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	R22-11

Document Change History			
Date	Release	Changed by	Description
2022-11-24	R22-11	AUTOSAR Release Management	• New chapter 6.3 added with generalized flowcharts and SWS items for non method profiles. • Protocol specific flowcharts and SWS items for method (P04m and P07m) and non method profiles replaced by generalized flowcharts and items • Use consistent function names (e.g. E2EXf_handling_PXXm_server, E2EXf_handling_PXXm_client) • Corrections of Min/MaxDataLength in P04m, P07m and P08m • Corrections of errors in state machine specification • Counter handling for client/server communication updated
2021-11-25	R21-11	AUTOSAR Release Management	• New profiles 08m,44m • New protocol independent flowcharts
2020-11-30	R20-11	AUTOSAR Release Management	• E2E for methods. • New profiles 08,44,4m,7m • Extension of E2E State Machine

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 State Machine. • 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

Contents

1 Introduction and functional overview

The concept of E2E communication protection assumes that safety-related [1] 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).

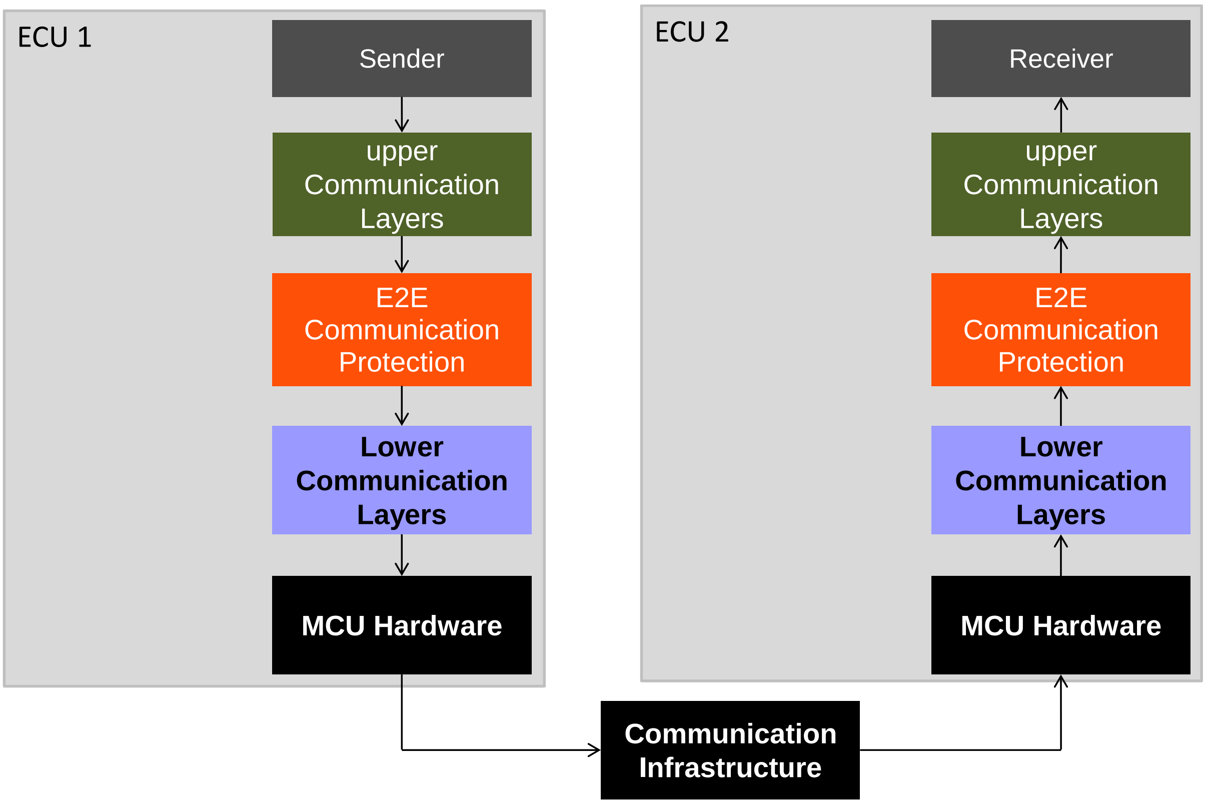

Figure 1.1: Overview of E2E communication protection between a sender and a receiver

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::com, 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 0xF4ACFB13 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 [2, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
Data ID	An identifier that uniquely identifies the message / data element / data.
Source ID	An identifier that uniquely identifies the source of the message / data element / data (i.e., in case different sources produce the same message / data element / data – like different clients invoking the same method at a server).
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.
FTTI	Fault tolerant time interval, maximum time a fault can be active before a hazard occurs.

Table 2.1: Acronyms and Abbreviations

3 Related documentation

3.1 Input documents

[1] ISO 26262:2018 (all parts) – Road vehicles – Functional Safety
http://www.iso.org

[2] Glossary
AUTOSAR_TR_Glossary

[3] Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary

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

3.2 Standards and Norms

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 depending on the used type of communication. From E2E perspective the following types are distinguished:

- Signal based communication
- Service oriented communication with events
- Service oriented communication in Client/Server architecture
- Signal to Service Translation

In general, the behavior of the E2E protection mechanisms should be the same. However, some limitations exist depending on the communication type.

4.1.1 Limitations in general

- Communication errors which are detected at **lower layer** (e.g. by Ethernet FCS, IP header checksum, UDP checksum, SOME/IP header irregularities) will lead to **discarding the related message**. Thus, these messages might not reach the application at all.

4.1.2 Limitations in signal based communication

It is also called sender/receiver communication.

- E2E communication protection is limited to periodic or semi-periodic data communication paradigm, where the receiver has any expectancy on the regular reception of data and in case of communication loss/timeout or error, it performs an error handling.
- If one of E2E functions to protect (sender) or to check (receiver) is not called periodically, then some communication failure modes (loss, delay) may not be detected.
- If E2E protection is invoked at the level of data elements (e.g. from SW-Cs or from E2E Protection Wrapper) and 1:N communication model is used and the data elements are sent using more than one IPDU, then all these I-PDUs shall have the same layout.
- Currently AUTOSAR does not provide the functionality to describe and handle more than one layout for the same data element (e.g. within the RTE) by using different protection mechanisms depending on Intra-ECU and Inter-ECU communication. Thus, for a 1:N sender-receiver relationship the user of E2E-protection

is responsible to select one appropriate layout for the to be protected data elements. E.g. for a 1:N sender-receiver relationship the message layout can be used for the transmission of data elements protected by protection to receivers located within and without the ECU.

- If a given sender-receiver communication is only intra-ECU (within microcontroller), then it is not defined within the configuration what the layout of the serialized Data shall be. On the other side, as the communication is intra-ECU, on both sides the software is probably generated by the same RTE generator, so the decision on the layout can be specific to the generator. It is recommended to serialize the data to have the CRC at the profile-specific position of the CRC and the Counter at the profile-specific position of the Counter (like for inter-ECU communication).
- Non-blocking characteristics of queued sender-receiver communication only is considered (blocking characteristic for queued communication is not supported).

4.1.3 Limitations in service oriented communication with events

It is also called event communication. (Note that here the name event is a bit confusing as a periodic communication is required. This is taken from the SOME/IP event.)

- E2E communication protection is limited to periodic or semi-periodic data communication paradigm, where the receiver has any expectancy on the regular reception of data and in case of communication loss/timeout or error, it performs an error handling.
- If one of the E2E functions to protect (sender) or to check (receiver) data is not called periodically, then some communication failure modes (loss, delay) may not be detected.

4.1.4 Limitations in service oriented communication in Client/Server architecture

The specified E2E communication protection for methods is limited to

- Communication between N clients and one dedicated server (N:1); this means, an E2E protected service is provided by exactly one server per system. In other words, if E2E protection is invoked at the level of data elements, then N:1 multiplicity, implicit communication, and remaining communication models (in particular client-server model) are not supported.

The specified E2E communication protection for methods may not detect all communication failure modes:

- If method requests are not invoked periodically, then some communication failure modes (loss, delay) may not be detected for received requests at the server.

- As the defined E2E protection mechanisms assume no a priori knowledge at the server about the client that is requesting data (N:1 communication), communication failure modes (repetition, insertion) which are client specific may not be detected for received requests at the server. In this case, additional measures need to be implemented at application level to address those non-detected failure modes and complete E2E protection or arguments are to be provided showing that these failure modes are not relevant for a particular project.

The values of the following E2E parameters are defined by the standard and shall not be changed.

- dataIdMode
- counterOffset
- crcOffset
- dataIdNibbleOffset
- offset

E2E Profiles 1, 2, 11 and 22 are not suggested to be used in Client-Server Communication.

4.1.5 Signal to Service Translation

Signal to service translation is limited to translation between sender/receiver and service oriented communication with events.

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 microcontroller, between CPU cores or microcontrollers).

5 Requirements Tracing

Requirement	Description	Satisfied by
[RS_E2E_08527]	Implementation of E2E protocol shall fulfill ISO 26262	[PRS_E2E_00219] [PRS_E2E_00372] [PRS_E2E_00394] [PRS_E2E_00479] [PRS_E2E_00480] [PRS_E2E_00503] [PRS_E2E_00522] [PRS_E2E_UC_00317]
[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]

Requirement	Description	Satisfied by
		[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_00298] [PRS_E2E_00299] [PRS_E2E_00300] [PRS_E2E_00301] [PRS_E2E_00306] [PRS_E2E_00400] [PRS_E2E_00420] [PRS_E2E_00508] [PRS_E2E_00526] [PRS_E2E_00540] [PRS_E2E_00541] [PRS_E2E_00588] [PRS_E2E_00589] [PRS_E2E_00591] [PRS_E2E_00592] [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]

Requirement	Description	Satisfied by
		[PRS_E2E_00611] [PRS_E2E_00612] [PRS_E2E_00613] [PRS_E2E_00614] [PRS_E2E_00641] [PRS_E2E_00644] [PRS_E2E_00645] [PRS_E2E_00646] [PRS_E2E_00647] [PRS_E2E_00648] [PRS_E2E_00651] [PRS_E2E_00652] [PRS_E2E_00653] [PRS_E2E_00654] [PRS_E2E_00655] [PRS_E2E_00656] [PRS_E2E_00657] [PRS_E2E_00660] [PRS_E2E_00661] [PRS_E2E_00662] [PRS_E2E_00663] [PRS_E2E_00664] [PRS_E2E_00665] [PRS_E2E_00666] [PRS_E2E_00667] [PRS_E2E_00668] [PRS_E2E_00669] [PRS_E2E_00670] [PRS_E2E_00673] [PRS_E2E_00677] [PRS_E2E_00706] [PRS_E2E_00735] [PRS_E2E_00738] [PRS_E2E_00739] [PRS_E2E_00743] [PRS_E2E_00826] [PRS_E2E_00827] [PRS_E2E_00850] [PRS_E2E_00851] [PRS_E2E_00852] [PRS_E2E_01154] [PRS_E2E_01159] [PRS_E2E_01160] [PRS_E2E_01161] [PRS_E2E_01162] [PRS_E2E_01163] [PRS_E2E_01164] [PRS_E2E_01165]

Requirement	Description	Satisfied by
		[PRS_E2E_01166] [PRS_E2E_01199] [PRS_E2E_01200] [PRS_E2E_01201] [PRS_E2E_01202] [PRS_E2E_01203] [PRS_E2E_01206] [PRS_E2E_01250] [PRS_E2E_01251] [PRS_E2E_01252] [PRS_E2E_CONSTR_03176] [PRS_E2E_CONSTR_03177] [PRS_E2E_CONSTR_03178] [PRS_E2E_CONSTR_03179] [PRS_E2E_CONSTR_03180] [PRS_E2E_CONSTR_03181] [PRS_E2E_CONSTR_06300] [PRS_E2E_CONSTR_06301] [PRS_E2E_CONSTR_06302] [PRS_E2E_CONSTR_06303] [PRS_E2E_UC_00051] [PRS_E2E_UC_00055] [PRS_E2E_UC_00061] [PRS_E2E_UC_00062] [PRS_E2E_UC_00063] [PRS_E2E_UC_00071] [PRS_E2E_UC_00072] [PRS_E2E_UC_00073] [PRS_E2E_UC_00170] [PRS_E2E_UC_00171] [PRS_E2E_UC_00173] [PRS_E2E_UC_00235] [PRS_E2E_UC_00308] [PRS_E2E_UC_00316] [PRS_E2E_UC_00320] [PRS_E2E_UC_00325] [PRS_E2E_UC_00351] [PRS_E2E_UC_00466]
[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] [PRS_E2E_00707] [PRS_E2E_00736] [PRS_E2E_00740] [PRS_E2E_00783] [PRS_E2E_01107] [PRS_E2E_01155]

Requirement	Description	Satisfied by
[RS_E2E_08530]	Each E2E profile shall define a set of protection mechanisms and its behavior	[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] [PRS_E2E_00707] [PRS_E2E_00736] [PRS_E2E_00740] [PRS_E2E_00783] [PRS_E2E_01107] [PRS_E2E_01155]
[RS_E2E_08531]	E2E Library shall call the CRC routines of CRC library	[PRS_E2E_00082] [PRS_E2E_00125] [PRS_E2E_00126] [PRS_E2E_00134] [PRS_E2E_00401] [PRS_E2E_00421] [PRS_E2E_00527] [PRS_E2E_00613] [PRS_E2E_01207]
[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] [PRS_E2E_00707] [PRS_E2E_00736] [PRS_E2E_00740] [PRS_E2E_00783] [PRS_E2E_01107] [PRS_E2E_01155]
[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_00677] [PRS_E2E_00678] [PRS_E2E_00828] [PRS_E2E_UC_00321]

Requirement	Description	Satisfied by
[RS_E2E_08537]	SW-Cs shall tolerate a number of invalid/corrupted received data elements	[PRS_E2E_00646] [PRS_E2E_00648] [PRS_E2E_00651] [PRS_E2E_00654] [PRS_E2E_00657] [PRS_E2E_00660] [PRS_E2E_00663] [PRS_E2E_00666] [PRS_E2E_00706] [PRS_E2E_00735] [PRS_E2E_00851] [PRS_E2E_00852]
[RS_E2E_08539]	An E2E protection mechanism for inter-ECU communication of short to large data shall be provided	[PRS_E2E_00345] [PRS_E2E_00354] [PRS_E2E_00375] [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]

Requirement	Description	Satisfied by
		[PRS_E2E_00425] [PRS_E2E_00426] [PRS_E2E_00427] [PRS_E2E_00428] [PRS_E2E_00429] [PRS_E2E_00430] [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_00504] [PRS_E2E_00505] [PRS_E2E_00506] [PRS_E2E_00507] [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] [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]

Requirement	Description	Satisfied by
		[PRS_E2E_00582] [PRS_E2E_00583] [PRS_E2E_00607] [PRS_E2E_00619] [PRS_E2E_00620] [PRS_E2E_00621] [PRS_E2E_00622] [PRS_E2E_00623] [PRS_E2E_00624] [PRS_E2E_00625] [PRS_E2E_00630] [PRS_E2E_00631] [PRS_E2E_00632] [PRS_E2E_00633] [PRS_E2E_00634] [PRS_E2E_00635] [PRS_E2E_00636] [PRS_E2E_00637] [PRS_E2E_00638] [PRS_E2E_00639] [PRS_E2E_00640] [PRS_E2E_00654] [PRS_E2E_00675] [PRS_E2E_00676] [PRS_E2E_01156] [PRS_E2E_01157] [PRS_E2E_01159] [PRS_E2E_01161] [PRS_E2E_01162] [PRS_E2E_01163] [PRS_E2E_01164] [PRS_E2E_01165] [PRS_E2E_01166] [PRS_E2E_01167] [PRS_E2E_01169] [PRS_E2E_01170] [PRS_E2E_01171] [PRS_E2E_01172] [PRS_E2E_01173] [PRS_E2E_01174] [PRS_E2E_01175] [PRS_E2E_01176] [PRS_E2E_01177] [PRS_E2E_01178] [PRS_E2E_01179] [PRS_E2E_01180] [PRS_E2E_01181] [PRS_E2E_01182]

Requirement	Description	Satisfied by
		[PRS_E2E_01183] [PRS_E2E_01184] [PRS_E2E_01185] [PRS_E2E_01186] [PRS_E2E_01187] [PRS_E2E_01188] [PRS_E2E_01189] [PRS_E2E_01190] [PRS_E2E_01191] [PRS_E2E_01192] [PRS_E2E_01193] [PRS_E2E_01194] [PRS_E2E_01195] [PRS_E2E_01196] [PRS_E2E_01197] [PRS_E2E_01198] [PRS_E2E_01203] [PRS_E2E_01205] [PRS_E2E_01206] [PRS_E2E_01209] [PRS_E2E_01210] [PRS_E2E_01211] [PRS_E2E_01212] [PRS_E2E_01213] [PRS_E2E_01214] [PRS_E2E_01215] [PRS_E2E_01216] [PRS_E2E_01217] [PRS_E2E_01218] [PRS_E2E_01219] [PRS_E2E_01220] [PRS_E2E_01221] [PRS_E2E_01222] [PRS_E2E_01223] [PRS_E2E_01224] [PRS_E2E_01225] [PRS_E2E_01226] [PRS_E2E_01227] [PRS_E2E_01228] [PRS_E2E_UC_00236] [PRS_E2E_UC_00463] [PRS_E2E_UC_00464] [PRS_E2E_UC_01158] [PRS_E2E_UC_01168] [PRS_E2E_UC_01204]
[RS_E2E_08540]	E2E protocol shall support protected periodic/mixed periodic communication	[PRS_E2E_UC_00238] [PRS_E2E_UC_00239] [PRS_E2E_USE_00741]
[RS_E2E_08541]	E2E protocol shall support protected non-periodic communication	[PRS_E2E_UC_00238] [PRS_E2E_UC_00239] [PRS_E2E_UC_00606]

Requirement	Description	Satisfied by
[RS_E2E_08542]	E2E protocol shall support dynamic restart of communication peers	[PRS_E2E_00324]
[RS_E2E_08543]	E2E protocol shall support variable length of transmitted data	[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_08544]	E2E protocol shall provide a timeout detection mechanism	[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_08545]	E2E protocol shall provide a detection mechanism for corrupted data	[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_08546]	E2E protocol shall provide a detection mechanism for masquerade or incorrect addressing	[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_08547]	E2E protocol shall provide a detection mechanism for repetition, insertion or incorrect sequence of data	[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_08548]	E2E protocol shall provide E2E overall state to the application	[PRS_E2E_00218] [PRS_E2E_00219] [PRS_E2E_00372] [PRS_E2E_00394] [PRS_E2E_00479] [PRS_E2E_00480] [PRS_E2E_00503] [PRS_E2E_00522] [PRS_E2E_00678]

Requirement	Description	Satisfied by
[RS_E2E_08549]	Each E2E profile shall have a unique Profile ID	[PRS_E2E_00372] [PRS_E2E_00394] [PRS_E2E_00479] [PRS_E2E_00503] [PRS_E2E_00522] [PRS_E2E_00736]
[RS_E2E_08550]	The implementation of the E2E Supervision shall provide at least one of the E2E Profiles	[PRS_E2E_00372]

6 Functional specification

This chapter contains the specification of the internal functional behavior of the E2E supervision, this includes how the layout of the E2E-Header is defined, how the E2E-Header is created and how the E2E-Header is evaluated, and how the E2E-Statemachine is defined. For general introduction of the E2E supervision, see chapter 1.

Use case items like [PRS_E2E_UC_xxx] represent general hints on how to use the E2E protocol. However, they should not be read as requirements or limitation.

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.

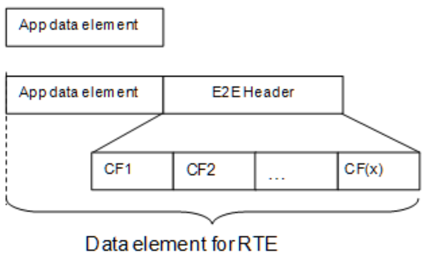

Figure 6.1: Safety protocol concept (with exemplary location of the E2E header)

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 Profiles have similar interfaces and behavior.

Each E2E Profile uses 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. A specific ID for every source (e.g., client) of a data element or message group
6. A message type distinguishing between requests and responses in case of E2E communication protection for methods
7. A message result distinguishing between normal and error responses in case of E2E communication protection for methods
8. 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.

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.

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)*

[PRS_E2E_00673] ⌈The AUTOSAR E2E-Protocol uses the following error codes: (see Table 6.1).⌋*(RS_E2E_08528)*

Type or error or status	Related code
At least one pointer parameter is a NULL pointer	E2E_E_INPUTERR_NULL
At least one input parameter is erroneous, e.g. out of range	E2E_E_INPUTERR_WRONG
Function completed successfully	E2E_E_OK

Table 6.1: Error Codes

6.2.2 Common Types of E2E Profiles

Some E2E profile make use of common data types which are shared among the different profiles. – Those shared types are introduced in this section instead of the profile specific sections.

6.2.2.1 Profile Xm Message Type Enumeration

[PRS_E2E_00739] ⌈The MessageType argument of the E2E_PXXmProtect, E2E_PXXmForward, and E2E_PXXmCheck functions shall be set to one of the following enumeration values (see Table 6.2).⌋*(RS_E2E_08528)*

Name	Value	Description
STD_MESSAGETYPE_REQUEST	0	The type of the message is a request message which is sent from the client to the server.
STD_MESSAGETYPE_RESPONSE	1	The type of the message is a response message which is sent from the server to the client.

Table 6.2: E2E Profile Xm Message Type Enumeration

6.2.2.2 Profile Xm Message Result Enumeration

[PRS_E2E_00743] ⌈The MessageResult argument of the E2E_PXXmProtect, E2E_PXXmForward, and E2E_PXXmCheck functions shall be set to one of the following enumeration values (see Table 6.3).⌋*(RS_E2E_08528)*

Name	Value	Description
STD_MESSAGERESULT_OK	0	The type of the result in the response message is a normal (i.e., a non erroneus) result. This value is also used for the request messages, where the value of this field is fixed to this value.
STD_MESSAGERESULT_ERROR	1	The type of the result int the response message is an error (i.e., an erroneus) result.

Table 6.3: E2E Profile Xm Message Result Enumeration

6.2.3 General Functionality of an E2E-Profile

Each E2E-Profile provides the following 3 functionalities:

1. Protect
2. Forward
3. Check

The ’protect’ functionality, simply called the ’protect function’ creates the E2E-Header and therefore protects the data to be sent over a communication medium.

The ’forward’ functionality, simply called the ’forward function’, is similar to the protect function and creates the header for the data to be transmitted but allows the additional replication of a received E2E-State. The main use-case for this function is Signal-Service-Translation where e.g. a E2E-protected signal is received, and the E2E-Status shall be replicated on the outgoing side.

The ’check’ functionality, simply called the ’check function’, evaluates the E2E-Header of the received message and checks for occurred communication faults. These faults are mirrored in the returned E2E-States.

In addition to the single E2E-Profiles a E2E-Statemachine evaluates the returned E2E-States over a longer period.

6.2.3.1 Functionality of the Counter

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.

6.2.3.2 Timeout detection

The previously mentioned mechanisms (e.g. for Profile 5: CRC, Counter, Data ID) enable to check the validity of received data element, when the receiver is running 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_PXXSTATUS_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.

6.2.3.3 Cyclic Redundancy Check

The Cyclic Redundancy Check, short CRC, is used to determine if bits flipped during the transmission of a message.

In contrast to errors indicated based on the evaluation of the counter - CRC-errors are unlikely to be a ‚false alarm’ (e.g. when using a good CRC-polynomial a detected CRC-error indicates that a data corruption occurred). Considering this fact, it is implausible that a stream of data without any detected CRC-errors contains a significant number of undetected corrupted data.

Due to this, a more stringent reaction upon CRC-errors is adequate. After detection of the first CRC-error on the subsequent data stream may contain a significant number of undetected corrupted data.

The maximum number of CRC-errors a receiver tolerates shall be limited, because the probability of receiving more than one undetected erroneous, within its error detection and qualification time interval, messages cannot be neglected. A wrong CRC indicates, that the integrity of the communication channel is affected.

The fault tolerance designed into the receiver (see UC_E2E_00170) may be exceeded as a possible consequence.

6.3 Specification of E2E Profiles - Generalized Part

This chapter contains the part of the specification for E2E profiles that is used in more than one profile specification. The behavior of E2E profiles is described independently of a specific profile. Text and figures use placeholder like "XX" which are replaced by a profile-specific value or text. All profile-specific content, including these placeholders, is defined in the corresponding profile-specific sub-chapter. This chapter does not apply to method profiles. This chapter does only apply to profiles where the fields of DataID and Length are part of the profile header. 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 6.4: Detectable communication faults

6.3.1 Counter

In E2E Profiles 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_01205] ⌈In E2E Profiles, 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, then it shall restart with 0 for the next send request. The maximum value of the counter depends on the size of the counter It is 0xFF (8bit counter), 0xFF'FF (16bit counter) or 0xFF'FF'FF'FF (32 bit).⌋*(RS_E2E_08539)*

6.3.2 Data ID

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

[PRS_E2E_UC_01204] ⌈In E2E profiles, the Data IDs should 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.

6.3.3 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).

6.3.4 CRC

E2E Profiles use CRC of different length depending on the length of the message to ensure a high detection rate and high Hamming Distance.

[PRS_E2E_01206] ⌈E2E Profiles shall use the CRC functions listed in chapters section 6.7, section 6.10, section 6.11, section 6.14, for calculating the CRC.⌋*(RS_E2E_08528, RS_E2E_08539)*

Note: The Crc_CalculateCRC32P4() is different from the 32 bit 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_01207] ⌈The CRC shall be calculated over the entire E2E header (excluding the CRC bytes) and over the user data.⌋*(RS_E2E_08531)*

6.3.5 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.

6.3.6 Creation of E2E-Header

6.3.6.1 E2E_PXXProtect()

The function E2E_PXXProtect() performs the steps as specified by the following diagrams in this section.

[PRS_E2E_01209] ⌈The function E2E_PXXProtect() shall have the overall behavior as shown in Figure 6.2.⌋ *(RS_E2E_08539)*

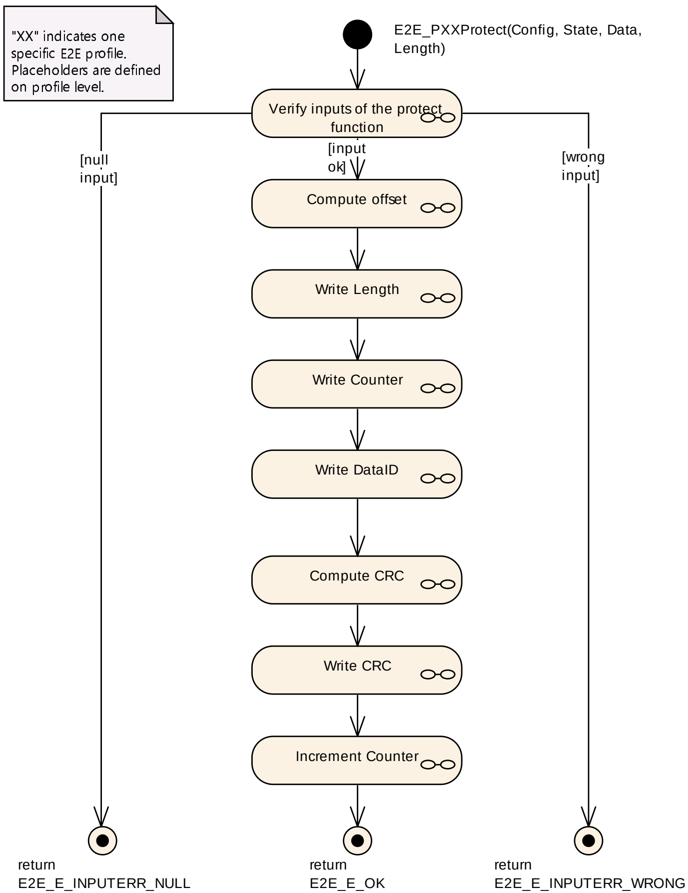

Figure 6.2: Behaviour of E2E_PXXProtect()

[PRS_E2E_01210] ⌈The step "Verify inputs of the protect function" in E2E_PXXProtect() shall behave as shown in Figure 6.3.⌋*(RS_E2E_08539)*

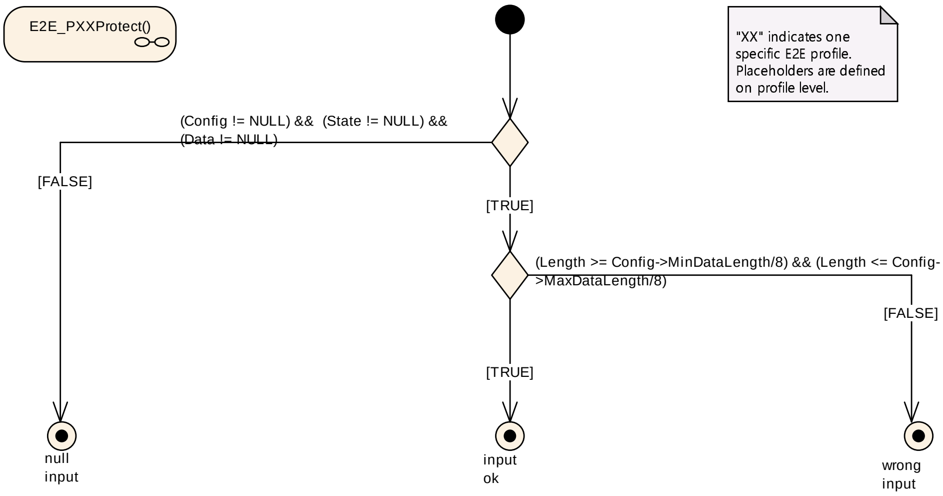

Figure 6.3: E2E_PXXProtect() step "Verify inputs of the protect function"

[PRS_E2E_01211] ⌈The step "Compute offset" in E2E_PXXProtect(), E2E_PXXForward() and E2E_PXXCheck() shall behave as shown in Figure 6.4.⌋*(RS_E2E_08539)*

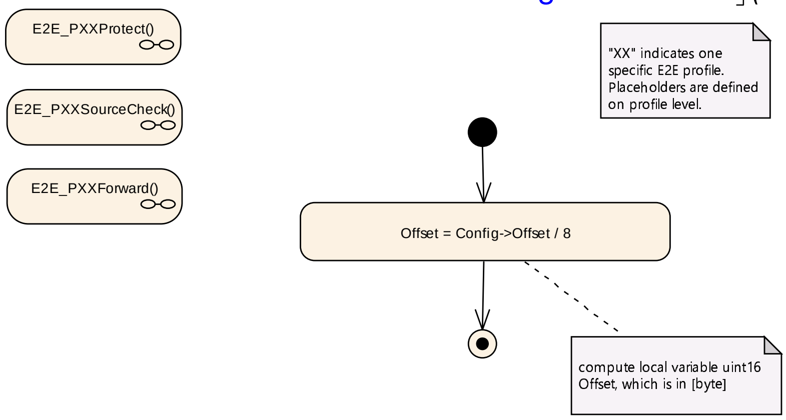

Figure 6.4: E2E_PXXProtect(), E2E_PXXForward() and E2E_PXXCheck() step "Compute offset"

[PRS_E2E_01212] ⌈The step "Write Length" in E2E_PXXProtect() and E2E_PXXForward() shall behave as shown in Figure 6.5.⌋*(RS_E2E_08539)*

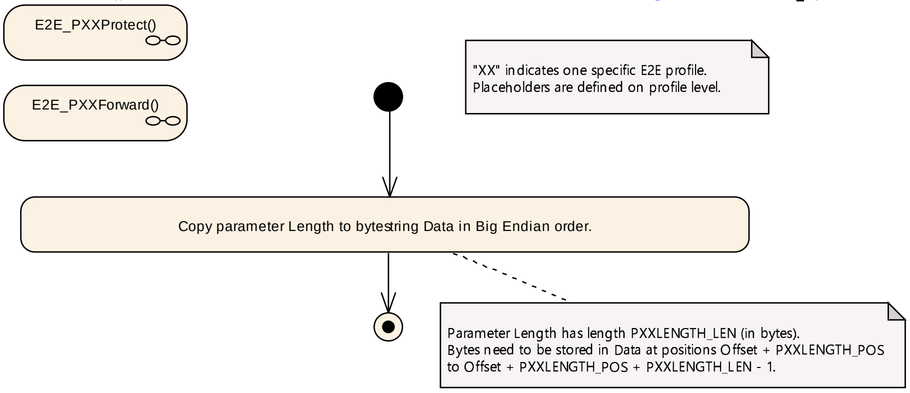

Figure 6.5: E2E_PXXProtect() and E2E_PXXForward() step "Write Length"

[PRS_E2E_01213] ⌈The step "Write Counter" in E2E_PXXProtect() shall behave as shown in Figure 6.6.⌋*(RS_E2E_08539)*

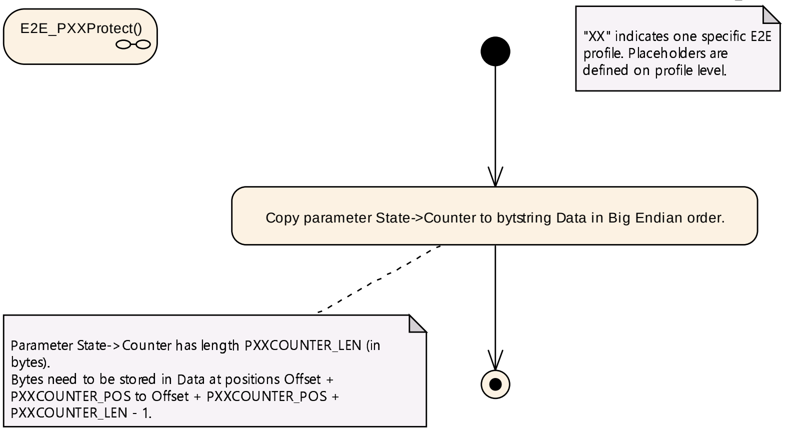

Figure 6.6: E2E_PXXProtect() step "Write Counter"

[PRS_E2E_01214] ⌈The step "Write DataID" in E2E_PXXProtect() shall behave as shown in Figure 6.7.⌋*(RS_E2E_08539)*

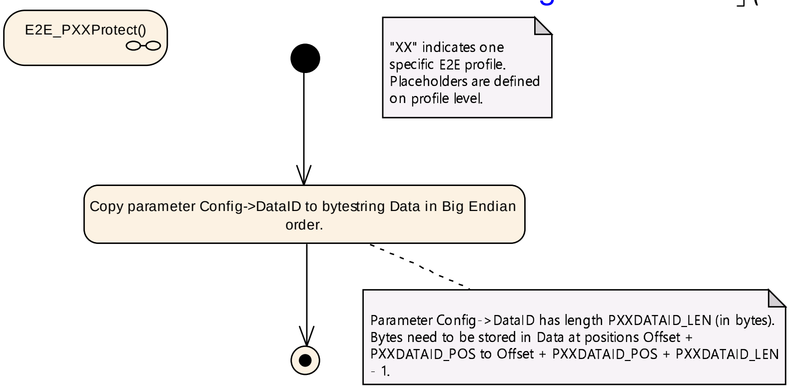

Figure 6.7: E2E_PXXProtect() step "Write DataID"

[PRS_E2E_01215] ⌈The step "Compute CRC" in E2E_PXXProtect(), E2E_PXXForward() and in E2E_PXXCheck()

shall behave as shown in Figure 6.8.⌋(*RS_E2E_08539*)

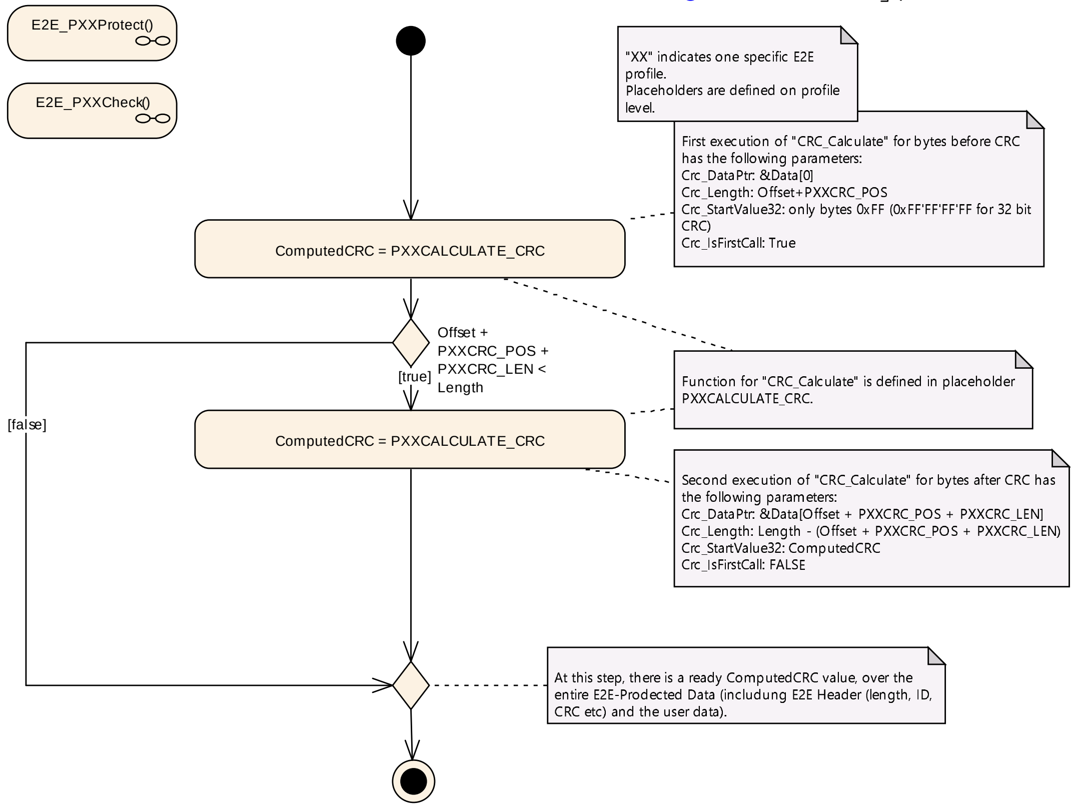

Figure 6.8: E2E_PXXProtect(), E2E_PXXCheck() and E2E_PXXForward() step "Compute CRC"

[PRS_E2E_01216] ⌈The step "Write CRC" in E2E_PXXProtect() and E2E_PXXForward() shall behave as shown in Figure 6.9.⌋(*RS_E2E_08539*)

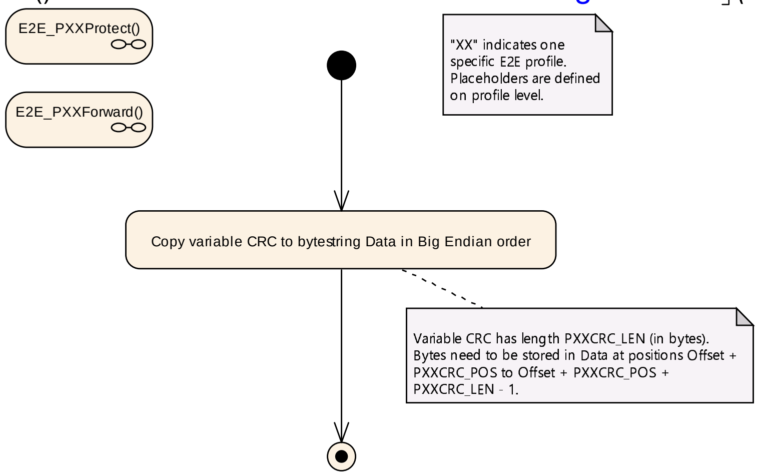

Figure 6.9: E2E_PXXProtect() and E2E_PXXForward() step "Write CRC"

[PRS_E2E_01217] ⌈The step "Increment Counter" in E2E_PXXProtect() and E2E_PXXForward() shall behave as shown in Figure 6.10.⌋*(RS_E2E_08539)*

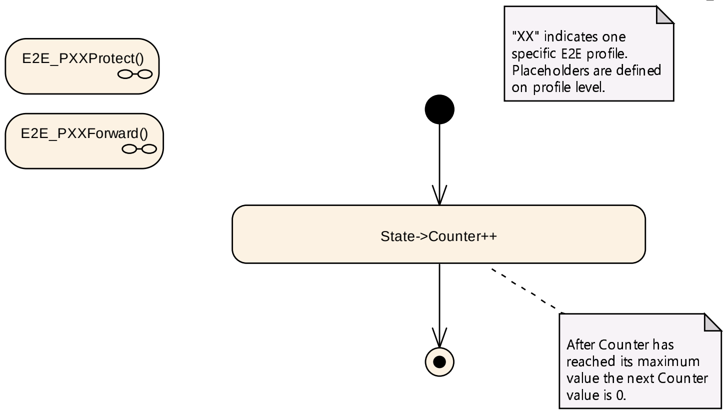

Figure 6.10: E2E_PXXProtect() and E2E_PXXForward() step "Increment Counter"

6.3.6.2 E2E_PXXForward()

The E2E_PXXForward() function of an E2E profile is called by a SW-C to protect its application data and to 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_PXXProtect(). The function E2E_PXXForward() performs the steps as specified by the following diagrams in this section.

[PRS_E2E_01218] ⌈The function E2E_PXXForward() shall have the overall behavior as shown in Figure 6.11.⌋*(RS_E2E_08539)*

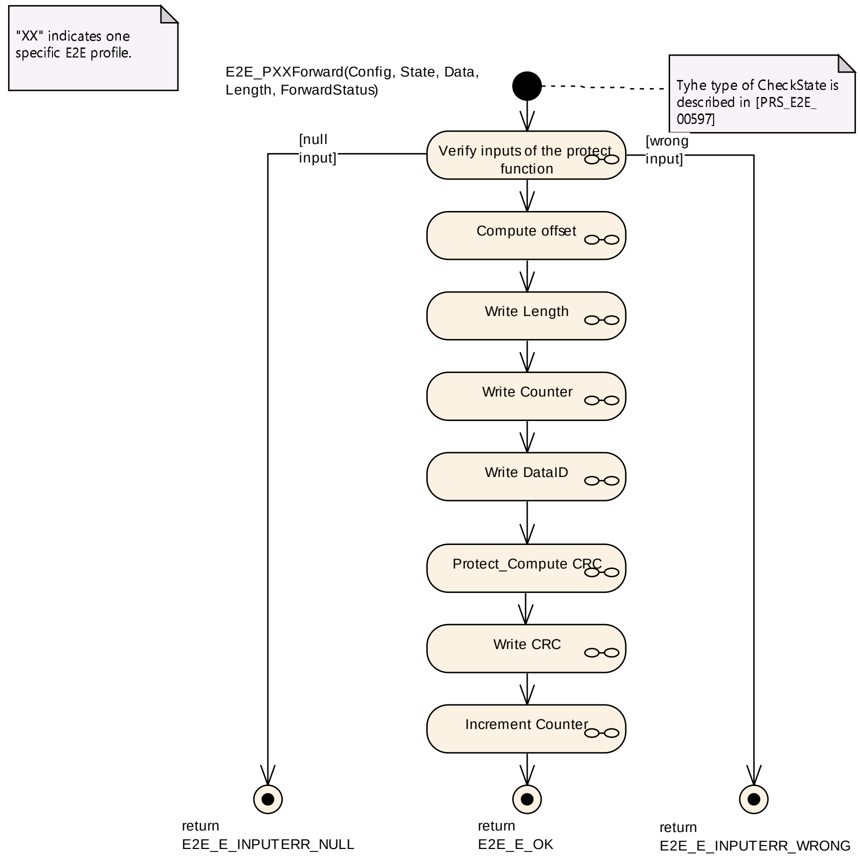

Figure 6.11: Behaviour of E2E_PXXForward()

Following steps are described in section 6.3.6.1

- ”Compute Offset” see [PRS_E2E_01211]
- ”Write Length” see [PRS_E2E_01212]
- ”Compute CRC” see [PRS_E2E_01215]
- ”Write CRC” see [PRS_E2E_01216]
- ”Increment Counter” see [PRS_E2E_01217]

[PRS_E2E_01228] ⌈The step ”Verify inputs of the forward function” in E2E_PXXForward() shall behave as shown in Figure 6.12.⌋*(RS_E2E_08539)*

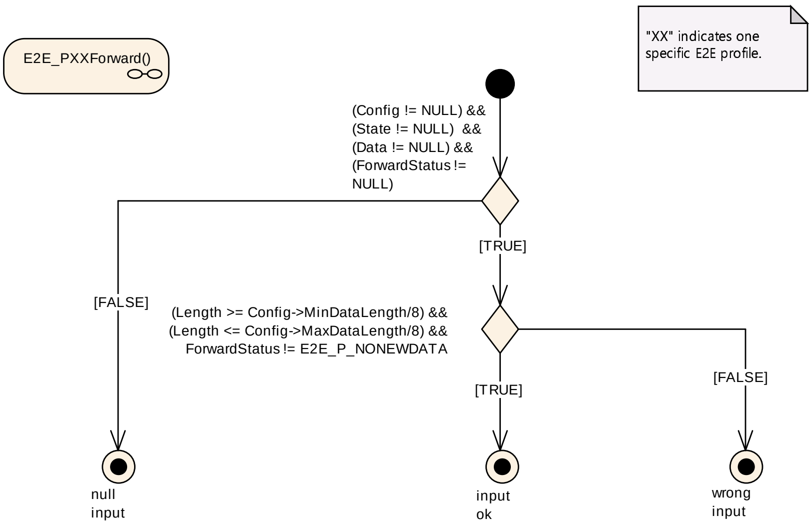

Figure 6.12: E2E_PXXForward() step "Verify inputs of the forward function"

[PRS_E2E_01219] ⌈The step "Write Counter" in E2E_PXXForward() shall behave as shown in Figure 6.13.⌋ *(RS_E2E_08539)*

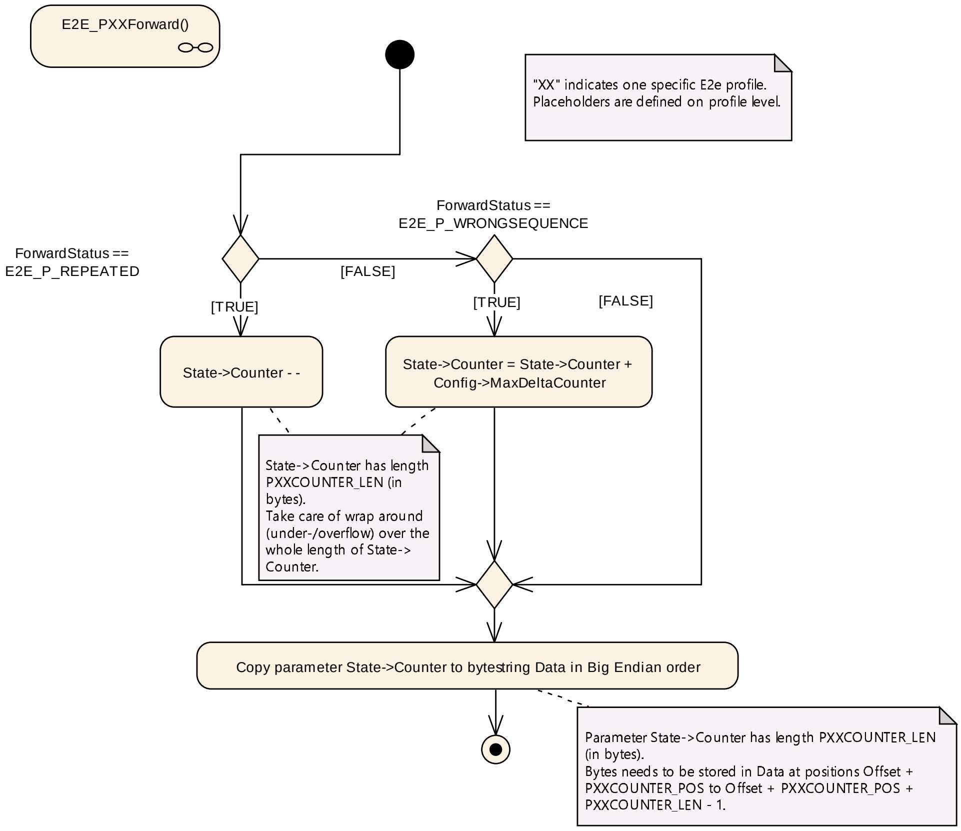

Figure 6.13: E2E_PXXForward() step "Write Counter"

[PRS_E2E_01220] ⌈The step "Write DataID" in E2E_PXXForward() shall behave as shown in Figure 6.14.⌋ *(RS_E2E_08539)*

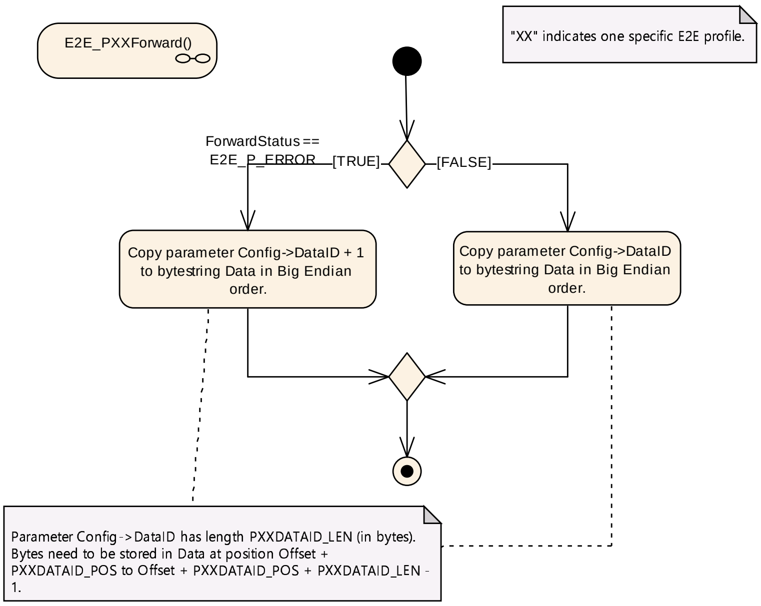

Figure 6.14: E2E_PXXForward() step ”Write DataID”

6.3.7 Evaluation of the E2E-Header

6.3.7.1 E2E_PXXCheck()

The function E2E_PXXCheck() performs the actions as specified by the following diagrams in this section and according to diagram Figure 6.15.

[PRS_E2E_01221] ⌈The function E2E_PXXCheck() shall have the overall behavior as shown in Figure 6.15.⌋ *(RS_E2E_08539)*

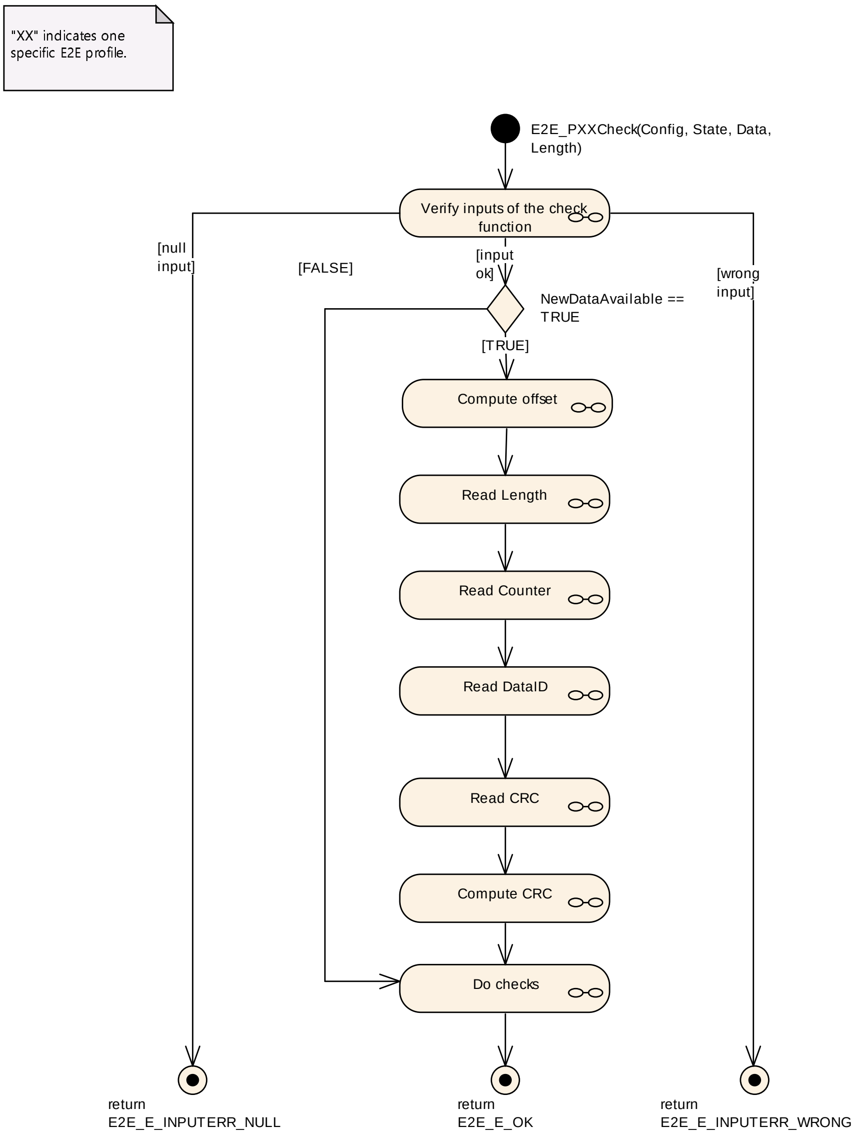

Figure 6.15: Behaviour of E2E_PXXCheck()

Following steps are described in section 6.3.6.1

- ”Compute Offset” see [PRS_E2E_01211]
- ”Compute CRC” see [PRS_E2E_01215]

[PRS_E2E_01222] ⌈The step ”Verify inputs of the check function” in E2E_PXXCheck() shall behave as shown in Figure 6.16.⌋*(RS_E2E_08539)*

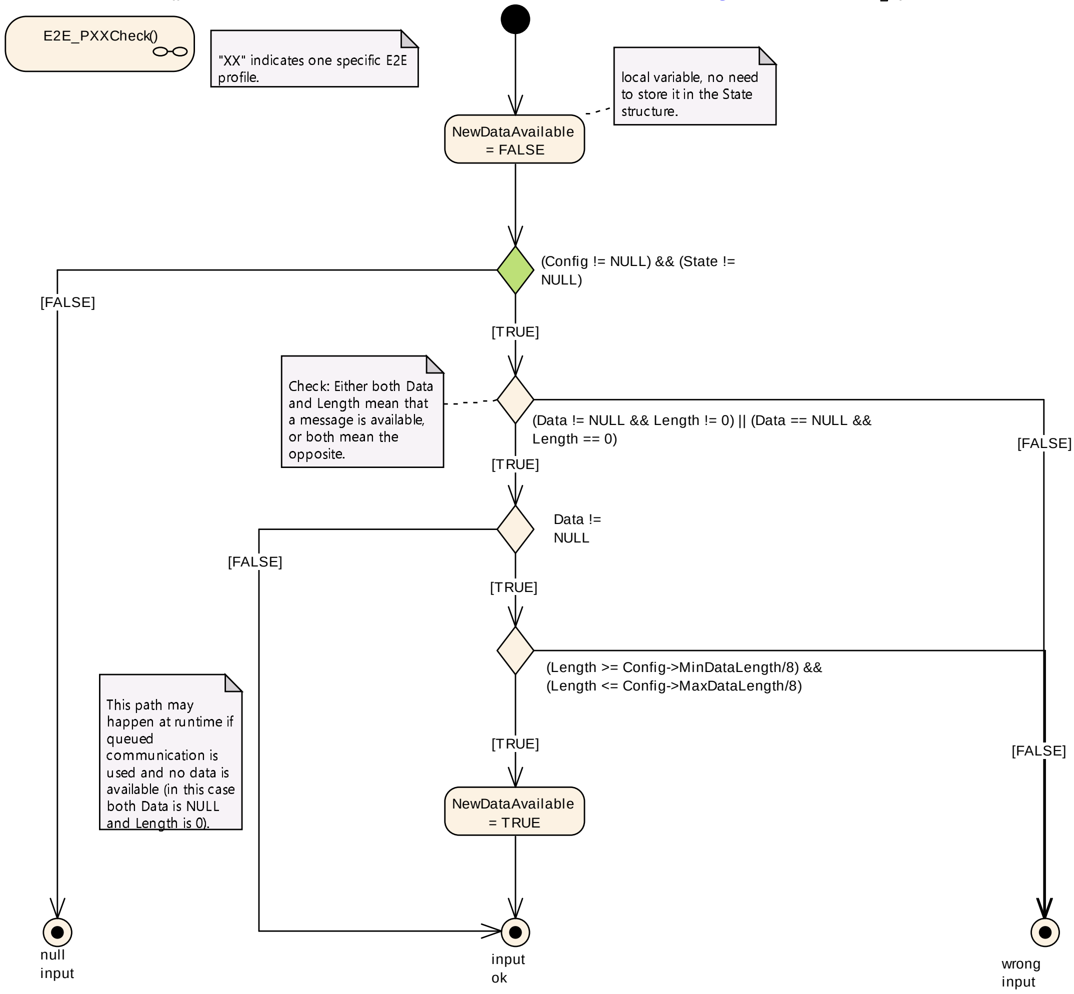

Figure 6.16: E2E_PXXCheck() step ’Verify inputs of the check function”

[PRS_E2E_01223] ⌈The step ”Read Length” in E2E_PXXCheck() shall behave as shown in Figure 6.17.⌋*(RS_E2E_08539)*

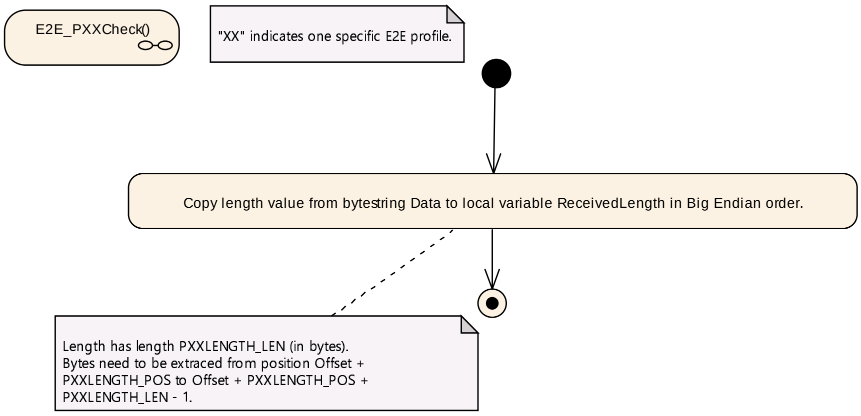

Figure 6.17: E2E_PXXCheck() step ”Read Length”

[PRS_E2E_01224] ⌈The step ”Read Counter” in E2E_PXXCheck() shall behave as shown in Figure 6.18.⌋*(RS_E2E_08539)*

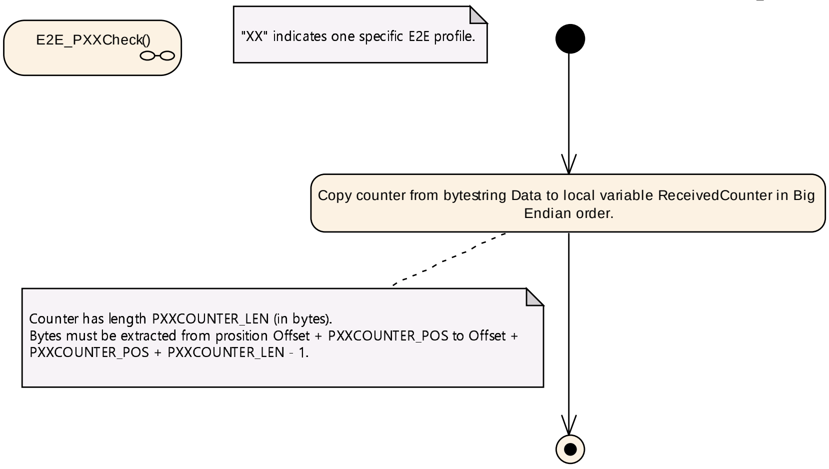

Figure 6.18: E2E_PXXCheck() step ”Read Counter”

[PRS_E2E_01225] ⌈The step ”Read DataID” in E2E_PXXCheck() shall behave as shown in Figure 6.19.⌋*(RS_E2E_08539)*

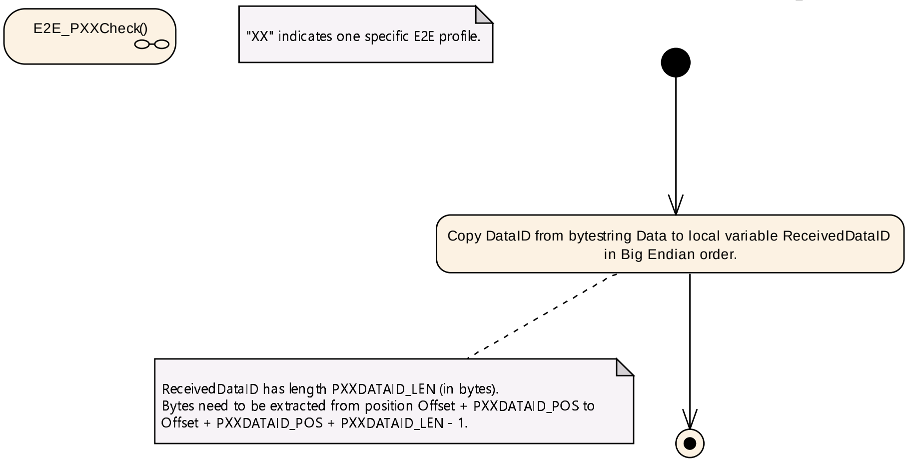

Figure 6.19: E2E_PXXCheck() step ”Read DataID”

[PRS_E2E_01226] ⌈The step ”Read CRC” in E2E_PXXCheck() shall behave as shown in Figure 6.20.⌋*(RS_E2E_08539)*

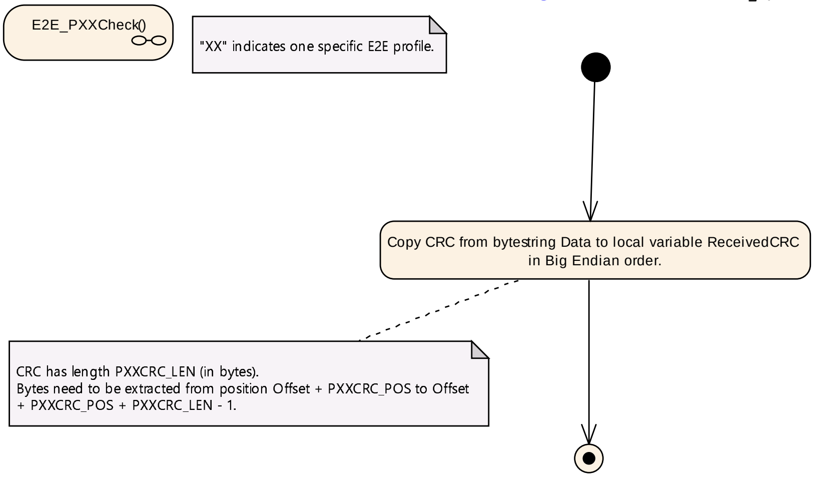

Figure 6.20: E2E_PXXCheck() step ”Read CRC”

[PRS_E2E_01227] ⌈The step ”Do Checks” in E2E_PXXCheck() shall behave as shown in Figure 6.21.⌋*(RS_E2E_08539)*

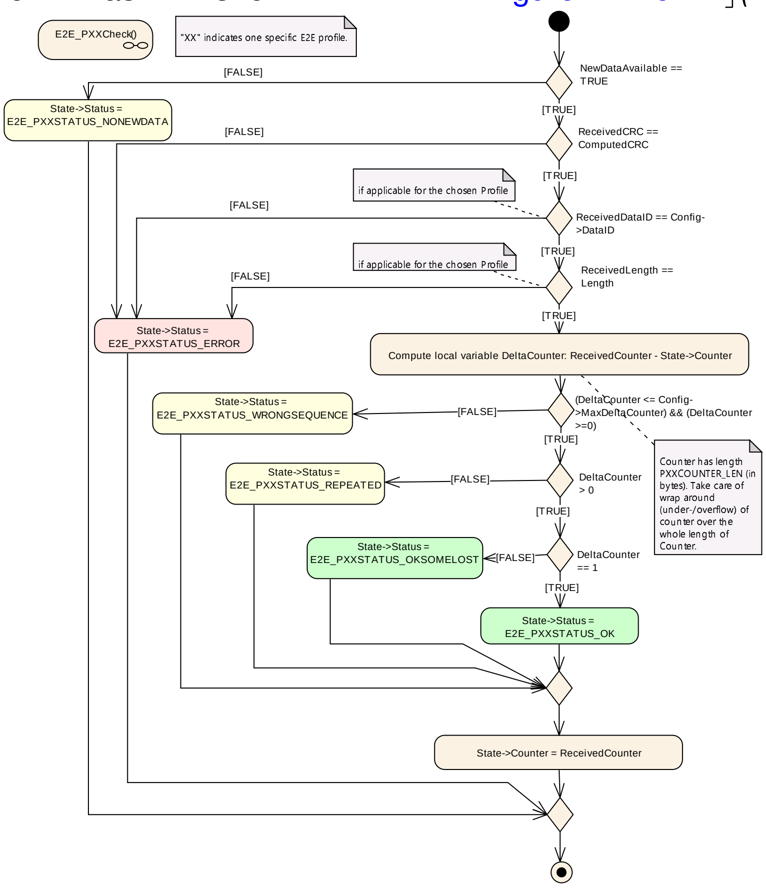

Figure 6.21: E2E_PXXCheck() step ”Do Checks”

6.3.8 Profile Data Types

6.3.8.1 Profile Protect State Type

[PRS_E2E_01250] ⌈The E2E_PXXProtect and E2E_PXXForward functions 'state' shall have the members defined in (see Table 6.5).⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter to be used for protecting the next Data. The initial value is 0, which means that in the first cycle, Counter is 0. Each time E2E_PXXProtect() is called, it increments the counter up to the maximum value (0xFF for 8 bit counter, 0xFF'FF for a 16 bit counter and 0xFF'FF'FF'FF for a 32 bit counter). After the maximum value is reached, the next value is 0x0. The overflow is not reported to the caller.

