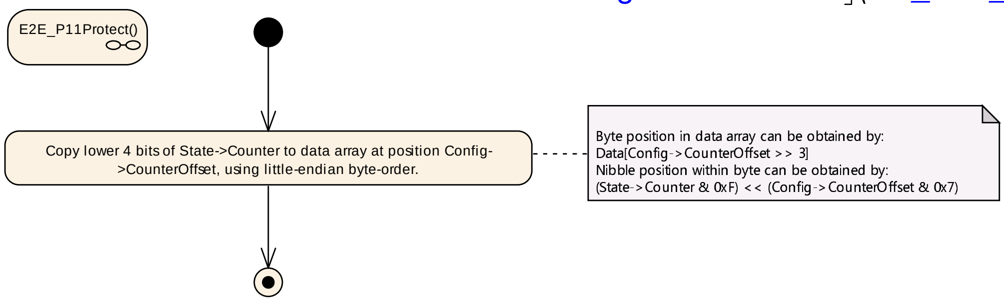

Figure 6.111: E2E Profile 11 Protect step "Write Counter"

[PRS_E2E_00513] ⌈The step "Compute CRC" in E2E_P11Protect() and in E2E_P11Check shall behave as shown in Figure 6.112.⌋ *(RS_E2E_08539)*

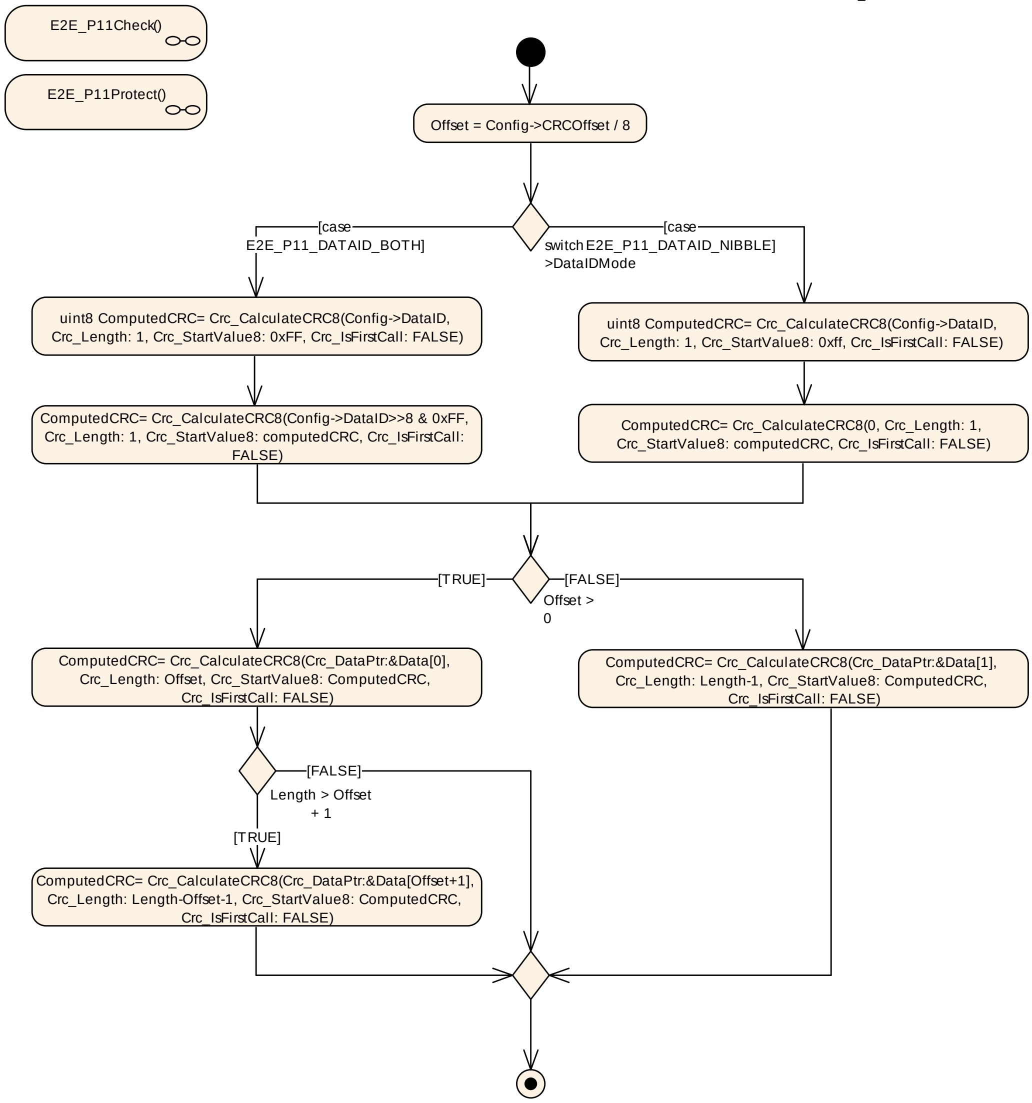

Figure 6.112: E2E Profile 11 Protect and Check step "Compute CRC"

[PRS_E2E_00514] ⌈The step "Write CRC" in E2E_P11Protect() and E2E_P11Forward() shall behave as shown in Figure 6.113.⌋ *(RS_E2E_08539)*

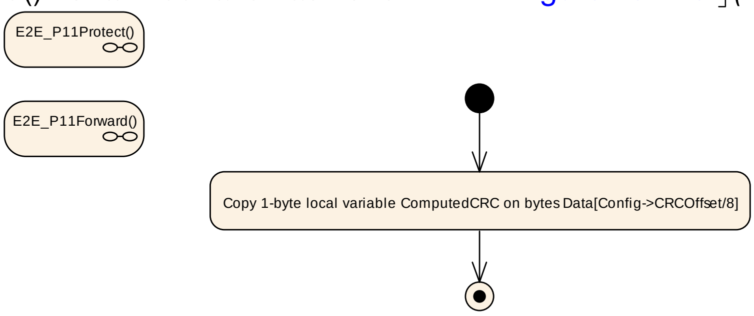

Figure 6.113: E2E Profile 11 Protect and Forward step "Write CRC"

[PRS_E2E_00515] ⌈The step "Increment Counter" in E2E_P11Protect() and E2E_P11Forward() shall behave as shown in Figure 6.114.⌋*(RS_E2E_08539)*

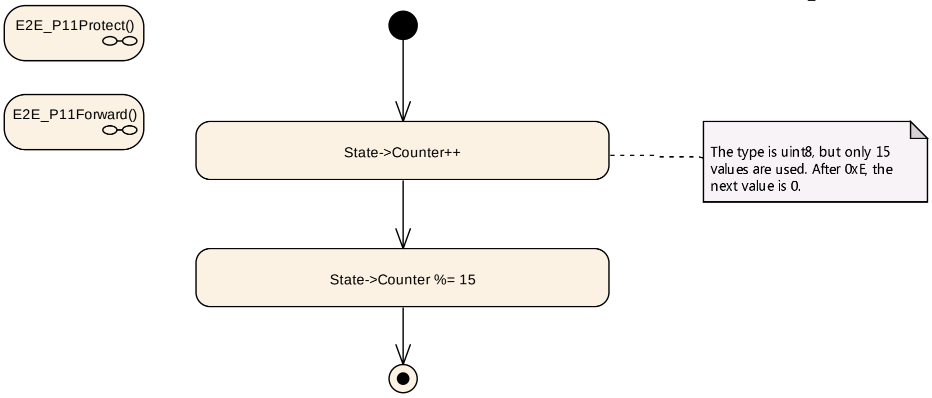

Figure 6.114: E2E Profile 11 Protect and Forward step "Increment Counter"

6.9.8 E2E_P11Forward

The E2E_P11Forward() function of E2E Profile 11 is called by a SW-C in order to protect its application data and forward an received E2E-Status for use cases like translation of signal based to service oriented communication. If the received E2E status equals E2E_P_OK the behavior of the function shall be the same like E2E_P11Protect(). The function E2E_P11Forward() performs the steps as specified by the following five diagrams in this section.

[PRS_E2E_00630] Draft ⌈The function E2E_P11Forward() shall have the overall behavior as shown in Figure 6.115.⌋*(RS_E2E_08739)*

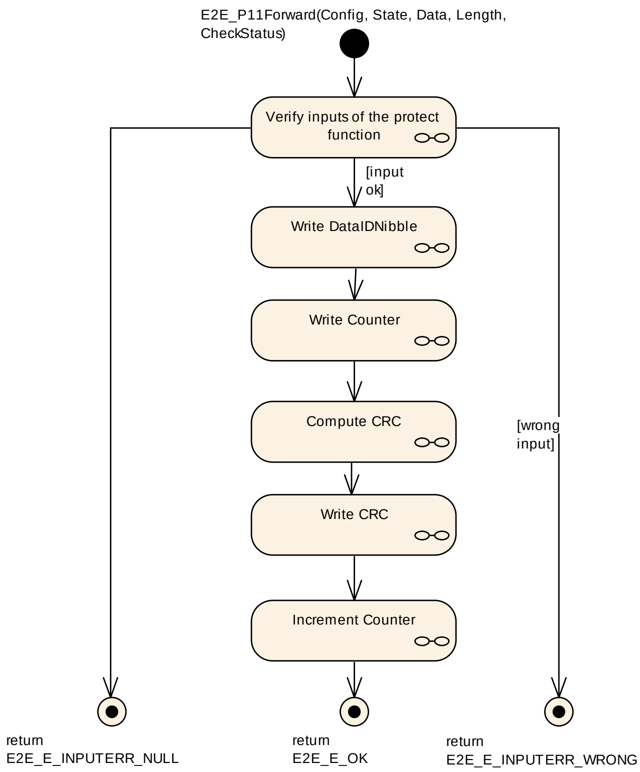

Figure 6.115: E2E Profile 11 Forward

Following steps are described in Section in Section 6.9.7

- ”Write CRC” see [PRS_E2E_00514]
- ”Increment Counter” see [PRS_E2E_00515]

[PRS_E2E_00631] Draft ⌈The step ”Verify inputs of the forward function” in E2E_P11Forward() shall behave as shown in Figure 6.116.⌋*(RS_E2E_08739)*

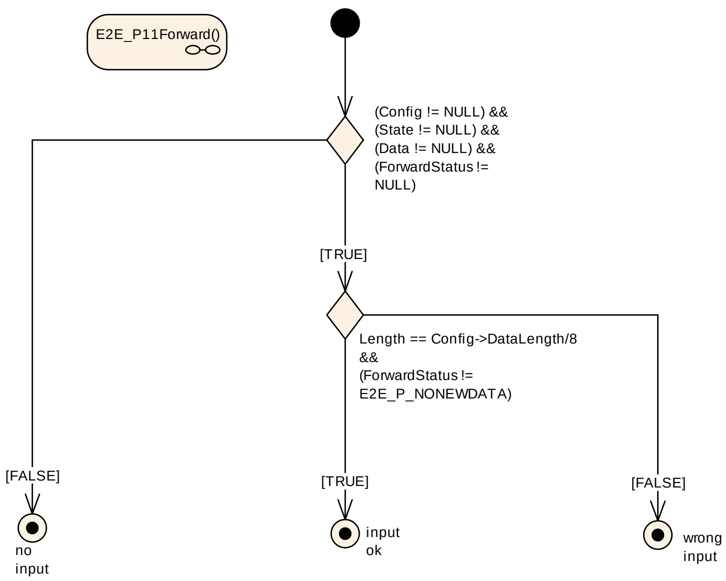

Figure 6.116: E2E Profile 11 Forward step ”Verify inputs of the forward function”

[PRS_E2E_00632] **Draft** ⌈The step „Write DataIDNibble" in E2E_P11Forward() shall behave as shown in Figure 6.117.⌋*(RS_E2E_08539)*

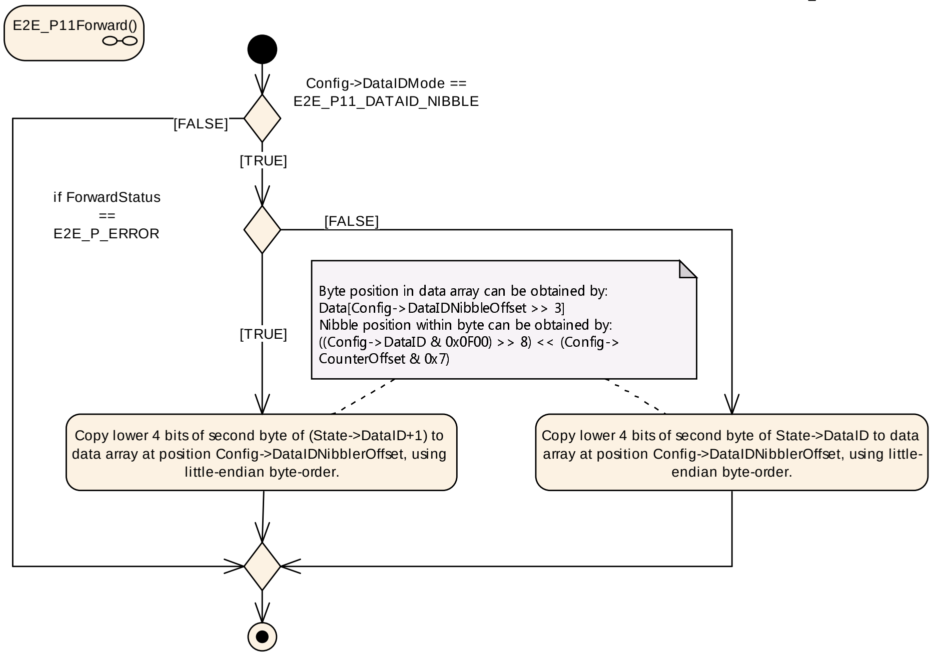

Figure 6.117: E2E Profile 11 Forward step "Write DataIDNibble"

[PRS_E2E_00633] **Draft** ⌈The step "Write Counter" in E2E_P11Forward() shall behave as shown in Figure 6.118.⌋*(RS_E2E_08739)*