Table 6.5: E2E Profile Protect State Type

6.3.8.2 Profile Check Status Type

[PRS_E2E_01251] ⌈The E2E_PXXCheck functions 'State' shall have the members defined in (see Table 6.6).⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter of the data in previous cycle.
Status	Enumeration	Result of the verification of the Data in this cycle, determined by the Check function.

Table 6.6: E2E Profile XX Check State Type

6.3.8.3 Profile Check Status Enumeration

[PRS_E2E_01252] ⌈The E2E_PXXCheck functions 'State->Status' enumeration type shall consist the following enumeration values (see Table 6.7).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_PXXSTATUS_OK	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented by 1).

E2E_PXXSTATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is 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_PXXSTATUS_REPEATED.
E2E_PXXSTATUS_ERROR	Error	Error not related to counters occurred (e.g. wrong crc, wrong length, wrong options, wrong Data ID).
E2E_PXXSTATUS_REPEATED	Error	The checks of the Data in this cycle were successful, with the exception of the repetition.
E2E_PXXSTATUS_OKSOMELOST	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented within the allowed configured delta).
E2E_PXXSTATUS_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 6.7: E2E Profile Check Status Enumeration

6.4 Specification of E2E Profiles for Methods - Generalized Part

This chapter contains the part of the specification for E2E profiles for methods that is used in more than one profile specification. The behavior of E2E profiles is described independently of a specific profile. Text and figures use placeholder like ”XXm” which are replaced by a profile-specific value or text. All profile-specific content, including these placeholders, is defined in the corresponding profile-specific sub-chapter. This chapter applies to all method profiles where the fields of DataID and Length are part of the profile header.

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, Message type, Message Result, Source ID
Masquerading	Data ID, Message type, Message Result, Source ID, CRC
Incorrect addressing	Data ID, Message type, Message Result, Source 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 6.8: Detectable communication faults using method profiles

6.4.1 Counter

In E2E profiles for methods, the counter is initialized, incremented, reset and checked by the E2E profile. The counter is not manipulated or used by the caller of the E2E supervision.

[PRS_E2E_01156] ⌈In E2E profiles for methods, the counter on the sender side shall be initialized with 0 for the first transmission request of a data element the counter and shall be incremented by 1 for every subsequent send request. When the counter reaches the maximum value (0xFF'FF for a 16bit counter, 0xFF'FF'FF'FF for a 32bit counter, 0xFF'FF'FF'FF' FF'FF'FF'FF for a 64bit counter), then it shall restart with 0 for the next send request.⌋*(RS_E2E_08539)*

Note that the maximum counter value (0xFF'FF for a 16bit counter, 0xFF'FF'FF'FF for a 32bit counter, 0xFF'FF'FF'FF' FF'FF'FF'FF for a 64bit counter) is not reserved as a special invalid value, but it is used as a normal counter value.

In E2E profiles for methods, the following is detected on the receiver side by evaluating the counter of received data against the counter of previously received data:

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

Case 1 corresponds to the failed alive counter check, and case 2 correspond to successful alive counter check.

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

6.4.2 Data ID

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

[PRS_E2E_01157] ⌈In the E2E profiles for methods, 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_01158] ⌈In the E2E profiles for methods, the Data IDs shall be globally unique within the network of communicating systems (made of several ECUs each sending different data).⌋*(RS_E2E_08539)*

In case of usage of E2E supervision to protect 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 to be 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.

6.4.3 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).

6.4.4 CRC

E2E profiles for methods uses a suitable CRC, to ensure a high detection rate and high Hamming Distance.

[PRS_E2E_01159] ⌈E2E profiles for methods shall use the function defined in PXXM-CALCULATE_CRC of the SWS CRC supervision for calculating the CRC.⌋*(RS_E2E_08528, RS_E2E_08539)*

Note: The CRC used by E2E profiles for methods 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_01160] ⌈In E2E profiles for methods, the CRC shall be calculated over the entire E2E header (excluding the CRC bytes) and over the user data.⌋*(RS_E2E_08528)*

6.4.5 Message Type

The Message Type field is used to distinguish request messages from response messages in method communication.

[PRS_E2E_01161] ⌈In the E2E profiles for methods the Message Type field shall be explicitly transmitted, i.e. it shall be the part of the transmitted E2E header.⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_01162] ⌈A Message Type field with a value of 0 indicates a request message.⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_01163] ⌈A Message Type field with a value of 1 indicates a response message.⌋*(RS_E2E_08528, RS_E2E_08539)*

6.4.6 Message Result

The Message Result field is used to distinguish normal response messages from error response messages in method communication.

[PRS_E2E_01203] ⌈In the E2E profiles for methods the Message Result field shall be explicitly transmitted, i.e. it shall be the part of the transmitted E2E header.⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_01164] ⌈A Message Result field with a value of 0 indicates a normal response message.⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_01165] ⌈A Message Result field with a value of 1 indicates an error response message.⌋*(RS_E2E_08528, RS_E2E_08539)*

[PRS_E2E_01166] ⌈The Message Result field shall be set to 0 for request messages (i.e., in case the Message Type field is set to 0).⌋*(RS_E2E_08528, RS_E2E_08539)*

6.4.7 Source ID

The unique Source IDs are to verify the identity of the source of each transmitted safety-related data element. In case of method communication, the Source ID identifies the client which performs a method call.

[PRS_E2E_01167] ⌈In the E2E profiles for methods, the Source ID shall be explicitly transmitted, i.e. it shall be the part of the transmitted E2E header.⌋*(RS_E2E_08539)*

[PRS_E2E_UC_01168] ⌈In the E2E profiles for methods, the Source IDs shall be globally unique within the network of communicating systems (made of several ECUs).⌋*(RS_E2E_08539)*

6.4.8 Timeout detection

The previously mentioned mechanisms (CRC, Counter, Data ID, Length, Message Type, Message Result, and Source ID) enable to check the validity of received messages, when the receiver is executed independently from the data transmission, i.e. when receiver is not blocked waiting for 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.

6.4.9 Creation of the E2E header

6.4.9.1 E2E_PXXmProtect()

The function E2E_PXXmProtect() performs the steps as specified by the following diagrams in this section.

[PRS_E2E_01169] ⌈The function E2E_PXXmProtect() shall have the overall behavior as shown in Figure 6.22.⌋*(RS_E2E_08539)*

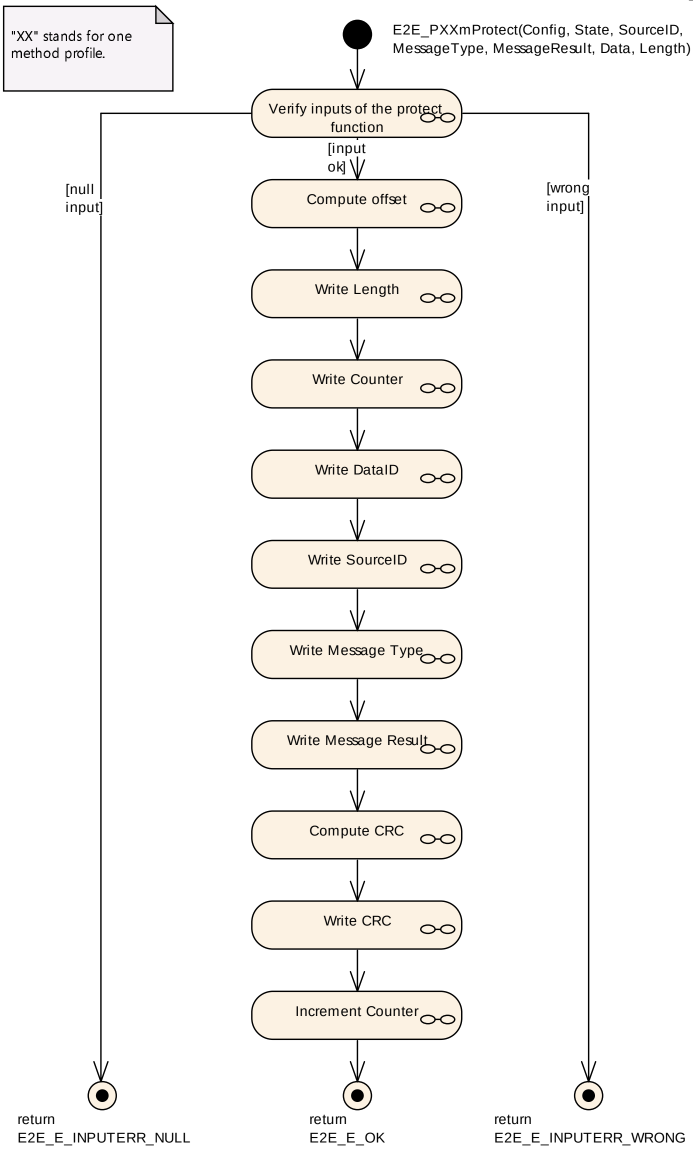

Figure 6.22: Behavior of E2E_PXXmProtect()

[PRS_E2E_01170] ⌈The step "Verify inputs of the protect function" in E2E_PXXmProtect() shall behave as shown in Figure 6.23.⌋*(RS_E2E_08539)*

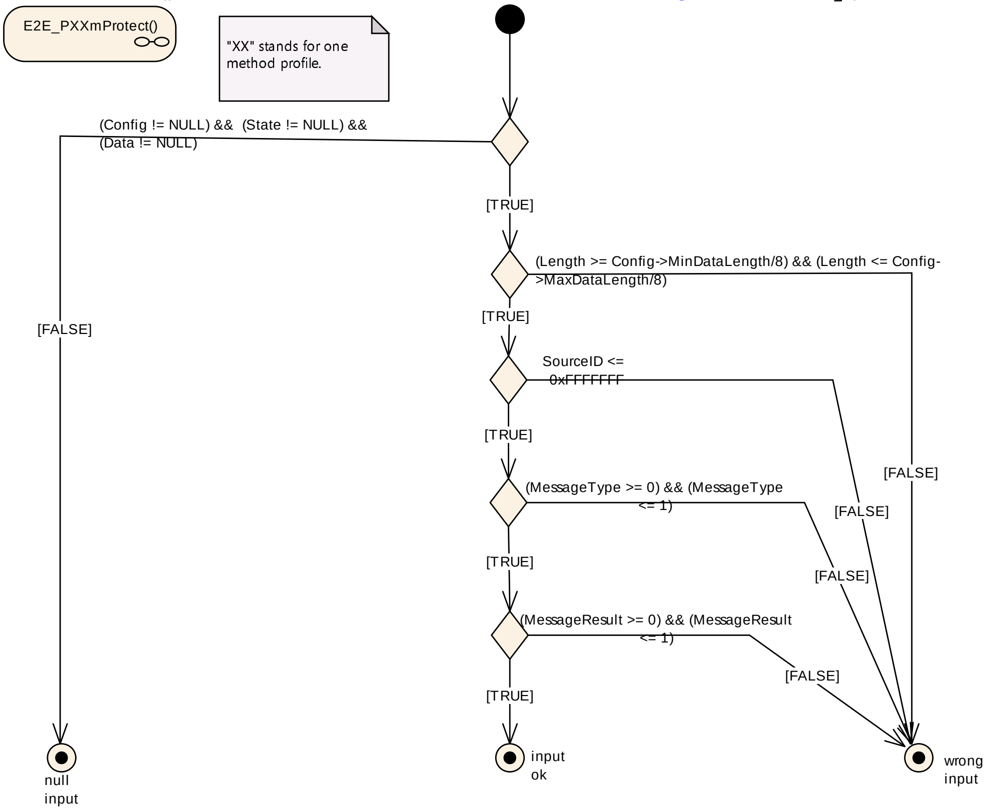

Figure 6.23: E2E_PXXmProtect() step "Verify inputs of the protect function"

[PRS_E2E_01171] ⌈The step "Compute offset" in E2E_PXXmProtect(), E2E_PXXmForward(), E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.24.⌋*(RS_E2E_08539)*

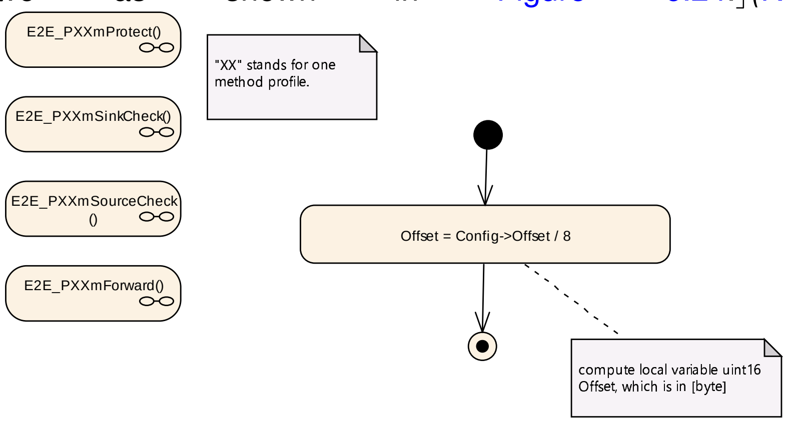

Figure 6.24: E2E_PXXmProtect(), E2E_PXXmForward(), E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() step "Compute offset"

[PRS_E2E_01172] ⌈The step "Write Length" in E2E_PXXmProtect() and E2E_PXXmForward() shall behave as shown in Figure 6.25.⌋*(RS_E2E_08539)*

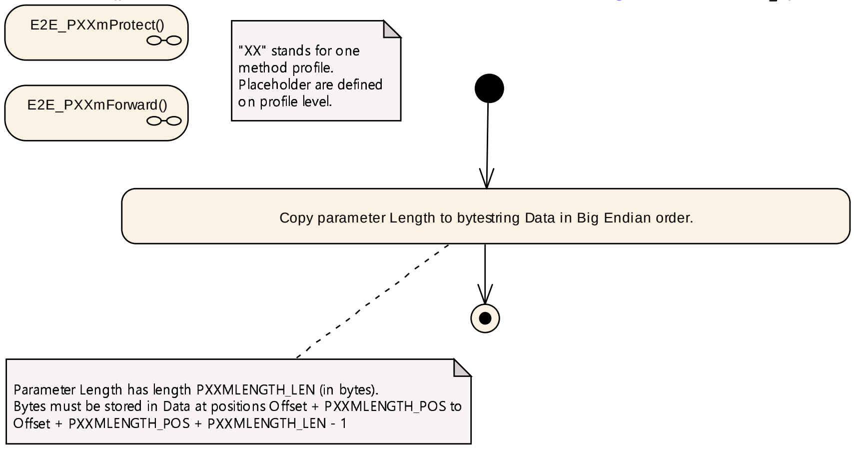

Figure 6.25: E2E_PXXmProtect() and E2E_PXXmForward() step "Write Length"

[PRS_E2E_01173] ⌈The step "Write Counter" in E2E_PXXmProtect() shall behave as shown in Figure 6.26.⌋*(RS_E2E_08539)*

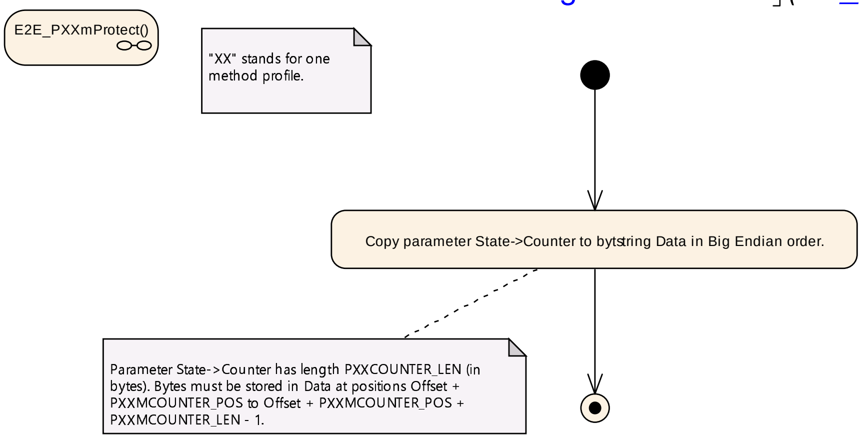

Figure 6.26: E2E_PXXmProtect() step "Write Counter"

[PRS_E2E_01174] ⌈The step "Write DataID" in E2E_PXXmProtect() shall behave as shown in Figure 6.27.⌋*(RS_E2E_08539)*

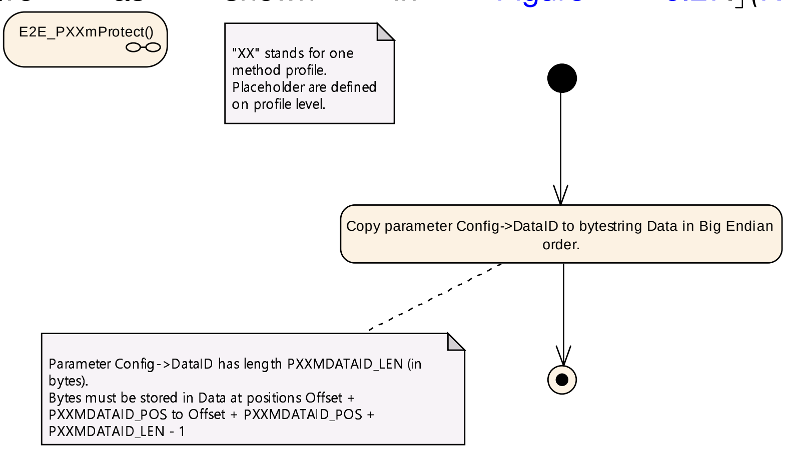

Figure 6.27: E2E_PXXmProtect() step "Write DataID"

[PRS_E2E_01175] ⌈The step "Write SourceID" in E2E_PXXmProtect() shall behave as shown in Figure 6.28.⌋*(RS_E2E_08539)*

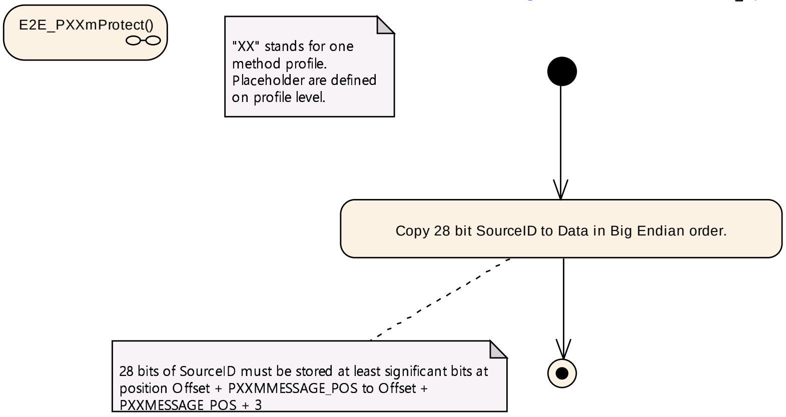

Figure 6.28: E2E_PXXmProtect() step "Write SourceID"

[PRS_E2E_01176] ⌈The step "Write Message Type" in E2E_PXXmProtect() shall behave as shown in Figure 6.29.⌋*(RS_E2E_08539)*

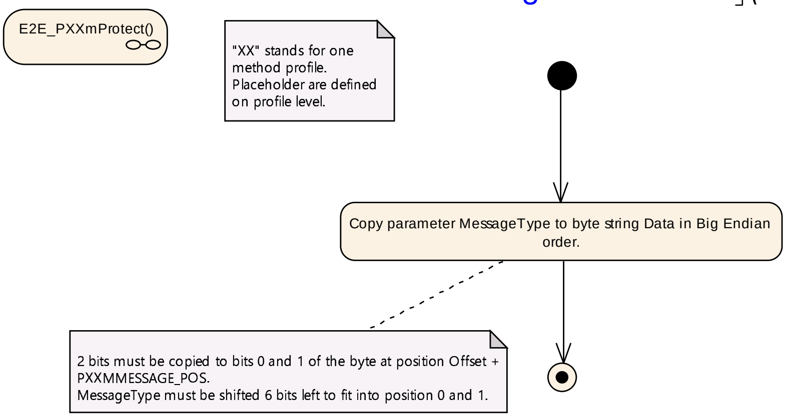

Figure 6.29: E2E_PXXmProtect() step "Write Message Type"

[PRS_E2E_01177] ⌈The step "Write Message Result" in E2E_PXXmProtect() shall behave as shown in Figure 6.30.⌋*(RS_E2E_08539)*

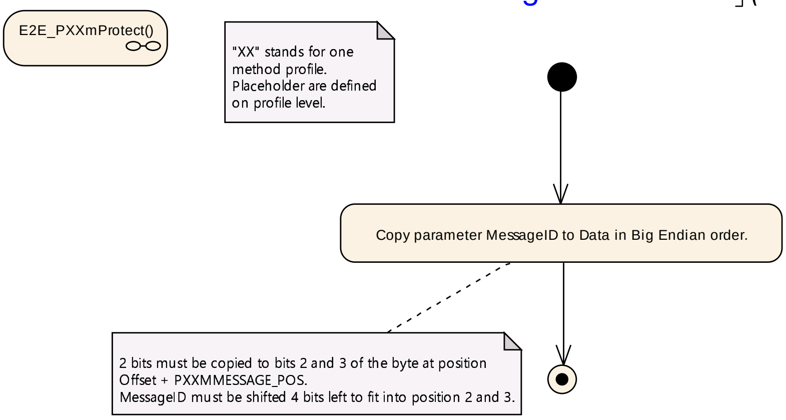

Figure 6.30: E2E_PXXmProtect() step "Write Message Result"

[PRS_E2E_01178] ⌈The step ”Compute CRC” in E2E_PXXmProtect(), E2E_PXXmForward(), E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.31.⌋*(RS_E2E_08539)*

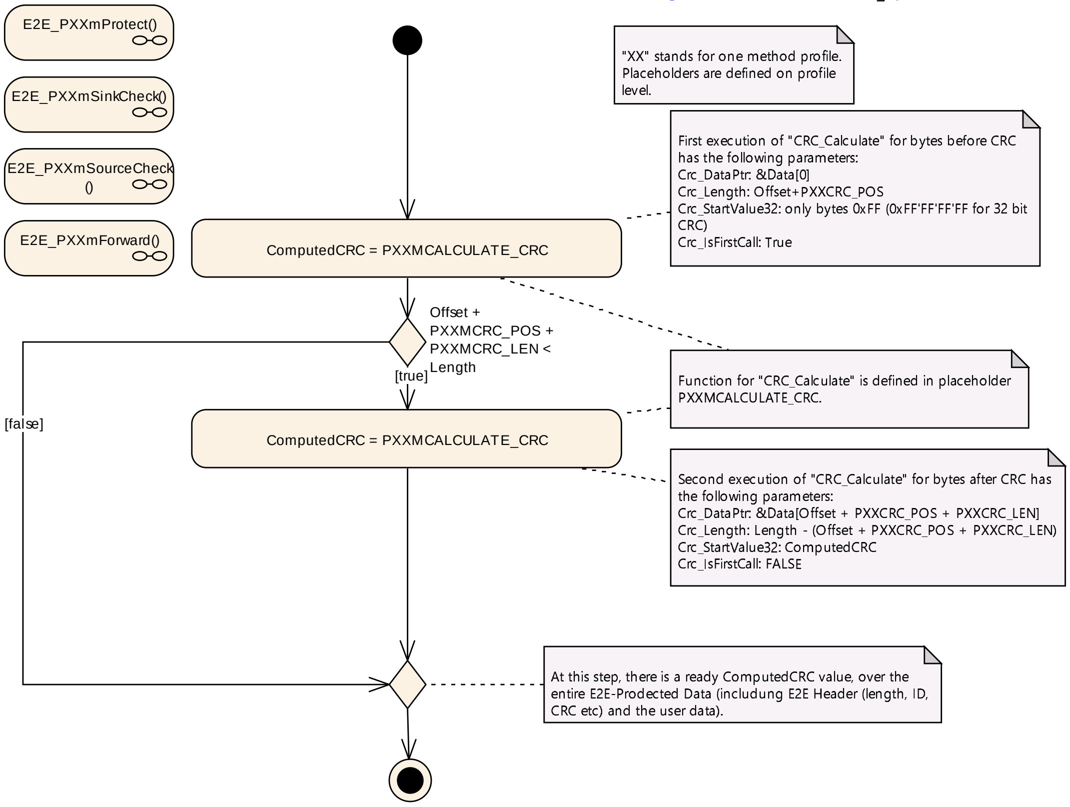

Figure 6.31: E2E_PXXmProtect(), E2E_PXXmForward(), E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() step ”Compute CRC”

[PRS_E2E_01179] ⌈The step ”Write CRC”in E2E_PXXmProtect() and E2E_PXXmForward() shall behave as shown in Figure 6.32.⌋*(RS_E2E_08539)*

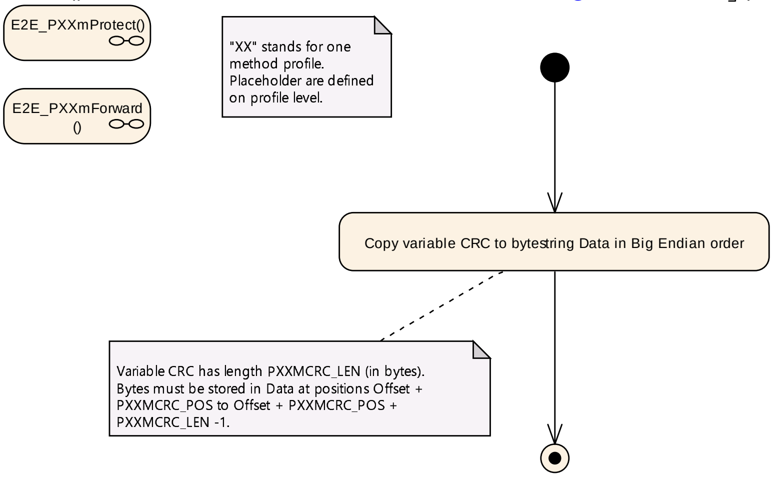

Figure 6.32: E2E_PXXmProtect() and E2E_PXXmForward() step ”Write CRC”

[PRS_E2E_01180] ⌈The step "Increment Counter" in E2E_PXXmProtect() and E2E_PXXmForward() shall behave as shown in Figure 6.33.⌋*(RS_E2E_08539)*

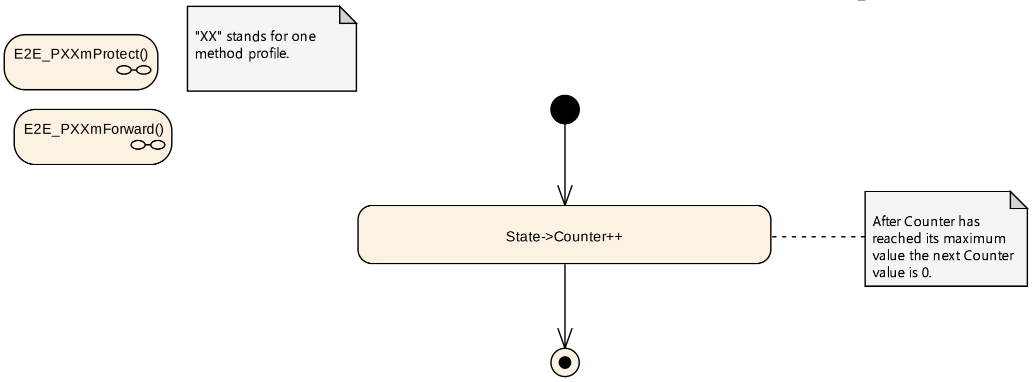

Figure 6.33: E2E_PXXmProtect() and E2E_PXXmForward() step "Increment Counter"

6.4.9.2 E2E_PXXmForward()

The E2E_PXXmForward() function of E2E profiles for methods is called by a SW-C to protect its application data and to forward a 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_PXXmProtect(). The function E2E_PXXmForward() performs the steps as specified by the following diagrams in this section.

[PRS_E2E_01181] Draft ⌈The function E2E_PXXmForward() shall have the overall behavior as shown in Figure 6.34.⌋*(RS_E2E_08539)*

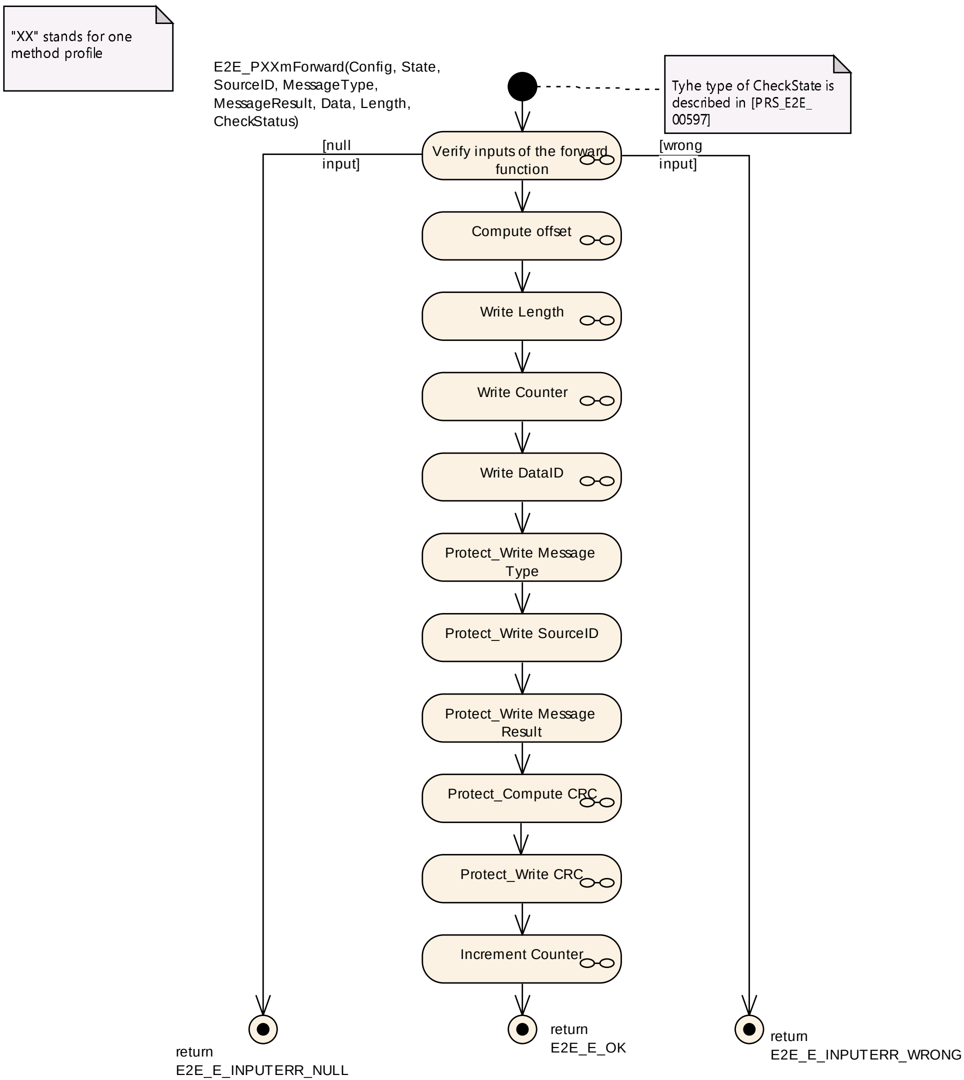

Figure 6.34: Behavior of E2E_PXXmForward()

Following steps are described in section 6.4.9.1

- ”Compute offset” see [PRS_E2E_01171]
- ”Write Length” see [PRS_E2E_01172]
- ”Write SourceID” see [PRS_E2E_01175]
- ”Write Message Type” see [PRS_E2E_01176]
- ”Write Message Result” see [PRS_E2E_01177]
- ”Compute CRC” see [PRS_E2E_01178]
- ”Write CRC” see [PRS_E2E_01179]
- ”Increment Counter” see [PRS_E2E_01180]

[PRS_E2E_01182] **Draft** ⌈The step "Verify inputs of the forward function" in E2E_PXXmForward() shall behave as shown in Figure 6.35.⌋*(RS_E2E_08539)*

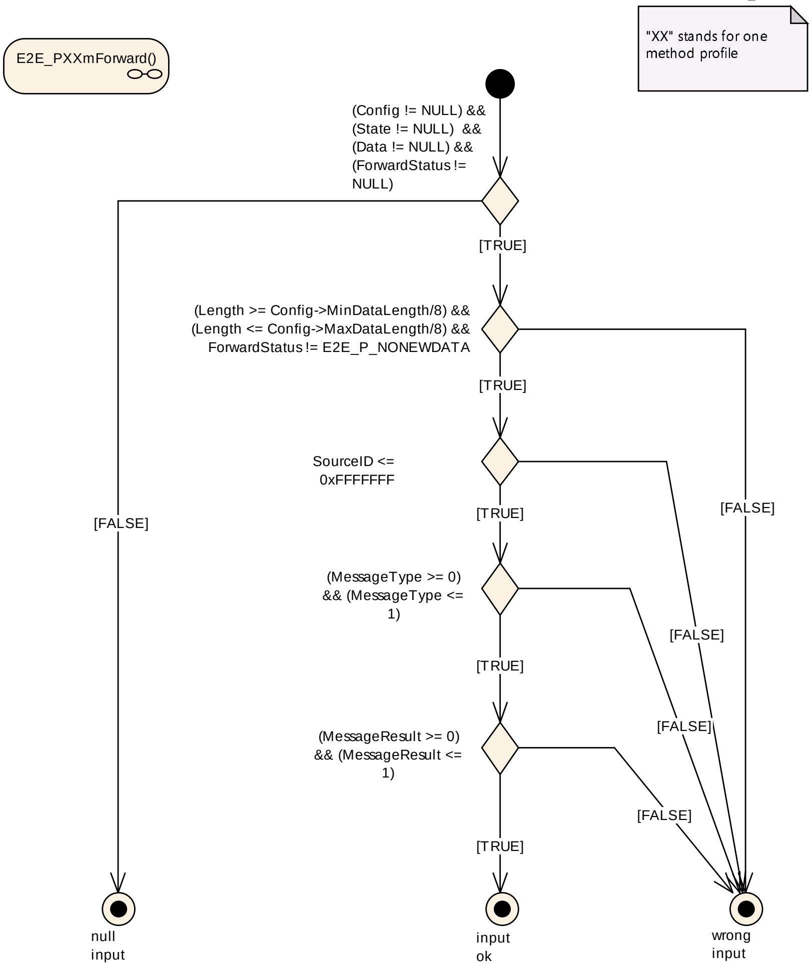

Figure 6.35: E2E_PXXmForward() step "Verify inputs of the forward function"

[PRS_E2E_01183] **Draft** ⌈The step ”Write Counter” in E2E_PXXmForward() shall behave as shown in Figure 6.36.⌋*(RS_E2E_08539)*

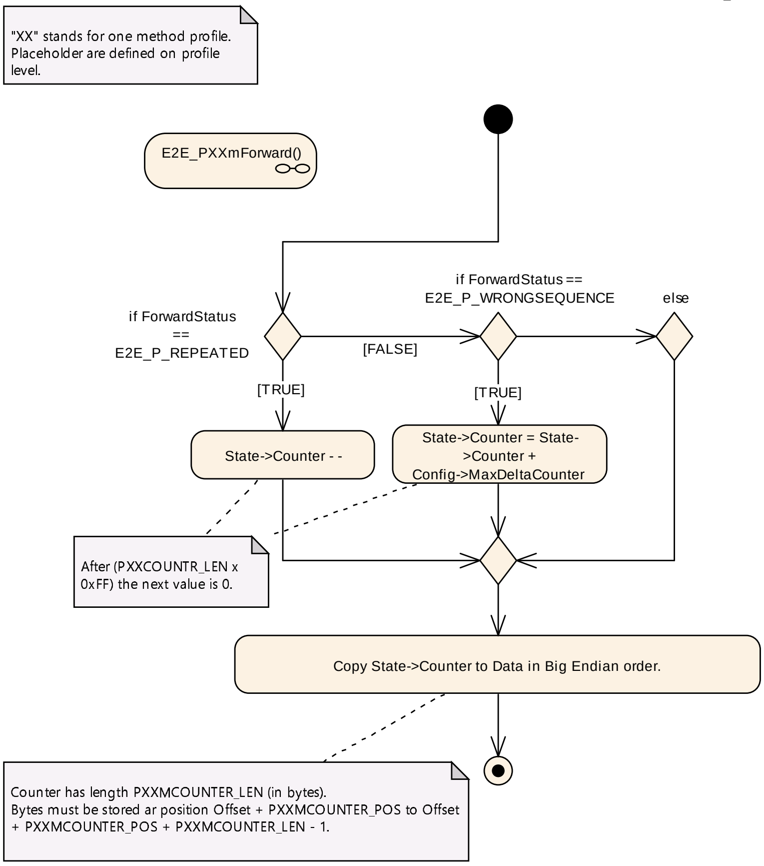

Figure 6.36: E2E_PXXmForward() step ”Write Counter”

[PRS_E2E_01184] **Draft** ⌈The step ”Write DataID” in E2E_PXXmForward() shall behave as shown in Figure 6.37.⌋*(RS_E2E_08539)*

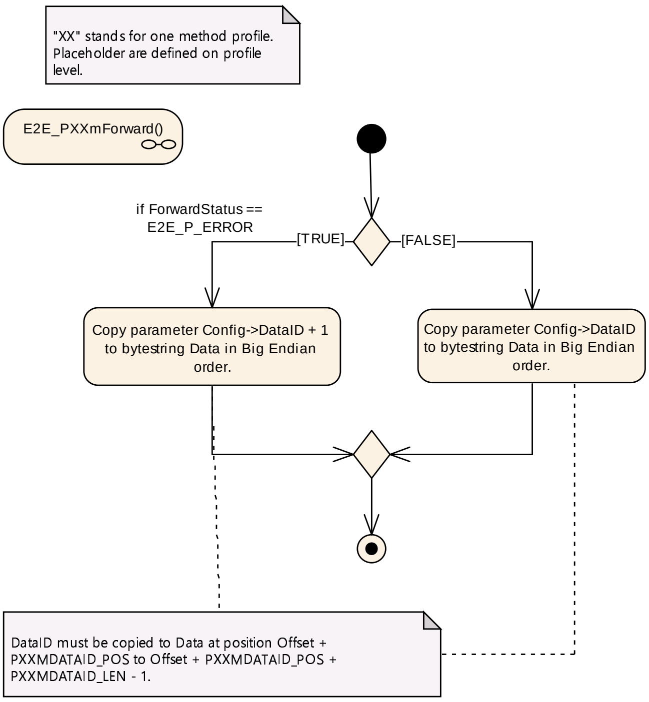

Figure 6.37: E2E_PXXmForward() step ”Write DataID”

6.4.10 Evaluation of the E2E Header

There are two check functions: E2E_PXXmSourceCheck() for the client side and E2E_PXXmSinkCheck() for the server side.

6.4.10.1 E2E_PXXmSourceCheck()

The function E2E_PXXmSourceCheck() for the client side performs the actions as specified by the following diagrams in this section.

[PRS_E2E_01185] ⌈"The function E2E_PXXmSourceCheck() for the client side shall have the overall behavior as shown in Figure 6.38.⌋*(RS_E2E_08539)*

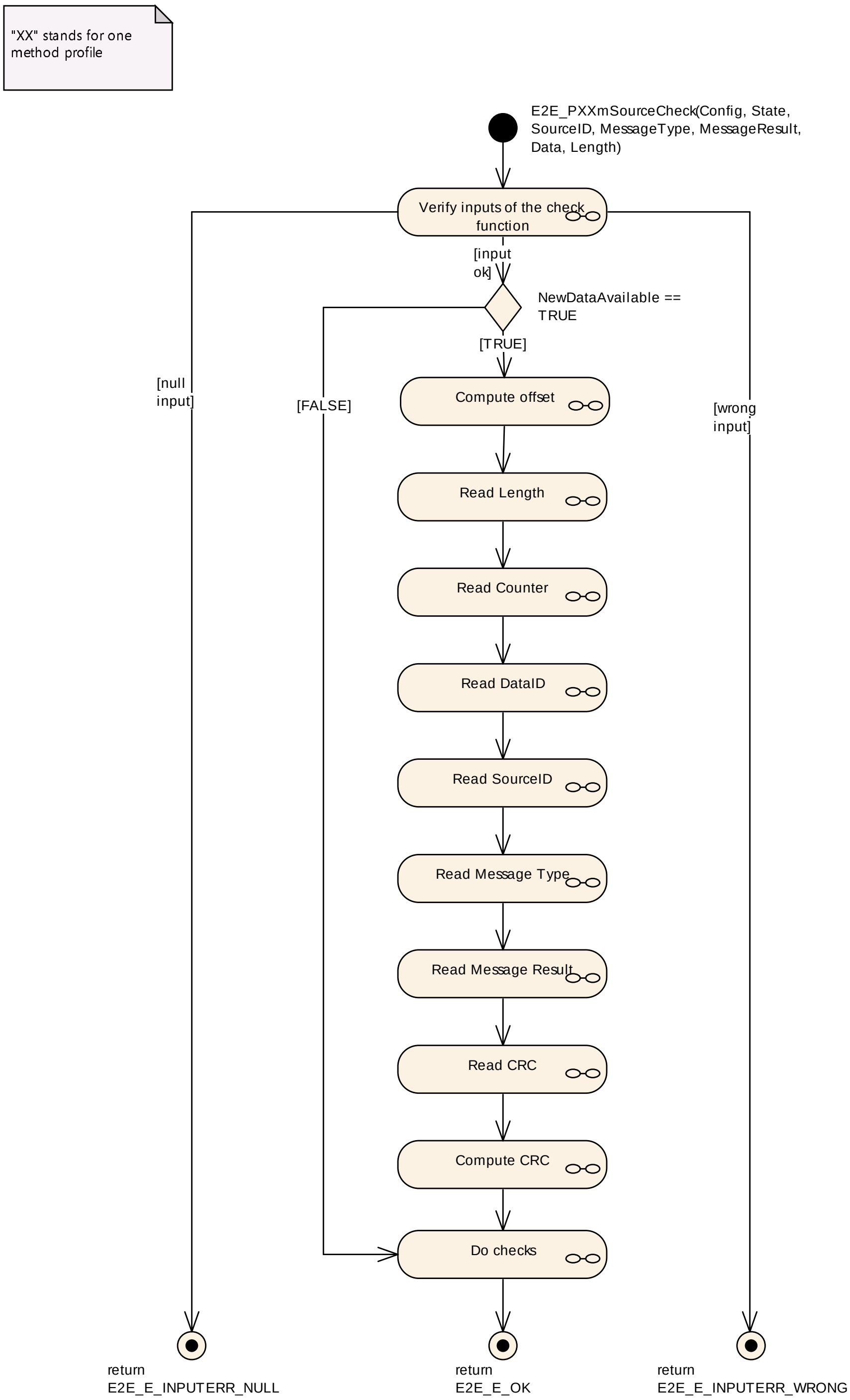

Figure 6.38: Behavior of E2E_PXXmSourceCheck()

[PRS_E2E_01186] ⌈The step ”Verify inputs of the function” in E2E_PXXmSourceCheck() shall behave as shown in Figure 6.39.⌋*(RS_E2E_08539)*

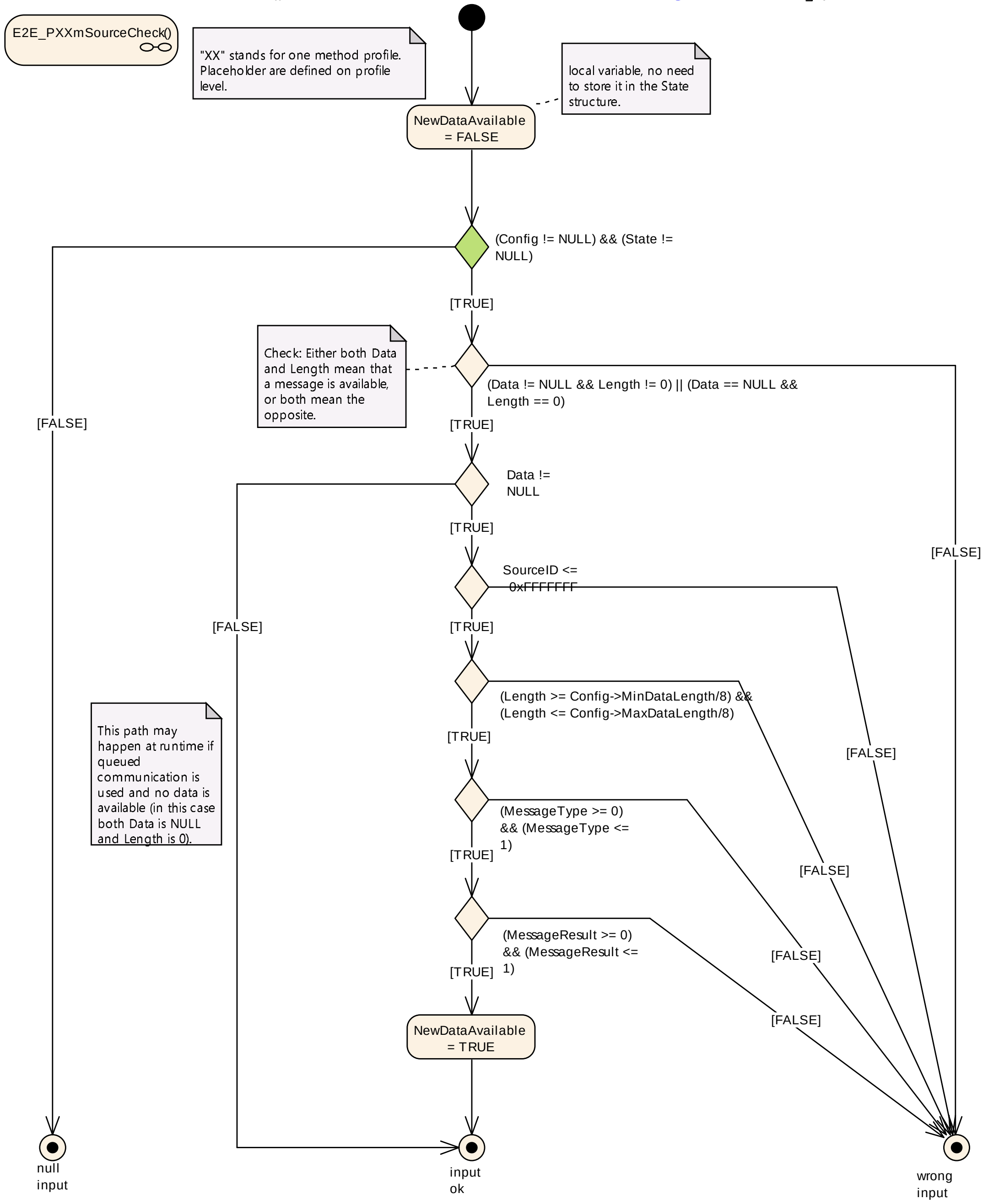

Figure 6.39: E2E_PXXmSourceCheck() step ”Verify inputs of the check function”

[PRS_E2E_01187] ⌈The step "Read Length" in E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.40.⌋*(RS_E2E_08539)*

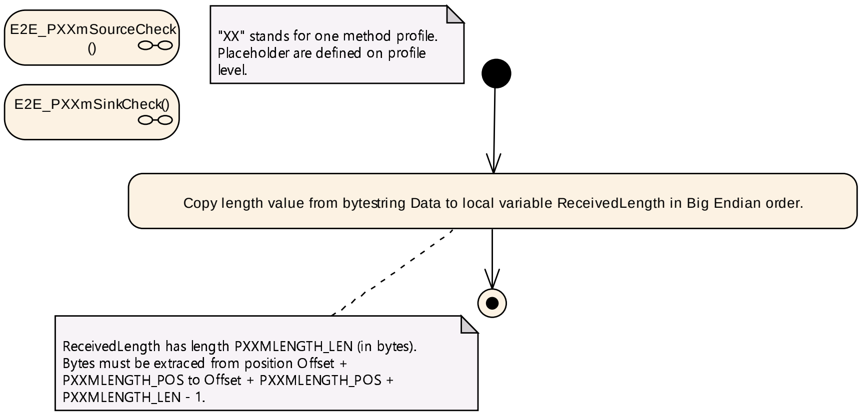

Figure 6.40: E2E_PXXmSourceCheck()/E2E_PXXmSinkCheck() step "Read Length"

[PRS_E2E_01188] ⌈The step "Read Counter" in E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.41.⌋*(RS_E2E_08539)*

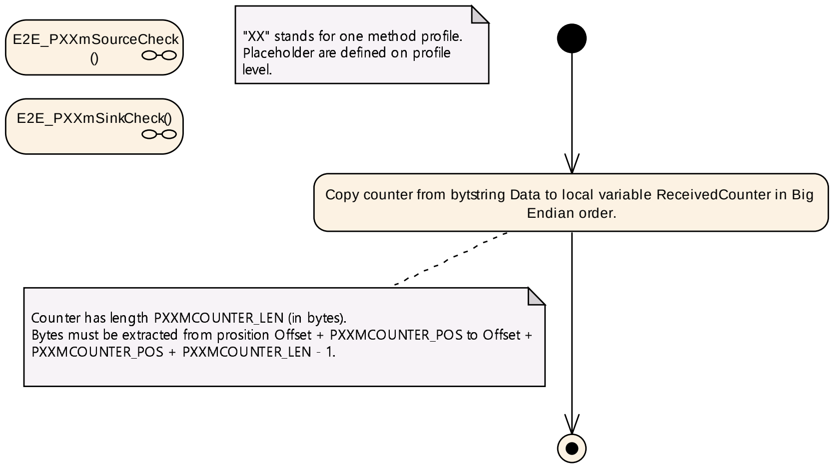

Figure 6.41: E2E_PXXmSourceCheck()/E2E_PXXmSinkCheck() step "Read Counter"

[PRS_E2E_01189] ⌈The step "Read DataID" in E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.42.⌋*(RS_E2E_08539)*

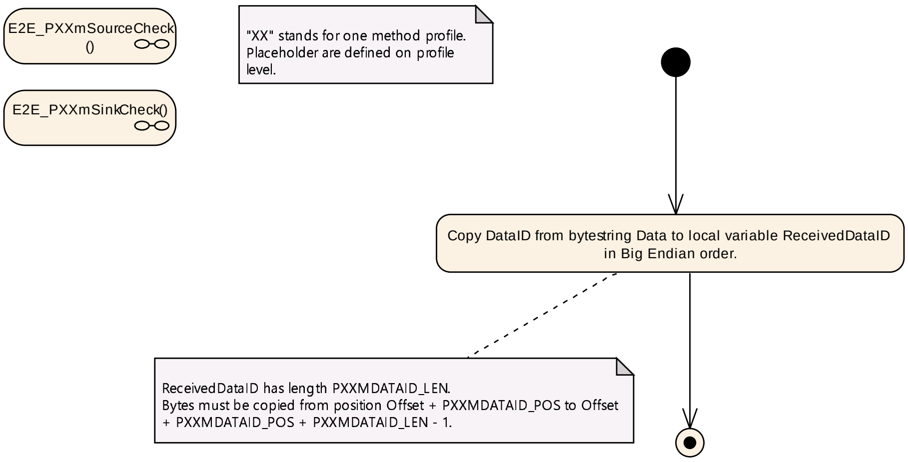

Figure 6.42: E2E_PXXmSourceCheck()/E2E_PXXmSinkCheck() step "Read DataID"

[PRS_E2E_01190] ⌈The step "Read SourceID" in E2E_PXXmSourceCheck() shall behave as shown in Figure 6.43.⌋*(RS_E2E_08539)*

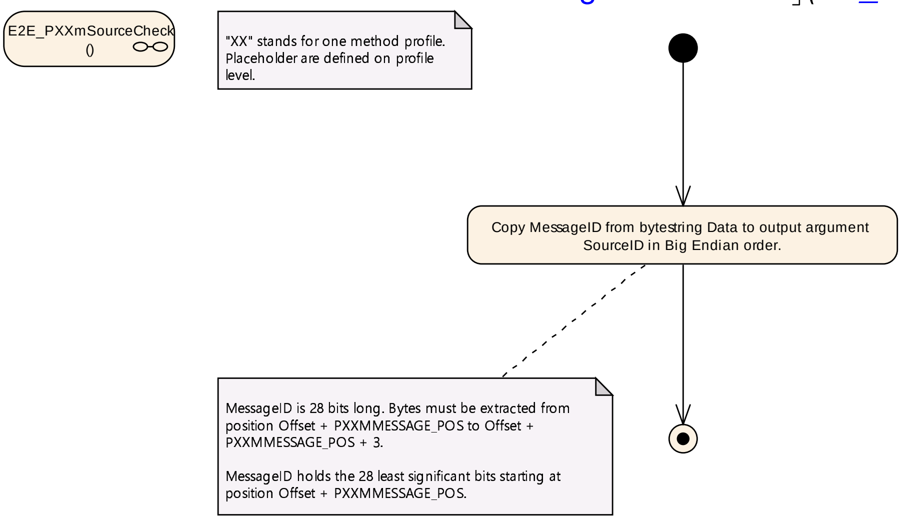

Figure 6.43: E2E_PXXmSourceCheck() step "Read SourceID"

[PRS_E2E_01191] ⌈The step "Read Message Type" in E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.44.⌋*(RS_E2E_08539)*

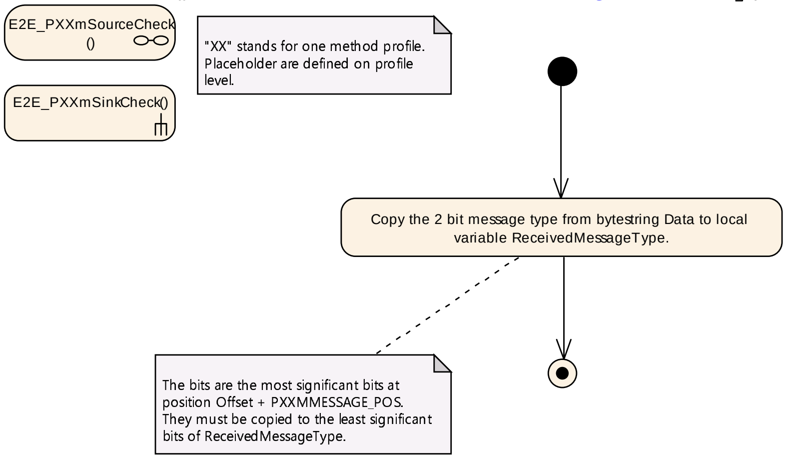

Figure 6.44: E2E_PXXmSourceCheck()/E2E_PXXmSinkCheck() step "Read Message Type"

[PRS_E2E_01192] ⌈The step "Read Message Result" in E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.45.⌋*(RS_E2E_08539)*

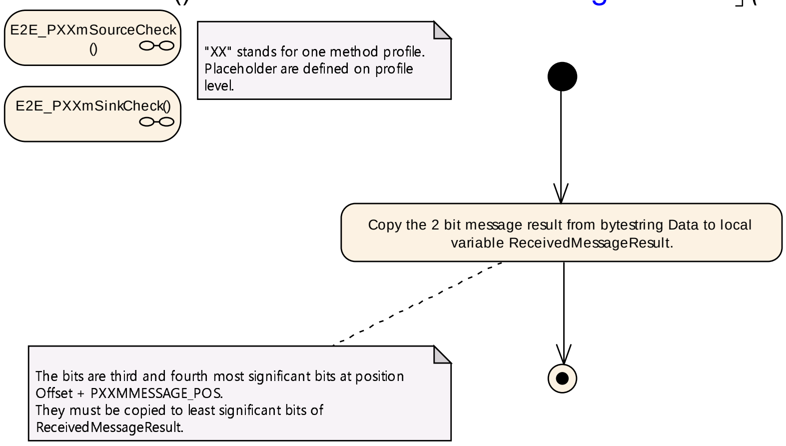

Figure 6.45: E2E_PXXmSourceCheck()/E2E_PXXmSinkCheck() step "Read Message Result"

[PRS_E2E_01193] ⌈The step "Read CRC" in E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall behave as shown in Figure 6.46.⌋*(RS_E2E_08539)*

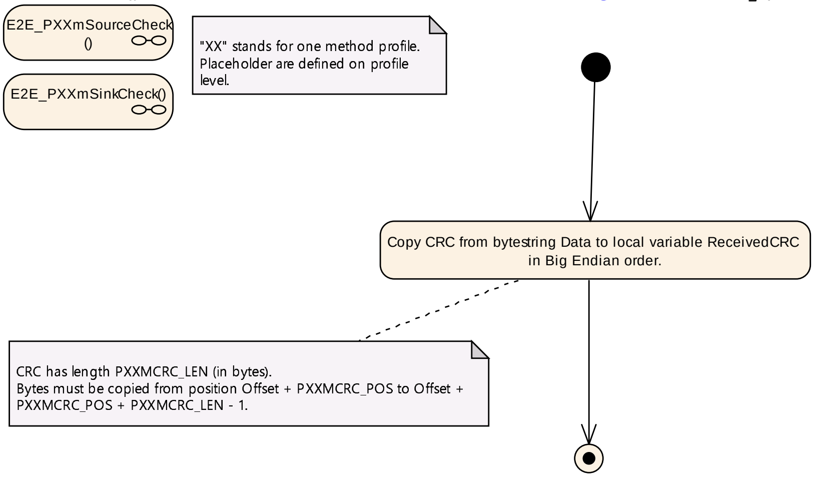

Figure 6.46: E2E_PXXmSourceCheck()/E2E_PXXmSinkCheck() step "Read CRC"

[PRS_E2E_01194] ⌈The step ”Do Checks” in E2E_PXXmSourceCheck() shall behave as shown in Figure 6.47.⌋*(RS_E2E_08539)*

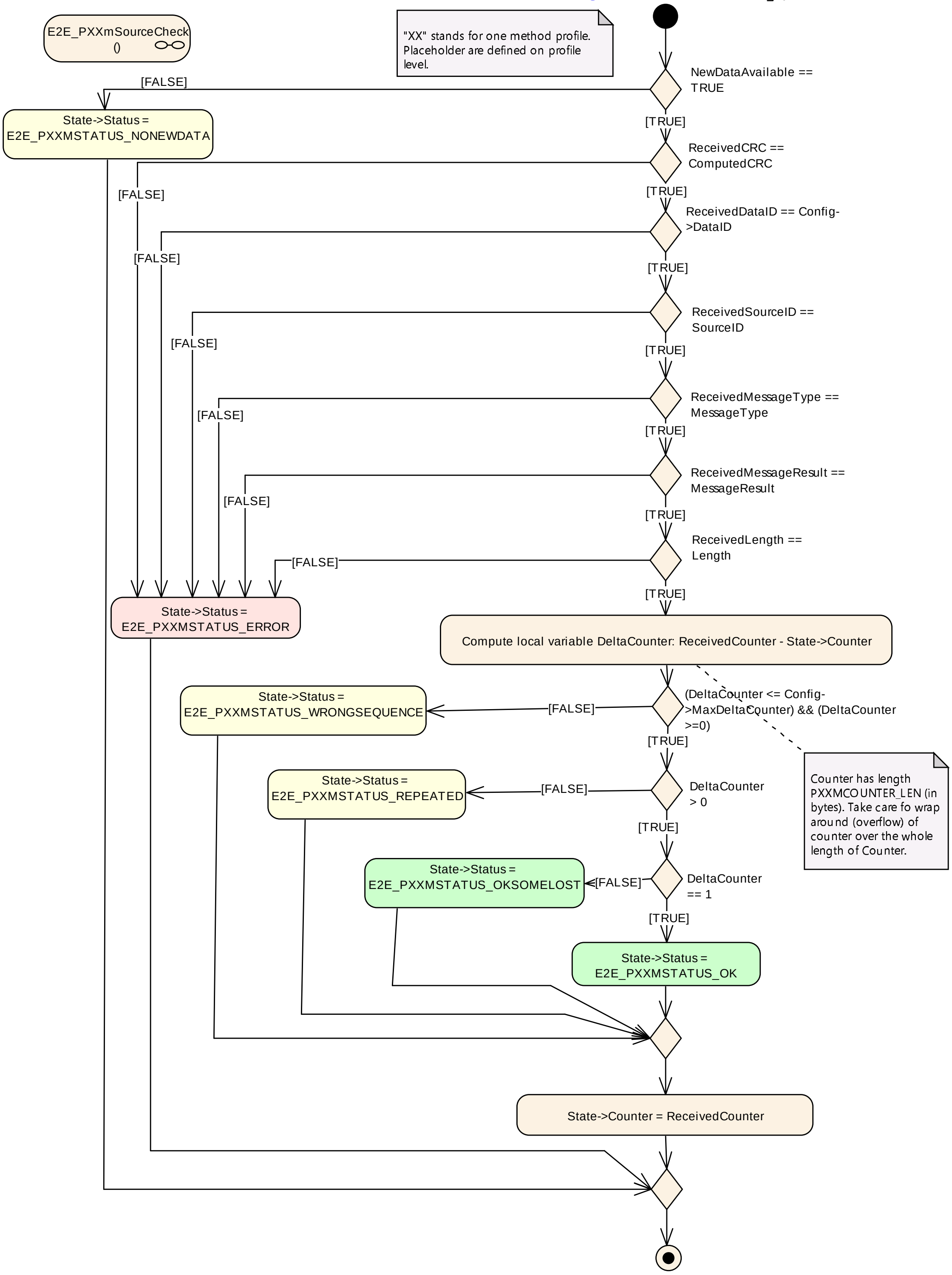

Figure 6.47: E2E_PXXmSourceCheck() step ”Do Checks”

6.4.10.2 E2E_PXXmSinkCheck()

The function E2E_PXXmSinkCheck() for the server side performs the actions as specified by the following diagrams in this section.