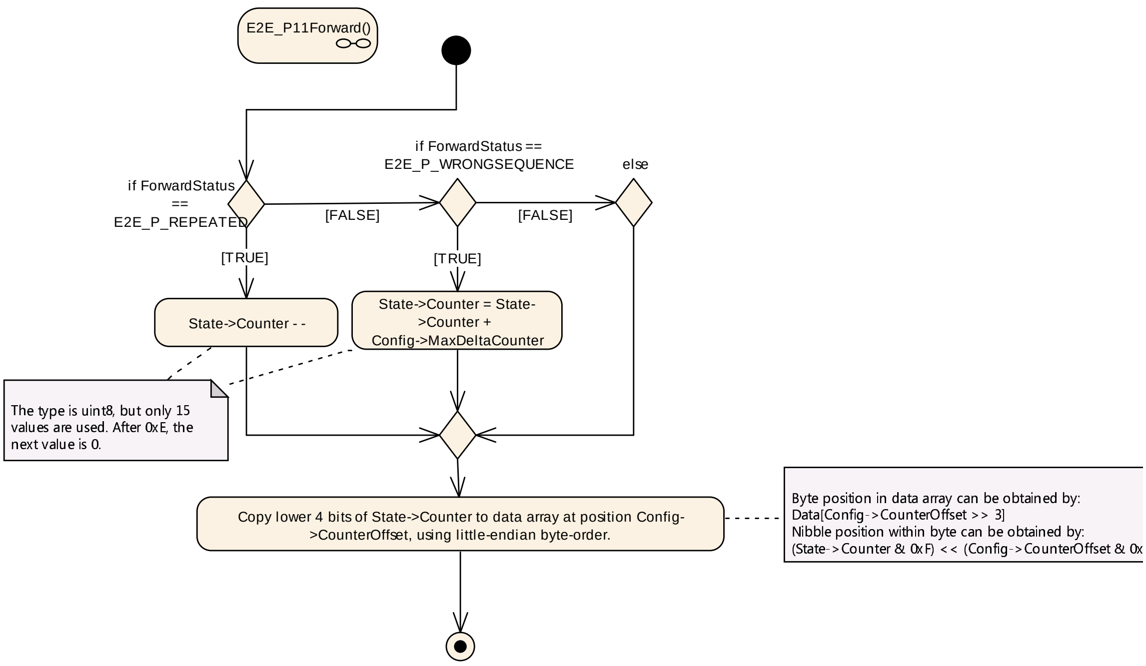

Figure 6.118: E2E Profile 11 Forward step "Write Counter"

[PRS_E2E_00634] **Draft** ⌈The step ”Compute CRC” in E2E_P11Forward() shall behave as shown in Figure 6.119.⌋*(RS_E2E_08739)*

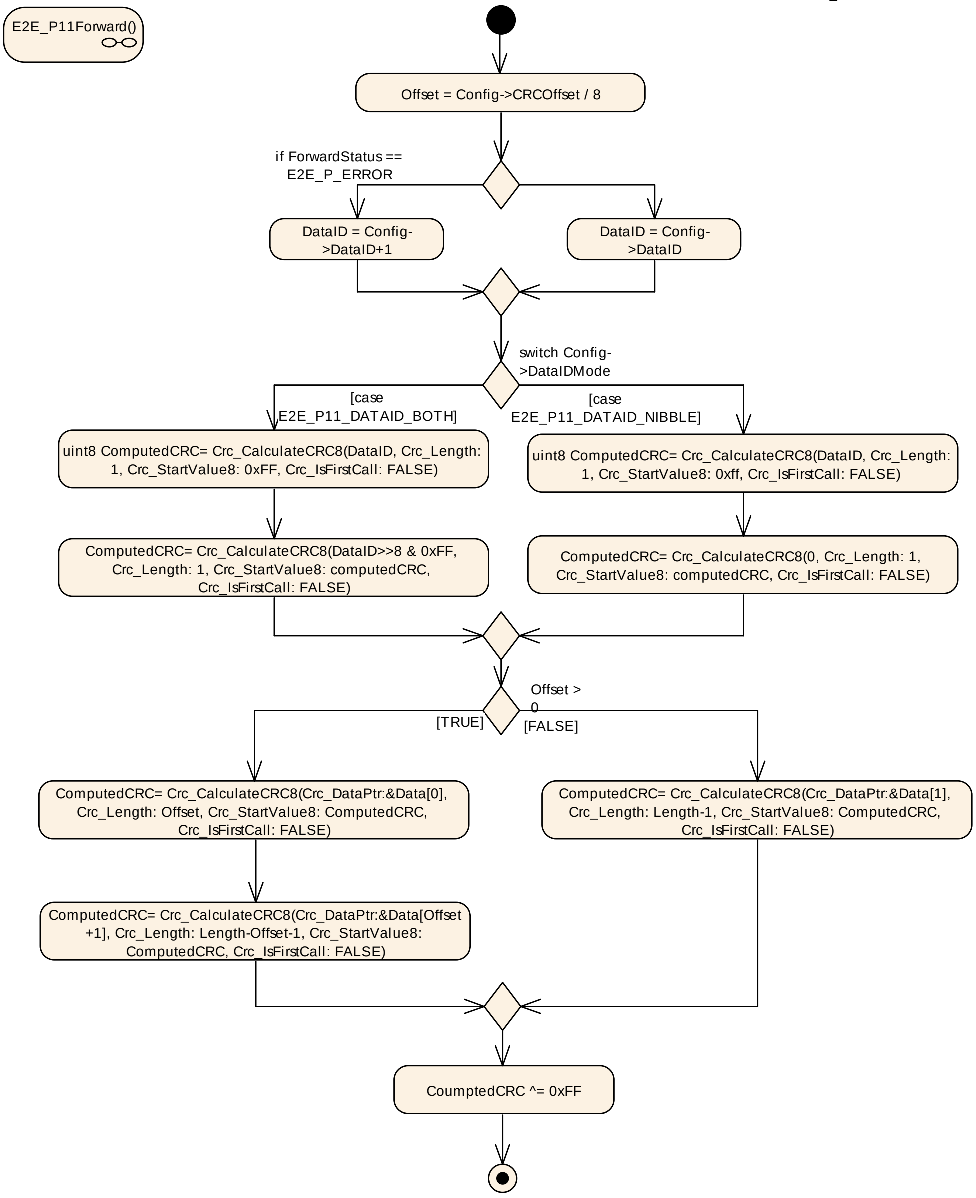

Figure 6.119: E2E Profile 11 Forward step ”Compute CRC”

6.9.9 E2E_P11Check

The function E2E_P11Check performs the actions as specified by the following six diagrams in this section.

[PRS_E2E_00516] ⌈The function E2E_P11Check() shall have the overall behavior as shown in Figure 6.120.⌋*(RS_E2E_08539)*

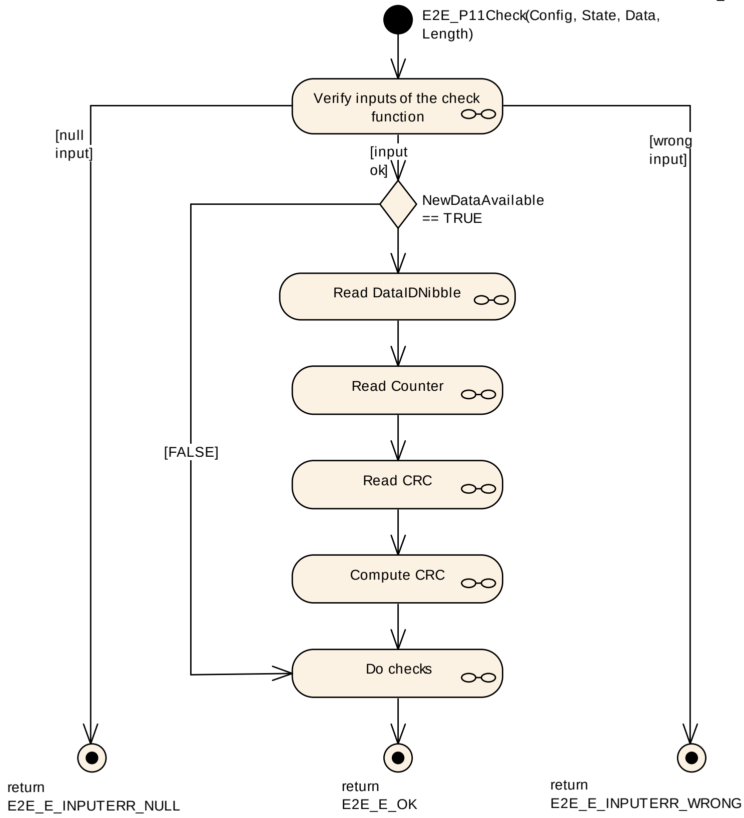

Figure 6.120: E2E Profile 11 Check

[PRS_E2E_00517] ⌈The step ”Verify inputs of the check function” in E2E_P11Check() shall behave as shown in Figure 6.121.⌋*(RS_E2E_08539)*

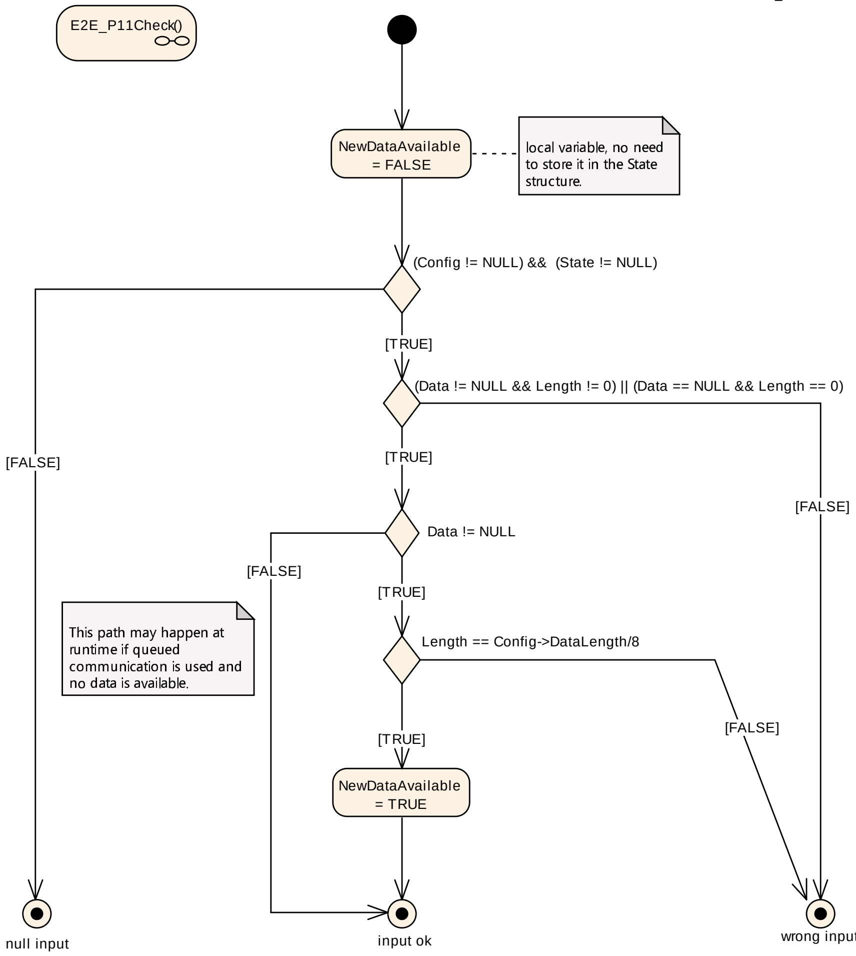

Figure 6.121: E2E Profile 11 Check step ”Verify inputs of the check function”

[PRS_E2E_00582] ⌈The step ”Read DataIDNibble” in E2E_P11Check() shall behave as shown in Figure 6.122.⌋*(RS_E2E_08539)*

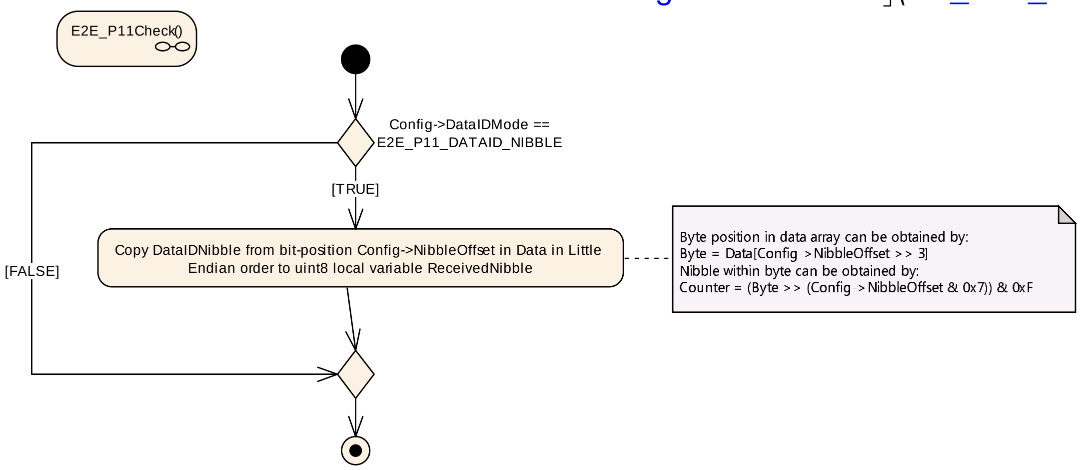

Figure 6.122: E2E Profile 11 Check step ”Read DataIDNibble”

[PRS_E2E_00518] ⌈The step "Read Counter" in E2E_P11Check() shall behave as shown in Figure 6.123.⌋*(RS_E2E_08539)*

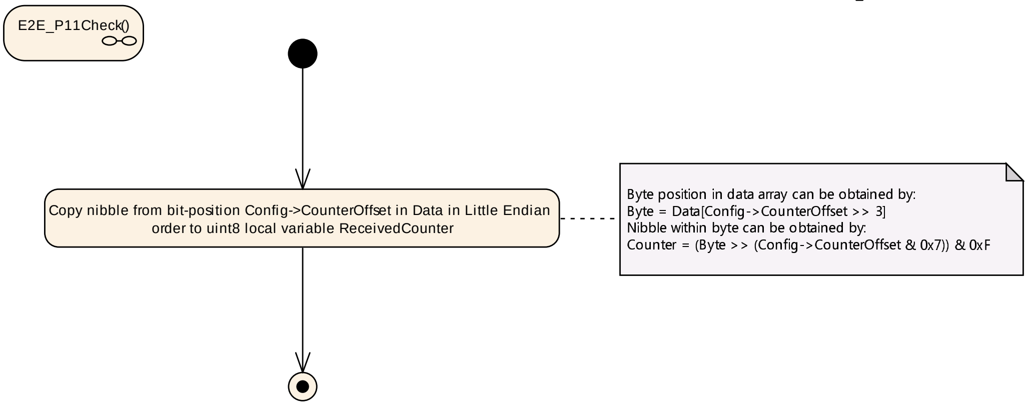

Figure 6.123: E2E Profile 11 Check step "Read Counter"

[PRS_E2E_00519] ⌈The step "Read CRC" in E2E_P11Check() shall behave as shown in Figure 6.124.⌋*(RS_E2E_08539)*

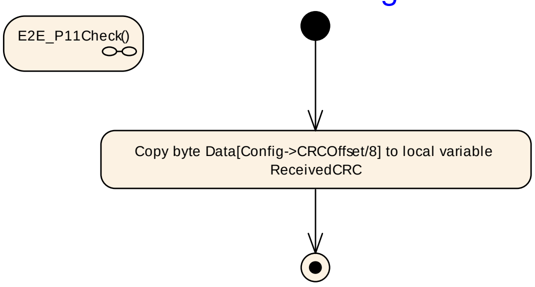

Figure 6.124: E2E Profile 11 Check step "Read CRC"

[PRS_E2E_00521] ⌈The step ”Do Checks’ in E2E_P11Check() shall behave as shown in Figure 6.125.⌋*(RS_E2E_08539)*

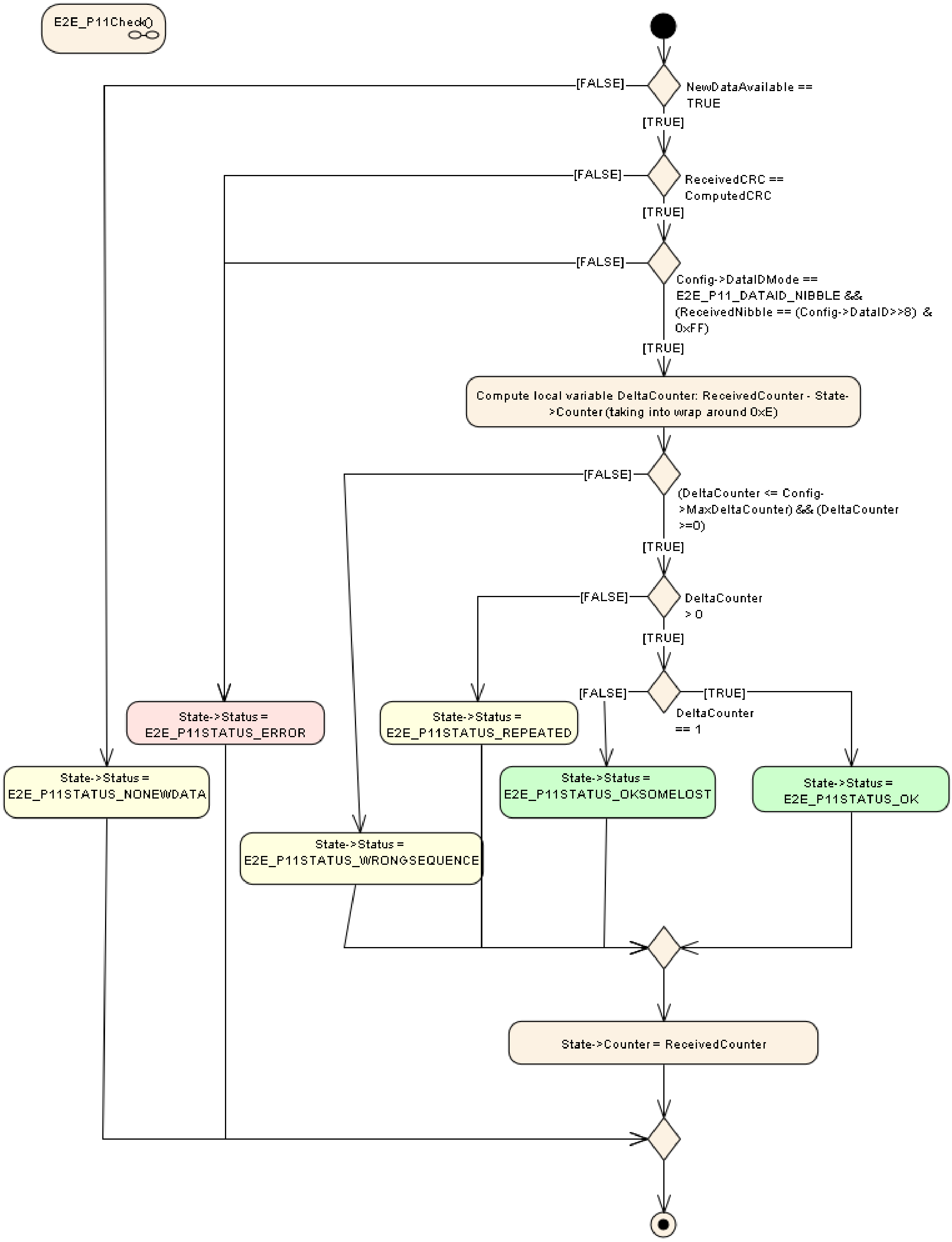

Figure 6.125: E2E Profile 11 Check step ”Do Checks”

6.9.9.1 Profile 11 Check Status Enumeration

[PRS_E2E_00594] ⌈The step "Do Checks" in E2E_P11Check shall set State->Status to one of the following enumeration values (see Table 6.50).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_P11STATUS_OK	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented by 1).
E2E_P11STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed. This may be considered similar to E2E_P11STATUS_REPEATED.
E2E_P11STATUS_ERROR	Error	Error not related to counters occurred (e.g. wrong crc, wrong length, wrong options, wrong Data ID).
E2E_P11STATUS_REPEATED	Error	The checks of the Data in this cycle were successful, with the exception of the repetition.
E2E_P11STATUS_OKSOMELOST	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented within the allowed configured delta).
E2E_P11STATUS_WRONGSEQUENCE	Error	The checks of the Data in this cycle were successful, with the exception of counter jump, which changed more than the allowed delta

Table 6.50: E2E Profile 11 Check Status Enumeration

6.9.10 E2E Profile 11 Protocol Examples

The default configuration assumed for the following examples, if not otherwise stated to be different:

E2E_P11ConfigType field	Value
CounterOffset	8
CRCOffset	0
DataID	0x123
DataIDNibbleOffset	12
DataIDMode	E2E_P11DATAID_BOTH
DataLength	64
MaxDeltaCounter	1
MaxNoNewOrRepeatedData	15

SyncCounterInit	0

Table 6.51: E2E Profile 11 protocol example configuration

E2E_P11ProtectStateType field	Value
Counter	0

Table 6.52: E2E Profile 11 example state initialization

Byte							
0	1	2	3	4	5	6	7
0xcc	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.53: E2E Profile 11 protect result DataIDMode = E2E_P11DATAID_BOTH, counter 0

Result data of E2E_P11Protect() with data equals all zeros (0x00), counter = 1:

Byte							
0	1	2	3	4	5	6	7
0x91	0x01	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.54: E2E Profile 11 protect result DataIDMode = E2E_P11DATAID_BOTH, counter 1

6.9.10.1 DataIDMode set to E2E_P11DATAID_NIBBLE

Result data of E2E_P11Protect() with data equals all zeros (0x00), counter = 0:

Byte							
0	1	2	3	4	5	6	7
0x2a	0x10	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.55: E2E Profile 11 protect result DataIDMode = E2E_P11DATAID_NIBBLE, counter 0

Result data of E2E_P11Protect() with data equals all zeros (0x00), counter = 1:

Byte							
0	1	2	3	4	5	6	7
0x77	0x11	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.56: E2E Profile 11 protect result DataIDMode = E2E_P11DATAID_NIBBLE, counter 1

6.9.10.2 DataIDMode set to E2E_P11DATAID_NIBBLE, Offset set to 64

This is a typical use-case for using P11 with SOME/IP serializer, which puts an 8 byte header in front of the serialized user data. “Offset 64” means CRCOffset set to 64, CounterOffset set to 72, DataIDNibbleOffset set to 76. Result data of E2E_P11Protect() with data equals all zeros (0x00), counter = 0:

Byte	0	1	2	3	4	5	6	7
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	8	9	10	11	12	13	14	15
Data	0x7d	0x10	0x00	0x00	0x00	0x00	0x00	0x00
Field	*CRC*	*DataID-Nibble |Counter*	*Data*					

Table 6.57: E2E Profile 11 example protect result with short data and SOME/IP

6.10 Specification of E2E Profile 22

[PRS_E2E_00522] ⌈Profile 22 shall provide the following control fields, transmitted at runtime together with the protected data: Counter, CRC, Data ID (see Table 6.58).⌋ *(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Control field	Description
Counter	4 bits. (explicitly sent)
CRC	8 bits, polynomial in normal form 0x2F (Autosar notation), provided by CRC library. (explicitly sent)
Data ID List	16 8 bits values, linked to Counter value. Effectively 16 different values, one for each counter value. The Data ID List must be unique system-wide.

Table 6.58: E2E Profile 22 mechanisms

The E2E mechanisms can detect the following faults or effects of faults:

E2E Mechanism	Detected communication faults
Counter	Repetition, loss, insertion, incorrect sequence, blocking

Transmission on a regular bases and timeout monitoring using E2E-Library [5]	Loss, delay, blocking
Data ID + CRC	Masquerade and icorrect addressing, insertion
CRC	Corruption, asymmetric information [6]

Table 6.59: Detectable communication faults using Profile 22

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[2].

6.10.1 Data Layout

6.10.1.1 User data layout

In the E2E Profile 22, the user data layout (of the data to be protected) is not constrained by E2E Profile 22. The total length of transmitted data must be a multiple of 8 bit (full bytes). Also, as the header only used 12 bit, there are 4 bit unused and available for user data in the byte where the 4 bit of the counter are placed.

6.10.1.2 Header layout

Profile 22 is backward compatible to the bus-layout of profile 2. In addition, the configuration field offset can be used to offset the header fields, then breaking with backward-compatibility to profile 2 bus-layout.

Byte Order	0								1							
Transmission Order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bit Order	7	6	5	4	3	2	1	0	15	14	12	12	11	10	9	8
0	E2E CRC												Counter			

Figure 6.126: E2E Profile22 header with offset 0.

The figure above shows Profile 22 with offset configured with 0. Offset is always given in bit and a multiple of 8 (full bytes).

The transmission order shown above represents the order in which bits are transmitted. For comparability to the figures of profile 2, also the bit order is given. The E2E header fields (e.g. CRC) are encoded like in CAN and FlexRay, i.e.:

1. Little Endian (least significant byte fist) applicable for both implicit and explicit header fields - imposed by profile

[5] Implementation by sender and receiver
[6] for a set of data protected by same CRC

2. MSB Fist (most significant bit within byte first) - imposed by Flexray/CAN bus.

6.10.2 Counter

In E2E Profile 22, the counter is initialized, incremented, reset and checked by E2E profile check and protect functions. The counter is not manipulated or used by the caller of the E2E Supervision. .

[PRS_E2E_00523] ⌈In E2E Profile 22, on the sender side, for the first transmission request of a data element the counter shall be initialized with 0 and shall be incremented by 1 for every subsequent send request. When the counter reaches the maximum value (0x0F), then it shall restart with 0 for the next send request.⌋*(RS_E2E_08539)*

Note that the counter value 0x0F is not reserved as a special invalid value.

In E2E Profile 22, on the receiver side, by evaluating the counter of received data against the counter of previously received data, the following is detected:

1. Repetition:

   a. no new data has arrived since last invocation of E2E Supervision check function,

   b. the data is repeated

2. OK:

   a. counter is incremented by one (i.e. no data lost),

   b. counter is incremented more than by one, but still within allowed limits (i.e. some data lost),

3. Error: a. counter is incremented more than allowed (i.e. too many data lost).

Case 1 corresponds to the failed alive counter check, and case 3 correspond to failed sequence counter check.

The above requirements are specified in more details by the UML diagrams in the following document sections.

6.10.3 Data ID

The unique Data ID List is used to verify the identity of each transmitted safety-related data element.

[PRS_E2E_00524] ⌈In the E2E Profile 22, the Data ID shall be implicitly transmitted, by adding the Data ID after the user data in the CRC calculation.⌋*(RS_E2E_08539)*

[PRS_E2E_00525] ⌈In the E2E profiles 2 and 22, the Data ID Lists shall be globally unique within the network of communicating system (made of several ECUs each sending different data.)⌋*(RS_E2E_08539)*

In case of usage of E2E Supervision for protecting data elements (i.e invocation from RTE), due to multiplicity of communication (1:1 or 1:N), a consumer of a data element expects only a specific data element, which is checked by E2E Supervision using Data ID.

In case of usage of E2E Supervision for protecting messages (i.e. invocation from COM), the receiver COM expects at a reception only a specific message, which is checked by E2E Supervision using Data ID.

6.10.4 Length

In Profile 22 there is no explicit transmission of the length.

6.10.5 CRC

E2E Profile 22 uses an 8-bit CRC, to ensure a sufficient detection rate and sufficient Hamming Distance. The CRC polynomial is the same as used in profile 2.

[PRS_E2E_00526] ⌈E2E Profile 22 shall use the Crc_CalculateCRC8H2F() function of the SWS CRC Supervision for calculating the CRC (Polynomial 0x2F, see also SWS_E2E_00117)⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_00527] ⌈In E2E Profile 22, the CRC shall be calculated over the entire E2E header (excluding the CRC bytes), including the user data extended at the end with the coresponding Data ID from the Data ID List.⌋*(RS_E2E_08539, RS_E2E_08536)*

6.10.6 Timeout detection

The previously mentioned mechanisms (for Profile 22: CRC, Counter, Data ID) enable to check the validity of received data element, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for Data Elements or respectively messages, but instead if the receiver reads the currently available data (i.e. checks if new data is available). Then, by means of the counter, the receiver can detect loss of communication and timeouts.

The attribute State->Status = E2E_P22STATUS_NONEWDATA means that the transmission medium (e.g RTE) reported that no new data element is available at the transmission medium. The attribute State->Status = E2E_P22STATUS_REPEATED means that 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 an unavailability of valid data that was updated since the previous cycle.

6.10.7 E2E_P22Protect

The function E2E_P22Protect() performs the steps as specified by the following diagrams in this section.