[PRS_E2E_01195] ⌈The function E2E_PXXmSinkCheck() for the server side shall have the overall behavior as shown in Figure 6.48.⌋*(RS_E2E_08539)*

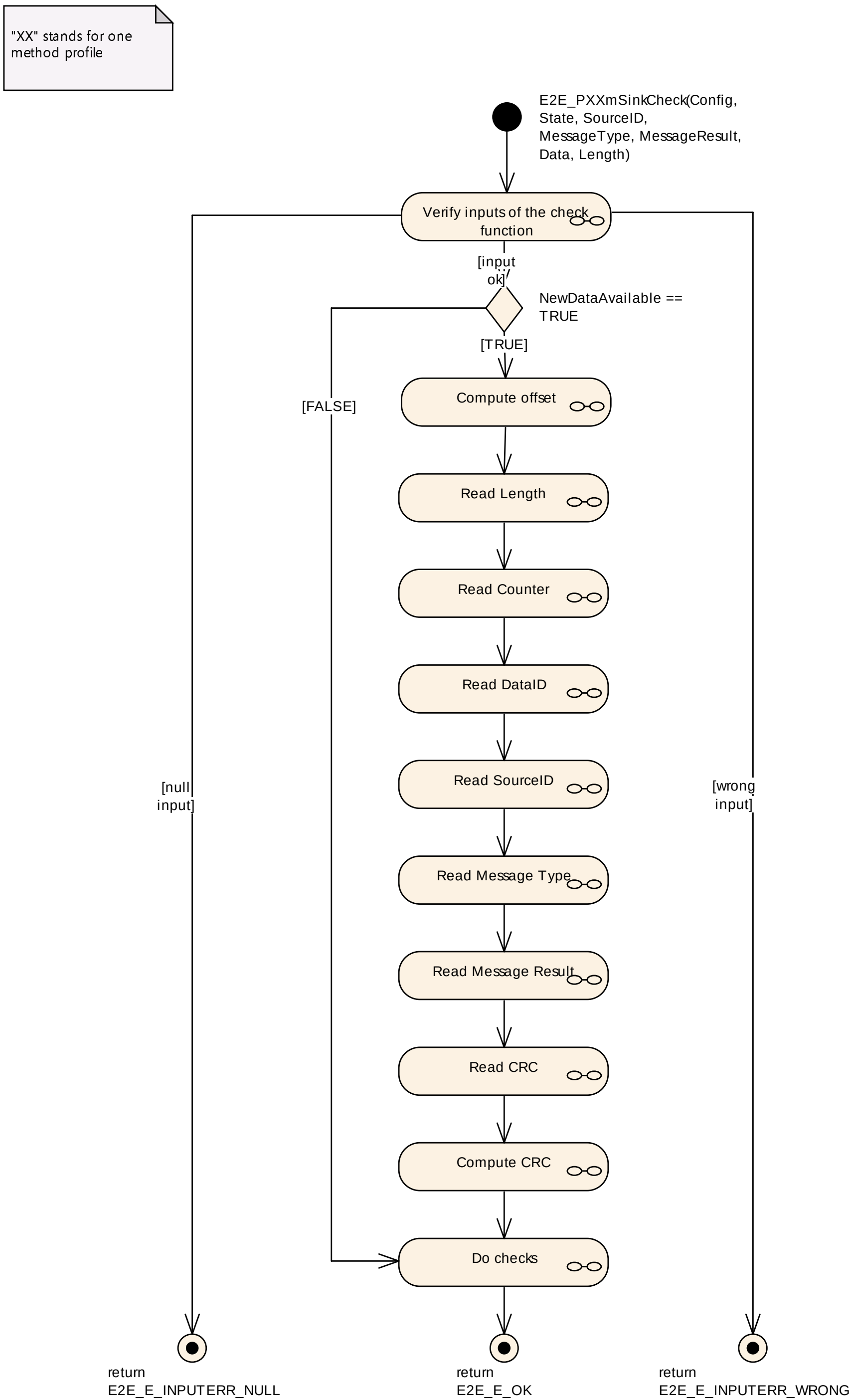

Figure 6.48: Behavior of E2E_PXXmSinkCheck()

Following steps are the same as for the client side and thus already described previously

- ”Read Length” see [PRS_E2E_01187]
- ”Read Counter” see [PRS_E2E_01188]
- ”Read DataID” see [PRS_E2E_01189]
- ”Read Message Type” see [PRS_E2E_01191]
- ”Read Message Result” see [PRS_E2E_01192]
- ”Read CRC” see [PRS_E2E_01193]
- ”Increment Counter” see [PRS_E2E_01180]

[PRS_E2E_01196] ⌈The step ”Verify inputs of the check function” in E2E_PXXmSinkCheck() shall behave as shown in Figure 6.49.⌋*(RS_E2E_08539)*

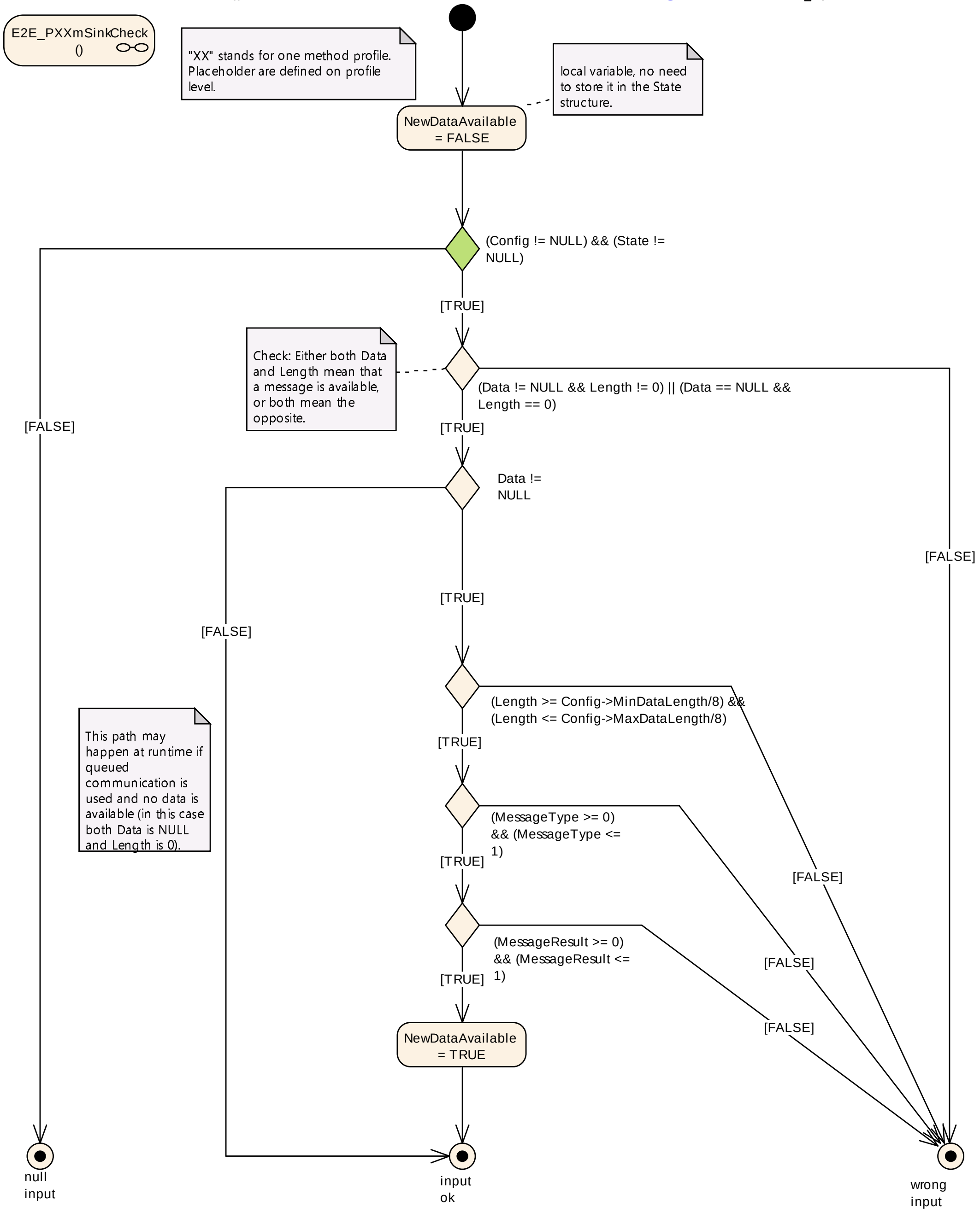

Figure 6.49: E2E_PXXmSinkCheck() step ’Verify inputs of the check function”

[PRS_E2E_01197] ⌈The step ”Read SourceID” in E2E_PXXmSinkCheck() shall behave as shown in Figure 6.50.⌋*(RS_E2E_08539)*

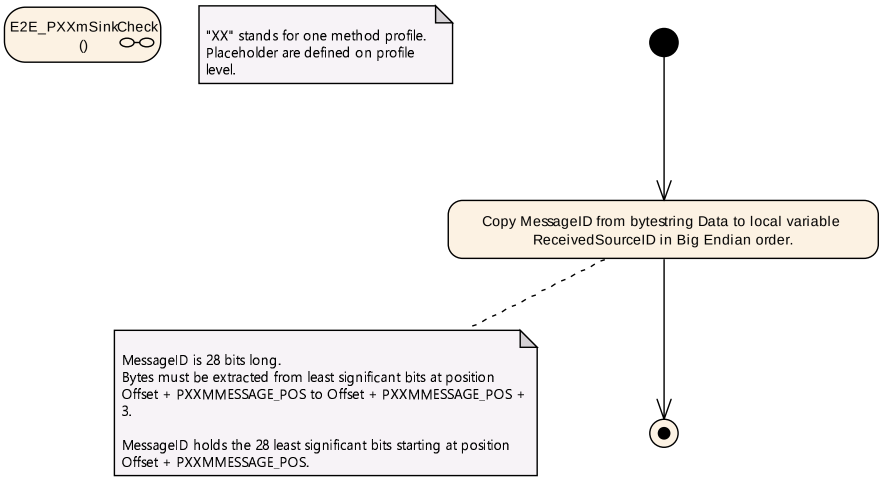

Figure 6.50: E2E_PXXmSinkCheck() step ”Read SourceID”

[PRS_E2E_01198] ⌈The step ”Do Checks” in E2E_PXXmSinkCheck() shall behave as shown in Figure 6.51.⌋*(RS_E2E_08539)*

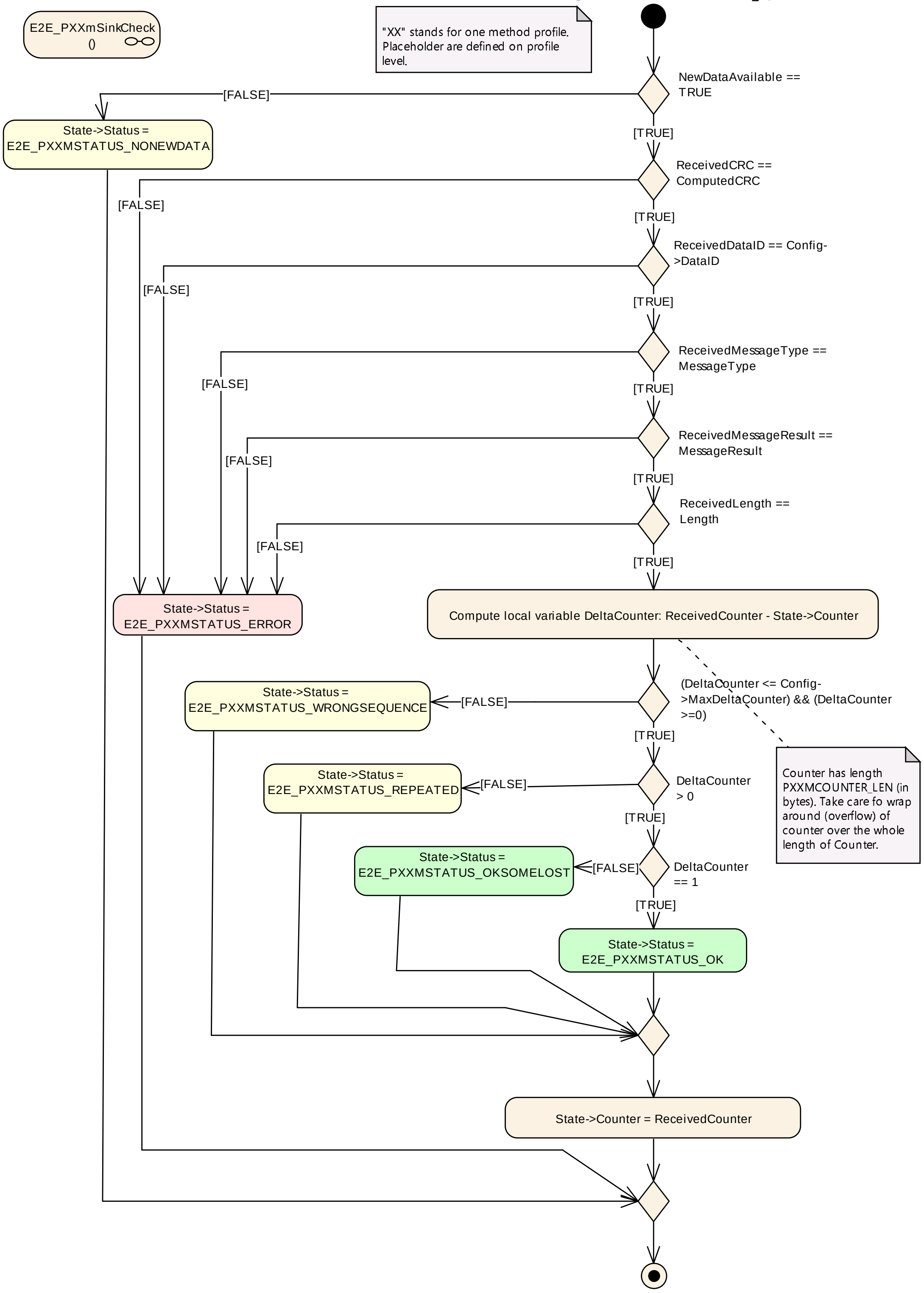

Figure 6.51: E2E_PXXmSinkCheck() step ”Do Checks”

6.4.11 Profile Data Types

6.4.11.1 Profile XXm Protect State Type

[PRS_E2E_01199] ⌈The E2E_PXXmProtect() and E2E_PXXmForward() functions' "state" shall have the members defined in Table 6.9.⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter to be used for protecting the next Data. The initial value is 0, which means that in the first cycle, counter is 0. Each time E2E_PXXmProtect() is called, it increments the counter up to its maximum value (0xFF'FF for a 16 bit counter, 0xFF'FF'FF'FF for a 32 bit counter, 0xFF'FF'FF'FF' FF'FF'FF'FF for a 64 bit counter). After the maximum value is reached, the next value is 0x0. The overflow is not reported to the caller.

Table 6.9: E2E Profile XXm Protect State Type

6.4.11.2 Profile XXm Check State Type

[PRS_E2E_01200] ⌈The E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() functions' "state" shall have the members defined in Table 6.10.⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter of the data in previous cycle.
Status	Enumeration	Result of the verification of the Data in this cycle, determined by the Check function.

Table 6.10: E2E Profile XXm Check State Type

6.4.11.3 Profile XXm Check Status Enumeration

[PRS_E2E_01201] ⌈The step "Do Checks" in E2E_PXXmSourceCheck() and E2E_PXXmSinkCheck() shall set State->Status to one of the following enumeration values (see Table 6.11).⌋*(RS_E2E_08528)*

Name	State Type	Description
E2E_PXXMSTATUS_OK	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented by 1).

E2E_PXXMSTATUS_NONEWDATA	Error	The Check function has been invoked but no new Data is 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_PXXMSTATUS_REPEATED.
E2E_PXXMSTATUS_ERROR	Error	Error not related to counters occurred (e.g. wrong CRC, wrong Length, wrong Options, wrong Data ID).
E2E_PXXMSTATUS_REPEATED	Error	The checks of the Data in this cycle were successful, except for the repetition.
E2E_PXXMSTATUS_OKSOMELOST	OK	The checks of the Data in this cycle were successful (including counter check, which was incremented within the allowed configured delta).
E2E_PXXMSTATUS_WRONGSEQUENCE	Error	The checks of the Data in this cycle were successful, except for a counter jump, which changed more than the allowed delta

Table 6.11: E2E Profile XXm Check Status Enumeration

6.5 Specification of E2E Profile 1

Profile 1 is a Legacy Profile and is only maintained for compatibility reasons. New Projects shall use Profile 11.

[PRS_E2E_00218] ⌈Profile 1 shall provide the following mechanisms: Counter, Timeout monitoring, Data ID, CRC (see Table 6.12).⌋*(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548)*

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 the first 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 6.12: 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 6.13: Detectable communication faults using Profile 1

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

6.5.1 Header 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 a message.

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

6.5.1.1 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)*

[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.

6.5.1.2 Data ID

The unique Data IDs are used 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.

6.5.1.3 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_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)*

Note: To calculate a CRC with CRCLib function Crc_CalculateCRC8() with start value = 0x00 this function must be called with third parameter (Crc_StartValue8) equal to the complement of the intended start value. For start value = 0x00 this parameter must be equal to 0xFF.

See also table 6.1 (E2E Profile 1 mechanisms) and figure 6.7 (Subdiagram 'getDataID-CRC', used by E2E_P01Protect() and E2E_P01Check()).

For details of CRC calculation, the usage of start values and XOR values see also SWS_CRCLibrary[3].

[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.

Figures 6.2 and 6.3 show how the CRC for signal based communication would be calculated. Figure 6.2 uses the following configuration

1. CRC is the 0th byte in message (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. unusedBitPattern = 0xFF.

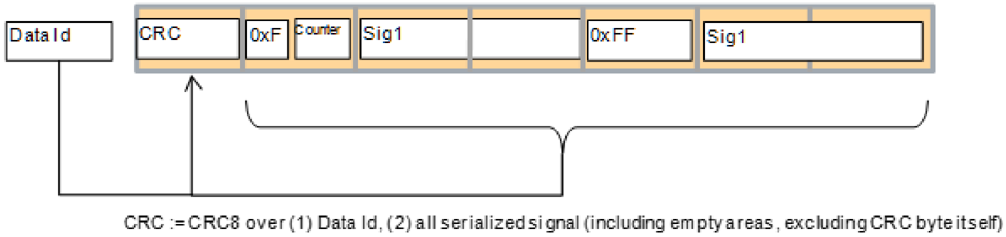

Figure 6.52: E2E Profile 1 variant 1A CRC calculation example

Figure 6.3 uses the following configuration

1. CRC is the 0th byte in the message (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. unusedBitPattern = 0xFF.

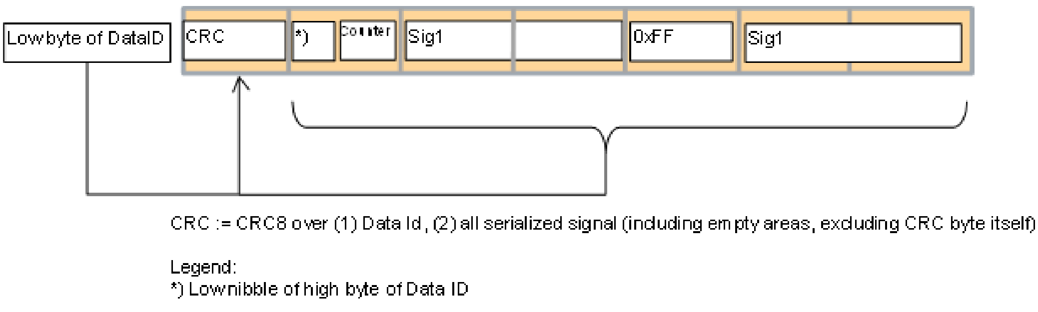

Figure 6.53: E2E Profile 1 variant 1C CRC calculation example

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_08531)*

[PRS_E2E_00640] ⌈If DataIDMode is set to E2E_P01_DATAID_NIBBLE, the CRC calculation shall be done by first calculating over the low byte of the Data ID, then a zero-byte (0x00), and then the user data.⌋*(RS_E2E_08539)*

6.5.2 Creation of E2E-Header

6.5.2.1 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
4. write CRC in Data
5. increment the Counter (which will be used in the next invocation of E2E_P01Protect()),as specified by Figure 6.54 and Figure 6.55

⌋*(RS_E2E_08528)*

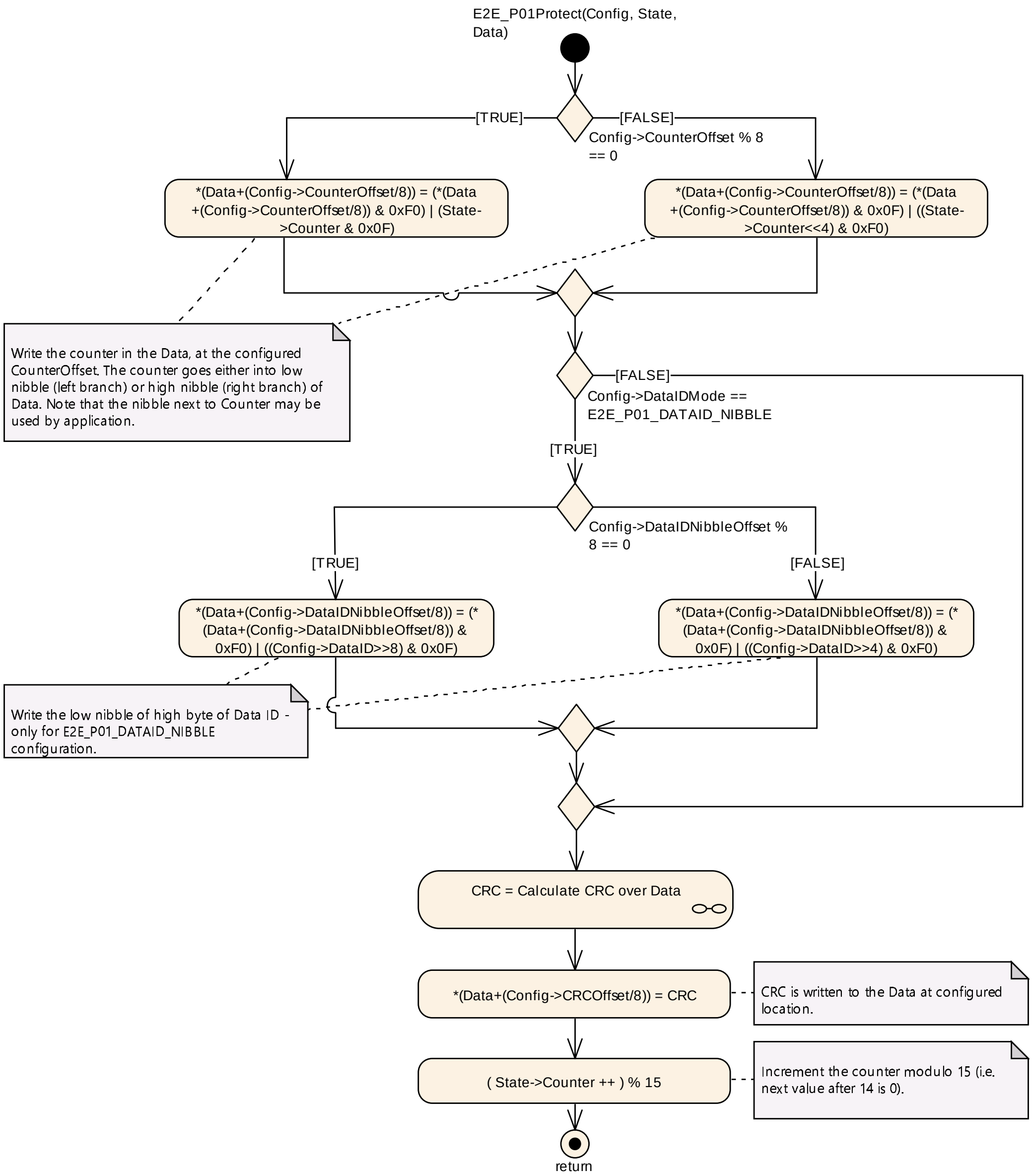

Figure 6.54: E2E_P01Protect()

6.5.2.2 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:

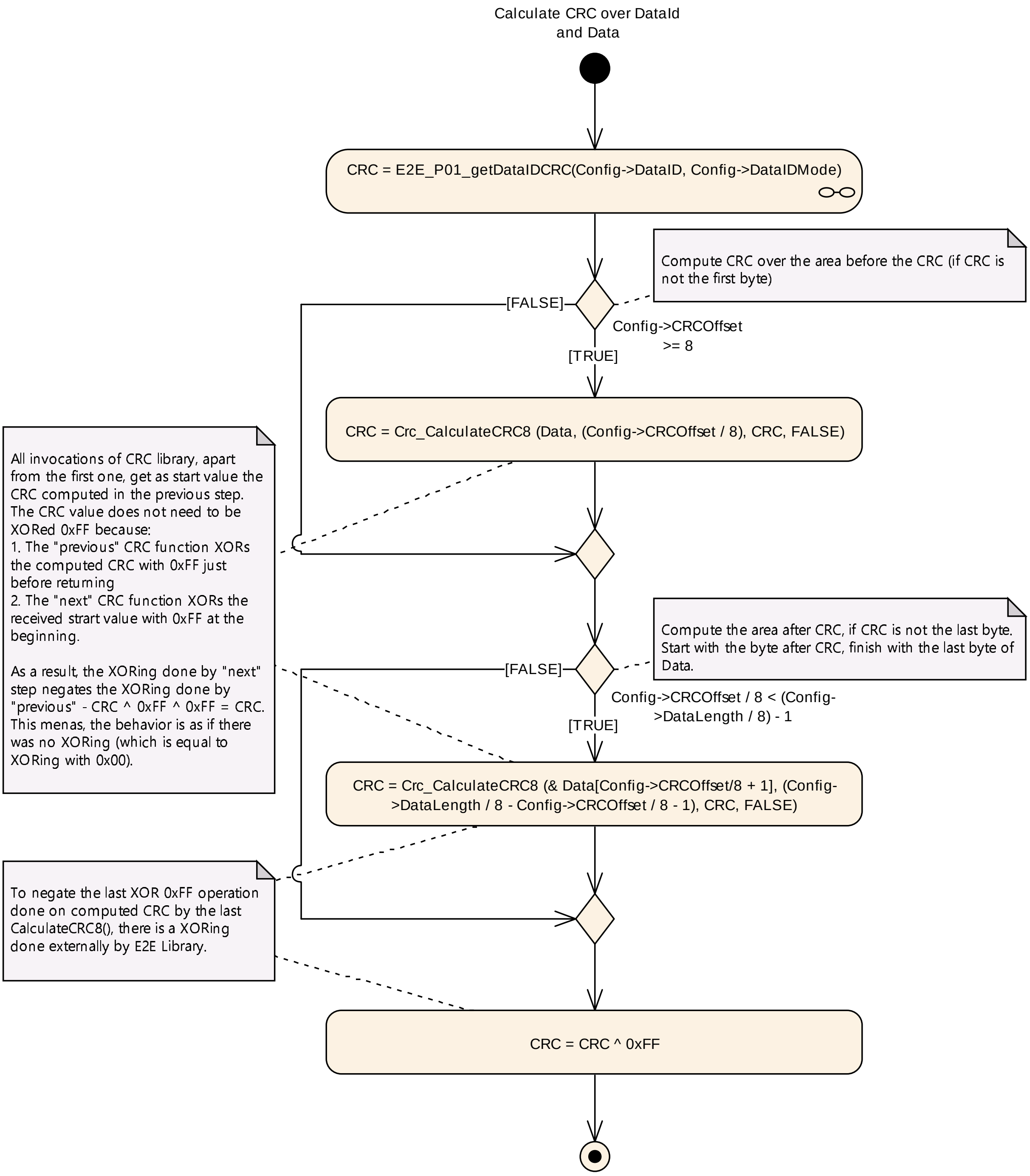

Figure 6.55: Subdiagram „Calculate CRC over Data ID and Data", used by E2E_P01Protect(), E2E_P01Forward() and E2E_P01Check()

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 6.56.

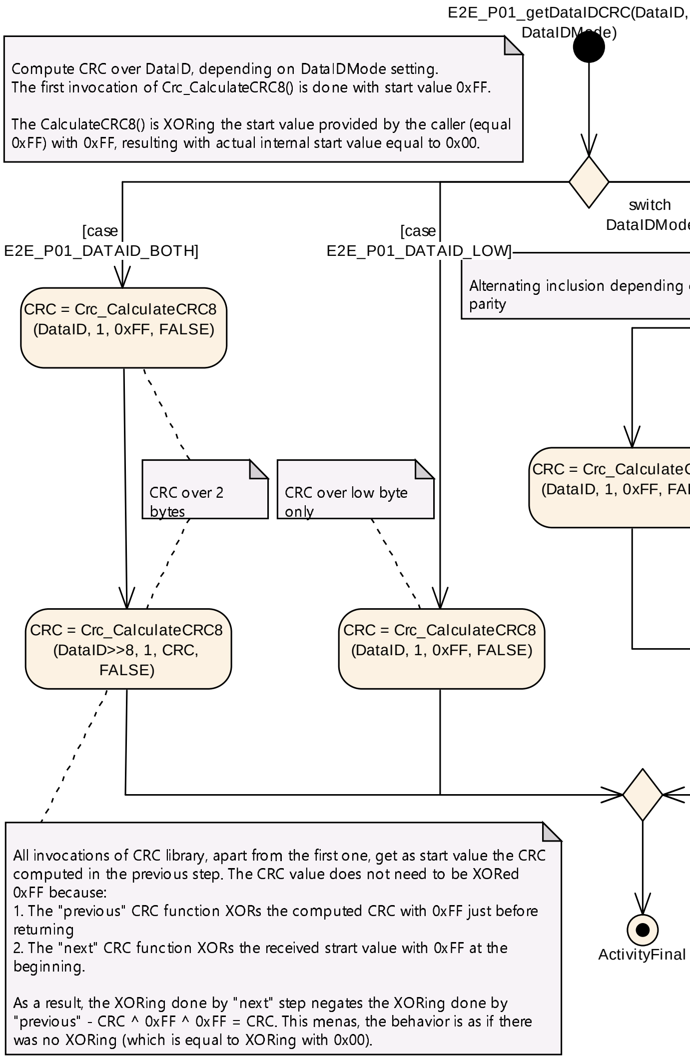

Figure 6.56: Subdiagram "getDataIDCRC", used by E2E_P01Protect() and E2E_P01Check()

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_P0-2-1Check() 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 6.55 in >=R4.0 and R3.2. This means Figure 6.55 is different in >=R4.0 and R3.2.

6.5.2.3 E2E_P01Forward

[PRS_E2E_00608] ⌈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] ⌈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 6.57 and Figure 6.55

⌋*(RS_E2E_08528)*

[PRS_E2E_00610] ⌈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 6.57 and Figure 6.55

⌋*(RS_E2E_08528)*

[PRS_E2E_00611] ⌈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 6.57 and Figure 6.55

⌋(*RS_E2E_08528*)

[PRS_E2E_00612] ⌈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 6.57 and Figure 6.55

⌋(*RS_E2E_08528*)

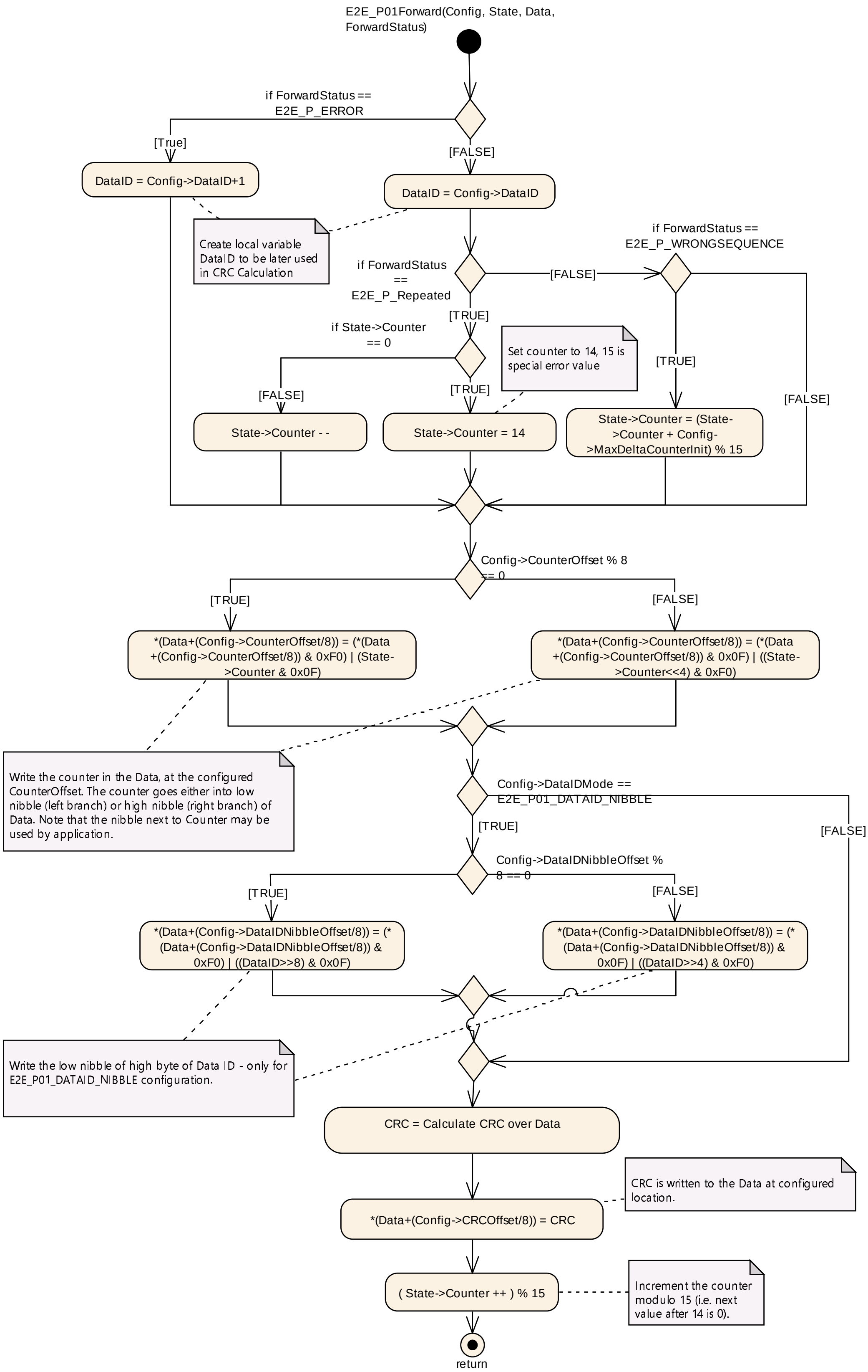

Figure 6.57: E2E_P01Forward()

6.5.3 Evaluation of E2E- Header

6.5.3.1 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 6.58 and Figure 6.55.

⌋*(RS_E2E_08528, RS_E2E_08530)*

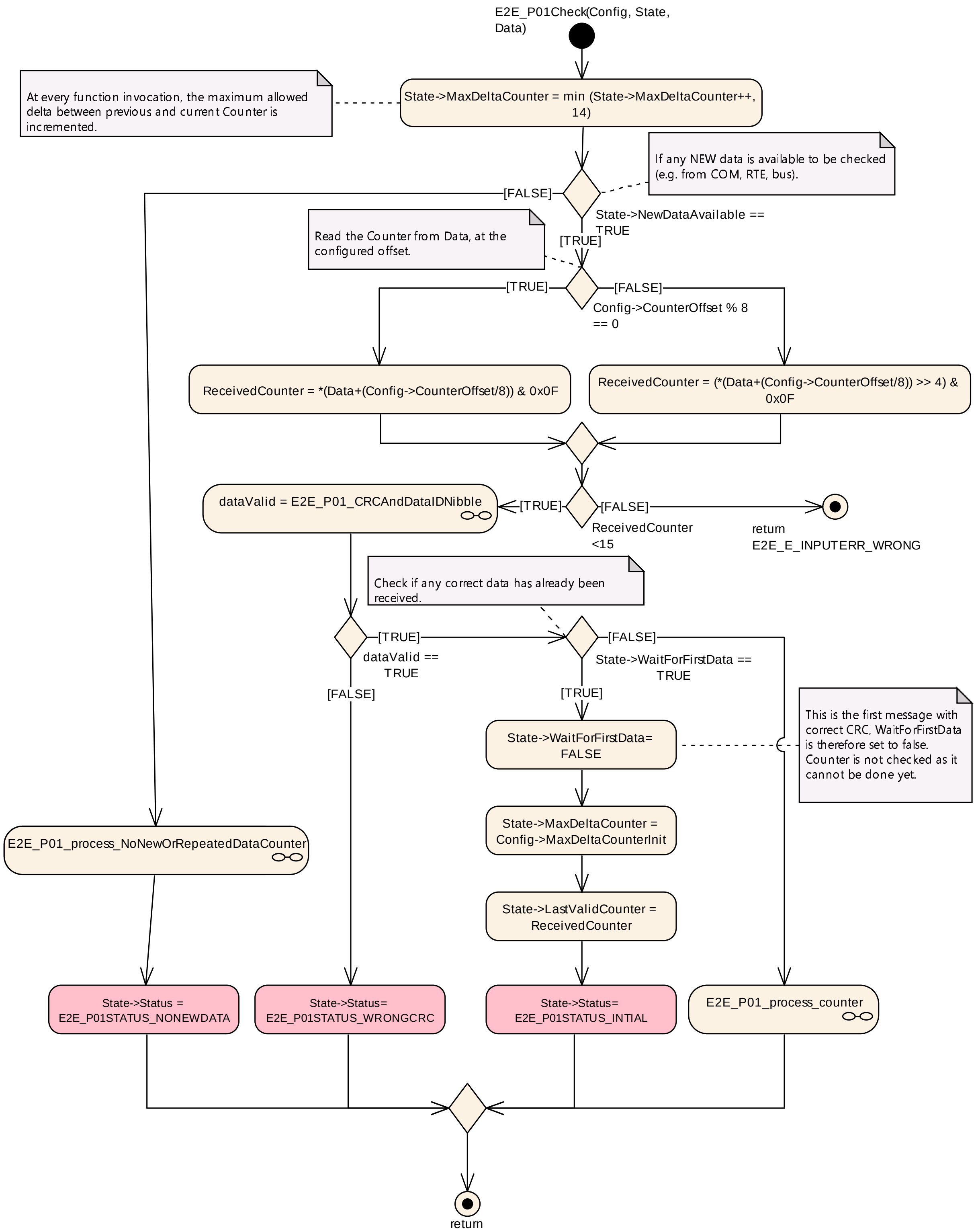

Figure 6.58: E2E_P01Check()

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 6.55. The subroutines of Figure 6.59 are described in Figure 6.55 and Figure 6.56

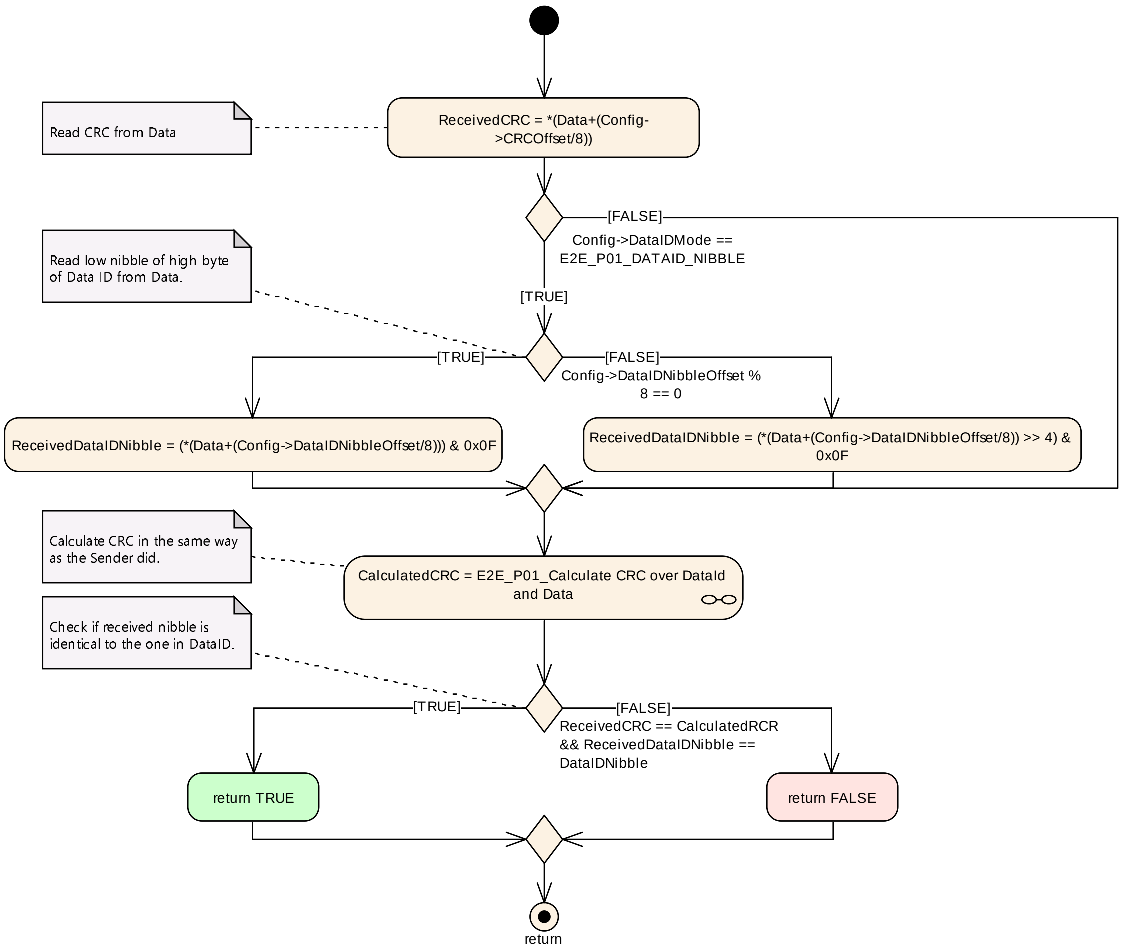

Figure 6.59: E2E Profile Check step "E2E_P01_CRCAndDataIDNibble"

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 6.60.

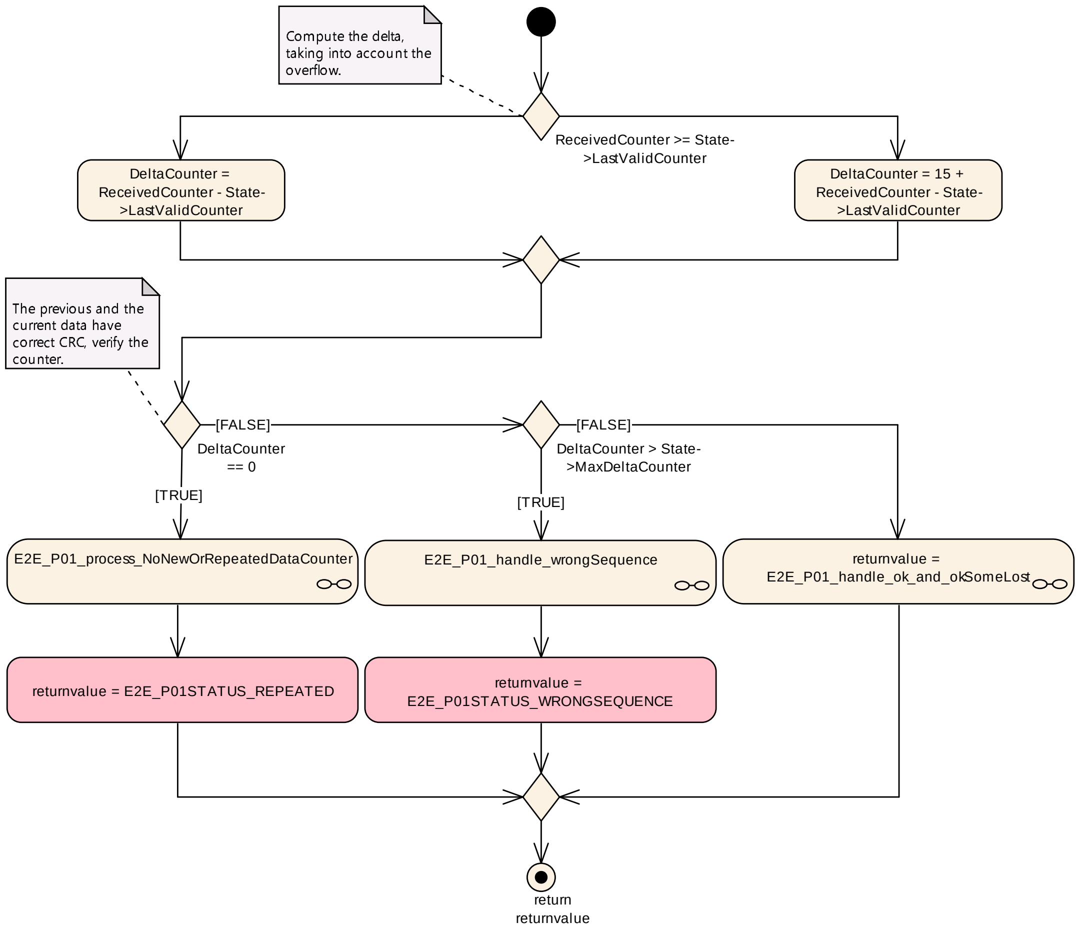

Figure 6.60: E2E Profile Check step "E2E_P01_process_counter"

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 6.61.

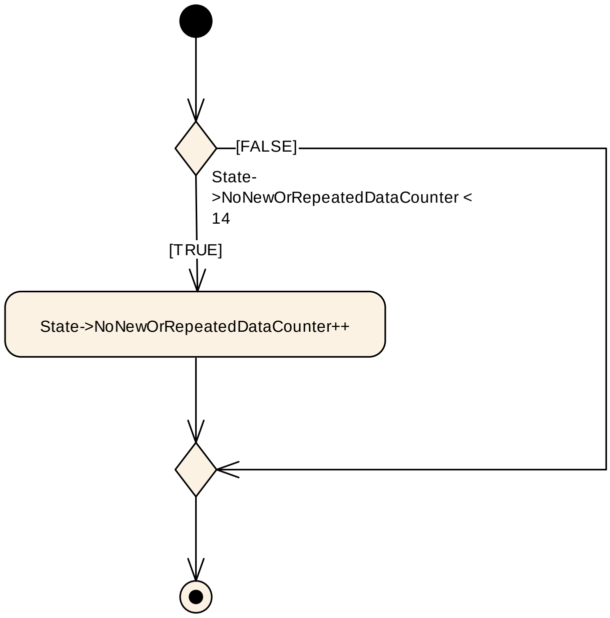

Figure 6.61: E2E Profile Check step "E2E_P01_process_NoNewOrRepeatedDataCounter"

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 6.62.

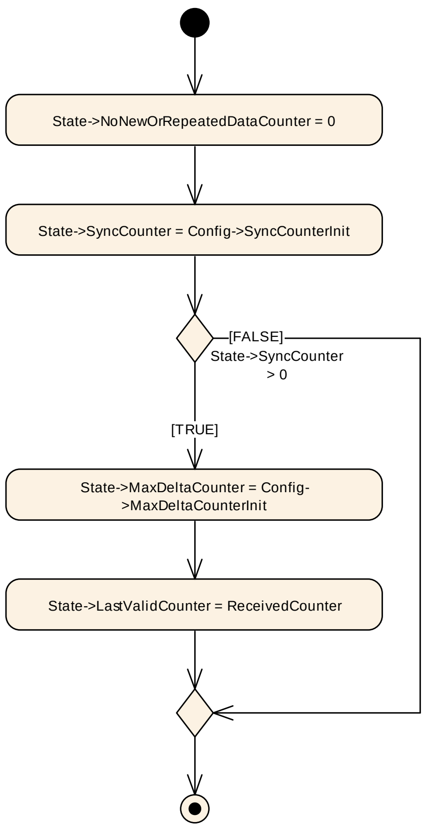

Figure 6.62: E2E Profile Check step "E2E_P01_handle_wrongSequence"

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 6.63.

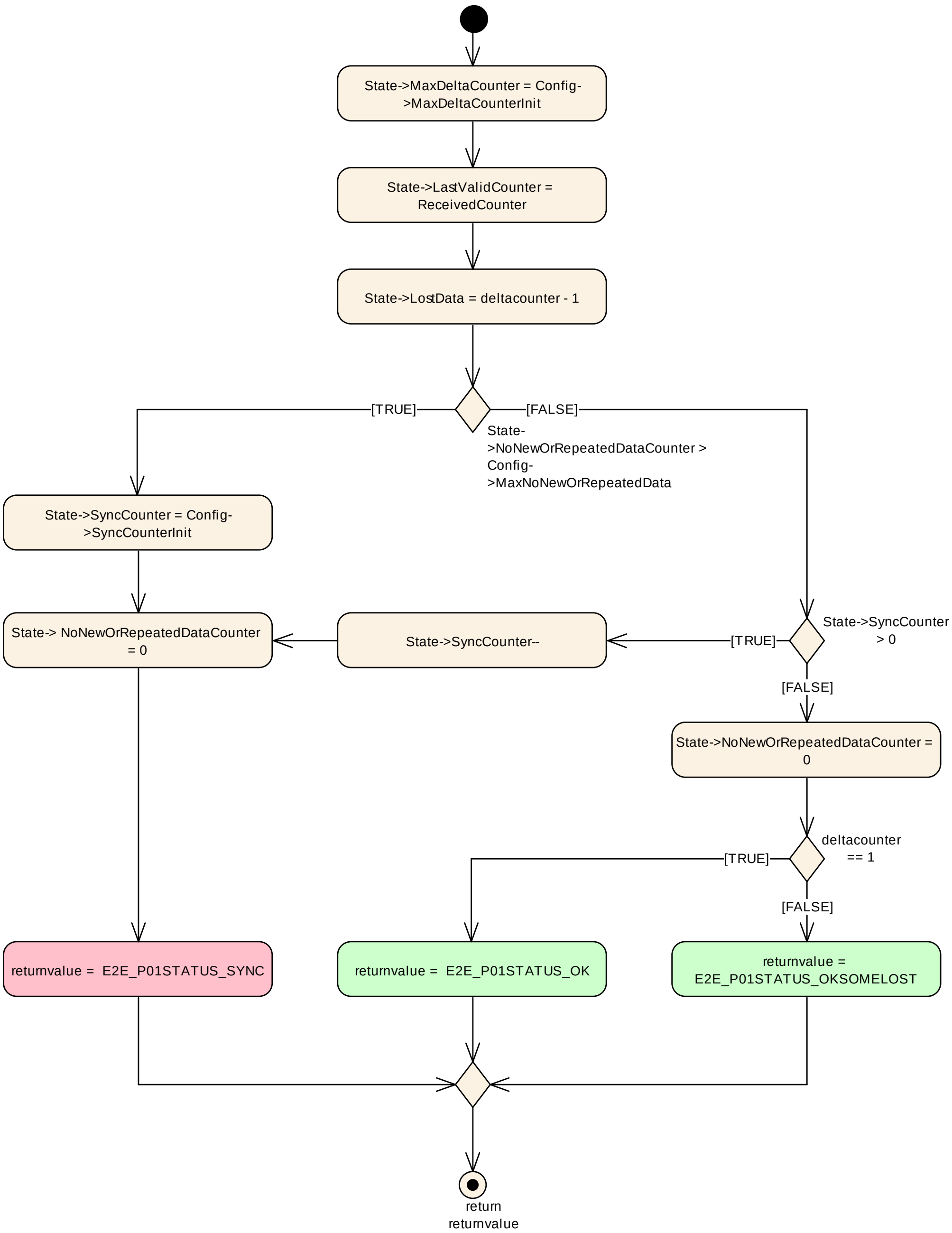

Figure 6.63: E2E Profile Check step ”E2E_P01_handle_ok_and_okSomeLost”

6.5.4 Profile Data Types

6.5.4.1 Profile 1 Protect State Type

[PRS_E2E_00644] ⌈The E2E_P01Protect and E2E_P01Forward functions 'state' shall have the members defined in Table 6.14.⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter to be used for protecting the next Data. The initial value is 0, which means that the first Data will have the counter 0. After the protection by the Counter, the Counter is incremented modulo 0xF. The value 0xF is skipped (after 0xE the next is 0x0), as 0xF value represents the error value. The four high bits are always 0.

Table 6.14: Profile 1 Protect State Type

6.5.4.2 Profile 1 Check Status Type

[PRS_E2E_00645] ⌈The E2E_P01Check functions 'State' shall have the members defined in (see Table 6.15).⌋*(RS_E2E_08528)*

Member Name	Type	Description
LastValidCounter	Unsigned Integer	Counter value most recently received. If no data has been yet received, then the value is 0x0. After each reception, the counter is updated with the value received.
MaxDeltaCounter	Unsigned Integer	MaxDeltaCounter specifies the maximum allowed difference between two counter values of consecutively received valid messages.
WaitForFirstData	Boolean	If true, that means no correct data (with correct Data ID and CRC) has been yet received after the receiver initialization or reinitialization.
NewDataAvailable	Boolean	Indicates that new data is available to be checked. This attribute has to be set by the caller of the E2E_P01Check function.
LostData	Unsigned Integer	Number of data (messages) lost since reception of last valid one. This attribute is set only if Status equals E2E_P01STATUS_OK or E2E_P01STATUS_OKSOMELOST. For other values of Status, the value of LostData is undefined.
Status	Enumeration	Result of the verification of the Data, determined by the Check function.

SyncCounter	Unsigned Integer	Number of Data required for validating the consistency of the counter that must be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
NoNewOrRepeatedData	Unsigned Integer	Amount of consecutive reception cycles in which either (1) there was no new data, or (2) when the data was repeated.

Table 6.15: E2E Profile 1 Check Status Type Members

6.5.4.3 Profile 1 Check Status Enumeration

[PRS_E2E_00588] ⌈The E2E_P01Check function 'State->Status' enumeration type shall have the following enumeration values (see Table 6.16).⌋*(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 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 6.16: E2E Profile 1 Check Status Enumeration

6.5.4.4 Profile 1 Configuration Type

[PRS_E2E_00646] ⌈The E2E_P01Protect, E2E_P01Forward and E2E_P01Check functions 'Config' shall have the following members defined in (see Table 6.17).⌋*(RS_E2E_08528, RS_E2E_08537)*

MemberName	Type	Description
CounterOffset	Unsigned Integer	Bit offset of Counter in MSB first order. CounterOffset shall be a multiple of 4. In variants 1A, 1B, and 1C, CounterOffset is 8.
CRCOffset	Unsigned Integer	Bit offset of CRC (i.e. since *Data) in MSB first order. The offset shall be a multiple of 8. In variants 1A, 1B, and 1C, CRCOffset is 0.

DataID	Unsigned Integer	A unique identifier, for protection against masquerading. There are some constraints on the selection of ID values, described in section "Configuration constraints on Data IDs".
DataIDNibbleOffset	Unsigned Integer	Bit offset of the low nibble of the high byte of Data ID.
DataIDMode	Enumeration	Inclusion mode of ID in CRC computation (both bytes, alternating, or low byte only of ID included).
DataLength	Unsigned Integer	Length of data, in bits. The value shall be a multiple of 8 and shall be <= 256.
MaxDeltaCounterInit	Unsigned Integer	Initial maximum allowed gap between two counter values of two consecutively received valid Data. For example, if the receiver gets Data with counter 1 and MaxDeltaCounterInit is 1, then at the next reception the receiver can accept Counters with values 2 and 3, but not 4. Note that if the receiver does not receive new Data at a consecutive read, then the receiver increments the tolerance by 1.
MaxNoNewOrRepeatedData	Unsigned Integer	The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions.
SyncCounterInit	Unsigned Integer	Number of Data required for validating the consistency of the counter that must be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.

Table 6.17: E2E Profile 1 Configuration Type

6.5.5 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 6.18: E2E Profile 1 protocol example configuration

E2E_P01ProtectStateType field	Value
Counter	0

Table 6.19: 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 6.20: 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 6.21: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_BOTH, counter 1

6.5.5.1 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
0xCE	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.22: 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
0x02	0x01	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.23: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_ALT, counter 1

6.5.5.2 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
0xCE	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.24: 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
0x93	0x01	0x00	0x00	0x00	0x00	0x00	0x00

Table 6.25: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_LOW, counter 1

6.5.5.3 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 6.26: 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 6.27: E2E Profile 1 protect result DataIDMode = E2E_P01_DATAID_NIBBLE, counter 1

6.6 Specification of E2E Profile 2

Profile 2 is a Legacy Profile and is only maintained for compatibility reasons. New Projects shall use Profile 22.

[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 6.28).⌋*(RS_E2E_08527, RS_E2E_08529, RS_E2E_-*

08530, RS_E2E_08533, RS_E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548)

Mechanism	Description
Counter (Sequence Number)	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
Data ID (Message Key used for CRC calculation)	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(Safety Code))	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.
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_P02CheckStatusType.

Table 6.28: 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 6.28):

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 basis and timeout monitoring using E2E-Library [3]	Loss, delay, blocking

[3]Implementation by sender and receiver

Data ID + CRC	Masquerade and incorrect addressing, insertion
CRC	Corruption, Asymmetric information [4]

Table 6.29: Detectable communication faults using Profile 2

6.6.1 Header Layout

[PRS_E2E_00121] ⌈In E2E Profile 2, the layout of the data buffer (Data) shall be as depicted in Figure 6.64, with a maximum length of 256 bytes (i.e. N=255)⌋*(RS_E2E_08528)*

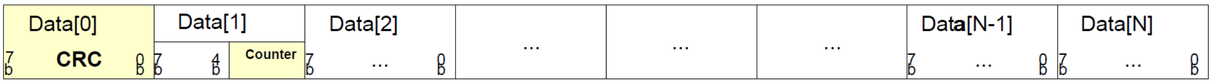

Figure 6.64: E2E Profile 2 data buffer layout

6.6.1.1 Counter

[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_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)*

6.6.1.2 DataID

[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

[4]for a set of data protected by same CRC

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, it is possible that 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.

6.6.1.3 CRC

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

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

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

6.6.2 Creation of E2E-Header

[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)*

6.6.2.1 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 6.28. E2E_P02Protect() therefore calculates the Counter and then the CRC and puts both into the data buffer (Data). A flow chart with the visual description of the function E2E_P02Protect() is depicted in Figure 6.65 and Figure 6.66.

[PRS_E2E_00126] ⌈In E2E Profile 2, the E2E_P02Protect() function shall perform the activities as specified in Figure 6.65 and Figure 6.66.⌋*(RS_E2E_08528, RS_E2E_08531)*

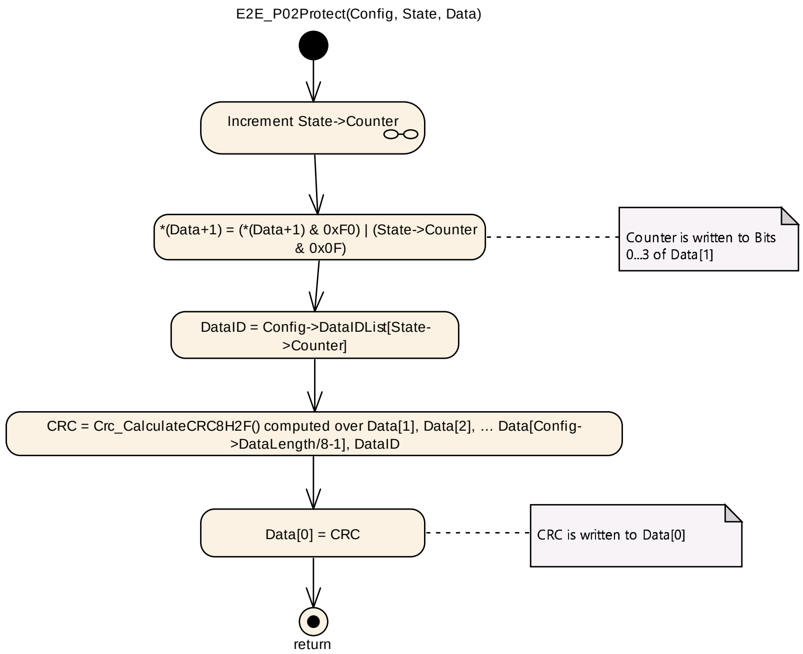

Figure 6.65: E2E_P02Protect()

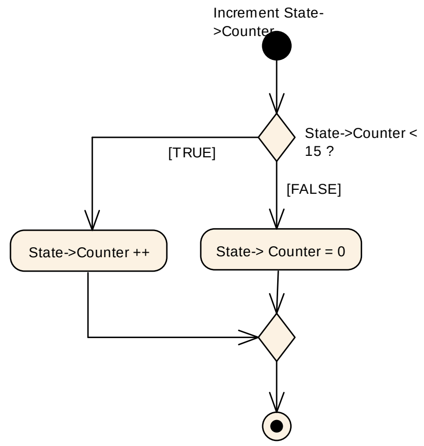

Figure 6.66: Increment Counter

[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_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.

6.6.2.2 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 a 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 6.67 and Figure 6.68.

[PRS_E2E_00613] ⌈In E2E Profile 2, the E2E_P02Forward() function shall perform the activities as specified in Figure 6.67 and Figure 6.66.⌋*(RS_E2E_08528, RS_E2E_-08531)*

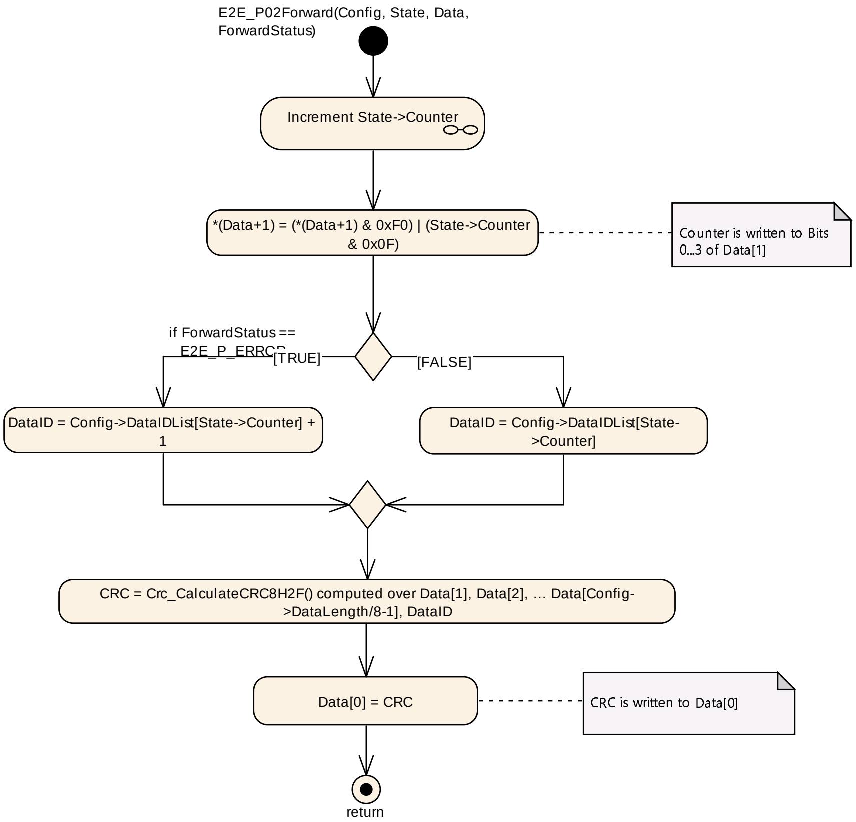

Figure 6.67: E2E_P02Forward()

[PRS_E2E_00614] ⌈In E2E Profile 2, the E2E_P02Forward() function shall increment the Counter according to Figure 6.68.⌋*(RS_E2E_08528)*

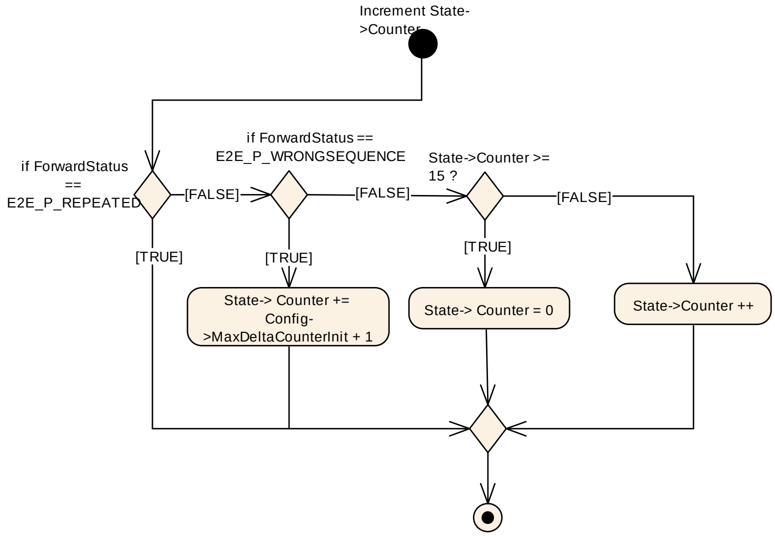

Figure 6.68: Increment Counter

6.6.3 Evaluation of the E2E-Check

[PRS_E2E_00125] ⌈In E2E Profile 2, the E2E_P02Check() function shall not modify any bit in Data.⌋*(RS_E2E_08528, RS_E2E_08531)*

6.6.3.1 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 6.69, Figure 6.70 and Figure 6.71.

[PRS_E2E_00134] ⌈In E2E Profile 2, the E2E_P02Check() function shall perform the activities as specified in Figure 6.69, Figure 6.70 and Figure 6.71.⌋*(RS_E2E_08528, RS_E2E_08531)*

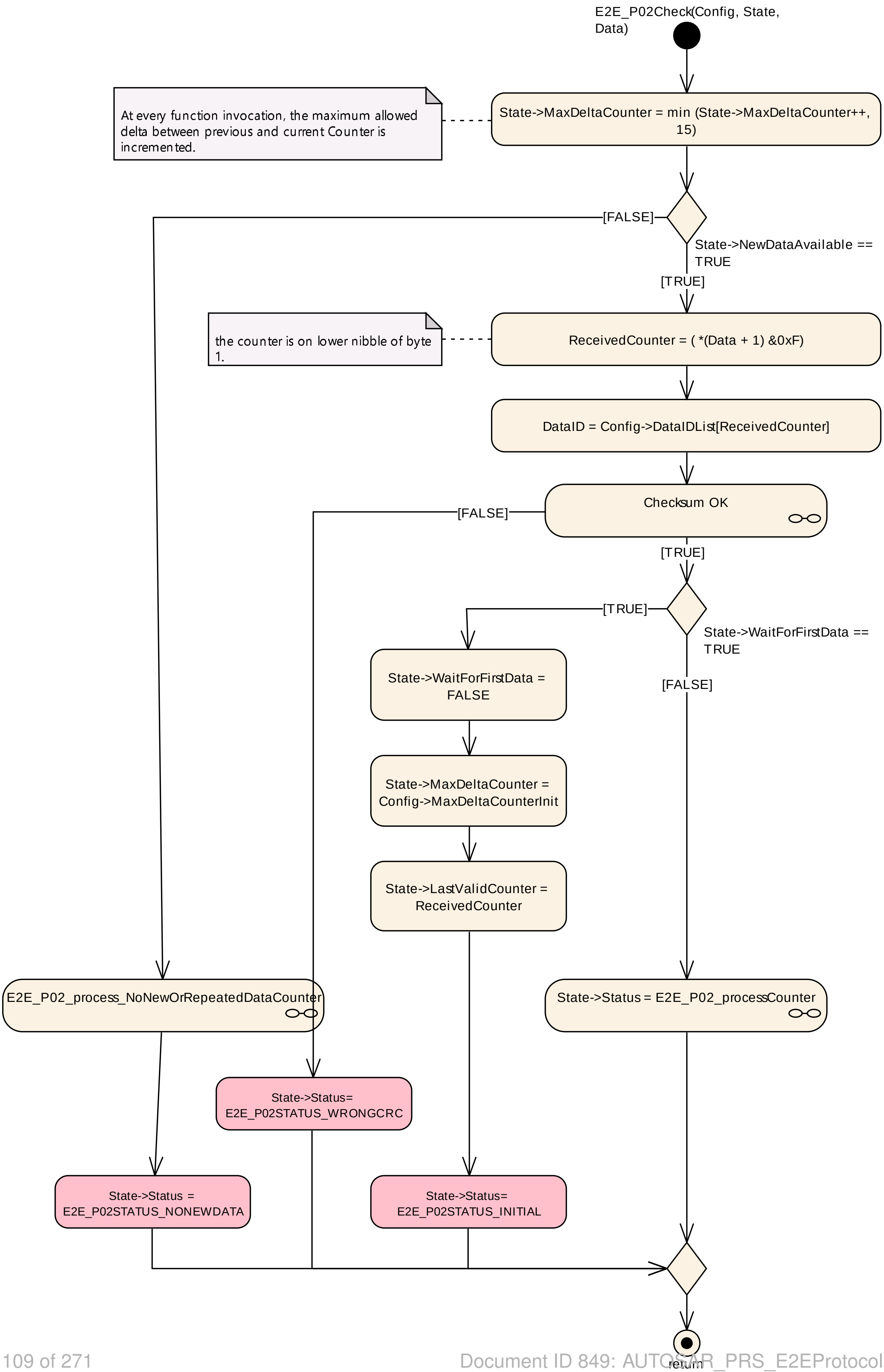

Figure 6.69: E2E_P02Check

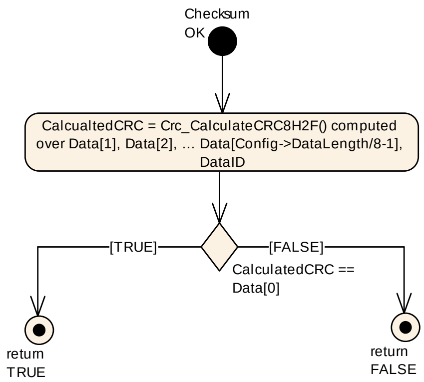

Figure 6.70: Checksum OK

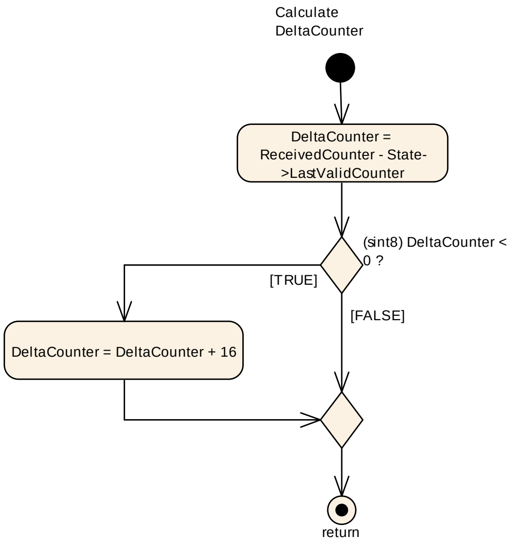

Figure 6.71: Calculate Delta Counter

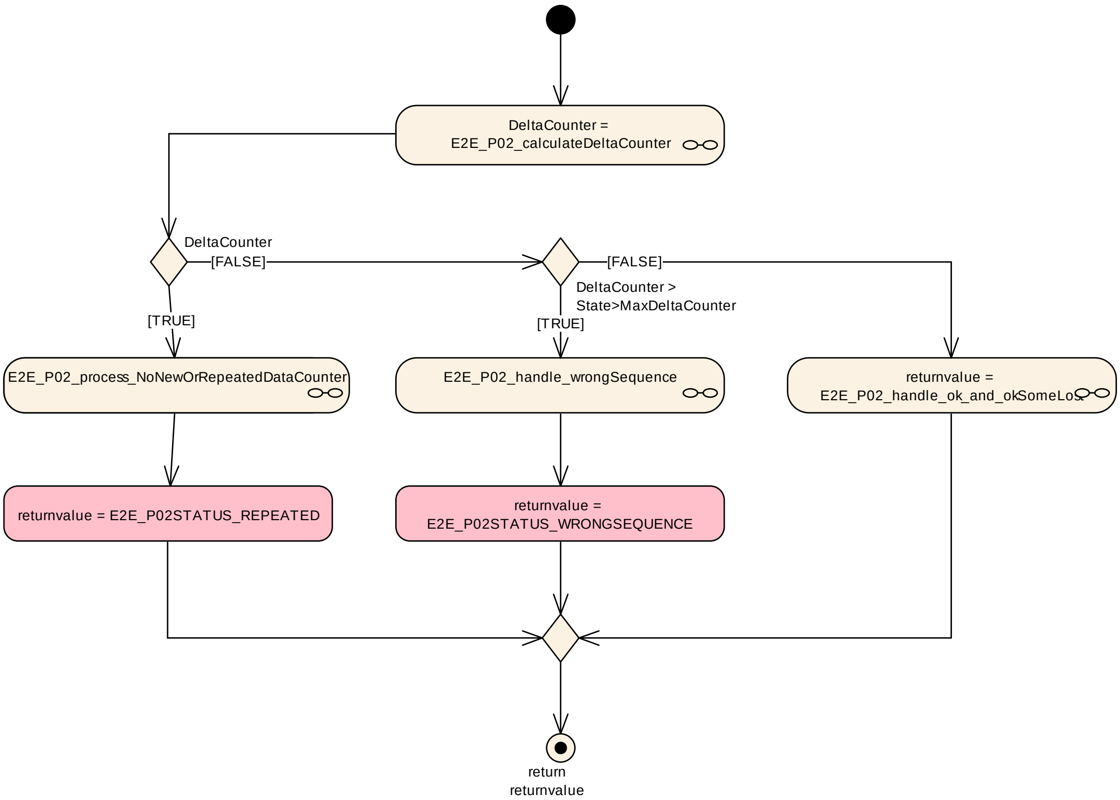

Figure 6.72: E2E Profile Check step "E2E_P02_process_counter"

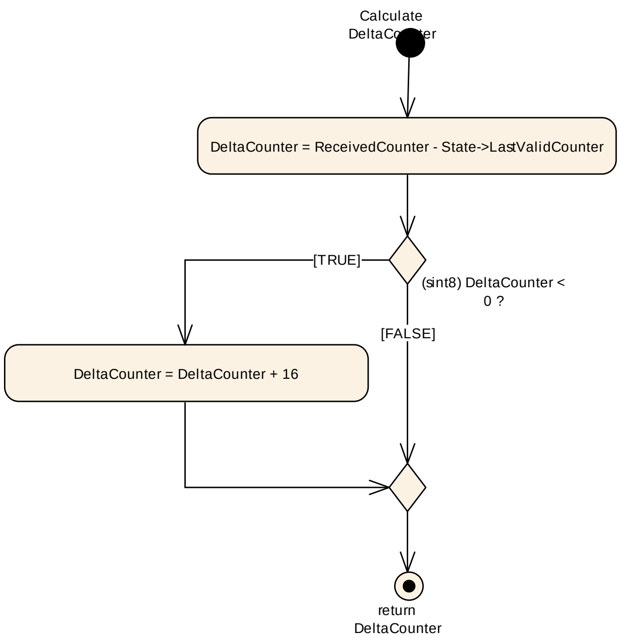

Figure 6.73: E2E Profile Check step "E2E_P02_process_counter"

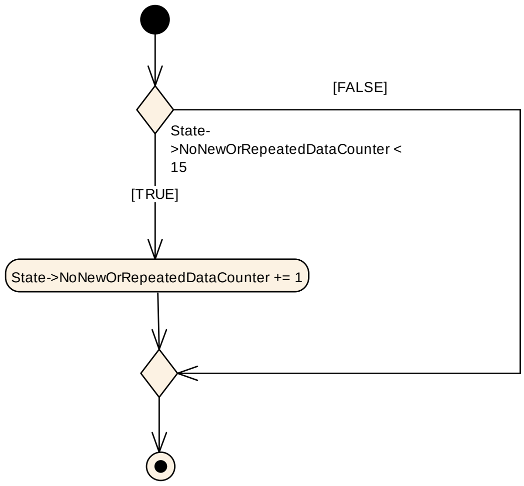

Figure 6.74: E2E Profile Check step ”E2E_P02_process_NoNewOrRepeatedDataCounter”

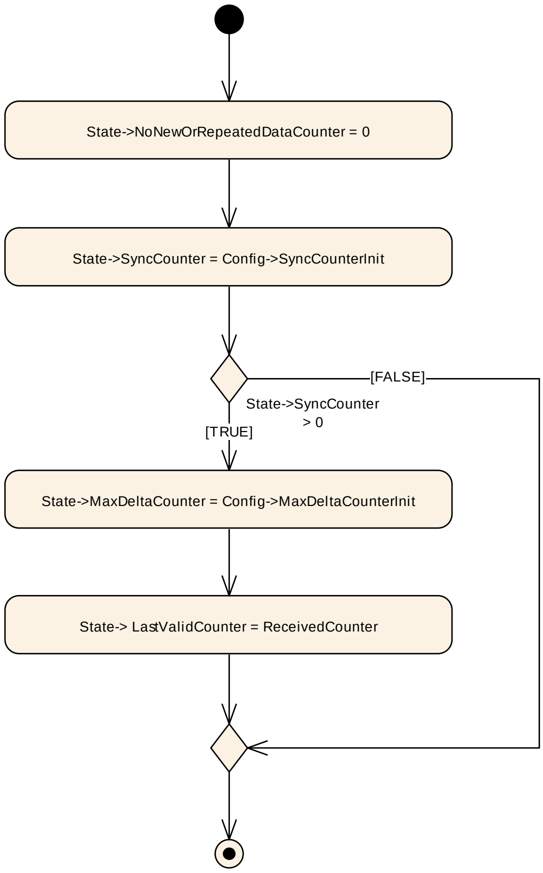

Figure 6.75: E2E Profile Check step ”E2E_P02_handle_wrongSequence”

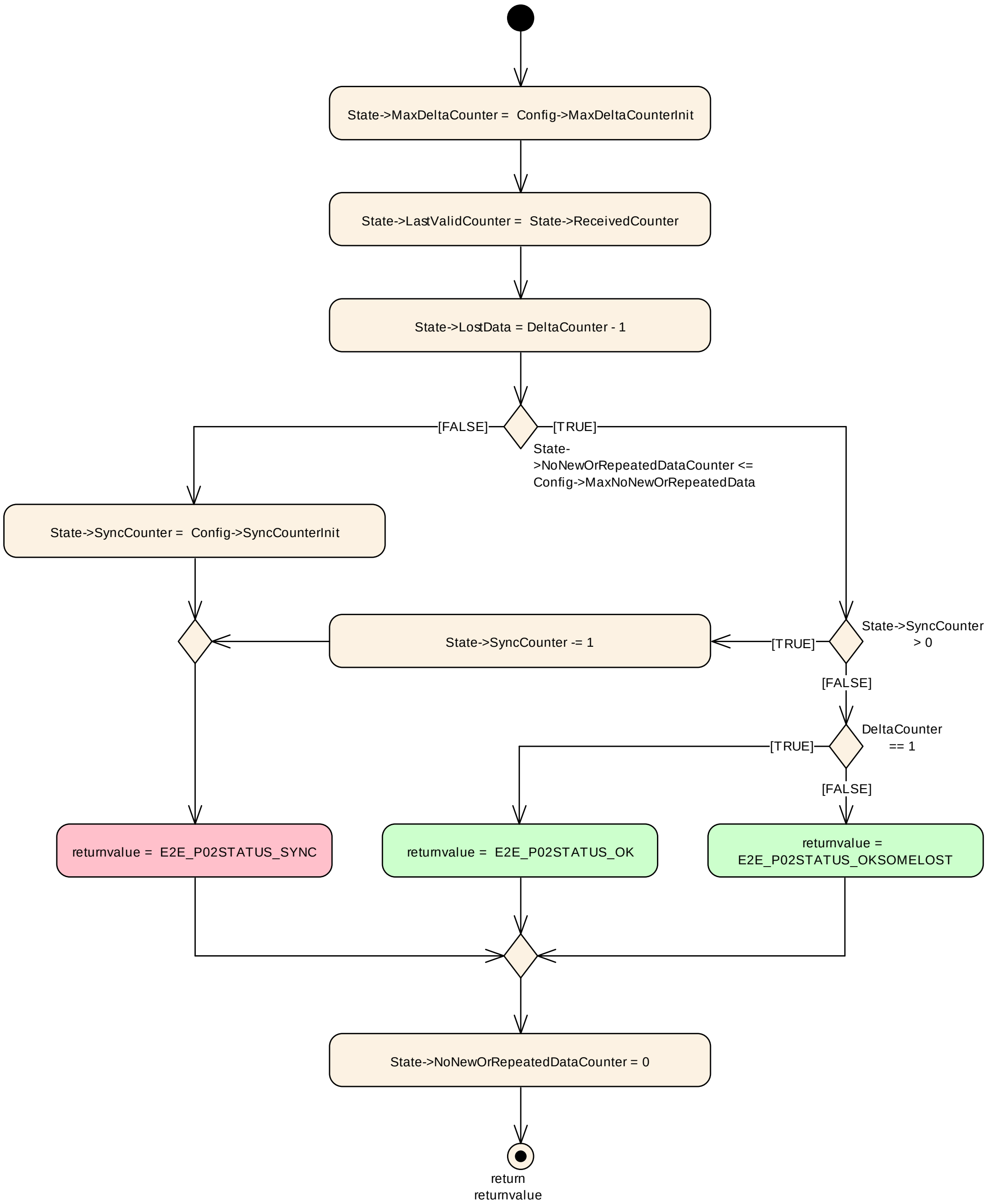

Figure 6.76: E2E Profile Check step "E2E_P02_handle_ok_and_okSomeLost"

First, the E2E_P02Check() function increments the value MaxDeltaCounter by 1 up to maximum value 15. 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 re-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)*

6.6.4 Profile Data Types

6.6.4.1 Profile 2 Protect State Type

[PRS_E2E_00647] ⌈The E2E_P02Protect and E2E_P02Forward functions 'state' shall have the members defined in (see Table 6.30).⌋*(RS_E2E_08528)*

Name	State Type	Description
Counter	Unsigned Integer	Counter to be used for protecting the Data. The initial value is 0. As the counter is incremented before sending, the first Data will have the counter value 1

Table 6.30: E2E Profile 2 Protect State Type

6.6.4.2 Profile 2 Check Status Type

[PRS_E2E_00648] ⌈The E2E_P02Check functions 'State' shall have the members defined in (see Table 6.31).⌋*(RS_E2E_08528, RS_E2E_08537)*

Member Name	Type	Description
LastValidCounter	Unsigned Integer	Counter of last valid received message.
MaxDeltaCounter	Unsigned Integer	MaxDeltaCounter specifies the maximum allowed difference between two counter values of consecutively received valid messages.
WaitForFirstData	Boolean	If true, that means no correct data (with correct Data ID and CRC) has been yet received after the receiver initialization or reinitialization.
NewDataAvailable	Boolean	Indicates that a new data is available to be checked. This attribute is set by the E2E_P02Check function caller, and not by the function itself.
LostData	Unsigned Integer	Number of data (messages) lost since reception of last valid one.
Status	Enumeration	Result of the verification of the Data, determined by the Check function.
SyncCounter	Unsigned Integer	Number of Data required for validating the consistency of the counter that must be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
NoNewOrRepeatedData	Unsigned Integer	Amount of consecutive reception cycles in which either (1) there was no new data, or (2) when the data was repeated.

Table 6.31: E2E Profile 2 Check Status Type

6.6.4.3 Profile 2 Check Status Enumeration

[PRS_E2E_00589] ⌈The E2E_P02Check functions 'State->Status' enumeration type shall consist of the following enumeration values (see Table 6.32).⌋*(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 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 6.32: E2E Profile 2 Check Status Enumeration

6.6.4.4 Profile 2 Configuration Type

[PRS_E2E_00667] ⌈The E2E_P02Protect, E2E_P02Forward and E2E_P02Check functions 'Config' shall have the following members defined in (see Table 6.33).⌋*(RS_E2E_08528)*

Member Name	Type	Description
DataLength	Unsigned Integer	Length of Data, in bits. The value shall be a multiple of 8.
DataIDList	Unsigned Integer Array	An array of appropriately chosen Data IDs for protection against masquerading.
MaxDeltaCounterInit	Unsigned Integer	Initial maximum allowed gap between two counter values of two consecutively received valid Data.
MaxNoNewOrRepeatedData	Unsigned Integer	The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions.
SyncCounterInit	Unsigned Integer	Number of Data required for validating the consistency of the counter that must be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
Offset	Unsigned Integer	Offset of the E2E header in the Data[] array in bits. It shall be: 0 <= Offset <= DataLength-(2*8).Note that if the receiver does not receive new Data at a consecutive read, then the receiver increments the tolerance by 1.

Table 6.33: E2E Profile 2 Configuration Type

6.6.5 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 6.34: E2E Profile 2 protocol example configuration

E2E_P02ProtectStateType field	Value
Counter	0

Table 6.35: 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 6.36: E2E Profile 2 example protect result

6.7 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 6.37).⌋*(RS_E2E_08527, RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548, RS_E2E_08549, RS_E2E_08550)*

Control field	Description
Length	16 bits, to support dynamic-size data.
Counter	16-bits.
CRC	32 bits, polynomial in normal form 0xF4ACFB13, 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 6.37: E2E Profile 4 mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P04LENGTH_POS	0
P04LENGTH_LEN	2
P04COUNTER_POS	2
P04COUNTER_LEN	2
P04DATAID_POS	4
P04DATAID_LEN	4
P04CRC_POS	8
P04CRC_LEN	4
P04CALCULATE_CRC	Crc_CalculateCRC32P4()

Table 6.38: Profile 4-specific data

For behavior and flowcharts of E2E Profile 04 see section 6.3.

6.7.1 Header 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.

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

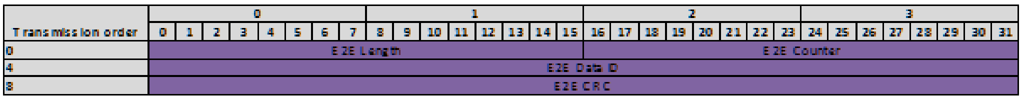

Figure 6.77: E2E Profile 4 Header

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 first) - imposed by profile
2. LSB First (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): 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.

6.7.2 Profile 4 Configuration Type

[PRS_E2E_00651] ⌈The E2E_P04Protect, E2E_P04Forward and E2E_P04Check functions 'Config' shall have the following members defined in (see Table 6.39).⌋*(RS_-E2E_08528, RS_E2E_08537)*

Member Name	**Type**	**Description**
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(12*8). Example: If Offset equals 8, then the high byte of the E2E Length (16 bit) is written to Byte 1, the low Byte is written to Byte 2.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that Length is >= MinDataLength. The value shall be <= 4096*8 (4kB) and shall be >= 12*8
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MinDataLength. The value shall be <= 4096*8 (4kB) and it shall be >= MinDataLength
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.39: E2E Profile 4 Configuration Type

6.7.3 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 6.40: E2E Profile 4 protocol example configuration

E2E_P04ProtectStateType field	Value
Counter	0

Table 6.41: 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	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x86	0x2b	0x05	0x56	0x00	0x00	0x00	0x00
Field	*CRC*				*Data*			

Table 6.42: 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	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0x69	0xd7	0x50	0x2e	0x00	0x00	0x00	0x00
Field	*CRC*				*Data*			

Table 6.43: E2E Profile 4 example short with SOME/IP use case

6.8 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 6.44).⌋*(RS_E2E_08527, RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548, RS_E2E_08549)*

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 6.44: 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
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 6.45: Detectable communication faults using Profile 5

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P05LENGTH_POS	0
P05LENGTH_LEN	0
P05COUNTER_POS	2
P05COUNTER_LEN	1
P05DATAID_POS	0

P05DATAID_LEN	0
P05CRC_POS	0
P05CRC_LEN	2
P05CALCULATE_CRC	Crc_CalculateCRC16()

Table 6.46: Profile 5-specific data

6.8.1 Header 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.

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

	0								1								2							
Transmission order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	E2E CRC																E2E Counter							

Figure 6.78: 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 first) applicable for both implicit and explicit header fields - imposed by profile
2. MSB First (most significant bit within byte first) - imposed by FlexrayCAN bus.

6.8.1.1 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 :The counter value 0xFF is not reserved as a special invalid value, but it is used as a normal counter value.

6.8.1.2 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 should 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.

6.8.1.3 Length

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

6.8.1.4 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_08531)*

6.8.2 Creation of the E2E-Header

6.8.2.1 E2E_P05Protect

The function E2E_P05Protect() performs the steps as specified by the following diagrams (see also section 6.3).

[PRS_E2E_00403] ⌈The function E2E_P05Protect() shall have the overall behavior as shown in Figure 6.79.⌋*(RS_E2E_08539)*

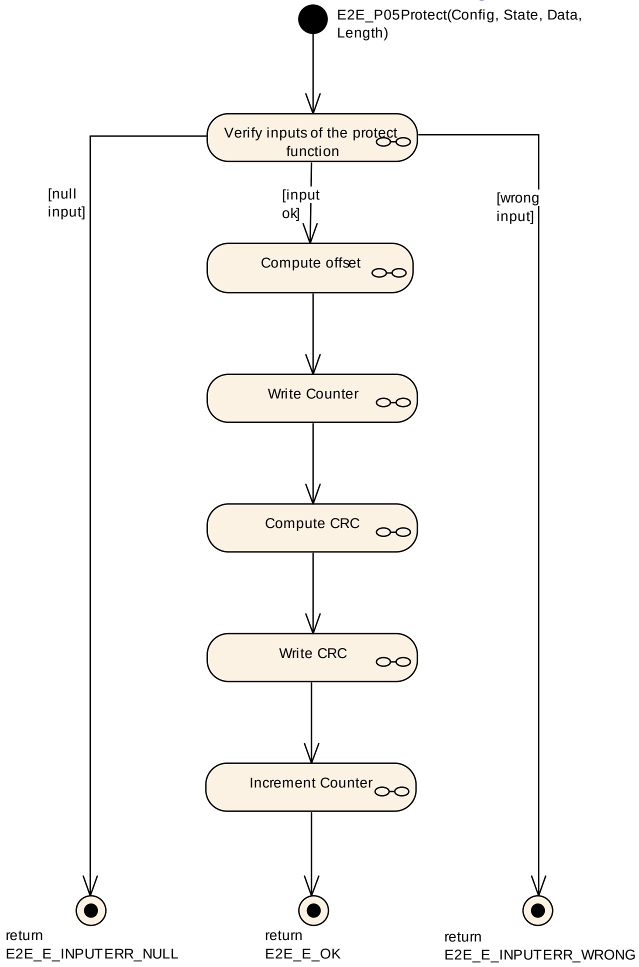

Figure 6.79: E2E Profile 5 Protect

[PRS_E2E_00404] ⌈The step "Verify inputs of the protect function" in E2E_P05Protect() shall behave as shown in Figure 6.3.⌋*(RS_E2E_08539)*

[PRS_E2E_00469] ⌈The step "Compute offset" in E2E_P05Protect(), E2E_P05Forward() and E2E_P05Check() shall behave as shown in Figure 6.4.⌋*(RS_E2E_08539)*

[PRS_E2E_00405] ⌈The step "Write Counter" in E2E_P05Protect() shall behave as shown in Figure 6.6.⌋*(RS_E2E_08539)*

[PRS_E2E_00406] ⌈The step "Compute CRC" in E2E_P05Protect() and in E2E_P05Check shall behave as shown in Figure 6.80.⌋*(RS_E2E_08539)*

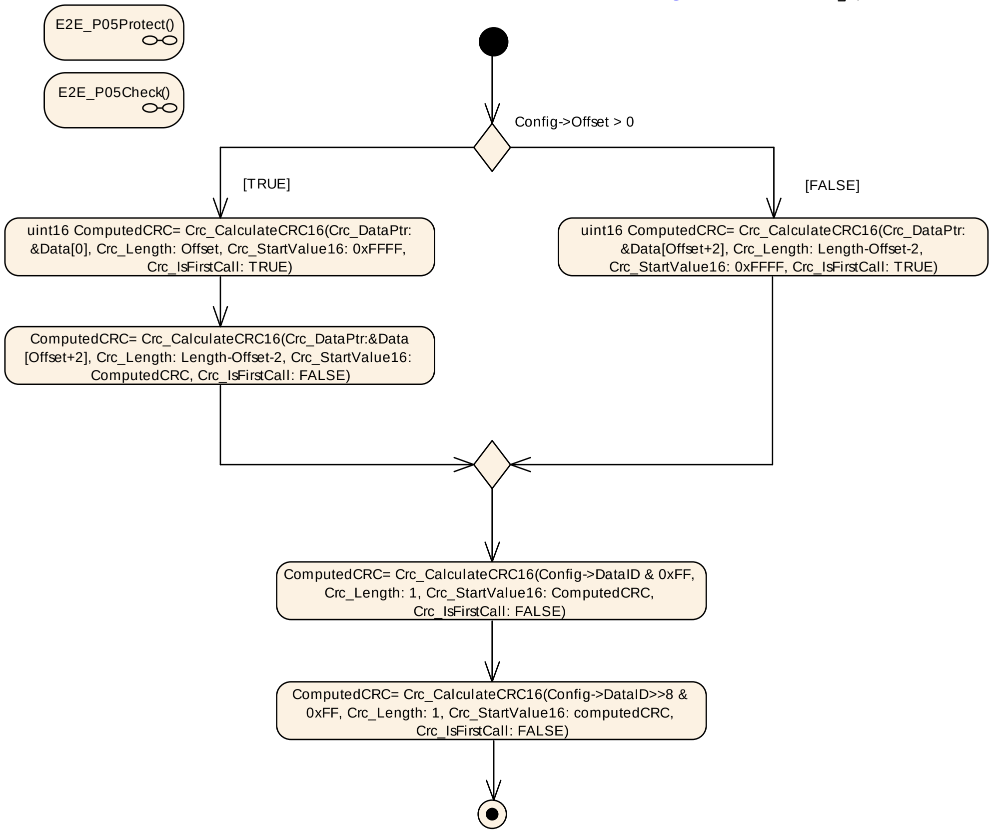

Figure 6.80: E2E Profile 5 Protect and Check step "Compute CRC"

[PRS_E2E_00407] ⌈The step "Write CRC" in E2E_P05Protect() and E2E_P05Forward() shall behave as shown in Figure 6.9.⌋*(RS_E2E_08539)*

[PRS_E2E_00409] ⌈The step "Increment Counter" in E2E_P05Protect() and E2E_P05Forward() shall behave as shown in Figure 6.10.⌋*(RS_E2E_08539)*

6.8.2.2 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 diagrams (see also section 6.3).

[PRS_E2E_00639] ⌈The function E2E_P05Forward() shall have the overall behavior as shown in Figure 6.81.⌋*(RS_E2E_08539)*

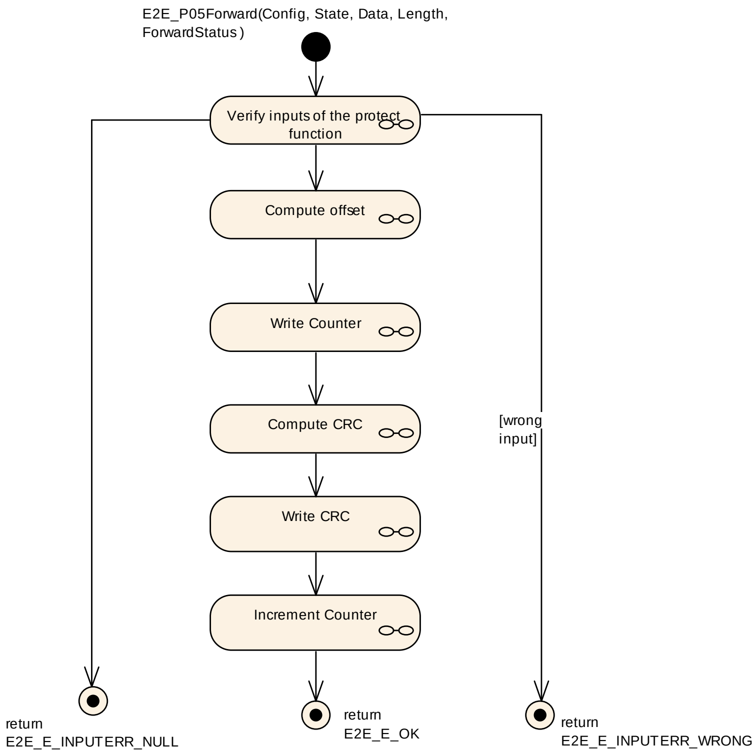

Figure 6.81: E2E Profile 5 Forward

Following steps are described in Section in Section 6.8.2.1

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

[PRS_E2E_00619] ⌈The step ”Verify inputs of the forward function” in E2E_P05Forward() shall behave as shown in Figure 6.12.⌋ *(RS_E2E_08539)*

[PRS_E2E_00620] ⌈The step ”Write Counter” in E2E_P05Forward() shall behave as shown in Figure 6.13.⌋ *(RS_E2E_08539)*

[PRS_E2E_00621] ⌈The step "Compute CRC" in E2E_P05Forward() shall behave as shown in Figure 6.82.⌋*(RS_E2E_08539)*

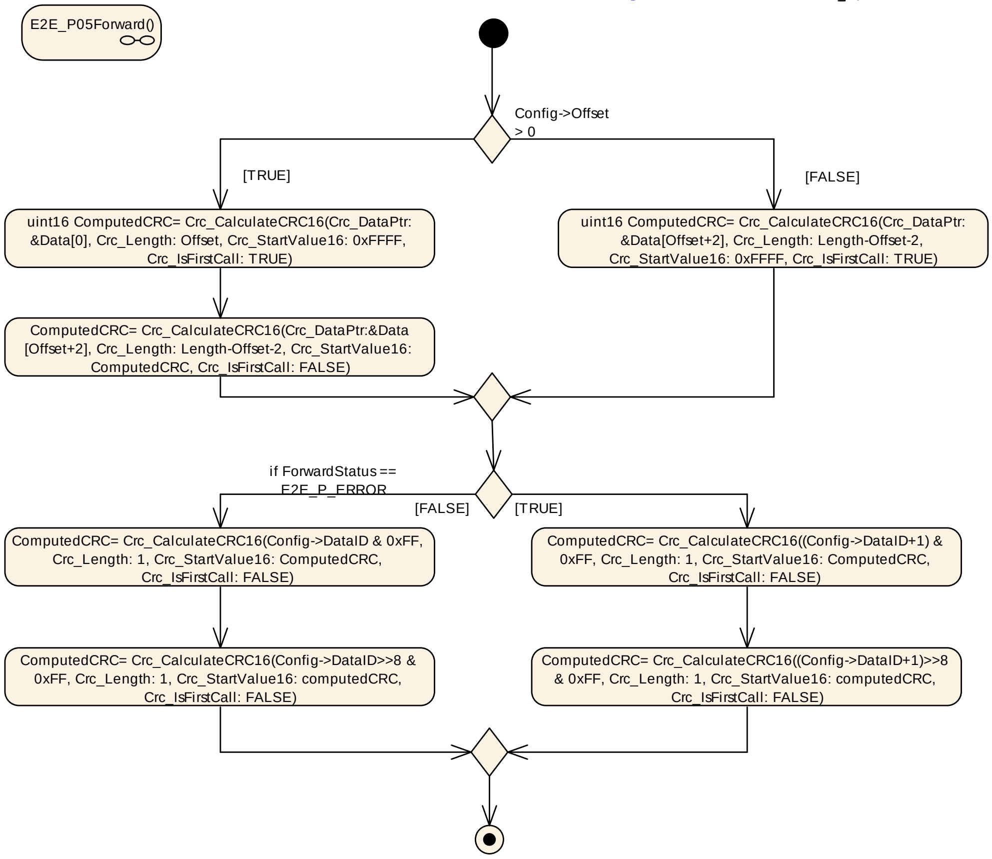

Figure 6.82: E2E Profile 5 Forward step "Compute CRC"

6.8.3 Evaluation of the E2E-Header

6.8.3.1 E2E_P05Check

The function E2E_P05Check() performs the actions as specified by the following diagrams (see also section 6.3).

[PRS_E2E_00411] ⌈The function E2E_P05Check() shall have the overall behavior as shown in Figure 6.83.⌋*(RS_E2E_08539)*

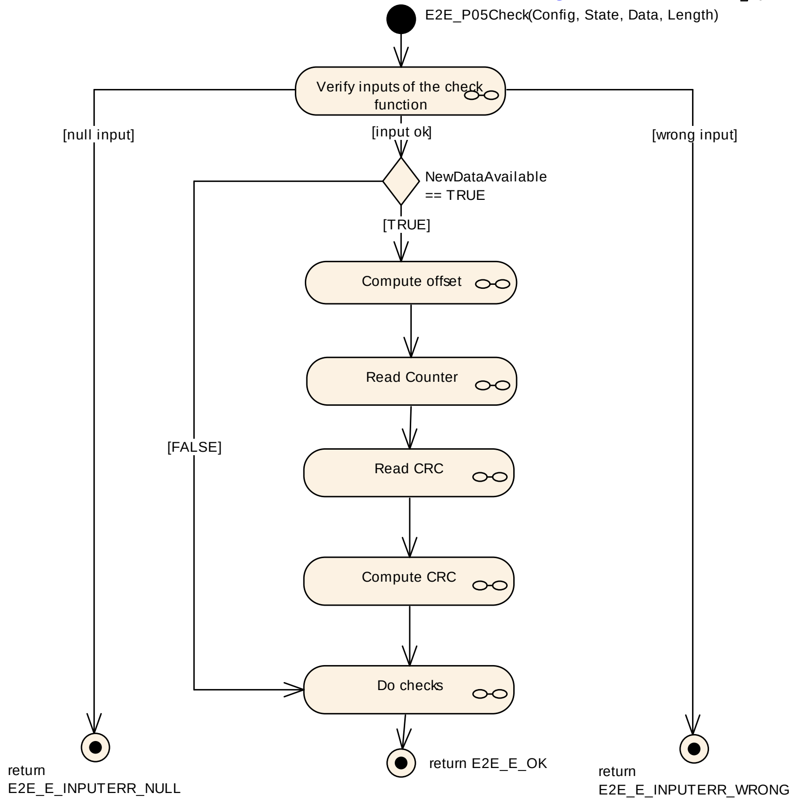

Figure 6.83: E2E Profile 5 Check

[PRS_E2E_00412] ⌈The step "Verify inputs of the check function" in E2E_P05Check() shall behave as shown in Figure 6.16.⌋*(RS_E2E_08539)*

[PRS_E2E_00413] ⌈The step "Read Counter" in E2E_P05Check() shall behave as shown in Figure 6.18.⌋*(RS_E2E_08539)*

[PRS_E2E_00414] ⌈The step "Read CRC" in E2E_P05Check() shall behave as shown in Figure 6.20.⌋*(RS_E2E_08539)*

[PRS_E2E_00416] ⌈The step "Do Checks' in E2E_P05Check() shall behave as shown in Figure 6.21.⌋*(RS_E2E_08539)*

6.8.4 Profile Data Types

6.8.4.1 Profile 5 Protect State Type

[PRS_E2E_00652] ⌈The E2E_P05Protect and E2E_P05Forward functions 'state' shall have the members defined in (see Table 6.47).⌋*(RS_E2E_08528)*

Name	State Type	Description

Counter	Unsigned Integer	Counter to be used for protecting the next Data. The initial value is 0, which means that in the first cycle, Counter is 0. Each time E2E_P05Protect() is called, it increments the counter up to 0xFF.

Table 6.47: E2E Profile 5 Protect State Type

6.8.4.2 Profile 5 Check Status Type

[PRS_E2E_00653] ⌈The E2E_P05Check functions 'State' shall have the members defined in (see Table 6.48).⌋*(RS_E2E_08528)*

Member Name	State Type	Description
Counter	Unsigned Integer	Counter of the data in previous cycle.
Status	Enumeration	Result of the verification of the Data in this cycle, determined by the Check function.

Table 6.48: E2E Profile 5 Check Status Type

6.8.4.3 Profile 5 Check Status Enumeration

[PRS_E2E_00591] ⌈The E2E_P05Check functions 'State->Status' enumeration type shall consist the following enumeration values (see Table 6.49).⌋*(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 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 6.49: E2E Profile 5 Check Status Enumeration

6.8.4.4 Profile 5 Configuration Type

[PRS_E2E_00654] ⌈The E2E_P05Protect, E2E_P05Forward and E2E_P05Check functions 'Config' shall have the following members defined in (see Table 6.50).⌋*(RS_E2E_08528, RS_E2E_08537, RS_E2E_08539)*

Member Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= DataLength-(3*8). Example: If Offset equals 8, then the low byte of the E2E Crc (16 bit) is written to Byte 1, the high Byte is written to Byte 2.
DataLength	Unsigned Integer	Length of Data, in bits. The value shall be <= 4096*8 (4kB) and shall be >= 3*8
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.50: E2E Profile 5 Configuration Type

6.8.5 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 6.51: E2E Profile 5 protocol example configuration

E2E_P05ProtectStateType field	Value

Counter	0

Table 6.52: 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 6.53: 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 6.54: E2E Profile 5 example short with SOME/IP use case

6.9 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 6.55).⌋*(RS_E2E_08527, RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548, RS_E2E_08549)*

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 6.55: 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 6.56: Detectable communication faults using Profile 6

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P06LENGTH_POS	2
P06LENGTH_LEN	2
P06COUNTER_POS	4
P06COUNTER_LEN	1
P06DATAID_POS	0
P06DATAID_LEN	0
P06CRC_POS	0
P06CRC_LEN	2
P06CALCULATE_CRC	Crc_CalculateCRC16()

Table 6.57: Profile 6-specific data

6.9.1 Header 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.

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

<table>
<tr><th rowspan="2">Transmission order</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td colspan="16">E2E CRC</td><td colspan="16">E2E Length</td></tr>
<tr><td>4</td><td colspan="8">E2E Counter</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Figure 6.84: 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 first), applicable for both implicit and explicit header fields - imposed by profile
2. LSB First (least significant bit within byte first) - imposed by TCP/IP bus

6.9.1.1 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.

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

6.9.1.2 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 should 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.

6.9.1.3 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 an explicit transmission of the length. The Length includes user data + E2E Header (CRC + Counter + Length).

6.9.1.4 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_08531)*

6.9.2 Creation of E2E-Header

6.9.2.1 E2E_P06Protect

The function E2E_P06Protect() performs the steps as specified by the following diagrams (see also section 6.3).

[PRS_E2E_00423] ⌈The function E2E_P06Protect() shall have the overall behavior as shown in Figure 6.85.⌋*(RS_E2E_08539)*

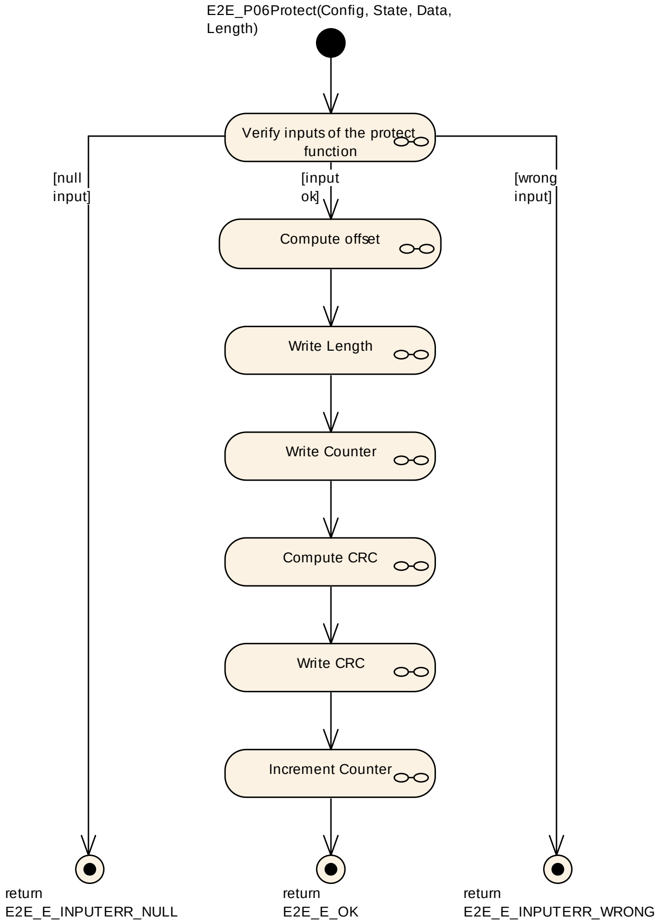

Figure 6.85: E2E Profile 6 Protect

[PRS_E2E_00424] ⌈The step "Verify inputs of the protect function" in E2E_P06Protect() shall behave as shown in Figure 6.3.⌋*(RS_E2E_08539)*

[PRS_E2E_00470] ⌈The step "Compute offset" in E2E_P06Protect(), E2E_P06Forward() and E2E_P06Check() shall behave as shown in Figure 6.4.⌋*(RS_E2E_08539)*

[PRS_E2E_00425] ⌈The step "Write Length" in E2E_P06Protect() and E2E_P06Forward() shall behave as shown in Figure 6.5.⌋*(RS_E2E_08539)*

[PRS_E2E_00426] ⌈The step "Write Counter" in E2E_P06Protect() shall behave as shown in Figure 6.6.⌋*(RS_E2E_08539)*

[PRS_E2E_00427] ⌈The step "Compute CRC" in E2E_P06Protect() and E2E_P06Check() shall behave as shown in Figure 6.86.⌋*(RS_E2E_08539)*

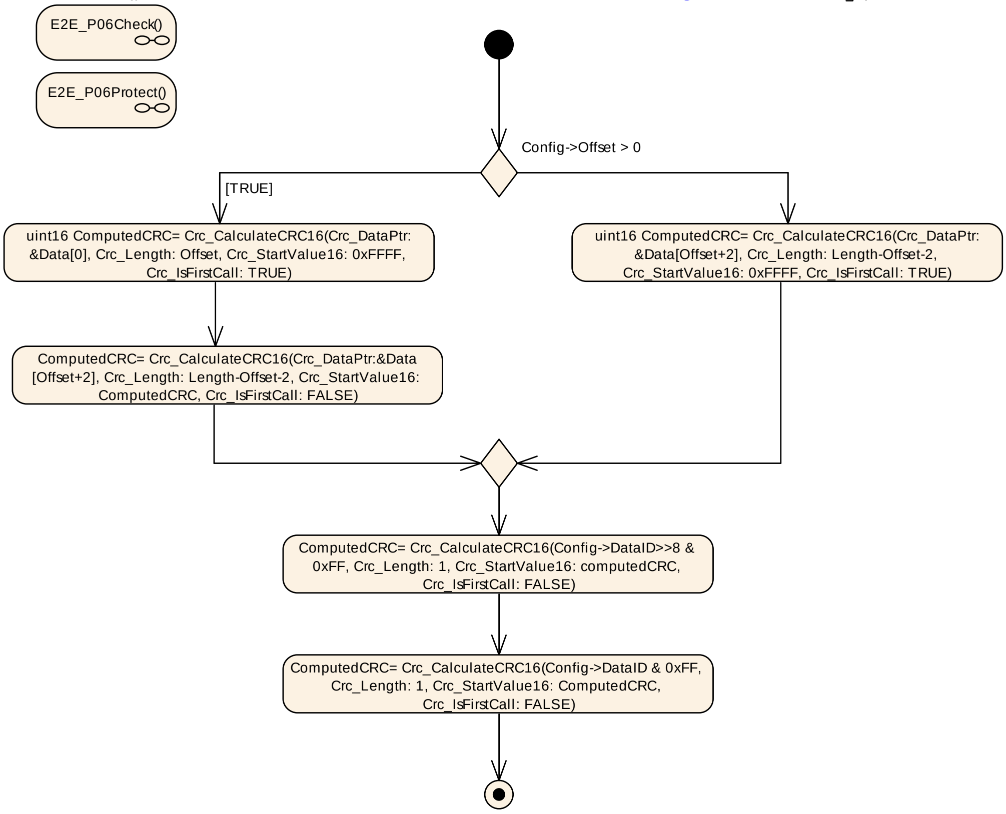

Figure 6.86: E2E Profile 6 Protect and Check step "Compute CRC"

[PRS_E2E_00428] ⌈The step "Write CRC" in E2E_P06Protect() and E2E_P06Forward() shall behave as shown in Figure 6.9.⌋*(RS_E2E_08539)*

[PRS_E2E_00429] ⌈The step "Increment Counter" in E2E_P06Protect() and E2E_P06Forward() shall behave as shown in Figure 6.10.⌋*(RS_E2E_08539)*

6.9.2.2 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 diagrams (see also section 6.3).

[PRS_E2E_00622] ⌈The function E2E_P06Forward() shall have the overall behavior as shown in Figure 6.87.⌋*(RS_E2E_08539)*

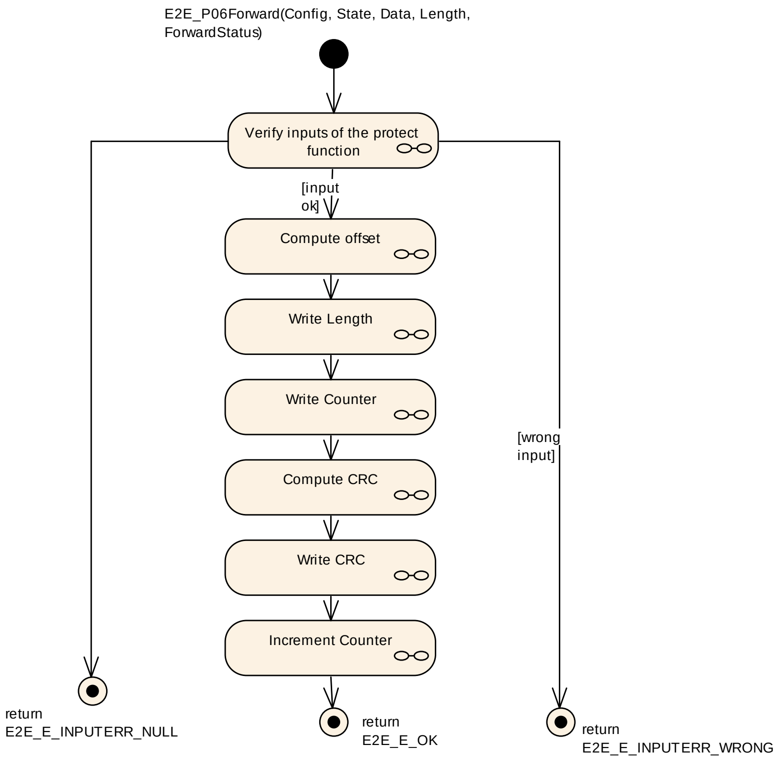

Figure 6.87: E2E Profile 6 Forward

Following steps are described in Section in Section 6.9.2.1

- ”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] ⌈The step ”Verify inputs of the forward function” in E2E_P06Forward() shall behave as shown in Figure 6.12.⌋*(RS_E2E_08539)*

[PRS_E2E_00624] ⌈The step ”Write Counter” in E2E_P06Forward() shall behave as shown in Figure 6.13.⌋*(RS_E2E_08539)*

[PRS_E2E_00625] ⌈The step "Compute CRC" in E2E_P06Forward() shall behave as shown in Figure 6.88.⌋*(RS_E2E_08539)*

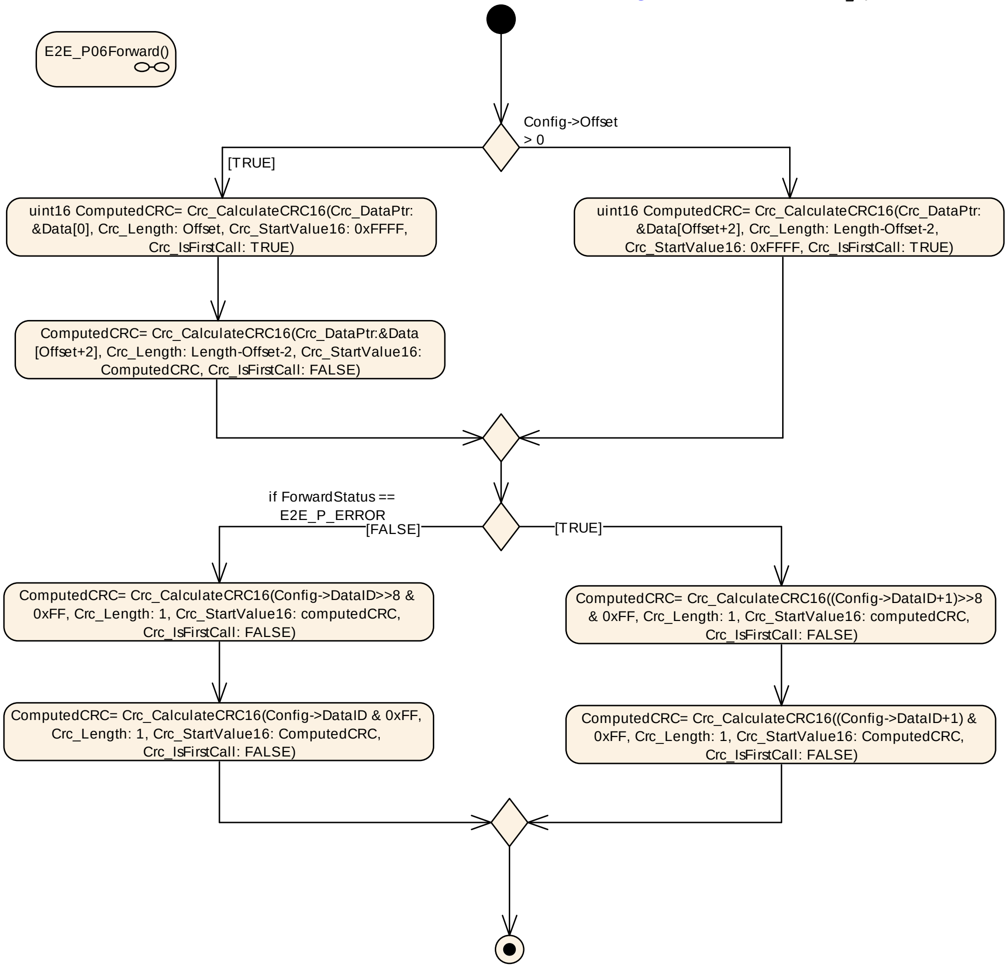

Figure 6.88: E2E Profile 6 Forward step "Compute CRC"

6.9.3 Evaluation of E2E-Header

6.9.3.1 E2E_P06Check

The function E2E_P06Check() performs the actions as specified by the following diagrams (see also section 6.3).

[PRS_E2E_00430] ⌈The function E2E_P06Check() shall have the overall behavior as shown in Figure 6.89.⌋*(RS_E2E_08539)*

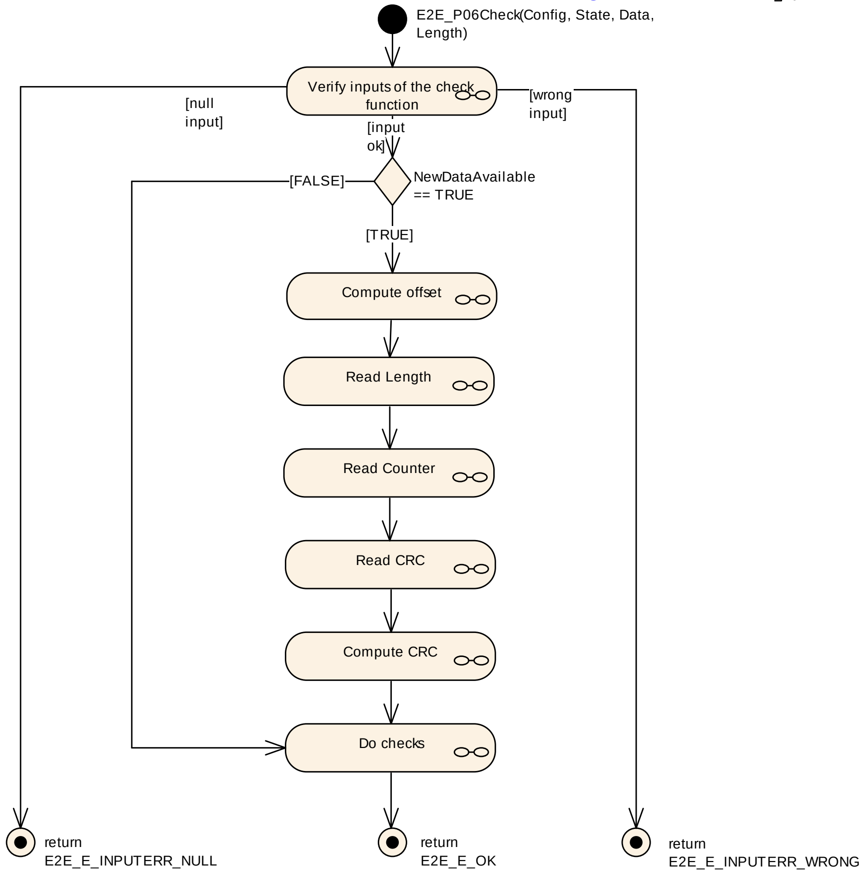

Figure 6.89: E2E Profile 6 Check

[PRS_E2E_00431] ⌈The step "Verify Inputs" in E2E_P06Check() shall behave as shown in Figure 6.16.⌋*(RS_E2E_08539)*

[PRS_E2E_00432] ⌈The step "Read Length" in E2E_P06Check() shall behave as shown in Figure 6.17.⌋*(RS_E2E_08539)*

[PRS_E2E_00433] ⌈The step "Read Counter" in E2E_P06Check() shall behave as shown in Figure 6.18.⌋*(RS_E2E_08539)*

[PRS_E2E_00434] ⌈The step "Read CRC" in E2E_P06Check() shall behave as shown in Figure 6.20.⌋*(RS_E2E_08539)*

[PRS_E2E_00436] ⌈The step "Do Checks" in E2E_P06Check() shall behave as shown in Figure 6.21.⌋*(RS_E2E_08539)*

6.9.4 Profile Data Types

6.9.4.1 Profile 6 Protect State Type

[PRS_E2E_00655] ⌈The E2E_P06Protect and E2E_P06Forward functions 'state' shall have the members defined in (see Table 6.58).⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter to be used for protecting the next Data. The initial value is 0, which means that in the first cycle, Counter is 0. Each time E2E_P06Protect() is called, it increments the counter up to 0xFF. After the maximum value is reached, the next value is 0x0. The overflow is not reported to the caller.

Table 6.58: E2E Profile 6 Protect State Type

6.9.4.2 Profile 6 Check Status Type

[PRS_E2E_00656] ⌈The E2E_P06Check functions 'State' shall have the members defined in (see Table 6.59).⌋*(RS_E2E_08528)*

Member Name	Type	Description
Counter	Unsigned Integer	Counter of the data in previous cycle.
Status	Enumeration	Result of the verification of the Data in this cycle, determined by the Check function.

Table 6.59: E2E Profile 6 Check Status Type

6.9.4.3 Profile 6 Check Status Enumeration

[PRS_E2E_00592] ⌈The E2E_P06Check functions 'State->Status' enumeration type shall consist of the following enumeration values (see Table 6.60).⌋*(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 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 6.60: E2E Profile 6 Check Status Enumeration

6.9.4.4 Profile 6 Configuration Type

[PRS_E2E_00657] ⌈The E2E_P06Protect, E2E_P06Forward and E2E_P06Check functions 'Config' shall have the following members defined in (see Table 6.61).⌋*(RS_E2E_08528, RS_E2E_08537)*

Member Name	State Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(5*8). Example: If Offset equals 8, then the high byte of the E2E Length (16 bit) is written to Byte 1, the low Byte is written to Byte 2. This may be considered similar to E2E_P06STATUS_REPEATED.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that Length is >= MinDataLength. The value shall be <= 4096*8 (4kB) and shall be >= 5*8
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MinDataLength. The value shall be <= 4096*8 (4kB) and it shall be >= MinDataLength
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data. For example, if the receiver gets Data with counter 1 and MaxDeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.

Table 6.61: E2E Profile 6 Configuration Type

6.9.5 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 6.62: E2E Profile 6 protocol example configuration

E2E_P06ProtectStateType field	Value
Counter	0

Table 6.63: 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 6.64: 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 6.65: E2E Profile 6 example short with SOME/IP use case

6.10 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 6.66).⌋*(RS_E2E_08527, RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_-E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548)*

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 6.66: E2E Profile 7 mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P07LENGTH_POS	8
P07LENGTH_LEN	4
P07COUNTER_POS	12
P07COUNTER_LEN	4
P07DATAID_POS	16
P07DATAID_LEN	4
P07CRC_POS	0
P07CRC_LEN	8
P07CALCULATE_CRC	Crc_CalculateCRC64()

Table 6.67: Profile 7-specific data

For behavior and flowcharts of E2E Profile 07 see section 6.3.

6.10.1 Header 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.

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

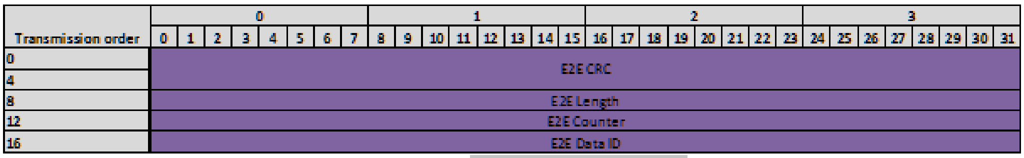

Figure 6.90: Profile 7 Header

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 first) - imposed by profile
2. LSB First (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.

6.10.2 Profile 7 Configuration Type

[PRS_E2E_00660] ⌈The E2E_P07Protect, E2E_P07Forward and E2E_P07Check functions 'Config' shall have the following members defined in (see Table 6.68).⌋*(RS_E2E_08528, RS_E2E_08537)*

Member Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(20*8). Example: If Offset equals 8, then the first byte of the E2E Length (32 bit) is written to byte 1, the next byte is written to byte 2 and so on.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that Length is >= MinDataLength. The value shall be >= 20*8 and <= MaxDataLength.
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MinDataLength. The value shall be >= MinDataLengthMaximal length of Data, in bits. E2E checks that DataLength is <= MinDataLength. The value shall be >= MinDataLength
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.68: E2E Profile 7 Configuration Type

6.10.3 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 6.69: E2E Profile 7 protocol example configuration

E2E_P07ProtectStateType field	Value
Counter	0

Table 6.70: 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 6.71: 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 6.72: E2E Profile 7 example short with SOME/IP use case

6.11 Specification of E2E Profile 8

[PRS_E2E_00736] ⌈Profile 8 shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID (see Table 6.73).⌋ *(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_E2E_08549)*

Control field	Description
Length	32 bits, to support dynamic-size data.
Counter	32 bits.
CRC	32 bits, polynomial in normal form 0xF4ACFB13, 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 6.73: E2E Profile 8 mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P08LENGTH_POS	4
P08LENGTH_LEN	4
P08COUNTER_POS	8
P08COUNTER_LEN	4
P08DATAID_POS	12
P08DATAID_LEN	4
P08CRC_POS	0
P08CRC_LEN	4
P08CALCULATE_CRC	Crc_CalculateCRC32P4()

Table 6.74: Profile 8-specific data

For behavior and flowcharts of E2E Profile 08 see section 6.3.

6.11.1 Header layout

In the E2E Profile 8, the user data layout (of the data to be protected) is not constrained by E2E Profile 8 - there is only a requirement that the length of data to be protected is multiple of 1 byte.

The header of the E2E Profile 8 has one fixed layout, as follows:

<table>
<tr><th></th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><td>Transmission order</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>0</td><td colspan="32">E2E CRC</td></tr>
<tr><td>4</td><td colspan="32">E2E Length</td></tr>
<tr><td>8</td><td colspan="32">E2E Counter</td></tr>
<tr><td>12</td><td colspan="32">E2E DataID</td></tr>
</table>

Figure 6.91: Profile 8 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 first) - imposed by profile
2. LSB First (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 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.

6.11.2 Profile 8 Configuration Type

[PRS_E2E_00706] ⌈The E2E_P08Protect, E2E_P08Forward and E2E_P08Check functions 'Config' shall have the following members defined in (see Table 6.75).⌋*(RS_E2E_08528, RS_E2E_08537)*

Member Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(16*8). Example: If Offset equals 8, then the first byte of the E2E Length (32 bit) is written to byte 1, the next byte is written to byte 2 and so on.

MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that Length is >= MinDataLength. The value shall be >= 16*8 and <= MaxDataLength.
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MaxDataLength. The value shall be <= 4294967295 and >= MinDataLength.
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.75: E2E Profile 8 Configuration Type

6.11.3 E2E Profile 8 Protocol Examples

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

E2E_P08ConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000 0000
MinDataLength	128
MaxDataLength	4294967295
MaxDeltaCounter	1

Table 6.76: E2E Profile 8 protocol example configuration

E2E_P08ProtectStateType field	Value
Counter	0

Table 6.77: E2E Profile 8 example state initialization

Result data of E2E_P08Protect() with short data length (length 20 bytes, means 4 actual data bytes), offset = 0, counter = 0:

Byte	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Data	0x41	0x49	0x4e	0x52	0x00	0x00	0x00	0x14
Field	*CRC32*				*Length*			
Byte	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Data	0x00	0x00	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Counter*				*DataID*			
Byte	**17**	**18**	**19**	**20**				
Data	0x00	0x00	0x00	0x00				
Field	*Data*							

Table 6.78: E2E Profile 8 example short

Result data of E2E_P08Protect() with minimum data length (4 data bytes), offset = 64 (as with SOME/IP header use case), datalength = 28, counter = 0:

Byte	1	2	3	4	5	6	7	8
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	9	10	11	12	13	14	15	16
Data	0xe8	0x91	0xe5	0xa8	0x00	0x00	0x00	0x1c
Field	*CRC32*				*Length*			
Byte	17	18	19	20	21	22	23	24
Data	0x00	0x00	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Counter*				*DataID*			
Byte	25	26	27	28				
Data	0x00	0x00	0x00	0x00				
Field	*Data*							

Table 6.79: E2E Profile 8 example short with SOME/IP use case

6.12 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, the following legacy DataIDModes are by now obsolete and omitted: E2E P11 DATAID LOW, E2E P11 DATAID ALT.

[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 6.80).⌋*(RS_E2E_08527, RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548, RS_E2E_08549)*

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 6.80: 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 6.81: Detectable communication faults using Profile 11

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

6.12.1 Header 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.

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 6.92: 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 6.93: 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 first) applicable for both implicit and explicit header fields - imposed by profile
2. MSB First (most significant bit within byte first) - imposed by Flexray/CAN bus.

6.12.1.1 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 shall never be used by the E2E profile 11.

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

6.12.1.2 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_00507] ⌈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.

6.12.1.3 Length

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

6.12.1.4 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.