[PRS_E2E_00528] ⌈The function E2E_P22Protect() shall have the overall behavior as shown in Figure 6.127.⌋*(RS_E2E_08539)*

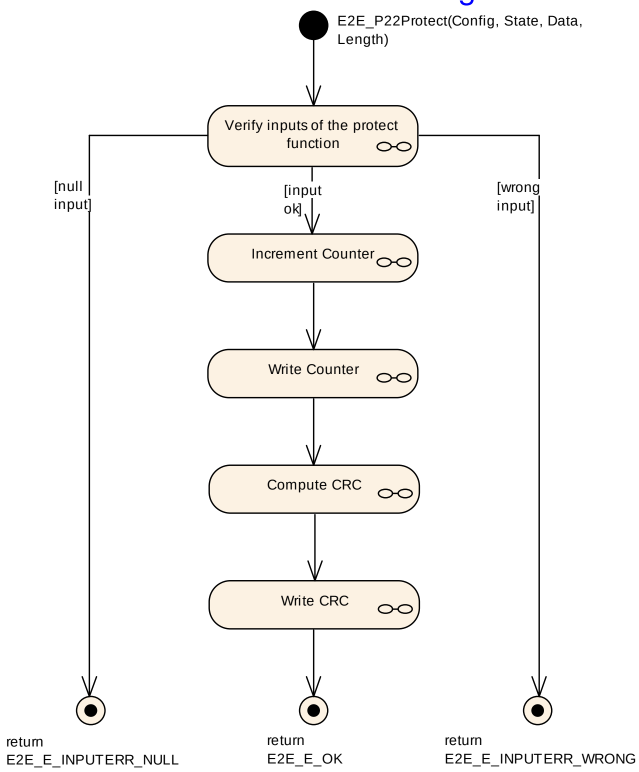

Figure 6.127: E2E Profile 22 Protect

[PRS_E2E_00529] ⌈The step ”Verify inputs of the protect function” in E2E_P22Protect() shall behave as shown in Figure 6.128.⌋*(RS_E2E_08539)*

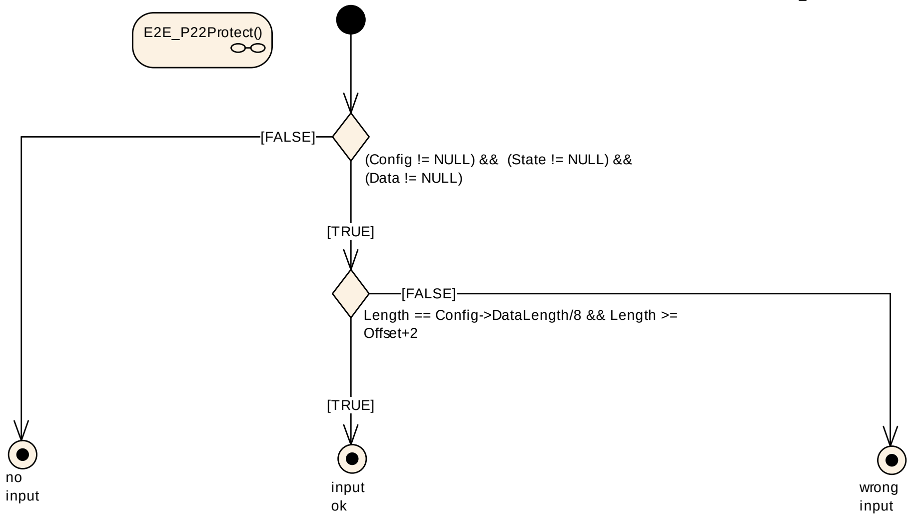

Figure 6.128: E2E Profile 22 Protect step ”Verify inputs of the protect function”

[PRS_E2E_00530] ⌈The step ”Write Counter” in E2E_P22Protect() and E2E_P22Forward() shall behave as shown in Figure 6.129.⌋*(RS_E2E_08539)*

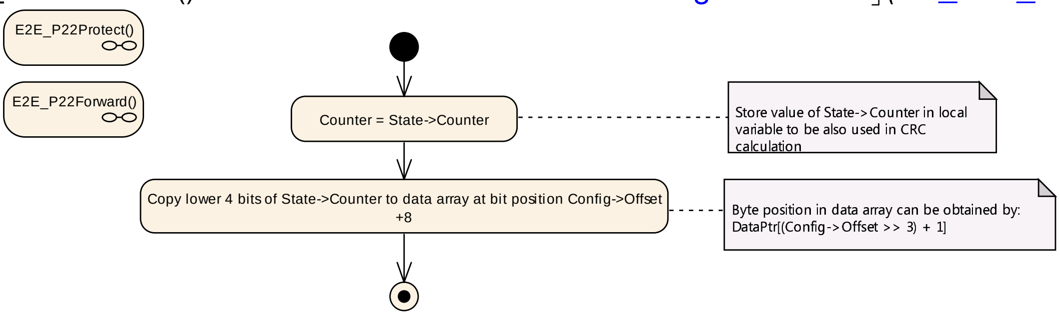

Figure 6.129: E2E Profile 22 Protect step ”Write Counter”

[PRS_E2E_00531] ⌈The step "Compute CRC" in E2E_P22Protect() and in E2E_P22Check shall behave as shown in Figure 6.130.⌋*(RS_E2E_08539)*

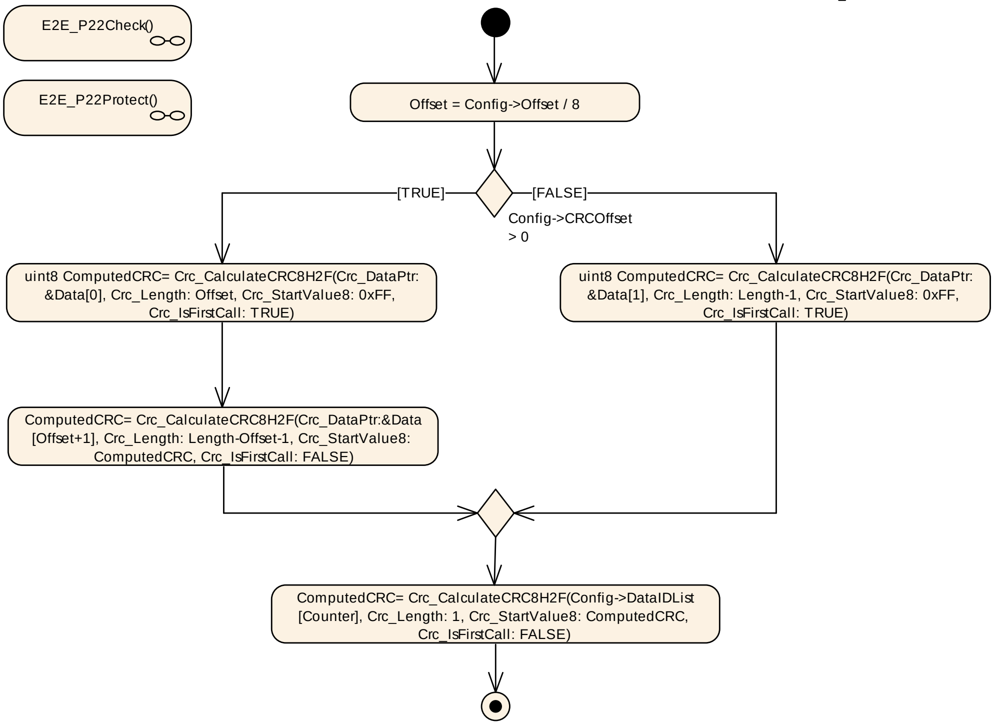

Figure 6.130: E2E Profile 22 Protect and Check step "Compute CRC"

[PRS_E2E_00532] ⌈The step "Write CRC" in E2E_P22Protect() and E2E_P22Forward() shall behave as shown in Figure 6.131.⌋*(RS_E2E_08539)*

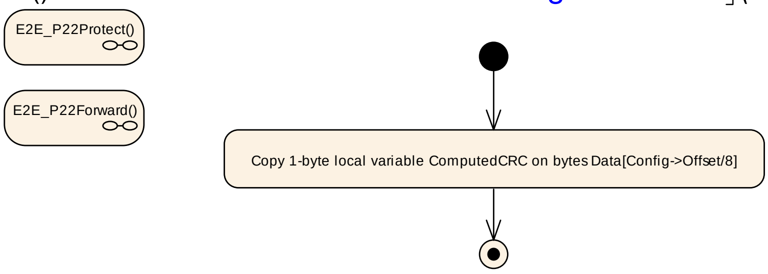

Figure 6.131: E2E Profile 22 Protect and Forward step "Write CRC"

[PRS_E2E_00533] ⌈The step ”Increment Counter” in E2E_P22Protect() shall behave as shown in Figure 6.132.⌋*(RS_E2E_08539)*

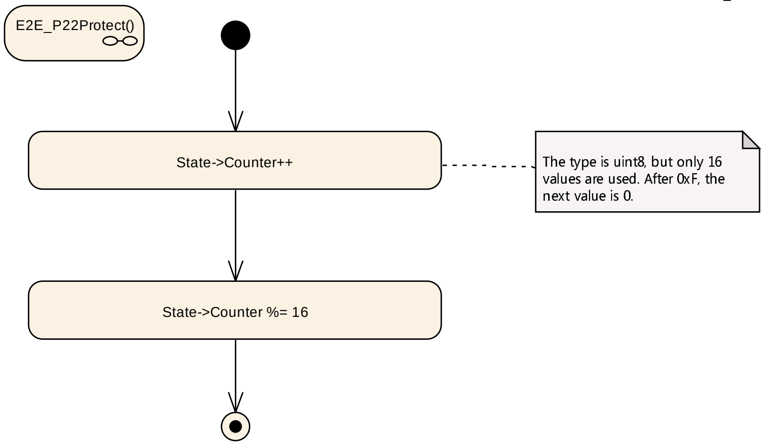

Figure 6.132: E2E Profile 22 Protect step ”Increment Counter”

6.10.8 E2E_P22Forward

The E2E_P22Forward() function of E2E Profile 22 is called by a SW-C in order to protect its application data and forward an received E2E-Status for use cases like translation of signal based to service oriented communication. If the received E2E status equals E2E_P_OK the behavior of the function shall be the same like E2E_P22Protect(). The function E2E_P22Forward() performs the steps as specified by the following four diagrams in this section.

[PRS_E2E_00635] Draft ⌈The function E2E_P22Forward() shall have the overall behavior as shown in Figure 6.133.⌋*(RS_E2E_08739)*

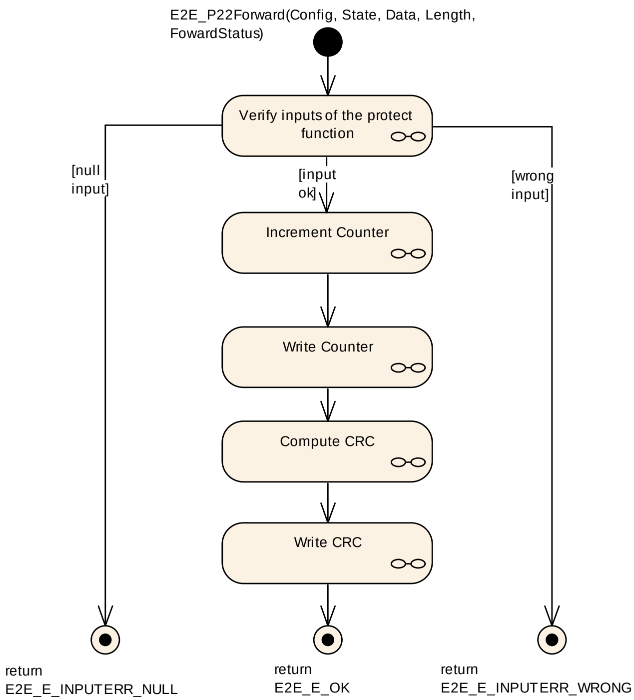

Figure 6.133: E2E Profile 22 Forward

Following steps are described in Section in Section 6.10.7

- ”Write Length” see [PRS_E2E_00530]
- ”Write CRC” see [PRS_E2E_00532]

[PRS_E2E_00636] Draft ⌈The step ”Verify inputs of the forward function” in E2E_P22Forward() shall behave as shown in Figure 6.134.⌋*(RS_E2E_08739)*

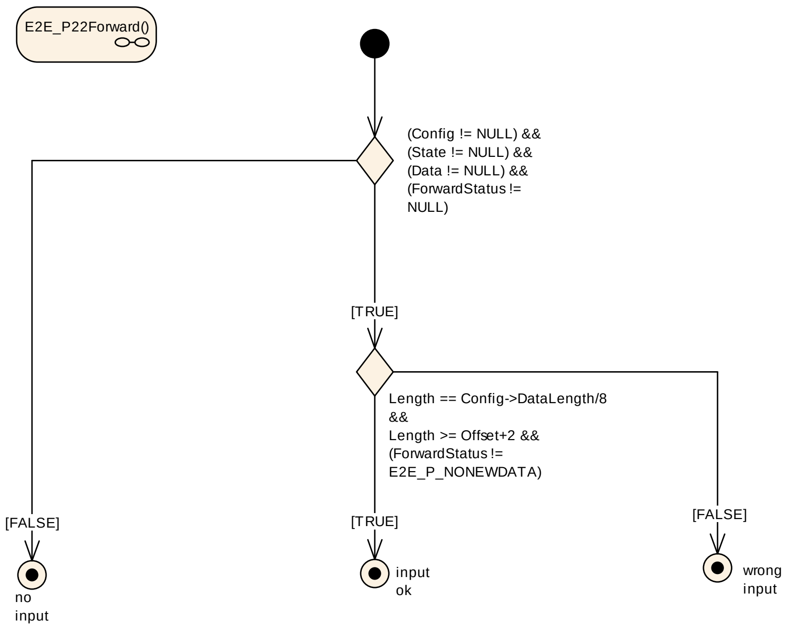

Figure 6.134: E2E Profile 22 Forward step ”Verify inputs of the forward function”

[PRS_E2E_00637] Draft ⌈The step "Increment Counter" in E2E_P22Forward() shall behave as shown in Figure 6.135.⌋*(RS_E2E_08739)*

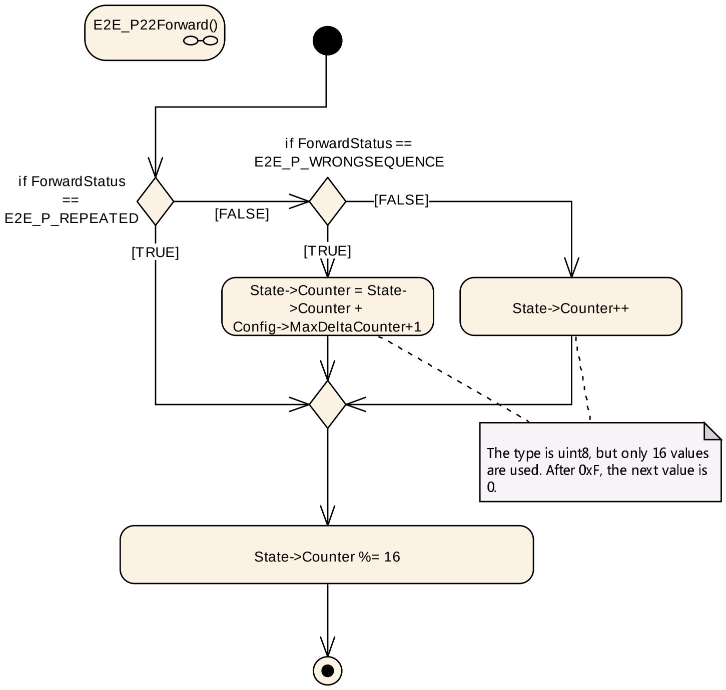

Figure 6.135: E2E Profile 22 Forward step "Increment Counter"

[PRS_E2E_00638] **Draft** ⌈The step "Compute CRC" in E2E_P22Forward() shall behave as shown in Figure 6.136.⌋*(RS_E2E_08739)*