6.12.2 Creation of the E2E-Header

6.12.2.1 E2E_P11Protect

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

[PRS_E2E_00509] ⌈The function E2E_P11Protect() shall have the overall behavior as shown in Figure 6.94.⌋*(RS_E2E_08539)*

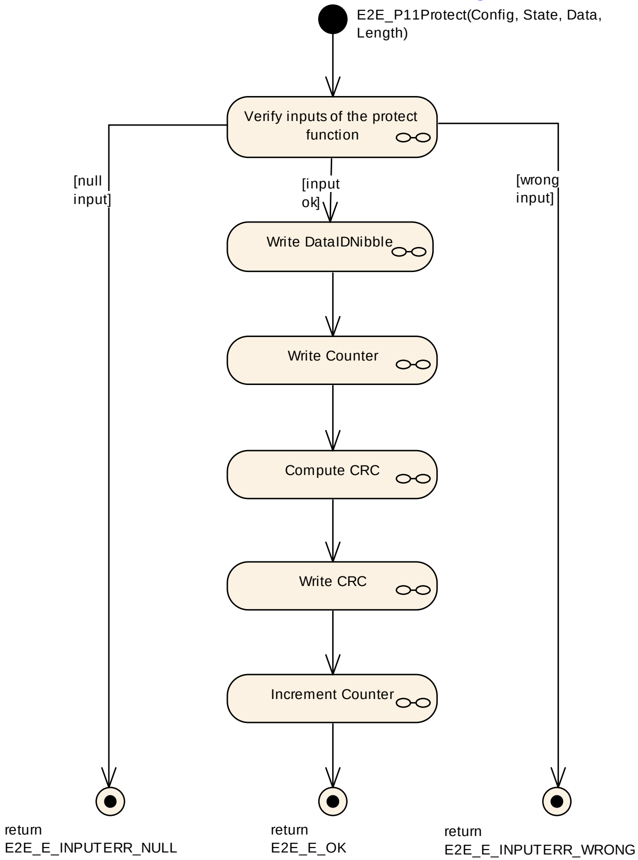

Figure 6.94: E2E Profile 11 Protect

[PRS_E2E_00510] ⌈The step "Verify inputs of the protect function" in E2E_P11Protect() shall behave as shown in Figure 6.95.⌋*(RS_E2E_08539)*

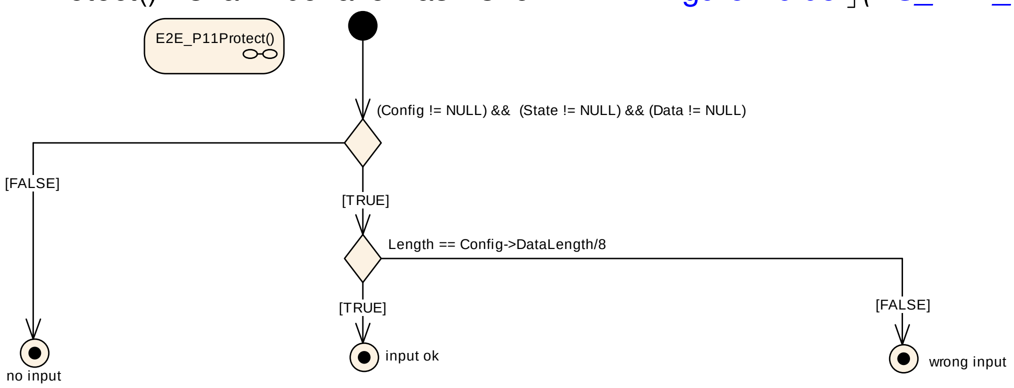

Figure 6.95: E2E Profile 11 Protect step "Verify inputs of the protect function"

[PRS_E2E_00511] ⌈The step „Write DataIDNibble" in E2E_P11Protect() shall behave as shown in Figure 6.96.⌋*(RS_E2E_08539)*

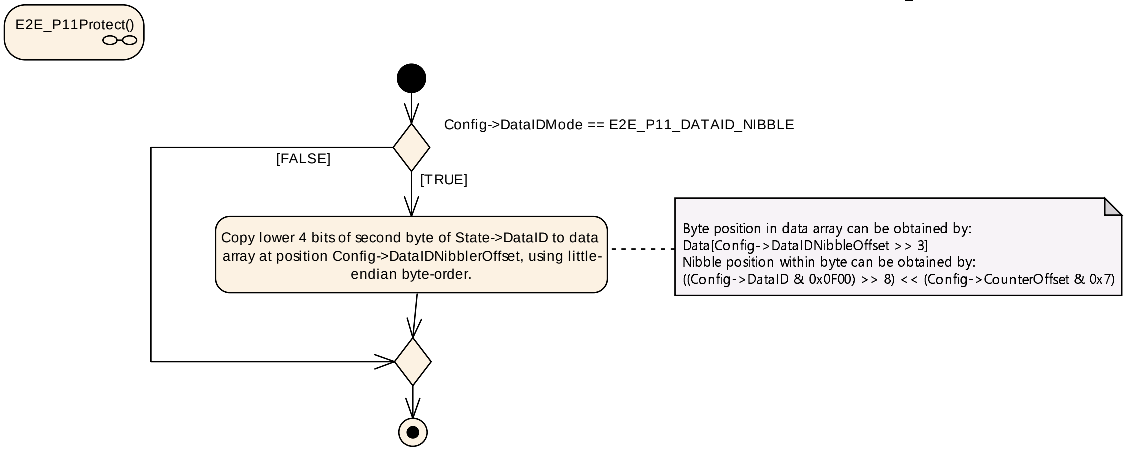

Figure 6.96: E2E Profile 11 Protect step "Write DataIDNibble"

[PRS_E2E_00512] ⌈The step "Write Counter" in E2E_P11Protect() shall behave as shown in Figure 6.97.⌋*(RS_E2E_08539)*

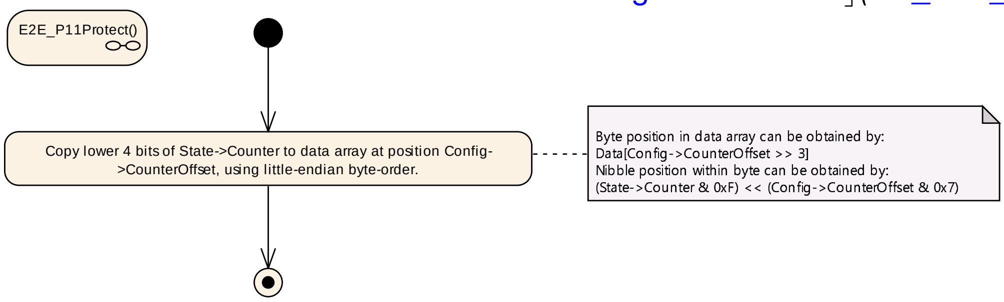

Figure 6.97: E2E Profile 11 Protect step "Write Counter"

[PRS_E2E_00513] ⌈The step "Compute CRC" in E2E_P11Protect() and in E2E_P11Check shall behave as shown in Figure 6.98.⌋*(RS_E2E_08539)*

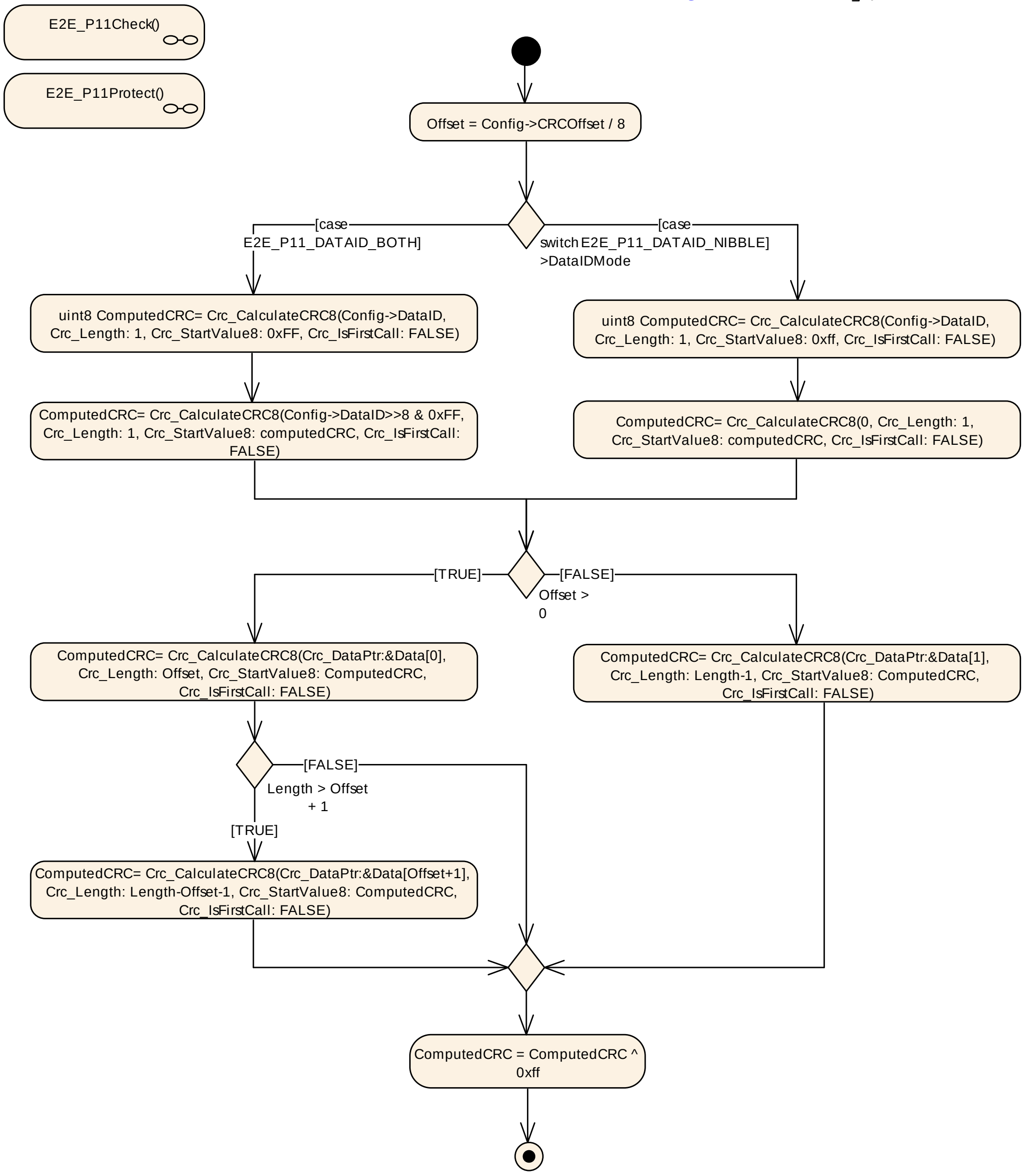

Figure 6.98: E2E Profile 11 Protect and Check step "Compute CRC"

[PRS_E2E_00514] ⌈The step "Write CRC" in E2E_P11Protect() and E2E_P11Forward() shall behave as shown in Figure 6.99.⌋*(RS_E2E_08539)*

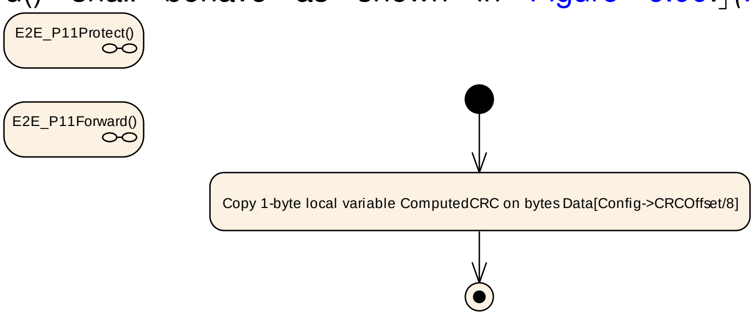

Figure 6.99: E2E Profile 11 Protect and Forward step "Write CRC"

[PRS_E2E_00515] ⌈The step ”Increment Counter” in E2E_P11Protect() and E2E_P11Forward() shall behave as shown in Figure 6.100.⌋*(RS_E2E_08539)*

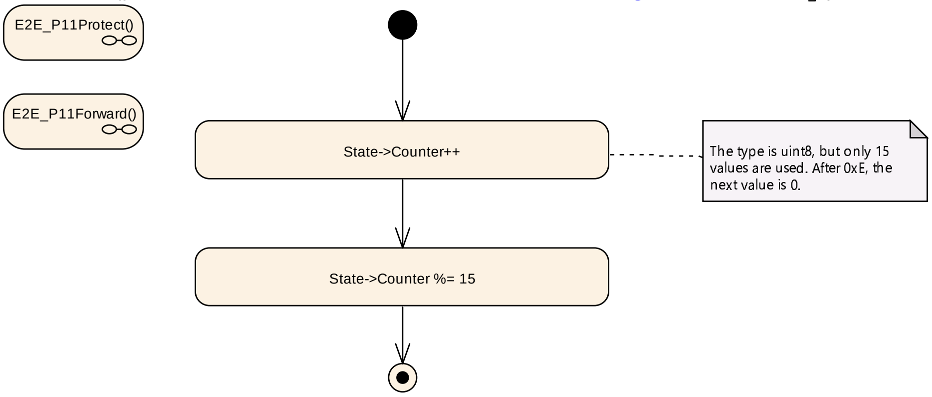

Figure 6.100: E2E Profile 11 Protect and Forward step ”Increment Counter”

6.12.2.2 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] ⌈The function E2E_P11Forward() shall have the overall behavior as shown in Figure 6.101.⌋*(RS_E2E_08539)*

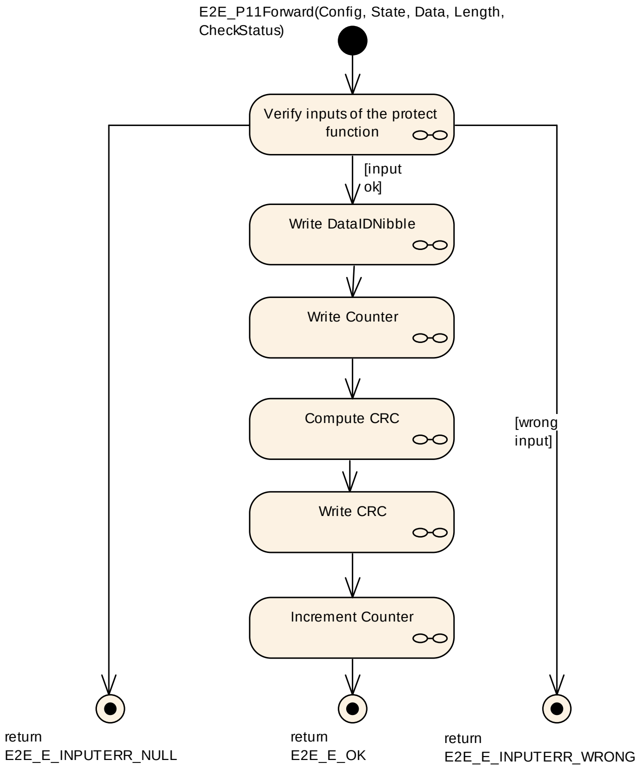

Figure 6.101: E2E Profile 11 Forward

Following steps are described in Section in Section 6.12.2.1

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

[PRS_E2E_00631] ⌈The step ”Verify inputs of the forward function” in E2E_P11Forward() shall behave as shown in Figure 6.102.⌋*(RS_E2E_08539)*

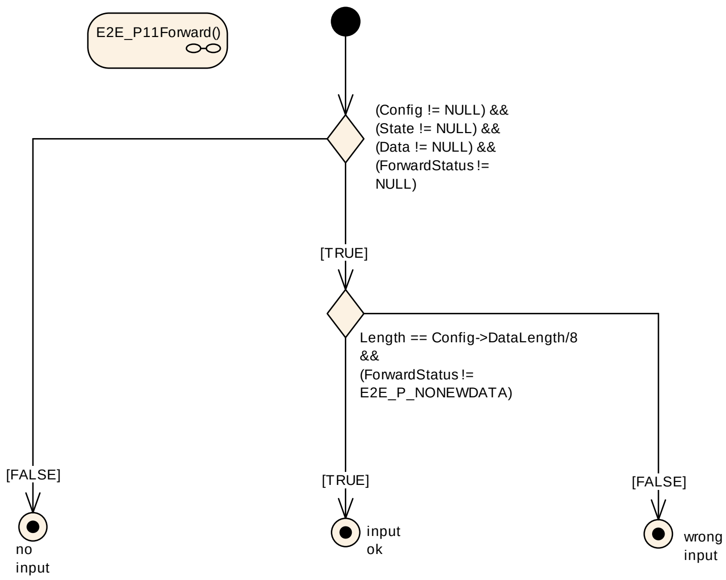

Figure 6.102: E2E Profile 11 Forward step ”Verify inputs of the forward function”

[PRS_E2E_00632] ⌈The step „Write DataIDNibble" in E2E_P11Forward() shall behave as shown in Figure 6.103.⌋*(RS_E2E_08539)*

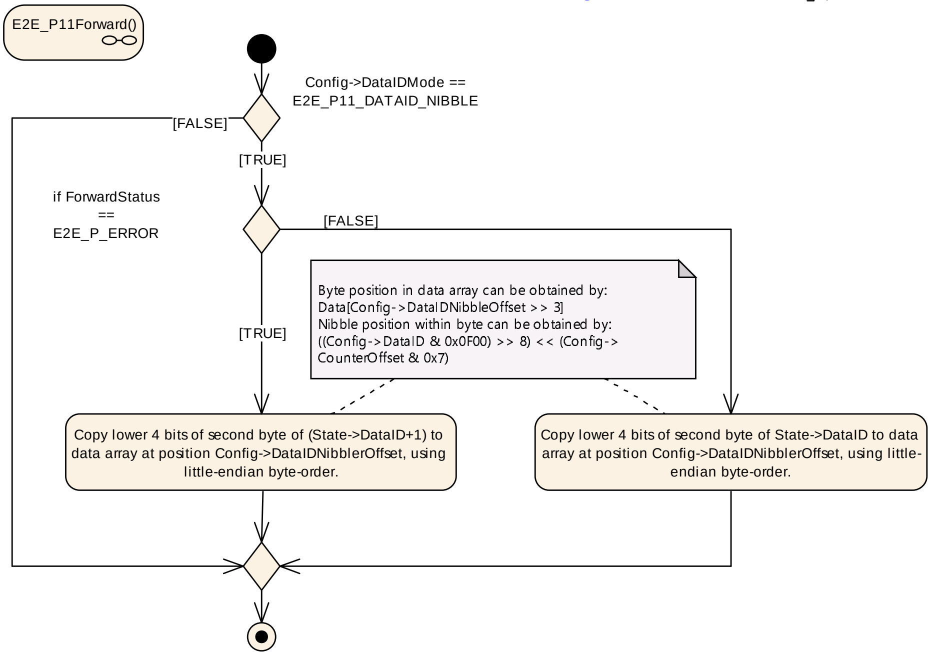

Figure 6.103: E2E Profile 11 Forward step "Write DataIDNibble"

[PRS_E2E_00633] ⌈The step "Write Counter" in E2E_P11Forward() shall behave as shown in Figure 6.104.⌋*(RS_E2E_08539)*

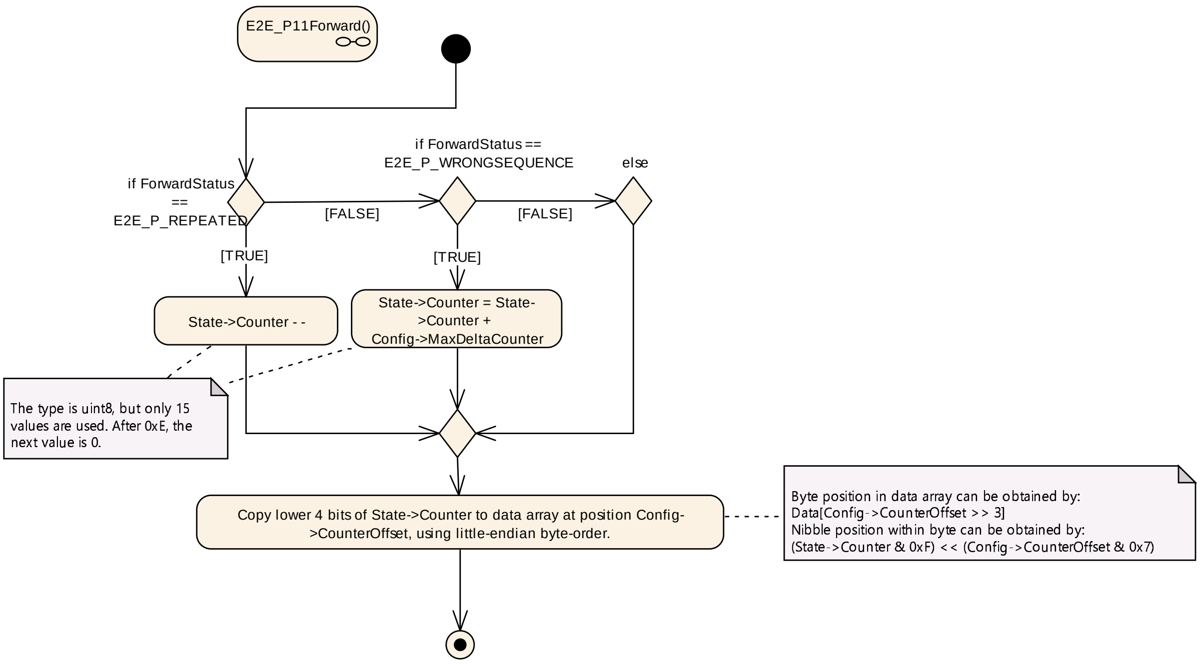

Figure 6.104: E2E Profile 11 Forward step "Write Counter"

[PRS_E2E_00634] ⌈The step "Compute CRC" in E2E_P11Forward() shall behave as shown in Figure 6.105.⌋*(RS_E2E_08539)*

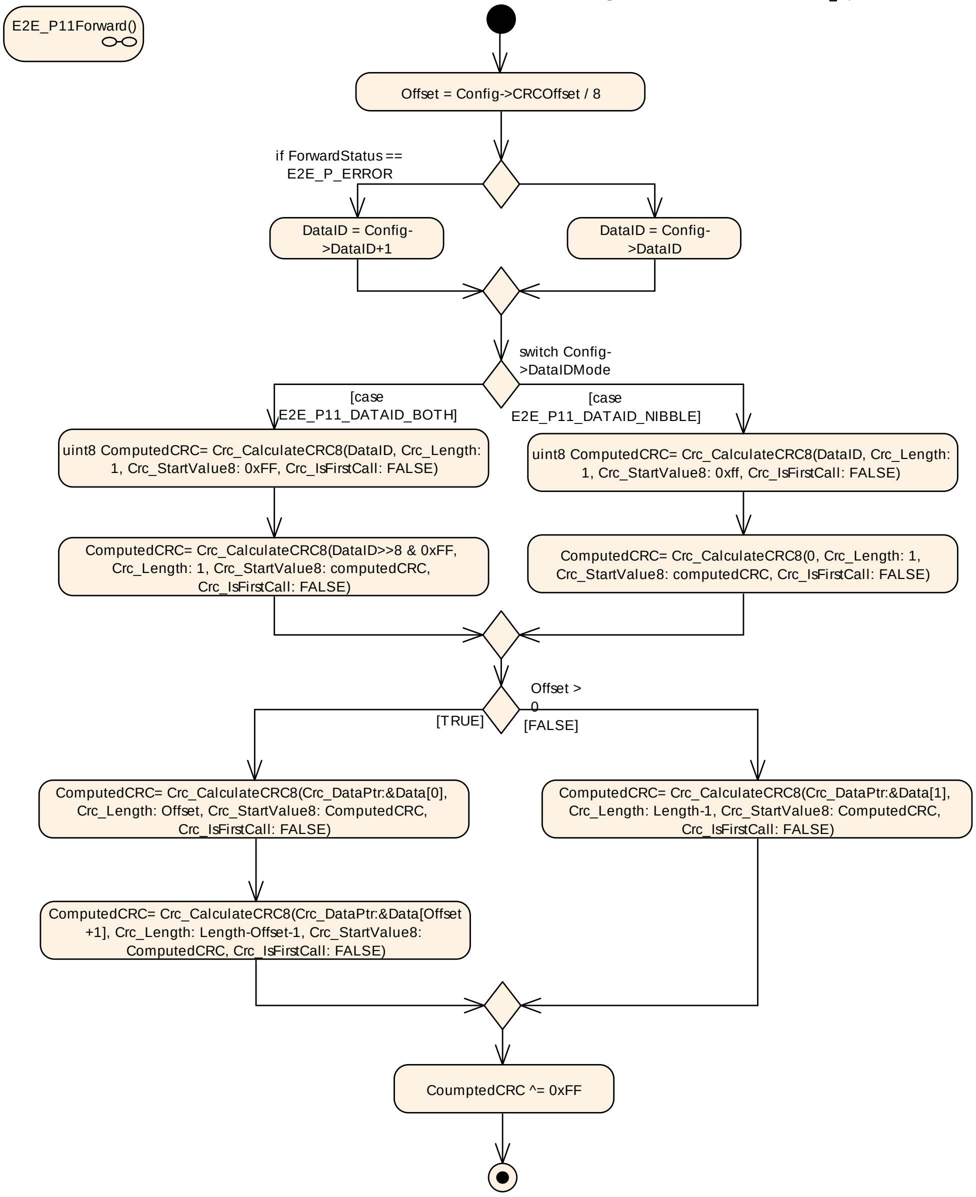

Figure 6.105: E2E Profile 11 Forward step "Compute CRC"

6.12.3 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 6.106.⌋*(RS_E2E_08539)*

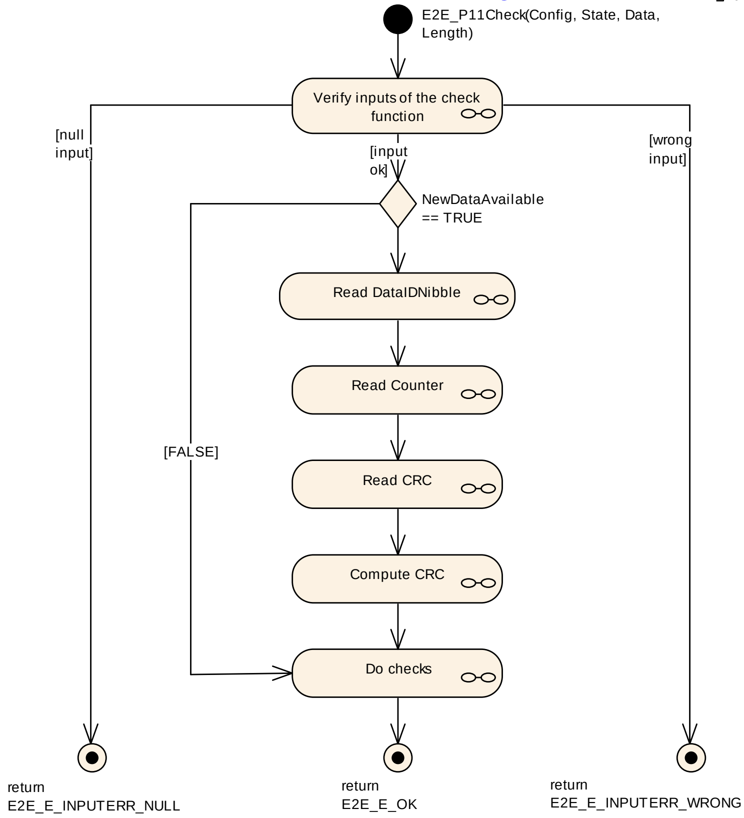

Figure 6.106: E2E Profile 11 Check

[PRS_E2E_00517] ⌈The step "Verify inputs of the check function" in E2E_P11Check() shall behave as shown in Figure 6.107.⌋*(RS_E2E_08539)*

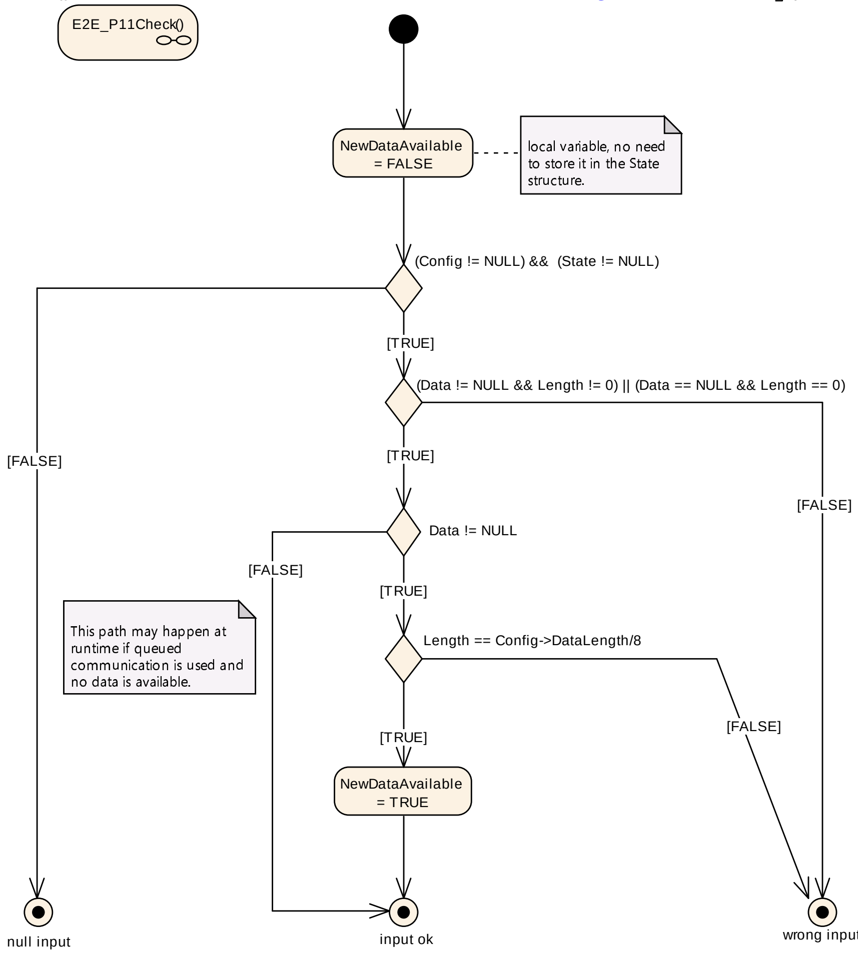

Figure 6.107: E2E Profile 11 Check step "Verify inputs of the check function"

[PRS_E2E_00582] ⌈The step "Read DataIDNibble" in E2E_P11Check() shall behave as shown in Figure 6.108.⌋*(RS_E2E_08539)*

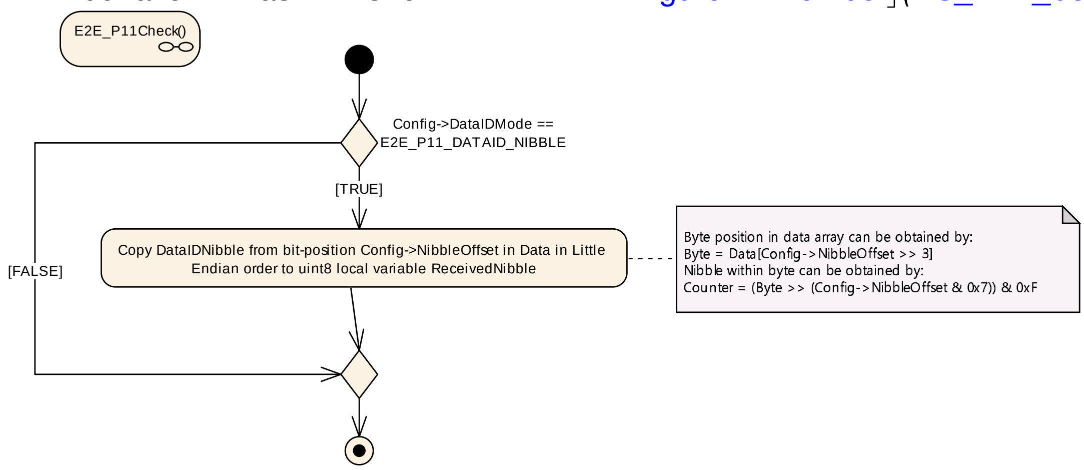

Figure 6.108: E2E Profile 11 Check step "Read DataIDNibble"

[PRS_E2E_00518] ⌈The step "Read Counter" in E2E_P11Check() shall behave as shown in Figure 6.109.⌋*(RS_E2E_08539)*

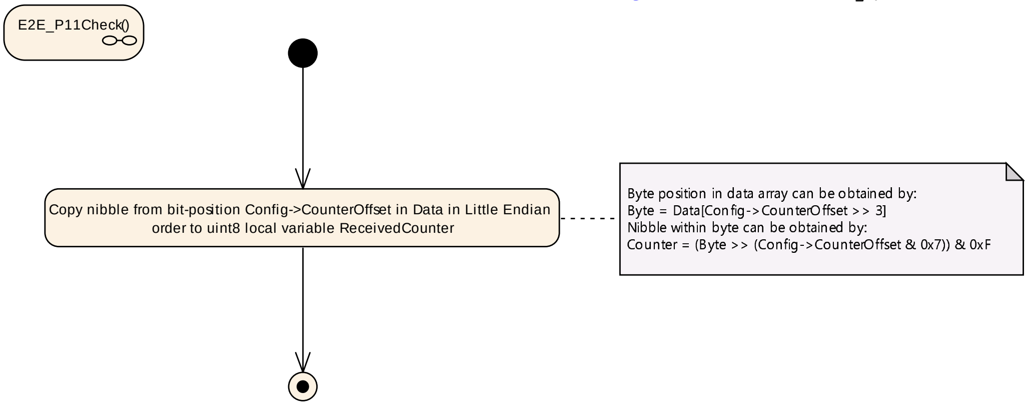

Figure 6.109: E2E Profile 11 Check step "Read Counter"

[PRS_E2E_00519] ⌈The step "Read CRC" in E2E_P11Check() shall behave as shown in Figure 6.110.⌋*(RS_E2E_08539)*

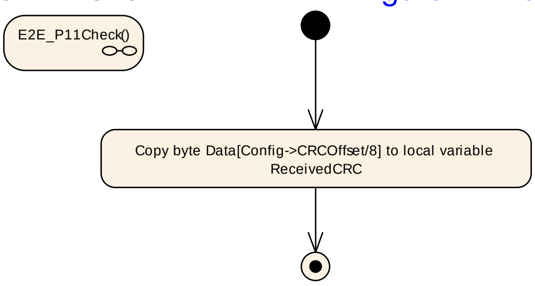

Figure 6.110: E2E Profile 11 Check step "Read CRC"

[PRS_E2E_00521] ⌈The step ”Do Checks’ in E2E_P11Check() shall behave as shown in Figure 6.111.⌋*(RS_E2E_08539)*

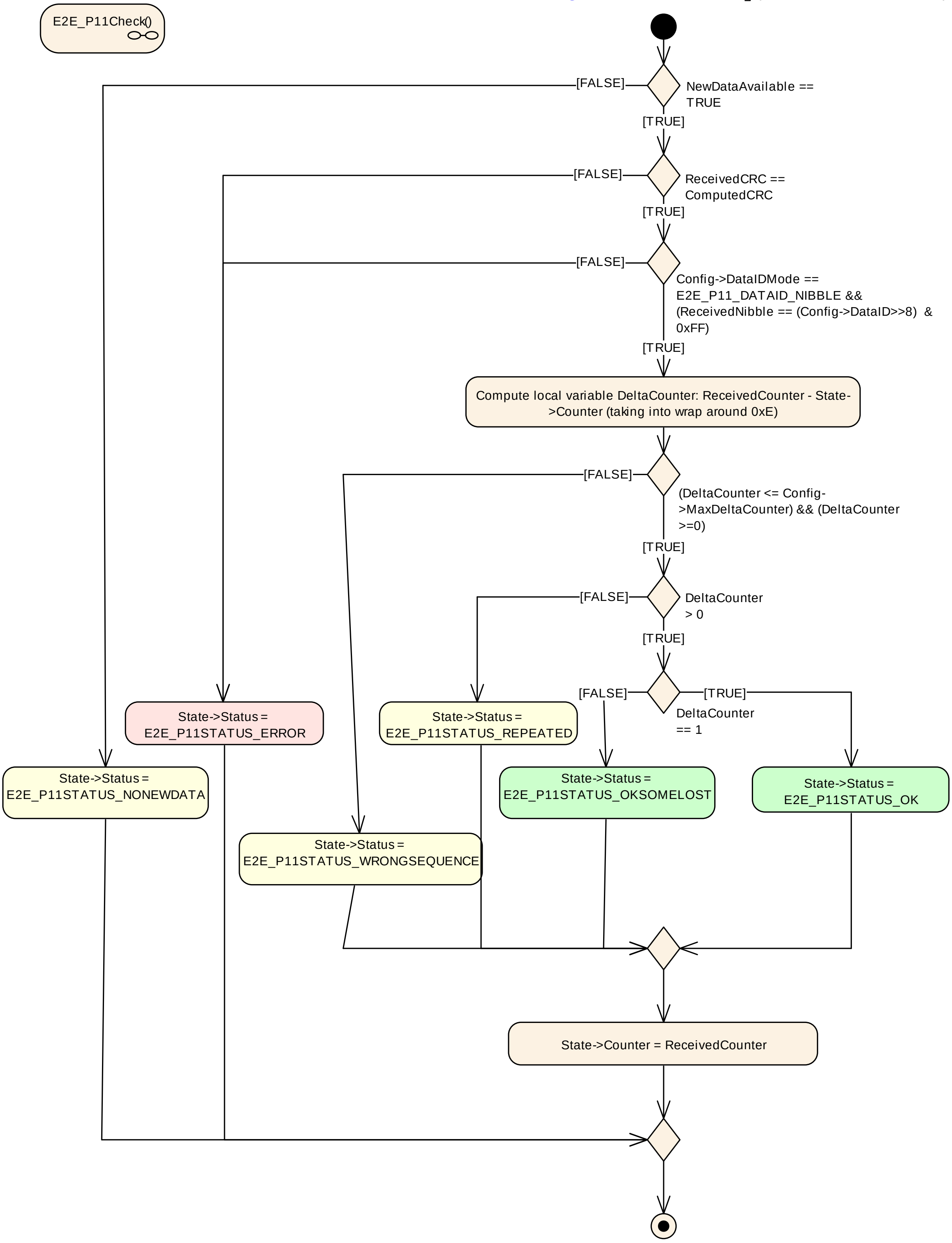

Figure 6.111: E2E Profile 11 Check step ”Do Checks”

6.12.4 Profile 11 Data Types

6.12.4.1 Profile 11 Protect State Type

[PRS_E2E_00661] ⌈The E2E_P11Protect and E2E_P11Forward functions 'State' shall have the members defined in (see Table 6.82).⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter to be used for protecting the next Data. The initial value is 0, which means that in the first cycle, Counter is 0. Each time E2E_P11Protect() is called, it increments the counter up to 0xFF.

Table 6.82: E2E Profile 11 Protect State Type

6.12.4.2 Profile 11 Check Status Type

[PRS_E2E_00662] ⌈The E2E_P11Check functions 'State' shall have the members defined in (see Table 6.83).⌋*(RS_E2E_08528)*

Member Name	Type	Description
Counter	Unsigned Integer	Counter of the data in previous cycle.
Status	Enumeration	Result of the verification of the Data in this cycle, determined by the Check function.

Table 6.83: E2E Profile 11 Check Status Type

6.12.4.3 Profile 11 Check Status Enumeration

[PRS_E2E_00594] ⌈The E2E_P11Check functions 'State->Status' shall consist of the following enumeration values (see Table 6.84).⌋*(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 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 6.84: E2E Profile 11 Check Status Enumeration

6.12.4.4 Profile 11 Configuration Type

[PRS_E2E_00663] ⌈The E2E_P11Protect, E2E_P11Forward and E2E_P11Check functions 'Config' shall have the following members defined in (see Table 6.85).⌋*(RS_E2E_08528, RS_E2E_08537)*

Member Name	Type	Description
CounterOffset	Unsigned Integer	Bit offset of Counter in MSB first order. In variants 1A and 1B, CounterOffset is 8. The offset shall be a multiple of 4.
CRCOffset	Unsigned Integer	Bit offset of CRC (i.e. since *Data) in MSB first order. The offset shall be a multiple of 8. In variants 11A and 11C, CRCOffset is 0.
DataID	Unsigned Integer	A unique identifier, for protection against masquerading. There are some constraints on the selection of ID values, described in section "Configuration constraints on Data IDs".
DataIDNibbleOffset	Unsigned Integer	Bit offset of the low nibble of the high byte of Data ID.
DataIDMode	Enumeration	Inclusion mode of ID in CRC computation (both bytes, alternating, or low byte only of ID included).
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data. For example, if the receiver gets Data with counter 1 and MaxDeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
DataLength	Unsigned Integer	Length of data, in bits. The value shall be a multiple of 8 and shall be <= 256.

Table 6.85: E2E Profile 11 Configuration Type

6.12.5 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 6.86: E2E Profile 11 protocol example configuration

E2E_P11ProtectStateType field	Value
Counter	0

Table 6.87: E2E Profile 11 example state initialization

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

Table 6.88: 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 6.89: E2E Profile 11 protect result DataIDMode = E2E_P11DATAID_BOTH, counter 1

6.12.5.1 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 6.90: 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 6.91: E2E Profile 11 protect result DataIDMode = E2E_P11DATAID_NIBBLE, counter 1

6.12.5.2 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. The CRC calculation includes the 8 byte header and then the payload data, excluding the CRC byte itself. "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 6.92: E2E Profile 11 example protect result with short data and SOME/IP

6.13 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 6.93).⌋*(RS_E2E_08527, RS_E2E_08529, RS_E2E_08530, RS_E2E_08533, RS_E2E_08543, RS_E2E_08544, RS_E2E_08545, RS_E2E_08546, RS_E2E_08547, RS_E2E_08548, RS_E2E_08549)*

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 shall be unique system-wide.

Table 6.93: 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 basis and timeout monitoring using E2E-Library [5]	Loss, delay, blocking
Data ID + CRC	Masquerade and incorrect addressing, insertion
CRC	Corruption, asymmetric information [6]

Table 6.94: Detectable communication faults using Profile 22

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

6.13.1 Header 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 shall 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.

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 6.112: 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).

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

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 first) applicable for both implicit and explicit header fields - imposed by profile
2. MSB First (most significant bit within byte first) - imposed by Flexray/CAN bus.

6.13.1.1 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.

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

6.13.1.2 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.

6.13.1.3 Length

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

6.13.1.4 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 byte), including the user data extended at the end with the corresponding Data ID from the Data ID List.⌋*(RS_E2E_08539, RS_E2E_08531)*

6.13.2 Creation of E2E-Header

6.13.2.1 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 6.113.⌋*(RS_E2E_08539)*

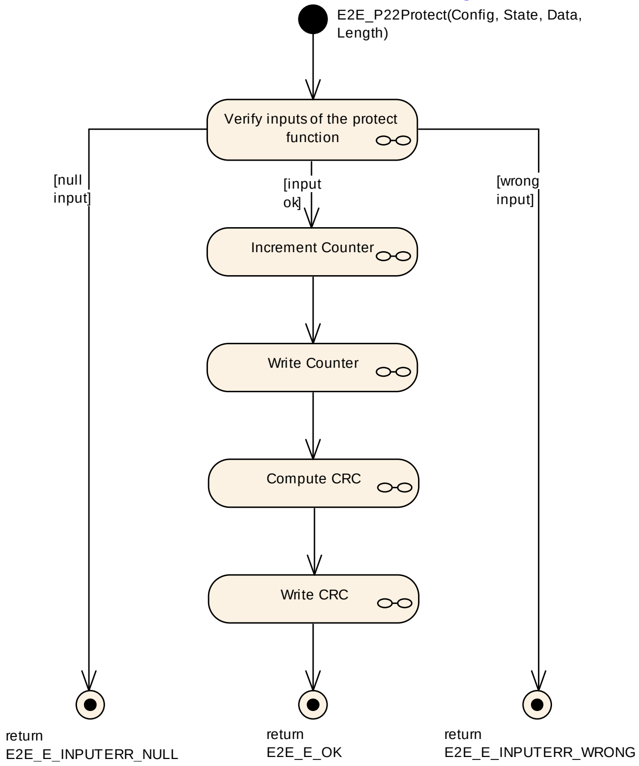

Figure 6.113: E2E Profile 22 Protect

[PRS_E2E_00529] ⌈The step "Verify inputs of the protect function" in E2E_P22Protect() shall behave as shown in Figure 6.114.⌋*(RS_E2E_08539)*

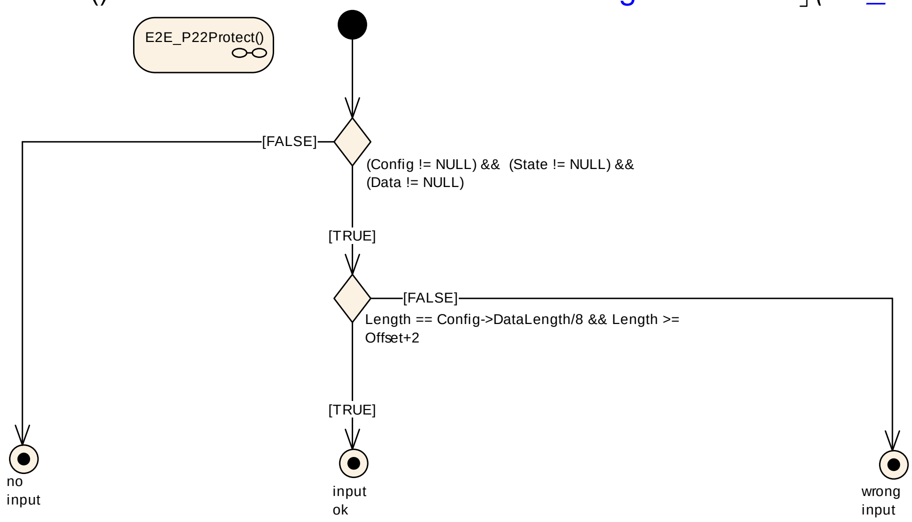

Figure 6.114: E2E Profile 22 Protect step "Verify inputs of the protect function"

[PRS_E2E_00530] ⌈The step "Write Counter" in E2E_P22Protect() and E2E_P22Forward() shall behave as shown in Figure 6.115.⌋*(RS_E2E_08539)*

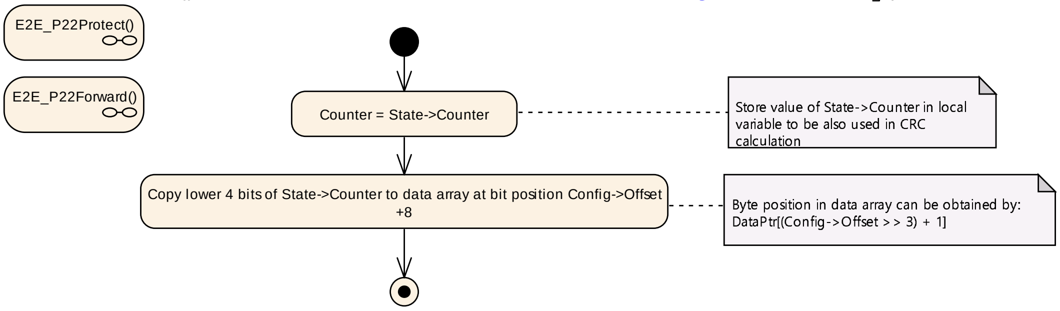

Figure 6.115: E2E Profile 22 Protect step "Write Counter"

[PRS_E2E_00531] ⌈The step "Compute CRC" in E2E_P22Protect() and in E2E_P22Check shall behave as shown in Figure 6.116.⌋*(RS_E2E_08539)*

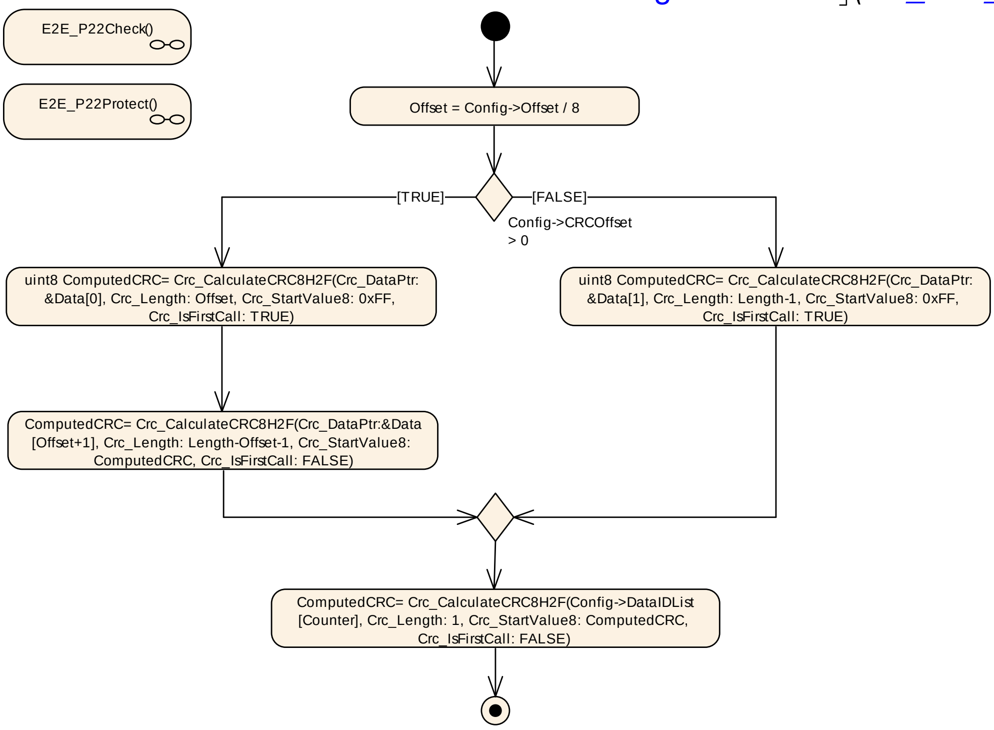

Figure 6.116: E2E Profile 22 Protect and Check step "Compute CRC"

[PRS_E2E_00532] ⌈The step "Write CRC" in E2E_P22Protect() and E2E_P22Forward() shall behave as shown in Figure 6.117.⌋*(RS_E2E_08539)*

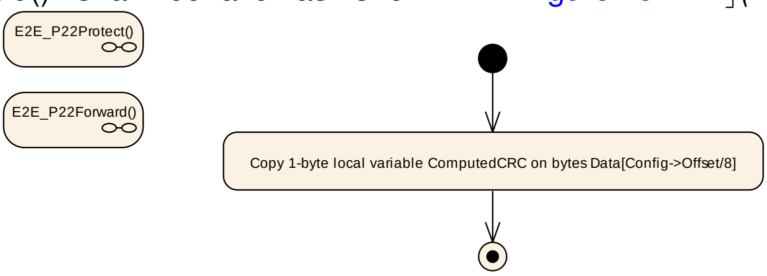

Figure 6.117: E2E Profile 22 Protect and Forward step "Write CRC"

[PRS_E2E_00533] ⌈The step "Increment Counter" in E2E_P22Protect() shall behave as shown in Figure 6.118.⌋*(RS_E2E_08539)*

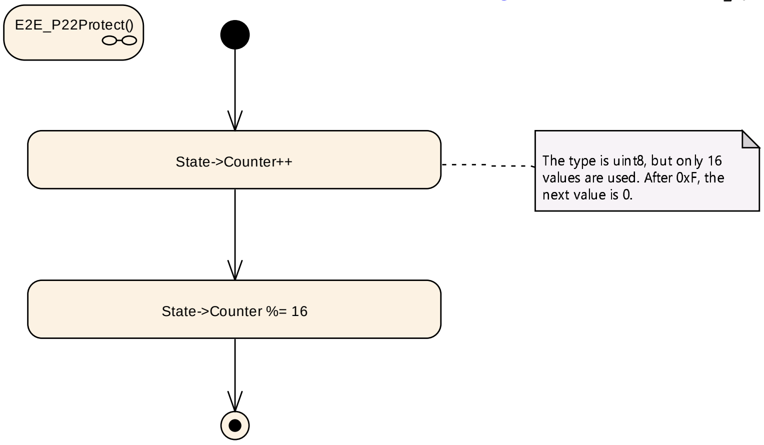

Figure 6.118: E2E Profile 22 Protect step "Increment Counter"

6.13.2.2 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] ⌈The function E2E_P22Forward() shall have the overall behavior as shown in Figure 6.119.⌋*(RS_E2E_08539)*

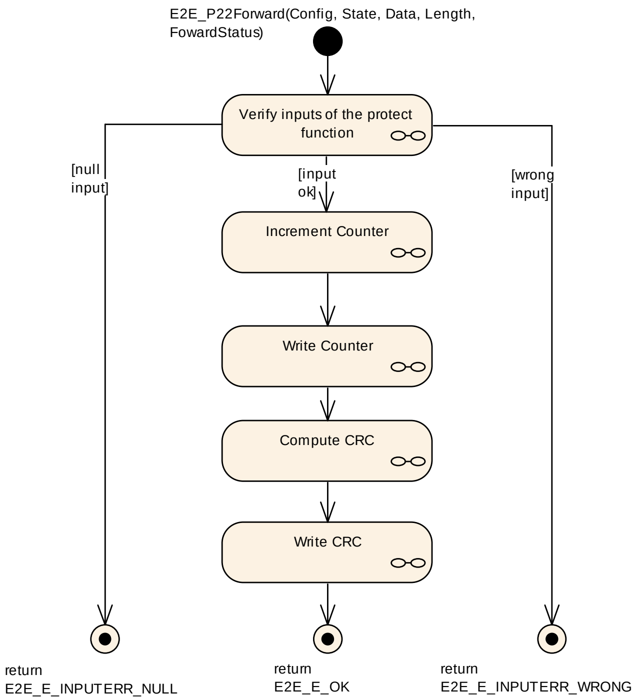

Figure 6.119: E2E Profile 22 Forward

Following steps are described in Section in Section 6.13.2.1

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

[PRS_E2E_00636] ⌈The step ”Verify inputs of the forward function” in E2E_P22Forward() shall behave as shown in Figure 6.120.⌋*(RS_E2E_08539)*

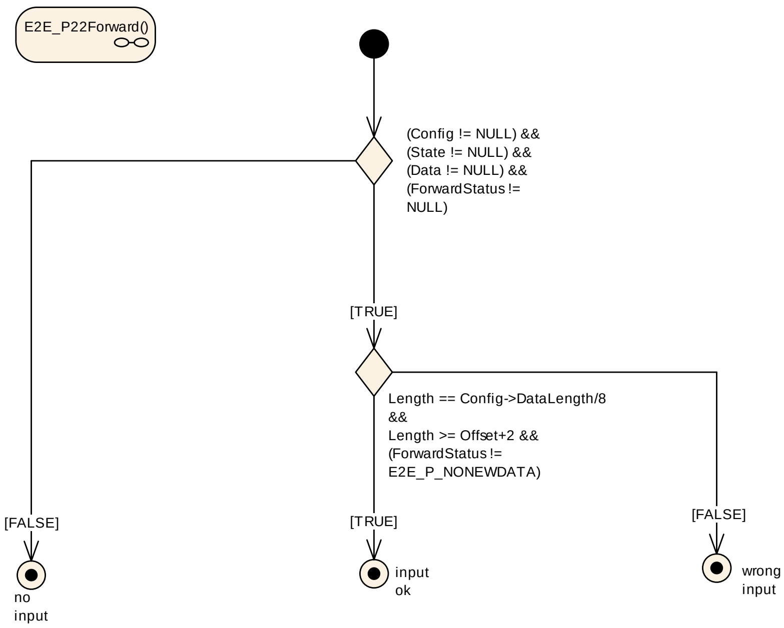

Figure 6.120: E2E Profile 22 Forward step ”Verify inputs of the forward function”

[PRS_E2E_00637] ⌈The step ”Increment Counter” in E2E_P22Forward() shall behave as shown in Figure 6.121.⌋*(RS_E2E_08539)*

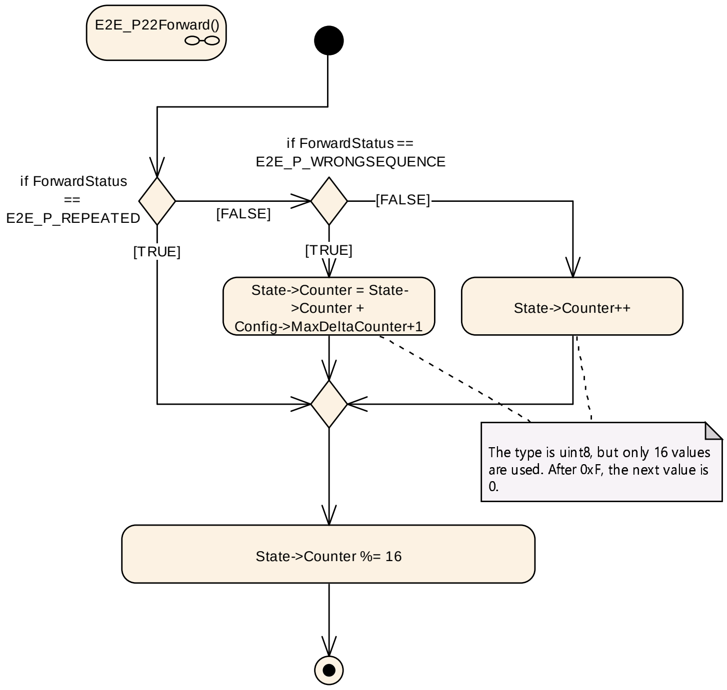

Figure 6.121: E2E Profile 22 Forward step ”Increment Counter”

[PRS_E2E_00638] ⌈The step ”Compute CRC” in E2E_P22Forward() shall behave as shown in Figure 6.122.⌋*(RS_E2E_08539)*

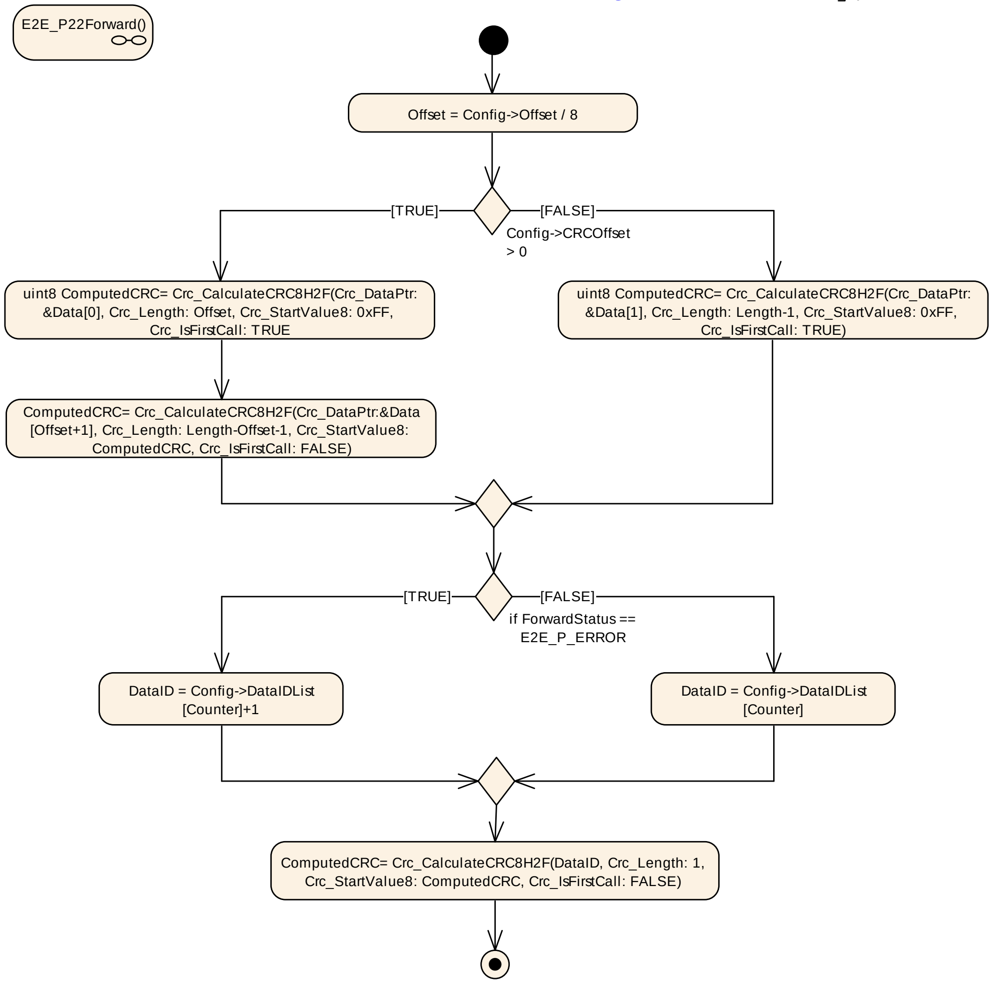

Figure 6.122: E2E Profile 22 Forward step ”Compute CRC”

6.13.3 Evaluation of E2E-Header

6.13.3.1 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 6.123.⌋*(RS_E2E_08539)*

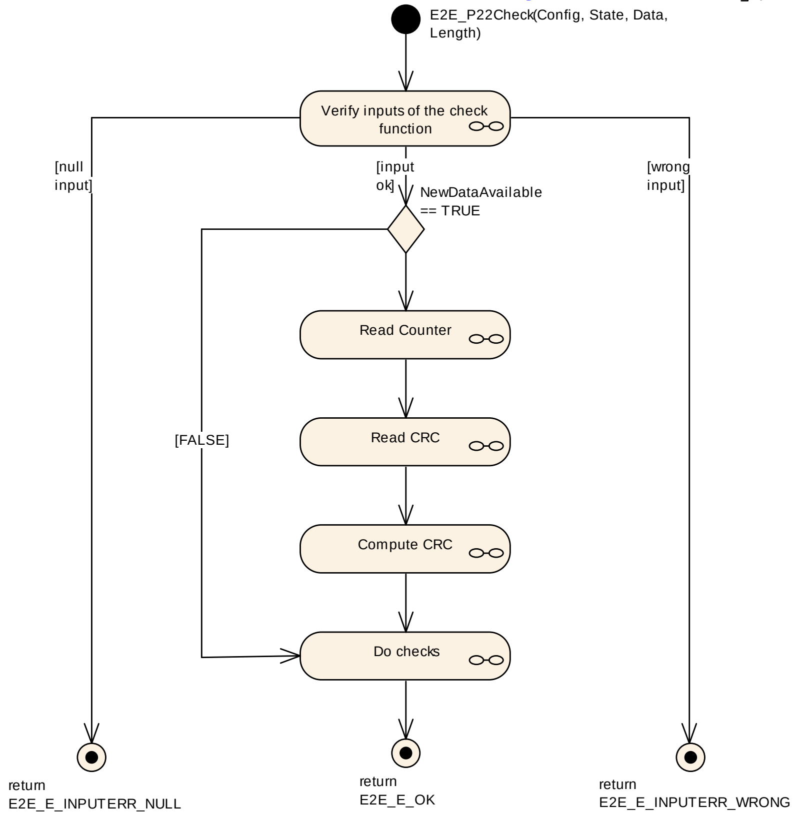

Figure 6.123: E2E Profile 22 Check

[PRS_E2E_00535] ⌈The step ”Verify inputs of the check function” in E2E_P22Check() shall behave as shown in Figure 6.124.⌋*(RS_E2E_08539)*

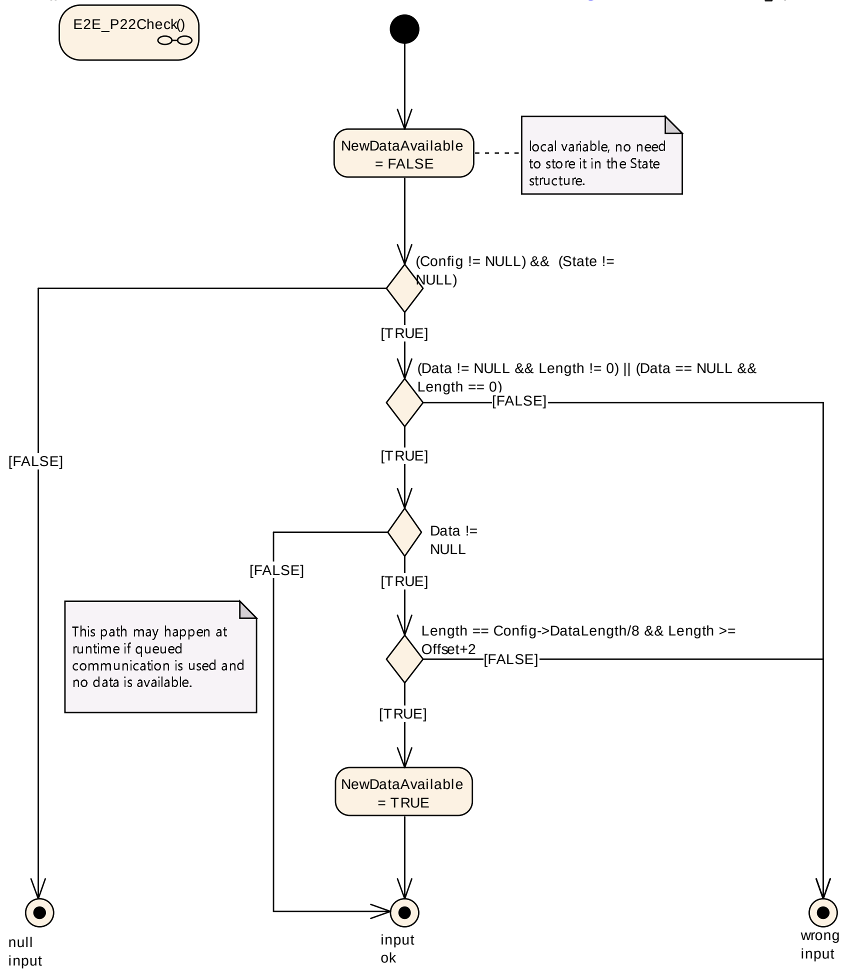

Figure 6.124: E2E Profile 22 Check step ”Verify inputs of the check function”

[PRS_E2E_00536] ⌈The step ”Read Counter” in E2E_P22Check() shall behave as shown in Figure 6.125.⌋*(RS_E2E_08539)*

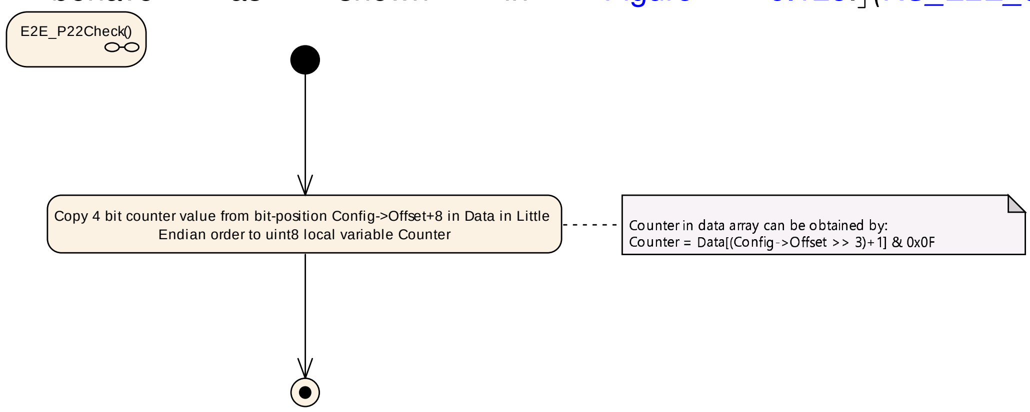

Figure 6.125: E2E Profile 22 Check step ”Read Counter”

[PRS_E2E_00537] ⌈The step "Read CRC" in E2E_P22Check() shall behave as shown in Figure 6.126.⌋*(RS_E2E_08539)*

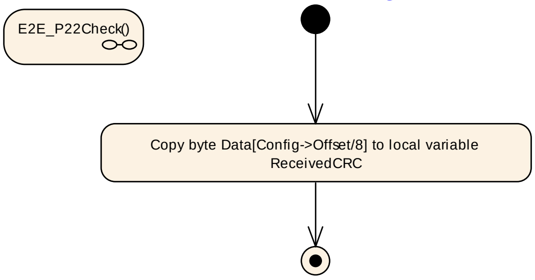

Figure 6.126: E2E Profile 22 Check step "Read CRC"

[PRS_E2E_00539] ⌈The step "Do Checks' in E2E_P22Check() shall behave as shown in Figure 6.127.⌋*(RS_E2E_08539)*

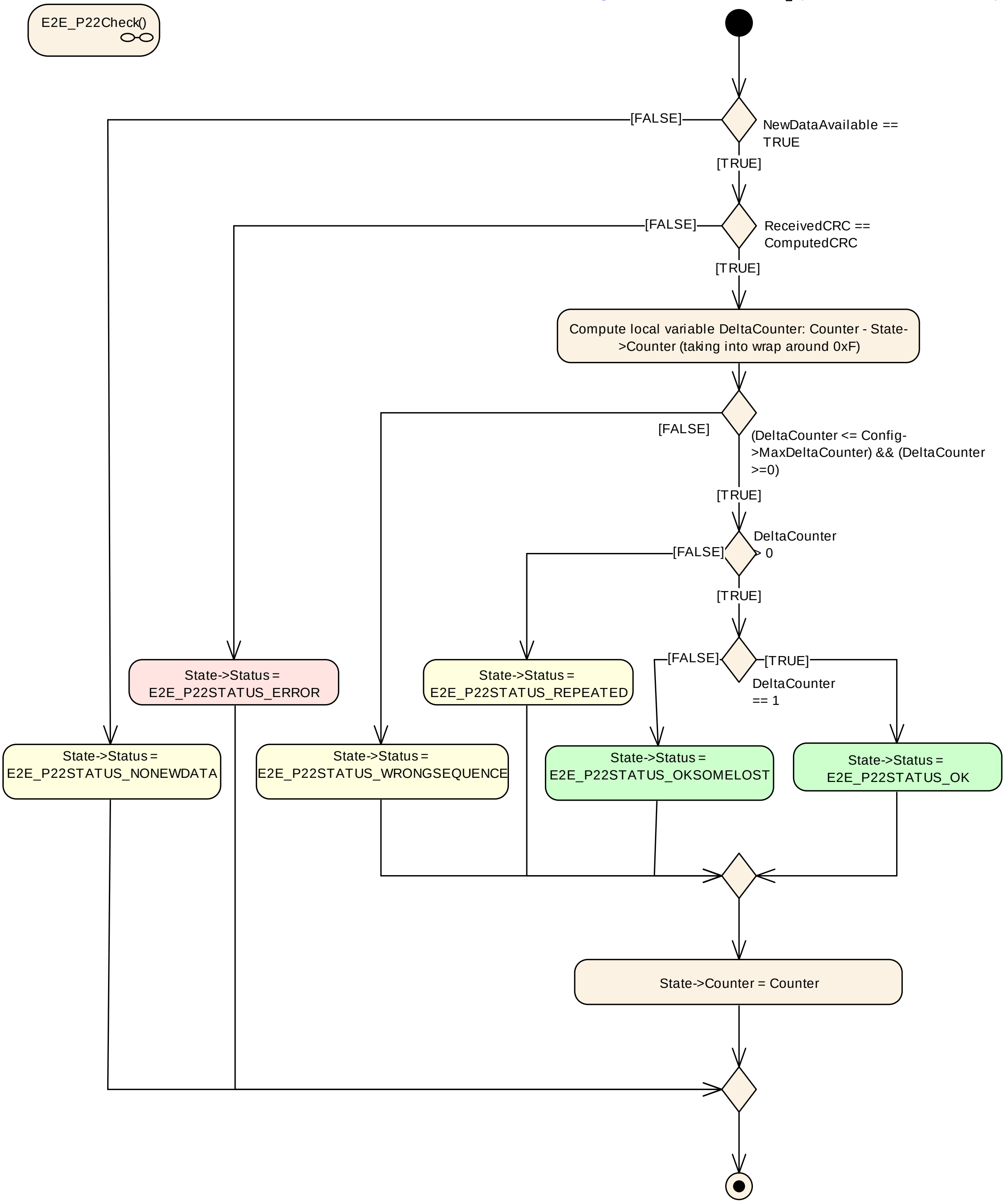

Figure 6.127: E2E Profile 22 Check step "Do Checks"

6.13.4 Profile 22 Data Types

6.13.4.1 Profile 22 Protect State Type

[PRS_E2E_00664] ⌈The E2E_P22Protect and E2E_P22Forward functions 'State' shall have the members defined in (see Table 6.95).⌋*(RS_E2E_08528)*

Name	Type	Description
Counter	Unsigned Integer	Counter to be used for protecting the Data. The initial value is 0, which means that the first Data will have the counter 0. After the protection by the counter, the counter is incremented modulo 16.

Table 6.95: E2E Profile 22 Protect State Type

6.13.4.2 Profile 22 Check Status Type

[PRS_E2E_00665] ⌈The E2E_P22Check functions 'State' shall have the members defined in (see Table 6.96).⌋*(RS_E2E_08528)*

Member Name	Type	Description
Counter	Unsigned Integer	Counter of the data in previous cycle.
Status	Enumeration	Result of the verification of the Data in this cycle, determined by the Check function.

Table 6.96: E2E Profile 22 Check Status Type

6.13.4.3 Profile 22 Check Status Enumeration

[PRS_E2E_00595] ⌈The E2E_P22Check functions 'State->Status' shall consist of the following enumeration values (see Table 6.97).⌋*(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 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 6.97: E2E Profile 22 Check Status Enumeration

6.13.4.4 Profile 22 Configuration Type

[PRS_E2E_00666] ⌈The E2E_P22Protect, E2E_P22Forward and E2E_P22Check functions 'Config' shall have the following members defined in (see Table 6.98).⌋*(RS_E2E_08528, RS_E2E_08537)*

Member Name	**Type**	**Description**
DataLength	Unsigned Integer	Length of Data, in bits. The value shall be a multiple of 8.
DataIDList	Unsigned Integer Array	An array of appropriately chosen Data IDs for protection against masquerading.
MaxDeltaCounter	Unsigned Integer	Initial maximum allowed gap between two counter values of two consecutively received valid Data.

Offset	Unsigned Integer	Offset of the E2E header in the Data[] array in bits. It shall be: 0 <= Offset <= MaxDataLength-(2*8).

Table 6.98: E2E Profile 22 Configuration Type

6.13.5 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 6.99: E2E Profile 22 protocol example configuration

E2E_P22ProtectStateType field	Value
Counter	0

Table 6.100: 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

▽

△

Counter	Byte							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
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 6.101: E2E Profile 22 example protect result

6.13.5.1 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 6.102: E2E Profile 2 example protect result with short data and SOME/IP

6.14 Specification of E2E Profile 44

[PRS_E2E_00707] ⌈Profile 44 shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID (see Table 6.103).⌋*(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 0xF4ACFB13, 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 6.103: E2E Profile 44 mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P44LENGTH_POS	0
P44LENGTH_LEN	2
P44COUNTER_POS	2
P44COUNTER_LEN	2
P44DATAID_POS	4
P44DATAID_LEN	4
P44CRC_POS	8
P44CRC_LEN	4
P44CALCULATE_CRC	Crc_CalculateCRC32P4()

Table 6.104: Profile 44-specific data

For behavior and flowcharts of E2E Profile 44 see section 6.3.

6.14.1 Header Layout

In the E2E Profile 44, the user data layout (of the data to be protected) is not constrained by E2E Profile 44 - there is only a requirement that the length of data to be protected is multiple of 1 byte.

The header of the E2E Profile 44 has one fixed layout, as follows:

<table>
<tr><td rowspan="2">Transmission order</td><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>0</td><td colspan="16">E2E Length</td><td colspan="16">E2E Counter</td></tr>
<tr><td>4</td><td colspan="32">E2E Data ID</td></tr>
<tr><td>8</td><td colspan="32">E2E CRC</td></tr>
</table>

Figure 6.128: E2E Profile 44 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 first) - imposed by profile
2. LSB First (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): 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.

6.14.2 Profile 44 Configuration Type

[PRS_E2E_00735] ⌈The E2E_P44Protect, E2E_P44Forward and E2E_P44Check functions 'Config' shall have the following members defined in Table 6.105. The current DataLength shall be a multiple of 8 as well as the MaxDataLength and MinDataLength.⌋ *(RS_E2E_08528, RS_E2E_08537)*

Member Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(12*8). Example: If Offset equals 8, then the high byte of the E2E Length (16 bit) is written to Byte 1, the low Byte is written to Byte 2.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that Length is >= MinDataLength. The value shall be >= 12*8 and <= MaxDataLength
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2Echecks that DataLength is <= MaxDataLength. The value shall be <= 65535*8 and >= MinDataLength.
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.105: E2E Profile 44 Configuration Type

6.14.3 E2E Profile 44 Protocol Examples

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

E2E_P44ConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000 0000
MinDataLength	96
MaxDataLength	524280
MaxDeltaCounter	1

Table 6.106: E2E Profile 44 protocol example configuration

E2E_P44ProtectStateType field	Value
Counter	0

Table 6.107: E2E Profile 44 example state initialization

Result data of E2E_P44Protect() with short data length (length 16 bytes, means 4 actual data bytes), offset = 0, counter = 0:

Byte	1	2	3	4	5	6	7	8
Data	0x00	0x10	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Data	0x86	0x2b	0x05	0x56	0x00	0x00	0x00	0x00
Field	*CRC*				*Data*			

Table 6.108: E2E Profile 44 example short

Result data of E2E_P44Protect() with minimum data length (4 data bytes), offset = 64 (as with SOME/IP header use case), datalength = 24, counter = 0:

Byte	1	2	3	4	5	6	7	8
Data	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Field	*Data (upper header)*							
Byte	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Data	0x00	0x18	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Data	0x69	0xd7	0x50	0x2e	0x00	0x00	0x00	0x00
Field	*CRC*				*Data*			

Table 6.109: E2E Profile 44 example short with SOME/IP use case

6.15 Specification of E2E Profile 4m

[PRS_E2E_00740] ⌈Profile 4m shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID, Source ID, Message Type, Message Result (see Table 6.110).⌋*(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 0xF4ACFB13, 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.
Message Type	2-bits, request (0) vs. response (1).
Message Result	2-bits, OK (0) vs. error (1) - fixed to OK (0) for Message Type ‘request’.
Source ID	28-bits, unique system-wide.

Table 6.110: E2E Profile 4m mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P04MLENGTH_POS	0
P04MLENGTH_LEN	2
P04MCOUNTER_POS	2
P04MCOUNTER_LEN	2
P04MDATAID_POS	4
P04MDATAID_LEN	4
P04MCRC_POS	8
P04MCRC_LEN	4
P04MMESSAGE_POS	12
P04MCALCULATE_CRC	Crc_CalculateCRC32P4()

Table 6.111: Profile 4m-specific data

For behavior and flowcharts of E2E method Profile 4m see section 6.4.

6.15.1 Header Layout

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

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

<table>
<tr><th colspan="33">Transmission Order</th></tr>
<tr><th rowspan="2">Pos</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>00</th><th>01</th><th>02</th><th>03</th><th>04</th><th>05</th><th>06</th><th>07</th><th>08</th><th>09</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td colspan="16">E2E Length</td><td colspan="16">E2E Counter</td></tr>
<tr><td>4</td><td colspan="32">E2E Data ID</td></tr>
<tr><td>8</td><td colspan="32">E2E CRC</td></tr>
<tr><td>12</td><td colspan="2">T</td><td colspan="2">R</td><td colspan="28">E2E Source ID</td></tr>
</table>

Table 6.112: E2E Profile 4m Header

Hereby ‘T’ denotes the E2E Message Type and ‘R’ denotes the E2E Message Result. 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 first) - imposed by profile
2. LSB First (least significant bit within byte first) - imposed by TCP/IP network

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.

6.15.2 Profile 4m Configuration Type

[PRS_E2E_00852] ⌈The E2E_P04mProtect, E2E_P04mForward, E2E_P04mSourceCheck and E2E_P04mSinkCheck functions “Config” shall have the following members defined in Table 6.113.⌋*(RS_E2E_08528, RS_E2E_08537)*

Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(16*8). Example: If Offset equals 8, then the high byte of the E2E Length (16 bit) is written to Byte 1, the low Byte is written to Byte 2.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that DataLength is >= MinDataLength. The value shall be <= 4096*8 (4kB) and shall be >= 16*8
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MaxDataLength. The value shall be <= 4096*8 (4kB) and it shall be >= MinDataLength

MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.113: E2E Profile 4m Configuration Type

6.15.3 E2E Profile 4m Protocol Examples

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

E2E_P04mConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000
MinDataLength	128
MaxDataLength	32768
MaxDeltaCounter	1

Table 6.114: E2E Profile 4m protocol example configuration

E2E_P04mProtectStateType field	Value
Counter	0

Table 6.115: E2E Profile 4m example state initialization

6.15.4 Request Example

The result data of an E2E_P04mProtect() call with short data length (length 20 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_REQUEST, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.116.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x00	0x14	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0xae	0x67	0x4c	0xa0	0x00	0x12	0x34	0x56
Field	*CRC*				*Message Type, Message Result, Source ID*			
Byte	**16**	**17**	**18**	**19**	**n/a**			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.116: E2E Profile 4m example short - request (Message Type = STD_MESSAGETYPE_REQUEST; Message Result = STD_MESSAGERESULT_OK)

6.15.5 Response Example

The result data of an E2E_P04mProtect() call with short data length (length 20 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.117.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x00	0x14	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x85	0x25	0x76	0x19	0x40	0x12	0x34	0x56
Field	*CRC*				*Message Type, Message Result, Source ID*			
Byte	**16**	**17**	**18**	**19**	**n/a**			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.117: E2E Profile 4m example short - normal response (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_OK)

6.15.6 Error Response Example

The result data of an E2E_P04mProtect() call with short data length (length 20 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_ERROR is depicted in Table 6.118.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x00	0x14	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x23	0x45	0x57	0x0f	0x50	0x12	0x34	0x56
Field	*CRC*				*Message Type, Message Result, Source ID*			
Byte	**16**	**17**	**18**	**19**	**n/a**			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.118: E2E Profile 4m example short - ERROR reponse (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_ERROR)

6.16 Specification of E2E Profile 7m

[PRS_E2E_00783] ⌈Profile 7m shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID, Source ID, Message Type, Message Result (see Table 6.119).⌋*(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.
Message Type	2-bits, request (0) vs. response (1).
Message Result	2-bits, OK (0) vs. error (1) - fixed to OK (0) for Message Type 'request'.
Source ID	28-bits, unique system-wide.

Table 6.119: E2E Profile 7m mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P07MLENGTH_POS	8
P07MLENGTH_LEN	4
P07MCOUNTER_POS	12
P07MCOUNTER_LEN	4
P07MDATAID_POS	16
P07MDATAID_LEN	4
P07MCRC_POS	0
P07MCRC_LEN	8
P07MMESSAGE_POS	20
P07MCALCULATE_CRC	Crc_CalculateCRC64()

Table 6.120: Profile 8m-specific data

For behavior and flowcharts of E2E method Profile 7m see section 6.4.

6.16.1 Header Layout

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

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

<table>
<tr><th colspan="33">Transmission Order</th></tr>
<tr><th rowspan="2">Pos</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>00</th><th>01</th><th>02</th><th>03</th><th>04</th><th>05</th><th>06</th><th>07</th><th>08</th><th>09</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td colspan="32">E2E CRC</td></tr>
<tr><td>4</td><td colspan="32">E2E CRC</td></tr>
<tr><td>8</td><td colspan="32">E2E Length</td></tr>
<tr><td>12</td><td colspan="32">E2E Counter</td></tr>
<tr><td>16</td><td colspan="32">E2E Data ID</td></tr>
<tr><td>20</td><td colspan="2">T</td><td colspan="2">R</td><td colspan="28">E2E Source ID</td></tr>
</table>

Table 6.121: E2E Profile 7m Header

Hereby ‘T’ denotes the E2E Message Type and ‘R’ denotes the E2E Message Result. 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 first) - imposed by profile
2. LSB First (least significant bit within byte first) - imposed by TCP/IP network

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.

6.16.2 Profile 7m Configuration Type

[PRS_E2E_00851] ⌈The E2E_P07mProtect, E2E_P07mForward, E2E_P07mSourceCheck and E2E_P07mSinkCheck functions “Config” shall have the following members defined in Table 6.122.⌋ *(RS_E2E_08528, RS_E2E_08537)*

Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.

Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(24*8). Example: If Offset equals 8, then the first byte of the E2E Length (32 bit) is written to byte 1, the next byte is written to byte 2 and so on.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that DataLength is >= MinDataLength. The value shall be <= 4194304*8 (4MB) and shall be >= 24*8
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MaxDataLength. The value shall be <= 4194304*8 (4MB) and it shall be >= MinDataLength
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.122: E2E Profile 7m Configuration Type

6.16.3 E2E Profile 7m Protocol Examples

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

E2E_P07mConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000
MinDataLength	192
MaxDataLength	32768
MaxDeltaCounter	1

Table 6.123: E2E Profile 7m protocol example configuration

E2E_P07mProtectStateType field	Value
Counter	0

Table 6.124: E2E Profile 7m example state initialization

6.16.4 Request Example

The result data of an E2E_P07mProtect() call with short data length (length 28 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID =

0x0123456, Message Type = STD_MESSAGETYPE_REQUEST, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.125.

Byte	0	1	2	3	4	5	6	7
Data	0xae	0x96	0xa7	0xd0	0xa5	0x01	0x75	0x94
Field	*CRC*							
Byte	8	9	10	11	12	13	14	15
Data	0x00	0x00	0x00	0x1c	0x00	0x00	0x00	0x00
Field	*Length*				*Counter*			
Byte	16	17	18	19	20	21	22	23
Data	0x0a	0x0b	0x0c	0x0d	0x00	0x12	0x34	0x56
Field	*DataID*				*Message Type, Message Result, Source ID*			
Byte	24	25	26	27	n/a			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.125: E2E Profile 7m example short - request (Message Type = STD_MESSAGETYPE_REQUEST; Message Result = STD_MESSAGERESULT_OK)

6.16.5 Response Example

The result data of an E2E_P07mProtect() call with short data length (length 28 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.126.

Byte	0	1	2	3	4	5	6	7
Data	0xa6	0x2d	0x64	0x86	0xe8	0x3f	0x2c	0xaf
Field	*CRC*							
Byte	8	9	10	11	12	13	14	15
Data	0x00	0x00	0x00	0x1c	0x00	0x00	0x00	0x00
Field	*Length*				*Counter*			
Byte	16	17	18	19	20	21	22	23
Data	0x0a	0x0b	0x0c	0x0d	0x40	0x12	0x34	0x56
Field	*DataID*				*Message Type, Message Result, Source ID*			
Byte	24	25	26	27	n/a			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.126: E2E Profile 7m example short - normal response (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_OK)

6.16.6 Error Response Example

The result data of an E2E_P07mProtect() call with short data length (length 28 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_ERROR is depicted in Table 6.127.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x09	0xd9	0xe8	0x0c	0x47	0x34	0x32	0x02
Field	*CRC*							
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x00	0x00	0x00	0x1c	0x00	0x00	0x00	0x00
Field	*Length*				*Counter*			
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0x0a	0x0b	0x0c	0x0d	0x50	0x12	0x34	0x56
Field	*DataID*				*Message Type, Message Result, Source ID*			
Byte	**24**	**25**	**26**	**27**	**n/a**			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.127: E2E Profile 7m example short - ERROR response (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_ERROR)

6.17 Specification of E2E Profile 8m

[PRS_E2E_01107] ⌈Profile 08m shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID (see Table 6.128).⌋*(RS_E2E_08529, RS_E2E_08530, RS_E2E_08533)*

Control field	**Description**
Length	32 bits, to support dynamic-size data.
Counter	32 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.
Message Type	2-bits, request (0) vs. response (1).
Message Result	2-bits, OK (0) vs. error (1) - fixed to OK (0) for Message Type 'request'.
Source ID	28-bits, unique system-wide.

Table 6.128: E2E Profile 8m mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P08MLENGTH_POS	4
P08MLENGTH_LEN	4
P08MCOUNTER_POS	8
P08MCOUNTER_LEN	4
P08MDATAID_POS	12
P08MDATAID_LEN	4
P08MCRC_POS	0
P08MCRC_LEN	4
P08MMESSAGE_POS	16
P08MCALCULATE_CRC	Crc_CalculateCRC32P4()

Table 6.129: Profile 8m-specific data

For behavior and flowcharts of E2E method Profile 8m see section 6.4.

6.17.1 Header Layout

In the E2E Profile 8m, the user data layout (of the data to be protected) is not constrained by E2E Profile 8m - there is only a requirement that the length of data to be protected is multiple of 1 byte.

The header of the E2E Profile 8m has one fixed layout, as follows:

<table>
<tr><th colspan="33">Transmission Order</th></tr>
<tr><th rowspan="2">Pos</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>00</th><th>01</th><th>02</th><th>03</th><th>04</th><th>05</th><th>06</th><th>07</th><th>08</th><th>09</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td colspan="32">E2E CRC</td></tr>
<tr><td>4</td><td colspan="32">E2E Length</td></tr>
<tr><td>8</td><td colspan="32">E2E Counter</td></tr>
<tr><td>12</td><td colspan="32">E2E Data ID</td></tr>
<tr><td>16</td><td colspan="2">T</td><td colspan="2">R</td><td colspan="28">E2E Source ID</td></tr>
</table>

Table 6.130: E2E Profile 8m 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 first) - imposed by profile
2. LSB First (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 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.

6.17.2 Profile 8m Configuration Type

[PRS_E2E_01154] ⌈The E2E_P08mProtect, E2E_P08mForward, and E2E_P08mCheck functions "Config" shall have the following members defined in Table 6.131.⌋*(RS_E2E_08528)*

Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(20*8). Example: If Offset equals 8, then the first byte of the E2E Length (32 bit) is written to byte 1, the next byte is written to byte 2 and so on.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that DataLength is >= MinDataLength. The value shall be >= 20*8 and <= MaxDataLength
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MaxDataLength. The value shall be <= 4294967295 >= MinDataLength
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.131: E2E Profile 8m Configuration Type

6.17.3 E2E Profile 8m Protocol Examples

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

E2E_P08mConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000
MinDataLength	160
MaxDataLength	32768

MaxDeltaCounter	1

Table 6.132: E2E Profile 8m protocol example configuration

E2E_P08mProtectStateType field	Value
Counter	0

Table 6.133: E2E Profile 8m example state initialization

6.17.4 Request Example

The result data of an E2E_P08mProtect() call with short data length (length 24 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_REQUEST, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.134.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x6d	0xf0	0x5e	0xba	0x00	0x00	0x00	0x18
Field	*CRC*				*Length*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x00	0x00	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Counter*				*DataID*			
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0x00	0x12	0x34	0x56	0x00	0x00	0x00	0x00
Field	*Message Type, Message Result, Source ID*				*Data*			

Table 6.134: E2E Profile 8m example short - request (Message Type = STD_MESSAGETYPE_REQUEST; Message Result = STD_MESSAGERESULT_OK)

6.17.5 Response Example

The result data of an E2E_P08mProtect() call with short data length (length 24 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.135.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x46	0xb2	0x64	0x03	0x00	0x00	0x00	0x18
Field	*CRC*				*Length*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**

▽

△

Data	0x00	0x00	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Counter*				*DataID*			
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0x40	0x12	0x34	0x56	0x00	0x00	0x00	0x00
Field	*Message Type, Message Result, Source ID*				*Data*			

Table 6.135: E2E Profile 8m example short - normal response (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_OK)

6.17.6 Error Response Example

The result data of an E2E_P08mProtect() call with short data length (length 24 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_ERROR is depicted in Table 6.136.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0xe0	0xd2	0x45	0x15	0x00	0x00	0x00	0x18
Field	*CRC*				*Length*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x00	0x00	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Counter*				*DataID*			
Byte	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Data	0x50	0x12	0x34	0x56	0x00	0x00	0x00	0x00
Field	*Message Type, Message Result, Source ID*				*Data*			

Table 6.136: E2E Profile 8m example short - normal response (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_OK)

6.18 Specification of E2E Profile 44m

[PRS_E2E_01155] ⌈Profile 44m shall provide the following control fields, transmitted at runtime together with the protected data: Length, Counter, CRC, Data ID, Source ID, Message Type, Message Result (see Table 6.137).⌋*(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.
Message Type	2-bits, request (0) vs. response (1).
Message Result	2-bits, OK (0) vs. error (1) - fixed to OK (0) for Message Type 'request'.
Source ID	28-bits, unique system-wide.

Table 6.137: E2E Profile 44m mechanisms

For details of CRC calculation, the usage of start values and XOR values see SWS_CRCLibrary[3].

The specification of the profile uses the following placeholders:

Placeholder	Replacement
P44MLENGTH_POS	0
P44MLENGTH_LEN	2
P44MCOUNTER_POS	2
P44MCOUNTER_LEN	2
P44MDATAID_POS	4
P44MDATAID_LEN	4
P44MCRC_POS	8
P44MCRC_LEN	4
P44MMESSAGE_POS	12
P44MCALCULATE_CRC	Crc_CalculateCRC32P4()

Table 6.138: Profile 44m-specific data

For behavior and flowcharts of E2E method Profile 44m see section 6.4.

6.18.1 Header Layout

In the E2E Profile 44m, the user data layout (of the data to be protected) is not constrained by E2E Profile 44m - there is only a requirement that the length of data to be protected is multiple of 1 byte.

The header of the E2E Profile 44m has one fixed layout, as follows:

<table>
<tr><th colspan="33">Transmission Order</th></tr>
<tr><th rowspan="2">Pos</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>00</th><th>01</th><th>02</th><th>03</th><th>04</th><th>05</th><th>06</th><th>07</th><th>08</th><th>09</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><th>0</th><td colspan="16">E2E Length</td><td colspan="16">E2E Counter</td></tr>
<tr><th>4</th><td colspan="32">E2E Data ID</td></tr>
<tr><th>8</th><td colspan="32">E2E CRC</td></tr>
<tr><th>12</th><td colspan="2">T</td><td colspan="2">R</td><td colspan="28">E2E Source ID</td></tr>
</table>

Table 6.139: E2E Profile 44m Header

Hereby 'T' denotes the E2E Message Type and 'R' denotes the E2E Message Result. 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 first) - imposed by profile
2. LSB First (least significant bit within byte first) - imposed by TCP/IP network

For example, the 16 bits of the E2E Counter are transmitted in the following order (higher number meaning higher significance): 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.

6.18.2 Profile 44m Configuration Type

[PRS_E2E_01202] ⌈The E2E_P44mProtect, E2E_P44mForward, and E2E_P44mCheck functions "Config" shall have the following members defined in Table 6.140.⌋ *(RS_E2E_08528)*

Name	Type	Description
DataID	Unsigned Integer	A system-unique identifier of the Data.
Offset	Unsigned Integer	Bit offset of the first bit of the E2E header from the beginning of the Data (bit numbering: bit 0 is the least important). The offset shall be a multiple of 8 and 0 <= Offset <= MaxDataLength-(16*8). Example: If Offset equals 8, then the high byte of the E2E Length (16 bit) is written to Byte 1, the low Byte is written to Byte 2.
MinDataLength	Unsigned Integer	Minimal length of Data, in bits. E2E checks that DataLength is >= MinDataLength. The value shall be <= 4096*8 (4kB) and shall be >= 16*8
MaxDataLength	Unsigned Integer	Maximal length of Data, in bits. E2E checks that DataLength is <= MaxDataLength. The value shall be <= 4096*8 (4kB) and it shall be >= MinDataLength
MaxDeltaCounter	Unsigned Integer	Maximum allowed gap between two counter values of two consecutively received valid Data.

Table 6.140: E2E Profile 44m Configuration Type

6.18.3 E2E Profile 44m Protocol Examples

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

E2E_P44mConfigType field	Value
DataID	0x0a0b0c0d
Offset	0x0000
MinDataLength	128
MaxDataLength	32768
MaxDeltaCounter	1

Table 6.141: E2E Profile 44m protocol example configuration

E2E_P44mProtectStateType field	Value
Counter	0

Table 6.142: E2E Profile 44m example state initialization

6.18.4 Request Example

The result data of an E2E_P44mProtect() call with short data length (length 20 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_REQUEST, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.143.

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Data	0x00	0x14	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Data	0x2e	0x82	0xb4	0x56	0x00	0x12	0x34	0x56
Field	*CRC*				*Message Type, Message Result, Source ID*			
Byte	**16**	**17**	**18**	**19**	**n/a**			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.143: E2E Profile 44m example short - request (Message Type = STD_MESSAGETYPE_REQUEST; Message Result = STD_MESSAGERESULT_OK)

6.18.5 Response Example

The result data of an E2E_P44mProtect() call with short data length (length 20 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_OK is depicted in Table 6.144.

Byte	0	1	2	3	4	5	6	7
Data	0x00	0x14	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	8	9	10	11	12	13	14	15
Data	0x1c	0xc0	0x8e	0xef	0x40	0x12	0x34	0x56
Field	*CRC*				*Message Type, Message Result, Source ID*			
Byte	16	17	18	19	n/a			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.144: E2E Profile 44m example short - normal response (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_OK)

6.18.6 Error Response Example

The result data of an E2E_P44mProtect() call with short data length (length 20 bytes, means 4 actual data bytes), offset = 0, counter = 0, Source ID = 0x0123456, Message Type = STD_MESSAGETYPE_RESPONSE, Message Result = STD_MESSAGERESULT_ERROR is depicted in Table 6.145.

Byte	0	1	2	3	4	5	6	7
Data	0x00	0x14	0x00	0x00	0x0a	0x0b	0x0c	0x0d
Field	*Length*		*Counter*		*DataID*			
Byte	8	9	10	11	12	13	14	15
Data	0x83	0xb1	0xed	0x25	0x50	0x12	0x34	0x56
Field	*CRC*				*Message Type, Message Result, Source ID*			
Byte	16	17	18	19	n/a			
Data	0x00	0x00	0x00	0x00	n/a			
Field	*Data*				*n/a*			

Table 6.145: E2E Profile 44m example short - ERROR response (Message Type = STD_MESSAGETYPE_RESPONSE; Message Result = STD_MESSAGERESULT_ERROR)

6.19 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.

6.19.1 Overview of the state machine

The state machine can be configured based on the parameter TransitToInvalidExtended:

[PRS_E2E_00675] ⌈By TransitToInvalidExtended==0 (false): the state machine shall behave according to Figure 6.129. There is no direct transition from NODATA to INVALID, no transition from INIT to INVALID due to counter-related faults (Autosar R19-11 or former behavior).⌋*(RS_E2E_08539)*

[PRS_E2E_00676] ⌈By TransitToInvalidExtended==1 (true): the state machine shall behave according to Figure 6.130. The direct transition from NODATA to INVALID is covered, transition from INIT to INVALID due to counter-related faults is covered (state machine extended).⌋*(RS_E2E_08539)*

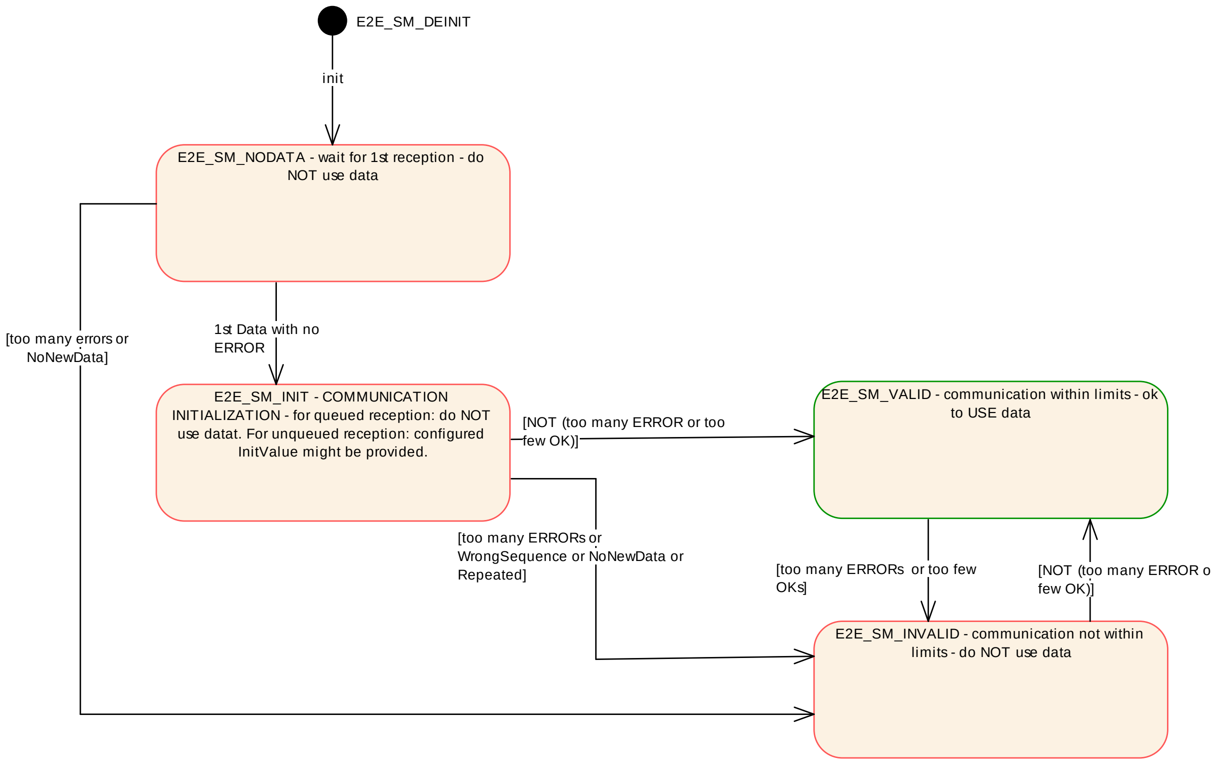

Figure 6.129: E2E state machine overview

6.19.2 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] ⌈Depending on the configuration parameter TransitToInvalidExtended, the E2E State machine shall exhibit the behavior with respect to the function E2E_SMCheck() as shown in Figure 6.129 or Figure 6.130:

1. In case of TransitToInvalidExtended==0 (false): The E2E_SMCheck() shall behave according to Figure 6.129.
2. In case TransitToInvalidExtended==1 (true): The E2E_SMCheck() shall behave according to Figure 6.130.
3. The current state (e.g. E2E_SM_VALID) is stored in State->SMState
4. At every invocation of E2E_SMCheck, the ProfileStatus can be processed (as shown by logical step E2E_SMAddStatus())
5. After that, there is an examination of two counters: State->ErrorCount and State->OKCount (continuously used through all different states). Depending on their values, there is a transition to a new state, stored in State->SMState.

⌋*(RS_E2E_08539)*

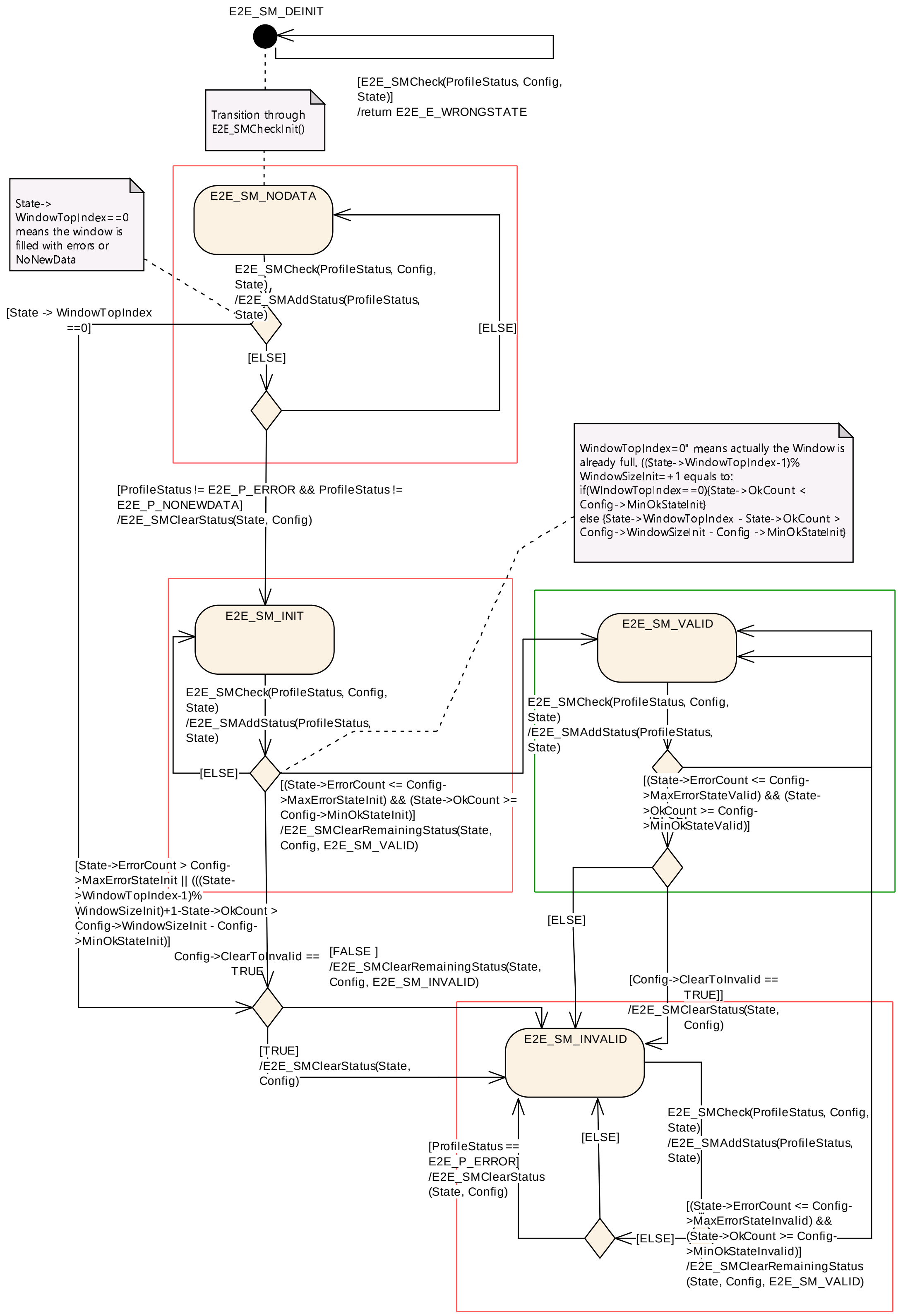

Figure 6.130: E2E state machine check

6.19.3 FTTI and E2E parameters

Considering the suggestion, switching to safe state by INVALID, the E2E state machine parameters can be chosen in such a way that the FTTI of the E2E fault can be fulfilled.

Usually, FTTI =tFD+tFR, where tFD is fault detection time (E2E fault-debouncing time), and tFR is fault reaction time (time for SW/HW fault reaction to switch to safe state after the fault detection). By neglectable tFR, FTTI=tFD.

With consideration of the cycle time of the E2E protected data, we can calculate nFD=tFD/cycleTime. The following hints can be considered by determining the E2E parameters:

1. WindowSizeInit-1 <= nFD. For example, by state NODATA, the state machine will switch to INVALID if the number of NoNewData reaches WindowSizeInit, and the duration is (WindowSizeInit-1)*CycleTime. By state INIT, the fault detection time is then (WindowSizeInit-MinOkStateInit)*CycleTime. By state transition from NODATA to INIT, and then to INVALID (e.g. first NoNewData, then WrongSequence), the total duration might exceed tFD. If this is relevant, then the following calculation can be used: 2 * WindowSizeInit-2 <= nFD.
2. WindowSizeValid-MinOkStateValid <= nFD. For example, by state VALID, the state machine will switch to INVALID, if the number of WrongSequence reaches (WindowSizeValid-MinOkStateValid+1), and the duration is (WindowSizeValid-MinOkStateValid)*CycleTime.

[PRS_E2E_00466] ⌈The step E2E_SMAddStatus(ProfileStatus, State) in E2E_SMCheck() shall behave as shown in Figure 6.131.⌋*(RS_E2E_08539)*

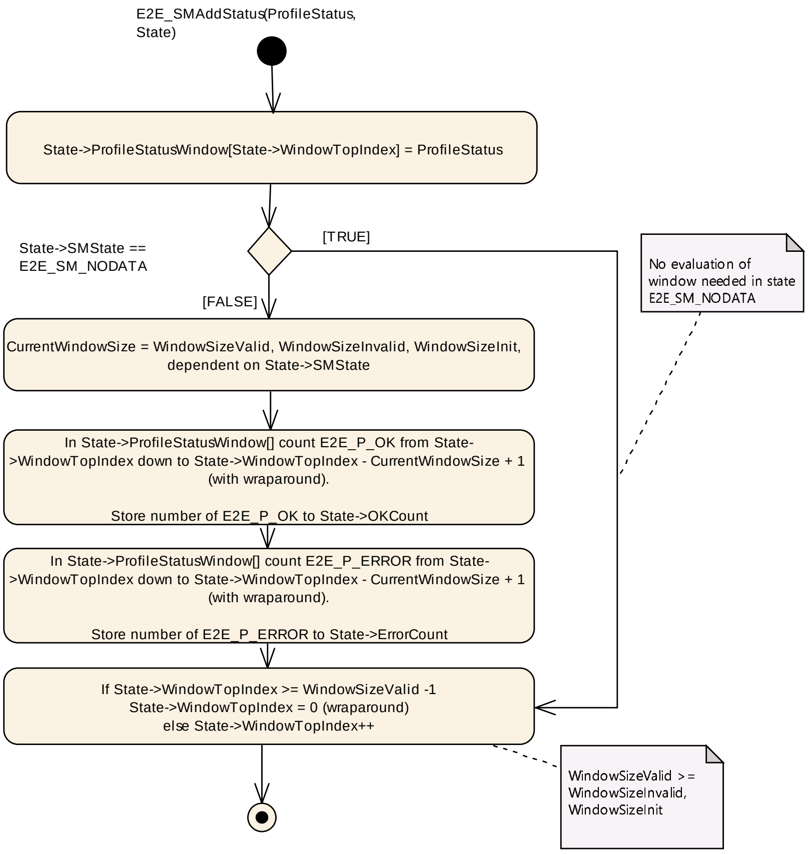

Figure 6.131: E2E state machine step E2E_SMAddStatus

E2E_SMAddStatus is just a logical step in the algorithm, it may (but it does not have to be) implemented as a separate function. It is not a module API function. The value State->OKCount represents the number of received E2E_P_OK status. The value State->ErrorCounter represents the number of E2E_P_ERROR status. The remaining status values represent counter errors, which don't contribute to State->OKCount or State->ErrorCount.

[PRS_E2E_00375] ⌈The E2E State machine shall have the behavior with respect to the function E2E_SMCheckInit() as shown in Figure 6.132.⌋*(RS_E2E_08539)*

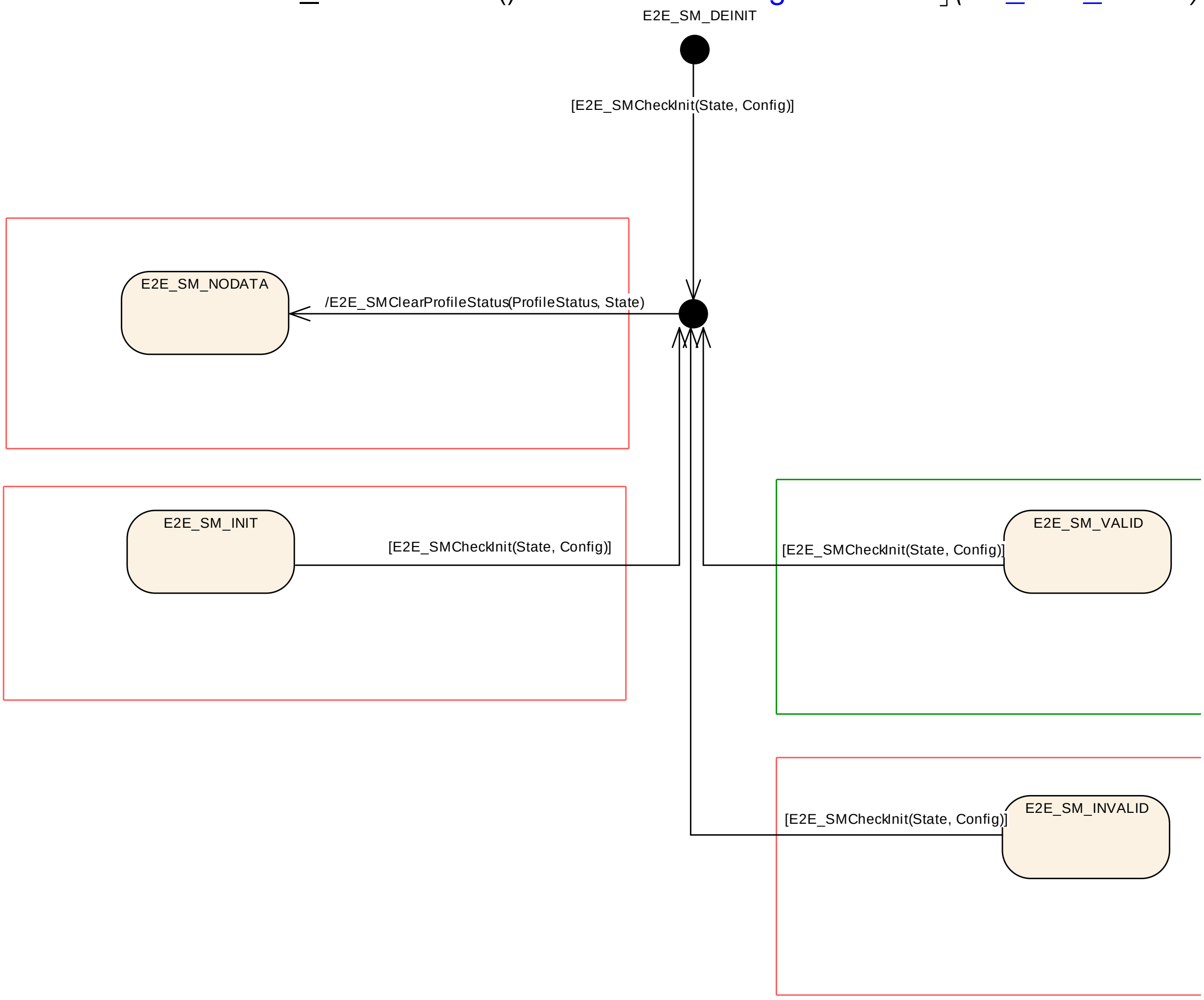

Figure 6.132: E2E state machine step E2E_SMCheckInit

[PRS_E2E_00467] ⌈The step E2E_SMClearStatus(State, Config) in E2E_SMCheck() shall behave as shown in Figure 6.133.⌋*(RS_E2E_08539)*

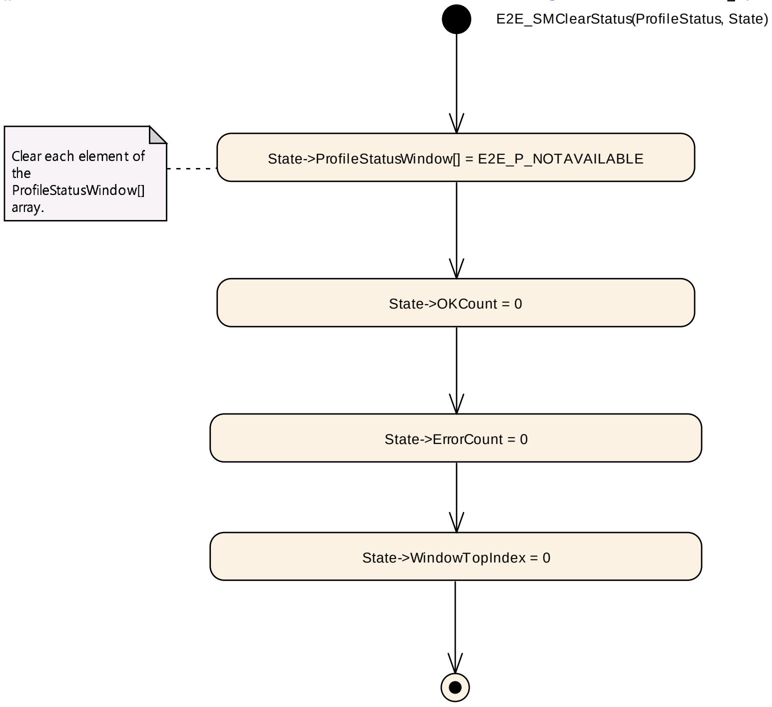

Figure 6.133: E2E state machine step E2E_SMCheck

[PRS_E2E_00607] ⌈The step E2E_SMClearRemainingStatus(Config, State) in E2E_SMCheck() shall have the following behavior: Figure 6.134.⌋*(RS_E2E_08539)*

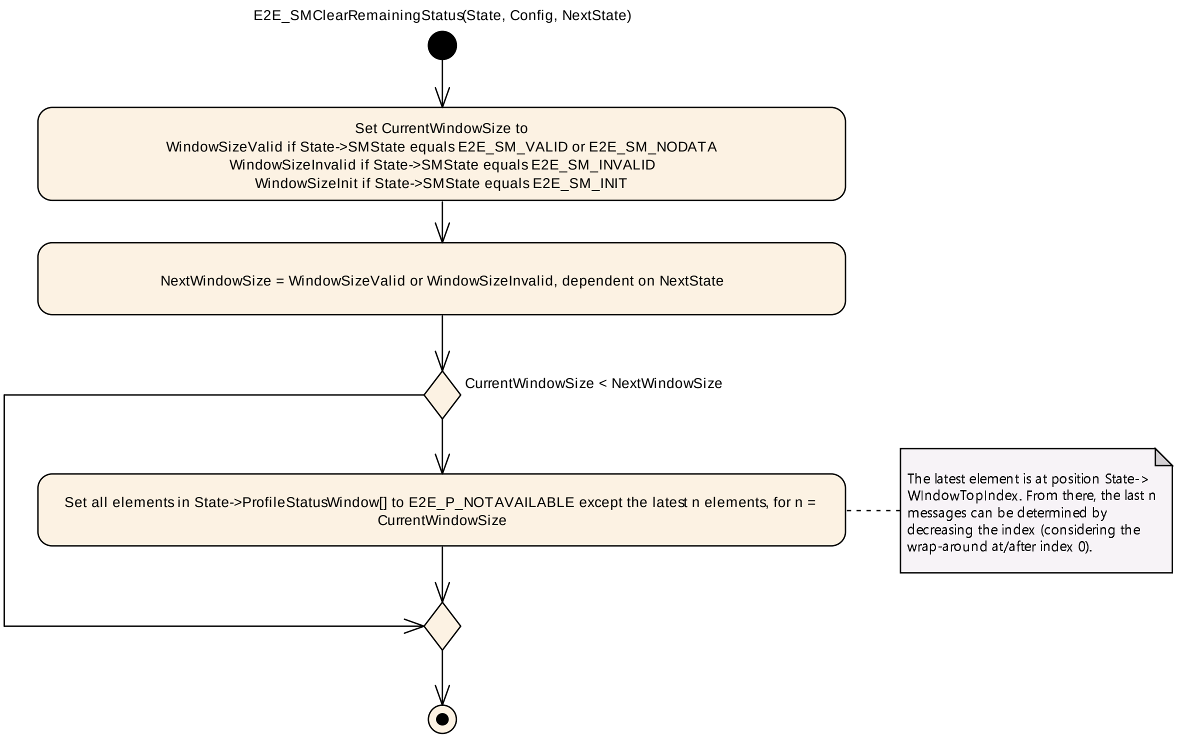

Figure 6.134: E2E state machine step E2E_SMClearRemainingStatus

6.19.3.1 E2E State Machine Configuration Type

[PRS_E2E_00668] ⌈The E2E State Machines 'Config' object shall have the members defined in (see Table 6.146).⌋*(RS_E2E_08528)*

Name	Type	Description
WindowSizeValid	Unsigned Integer	Size of the monitoring window for the state machine during state VALID.
WindowSizeInit	Unsigned Integer	Size of the monitoring windows for the state machine during state INIT.
WindowSizeInvalid	Unsigned Integer	Size of the monitoring window for the state machine during state INVALID.
MaxErrorStateInit	Unsigned Integer	Maximal number of checks in which ProfileStatus equals to E2E_P_ERROR was determined, within the last WindowSize checks (for the state E2E_SM_INIT).
MinOkStateValid	Unsigned Integer	Minimal number of checks in which ProfileStatus equals to E2E_P_OK was determined within the last WindowSize checks (for the state E2E_SM_VALID) required to keep in state E2E_SM_VALID.
MaxErrorStateValid	Unsigned Integer	Maximal number of checks in which ProfileStatus equals to E2E_P_ERROR was determined, within the last WindowSize checks (for the state E2E_SM_VALID).
MinOkStateInvalid	Unsigned Integer	Minimum number of checks in which ProfileStatus equals to E2E_P_OK was determined within the last WindowSize checks (for the state E2E_SM_INVALID) required to change to state E2E_SM_VALID.
MaxErrorStateInvalid	Unsigned Integer	Maximal number of checks in which ProfileStatus equals to E2E_P_ERROR was determined, within the last WindowSize checks (for the state E2E_SM_INVALID).
ClearToInvalid	Boolean	Clear monitoring window data on transition to state INVALID.

Table 6.146: E2E State Machine Configuration Type

6.19.3.2 E2E State Machine State Type

[PRS_E2E_00669] ⌈The E2E State Machines 'State' object shall have the members defined in (see Table 6.147).⌋*(RS_E2E_08528)*

Name	Type	Description
ProfileStatusWindow	Unsigned Integer Array	An array in which the ProfileStatus values of the last E2E-checks are stored. The array size shall be WindowSizeValid.

WindowTopIndex	Unsigned Integer	index in the array, at which the next ProfileStatus is to be written.
OkCount	Unsigned Integer	Count of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks.
ErrorCount	Unsigned Integer	Count of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks.
SMState	Enumeration	The current state in the state machine. The value is not explicitly used in the pseudocode of the state machine, because it is expressed in UML as UML states.

Table 6.147: E2E State Machine State Type

6.19.3.3 E2E State Machine Status Enumeration

[PRS_E2E_00596] ⌈The E2E State Machine uses the following enumeration values to indicate its current status (see Table 6.148).⌋*(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 6.148: E2E State Machine Check Status Enumeration

6.19.3.4 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 State Machine.

[PRS_E2E_00597] ⌈The E2E State Machine uses the following enumeration values as input from the Profile specific check functions (see Table 6.149).⌋*(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 counter was received
E2E_P_NONEWDATA	No new message was received

E2E_P_WRONGSEQUENCE	The received message contains wrong counter

Table 6.149: 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 since R4.2 is described in table Table 6.150).⌋*(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 6.150: E2E Profile 1 specific Check Status Mapping since R4.2

[PRS_E2E_00641] Mapping Profile 1 to State Machine ⌈The mapping between Profile 1 specific check states to the input for the E2E State Machine prior to R4.2 is described in table Table 6.151).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P01STATUS_OK, E2E_P01STATUS_OKSOMELOST, E2E_P01STATUS_INITIAL	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_SYNC	E2E_P_WRONGSEQUENCE

Table 6.151: E2E Profile 1 specific Check Status Mapping prior to R4.2

[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 since R4.2 is described in table Table 6.152).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P02STATUS_OK, E2E_P02STATUS_OKSOMELOST, E2E_P02STATUS_SYNC	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_INITIAL	E2E_P_WRONGSEQUENCE

Table 6.152: E2E Profile 2 specific Check Status Mapping since R4.2

[PRS_E2E_00670] Mapping Profile 2 to State Machine ⌈The mapping between Profile 2 specific check states to the input for the E2E State Machine prior to R4.2 is described in table Table 6.153).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P02STATUS_OK, E2E_P02STATUS_OKSOMELOST, E2E_P02STATUS_SYNC	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_INITIAL	E2E_P_WRONGSEQUENCE

Table 6.153: E2E Profile 2 specific Check Status Mapping prior to R4.2

[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 6.154).⌋*(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 6.154: 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 6.155).⌋*(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 6.155: 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 6.156).⌋*(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 6.156: 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 6.157).⌋*(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 6.157: E2E Profile 7 specific Check Status Mapping

[PRS_E2E_00850] Mapping Profile 8 to State Machine ⌈The mapping between Profile 8 specific check states to the input for the E2E State Machine is described in table Table 6.158).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P08STATUS_OK, E2E_P08STATUS_OKSOMELOST	E2E_P_OK
E2E_P08STATUS_ERROR	E2E_P_ERROR
E2E_P08STATUS_REPEATED	E2E_P_REPEATED
E2E_P08STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P08STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.158: E2E Profile 8 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 6.159).⌋*(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 6.159: 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 6.160).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P22STATUS_OK, E2E_P22STATUS_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 6.160: E2E Profile 22 specific Check Status Mapping

[PRS_E2E_00738] Mapping Profile 44 to State Machine ⌈The mapping between Profile 44 specific check states to the input for the E2E State Machine is described in table Table 6.161).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P44STATUS_OK, E2E_P44STATUS_OKSOMELOST	E2E_P_OK
E2E_P44STATUS_ERROR	E2E_P_ERROR
E2E_P44STATUS_REPEATED	E2E_P_REPEATED
E2E_P44STATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P44STATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.161: E2E Profile 44 specific Check Status Mapping

[PRS_E2E_00826] Mapping Profile 4m to State-Machine ⌈The mapping between Profile 4m specific check states to the input for the E2E-State Machine is described in table Table 6.162).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P04MSTATUS_OK, E2E_P04MSTATUS_OKSOMELOST	E2E_P_OK
E2E_P04MSTATUS_ERROR	E2E_P_ERROR
E2E_P04MSTATUS_REPEATED	E2E_P_REPEATED
E2E_P04MSTATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P04MSTATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.162: E2E Profile 4m specific Check Status Mapping

[PRS_E2E_00827] Mapping Profile 7m to State-Machine ⌈The mapping between Profile 7m specific check states to the input for the E2E-State Machine is described in table Table 6.163).⌋*(RS_E2E_08528)*

Profile Specific State	State Machine State
E2E_P07MSTATUS_OK, E2E_P07MSTATUS_OKSOMELOST	E2E_P_OK
E2E_P07MSTATUS_ERROR	E2E_P_ERROR
E2E_P07MSTATUS_REPEATED	E2E_P_REPEATED
E2E_P07MSTATUS_NONEWDATA	E2E_P_NONEWDATA
E2E_P07MSTATUS_WRONGSEQUENCE	E2E_P_WRONGSEQUENCE

Table 6.163: E2E Profile 7m 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.

Use case items like [PRS_E2E_UC_xxx] represent general hints on how to use the E2E protocol. However, they should not be read as requirements or limitation.

7.1 API of middleware to applications

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

[PRS_E2E_UC_00321] ⌈The middleware should 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, WRONGCRC, WRONGSEQUENCE, NOTAVAILABLE, NONEWDATA.
- e2eState: Status of the communication channel exchanging the data. Possible values are: VALID, DEINIT, NODATA, INIT, INVALID.

⌋*(RS_E2E_08534)*

[PRS_E2E_00677] ⌈The mapping between profile independent and profile specific states shall be as described in the table(Table 7.1): (yy is one of the profiles 01,02,04,05,06,07,08,11,22,44,04m,07m)⌋*(RS_E2E_08534, RS_E2E_08528)*

Profile independent result	Profile specific result with yy one of profiles (01,02,04,05,06,07,08,11,22,44,04m,07m)
OK	E2E_PyySTATUS_OK
OK	E2E_PyySTATUS_OKSOMELOST
WRONGCRC	E2E_PyySTATUS_ERROR
REPEATED	E2E_PyySTATUS_REPEATED
NONEWDATA	E2E_PyySTATUS_NONEWDATA
WRONGSEQUENCE	E2E_PyySTATUS_WRONGSEQUENCE

Table 7.1: Mapping between profile independent and specific states

[PRS_E2E_00678] ⌈The mapping between communication channel status and state machine states shall be as described in the tableTable 7.2):⌋*(RS_E2E_08534, RS_-E2E_08548)*

Communication channel status	Result of E2E_SMCheck
VALID	E2E_SM_VALID
NODATA	E2E_SM_NODATA
DEINIT	E2E_SM_DEINIT
INIT	E2E_SM_INIT
INVALID	E2E_SM_INVALID

Table 7.2: Mapping between Communication Channel and State Machine States

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): e2eResult
- E2E_protect(in dataID, inout serializedData)
- E2E_forward(in dataID, inout serializedData)

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)*

[PRS_E2E_00828] ⌈For C/S (method) communication the E2E protocol shall provide the following interface:

- E2E_check(in dataID, in messageType, in messageResult, in sourceID, in serializedData): e2eResult
- E2E_check(in dataID, in messageType, in messageResult, out sourceID, in serializedData): e2eResult

- E2E_protect(in dataID, in messageType, in messageResult, in sourceID, inout serializedData)
- E2E_forward(in dataID, in messageType, in messageResult, in sourceID, inout serializedData)

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
- messageType is used to distinguish between request and response messages
- messageResult is used to distinguish between normal response and error response messages
- sourceID is a unique identifier of the source of the exchanged data/information. In case C/S (method) communication, each single client gets a separate sourceID.
- 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.