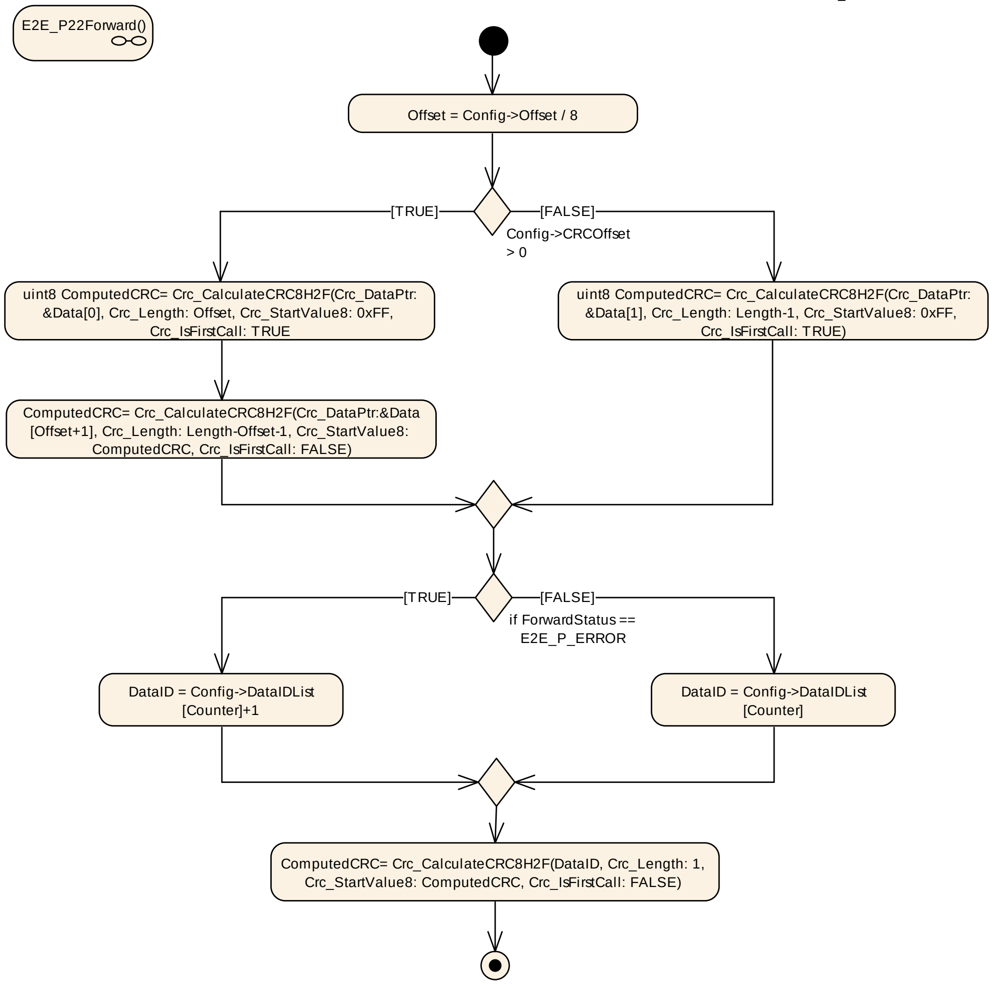

Figure 6.136: E2E Profile 22 Forward step "Compute CRC"

6.10.9 E2E_P22Check

The function E2E_P22Check performs the actions as specified by the following six diagrams in this section.

[PRS_E2E_00534] ⌈The function E2E_P22Check() shall have the overall behavior as shown in Figure 6.137.⌋*(RS_E2E_08539)*

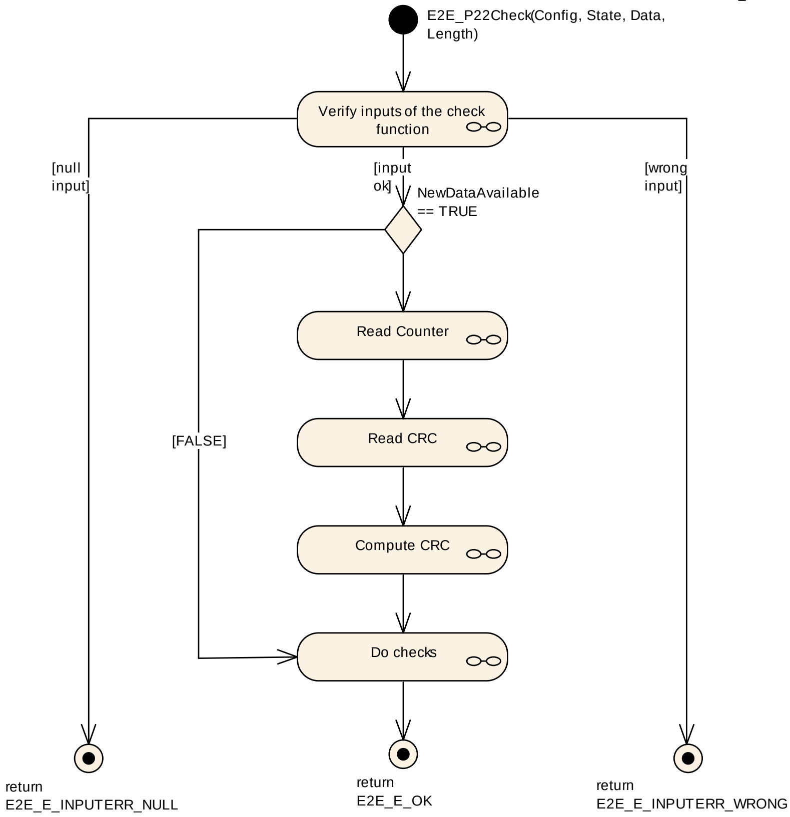

Figure 6.137: E2E Profile 22 Check

[PRS_E2E_00535] ⌈The step "Verify inputs of the check function" in E2E_P22Check() shall behave as shown in Figure 6.138.⌋*(RS_E2E_08539)*

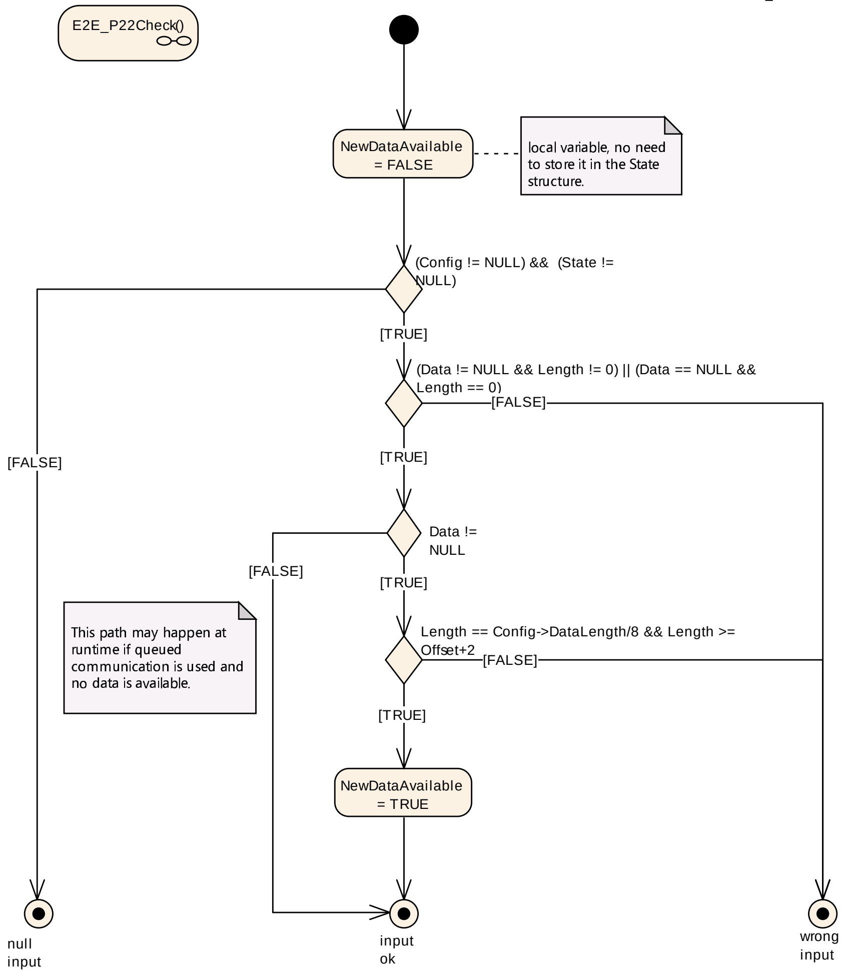

Figure 6.138: E2E Profile 22 Check step "Verify inputs of the check function"

[PRS_E2E_00536] ⌈The step "Read Counter" in E2E_P22Check() shall behave as shown in Figure 6.139.⌋*(RS_E2E_08539)*

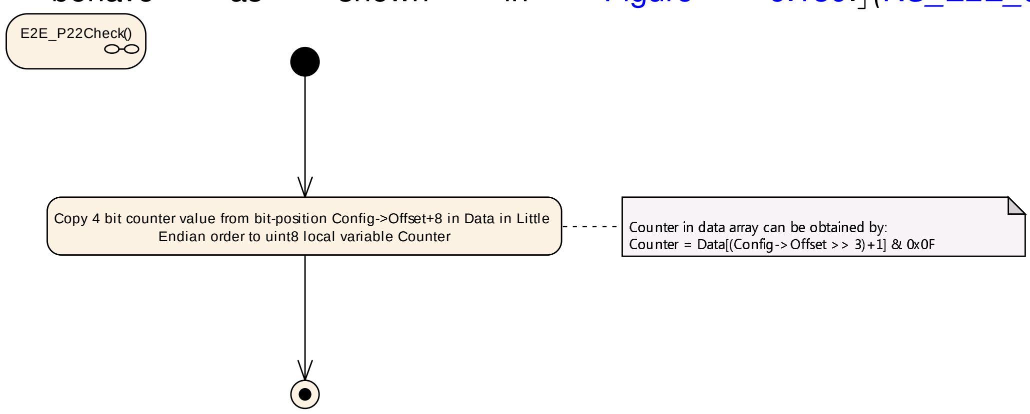

Figure 6.139: E2E Profile 22 Check step "Read Counter"

[PRS_E2E_00537] ⌈The step "Read CRC" in E2E_P22Check() shall behave as shown in Figure 6.140.⌋(*RS_E2E_08539*)

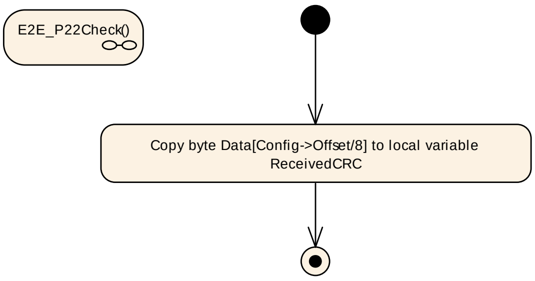

Figure 6.140: E2E Profile 22 Check step "Read CRC"

[PRS_E2E_00539] ⌈The step "Do Checks' in E2E_P22Check() shall behave as shown in Figure 6.141.⌋ *(RS_E2E_08539)*

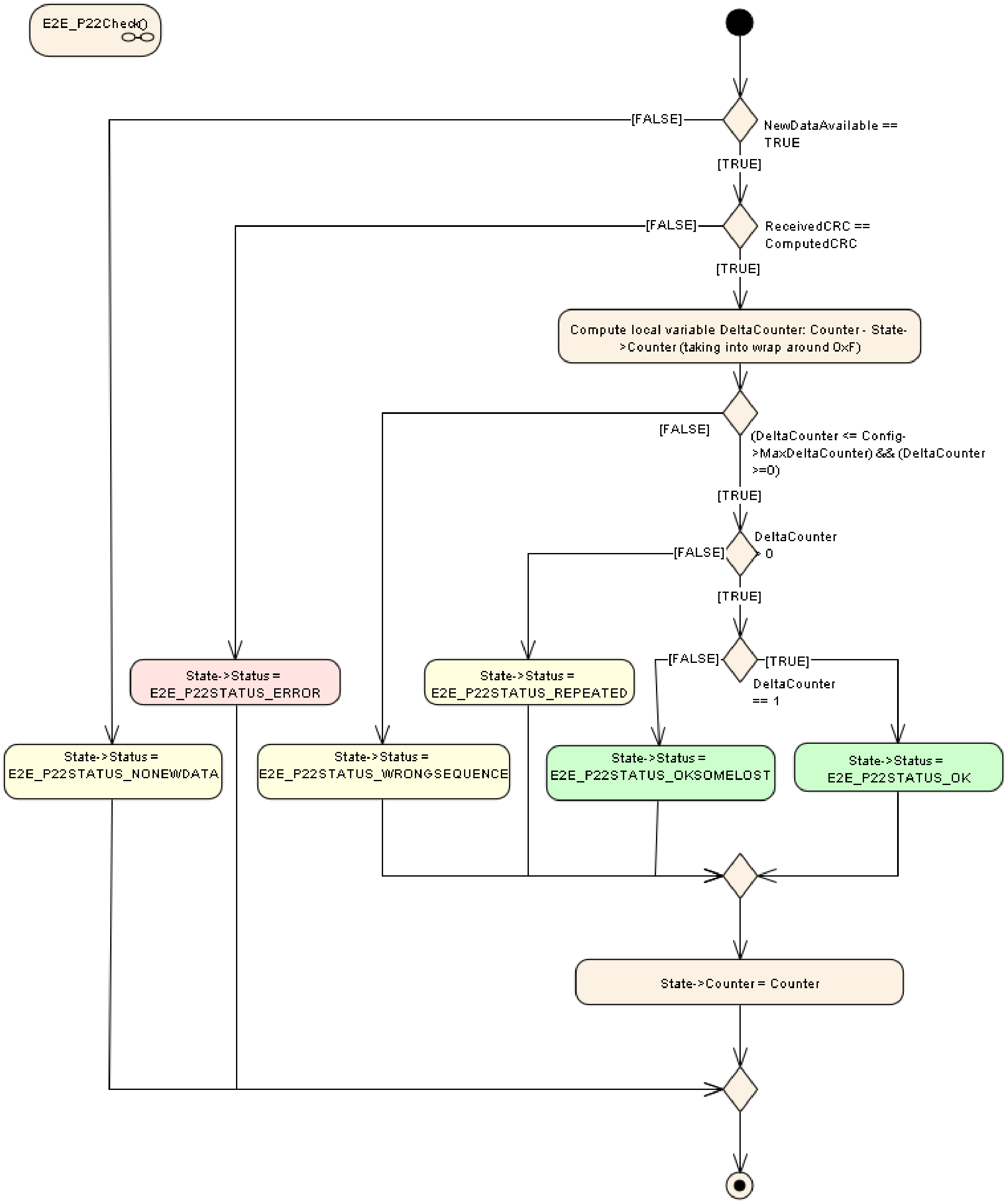

Figure 6.141: E2E Profile 22 Check step "Do Checks"

6.10.9.1 Profile 22 Check Status Enumeration

[PRS_E2E_00595] ⌈The step "Do Checks" in E2E_P22Check shall set State->Status to one of the following enumeration values (see Table 6.60).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_P22STATUS_OK	OK	The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by 1 with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that no Data has been lost since the last correct data reception.
E2E_P22STATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is not available since the last call, according to communication medium (e.g. RTE, COM). As a result, no E2E checks of Data have been consequently executed.
E2E_P22STATUS_ERROR	Error	The data has been received according to communication medium, but the CRC is incorrect.
E2E_P22STATUS_REPEATED	Error	The new data has been received according to communication medium, the CRC is correct, but the Counter is identical to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST.
E2E_P22STATUS_OKSOMELOST	OK	The new data has been received according to communication medium, the CRC is correct, the Counter is incremented by DeltaCounter (1 < DeltaCounter =MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that some Data in the sequence have been probably lost since the last correct/initial reception, but this is within the configured tolerance range.
E2E_P22STATUS_WRONGSEQUENCE	Error	The new data has been received according to communication medium, the CRC is correct, but the Counter Delta is too big (DeltaCounter > MaxDeltaCounter) with respect to the most recent Data received with Status _INITIAL, _OK, or _OKSOMELOST. This means that too many Data in the sequence have been probably lost since the last correct/initial reception.