Note that there are two overloads for E2E_check() that differ w.r.t. the direction of the sourceID. On the source side (i.e., the client side in case of C/S communication) the sourceID shall be provided by the caller to the E2E_check() in order to have the sourceID in the E2E header verified against the passed one. On the sink side (i.e., the server side in case of C/S communication) the sourceID shall be extracted from the E2E header by E2E_check() and shall be provided to caller.⌋*(RS_E2E_08534)*

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

[PRS_E2E_00318] ⌈The middleware shall determine the DataID, the Message Type, the Message Result, and the SourceID of the currently exchanged information.⌋*(RS_E2E_08534)*

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

[PRS_E2E_00319] ⌈The middleware invokes E2E functions providing them the DataID the Message Type, the Message Result, and the SourceID 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, RS_E2E_08542)*

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.
sourceID	4m, 7m	This represents a unique numerical identifier. Note: ID is used for protection against masquerading among different sources producing the same data w.r.t. the dataID. sourceId is used as a unique identifier of the source producing certaing data. One sourceId 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 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 8.
crcOffset	1, 11	Offset of the CRC in the Data[] array in bits. Fixed for AP to 0.
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 State Machine Settings

[PRS_E2E_CONSTR_03176] Value range of `windowSizeValid` ⌈ The value of the `windowSizeValid`attribute shall be greater or equal to 1.⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_06301] Dependency between `windowSizeInvalid` and `windowSizeValid` ⌈ The following restriction shall be respected: `WindowSizeInvalid` <= `WindowSizeValid`⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_06302] Dependency between `windowSizeInit` and `windowSizeValid` ⌈ The following restriction shall be respected: `windowSizeInit` <= `WindowSizeValid`⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_03177] Dependency between `maxErrorStateValid` ⌈ `maxErrorStateInit` and `maxErrorStateInvalid` The following restriction shall be respected: `maxErrorStateValid >= maxErrorStateInit >= maxErrorStateInvalid` >= 0⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_03178] Dependency between `minOkStateValid`, `minOkStateInit` and `minOkStateInvalid` ⌈The following restriction shall be respected: 1 <= `minOkStateValid` <= `minOkStateInit` <= `minOkStateInvalid`⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_03179] Dependency between `minOkStateInit`, `maxErrorStateInit` and `WindowSizeInit` ⌈The following restriction shall be respected: `minOkStateInit` + `maxErrorStateInit` <= `WindowSizeInit`⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_03180] Dependency between `minOkStateValid`, `maxErrorStateValid` and `windowSizeValid` ⌈The following restriction shall be respected: `minOkStateValid + maxErrorStateValid <= windowSizeValid`⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_03181] Dependency between `minOkStateInvalid`, `maxErrorStateInvalid` and `WindowSizeInvalid` ⌈The following restriction shall be respected: `minOkStateInvalid` + `maxErrorStateInvalid` <= `WindowSizeInvalid`⌋*(RS_E2E_08528)*

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 usages.

Use case items like [PRS_E2E_UC_xxx] represent general hints on how to use the E2E protocol. However, they should not be read as requirements or limitation.

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_UC_00239] ⌈For profiles 1, 2, 4, 5, 6, 7, 11, and 22 the E2E CRC should 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, RS_E2E_08541)*

[PRS_E2E_USE_00741] ⌈For profiles 4m, 7m, 8m and 44m the E2E CRC shall be calculated over the following parts of the serialized SOME/IP message.

1. Payload [variable size]

⌋*(RS_E2E_08540)*

[PRS_E2E_UC_00238] ⌈The E2E header should 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, RS_E2E_08541)*

9.2 Client-Server Communication

[PRS_E2E_UC_00606] ⌈When a client sends a request to the server, the server should use the received counter as sequence counter for the response, no matter if regular response or error response.⌋*(RS_E2E_08541)*

[PRS_E2E_CONSTR_06300] MaxDeltaCounter for Client-Server Communication (server) ⌈ For Client-Server Communication the MaxDeltaCounter on server-side shall be set to the maximum of the value range of the sequence counter.⌋*(RS_E2E_08528)*

[PRS_E2E_CONSTR_06303] MaxDeltaCounter for Client-Server Communication (client) ⌈ For Client-Server Communication the MaxDeltaCounter on client-side shall be set to 1.⌋*(RS_E2E_08528)*

Due to different send intervals of Clients the used counters will increment in a different rate, therefore counter jumps are inevitable. Due to the usage of the maximum value of the counter range all possible jumps are valid.

Nonetheless the server could receive the same counter from two or more clients, which will raise a E2E_P_REPEATED error on the server side. Since the E2E-protection of Client-Server Communication cannot detect counter related errors reliably on the server side, the potentially raised E2E_P_REPEATED status code from the check function can be interpreted as E2E_P_OK.

To detect specific communication failure modes e. g. loss a deadline monitoring on client side is required. Since the request and the response counter have to be equal, no deadline monitoring is possible via the E2E Protocol, this has to be implemented by the E2E Protocol user.

9.3 Periodic use of E2E check

[PRS_E2E_UC_00325] ⌈The E2E check function should 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 handling of an application is usually not possible.

[PRS_E2E_UC_00235] ⌈The user (caller) of E2E Supervision, in particular the receiver, should 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 and 4m	4 kB
E2E Profile 5	4 kB
E2E Profile 6	4 kB
E2E Profile 7 and 7m	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 E2EProfiles 1, 2, 11, or 22 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 be 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 Profiles 1 or 11 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 Profiles 4, 5, 6 or 4m should 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 or 7m should not exceed 4MB.⌋*(RS_E2E_08528)*

9.6 Functional Safety Requirements

[PRS_E2E_UC_00236] ⌈When using E2E Supervision, the designer of the functional or technical safety concept of a particular system using E2E Supervision should 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 should 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.⌋*(RS_E2E_08528)*

In other words, SW-C shall be able to tolerate the reception of one erroneous data element, whose 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.

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 PRS_E2E_UC_00072 and PRS_E2E_UC_00073 [4].

[PRS_E2E_UC_00171] ⌈Any user of E2E Supervision should ensure, that within one implementation of a communication network every source of safety-related Data, protected by E2E Supervision, has a unique Source ID (E2E Profiles 4m, 7m).⌋*(RS_E2E_08528)*

9.7 Message Layout

This chapter provides some requirements and recommendations on how safety-related messages shall or should be defined. These recommendations can be also extended to non-safety-related data transmissions.

9.7.1 Alignment of signals to byte limits

This chapter provides some requirements and recommendation on how safety-related data structures shall or can be defined. They could also be extended to non-safety-related data structures if found adequate.

[PRS_E2E_UC_00062] ⌈When using E2E Profiles, messages that have length < 8 bits should be allocated to one byte of a message, i.e. they should not span over two bytes.⌋*(RS_E2E_08528)*

[PRS_E2E_UC_00063] ⌈When using E2E Profiles, messages that have length >= 8 bits should start or finish at the byte limit of a message⌋*(RS_E2E_08528)*

[PRS_E2E_UC_00320] ⌈When using E2E Profiles, the length of the data to be protected should be multiple of 8 bits.⌋*(RS_E2E_08528)*

The previous recommendations cause that signals of type uint8, uint16 and uint32 fit exactly to respectively one, two or four byte(s) of a message. These recommendations also cause that for uint8, uint16 and uint32, the bit offsets are a multiple of 8.

Figure Figure 9.1 is an example of signals (CRC, Alive and Sig1) that are not aligned to the Byte limits.

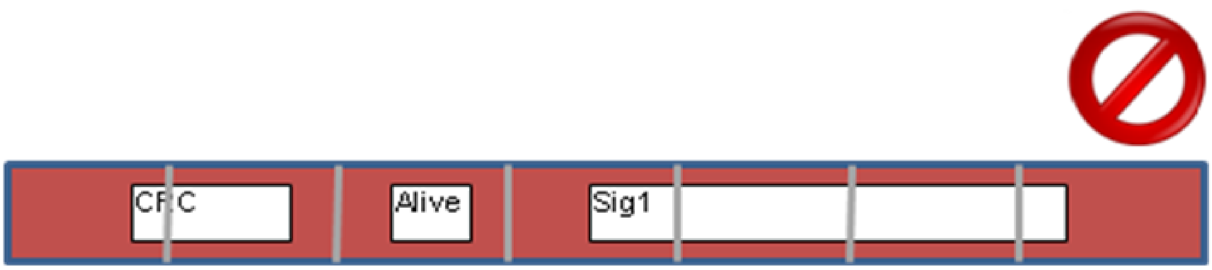

Figure 9.1: Example for alignment not following recommendations

9.7.2 Unused bits

It can happen that some bits in a protected data structure are unused. In such a case, the sender does not send signals represented by these bits, and the receiver does not expect to receive signals represented by these bits. In order to have a systematically defined data structure and sender-receiver behavior, the unused bits are set to the defined default value before calculation of the CRC.

[PRS_E2E_UC_00173] ⌈Any sender, which uses the E2E-Profiles, should fill all unused areas of a messages to a default value configured for the message).⌋*(RS_E2E_-08528)*

For signal based communication of I-PDU's this is defined in the system template parameter ISignalIpdu.unusedBitPattern. The attribute unusedBitPattern is actually an 8-bit Byte pattern. It can take any value from 0x00 to 0xFF. Often 0xFF is used.

9.7.3 Byte order (Endianness)

For each signal that is longer than 1 byte (e.g. uint16, uint32), the bytes of the signal need to be placed in associated endianness. There are two ways to do it:

1. start with the least significant byte first - the significance of the byte increases with the increasing byte significance. This is called little Endian (i.e. little end first).
2. start with the most significant byte first - the significance of the byte decreases with the increasing byte significance. This is called big Endian (i.e. big end first).

For signal communication the underlaying COM-stack in contrary is responsible for copying each signal into/from an I-PDU (i.e. for serialization of set of variables into an array). An I-PDU is transmitted over a network without any alteration. Before placing a signal in an I-PDU, COM can, if needed, change the byte Endianness the value:

1. Sender converts the byte Endianness of the data
2. Sender copies the converted data on I-PDU (serializes the signal), while copying only used bits from the signals,
3. Sender COM delivers unaltered I-PDU to receiver COM (an I-PDU is just a byte array unaltered by lower layers of the network stack),
4. Receiver COM converts the Endianness of the signals in the received I-PDU (if configured). It may also do the sign extension (if configured),
5. Receiver COM returns the converted signals.

To achieve high level of interoperability, the automotive networks recommend a particular byte order, which is depicted in table Table 9.2.

Network	Byte order

FlexRay	Little Endian
CAN	Little Endian
LIN	Little Endian
TCP/IP	Big Endian
Byteflight	Big Endian
MOST	Big Endian

Table 9.2: Networks and their byte order

The networks that have been initially targeted by E2E, which have been FlexRay, CAN and LIN are Little Endian, which results with the following requirement:

[PRS_E2E_UC_00055] ⌈Any user of an E2E Profile should place multi-Byte data in the same byte order as the underlaying communication network.⌋*(RS_E2E_08528)*

9.8 Configuration constraints on Data IDs

9.8.1 Data IDs

To be able to verify the identity of the data elements or signal groups, none of two are allowed to have the same Data ID (E2E Profiles 1, 4, 5, 6, 7, 11, 4m, 7m) or same DataIDList[] (E2E Profile 2, 22) within one system of communicating ECUs.

It is recommended that the value of the Data ID be assigned by a central authority rather than by the developer of the software-component. The Data IDs are defined in Software Component Template, and then realized in E2E_PXXConfig structures.

[PRS_E2E_UC_00071] ⌈Any user of E2E Protocol should ensure that within one implementation of a communication network every safety-related data element, protected by E2E Protocol, has a unique Data ID (E2E Profiles 1, 4, 5, 6, 7, 11, 4m, 7m) or a unique DataIDList[] (for E2EProfile 2, 22).⌋*(RS_E2E_08528)*

Note: For Profile 1 requirement (PRS_E2E_UC_00071) may not be sufficient in some cases, because Data ID is longer than CRC, which results with additional requirements PRS_E2E_UC_00072 and PRS_E2E_UC_00073. In Case of Profile 1 the ID can be encoded in CRC by double Data ID configuration (both bytes of Data ID are included in CRC every time), or in alternating Data ID configuration (high byte or low byte of Data ID are put in CRC alternatively, depending of parity of Counter), there are different additional requirements/constraints described in the sections below.

9.8.2 Double Data ID configuration of E2E Profile 1 and 11

In E2E Profiles 1 and 11, the CRC is 8 bits, whereas Data ID is 16bits. In the double Data ID configuration (both bytes of Data ID are included in CRC every time), like it is in the E2E variant 1A, all 16 bits are always included in the CRC calculation. In consequence, two different 16 bit Data IDs DI1 and DI2 of data elements DE1 and DE2 may have the same 8 bit CRC value. Now, a possible failure mode is for example that a gateway incorrectly routes a safety-related signal DE1 to the receiver of DE2. The receiver of DE2 receives DE1, but because the DI1 and DI2 are identical, the receiver might accept the message (this assumes that by accident the counter was also correct and that possibly data length was the same for DE1 and DE2).

To resolve this, there are additional requirements limiting the usage of ID space. Data elements with ASIL B and above shall have unique CRC over their Data ID, and signals having ASIL A requirements shall have a unique CRC over their Data IDs for a given data element/signal length.

[PRS_E2E_UC_00072] ⌈Any user of Profile 1 or 11 in Double Data ID configuration should ensure that assuming two data elements DE1 and DE2 on the same system (vehicle): for any data element DE1 having ASIL B, ASIL C or ASIL D requirements with Data ID DI1, there should not exist any other data element DE2 (of any ASIL) with Data ID DI2, where:

exist any other data element DE2 (of any ASIL) with Data ID DI2, where:

```
Crc_CalculateCRC8( start value: 0x00, data[2]: {lowbyte (DI1),highbyte(DI1)} )
=
Crc_CalculateCRC8( start value: 0x00, data[2]: {lowbyte (DI2),highbyte(DI2)} ).
```

⌋*(RS_E2E_08528)*

The above requirement limits the usage of Data IDs of data having ASIL B, C, D to 255 distinct values in a given ECU, but gives the flexibility to define the Data IDs within the 16-bit naming space.

For data elements having ASIL A requirements, the requirement is weaker - it requires that there are no CRC collisions for the ASIL A signals of the same length:

[PRS_E2E_UC_00073] ⌈Any user of Profile 1 or 11 in Double Data ID configuration should ensure, that assuming two data elements DE1 and DE2, on the same system (vehicle): for any data element DE1 having ASIL A requirements with Data ID DI1, there should not exist any other data element DE2 (having ASIL A requirements) with Data ID DI2 and of the same lengthas DE1, where

```
Crc_CalculateCRC8( start value: 0x00, data[2]: {lowbyte (DI1),highbyte(DI1)} )
=
Crc_CalculateCRC8( start value: 0x00, data[2]: {lowbyte (DI2),highbyte(DI2)} ).
```

⌋*(RS_E2E_08528)*
The above two requirements PRS_E2E_UC_00072 and PRS_E2E_UC_00073 assume that DE1 and DE2 are on the same system. If DE1 and DE2 are exclusive

(i.e. either DE1 or DE2 are used, but never both together in the same system / vehicle configuration, e.g. DI is available in coupe configuration and DI2 in station wagon configuration), then CRC(DI1) = CRC(DI2) is allowed.

9.8.3 Alternating Data ID configuration of E2E Profile 1 and 11

In the alternating Data ID configuration, either high byte or low byte of Data ID is put in CRC alternatively, depending of parity of Counter. In this configuration, two consecutive Data are needed to verify the data identity. This is not about the reliability of the checksum or software, but really the algorithm constraint, as on every single Data only a single byte of the Data ID is transmitted and therefore it requires two consecutive receptions to verify the Data ID of received Data.

9.8.4 Nibble configuration of E2E Profile 1 and 11

In the nibble Data ID configuration of E2E Profile 1 and 11, the low byte is not transmitted, but included in the CRC. Because the low byte has the length of 8 bits, it is the same as the CRC.Therefore, if two Data IDs are different in the low byte, this results with a different CRC over the Data ID low byte.

[PRS_E2E_UC_00308] ⌈Any user of Profile 1 or 11 in Nibble Data ID configuration should ensure that:

1. the high nibble of high byte of Data ID is equal to 0
2. the low nibble of high byte of Data ID is within the range 0x1..0xE (to avoid collisions with other E2E Profile 1 configurations that have 0x0 on this nibble, and to exclude the invalid value 0xF).
3. The low byte of Data ID is different to low byte of any Data ID present in the same bus that uses E2E Profile in Double Data ID configuration.

⌋*(RS_E2E_08528)*

[PRS_E2E_UC_00317] ⌈When using E2E Profile Variants 1/11A and 1/11C in one bus/system, the following should be respected:

1. 1/11A data should use IDs that are < 256 (this means high byte shall be always = 0)
2. 1/11C data should use IDs that are >= 256 (this means high byte is always != 0) and < 4‘096 (0x10’00 - it means they fit to 12 bits).
3. Any low byte of 1/11C data id should be different to any low byte of 1/11A data ID.

⌋*(RS_E2E_08527)*

Thanks to the Data ID distribution according to the above requirement, addressing errors can be detected: in particular, it can be detected when 1/11C message arrives to 1/11A destination. If 1/11C message receives to a 1/11A destination, then the CRC check will pass if low byte of the sent 1/11C message equals to the expected 1/11A address - and this is excluded by the above requirement.

Example: 1A may use addresses 0 to 199, while 1C may use addresses where low byte is 200 to 255 and high byte is between 1 and 15. This allows to use additional (256-200)*15 = 840 Data IDs. Once it is known, the corresponding E2E Library CRC routines should be used.

A Usage and generation of DataID Lists for E2E profile 2 and 22

An appropriate selection of DataIDs for the DataIDList in E2E Profiles 2 and 22 allows increasing the number of messages for which detection of masquerading is possible. The DataID is used when calculating the CRC checksum of a message, whereas the DataID is not part of the transmitted message itself, i.e. the message received by the receiver does not contain this information.

Any receiver of the intended message needs to know the DataID a priori. The performed check of the received CRC at the receiver side does only match if and only if the assumed DataID on the receiver side is identical to the DataID used at the sender side. Thus, the DataID allows protecting messages against masquerading. It is important that the used DataID is known solely by the intended sender and the intended receiver.

With a constant DataID (independent of the Counter) the maximum number of messages that can be protected independently using E2E Profile 2 is limited by the length of the CRC (i.e. with a CRC length of 8 bits the number of independent DataID is 2 * 8 = 256, this equates to the maximum number of independent messages for detection of masquerading).

However, E2E Profiles 2 and 22 uses a method to allow more messages to be protected against masquerading by exploiting the prerequisite that a single erroneously received message content does not violate the safety goal (a basic assumption taken in the design of applications of receiving SW-Cs).

The basic idea in E2E Profiles 2 and 22 is to use a DataIDList with several DataIDs that are selected in a dynamic behavior for the calculation of the CRC checksum. The DataID is determined by selecting one element out of DataIDList, using the value of Counter as an index (for detailed description see E2E profile 2).

The examples given below were selected to show two exemplary use cases. It is demonstrated how the detection of masquerading is performed.

Although the examples take some assumptions on the configuration, the argumentation is valid without loss of generality. For sake of simplicity, these additional constraints are not explained in the following examples.

A.1 Example A (persistent routing error)

A.1.1 Assumptions

Consider a network with one or more nodes as sender (messages A to F) and one node as the intended receiver of the safety relevant message (message B). The messages are configured to use the DataIDList as shown in Figure A.1 and Figure A.2.

Sender-ECU		**DataIDList**																
		DataID for Counter =																
	message	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Sender	A	177	103	29	206	132	58	235	161	87	13	190	116	42	219	145	71	
Sender	B	146	41	187	82	228	123	**18**	164	59	205	100	246	141	36	182	77	
Sender	C	102	204	55	157	8	110	212	63	165	16	118	220	71	173	24	126	
Sender	D	225	199	173	147	121	95	69	43	17	242	216	190	164	138	112	86	
Sender	E	181	112	43	225	156	87	**18**	200	131	62	244	175	106	37	219	150	
Sender	F	244	244	244	244	244	244	244	244	244	244	244	244	244	244	244	244	←special case of static DataID

Figure A.1: Example for alignment not following recommendations -> Sender-ECU IDs

Receiver-ECU		**DataIDList**															
		Counter =															
	message	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Receiver	B	146	41	187	82	228	123	**18**	164	59	205	100	246	141	36	182	77

Figure A.2: Example for alignment not following recommendations -> Receiver-ECU IDs

In the example of Figure A.3 it is assumed that a routing error occurs at a specific point in time. All messages are of same length. The routing error persists until it is detected. For instance a bit flip of the routing table in a gateway could lead to such a constant misrouting. It is further assumed that the senders of messages B and E have the same sequence counter (worst case situation for detection in the receiver).

The receiver should only receive message B and expects therefore the DataIDs of DataIDList of message B. Every time the expected DataID matches with the used DataID in the CRC-protected message, the result of the CRC check will be valid. In any other case the CRC checksum in the message differs from the expected CRC result and the outcome of the CRC check is not valid.

A.1.2 Solution

As depicted, the first routing error occurs when both senders reach Counter = 6. Since the DataIDList in both senders have DataID = 18 for Counter = 6, the receiver will not detect the erroneously routed message of sender E. However, for any other Counter the values of DataIDs do not match, thus the CRC check in the receiver will be not valid. With this, it is obvious that the misrouting is detected at least for the second received misrouted message (even if some messages were not received at all).

	Sender of B		Sender of E		Receiver expects message B				
	Counter	DataID	Counter	DataID	Counter	DataID used	check	DataID expected	result of CRC-Check
	0	146	0	181	0	146	=	146	valid
	1	41	1	112	1	41	=	41	valid
	2	187	2	43	2	187	=	187	valid
	3	82	3	225	3	82	=	82	valid
	4	228	4	156	4	228	=	228	valid
	5	123	5	87	5	123	=	123	valid
here 1st → routing error	6	18	**6**	**18**	**6**	**18**	=	18	erroneously undetected! (valid)
	7	164	**7**	**200**	**7**	**200**	≠	164	error detected (not valid)
	8	59	**8**	**131**	**8**	**131**	≠	59	error detected (not valid)
	9	205	**9**	**62**	**9**	**62**	≠	205	error detected (not valid)
	10	100	**10**	**244**	**10**	**244**	≠	100	error detected (not valid)
	11	246	**11**	**175**	**11**	**175**	≠	246	error detected (not valid)
	12	141	**12**	**106**	**12**	**106**	≠	141	error detected (not valid)
	13	36	**13**	**37**	**13**	**37**	≠	36	error detected (not valid)
	14	182	**14**	**219**	**14**	**219**	≠	182	error detected (not valid)
	15	77	**15**	**150**	**15**	**150**	≠	77	error detected (not valid)
	...	...	...	...	...	...	...	...	...
	5	123	**5**	**87**	**5**	**87**	≠	123	error detected (not valid)

Figure A.3: Example A configuration

A.1.3 Example B (forbidden configuration)

Not every DataIDList is allowed to be used for every message length. A short explanation to demonstrate this is shown in this example.

Consider a message G with a total length of 8 bytes. Both, sender and receiver are configured to use the DataIDList depicted in Figure A.4

Receiver-ECU		**DataIDList**															
		Counter =															
	message	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Receiver	G	73	144	215	35	106	177	248	68	139	210	30	101	172	243	63	134

Figure A.4: Example B configuration (forbidden configuration)

Without loss of generality the payload is assumed to be [22,33,44,55,66,77].

For the defined CRC generator polynomial in profile 2 and 22 the CRC checksums are as follows:

```
Counter Data                          DataID CRC-result
CRC( 0,22,33,44,55,66,77, 73) = 114
CRC( 1,22,33,44,55,66,77,144) = 197
CRC( 2,22,33,44,55,66,77,215) =  66
CRC( 3,22,33,44,55,66,77, 35) =  66
CRC( 4,22,33,44,55,66,77,106) = 207
CRC( 5,22,33,44,55,66,77,177) =  38
CRC( 6,22,33,44,55,66,77,248) =  20
CRC( 7,22,33,44,55,66,77, 68) = 165
CRC( 8,22,33,44,55,66,77,139) = 120
CRC( 9,22,33,44,55,66,77,210) =  44
CRC(10,22,33,44,55,66,77, 30) = 110
CRC(11,22,33,44,55,66,77,101) =  23
```

```
CRC(12,22,33,44,55,66,77,172) = 121
CRC(13,22,33,44,55,66,77,243) = 207
CRC(14,22,33,44,55,66,77, 63) = 141
CRC(15,22,33,44,55,66,77,134) = 175
```

One can see that DataID = 215 for Counter = 2 leads to the same CRC checksum as DataID = 35 for Counter = 3. Moreover, DataID = 106 for Counter = 4 leads to the same CRC checksum as DataID = 243 for Counter = 13.

A routing error of a non-CRC-protected message with constant payload and a sequence counter could be undetected at the receiver side if

1. the first routing error occurs at Counter = 2 and is persistent, or
2. the routing error occurs only at Counter = 4 and Counter = 13.

In both cases the second masquerading error is not detected.

Thus, the considered DataIDList of message G in Figure A.4 shall not be used for messages with a total length of 8 bytes. (Remember: the DataID itself is never transmitted on the bus).

A.2 Conclusion

The proposed method with dynamic DataIDs for CRC calculation allows protecting significantly (several orders of magnitude) more messages against masquerading than with a static DataID.

The set of DataIDList needs to be generated with appropriate care to utilize the strength of the shown method. Every DataIDList is only allowed to be assigned once to a message within the network/system. The message length needs to be considered in the assignment process since not every DataIDList is allowed to be used for every message length.

A.3 DataID List example

This section presents a part of exemplary DataIDList. The example has 500 lines, which means that this enables to identify 500 different data.

This DataIDList of 9 subtables has been selected and tested with appropriate care to comply with current safety standards. Every user of the provided DataIDLists is responsible to check if the following list is suitable to fulfill his constraints of the intended target network.

For each value of counter: **DataID value to be used**

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	28	8	73	140	234	88	71	199	14	46	38	103	158	122	129	242
2	180	134	53	233	243	121	67	176	9	94	168	132	198	142	201	138
3	78	206	124	3	132	58	100	23	154	160	136	220	127	93	37	11
4	158	161	150	33	238	151	203	19	139	104	55	2	184	16	105	67
5	137	17	243	195	157	94	103	59	201	1	187	183	11	159	143	146
6	199	242	159	165	168	194	21	90	20	167	109	136	19	69	217	119
7	12	243	207	94	6	201	30	183	35	143	78	190	188	108	16	38
8	15	209	107	99	158	116	34	148	215	161	231	111	156	207	150	221
9	245	53	92	121	207	9	230	132	113	201	5	47	4	179	35	160
10	4	65	1	42	110	208	168	45	211	25	191	226	12	167	122	106
11	125	130	213	139	180	235	221	187	162	134	199	78	95	229	53	123
12	159	194	20	136	217	173	214	95	89	144	158	77	230	186	51	205
13	246	35	39	72	84	175	138	251	13	135	106	15	71	120	49	79
14	186	170	140	32	215	199	25	141	103	102	16	242	163	113	91	114
15	29	68	44	31	8	152	1	56	19	140	246	219	209	208	86	247
16	149	178	48	27	29	185	17	203	148	68	204	36	101	195	44	16
17	250	60	120	89	66	216	164	137	202	112	125	12	206	73	74	17
18	160	145	248	202	149	236	182	34	83	178	198	196	106	105	48	207
19	42	45	226	106	113	62	80	154	133	115	60	98	94	63	21	212
20	73	86	14	103	129	30	118	159	191	193	229	108	150	131	127	194
21	203	16	84	189	56	135	146	117	49	247	47	109	88	66	232	85
22	23	11	188	245	202	204	151	53	110	34	129	92	51	2	196	121
23	136	95	77	205	225	52	170	129	17	198	33	10	128	32	4	193
24	45	106	62	154	115	98	63	212	227	101	89	24	183	241	69	171
25	126	6	203	143	54	16	127	211	84	240	121	189	236	210	56	224
26	39	175	13	15	49	40	145	209	184	197	98	107	118	202	100	99
27	114	177	90	184	43	119	50	223	95	237	29	234	232	83	63	215
28	148	221	7	229	19	81	190	49	200	67	41	82	170	169	230	197
29	1	208	211	226	122	224	217	62	117	41	138	133	207	144	131	98
30	88	218	87	39	221	57	132	175	114	229	226	13	54	160	81	15
31	107	116	215	111	150	102	246	7	91	151	240	244	28	35	139	81
32	242	165	194	90	167	136	69	119	173	80	76	95	67	239	144	234
33	37	248	50	236	79	83	180	196	249	48	4	88	62	134	99	246
34	219	232	142	64	159	96	133	222	128	194	15	172	7	227	20	164
35	33	19	2	67	14	230	189	142	123	30	57	179	215	117	191	96
36	132	160	37	145	169	248	61	202	50	149	225	236	45	182	79	34
37	104	14	70	30	10	191	247	108	176	127	81	250	130	192	59	20
38	175	15	40	209	197	107	202	99	223	158	24	116	131	34	93	148
39	35	72	175	251	135	15	120	79	40	109	154	209	122	216	197	48
40	80	63	225	241	23	198	116	213	4	11	73	153	20	111	188	235
41	182	105	134	70	17	233	219	191	121	195	244	176	241	232	94	250
42	106	154	98	212	101	24	241	171	22	50	137	231	38	213	239	110
43	202	34	196	207	27	246	70	201	218	203	162	35	212	191	36	190
44	163	44	33	152	73	19	143	219	2	86	218	67	107	211	14	232
45	19	67	230	142	30	179	117	96	97	108	13	128	102	85	250	172
46	112	181	126	157	24	6	129	1	203	231	134	143	145	193	54	208
47	118	131	192	156	114	166	154	214	222	177	149	28	84	212	90	186
48	247	192	64	166	165	222	98	28	172	90	79	164	229	24	136	8
49	50	83	249	88	99	55	53	218	32	148	1	87	227	121	111	39
50	70	191	176	250	59	138	64	60	160	146	43	120	235	222	38	89
51	251	79	209	48	76	99	12	185	116	163	171	148	21	243	161	36
52	200	43	80	237	97	63	209	125	225	75	28	241	159	99	23	130
53	188	204	110	92	196	25	187	9	122	246	127	113	181	123	218	47
54	87	57	114	13	81	177	37	40	90	82	133	184	56	248	43	107
55	122	41	131	21	57	156	58	69	166	13	18	214	16	220	177	239
56	53	121	9	132	201	47	179	160	115	190	217	37	65	128	72	145
57	121	132	47	160	190	37	128	145	101	169	144	248	42	61	251	202
58	204	92	25	9	246	113	123	47	41	35	210	115	157	97	39	37
59	89	137	12	17	181	243	86	195	207	157	139	94	23	103	6	59

For for a message with length [bytes]: " ": **not yet assigned**, "X": **not allowed**

#	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
1	X		X	X	X	X	X		X	X		X	X		X		X	X			X				X	X			X	X		X	X	X			X	X	X	X	X
2	X				X	X	X		X		X	X			X	X	X	X		X	X		X	X	X	X	X		X			X			X	X	X			X	X
3		X			X	X			X	X		X		X	X		X	X			X	X		X	X			X			X		X	X	X	X		X	X		X
4	X			X	X	X	X	X				X	X	X			X		X	X		X	X	X			X		X	X		X			X		X	X			X
5	X	X	X		X			X	X			X	X	X	X	X	X	X		X		X		X			X	X	X	X			X	X		X	X			X	
6		X		X			X		X			X	X	X	X	X					X	X	X	X	X		X		X	X	X		X		X	X			X		X
7	X	X		X				X	X		X	X	X	X		X		X	X			X	X	X	X				X	X	X	X				X	X		X		X
8		X	X	X		X	X			X		X			X	X	X		X		X					X	X	X	X	X	X			X	X		X	X	X		
9	X	X	X		X	X	X	X	X	X	X	X	X	X		X		X		X	X	X	X				X		X	X	X	X	X		X	X	X	X	X	X	
10	X	X		X	X					X			X		X	X				X	X	X	X	X	X	X	X			X		X	X	X	X		X	X		X	X
11		X	X	X	X	X		X	X	X	X			X	X	X	X		X		X	X		X	X	X				X			X	X	X	X	X	X	X		
12	X	X	X			X	X	X	X	X	X	X					X	X		X						X	X	X			X			X	X	X		X	X		X
13		X		X	X	X		X		X			X	X	X				X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X		X	X
14			X		X	X	X	X	X		X		X			X	X			X	X		X	X	X	X	X	X	X	X			X		X		X	X	X		
15	X	X	X	X		X	X	X	X	X		X			X	X		X	X	X	X		X	X	X	X	X		X	X	X	X	X		X	X		X	X	X	X
16	X		X		X							X			X	X		X	X	X			X		X		X	X		X		X	X		X	X	X	X	X	X	X
17			X			X				X	X		X	X	X	X			X	X	X	X	X	X	X	X			X		X	X			X		X	X		X	
18		X	X			X					X	X	X	X		X	X		X	X		X			X		X		X			X	X	X	X	X	X	X	X	X	
19	X	X	X	X	X	X		X	X	X	X	X	X	X			X	X	X	X	X			X	X		X		X	X	X	X	X		X		X	X	X		
20	X	X			X	X	X		X	X	X			X	X	X			X	X		X		X	X		X				X		X	X		X	X	X	X	X	
21		X	X	X	X			X	X	X	X	X		X	X	X	X			X	X	X	X			X			X		X	X		X		X	X			X	
22		X	X	X		X	X	X	X			X	X	X		X	X	X	X	X			X		X	X	X	X	X	X				X		X		X			X
23			X		X	X	X		X		X		X				X		X	X		X		X		X			X	X	X		X	X			X	X	X		
24	X	X	X	X			X	X	X		X				X	X			X				X	X	X			X	X		X		X	X	X	X	X	X		X	X
25		X		X	X		X	X	X	X	X	X		X		X	X	X	X	X		X	X	X	X		X	X	X		X	X		X	X		X	X	X		X
26		X	X			X				X	X			X	X	X	X	X		X			X	X	X		X		X	X	X		X		X	X			X		X
27	X	X		X		X	X	X		X		X	X			X		X		X				X	X		X		X	X	X		X	X	X	X		X		X	
28	X	X	X	X	X	X		X		X	X		X	X	X	X	X	X	X	X		X		X	X		X	X	X	X	X			X	X						X
29	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X	X			X	X			X	X		X	X	X	X		X	X				X	X
30		X			X		X	X	X		X	X	X	X			X	X	X	X		X		X	X	X				X	X	X	X			X	X	X		X	X
31		X	X	X		X	X				X	X	X		X	X	X	X	X				X	X	X			X	X	X		X	X	X			X	X	X	X	
32	X		X	X				X	X	X		X	X			X		X					X		X		X		X		X	X	X					X	X	X	
33	X	X	X	X	X	X	X	X		X			X	X					X	X	X			X	X	X		X	X	X	X	X	X		X		X	X		X	
34		X		X		X	X	X	X			X	X	X		X	X	X			X		X	X	X	X		X	X	X	X		X		X	X			X		X
35	X		X	X	X			X	X		X			X		X	X				X	X	X	X	X	X	X		X	X	X		X		X	X	X		X	X
36		X			X	X			X		X				X	X	X	X	X				X		X	X	X	X	X	X	X	X	X			X		X	X		X
37	X	X	X	X	X		X	X		X		X	X	X			X	X	X			X	X	X	X	X		X	X					X	X			X	X		X
38	X	X	X		X		X	X	X	X	X	X	X	X	X	X			X		X	X	X	X	X	X		X		X		X	X		X		X	X	X	X	X
39		X	X	X	X	X		X	X	X	X	X	X	X			X	X	X	X		X	X	X	X		X			X	X	X		X	X	X		X	X		X
40	X	X	X			X		X	X		X	X		X		X		X	X			X	X	X			X	X	X			X		X	X	X	X	X	X		X
41		X		X	X	X		X		X			X	X	X	X		X	X				X					X			X	X		X	X						X
42		X		X	X	X	X		X	X	X	X			X		X	X		X	X		X	X		X	X			X	X	X	X	X	X	X		X	X		X
43	X	X	X		X					X	X	X	X	X	X	X	X	X	X	X	X			X			X	X	X	X	X		X	X		X	X		X		
44	X	X			X	X	X	X	X		X	X			X	X	X	X		X		X	X	X		X	X	X	X	X	X	X	X		X	X	X				
45	X	X	X	X	X	X	X		X	X					X	X					X		X	X		X	X	X	X		X	X		X		X		X			X
46	X		X	X	X		X			X	X	X		X	X	X						X		X				X	X	X	X			X	X		X	X	X	
47		X		X		X		X		X			X	X		X			X		X			X	X	X	X					X	X	X	X	X		X	X	X	X
48		X	X	X	X		X	X	X	X	X			X				X		X	X	X		X		X		X	X	X	X	X		X	X	X		X	X	X	
49	X			X	X	X				X	X		X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X		X	X	X	X	X				X	X	X
50		X	X	X	X	X	X				X		X		X	X		X		X	X			X	X	X		X	X		X	X		X			X	X	X	X	
51	X	X		X	X			X	X	X			X	X	X	X				X	X	X			X	X	X		X	X	X			X		X	X			X	X
52	X	X	X	X		X	X	X	X	X							X						X	X			X					X	X	X		X			X	X	X
53	X	X	X	X			X	X	X	X	X		X		X	X	X				X	X	X		X	X	X	X	X		X		X	X			X			X	X
54	X		X			X			X	X		X	X	X		X	X	X	X		X	X	X	X	X	X	X	X			X	X	X	X	X			X	X	X	X
55	X					X		X	X		X	X	X	X	X		X	X			X	X	X		X		X		X	X	X	X		X	X	X	X	X		X	
56		X		X	X	X	X			X		X	X	X	X		X	X			X	X		X			X		X	X	X	X		X		X	X		X	X	
57	X				X	X	X			X	X	X	X	X	X			X		X		X	X	X			X		X	X	X	X	X	X	X	X	X	X	X	X	
58	X		X	X		X		X		X	X		X			X		X		X		X	X	X	X	X		X		X		X		X	X		X		X	X	X
59	X	X	X	X	X	X		X	X	X				X				X		X		X	X	X	X		X				X	X	X	X	X	X					X

Figure A.5

#	For each value of counter: **DataID value to be used**																For for a message with length [bytes]: " ": not yet assigned , "X": not allowed																																							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
60	174	124	106	58	47	154	237	220	98	37	173	212	59	125	101	188	X		X	X		X		X			X		X	X	X					X		X	X	X	X	X	X				X			X	X	X				X	X
61	157	1	143	208	71	211	5	226	189	122	176	224	34	217	118	62		X		X		X	X	X	X	X	X	X	X	X			X			X	X	X	X	X	X	X		X			X	X	X				X		X	X	X
62	144	225	26	198	220	4	215	153	157	188	104	65	60	102	171	162	X	X		X		X	X	X	X		X		X	X		X	X	X	X	X		X		X			X		X		X	X	X	X			X	X	X		
63	117	85	76	29	166	163	181	68	161	28	236	44	175	157	164	31		X		X		X	X	X		X	X	X	X		X		X	X		X	X		X	X		X	X	X	X	X	X	X		X		X	X	X			X
64	146	66	18	112	62	74	95	181	178	98	228	126	132	205	227	157		X	X		X			X		X			X	X	X	X	X	X	X			X				X	X	X	X			X		X	X	X	X		X	X	X
65	115	101	69	50	15	239	11	83	186	209	181	249	224	245	107	88	X	X	X	X		X	X			X	X	X	X			X	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X		X	X		X	X		X
66	3	23	220	11	145	188	150	245	171	202	205	204	147	151	236	53	X	X	X	X		X	X	X	X		X		X	X		X		X	X			X		X	X	X	X	X	X		X	X	X			X	X	X	X		
67	24	231	54	71	249	240	162	122	56	55	94	118	178	174	32	41	X		X	X				X			X	X	X	X	X		X		X	X		X	X	X	X	X	X	X	X		X	X	X					X	X		X
68	172	238	182	104	77	105	31	14	134	52	111	70	237	56	17	30		X	X	X			X	X	X	X	X	X	X		X	X	X	X	X	X	X		X	X		X	X	X	X	X	X		X	X	X	X		X	X	X	X
69	11	245	204	53	34	92	2	121	25	207	193	9	26	230	246	132		X		X	X	X	X	X	X		X	X	X	X	X	X		X	X	X			X	X	X	X		X	X	X	X	X	X		X	X	X	X		X	X
70	197	158	93	161	172	150	27	33	130	238	249	151	177	203	182	19	X		X		X		X	X		X		X		X	X	X		X	X	X			X	X	X		X	X		X	X		X	X	X		X	X			X
71	213	235	162	78	53	174	81	206	45	121	159	124	52	82	9	3	X		X			X			X	X	X			X		X			X	X	X		X			X		X	X	X			X		X	X		X	X	X	
72	131	156	166	214	177	28	212	186	164	184	178	8	135	171	119	170			X	X	X			X		X	X	X	X	X	X	X		X	X	X	X			X		X		X	X					X	X					X	
73	212	171	231	110	83	71	235	25	240	88	195	122	74	78	55	113	X	X	X	X		X		X	X		X								X	X		X	X	X	X	X	X	X	X	X	X	X	X				X	X			X
74	9	47	115	37	72	101	75	248	69	251	51	50	208	228	15	236	X		X			X		X	X	X	X	X	X		X				X	X	X			X		X	X		X		X	X	X	X	X			X	X	X	
75	173	77	137	52	26	17	140	10	243	4	151	195	75	199	157	127	X	X	X	X	X	X	X	X	X	X		X	X				X	X	X		X	X				X					X	X		X	X		X			X	X
76	192	166	222	28	90	164	24	8	238	119	48	73	49	231	95	140		X		X		X	X	X		X	X						X	X	X	X	X			X		X	X	X	X	X		X	X	X	X	X	X	X	X	X	
77	116	111	102	7	151	244	35	81	168	2	118	200	8	72	187	82	X	X		X	X		X	X		X	X		X	X		X	X	X	X	X		X	X		X		X	X	X		X	X	X		X	X		X		X	X
78	123	97	3	75	138	23	158	228	220	120	119	11	5	161	145	180		X		X		X		X		X		X	X	X	X	X		X	X		X	X	X		X	X				X	X	X		X		X	X	X			X
79	46	91	193	168	235	5	57	167	210	78	232	217	104	13	174	80	X		X			X	X			X	X	X	X			X		X		X	X			X		X	X		X	X				X				X	X		
80	151	2	187	230	70	123	135	179	206	191	114	97	46	109	176	128	X	X		X		X	X	X		X	X	X	X	X	X	X	X	X		X		X		X		X	X	X		X	X			X	X	X			X	X	X
81	229	49	82	197	142	100	149	158	237	96	69	93	141	178	128	161	X	X	X		X	X		X	X	X	X	X	X		X	X			X		X				X		X		X	X					X	X	X		X	X	X
82	21	69	214	239	40	186	188	249	8	107	126	170	117	204	223	55	X		X			X	X		X	X	X	X	X		X	X	X			X			X	X	X	X	X			X			X		X	X	X			X	X
83	47	37	101	248	251	50	228	236	239	79	26	83	226	180	209	196		X	X	X			X		X		X			X	X	X					X	X	X	X	X			X		X				X	X		X	X	X	X	X
84	128	61	228	182	89	180	44	105	245	137	215	134	80	152	12	70	X	X		X	X	X			X		X		X		X	X	X			X	X	X		X	X	X	X	X				X	X	X			X	X			
85	76	163	161	44	164	33	6	152	151	73	88	19	40	143	104	219	X		X	X				X	X	X	X		X	X		X	X		X	X								X	X		X	X	X		X	X	X		X	X	X
86	240	118	247	131	141	192	106	156	64	114	169	166	36	154	165	214		X							X	X	X		X	X		X	X	X	X	X		X	X			X	X	X		X		X	X	X	X	X	X	X	X	X	X
87	183	38	169	18	224	149	173	74	79	133	23	178	9	77	85	126	X			X	X		X	X	X	X	X	X			X		X	X		X	X	X		X	X	X	X			X	X	X			X	X					
88	169	149	79	178	85	48	137	27	99	29	188	185	115	17	163	203				X					X	X			X	X	X	X	X	X	X			X		X	X		X	X		X	X	X	X	X	X			X	X		
89	248	236	83	196	48	88	134	246	55	185	65	218	98	233	148	35		X			X			X	X			X	X	X		X		X	X	X	X	X	X		X	X	X	X	X	X			X	X	X	X		X	X	X	X
90	67	142	179	96	108	128	85	172	75	20	40	61	244	29	60	238		X	X	X	X	X		X				X		X	X			X				X	X		X	X	X	X		X	X	X	X		X	X	X	X	X		X
91	55	87	141	57	7	114	47	13	165	81	224	177	31	37	200	40	X	X		X	X		X				X	X				X		X			X	X	X	X		X		X		X	X	X	X	X	X	X				X	X
92	209	99	116	148	161	111	207	221	102	33	71	7	214	201	151	229				X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X		X		X	X		X					X		X						X
93	49	197	100	158	96	93	178	161	125	172	239	150	114	27	61	33							X	X			X	X	X	X	X	X			X	X		X	X	X	X				X		X		X		X			X			
94	93	150	130	151	182	139	36	2	235	105	32	187	119	84	134	230		X		X		X		X			X			X	X	X		X	X	X				X				X		X	X	X	X	X	X	X	X	X	X		X
95	138	120	145	216	18	202	238	12	236	74	241	34	124	104	178	243			X	X		X		X		X	X	X	X	X			X		X	X	X	X	X	X	X	X		X	X	X	X	X	X	X		X	X	X			X
96	189	117	109	85	192	76	112	29	158	166	248	163	39	181	222	68	X	X		X	X	X	X			X							X		X	X		X		X			X		X		X	X	X		X	X		X		X	
97	224	133	85	227	156	29	26	22	163	214	202	68	72	4	28	54		X		X	X	X	X		X	X	X		X				X				X	X					X	X		X	X	X				X	X		X		
98	22	54	31	240	170	56	42	118	219	32	201	247	48	45	199	131	X		X			X		X			X	X	X	X	X	X				X	X	X		X					X	X		X	X	X	X	X	X		X		
99	6	143	16	211	240	189	210	224	135	118	132	117	196	147	247	133							X			X	X	X	X	X			X	X		X	X	X	X	X		X		X	X	X	X	X		X	X			X	X	X	X
100	208	226	224	62	41	133	144	98	85	21	120	227	201	225	156	24		X	X	X	X	X	X	X			X	X	X	X		X		X	X	X	X		X	X	X	X			X		X		X	X				X	X	X	
101	218	39	57	175	229	13	160	15	177	49	62	40	240	145	82	209	X	X	X	X	X	X		X	X				X	X	X	X		X			X	X	X		X	X										X		X		X	X
102	8	140	86	199	46	103	122	242	30	91	84	159	161	41	193	165	X		X		X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X				X	X	X	X	X	X	X	X	X	X	X			X			
103	31	56	219	247	199	232	226	192	142	242	72	64	148	62	159	166	X	X		X		X	X	X	X	X	X	X	X						X		X					X	X				X					X	X		X	X	X
104	184	223	234	215	125	46	88	102	129	130	31	91	222	218	213	244		X				X			X	X		X		X	X		X	X			X	X		X	X	X		X			X	X	X	X					X		
105	135	109	197	76	64	158	74	163	93	222	50	161	57	126	172	44	X	X	X			X	X		X	X	X		X				X		X			X	X	X	X		X	X	X							X	X		X	X	X
106	62	98	227	24	69	22	198	231	68	239	12	54	169	153	186	71		X		X	X	X				X	X	X	X	X	X		X	X	X			X		X	X						X		X	X	X	X		X	X	X	
107	44	152	19	219	86	67	211	232	230	103	39	142	116	224	30	64	X		X			X	X	X				X		X	X	X	X	X	X		X	X		X					X	X	X		X	X	X	X		X		X	X
108	105	70	233	191	195	176	232	250	132	59	200	138	213	64	183	60	X					X	X				X	X	X	X	X					X		X	X	X				X	X	X	X	X		X	X		X		X	X	
109	124	58	154	220	37	212	125	188	24	248	77	171	146	130	50	204		X	X	X	X	X	X	X						X		X		X	X			X	X	X	X	X	X	X	X	X	X	X	X		X				X		X
110	215	102	91	244	139	168	39	200	5	187	247	167	73	175	78	43	X	X	X					X	X		X	X	X		X				X		X		X		X	X	X	X	X		X	X		X	X	X	X	X	X	X	X
111	92	9	113	47	35	115	97	37	21	72	147	101	1	75	175	248	X		X	X	X	X	X	X							X	X	X	X		X	X		X		X	X	X	X	X	X	X	X	X	X			X				X
112	85	29	163	68	28	44	157	31	33	8	196	152	15	1	73	56		X	X		X			X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X				X		X	X	X			X	X	X	X	X		X
113	249	55	32	87	111	141	9	57	242	7	211	114	68	47	244	13	X		X	X	X	X		X	X	X		X	X			X		X	X				X	X	X		X	X	X	X	X	X	X		X		X	X	X	X	
114	17	195	94	59	1	183	159	146	190	208	123	38	245	194	211	66	X	X	X			X			X	X		X	X	X	X		X	X			X	X			X		X	X				X			X	X	X			X	
115	38	18	149	74	133	178	77	126	48	227	11	27	47	52	29	6	X		X	X		X		X	X				X		X	X			X		X						X	X		X	X		X			X		X			
116	69	239	186	249	107	170	204	55	140	116	6	32	85	92	215	87			X	X		X	X	X	X	X	X			X			X			X	X	X	X	X	X		X	X	X		X		X	X		X	X		X		X
117	233	176	132	138	183	160	96	120	37	38	80	145	162	172	169	216		X	X	X	X	X		X		X	X	X		X	X		X			X	X	X	X		X				X		X	X	X	X	X	X	X		X	X	X
118	56	247	232	192	242	64	62	166	96	165	251	222	221	98	194	28			X	X	X	X				X	X		X	X	X		X	X			X					X		X	X	X	X			X	X				X	X	

Figure A.6

For each value of counter: **DataID value to be used**

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
119	168	167	217	80	206	144	40	63	51	3	222	225	30	107	58	241
120	7	81	200	82	230	43	251	100	80	179	156	237	199	79	97	93
121	113	115	21	101	175	69	23	50	214	15	112	239	211	11	40	83
122	170	32	199	141	102	242	113	114	159	244	189	165	44	115	168	177
123	108	20	60	173	147	89	28	77	216	51	93	137	123	8	112	52
124	94	183	190	38	211	169	20	18	251	224	3	149	92	173	117	74
125	194	136	173	95	144	77	186	205	137	225	161	52	179	170	26	129
126	217	144	51	225	58	26	223	198	181	220	238	4	250	215	212	153
127	216	12	34	243	126	207	14	94	246	6	235	201	220	30	203	183
128	171	110	71	25	88	122	78	113	118	218	59	41	126	206	87	115
129	211	224	117	133	131	85	51	227	76	156	145	29	35	26	166	22
130	141	114	165	177	200	90	101	184	136	43	85	119	219	50	80	223
131	20	173	89	77	51	137	8	52	12	26	150	17	97	140	181	10
132	206	3	58	23	160	220	93	11	212	145	95	188	210	150	248	245
133	129	193	127	5	162	210	114	217	146	174	142	147	105	177	45	144
134	201	190	72	169	189	251	60	149	15	117	58	79	25	89	109	178
135	147	51	112	26	154	181	234	4	126	212	182	157	138	46	24	65
136	18	74	178	126	227	27	52	6	185	22	245	203	37	10	68	143
137	161	33	151	19	104	2	16	67	187	14	87	230	223	189	70	142
138	120	216	202	12	74	34	104	243	196	126	213	207	58	14	27	94
139	232	64	96	222	194	172	227	164	61	136	209	238	81	22	173	73
140	68	31	152	56	140	219	208	247	67	199	35	232	99	226	103	192
141	214	186	8	170	223	140	110	32	86	215	203	199	76	25	46	141
142	166	28	164	8	119	73	231	140	104	234	185	86	197	71	205	199
143	156	214	28	186	184	8	171	170	73	223	27	140	109	110	234	32
144	72	251	15	79	109	209	216	48	107	76	212	99	41	12	158	185
145	152	219	67	232	103	142	224	64	179	159	175	96	111	133	108	222
146	241	213	153	235	245	162	7	78	42	53	103	174	77	81	92	206
147	145	202	236	34	178	196	105	207	88	27	153	246	154	70	185	201
148	234	46	129	91	213	193	87	168	127	235	219	5	238	57	162	167
149	154	212	24	171	50	231	213	110	54	83	17	71	18	235	249	25
150	10	127	59	210	42	146	165	147	38	45	179	66	134	90	226	51
151	2	230	123	179	191	97	109	128	3	250	177	75	91	76	138	61
152	52	10	195	127	65	59	242	210	183	42	230	146	180	165	208	147
153	43	237	63	125	75	241	99	130	198	228	8	213	194	148	11	139
154	86	103	30	159	193	108	131	194	250	5	49	20	151	156	210	136
155	14	30	191	108	127	250	192	20	138	210	82	60	139	166	146	173
156	109	76	158	163	222	161	126	44	150	164	83	33	13	6	238	152
157	79	48	99	185	163	148	243	36	111	44	110	221	69	94	33	84
158	185	36	221	84	152	229	183	135	81	219	113	49	249	38	67	109
159	26	4	157	65	171	1	91	42	143	110	70	208	216	168	71	45
160	193	5	210	217	174	147	177	144	66	124	96	51	70	184	106	225
161	5	217	147	144	124	51	184	225	112	58	172	26	191	223	154	198
162	82	100	237	93	128	125	48	150	241	61	186	130	165	185	228	151
163	95	205	52	129	198	10	32	193	195	153	19	127	61	141	65	5
164	102	244	168	200	187	167	175	43	217	123	192	80	86	15	206	237
165	231	71	240	122	55	118	174	41	247	87	183	131	27	124	141	21
166	96	172	61	238	173	182	68	104	180	77	116	105	43	31	137	14
167	90	119	95	234	63	205	249	46	52	241	44	129	96	55	198	91
168	127	210	146	147	45	66	90	51	18	106	128	112	233	119	62	26
169	176	138	160	120	38	145	172	216	248	18	63	202	174	238	149	12
170	63	241	198	213	11	153	111	235	65	245	86	162	173	7	204	78
171	41	21	156	69	13	214	220	239	28	40	74	186	189	188	184	249
172	162	174	45	124	9	106	43	58	62	47	20	154	195	237	115	220
173	142	96	128	172	20	61	29	238	228	173	107	182	200	68	89	104
174	205	129	10	193	153	127	141	5	59	162	67	210	182	114	42	217
175	75	228	11	180	216	245	33	134	204	12	46	53	144	19	34	233
176	178	27	185	203	68	36	195	16	221	31	92	84	50	59	152	189
177	225	198	4	153	188	65	102	162	1	204	14	42	89	244	110	174

For for a message with length [bytes]: " ": not yet assigned , "X": not allowed

#	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
119		X		X	X		X	X	X	X	X	X		X		X	X	X		X	X	X		X	X					X	X	X	X			X	X	X	X	X	
120	X		X	X		X				X			X	X	X			X	X		X	X		X	X	X	X	X		X	X	X		X	X			X		X	X
121	X	X	X				X		X	X	X	X	X	X		X	X	X	X		X	X	X	X	X	X			X			X	X	X		X	X			X	X
122	X	X		X		X	X		X	X	X		X		X		X	X	X	X	X					X	X		X					X	X	X		X	X	X	X
123	X				X	X		X	X		X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X			X		X	X	X	X			X		X
124	X	X	X	X	X		X	X		X	X	X	X	X	X		X	X	X	X		X				X	X	X		X	X		X			X		X	X	X	X
125	X	X			X		X	X	X		X			X	X	X	X	X		X	X		X	X	X	X	X					X	X		X	X			X	X	X
126		X		X		X		X	X			X	X	X		X	X		X		X	X	X		X	X			X	X						X	X	X	X		X
127	X	X	X			X	X	X		X	X	X				X	X	X		X	X	X	X	X	X	X		X	X			X		X	X	X	X		X	X	X
128	X		X	X		X		X		X	X	X	X	X		X	X	X		X	X	X	X		X		X	X	X	X	X	X	X		X	X			X		X
129				X	X	X	X		X			X		X	X			X			X	X	X			X		X	X		X	X		X	X	X	X	X	X	X	X
130		X			X	X	X		X	X	X	X	X	X	X		X	X	X	X	X	X			X	X		X	X	X		X		X	X			X	X	X	X
131	X			X		X				X		X		X	X	X			X	X		X	X	X		X	X		X	X	X	X				X		X	X	X	
132		X	X	X		X			X	X	X	X	X					X	X	X			X	X	X		X	X		X			X	X	X			X			X
133	X		X			X			X	X	X	X	X				X	X	X	X	X	X	X			X	X	X	X	X		X			X		X	X	X	X	X
134	X			X		X	X	X		X				X			X	X	X		X	X		X	X		X			X	X		X	X	X		X	X		X	
135	X	X			X		X	X		X	X	X	X	X	X	X			X	X	X			X		X		X			X		X	X	X	X			X	X	X
136	X	X	X		X				X	X	X	X	X	X			X		X			X	X	X						X	X	X	X	X	X		X	X	X	X	X
137		X	X	X		X			X	X	X	X	X	X		X	X			X	X	X	X	X	X	X		X		X		X		X	X	X	X	X	X		
138	X		X		X				X		X	X	X	X		X	X	X	X	X		X	X					X	X	X		X	X				X		X	X	X
139		X		X		X		X			X	X	X	X			X	X		X	X	X	X	X		X			X	X		X			X	X	X		X	X	X
140	X	X	X	X	X	X	X		X		X	X		X	X	X	X		X			X	X	X		X	X	X		X				X	X			X	X	X	
141	X	X		X		X	X			X	X	X		X	X		X		X	X		X		X	X		X	X	X		X	X	X	X	X	X	X				X
142			X		X					X	X					X	X		X	X	X		X		X	X	X		X		X	X			X		X	X	X	X	X
143	X	X	X	X	X	X	X		X	X	X		X	X		X	X	X					X		X		X	X	X		X	X	X	X	X	X			X	X	
144	X	X		X		X	X	X		X			X	X				X		X	X	X		X		X	X	X	X	X			X	X	X	X		X	X		
145	X	X	X	X	X		X				X	X				X	X	X	X	X	X				X			X		X	X			X		X	X	X	X		
146	X	X	X		X	X	X		X	X		X			X	X	X	X	X	X	X		X		X	X	X	X		X	X			X	X					X	X
147	X	X			X	X	X		X	X		X		X	X	X		X						X	X		X			X				X		X		X	X		
148		X		X	X	X	X		X	X	X		X				X	X	X		X		X		X	X			X				X			X	X	X		X	
149	X	X	X	X	X	X	X	X		X	X	X	X	X		X				X		X	X	X			X			X		X	X	X		X		X	X	X	
150		X	X	X	X		X			X	X		X	X	X	X		X		X	X	X		X	X	X				X	X	X					X				X
151		X	X				X	X	X			X	X							X	X		X	X		X		X	X		X	X		X	X	X	X	X	X	X	X
152					X		X		X			X	X	X	X		X	X	X					X	X				X	X		X	X			X		X	X	X	
153		X	X			X		X			X	X	X		X				X	X	X			X		X		X	X	X			X		X	X		X		X	X
154	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X			X		X			X	X	X		X	X	X		X			X	X	
155			X	X		X		X	X			X	X	X			X	X	X	X	X	X	X	X	X	X				X	X	X	X				X	X		X	X
156	X		X		X		X		X	X		X		X		X	X	X			X	X		X	X	X	X							X	X			X		X	
157	X	X	X		X		X	X	X					X	X		X	X	X		X	X			X	X	X	X			X	X			X	X	X	X		X	X
158	X	X						X	X	X	X	X	X	X	X					X	X	X		X	X				X			X	X	X	X	X	X	X	X	X	
159	X	X	X	X		X		X		X			X		X		X	X	X		X	X	X		X	X		X	X			X	X	X	X			X			X
160	X	X			X			X		X	X		X	X			X			X	X	X		X	X	X	X	X	X	X		X		X		X			X		X
161			X		X	X	X	X	X		X		X	X	X		X		X	X	X	X	X						X	X	X			X		X			X		X
162	X	X		X				X	X	X	X	X		X	X		X	X	X	X			X	X	X	X	X	X					X	X	X	X			X	X	X
163			X	X	X	X	X		X	X	X		X			X	X	X	X		X	X		X	X	X	X	X		X	X	X	X				X		X		
164	X			X		X				X		X	X		X	X	X		X	X	X		X	X	X	X	X				X	X	X	X			X	X		X	X
165	X		X		X						X	X	X	X	X	X			X			X	X			X	X				X	X	X	X	X	X		X			X
166		X	X			X	X			X		X				X		X	X				X		X	X			X	X		X	X		X		X	X	X		X
167				X	X	X	X	X				X		X	X			X		X		X	X	X			X		X	X	X	X	X	X	X	X	X		X	X	X
168	X	X		X	X	X	X		X	X		X			X	X	X		X	X	X	X		X	X	X		X	X	X				X		X			X	X	
169	X		X		X	X	X	X	X	X	X	X		X			X	X	X	X	X	X	X	X	X		X			X	X	X	X		X	X	X	X	X		X
170	X	X			X		X			X	X					X		X		X	X			X	X	X	X	X	X		X	X	X	X	X		X			X	X
171			X	X	X	X	X	X	X		X		X		X		X	X		X	X		X	X	X	X		X					X		X			X			X
172		X	X	X		X	X	X	X	X	X	X	X			X		X		X		X		X	X		X		X	X	X	X		X		X				X	
173	X	X		X		X	X	X	X	X	X		X	X			X		X	X	X	X			X	X			X	X	X			X	X	X		X			X
174	X	X	X	X	X		X		X	X	X	X	X	X		X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X		X	X			
175	X	X	X	X	X	X	X	X	X		X	X		X	X	X	X	X	X	X				X	X		X	X	X	X	X		X		X	X	X	X	X	X	
176	X		X	X		X	X	X		X	X		X	X		X				X	X	X	X		X		X			X	X						X				X
177	X		X	X	X	X	X							X		X		X	X	X			X		X	X	X		X		X	X	X						X	X	X

Figure A.7

For each value of counter: **DataID value to be used**

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
178	32	141	242	114	244	165	115	177	194	200	117	90	152	101	167	184
179	51	26	181	4	212	157	46	65	6	171	105	1	120	91	231	42
180	54	240	56	118	32	247	45	131	232	141	190	192	185	106	242	156
181	223	215	46	102	130	91	218	244	193	139	56	168	164	39	235	200
182	101	50	239	83	209	249	245	88	170	99	157	55	133	53	116	218
183	58	220	212	188	248	171	130	204	231	236	52	110	66	139	83	92
184	179	128	75	61	60	228	163	182	11	89	223	180	167	44	216	105
185	110	25	122	113	218	41	206	115	131	39	146	21	6	3	57	101
186	71	122	118	41	87	131	124	21	192	57	38	156	203	58	114	69
187	153	162	42	174	92	45	200	124	226	9	108	106	17	43	113	58
188	119	234	205	46	241	129	55	91	10	213	152	193	172	87	153	168
189	91	168	5	167	78	217	13	80	147	206	64	144	14	40	124	63
190	190	169	251	149	117	79	89	178	209	85	220	48	113	137	76	27
191	187	123	206	97	176	3	197	75	58	138	90	23	193	158	160	228
192	165	90	136	119	80	95	239	234	77	63	163	205	142	249	225	46
193	60	89	216	137	112	12	73	17	34	181	130	243	3	86	126	195
194	98	24	22	231	239	54	153	71	31	249	243	240	149	162	170	122
195	140	199	103	242	91	159	41	165	108	168	135	194	33	21	5	90
196	139	187	78	123	233	206	49	97	124	176	165	3	129	197	132	75
197	130	139	235	187	134	78	229	123	174	233	242	206	205	49	121	97
198	230	179	97	128	250	75	76	61	23	60	184	228	168	163	120	182
199	16	189	135	117	247	109	66	85	197	192	37	76	218	112	64	29
200	84	135	49	109	232	197	18	76	100	64	101	158	87	74	96	163
201	77	52	17	10	4	195	199	127	94	65	2	59	228	242	1	210
202	111	7	244	81	2	200	72	82	167	230	131	43	140	251	123	100
203	228	180	245	134	12	53	19	233	92	243	91	121	225	67	207	176
204	227	22	68	54	186	31	65	240	152	170	207	56	79	42	140	118
205	238	104	105	14	52	70	56	30	233	10	7	191	125	247	195	108
206	207	201	35	190	16	72	250	169	175	189	124	251	110	60	135	149
207	40	107	223	116	93	215	196	111	46	150	54	102	166	246	130	7
208	133	227	29	22	214	68	4	54	44	186	34	31	251	65	8	240
209	97	75	23	228	120	11	161	180	188	216	234	245	217	33	202	134
210	57	13	177	40	82	184	248	107	119	100	227	223	247	236	237	116
211	236	196	88	246	185	218	233	35	87	36	42	39	24	176	221	72
212	191	250	138	60	146	120	222	89	145	66	237	216	78	164	18	137
213	81	82	43	100	179	237	79	93	63	128	214	125	242	48	75	150
214	30	108	250	20	210	60	166	173	120	147	100	89	187	28	66	77
215	99	148	111	221	33	7	201	229	244	19	122	81	186	190	2	49
216	237	125	241	130	228	213	148	139	153	180	140	235	136	221	245	187
217	244	200	167	43	123	80	15	237	144	97	166	63	103	209	3	125
218	100	93	125	150	61	130	185	151	213	182	170	139	90	36	180	2
219	164	73	104	86	205	14	240	103	70	129	221	30	93	118	10	159
220	221	229	81	49	67	82	169	197	43	142	21	100	32	149	179	158
221	27	203	36	16	31	84	59	189	229	56	9	135	83	146	219	117
222	13	40	184	107	100	223	236	116	234	93	22	215	192	196	125	111
223	177	184	119	223	237	234	83	215	205	125	68	46	64	88	241	102
224	34	207	246	201	203	35	191	190	39	16	174	72	171	250	84	169
225	36	84	229	135	219	49	38	109	82	232	115	197	55	18	142	76
226	198	153	65	162	204	42	244	174	208	92	30	45	137	200	25	124
227	103	159	108	194	5	20	156	136	60	217	197	173	2	214	147	95
228	222	164	238	73	95	104	54	86	105	205	148	14	100	240	52	103
229	65	42	208	45	25	226	167	106	224	113	250	62	243	80	41	154
230	167	80	144	63	3	225	107	241	26	23	164	198	108	116	220	213
231	48	185	148	36	44	221	94	84	7	152	25	229	239	183	19	135
232	66	112	74	181	98	126	205	157	27	24	180	6	160	129	22	1
233	59	146	38	66	226	18	136	112	149	62	75	74	121	95	133	181
234	243	94	201	183	143	190	108	38	72	211	206	169	204	20	189	18
235	150	151	139	2	105	187	84	230	78	70	141	123	234	135	233	179
236	220	188	171	204	236	110	139	92	71	196	10	25	112	187	88	9