Table 6.60: E2E Profile 22 Check Status Enumeration

6.10.10 E2E Profile 22 Protocol Examples

E2E_P22ConfigType field	Value
DataLength	64
DataIDList	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0x10
MaxDeltaCounter	1
MaxNoNewOrRepeatedData	15
SyncCounterInit	0
Offset	0

Table 6.61: E2E Profile 22 protocol example configuration

E2E_P22ProtectStateType field	Value
Counter	0

Table 6.62: E2E Profile 22 example state initialization

Result data of E2E_P22Protect() with data equals all zeros (0x00), counter starting with 1 (note: first used counter is 1, although counter field is initialized with 0, as counter is incremented before usage):

Counter	Byte							
	0	1	2	3	4	5	6	7
1	0x1b	0x01	0x00	0x00	0x00	0x00	0x00	0x00
2	0x98	0x02	0x00	0x00	0x00	0x00	0x00	0x00
3	0x31	0x03	0x00	0x00	0x00	0x00	0x00	0x00
4	0x0d	0x04	0x00	0x00	0x00	0x00	0x00	0x00
5	0x18	0x05	0x00	0x00	0x00	0x00	0x00	0x00
6	0x9b	0x06	0x00	0x00	0x00	0x00	0x00	0x00
7	0x65	0x07	0x00	0x00	0x00	0x00	0x00	0x00
8	0x08	0x08	0x00	0x00	0x00	0x00	0x00	0x00
9	0x1d	0x09	0x00	0x00	0x00	0x00	0x00	0x00
10	0x9e	0x0a	0x00	0x00	0x00	0x00	0x00	0x00
11	0x37	0x0b	0x00	0x00	0x00	0x00	0x00	0x00
12	0x0b	0x0c	0x00	0x00	0x00	0x00	0x00	0x00
13	0x1e	0x0d	0x00	0x00	0x00	0x00	0x00	0x00
14	0x9d	0x0e	0x00	0x00	0x00	0x00	0x00	0x00
15	0xcd	0x0f	0x00	0x00	0x00	0x00	0x00	0x00
0	0x0e	0x00	0x00	0x00	0x00	0x00	0x00	0x00
	CRC	*4 bit Data + 4 bit Counter*	*Data*					

Table 6.63: E2E Profile 22 example protect result

6.10.10.1 Offset set to 64

This is a typical use-case for using P22 with SOME/IP serializer, which puts an 8 byte header in front of the serialized user data. Result data of E2E_P22Protect() with data equals all zeros (0x00), counter = 1:

Byte	0	1	2	3	4	5	6	7
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	8	9	10	11	12	13	14	15
Data	0x14	0x01	0x00	0x00	0x00	0x00	0x00	0x00
Field	*CRC*	*DataID |Counter*	*Data*					

Table 6.64: E2E Profile 2 example protect result with short data and SOME/IP

6.11 Specification of E2E state machine

The E2E Profile check()-functions verifies data in one cycle. This function only determines if data in that cycle are correct or not. In contrary, the state machine builds up a state out of several results of check() function within a reception window, which is then provided to the consumer (RTE/SWC/COM).

The state machine is applicable for all E2E profiles. Profiles P01 and P02 can be configured to work together with the state machine. However, the behavior of P01/P02 alone, regardless how it is configured, is different to the behavior of P01/P02 + state machine.

6.11.1 Overview of the state machine

The diagram below summarizes the state machine.

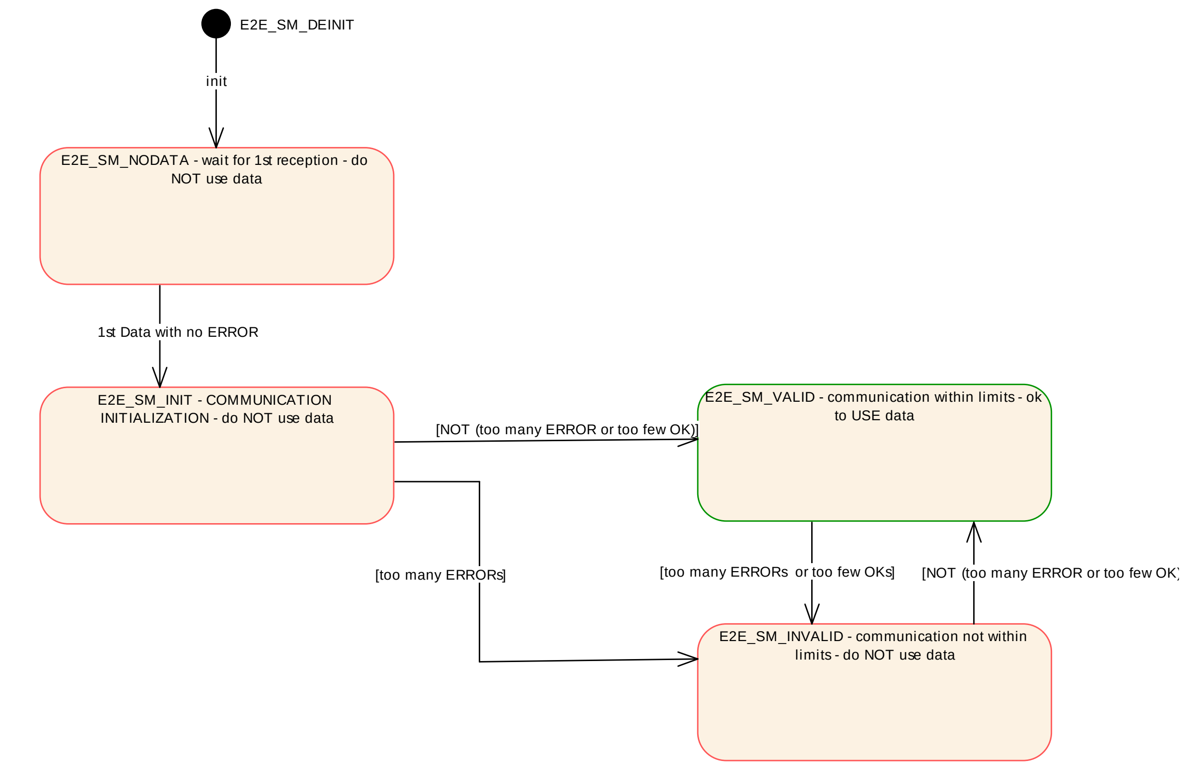

Figure 6.142: E2E state machine overview

6.11.2 State machine specification

[PRS_E2E_00354] ⌈The E2E state machine shall be implemented by the functions E2E_SMCheck() and E2E_SMCheckInit()⌋*(RS_E2E_08539)*

[PRS_E2E_00345] ⌈The E2E State machine shall have the behavior with respect to the function E2E_SMCheck() as shown in Figure 6.143.

This shall be understood as follows:

1. The current state (e.g. E2E_SM_VALID) is stored in State->SMState
2. At every invocation of E2E_SMCheck, the ProfileStatus is processed (as shown by logical step E2E_SMAddStatus()
3. After that, there is an examination of two counters: State->ErrorCount and State->OKCount. Depending on their values, there is a transition to a new state, stored in State->SMState.

⌋*(RS_E2E_08539)*

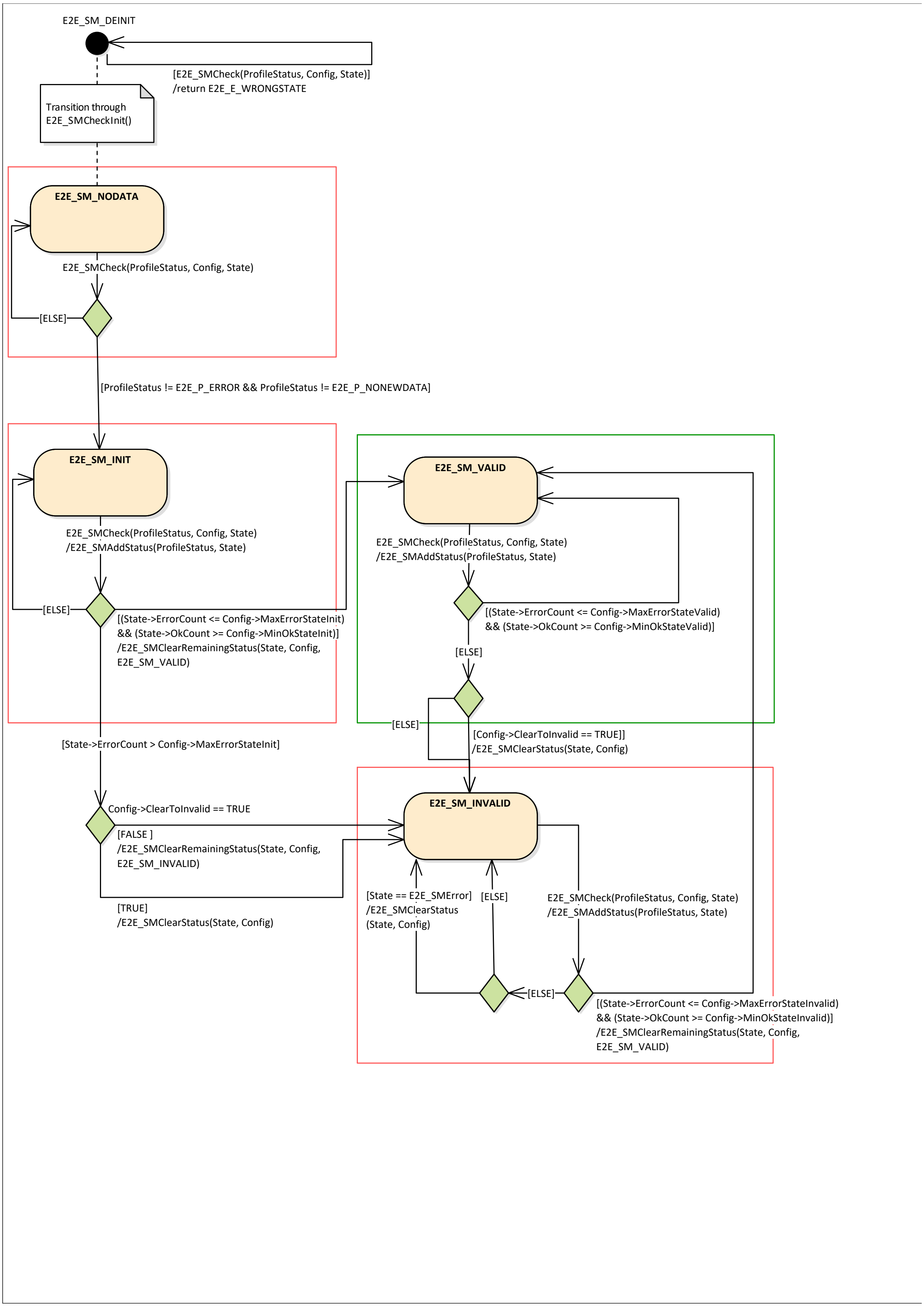

Figure 6.143: E2E state machine check

[PRS_E2E_00466] ⌈The step E2E_SMAddStatus(ProfileStatus, State) in E2E_SMCheck() shall behave as shown in Figure 6.144.⌋*(RS_E2E_08539)*

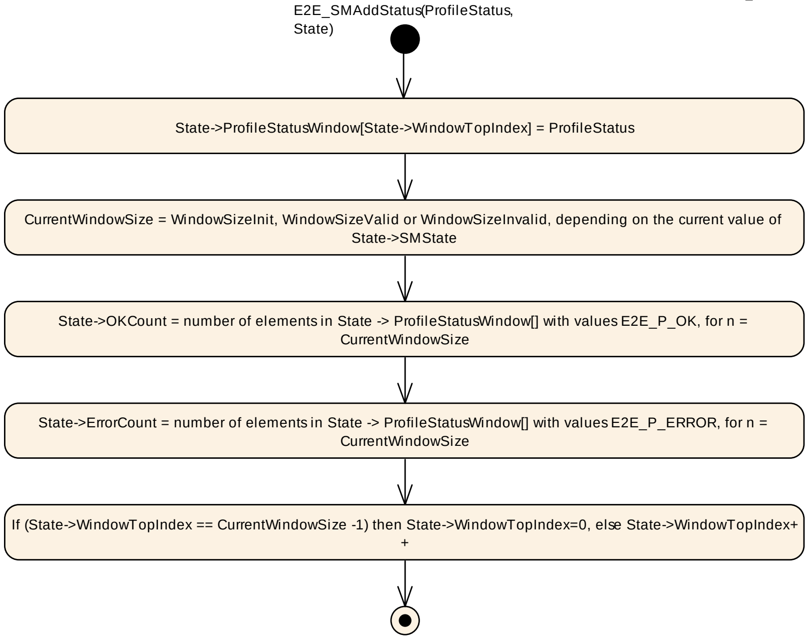

Figure 6.144: E2E state machine step E2E_SMAddStatus

E2E_SMAddStatus is just a logical step in the algorithm, it may (but it does not have to be) implemented a a separate function. It is not a module API function.

[PRS_E2E_00375] ⌈The E2E State machine shall have the behavior with respect to the function E2E_SMCheckInit() as shown in Figure 6.145.⌋*(RS_E2E_08539)*

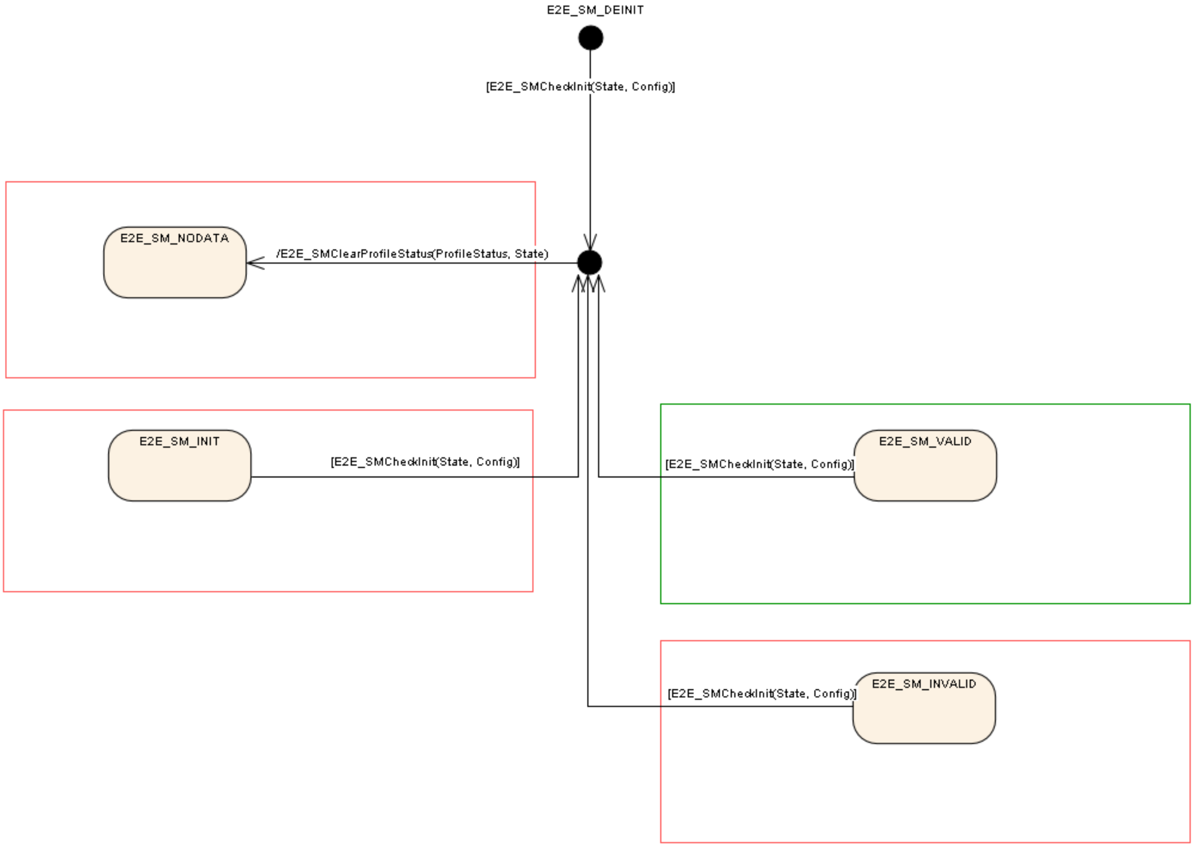

Figure 6.145: E2E state machine step E2E_SMCheckInit

[PRS_E2E_00467] ⌈The step E2E_SMClearStatus(State, Config) in E2E_SMCheck() shall behave as shown in Figure 6.146.⌋*(RS_E2E_08539)*

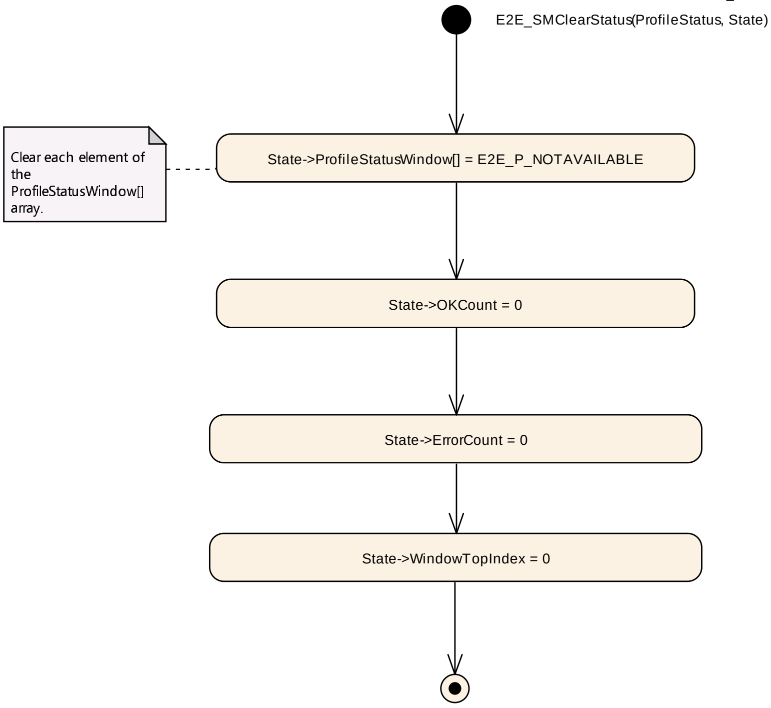

Figure 6.146: E2E state machine step E2E_SMCheck

[PRS_E2E_00607] ⌈The step E2E_SMClearRemainingStatus(Config, State) in E2E_SMCheck() shall have the following behavior: Figure 6.147.⌋*(RS_E2E_08539)*

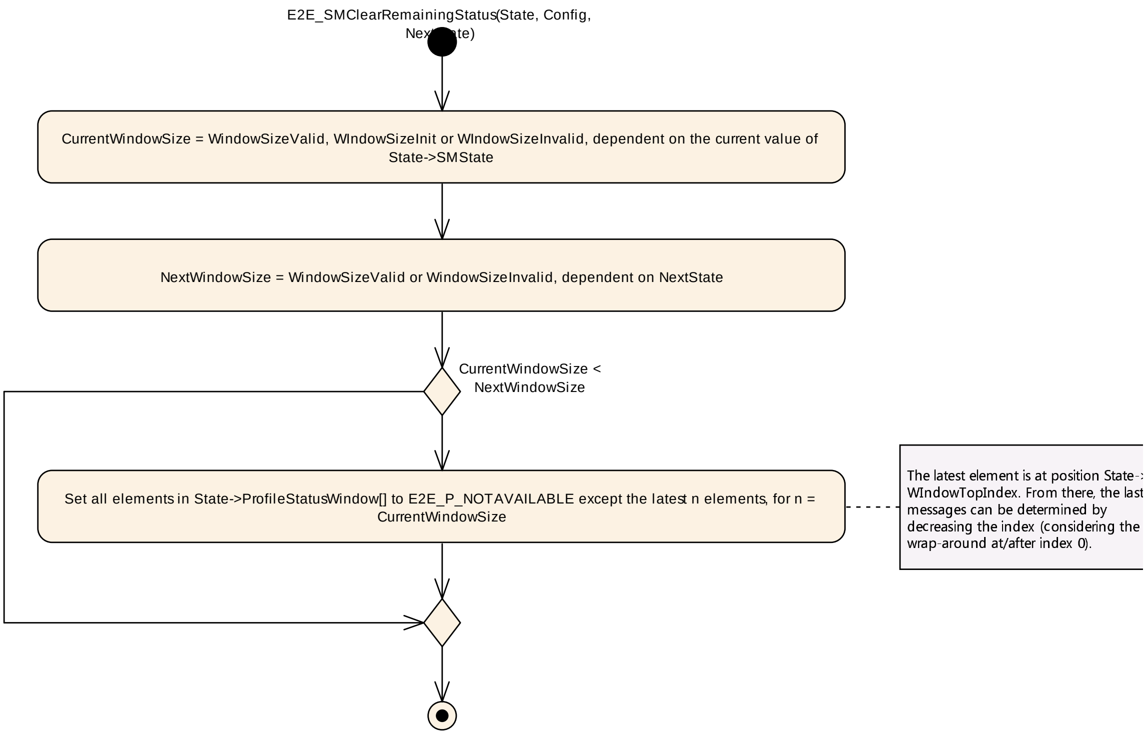

Figure 6.147: E2E state machine step E2E_SMClearRemainingStatus

6.11.2.1 E2E State Machine Status Enumeration

[PRS_E2E_00596] ⌈The E2E State Machine uses the following enumeration values to indicate its current status (see Table 6.65).⌋*(RS_E2E_08528)*

Name	Description
E2E_SM_VALID	Communication functioning properly according to E2E, data can be used.
E2E_SM_DEINIT	State before E2E_SMCheckInit() is invoked, data cannot be used.
E2E_SM_NODATA	No data from the sender is available since the initialization, data cannot be used.
E2E_SM_INIT	There has been some data received since startup, but it is not yet possible use it, data cannot be used.
E2E_SM_INVALID	Communication not functioning properly, data cannot be used.

Table 6.65: E2E State Machine Check Status Enumeration

6.11.2.2 Profile specific Check Status to State Machine Check Status mappings

This section targets the single mappings between each Profile specific check state to the check states used by the E2E Statemachine

[PRS_E2E_00597] ⌈The E2E State Machine uses the following enumeration values as input from the Profile specific check functions (see Table 6.66).⌋*(RS_E2E_08528)*

Name	Description
E2E_P_OK	Check of the message was successful and no error was found
E2E_P_ERROR	An error was detected in the received message.
E2E_P_REPEATED	A repeated messages was received
E2E_P_NONEWDATA	No new message was received
E2E_P_WRONGSEQUENCE	The received message contains wrong counter

Table 6.66: E2E State Machine Check Status Enumeration

[PRS_E2E_00598] Mapping Profile 1 to State-Machine ⌈The mapping between Profile 1 specific check states to the input for the E2E-State Machine is described in table Table 6.67).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P01STATUS_OK, E2E_P01STATUS_OKSOMELOST, E2E_P01STATUS_SYNC	E2E_P_OK
E2E_P01STATUS_WRONGCRC	E2E_P_ERROR
E2E_P01STATUS_REPEATED	E2E_P_REPEATED
E2E_P01STATUS_NONEWDATA	E2E_P_NONEWDATA

E2E_P01STATUS_WRONGSEQUENCE E2E_P01STATUS_INITIAL	E2E_P_WRONGSEQUENCE

Table 6.67: E2E Profile 1 specific Check Status Mapping

[PRS_E2E_00599] Mapping Profile 2 to State-Machine ⌈The mapping between Profile 2 specific check states to the input for the E2E-State Machine is described in table Table 6.68).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P02STATUS_OK, E2E_P02STATUS_OKSOMELOST, E2E_P02STATUS_INITIAL	E2E_P_OK
E2E_P02STATUS_WRONGCRC	E2E_P_ERROR
E2E_P02STATUS_REPEATED	E2E_P_REPEATED
E2E_P02STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P02STATUS_WRONGSEQUENCE E2E_P02STATUS_SYNC	E2E_P_WRONGSEQUENCE

Table 6.68: E2E Profile 2 specific Check Status Mapping

[PRS_E2E_00600] Mapping Profile 4 to State-Machine ⌈The mapping between Profile 4 specific check states to the input for the E2E-State Machine is described in table Table 6.69).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P04STATUS_OK, E2E_P04STATUS_OKSOMELOST	E2E_P_OK
E2E_P04STATUS_ERROR	E2E_P_ERROR
E2E_P04STATUS_REPEATED	E2E_P_REPEATED
E2E_P04STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P04STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.69: E2E Profile 4 specific Check Status Mapping

[PRS_E2E_00601] Mapping Profile 5 to State-Machine ⌈The mapping between Profile 5 specific check states to the input for the E2E-State Machine is described in table Table 6.70).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P05STATUS_OK, E2E_P05STATUS_OKSOMELOST	E2E_P_OK
E2E_P05STATUS_ERROR	E2E_P_ERROR
E2E_P05STATUS_REPEATED	E2E_P_REPEATED
E2E_P05STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P05STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.70: E2E Profile 5 specific Check Status Mapping

[PRS_E2E_00602] Mapping Profile 6 to State-Machine ⌈The mapping between Profile 6 specific check states to the input for the E2E-State Machine is described in table Table 6.71).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P06STATUS_OK, E2E_P06STATUS_OKSOMELOST	E2E_P_OK
E2E_P06STATUS_ERROR	E2E_P_ERROR
E2E_P06STATUS_REPEATED	E2E_P_REPEATED
E2E_P06STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P06STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.71: E2E Profile 6 specific Check Status Mapping

[PRS_E2E_00603] Mapping Profile 7 to State-Machine ⌈The mapping between Profile 7 specific check states to the input for the E2E-State Machine is described in table Table 6.72).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P07STATUS_OK, E2E_P07STATUS_OKSOMELOST	E2E_P_OK
E2E_P07STATUS_ERROR	E2E_P_ERROR
E2E_P07STATUS_REPEATED	E2E_P_REPEATED
E2E_P07STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P07STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.72: E2E Profile 7 specific Check Status Mapping

[PRS_E2E_00604] Mapping Profile 11 to State-Machine ⌈The mapping between Profile 11 specific check states to the input for the E2E-State Machine is described in table Table 6.73).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P11STATUS_OK, E2E_P11STATUS_OKSOMELOST	E2E_P_OK
E2E_P11STATUS_ERROR	E2E_P_ERROR
E2E_P11STATUS_REPEATED	E2E_P_REPEATED
E2E_P11STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P11STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.73: E2E Profile 11 specific Check Status Mapping

[PRS_E2E_00605] Mapping Profile 22 to State-Machine ⌈The mapping between Profile 22 specific check states to the input for the E2E-State Machine is described in table Table 6.74).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P22STATUS_OK, E2E_P11STATUS_OKSOMELOST	E2E_P_OK
E2E_P22STATUS_ERROR	E2E_P_ERROR
E2E_P22STATUS_REPEATED	E2E_P_REPEATED
E2E_P22STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P22STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.74: E2E Profile 22 specific Check Status Mapping

7 E2E API specification

This chapter defines an abstract API of E2E supervision. E2E is supposed to be invoked by middleware, but the results of checks are visible to the application, so this chapter is split into two parts.

7.1 API of middleware to applications

The API to the applications is imposed by the middleware (e.g. RTE or ARA). E2E provides an additional output object providing E2E check results.

[PRS_E2E_USE_00321] ⌈The middleware shall provide, for each exchanged dataRecord, a set of functions:

- middleware_send(in dataRecord)
- middleware_receive(out dataRecord, out e2eResult)

⌋*(RS_E2E_08534)*

[PRS_E2E_00322] ⌈The e2eResult shall contain pieces of information:

- e2eStatus: Profile-independent status of the reception on one single Data in one cycle. Possible values are: OK, REPEATED, WRONGSEQUENCE, NOTAVAILABLE, NONEWDATA.
- e2eState: Status of the communication channel exchanging the data. Possible values are: VALID, DEINIT, NODATA, INIT, INVALID.

⌋*(RS_E2E_08534)*

7.2 API of E2E

The E2E interface is independent from any middleware. It is designed with SOME/IP in mind, but it could work for any other middleware or software services, e.g. a database requesting to protect its data.

The interface between the middleware and E2E operates on the serialized data, where: E2E adds E2E header (sender side) and E2E check E2E header (receiver side).

[PRS_E2E_00323] ⌈E2E protocol shall provide the following interface:

- E2E_check(in dataID, inout serializedData)
- E2E_protect(in dataID, inout serializedData): e2eResult

where:

- dataID is a unique identifier of the exchanged data/information. In case of multiple instantiation, each single instance gets typically a separate dataID, even if the same type of information is transmitted
- serializedData - vector/array of serialized data, where E2E header is located, next to serialized data
- e2eResult - result of E2E checks, see previous section for the definition.

⌋*(RS_E2E_08534)*

The middleware is responsible to provide an adaptation to E2E functional interface.

[PRS_E2E_00318] ⌈The middleware shall determine the DataID of the currently exchanged information.⌋*(RS_E2E_08534)*

For example, in case of vsomeip, it needs to determine DataID based on serviceid/eventid/instanceid tuple.

[PRS_E2E_00319] ⌈The middleware invoke E2E functions providing them the DataID together with the data.⌋*(RS_E2E_08534)*

[PRS_E2E_00320] ⌈On the receiver side, the middleware shall provide the e2eResult determined by E2E to the receiver.⌋*(RS_E2E_08534)*

8 Configuration Parameters

E2E supervision has the following configuration options for each protected data. Note that it is platform-specific how middleware associates a middleware communication channel with E2E communication protection.

For each DataID, which uniquely represents data exchanged, there is a set of configuration options.

[PRS_E2E_00324] ⌈The options for a E2E-protected data shall be available as defined in Table 8.1⌋*(RS_E2E_08534)*

Parameters	Profile	Description
dataID	1, 4, 5, 6, 7, 11	This represents a unique numerical identifier. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd-Protection. dataId is used as a unique identifier of a configuration object. One dataId can appear only once in the configuration.
profileName	all	This represents the identification of the concrete E2E profile. Possible profiles: 1 (only CP), 2 (only CP), 4, 5, 6, 7, 11, 22.
dataLength	1, 2, 5, 11, 22	For fixed size data: length of data in bits.
minDataLength	4, 6, 7	For variable size data: minimum length of data in bits.
maxDataLength	4, 6, 7	For variable size data: maximum length of data in bits.
dataIdList	2, 22	List of 16 dataID values, where a a different value is transmitted depending on counter value.
dataIdMode	1, 11	This attribute describes the inclusion mode that is used to include the two-byte Data ID in E2E communication protection.
offset	2, 4, 5, 6, 7, 22	Offset of the E2E header in the Data[] array in bits.
counterOffset	1, 11	Offset of the counter in the Data[] array in bits. Fixed for AP to 0.
crcOffset	1, 11	Offset of the CRC in the Data[] array in bits. Fixed for AP to 8.
dataIdNibbleOffset	1, 11	Offset of the dataID nibble in the Data[] array in bits. Fixed for AP to 12.
maxDeltaCounter	4, 5, 6, 7, 11, 22	Maximum allowed difference between the counter value of the current message and the previous valid message.
Parameters of legacy profiles (Only CP)		
maxDeltaCounterInit	1, 2	Initial maximum allowed gap between two counter values of two consecutively received valid Data. The maxDeltaCounter is increased on each reception try but only reset when receiving a valid message. This is to compensate for and tolerate lost messages.
maxNoNewOrRepeated-Data	1, 2	The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions.
syncCounterInit	1, 2	The number of messages required for validating the consistency of the counter after exceeding the maxNoNewOrRepeatedData threshold.
profileBehavior	1, 2	Mapping of specific profile status values to unified profileStatus. False: legacy behavior, as before AUTOSAR Classic Platform Release 4.2,True: mapping according to new profiles (profile 4 and newer) interpretation of status, introduced in AUTOSAR Classic Platform Release 4.2.
Parameters of E2E State Machine		

▽

△

windowSizeValid	Size of the monitoring window of state Valid for the E2E state machine.
windowSizeInvalid	Size of the monitoring window of state Invalid for the E2E state machine.
windowSizeInit	Size of the monitoring window of state Init for the E2E state machine.
clearFromValidToInvalid	Clear monitoring window on transition from state Valid to state Invalid.
maxErrorStateInit	Maximum number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSizeInit checks, for the state E2E_SM_INIT.
maxErrorStateInvalid	Maximum number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSizeInvalid checks, for the state E2E_SM_INVALID.
maxErrorStateValid	Maximum number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSizeValid checks, for the state E2E_SM_VALID.
minOkStateInit	Minimum number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSizeInit checks, for the state E2E_SM_INIT.
minOkStateInvalid	Minimum number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSizeInvalid checks, for the state E2E_SM_INVALID.
minOkStateValid	Minimum number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSizeValid checks, for the state E2E_SM_VALID.

Table 8.1: E2E configuration parameters

8.1 General Constraints

This section contains general platform independent constraints. These belong to the configuration parameters mentioned in Table 8.1.

8.1.1 E2E-Statemachine Settings

[constr_3176] Value range of `windowSizeValid` ⌈The value of the `windowSizeValid` attribute shall be greater or equal to 1.⌋*()*

[constr_6301] Dependency between `windowSizeInvalid` and `windowSizeValid` ⌈The following restriction shall be respected: `WindowSizeInvalid` <= `WindowSizeValid`⌋*()*

[constr_6302] Dependency between `windowSizeInit` and `windowSizeValid` ⌈The following restriction shall be respected: `windowSizeInit` <= `WindowSizeValid`⌋*()*

[constr_3177] Dependency between `maxErrorStateValid`, `maxErrorStateInit` and `maxErrorStateInvalid` ⌈The following restriction shall be respected:
`maxErrorStateValid` >= `maxErrorStateInit` >= `maxErrorStateInvalid` >= 0⌋*()*

[constr_3178] Dependency between `minOkStateValid`, `minOkStateInit` and `minOkStateInvalid` ⌈The following restriction shall be respected:
1 <= `minOkStateValid` <= `minOkStateInit` <= `minOkStateInvalid`⌋*()*

[constr_3179] Dependency between `minOkStateInit`, `maxErrorStateInit` and `windowSizeValid` ⌈The following restriction shall be respected:
`minOkStateInit` + `maxErrorStateInit` <= `windowSizeValid`⌋*()*

[constr_3180] Dependency between `minOkStateValid`, `maxErrorStateValid` and `windowSizeValid` ⌈The following restriction shall be respected:
`minOkStateValid` + `maxErrorStateValid` <= `windowSizeValid`⌋*()*

[constr_3181] Dependency between `minOkStateInvalid`, `maxErrorStateInvalid` and `windowSizeValid` ⌈The following restriction shall be respected: `minOkStateInvalid` + `maxErrorStateInvalid` <= `windowSizeValid`⌋*()*

9 Protocol usage and guidelines

This chapter contains requirements on usage of E2E Supervision when designing and implementing safety-related systems, which are depending on E2E communication protection and which are not directly related to some specific functionality. Note that chapter 6 also provides several requirements on usage.

9.1 E2E and SOME/IP

For the combination of E2E communication protection with SOME/IP, there needs to be a common definition of the on-wire protocol. Depending on architecture properties, the implementing components need to be configured and used accordingly.

In general, all available E2E profiles can be used in combination with SOME/IP. However, they may have limitations, as for the maximum usable length of data, or being limited to fixed length messages.

The size of the E2E Header is dependent on the selected E2E profile.

[PRS_E2E_USE_00236] ⌈The E2E CRC shall be calculated over the following parts of the serialized SOME/IP message.

1. Request ID (Client ID / Session ID) [32 bit]
2. Protocol Version [8 bit]
3. Interface Version [8 bit]
4. Message Type [8 bit]
5. Return Code [8 bit]
6. Payload [variable size]

⌋*(RS_E2E_08540)*

[PRS_E2E_USE_00237] ⌈The E2E header shall be placed after the Return Code depending on the chosen Offset value. The default Offset is 64 bit, which puts the E2E header exactly after the Return Code.⌋*(RS_E2E_08540)*

9.2 Client-Server Communication

[PRS_E2E_USE_00606] ⌈When a client sends a request to the server, the server shall use the received counter as sequence counter for the response, no matter if regular response or error response.⌋*(RS_E2E_08541)*

[constr_6300] MaxDeltaCounter for Client-Server Communication ⌈For Client-Server Communication the MaxDeltaCounter on server-side shall be set to the maximum of the value range of the sequence counter⌋*()*

9.3 Periodic use of E2E check

[PRS_E2E_USE_00325] ⌈The E2E check function shall be invoked at least once within FTTI (FTTI is for the safety goals from which the requirements for this E2E checks are derived).⌋*(RS_E2E_08528)*

9.4 Error handling

The E2E Supervision itself does not handle detected communication errors. It only detects such errors for single received data elements and returns this information to the callers (e.g. SW-Cs), which have to react appropriately. A general standardization of the error handing of an application is usually not possible.

[PRS_E2E_USE_00235] ⌈The user (caller) of E2E Supervision, in particular the receiver, shall provide the error handling mechanisms for the faults detected by the E2E Supervision.⌋*(RS_E2E_08528)*

9.5 Maximal lengths of Data, communication buses

The length of the message and the achieved hamming distance for a given CRC are related. To ensure the required diagnostic coverage the maximum length of data elements protected by a CRC needs to be selected appropriately. The E2E profiles are intended to protect inter-ECU communication with lengths as listed in Table 9.1

All length values stated in this section are based on assumptions on suitable hamming distances for specific scenarios, without explicitly listing those assumptions. As such, actual suitable values may differ based on the use case scenarios.

E2E Profile	Suggested maximum applicable length including control fields for inter-ECU communication
E2E Profile 1 and 11	32B
E2E Profile 2 and 22	32B
E2E Profile 4	4 kB
E2E Profile 5	4 kB
E2E Profile 6	4 kB
E2E Profile 7	4 MB

Table 9.1: E2E maximum data length

In E2E Profiles 1 and 2, the Hamming Distance is 2, up to the given lengths. Due to 8 bit CRC, the burst error detection is up to 8 bits.

[PRS_E2E_UC_00051] ⌈In case of inter-ECU communication over FlexRay, the length of the complete Data (including application data, CRC and counter) protected by E2E Profile 1 or E2E Profile 2 should not exceed 32 bytes.⌋*(RS_E2E_08528)*

This requirement only contains a reasonable maximum length evaluated during the design of the E2E profiles. The responsibility to ensure the adequacy of the implemented E2E communication protection using E2E Supervision for a particular system remains by the user.

[PRS_E2E_UC_00466] ⌈In case of inter-ECU communication over FlexRay, CAN, CAN FD, Ethernet suggested max. data length can be adopted (extended or reduced) if it can by justified by an analysis of a particular use case or network architecture.⌋*(RS_E2E_08528)*

[PRS_E2E_UC_00061] ⌈In case of CAN or LIN the length of the complete data element (including application data, CRC and counter) protected by E2E Profile 1 should not exceed 8 bytes.⌋*(RS_E2E_08528)*

[PRS_E2E_UC_00351] ⌈The length of the complete Data (including application data and E2E header) protected by E2E Profile 4, 5 or 6 shall not exceed 4kB.⌋*(RS_E2E_08528)*

[PRS_E2E_UC_00316] ⌈The length of the complete Data (including application data and E2E header) protected by E2E Profile 7 shall not exceed 4MB.⌋*(RS_E2E_08528)*

[PRS_E2E_UC_00236] ⌈When using E2E Supervision, the designer of the functional or technical safety concept of a particular system using E2E Supervision shall evaluate the maximum permitted length of the protected Data in that system, to ensure an appropriate error detection capability.⌋*(RS_E2E_08539)*

Thus, the specific maximum lengths for a particular system may be shorter (or maybe in some rare cases even longer) than the recommended maximum applicable lengths defined for the E2E Profiles.

If the protected data length exceeds the network bus frame limit (or payload limit), the data can be segmented on the sender side after the E2E communication protection, and be assembled on the receiver side before the E2E evaluation. The possible faults happening during segmentation/desegmentation can be considered as "corruption of information".

[PRS_E2E_UC_00170] ⌈When designing the functional or technical safety concept of a particular system any user of E2E shall ensure that the transmission of one undetected erroneous data element in a sequence of data elements between sender and receiver, protected with profile 1, 11, 2, 22, will not directly lead to the violation of a safety goal of this system.⌋*()*

In other words, SW-C shall be able to tolerate the reception of one erroneous data element, which error was not detected by the E2E Supervision. What is not required is that an SW-C tolerates two consecutive undetected erroneous data elements, because it is enough unlikely that two consecutive Data are wrong AND that for both Data the error remains undetected by the E2E Supervision.

When using LIN as the underlying communication network the residual error rate on protocol level is several orders of magnitude higher (compared to FlexRay and CAN) for the same bit error rate on the bus. The LIN checksum compared to the protocol CRC of FlexRay (CRC-24) and CAN (CRC-15) has different properties (e.g. hamming distance) resulting in a higher number of undetected errors coming from the bus (e.g. due to EMV). In order to achieve a maximum allowed residual error rate on application level, different error detection capabilities of the application CRC may be necessary, depending on the strength of the protection on the bus protocol level.

[PRS_E2E_UC_00237] ⌈Any user of E2E Supervision shall ensure, that within one implementation of a communication network every safety-related Data, protected by E2E Supervision, has a unique Data ID (E2E Profiles 1, 4, 5, 6, 7, 11) or a unique DataIDList[] (E2E Profiles 2, 22).⌋*(RS_E2E_08528)*

E2E Profile 1 with E2E_P01DataIDMode = E2E_P01_DATAID_BOTH and E2E Profile 11 with E2E_P11DataIDMode = E2E_P11_DATAID_BOTH uses an implicit 2-byte Data ID, over which CRC8 is calculated. As a CRC over two different 2-byte numbers may result with the same CRC, some precautions must be taken by the user. See SWS_E2ELibrary items UC_E2E_00072 and UC_E2E_00073 [3].

A Constraint History

A.1 Constraint History R19-11

A.1.1 Added Constraints

Number	Heading
[constr_3176]	Value range of `windowSizeValid`
[constr_3177]	Dependency between `maxErrorStateValid`, `maxErrorStateInit` and `maxErrorStateInvalid`
[constr_3178]	Dependency between `minOkStateValid`, `minOkStateInit` and `minOkStateInvalid`
[constr_3179]	Dependency between `minOkStateInit`, `maxErrorStateInit` and `windowSizeValid`
[constr_3180]	Dependency between `minOkStateValid`, `maxErrorStateValid` and `windowSizeValid`
[constr_3181]	Dependency between `minOkStateInvalid`, `maxErrorStateInvalid` and `windowSizeValid`
[constr_6300]	MaxDeltaCounter for Client-Server Communication
[constr_6301]	Dependency between `windowSizeInvalid` and `windowSizeValid`
[constr_6302]	Dependency between `windowSizeInit` and `windowSizeValid`

Number	Heading

Table A.1: changed Constraints in R19-11

A.1.2 Changed Constraints

N/A

A.1.3 Deleted Constraints

N/A

A.1.4 Added Specification Items

N/A

A.1.5 Changed Specification Items

N/A

A.1.6 Deleted Specification Items

N/A