For for a message with length [bytes]: **" ": not yet assigned , "X": not allowed**

#	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
178	X	X	X	X	X	X	X		X	X	X	X	X			X	X	X	X	X	X		X	X	X	X	X		X	X	X				X	X		X			X
179		X	X		X	X	X		X	X	X	X		X	X	X	X	X			X	X	X		X	X	X					X		X	X			X	X	X	
180		X		X	X		X		X		X	X	X	X		X	X	X			X	X	X	X		X					X	X	X		X	X	X		X		
181			X	X	X	X	X	X				X		X	X	X			X			X			X	X		X			X	X	X	X	X		X	X	X		X
182		X		X	X	X	X	X				X	X		X			X		X			X	X	X	X			X	X	X	X	X	X	X	X	X		X		
183	X	X			X			X		X	X	X		X					X	X	X		X	X	X	X				X	X	X		X	X	X		X	X	X	X
184	X		X	X	X			X	X	X		X	X	X			X	X	X		X	X			X	X	X	X		X	X		X					X	X	X	
185			X	X	X	X	X	X		X	X		X		X		X	X	X	X	X				X	X	X	X	X	X	X	X		X		X		X			X
186			X	X		X	X	X		X	X			X		X			X	X	X	X			X					X		X			X		X	X	X	X	
187			X	X	X			X	X	X	X			X	X	X	X	X				X				X					X	X	X		X	X	X			X	
188	X	X	X		X	X		X	X	X	X		X	X	X		X	X		X	X	X			X	X	X		X			X		X	X	X	X		X	X	X
189	X	X		X		X	X	X	X		X	X	X	X	X	X				X	X	X	X			X	X	X			X		X	X	X	X	X		X	X	X
190	X	X	X	X				X			X	X			X	X	X	X	X	X		X	X			X	X		X	X	X	X	X	X	X		X	X	X	X	X
191	X		X		X			X	X	X	X	X		X	X	X		X	X	X	X		X		X	X		X	X		X	X	X	X			X	X		X	
192		X					X		X		X	X		X	X	X	X	X	X		X		X	X	X	X			X	X	X					X		X	X	X	X
193	X						X	X	X		X			X	X		X	X	X		X	X	X		X		X	X	X	X	X		X	X	X	X		X		X	
194	X		X	X			X		X	X	X	X	X		X	X	X	X	X	X	X		X			X	X						X				X	X	X		X
195					X	X	X	X	X	X	X	X				X	X	X	X	X	X	X			X	X		X	X			X	X							X	X
196	X		X	X	X	X	X	X	X			X		X								X		X	X	X		X	X	X	X	X		X			X	X	X	X	
197	X		X	X		X		X		X		X				X	X					X		X			X		X			X	X	X	X				X	X	
198			X	X	X	X		X		X		X				X	X				X	X	X	X	X			X			X	X		X	X	X	X			X	X
199	X				X		X	X	X		X	X		X	X				X	X	X	X	X	X		X	X		X		X	X	X	X	X				X	X	
200	X	X	X	X	X	X					X	X		X	X	X	X	X	X	X	X	X			X					X					X	X	X			X	
201			X		X		X	X	X	X		X	X		X	X	X		X		X	X	X	X	X	X		X	X	X				X	X		X	X		X	
202	X				X	X	X		X	X	X	X	X	X	X		X	X		X		X	X				X	X		X			X	X							X
203		X	X			X	X		X	X		X	X		X		X	X	X	X	X		X	X		X		X	X	X			X		X	X		X	X	X	X
204	X	X		X			X	X			X	X	X	X	X	X		X	X	X	X	X		X		X	X		X	X			X	X		X	X	X			X
205		X	X	X	X	X	X	X	X		X		X	X	X				X	X	X			X	X	X			X		X	X	X		X	X	X	X	X	X	
206	X	X	X		X	X		X	X	X		X	X	X	X		X	X		X	X	X		X	X	X	X	X		X		X		X	X	X	X	X		X	
207	X		X	X	X			X	X					X							X	X				X		X	X	X	X			X	X	X	X		X	X	X
208	X	X	X	X	X	X	X	X		X	X		X	X	X		X	X	X	X	X	X	X	X	X		X	X		X		X	X	X	X			X		X	X
209		X	X	X	X	X		X		X	X							X	X	X	X		X	X	X		X		X	X	X	X	X			X	X	X	X	X	X
210	X			X		X	X				X	X	X											X		X	X	X	X	X		X			X		X	X	X		X
211	X	X	X	X		X	X	X	X	X		X	X		X		X		X			X				X	X	X	X	X				X		X			X	X	X
212		X		X		X			X	X	X	X	X	X		X	X			X	X	X		X	X	X		X	X			X	X	X	X	X	X			X	
213		X	X		X	X		X	X	X				X		X	X				X		X			X	X	X	X		X	X	X	X		X				X	X
214	X	X	X	X	X		X	X	X	X	X	X	X		X	X		X	X	X	X	X		X	X			X	X		X	X	X	X		X		X	X	X	
215	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X							X	X	X		X				X	X	X	X	X		X		X		X
216	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X				X	X		X		X	X					X
217	X	X		X	X	X	X	X	X		X					X		X	X					X	X	X	X				X		X	X	X	X		X			
218				X	X			X	X	X			X		X	X		X	X		X	X				X	X		X	X		X	X		X			X		X	
219		X		X	X					X	X			X	X	X	X			X		X	X					X		X	X	X	X	X		X			X	X	X
220	X			X				X		X	X	X	X	X	X		X	X		X	X				X	X	X			X	X	X		X	X		X	X		X	
221		X	X		X	X		X	X				X		X		X		X				X	X			X	X			X	X	X	X		X	X				
222			X		X	X	X	X	X	X		X	X		X		X			X		X	X	X			X	X	X			X	X		X		X			X	X
223	X			X	X	X	X	X	X	X	X	X				X	X	X		X	X	X		X		X	X					X	X	X	X						X
224	X		X	X	X	X	X		X	X		X	X		X	X	X	X	X		X								X	X	X	X	X		X	X		X	X	X	
225	X	X		X	X	X		X	X	X	X	X	X	X		X		X	X	X		X		X	X			X	X	X	X			X		X	X				
226		X		X	X	X	X		X				X	X					X		X	X	X	X	X		X		X		X		X	X	X	X	X	X			X
227				X		X		X		X		X			X	X		X	X				X	X	X	X	X	X	X	X	X	X	X			X		X		X	X
228	X	X		X	X	X	X	X			X	X	X			X		X	X	X	X			X	X		X		X		X	X	X	X	X		X		X		X
229	X	X		X	X	X			X			X			X	X	X	X	X				X		X	X	X	X	X	X	X				X	X	X	X	X	X	X
230	X			X	X	X	X	X	X		X	X	X	X	X		X		X		X	X	X	X	X		X	X	X	X	X	X		X		X	X		X	X	X
231	X	X	X	X	X		X		X		X	X		X	X	X	X		X	X	X	X		X	X				X	X		X	X		X	X		X		X	
232	X	X	X			X		X	X	X		X			X	X	X	X	X		X					X	X	X	X	X	X	X	X	X		X			X	X	X
233	X	X		X	X		X	X	X		X		X	X		X	X	X	X		X	X		X			X		X			X	X		X	X	X	X	X		X
234	X		X		X			X	X	X		X	X	X	X	X	X		X	X		X		X			X					X		X	X	X					
235	X	X		X		X	X		X	X				X	X		X	X	X		X	X			X	X	X			X	X		X	X	X			X		X	X
236	X	X			X	X							X	X	X	X	X	X			X	X		X			X	X	X	X	X	X			X	X	X	X	X	X	X

Figure A.8

	For each value of counter: **DataID value to be used**																For for a message with length [bytes]: " ": **not yet assigned , "X": not allowed**																																								
#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
237	210	147	66	51	106	112	119	26	74	154	61	181	176	234	98	4	X	X	X		X		X			X	X							X	X	X		X	X				X	X		X				X		X	X		X	X	
238	235	78	174	206	121	124	82	3	106	132	194	58	10	100	47	23		X		X		X			X	X					X				X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X	X		X	X
239	143	211	189	224	118	117	147	133	109	131	160	85	246	51	192	227		X			X		X	X				X			X		X						X			X			X	X	X			X	X		X	X		X	X
240	134	233	121	176	94	132	142	138	47	183	167	160	153	96	190	120	X	X		X	X	X			X		X		X		X	X	X	X	X	X			X				X	X			X	X	X			X	X	X			X
241	74	126	27	6	22	203	10	143	36	54	53	16	248	127	31	211	X	X	X				X	X		X	X	X	X		X	X	X	X		X		X	X	X			X			X	X	X		X	X					X	X
242	181	157	6	1	231	143	193	208	16	71	233	211	202	5	240	226	X	X							X	X	X	X	X				X	X	X		X		X	X	X	X		X	X	X		X	X	X	X	X	X		X	X	X
243	61	182	180	105	137	134	152	70	53	17	102	233	63	219	243	191	X	X		X	X			X		X	X	X	X		X			X		X				X			X	X	X	X		X			X	X	X	X	X	X	
244	83	88	55	218	148	87	121	39	141	221	208	57	22	132	7	175		X	X	X		X	X	X		X		X		X	X	X	X	X	X	X	X	X	X		X		X		X	X	X	X			X		X		X	X	
245	195	59	183	146	208	38	194	66	169	226	97	18	53	136	224	112		X				X	X	X		X		X			X		X	X	X	X	X			X	X	X			X	X			X	X	X	X		X	X		X
246	64	222	172	164	136	238	22	73	182	95	99	104	82	54	77	86	X	X	X	X			X	X	X	X	X		X	X				X	X		X	X	X		X	X	X		X	X	X					X	X		X	X	
247	196	246	218	35	36	39	176	72	57	84	45	175	231	138	229	251	X						X	X	X	X	X			X	X	X		X	X		X	X	X	X	X	X	X				X	X	X	X	X		X	X	X		X
248	25	113	41	115	39	21	3	101	156	175	66	69	143	23	13	50	X	X	X	X	X		X	X	X		X	X		X	X				X	X	X	X	X		X	X	X	X		X	X			X	X	X			X	X	
249	226	62	133	98	21	227	225	24	29	69	216	22	190	198	214	231			X	X		X	X	X	X					X	X	X		X	X			X		X		X		X	X	X	X	X	X		X		X	X		X	X
250	239	249	170	55	116	32	92	87	199	111	143	141	29	9	102	57		X	X	X	X	X	X		X	X		X		X		X	X	X	X		X	X			X	X		X	X		X	X		X			X	X	X		
251	50	187	202	251	61	42	248	144	169	101	145	128	37	190	160	47	X						X			X	X	X		X	X	X	X	X	X	X	X			X		X	X	X	X	X		X	X	X	X				X		
252	24	10	208	90	84	149	222	154	114	192	118	228	245	12	19	92	X	X		X			X	X	X		X	X				X	X	X	X	X	X	X		X	X	X	X		X	X		X	X	X	X	X	X	X		X	
253	138	137	193	104	228	96	88	41	119	6	38	175	161	166	117	162	X				X	X		X	X				X	X		X	X			X	X	X	X				X	X	X	X	X			X	X		X		X	X	
254	37	71	91	229	237	35	20	186	58	17	9	200	174	153	76	222			X	X	X		X			X	X		X	X	X	X			X	X	X	X	X		X			X	X	X	X		X	X	X		X	X	X		X
255	120	125	84	196	205	192	128	215	19	22	83	93	162	234	113	116			X	X			X			X	X		X	X				X		X	X		X			X		X		X	X	X	X						X	X	X
256	144	47	234	158	118	40	77	127	1	230	63	38	186	25	163	51	X	X	X		X		X	X		X		X	X		X	X	X	X					X	X		X	X	X			X	X	X	X	X	X	X		X	X	X
257	164	84	214	192	180	19	121	93	168	113	138	236	179	227	153	107	X	X	X	X	X	X					X	X	X	X		X		X	X	X			X	X		X	X		X			X	X		X	X		X	X	X	X
258	156	207	150	221	37	102	206	33	246	71	232	7	238	214	91	201	X			X		X	X	X	X		X		X		X		X	X		X		X	X	X		X	X			X	X	X		X	X		X				X
259	184	208	212	149	214	114	131	228	134	19	243	225	176	15	168	6		X		X	X		X	X					X	X		X	X		X		X		X	X	X	X	X	X		X	X		X	X		X	X	X		X	X
260	176	78	22	34	152	223	118	209	13	52	199	59	164	230	190	147	X		X	X	X	X		X	X		X		X	X	X	X	X		X	X		X			X	X	X	X	X			X		X				X	X	X	
261	216	142	86	58	24	204	201	83	21	10	144	138	62	122	208	100		X			X	X	X		X		X		X	X	X		X	X	X		X	X	X		X	X		X			X	X		X	X		X			X	X
262	234	40	1	38	163	53	3	62	249	79	37	30	131	49	210	126			X	X				X	X	X	X	X	X	X	X			X	X	X	X	X			X	X		X	X			X		X			X	X	X	X	
263	240	168	105	82	73	172	92	123	115	213	147	156	248	55	23	44	X	X	X	X					X	X		X	X	X			X		X	X	X	X	X	X	X		X					X	X	X		X		X	X		
264	49	29	251	57	246	144	52	211	128	151	153	47	94	134	28	171	X	X			X	X	X			X	X	X	X	X			X	X		X	X	X	X	X	X	X	X				X	X	X	X		X	X	X		X	X
265	197	165	97	233	187	133	22	4	186	251	240	108	207	223	42	241		X	X	X	X	X			X		X	X		X		X	X			X			X		X	X			X			X		X	X		X	X	X		X
266	226	95	110	157	197	69	125	20	33	165	103	145	182	196	97	153	X	X	X	X	X		X				X	X		X	X	X	X	X		X	X		X		X			X	X	X	X		X	X	X	X	X			X	
267	87	32	55	249	155	250	60	120	89	66	216	164	137	202	112	125	X			X	X		X				X	X	X	X	X	X							X	X		X	X	X	X			X		X	X	X	X	X	X		X
268	6	239	64	89	226	67	56	206	220	95	245	130	147	76	110	214	X	X			X	X	X		X	X	X	X	X	X	X					X	X	X	X	X	X	X			X	X				X	X	X	X	X	X	X	
269	34	209	59	147	134	18	188	119	75	31	32	181	192	235	77	159		X		X	X	X			X				X	X	X			X			X	X	X		X	X	X					X	X	X		X		X	X	X	
270	202	42	169	128	160	121	8	234	199	36	122	227	91	43	161	40	X	X			X	X	X	X	X	X	X		X			X	X	X	X	X		X					X		X		X		X		X		X	X	X		
271	20	153	73	11	4	213	116	198	23	241	225	63	80	155	203	16	X	X	X	X		X		X	X	X	X	X		X		X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X				X
272	41	162	179	184	89	103	35	208	152	206	160	212	210	200	130	149						X		X		X	X	X		X		X	X	X	X	X	X	X	X		X	X		X	X	X							X		X	X	X
273	38	62	30	126	177	219	29	160	125	245	229	231	45	57	176	36			X	X	X		X	X	X		X	X			X	X		X	X	X		X	X	X			X	X	X	X	X	X	X		X		X	X		X	X
274	209	147	18	119	31	181	235	159	132	182	249	27	93	65	25	224	X			X	X	X	X	X	X	X		X	X		X	X	X	X	X		X		X		X	X			X	X	X	X	X		X		X	X		X	X
275	98	180	20	113	124	153	161	13	73	139	149	11	67	148	4	194	X	X			X		X	X	X				X		X				X						X	X	X			X	X			X		X		X		X	X
276	105	172	115	156	23	46	239	207	174	189	181	150	8	89	88	221	X		X	X	X	X			X		X	X	X		X				X									X	X	X	X	X	X	X	X		X	X	X	X	
277	171	48	127	174	140	51	39	54	62	191	197	61	203	220	235	246		X						X		X	X	X	X	X	X	X			X	X			X	X	X	X			X	X				X	X	X	X	X		X	
278	33	201	244	122	186	2	236	23	72	222	14	167	29	195	131	189	X		X		X			X		X		X	X	X	X			X		X		X	X	X	X		X	X		X	X						X	X	X	X	X
279	22	223	13	59	190	194	163	18	198	14	92	75	214	79	43	181	X	X	X	X			X	X	X		X	X	X	X	X	X		X	X	X		X		X			X	X	X		X	X	X		X	X	X	X		X	
280	80	109	242	105	201	68	228	172	171	122	59	115	124	6	2	156	X			X	X	X	X		X	X		X		X	X	X		X	X		X	X	X		X	X	X	X	X	X	X	X	X				X	X	X		X
281	150	102	246	7	91	151	240	244	28	35	139	81	50	168	58	2			X	X	X		X			X	X	X	X				X		X				X	X		X		X		X	X	X	X	X	X					X	X
282	47	158	40	127	230	38	25	51	53	188	247	62	222	143	79	61	X	X	X	X	X	X	X	X		X	X			X	X	X	X	X	X	X	X	X	X	X	X					X	X	X		X		X		X	X	X	X
283	117	64	39	67	44	220	24	130	141	110	176	80	18	10	221	240	X			X	X	X	X	X	X	X			X	X		X	X	X	X	X		X	X	X	X		X	X	X	X		X					X	X	X		
284	108	203	12	85	234	5	75	170	45	40	156	70	244	132	1	60					X		X		X		X	X	X	X				X				X	X						X		X	X	X	X			X		X		X
285	229	186	200	222	169	131	11	12	100	121	68	15	246	198	199	5		X		X	X	X		X			X			X	X			X				X		X	X	X	X	X	X			X	X		X		X	X		X	
286	228	6	117	239	195	64	26	89	39	226	177	67	209	56	44	206	X	X			X		X	X		X		X	X	X	X			X	X		X	X	X		X		X	X	X	X	X	X		X	X	X	X	X	X	X	
287	17	169	217	121	231	199	241	227	154	161	2	87	142	135	215	1	X	X	X	X				X	X	X	X	X		X		X	X	X	X	X	X	X	X	X		X	X			X	X	X		X	X	X	X	X		X	X
288	172	156	46	207	189	150	89	221	54	37	27	102	43	206	166	33	X		X		X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X		X	X	X				X	X	X		X	X	X	X	
289	168	82	172	123	213	156	55	44	46	63	119	207	190	250	189	110	X				X		X	X	X	X		X			X	X	X	X		X				X	X	X			X	X	X	X			X			X		X	X
290	188	235	177	65	41	245	102	86	176	162	202	173	1	7	179	204	X			X		X	X		X	X	X	X			X						X			X	X	X		X	X			X		X		X	X	X	X		
291	56	76	10	217	125	90	108	199	149	196	81	154	65	203	192	87	X	X				X		X	X		X	X	X	X			X	X	X	X	X			X		X				X		X		X	X	X	X	X	X		
292	147	119	181	159	182	27	65	224	129	39	120	232	107	86	143	152	X				X	X	X	X				X	X		X	X			X	X		X		X	X	X	X	X			X	X	X	X	X	X	X	X	X		X
293	89	206	130	214	157	240	217	19	109	20	21	168	159	199	145	236	X		X	X		X	X	X	X	X	X	X	X		X		X			X	X		X		X	X	X	X		X		X	X		X	X	X	X			X
294	94	97	229	133	202	186	113	108	200	42	109	222	150	13	169	203			X	X	X	X	X	X	X				X	X	X		X		X		X	X					X	X	X	X	X		X					X	X		
295	186	222	131	12	121	15	198	5	183	227	115	45	151	16	87	70					X	X	X	X		X	X	X	X	X	X	X	X	X		X		X		X	X			X	X		X			X			X			X	

Figure A.9

	For each value of counter: **DataID value to be used**																For for a message with length [bytes]: " ": not yet assigned , "X": not allowed																																								
#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
296	66	74	98	205	27	180	160	22	20	232	10	113	239	36	124	223	X	X	X	X	X	X		X	X			X		X	X		X	X	X		X	X	X	X	X	X	X	X		X			X	X	X	X	X		X	X	X
297	8	43	135	238	171	218	119	3	170	48	226	50	73	159	127	49		X					X	X	X						X				X		X		X	X	X	X			X		X		X		X	X	X		X	X	X
298	241	135	85	218	158	170	181	50	70	127	44	60	122	27	38	187				X	X			X	X		X		X		X	X	X		X			X		X	X		X	X		X	X	X	X	X	X		X	X	X	X	X
299	39	220	141	80	221	9	208	109	57	33	22	242	132	149	7	105	X	X				X				X	X	X	X	X	X	X	X	X	X	X	X			X	X	X			X	X	X			X	X			X	X	X	
300	217	199	154	87	215	92	85	32	6	116	140	55	204	170	107	249		X	X				X	X			X		X		X			X	X	X	X		X	X	X	X	X	X	X	X		X	X		X	X	X	X	X		X
301	123	44	207	110	247	221	164	69	102	94	224	33	77	84	71	145		X	X	X		X		X		X			X	X	X	X	X					X	X				X	X		X	X		X		X		X	X	X		
302	72	140	188	191	183	235	207	216	177	136	50	65	234	221	41	142	X			X		X	X		X	X		X	X	X					X		X		X	X	X		X		X		X		X		X		X	X		X	X
303	179	103	152	212	130	52	83	114	190	240	161	134	74	100	109	225	X	X	X	X	X	X		X	X	X	X	X		X	X	X	X				X	X		X	X	X	X		X		X	X	X	X	X				X	X	X
304	242	68	171	115	2	48	117	46	127	167	75	174	4	64	140	150	X		X	X	X		X	X	X	X	X	X		X	X		X		X		X		X	X		X	X	X									X	X	X	X	X
305	190	14	43	77	68	238	96	25	218	115	146	3	153	175	48	143	X			X		X	X	X				X	X		X			X		X	X			X		X			X		X	X			X		X			X	
306	208	149	114	228	19	225	15	6	31	236	163	117	78	45	82	239		X	X			X	X	X	X	X	X	X	X				X	X		X	X			X	X	X					X			X		X	X			X	
307	149	228	225	6	236	117	45	239	182	195	79	64	34	26	123	89		X	X	X	X	X		X	X				X	X		X	X		X		X		X	X	X	X	X	X					X		X	X	X	X	X		X
308	103	212	52	114	240	134	100	225	14	168	148	31	205	183	105	117	X	X			X	X	X	X	X	X		X		X			X	X	X		X				X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X
309	249	120	164	125	74	84	42	196	214	205	58	192	41	128	180	215	X		X	X	X	X		X	X	X		X	X	X	X			X	X			X		X		X		X		X			X		X		X	X			
310	185	193	147	96	225	119	191	175	181	117	53	159	215	216	182	179	X	X	X			X	X	X		X	X	X	X	X	X	X			X	X	X		X	X		X	X	X	X		X	X	X			X	X	X	X		
311	205	22	113	223	248	13	243	59	11	190	154	194	206	163	8	18	X		X	X	X		X	X	X	X	X			X		X	X	X	X	X	X		X	X	X	X	X		X			X		X			X	X	X	X	X
312	79	177	106	245	175	176	246	173	205	179	17	78	32	151	224	21	X	X	X	X	X					X		X	X	X	X				X			X	X	X	X	X	X	X		X	X	X		X	X			X	X	X	X
313	198	16	247	197	218	94	74	165	71	50	67	97	172	205	60	233	X	X	X		X	X	X		X	X	X		X		X	X	X	X	X	X	X		X		X		X		X				X	X	X	X	X	X	X		X
314	116	155	146	66	18	112	62	74	95	181	178	98	228	126	132	205		X	X		X	X	X	X		X	X	X	X	X		X	X		X	X			X	X		X					X	X	X		X	X	X	X			
315	31	182	25	39	156	143	162	220	49	207	74	141	194	184	150	80	X				X	X	X	X			X	X	X		X	X	X	X		X	X		X	X		X					X	X	X	X	X						
316	104	41	175	162	239	179	91	184	224	89	126	103	99	35	67	208	X		X			X	X	X	X	X		X		X		X		X			X	X	X		X	X		X	X	X	X	X			X				X		
317	27	232	124	248	141	139	21	190	4	9	192	8	95	185	57	14		X	X	X	X		X	X		X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X		X	X		X		X	X	X		X	X
318	70	237	56	17	30	76	233	169	10	219	7	217	191	4	125	121	X		X			X	X	X	X		X	X		X	X	X	X		X	X		X			X	X	X	X	X	X		X	X	X			X			X	X
319	162	184	103	208	206	212	200	149	52	214	36	114	106	131	240	228	X	X	X					X			X			X		X		X	X	X		X	X	X	X		X	X	X	X	X	X	X	X	X			X	X	X	X
320	189	37	166	71	70	91	157	229	142	237	141	35	48	20	56	186			X	X		X	X	X	X	X	X	X	X			X			X	X	X	X		X					X	X	X	X			X					X	
321	86	204	21	138	208	185	2	137	209	149	234	193	219	167	114	104	X	X	X	X	X	X	X		X		X	X			X	X			X	X	X		X	X		X		X	X			X	X	X		X	X	X		X	X
322	43	238	218	3	48	50	159	49	60	174	95	187	213	224	51	29		X		X	X	X	X					X	X	X		X		X		X		X	X	X				X			X	X	X		X	X	X		X		X
323	159	224	232	152	220	248	204	52	139	80	196	190	112	138	9	134		X	X	X	X	X					X							X	X	X	X				X	X	X	X		X		X	X		X	X		X	X	X	X
324	114	225	31	117	82	182	136	64	25	123	210	39	223	178	156	67		X	X		X	X			X	X		X	X			X	X	X	X	X		X		X	X					X		X	X	X		X		X	X	X	
325	53	30	120	219	106	125	251	231	84	176	35	196	136	144	205	161	X			X	X			X		X	X	X	X	X			X	X	X		X		X		X	X	X	X		X	X	X	X	X			X	X	X	X	X
326	135	218	170	50	127	60	27	187	237	51	110	202	23	232	62	251	X	X	X		X	X	X	X	X	X	X	X	X		X			X	X	X	X			X	X		X	X	X				X	X	X		X	X	X	X	X
327	102	7	151	244	35	81	168	2	118	200	8	72	187	82	83	167	X	X	X		X	X	X	X		X		X	X	X	X	X	X	X	X	X	X	X								X		X	X		X			X	X	X	
328	100	183	104	136	92	41	37	178	175	55	51	162	243	71	239	24			X			X	X	X	X	X	X	X				X			X	X	X		X	X				X	X	X	X	X	X					X		X	
329	101	28	36	118	204	243	68	230	148	138	203	163	17	115	185	188			X	X	X	X		X		X	X		X	X	X			X	X	X		X			X		X	X	X	X	X		X	X	X			X		X	X
330	35	200	83	131	217	100	213	15	137	199	171	183	101	63	154	45	X	X			X	X			X	X	X	X	X	X	X		X		X		X	X		X			X		X	X	X			X	X	X				X	
331	122	23	167	189	12	88	226	37	191	5	25	166	211	95	45	71	X		X	X	X	X	X	X		X	X	X	X	X	X			X		X		X	X		X		X	X				X	X	X		X	X	X	X	X	
332	77	25	3	143	46	49	179	141	187	150	98	29	198	103	54	9		X	X		X	X		X			X		X		X	X	X		X	X	X		X	X		X		X		X		X	X	X		X	X	X	X		
333	225	117	182	64	123	39	178	67	143	44	106	220	59	24	207	130			X		X					X		X	X	X	X	X	X			X	X	X	X	X	X			X		X	X	X	X			X			X		X
334	45	26	178	56	249	24	97	76	184	120	246	10	188	133	164	217		X	X						X				X		X		X					X	X	X	X		X	X		X	X			X		X	X	X	X	X	X
335	247	94	71	97	60	229	180	133	35	202	80	186	46	113	17	108	X	X	X		X	X	X	X	X	X	X	X						X					X		X				X		X		X	X	X	X	X		X	X	
336	139	8	241	43	211	135	147	238	85	171	195	218	145	119	158	3	X	X		X		X	X	X	X			X	X	X			X	X							X	X		X		X	X	X	X	X	X	X	X		X		
337	93	107	195	146	194	226	53	112	44	75	41	95	114	30	16	98			X			X	X	X	X		X	X	X	X				X		X		X		X	X	X		X	X	X		X		X			X	X			
338	195	226	44	95	16	110	120	157	221	197	179	69	31	125	94	20		X	X			X	X	X	X	X	X	X	X	X	X			X			X	X		X		X			X	X	X	X				X	X	X			
339	64	67	220	130	110	80	10	240	9	69	78	109	181	90	33	168	X	X				X	X		X	X			X		X	X	X	X			X	X		X		X	X	X	X					X	X	X	X		X		
340	222	12	15	5	227	45	16	70	136	1	46	26	81	197	32	237		X	X	X		X			X		X	X	X	X				X	X		X		X	X	X	X	X	X		X	X	X		X	X	X	X	X		X	
341	9	242	211	68	244	171	225	115	158	2	194	48	233	117	72	46		X		X	X	X		X		X	X			X	X			X	X	X	X	X	X		X	X	X		X	X	X		X	X	X	X	X			X	
342	207	221	102	33	71	7	214	201	151	229	248	244	3	19	35	122					X		X			X	X	X			X	X		X	X	X	X		X			X			X	X			X			X	X		X	X	X
343	23	189	88	37	5	166	95	71	216	70	143	91	171	157	26	229			X	X	X	X		X					X	X	X		X	X	X				X	X	X		X	X	X	X	X		X					X			X
344	166	91	142	35	56	58	145	200	204	76	57	83	51	73	10	131			X	X	X	X	X		X			X	X	X		X	X	X		X	X	X		X		X	X	X	X				X	X	X				X		
345	92	55	239	250	146	89	237	164	67	112	65	206	193	17	95	84		X	X	X	X	X			X	X	X	X	X	X		X	X	X		X		X		X		X	X		X		X	X	X	X			X				X
346	11	198	63	16	135	247	66	197	37	218	64	94	105	74	170	165	X	X			X		X		X					X	X			X		X	X	X		X				X	X	X			X	X		X	X	X	X	X	
347	194	75	16	132	238	197	106	129	94	3	89	165	82	176	50	124		X	X		X			X		X	X		X	X	X	X	X		X		X		X	X		X		X	X		X	X		X	X		X	X			X
348	129	124	233	139	29	4	34	8	108	57	19	241	110	209	144	43			X					X					X		X	X	X	X		X	X	X		X		X	X		X	X		X	X	X				X	X	X	X
349	180	113	153	13	139	11	148	194	213	8	228	198	130	99	241	75		X	X				X	X			X	X	X	X	X		X		X	X	X	X	X	X		X		X		X								X		X	X
350	223	59	194	18	14	75	79	181	16	77	55	132	19	177	238	27		X	X			X	X	X	X			X	X			X	X	X	X	X	X		X		X					X		X			X		X	X			X
351	109	105	68	172	122	115	6	156	48	23	18	46	139	239	167	207								X		X		X	X		X	X			X				X	X	X	X	X	X			X	X		X	X	X	X		X	X	X
352	187	251	42	144	101	128	190	47	121	28	73	234	71	14	36	158				X	X	X		X	X		X	X	X	X			X	X		X		X	X					X	X	X	X		X		X				X	X	
353	233	4	108	241	144	203	59	135	12	47	82	85	33	18	234	218	X			X	X	X		X		X	X	X			X	X	X		X	X	X	X	X	X	X		X	X	X	X			X	X	X	X	X	X	X	X	
354	174	54	61	246	216	101	80	151	160	142	233	28	170	109	86	81	X	X			X	X	X	X	X	X		X	X	X	X	X		X				X	X		X		X	X	X				X	X		X			X		X

Figure A.10

For each value of counter: **DataID value to be used**

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
355	213	63	189	247	85	37	112	94	166	170	39	71	68	98	70	97
356	18	181	132	27	25	129	245	232	165	143	164	124	195	173	49	248
357	97	133	186	108	42	222	13	203	131	128	105	12	102	194	121	85
358	76	217	90	199	196	154	203	87	228	215	72	92	86	85	93	32
359	193	96	119	175	117	159	216	179	27	64	30	224	116	142	39	103
360	246	151	28	81	58	118	105	72	243	83	241	230	202	172	138	140
361	212	114	134	225	168	31	183	117	77	82	99	182	22	136	172	64
362	16	197	94	165	50	97	205	233	229	187	130	133	156	22	202	4
363	141	9	57	242	7	211	114	68	47	244	13	171	165	225	81	115
364	125	196	192	215	22	93	234	116	236	223	100	107	184	40	13	155
365	107	146	226	112	75	95	30	98	110	132	162	157	225	219	197	180
366	25	143	49	141	150	29	103	9	251	102	180	57	16	212	246	242
367	90	154	228	92	93	6	5	55	117	107	188	239	21	70	195	250
368	4	241	203	135	47	85	18	218	5	158	123	170	201	181	40	50
369	118	230	163	188	137	79	46	235	210	104	170	177	121	150	96	65
370	244	2	72	167	131	140	123	88	188	15	238	191	144	44	183	166
371	134	31	77	182	172	25	41	39	3	156	66	143	13	162	46	220
372	29	57	144	211	151	47	134	171	234	81	11	158	97	31	118	48
373	227	1	32	53	99	249	50	30	250	210	166	120	100	187	66	219
374	192	93	236	107	13	195	1	146	123	194	104	226	212	53	198	112
375	40	38	53	62	79	30	49	126	120	177	71	219	15	29	106	160
376	163	79	210	177	96	106	54	245	74	175	237	176	87	246	159	173
377	250	164	206	84	98	214	169	192	240	180	204	19	175	121	20	93
378	224	152	248	52	80	190	138	134	8	109	215	14	98	137	242	31
379	13	194	198	75	43	16	210	132	247	238	239	197	168	106	218	129
380	211	171	158	48	72	127	182	174	38	140	16	51	108	39	188	54
381	1	53	249	30	210	120	187	219	164	106	91	125	183	251	74	231
382	137	104	96	41	6	175	166	162	159	239	62	179	148	91	64	184
383	160	36	161	243	21	148	171	163	116	185	12	99	76	48	209	79
384	19	236	82	195	11	123	32	226	156	198	96	44	52	249	63	95
385	99	210	66	106	119	74	61	176	98	159	56	205	92	101	27	78
386	146	112	95	98	132	157	219	180	69	129	184	20	117	231	165	113
387	71	229	35	186	17	200	153	222	83	169	242	131	54	11	217	12
388	55	250	89	164	112	206	17	84	130	98	86	214	96	169	157	192
389	203	85	5	170	40	70	132	60	26	38	207	237	2	129	53	202
390	165	233	133	4	251	108	223	241	222	144	168	203	221	59	128	135
391	3	49	187	29	54	251	152	57	42	246	20	144	247	52	101	211
392	218	50	60	187	51	202	232	251	17	61	69	42	189	248	126	144
393	238	3	50	49	174	187	224	29	202	54	157	251	63	152	61	57
394	82	123	156	44	63	207	250	110	150	247	159	221	14	164	37	69
395	133	108	222	203	128	12	194	85	15	234	172	5	7	75	227	170
396	65	86	173	204	184	21	244	138	34	208	128	185	30	2	212	137
397	239	89	67	206	95	130	76	214	80	157	173	240	119	217	69	19
398	61	101	160	28	86	36	242	118	161	204	108	243	237	68	21	230
399	119	159	27	224	39	232	86	152	124	220	125	248	146	204	141	52
400	231	161	215	148	34	116	158	99	107	209	15	155	10	127	59	210
401	200	131	100	15	199	183	63	45	104	87	48	136	28	247	92	26
402	219	231	196	161	78	215	47	148	93	34	131	116	24	158	223	99
403	140	191	235	216	136	65	221	142	245	178	187	86	40	33	162	58
404	42	128	121	234	36	227	43	40	87	243	23	1	35	238	148	38
405	169	121	199	227	161	87	135	1	92	148	167	32	58	218	116	53
406	63	247	37	94	170	71	98	97	91	60	220	229	115	180	237	133
407	44	110	221	69	94	33	84	145	7	97	152	201	25	192	229	73
408	69	145	201	73	133	122	93	213	2	108	134	23	141	107	222	63
409	106	176	205	78	224	22	28	34	113	152	217	223	250	118	248	209
410	36	243	148	163	185	99	48	79	155	193	5	210	217	174	147	177
411	243	163	99	79	193	210	174	177	66	96	70	106	199	54	119	245
412	167	88	191	166	45	216	110	91	65	26	49	142	158	69	178	35
413	62	126	219	160	245	231	57	36	196	173	186	161	26	211	78	243

For for a message with length [bytes]: **" ": not yet assigned , "X": not allowed**

#	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
355		X		X	X	X	X	X	X	X	X	X	X	X		X	X	X	X			X	X	X	X	X		X	X				X	X	X	X	X	X	X		X
356	X		X	X	X	X		X	X			X		X	X	X	X		X				X	X	X		X	X	X	X	X	X					X	X		X	
357	X	X			X	X	X		X		X	X	X		X	X		X		X	X	X		X	X		X	X	X	X	X	X	X		X	X		X	X	X	X
358	X	X		X	X	X		X		X	X		X	X	X		X			X	X	X		X		X	X	X	X		X								X	X	X
359	X	X	X		X	X	X	X		X			X	X			X		X		X	X	X		X	X		X	X	X		X	X	X	X	X	X	X	X		X
360		X		X	X	X				X	X	X	X	X	X		X	X	X		X	X	X	X	X		X	X					X		X		X	X		X	
361	X		X	X			X	X	X		X	X	X	X	X	X		X	X	X		X	X	X	X			X				X		X		X	X	X	X	X	X
362		X		X	X		X	X	X	X	X	X	X			X	X					X	X		X				X		X	X		X		X	X		X	X	X
363	X	X	X	X	X	X			X		X	X	X	X	X	X		X		X	X	X		X	X	X	X			X						X	X		X		X
364	X	X	X	X	X	X			X	X		X		X		X	X					X	X			X				X	X	X	X		X	X	X			X	
365	X			X	X	X		X			X		X			X		X	X				X		X		X		X	X			X	X			X	X			X
366		X	X	X	X	X			X	X	X						X	X	X	X	X	X	X	X		X		X	X		X	X	X	X	X			X	X		
367	X					X		X	X	X	X	X	X	X	X	X	X			X	X	X	X	X					X		X		X	X	X		X		X	X	X
368		X	X	X		X	X		X		X		X	X			X		X		X	X	X	X	X	X	X		X	X			X	X	X		X	X	X		
369		X	X	X		X			X			X	X	X	X	X	X	X	X		X	X	X		X	X		X	X		X				X		X	X	X	X	X
370	X		X			X			X	X		X	X	X	X	X	X	X	X	X		X	X		X		X					X	X		X	X		X	X		X
371		X	X	X	X				X	X			X		X	X		X	X	X		X	X		X			X		X			X	X	X	X		X	X		X
372				X			X	X	X	X		X		X		X		X		X	X	X	X		X			X	X	X	X	X	X	X	X				X		X
373		X	X		X	X	X	X			X		X				X	X	X	X	X	X	X	X	X					X	X	X			X	X			X	X	X
374		X		X		X		X	X		X		X	X	X					X	X	X	X			X	X		X	X	X	X	X		X	X		X		X	X
375			X		X	X	X	X	X	X	X	X			X	X	X	X	X	X	X			X	X	X	X		X	X	X	X	X	X	X	X	X	X	X		X
376			X		X	X	X	X	X	X	X			X	X	X	X	X	X	X		X		X	X		X	X	X		X			X	X			X	X	X	X
377								X	X	X	X		X	X		X	X	X	X	X	X	X			X	X	X	X	X	X	X		X	X	X		X		X	X	X
378	X	X	X		X		X	X	X	X	X	X	X		X	X	X	X	X	X	X		X				X	X		X				X		X		X	X	X	
379		X				X	X	X		X			X	X	X				X	X	X	X	X	X		X			X		X	X		X		X	X	X	X	X	
380	X	X	X		X		X			X	X	X			X	X		X	X		X	X		X		X	X	X	X					X		X	X	X	X		
381		X	X	X	X		X	X	X	X	X			X			X					X	X	X		X	X		X			X	X	X	X	X		X	X	X	X
382	X	X	X	X	X	X	X	X		X		X		X	X	X	X				X	X	X			X		X		X		X			X	X	X	X	X	X	X
383	X	X	X			X			X	X			X	X		X		X				X			X		X	X	X							X	X		X	X	X
384			X	X			X		X	X	X	X	X	X		X	X				X	X			X		X	X	X	X		X	X	X	X		X		X	X	X
385	X		X		X	X		X	X	X	X		X			X		X	X		X		X	X	X	X	X		X	X	X	X	X					X	X	X	X
386		X	X	X	X	X	X	X	X		X	X		X			X	X		X		X		X	X				X	X	X	X				X				X	
387				X		X		X		X	X	X		X	X	X	X		X	X	X		X	X	X	X		X	X	X	X	X		X	X		X		X	X	
388	X	X	X		X	X		X	X			X		X	X	X		X	X	X		X			X		X	X				X			X			X		X	
389	X	X		X	X	X		X	X			X	X			X	X	X	X		X			X				X	X	X		X	X	X	X	X	X	X		X	X
390	X							X	X	X		X		X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X		X	X						X		X
391				X	X	X	X	X			X		X		X	X	X		X	X				X	X	X			X		X	X	X		X		X	X	X	X	
392	X	X			X		X			X				X						X	X	X				X	X			X	X				X		X	X	X	X	
393				X			X	X	X	X			X		X	X	X	X			X	X	X		X	X						X	X	X	X		X		X	X	X
394	X					X	X	X	X	X	X	X	X						X				X		X	X	X	X	X	X		X	X	X				X	X	X	X
395	X	X	X	X	X	X	X			X		X		X	X	X	X		X			X			X		X	X	X				X		X	X	X		X		
396		X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X	X	X	X				X	X	X	X		X	X		X	X	X	X		X	X	
397	X	X			X		X			X	X	X	X			X	X		X	X	X		X		X		X		X	X	X		X		X				X		X
398	X		X	X			X		X	X		X		X	X	X		X	X		X	X			X	X	X				X	X	X		X	X	X	X	X	X	X
399	X	X	X	X	X	X	X		X		X		X	X			X	X			X	X			X	X	X	X		X		X			X	X	X	X	X		
400	X	X	X	X	X		X	X	X	X		X	X	X	X	X	X	X			X	X			X	X	X	X	X	X	X	X	X	X	X					X	
401	X	X						X	X	X		X		X	X	X					X	X	X		X	X		X	X	X	X	X		X	X	X	X		X		
402	X	X	X		X		X	X	X			X	X		X		X				X	X	X	X		X			X	X	X		X			X	X	X			
403		X	X	X			X	X	X			X	X	X	X	X			X	X		X	X	X	X	X			X		X	X		X		X			X	X	X
404	X	X			X	X	X	X	X						X		X						X		X	X	X		X	X	X	X	X	X	X						X
405	X	X			X		X		X			X	X	X	X	X	X		X			X	X	X			X	X	X	X	X				X	X	X	X	X		
406		X		X	X	X		X		X			X		X		X	X		X	X			X	X	X	X	X		X		X		X		X	X	X		X	X
407	X	X	X			X	X				X		X	X	X	X	X	X	X	X		X		X	X	X		X	X	X	X	X	X	X	X	X		X	X		X
408		X	X	X	X		X		X		X	X		X	X	X	X			X					X	X	X	X	X		X	X	X	X	X		X		X	X	
409		X		X	X	X			X	X			X			X			X					X	X	X		X	X	X		X		X	X	X	X	X		X	X
410	X		X		X	X		X	X	X			X	X					X		X	X		X		X	X	X		X		X		X	X	X	X	X			X
411	X				X		X	X		X	X		X	X	X	X		X	X	X	X	X				X		X		X	X	X	X				X	X		X	X
412		X		X		X	X	X			X	X		X	X	X	X	X	X	X	X	X		X	X	X				X			X		X	X	X	X	X	X	X
413		X	X	X	X		X	X	X		X	X	X		X	X		X	X	X		X		X	X		X	X		X				X	X		X				

Figure A.11

For each value of counter: **DataID value to be used** | For for a message with length [bytes]: **" ": not yet assigned , "X": not allowed**

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
414	151	81	118	72	83	230	172	140	163	100	135	188	42	156	137	191	X		X	X		X						X				X		X			X	X	X		X	X				X	X	X	X	X	X	X	X		X			
415	232	248	139	190	9	8	185	14	241	242	93	43	157	193	211	77	X	X	X			X	X	X	X			X	X		X				X				X	X	X		X	X		X	X	X	X	X	X	X	X		X		
416	5	70	26	237	53	56	165	17	24	30	102	76	140	233	120	169		X	X		X			X	X	X		X	X	X			X		X			X		X		X		X	X	X	X			X	X	X	X	X	X		X
417	127	51	62	61	235	126	141	101	219	65	97	160	5	9	245	28	X	X	X	X	X	X	X		X		X			X	X				X			X	X			X	X	X		X		X	X		X	X	X			X	X
418	161	148	116	99	209	155	127	210	146	147	45	66	90	51	18	106		X	X		X		X	X	X	X		X	X				X	X	X	X				X		X	X		X	X	X		X		X			X			
419	154	92	6	55	107	239	70	250	64	146	235	89	185	237	226	164		X		X	X	X	X	X	X	X		X	X		X	X		X					X	X	X		X		X	X	X	X	X			X	X		X		X
420	235	65	245	86	162	173	7	204	78	184	42	21	53	244	103	138					X	X	X		X	X	X			X		X				X		X			X			X		X	X					X	X				
421	14	77	238	25	115	3	175	143	50	46	112	49	11	179	174	141	X		X	X	X	X		X	X	X	X		X	X	X	X		X	X	X	X	X	X		X	X		X	X	X	X		X	X	X	X			X			
422	21	185	209	193	114	147	140	96	18	225	1	119	196	191	31	175	X	X	X		X		X		X	X	X		X	X	X	X		X			X		X		X	X	X		X	X	X		X	X			X	X			
423	12	5	45	70	1	26	197	237	178	53	150	56	72	165	249	17	X		X		X	X			X		X	X	X	X	X	X		X		X			X		X	X	X	X	X		X	X	X		X		X			X	
424	126	160	231	36	173	161	211	243	215	21	222	148	56	171	34	163	X			X		X	X		X		X	X		X	X		X	X		X		X				X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
425	136	178	162	24	250	184	229	10	103	164	101	208	79	186	206	90	X		X	X		X	X		X	X		X	X							X	X		X		X	X	X	X	X		X		X	X	X		X			X	X
426	95	157	69	20	165	145	196	153	201	233	212	73	39	215	133	11	X			X	X			X	X	X						X							X	X	X		X	X	X	X		X	X	X	X			X		X	X
427	177	245	176	173	179	78	151	21	22	103	169	34	249	81	152	185	X	X	X	X	X	X	X				X		X		X		X	X	X			X		X		X	X	X	X	X	X	X		X		X			X	X	X
428	57	211	47	171	81	158	31	48	40	72	198	127	133	182	230	174	X		X	X		X			X	X		X		X		X		X	X		X	X		X						X				X	X	X	X	X	X	X	
429	210	106	74	176	159	205	101	78	180	224	76	22	55	28	232	34	X		X		X	X		X	X	X	X		X	X			X	X	X	X	X	X			X	X	X	X		X		X		X		X		X	X	X	
430	124	139	4	8	57	241	209	43	203	211	236	135	69	147	47	238	X	X	X	X		X		X	X	X		X		X		X		X	X		X	X	X	X		X	X			X	X		X	X		X			X	X	X
431	54	246	101	151	142	28	109	81	36	58	4	118	60	105	204	72	X				X		X		X		X	X	X			X	X	X			X	X		X	X		X		X		X		X		X		X				
432	182	39	143	220	207	141	184	80	29	221	205	9	75	208	102	109		X		X	X			X		X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X		X		X	X	X	X		X
433	96	175	159	179	64	224	142	103	232	67	219	152	155	58	220	212	X	X	X		X				X		X			X		X	X	X	X	X	X	X	X	X	X	X			X	X		X	X		X	X		X	X		X
434	15	45	136	26	32	178	94	56	162	249	54	24	230	97	250	76	X	X	X	X		X			X	X	X	X				X			X	X		X	X	X	X		X			X	X			X		X	X	X	X		X
435	152	52	190	134	109	14	137	31	43	105	116	77	180	104	68	182	X	X		X	X		X	X		X				X			X	X	X	X	X	X	X	X		X		X	X	X	X	X			X	X	X	X		X	X
436	248	190	8	14	242	43	193	77	135	68	107	238	20	96	171	25		X		X	X			X		X	X				X	X	X	X	X	X			X	X	X			X	X		X		X	X	X			X	X	X	X
437	173	21	34	185	212	209	72	193	59	114	227	147	125	140	134	96		X						X	X	X	X			X				X	X	X	X	X	X	X	X	X	X		X		X	X			X	X	X		X		X
438	51	61	126	101	65	160	9	28	231	86	133	36	70	242	173	118	X	X	X		X	X	X	X	X	X				X				X	X	X	X		X		X	X	X	X	X			X	X	X	X	X	X	X	X	X	X
439	28	118	243	230	138	163	115	188	99	137	85	79	169	46	193	235	X	X	X		X		X		X		X	X	X	X	X			X	X	X	X	X	X	X		X	X						X	X		X			X		
440	236	195	123	226	198	44	249	95	207	16	175	110	134	120	247	157	X		X		X	X	X	X	X	X		X	X	X	X	X		X	X		X	X	X	X		X			X	X			X			X	X	X			
441	83	100	137	183	154	104	189	136	96	92	127	41	36	37	6	178		X	X	X	X	X	X	X		X					X	X		X	X	X	X			X		X		X	X	X	X	X	X		X				X	X	X
442	158	127	38	51	188	62	143	61	30	235	94	126	12	141	177	101	X	X				X	X	X	X	X	X		X	X	X		X		X		X	X			X	X	X			X		X		X		X			X	X	X
443	32	249	250	120	66	164	202	125	206	74	142	84	104	42	98	196			X		X		X	X	X			X	X	X	X	X	X		X	X		X		X			X		X	X		X		X	X	X	X	X	X		X
444	181	27	129	232	143	124	173	248	233	141	84	139	226	21	29	190		X					X	X		X	X			X	X		X	X	X		X	X		X	X	X			X		X	X	X	X	X	X			X		X
445	251	144	128	47	28	234	14	158	227	118	213	40	229	77	243	127	X	X			X	X	X	X	X	X		X		X	X	X	X	X		X	X	X	X	X		X	X		X	X			X	X	X				X	X	X
446	58	83	138	100	90	137	23	183	193	154	158	104	160	189	228	136		X		X	X	X		X		X	X	X	X	X	X		X	X	X	X		X				X		X	X	X		X		X	X	X	X	X	X		X
447	142	58	204	83	10	138	122	100	185	90	47	137	126	23	149	183	X	X		X	X	X		X		X		X		X	X	X			X	X		X	X		X		X		X	X	X			X	X	X	X	X		X	X
448	131	15	183	45	87	136	247	26	41	32	174	178	118	94	55	56			X	X				X	X	X	X		X	X		X	X		X	X	X	X		X	X	X			X	X	X	X	X	X					X		
449	153	11	213	198	241	63	155	16	189	135	117	247	109	66	85	197	X		X	X				X	X	X	X		X	X	X	X			X	X	X	X			X	X		X			X	X	X		X	X			X	X	
450	73	213	23	63	203	189	146	247	88	85	182	37	242	112	5	94	X	X		X	X			X		X		X			X	X		X		X	X		X	X	X	X		X	X	X	X	X		X	X		X		X		
451	30	219	125	231	176	196	144	161	192	78	200	215	178	47	22	148	X	X		X	X	X		X	X		X	X	X		X		X	X	X		X				X	X	X	X	X	X	X	X	X	X	X	X	X	X			
452	7	244	81	2	200	72	82	167	230	131	43	140	251	123	100	88	X	X	X	X	X		X				X	X		X		X	X	X	X	X	X	X	X	X		X	X				X	X	X	X	X	X	X	X	X			X
453	178	24	184	10	164	208	186	90	212	84	28	149	177	222	214	154		X	X	X	X			X		X	X		X		X	X	X	X	X	X		X	X				X	X		X					X	X		X		X	X
454	132	129	165	124	49	233	78	139	133	29	214	4	44	34	251	8		X	X	X	X	X	X		X	X	X			X	X	X	X	X	X	X	X	X					X		X	X		X	X	X	X			X	X	X	
455	237	17	76	169	219	217	4	121	90	231	244	199	216	241	196	227			X	X		X		X	X	X		X		X		X			X		X	X	X		X	X	X	X	X	X	X					X			X	X	
456	214	19	168	236	153	82	87	195	172	11	193	123	152	32	213	226	X					X	X							X	X		X	X			X	X	X			X	X	X	X	X	X			X	X				X		X
457	46	150	54	102	166	246	130	7	101	91	124	151	218	240	142	244	X				X		X	X	X	X	X		X		X	X	X		X	X		X			X		X	X	X			X		X	X	X		X			X
458	75	132	197	129	3	165	176	124	97	49	206	233	123	78	187	139	X		X	X	X	X	X	X	X			X		X								X		X	X	X		X	X	X	X	X		X			X	X	X	X	
459	183	136	41	178	55	162	71	24	179	250	61	184	163	229	89	10	X			X		X	X		X					X			X	X			X	X	X	X	X	X	X		X	X	X	X	X	X	X			X		X	X
460	68	115	48	46	167	174	64	150	51	88	132	54	241	67	191	102			X		X	X				X	X	X	X	X		X		X	X	X		X	X		X				X	X		X	X			X			X	X	
461	67	130	80	240	69	109	90	168	242	145	34	105	27	154	201	82			X			X		X		X	X				X	X	X	X	X	X			X	X	X	X		X	X		X		X		X					X	X
462	199	87	92	32	116	55	170	249	239	155	191	250	138	60	146	120		X	X		X	X				X	X	X	X			X	X	X	X	X	X		X		X	X			X	X	X	X		X		X	X	X		X	X
463	2	167	140	88	15	191	44	166	235	45	3	216	47	110	136	91	X						X		X	X		X	X	X	X					X		X	X		X			X	X	X	X	X	X			X	X		X	X	
464	121	227	87	1	148	32	218	53	55	99	88	249	83	50	155	30			X		X	X	X	X		X	X	X	X	X	X			X	X		X	X		X	X	X	X	X	X			X		X			X		X	X	
465	221	33	7	201	229	244	19	122	81	186	190	2	49	236	200	23	X	X	X		X	X	X			X		X	X			X		X	X		X			X	X	X	X					X	X			X	X	X	X		
466	191	216	65	142	178	86	33	58	173	24	251	204	38	201	184	83	X			X	X		X		X	X	X	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X					X				X	X	X	X	X
467	81	72	230	140	100	188	156	191	79	183	218	235	128	207	104	216		X	X		X			X	X		X	X	X		X	X	X			X		X	X	X		X	X	X		X			X		X	X	X		X	X	X
468	88	166	216	91	26	142	69	35	86	56	29	58	127	145	24	200		X	X		X	X	X	X	X	X	X	X	X				X	X			X	X	X	X		X		X		X	X		X			X			X	X	X
469	60	202	17	42	126	169	139	128	217	160	201	121	166	8	231	234	X	X				X	X	X	X	X	X	X		X	X		X		X			X	X		X			X				X	X	X		X			X		
470	201	122	2	23	222	167	195	189	140	12	77	88	57	226	15	37	X	X	X	X	X		X		X			X				X		X	X	X			X	X	X	X	X	X	X	X		X	X		X		X	X	X	X	
471	157	20	145	153	233	73	215	11	122	4	114	213	220	116	108	198		X					X		X	X	X			X	X	X	X	X	X	X	X	X	X	X			X			X		X	X		X	X			X	X	X	
472	115	46	174	150	88	54	67	102	61	166	129	246	135	130	216	7						X	X		X				X	X	X		X		X		X		X			X	X		X	X				X	X	X		X	X	X	X

Figure A.12

For each value of counter: **DataID value to be used** | For for a message with length [bytes]: **" ": not yet assigned , "X": not allowed**

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
473	48	174	51	54	191	61	220	246	126	216	165	101	85	80	65	151	X	X		X	X		X		X				X	X	X	X		X		X			X	X	X	X	X		X	X	X	X					X		X	X	X
474	206	214	240	19	20	168	199	236	105	153	185	82	224	87	73	195		X	X		X	X		X		X	X	X		X		X		X	X	X	X	X	X	X								X	X	X			X		X	X		X
475	145	73	122	213	108	23	107	63	167	203	31	189	9	146	12	247	X	X	X	X	X	X	X	X	X	X	X			X	X		X	X		X	X		X	X	X	X	X	X			X	X	X	X		X	X				X
476	245	173	78	21	103	34	81	185	223	212	121	209	120	72	52	193	X	X			X	X		X	X	X	X		X		X	X	X	X	X	X		X			X		X				X	X		X		X		X	X	X	X
477	128	234	227	40	243	1	238	38	32	163	189	53	200	3	99	62			X	X				X		X	X		X	X		X	X		X			X	X	X	X		X	X			X		X	X				X	X		X
478	170	60	237	202	62	17	124	42	76	126	33	169	88	139	219	128	X	X	X		X	X			X					X				X	X	X			X		X		X			X			X	X	X	X	X	X	X	X	X
479	78	34	223	209	52	59	230	147	194	134	87	18	84	188	14	119	X		X	X		X	X		X	X	X		X				X			X	X		X	X		X	X	X		X			X		X	X	X			X	X
480	230	188	79	235	104	177	150	65	106	41	60	245	227	102	175	86		X	X		X	X			X		X	X				X	X			X	X		X			X				X		X	X	X	X	X	X	X	X		X
481	110	69	33	145	97	201	192	73	244	133	52	122	143	93	186	213		X			X	X	X	X	X	X		X	X		X	X	X	X			X	X	X	X		X			X	X	X			X		X	X		X		
482	155	66	112	74	181	98	126	205	157	27	24	180	6	160	129	22	X	X	X	X		X		X	X				X				X			X				X	X	X		X	X		X	X	X			X	X		X	X	X
483	215	116	107	155	59	146	38	66	226	18	136	112	149	62	75	74	X	X			X		X	X	X	X			X			X	X	X	X				X	X	X	X	X	X				X	X	X	X	X	X	X			X
484	84	192	19	93	113	236	227	107	82	13	137	195	103	1	11	146	X		X	X	X	X		X		X	X	X	X	X		X	X	X	X	X		X	X	X	X		X	X		X	X	X		X			X		X	X	X
485	91	35	58	200	76	83	73	131	138	217	211	100	61	213	90	15	X	X		X		X	X	X			X		X	X	X	X			X	X		X		X			X					X	X	X	X	X			X	X	X
486	175	179	224	103	67	152	58	212	248	130	231	52	66	83	80	114	X	X	X	X			X		X	X	X	X	X	X	X		X		X				X		X				X	X		X	X	X	X	X	X	X	X	X	X
487	148	99	155	210	147	66	51	106	112	119	26	74	154	61	181	176	X	X		X	X			X	X			X	X					X			X	X	X		X	X	X	X	X	X				X	X	X				X	X
488	204	138	185	137	149	193	167	104	147	228	40	96	231	88	225	41	X		X	X	X	X	X		X			X	X	X		X		X	X	X	X	X	X	X		X	X		X	X	X	X		X	X	X		X	X	X	X
489	113	13	11	194	8	198	99	75	63	43	6	16	240	210	135	132	X			X	X	X	X	X	X				X		X	X	X		X			X		X		X	X	X	X				X			X	X	X		X	X
490	10	90	149	154	192	228	12	92	225	93	230	6	173	5	236	55	X		X				X	X	X		X	X	X	X	X	X	X		X	X	X	X	X			X	X	X	X			X	X	X				X	X	X	
491	112	98	157	180	129	20	231	113	145	124	208	153	64	161	233	13	X	X		X	X	X	X	X	X	X	X		X	X	X			X		X	X	X			X	X	X		X			X		X	X	X	X			X	
492	74	205	180	22	232	113	36	223	153	248	90	13	89	243	139	59	X	X	X	X			X	X					X	X	X	X	X		X			X	X	X	X			X	X		X				X	X		X		X	X
493	26	56	24	76	120	10	133	217	208	125	151	90	235	108	84	199		X			X	X	X				X	X		X	X	X	X	X		X		X	X	X			X			X			X					X	X	X	X
494	220	80	9	109	33	242	149	105	211	201	223	68	129	228	244	172		X			X	X	X	X			X		X	X		X	X	X		X		X	X	X			X	X			X		X	X	X	X			X	X	
495	196	215	93	116	223	107	40	155	195	59	183	146	208	38	194	66	X	X	X		X	X			X	X			X	X	X		X			X	X	X		X	X	X		X		X	X	X	X	X	X		X	X			
496	85	170	70	60	38	237	129	202	56	62	221	17	167	124	30	42				X	X	X	X	X			X			X		X		X	X	X		X		X	X		X	X	X			X					X	X	X	X	
497	130	240	109	168	145	105	154	82	68	73	209	172	232	92	122	123	X			X	X	X	X		X	X	X					X	X	X	X	X			X	X			X	X			X	X	X	X	X	X		X	X		X
498	52	134	14	31	105	77	104	182	238	172	155	25	113	41	115	39	X		X	X	X	X			X		X		X	X				X		X		X	X	X	X		X	X	X	X		X	X		X	X	X			X	X
499	59	18	75	181	77	132	177	27	197	25	250	129	236	245	3	232	X	X	X		X	X		X	X	X		X	X	X	X	X	X	X			X	X	X		X	X	X		X		X	X	X	X	X	X	X				X
500	143	141	29	9	102	57	212	242	144	7	113	211	197	114	151	68	X	X		X		X			X		X				X	X			X	X	X		X	X	X		X	X	X				X	X	X	X		X	X	X	X

Figure A.13

B Constraint History

B.1 Constraint History R19-11

B.1.1 Added Constraints

Number	Heading
Constraint 03176 in chapter 8.1.1 "E2E State Machine Settings"	Value range of windowSizeValid
Constraint 03177 in chapter 8.1.1 "E2E State Machine Settings"	Dependency between maxErrorStateValid, maxErrorStateInit and maxErrorStateInvalid
Constraint 03178 in chapter 8.1.1 "E2E State Machine Settings"	Dependency between minOkStateValid, minOkStateInit and minOkStateInvalid
Constraint 03179 in chapter 8.1.1 "E2E State Machine Settings"	Dependency between minOkStateInit, maxErrorStateInit and windowSizeInit
Constraint 03180 in chapter 8.1.1 "E2E State Machine Settings"	Dependency between minOkStateValid, maxErrorStateValid and windowSizeValid
Constraint 03181 in chapter 8.1.1 "E2E State Machine Settings"	Dependency between minOkStateInvalid, maxErrorStateInvalid and windowSizeInvalid
Constraint 06300 in chapter 9.2 "Client-Server Communication"	MaxDeltaCounter for Client-Server Communication
Constraint 06301 in chapter 8.1.1 "E2E State Machine Settings"	Dependency between windowSizeInvalid and windowSizeValid
Constraint 06302 in chapter 8.1.1 "E2E State Machine Settings"	Dependency between windowSizeInit and windowSizeValid

Table B.1: added Constraints in R19-11

B.1.2 Changed Constraints

N/A

B.1.3 Deleted Constraints

N/A

B.1.4 Added Specification Items

N/A

B.1.5 Changed Specification Items

N/A

B.1.6 Deleted Specification Items

N/A

B.2 Constraint History R20-11

B.2.1 Added Constraints in R20-11

none

B.2.2 Changed Constraints in R20-11

none

B.2.3 Deleted Constraints in R20-11

none

B.2.4 Added Traceables in R20-11

Number	Heading
[PRS_E2E_00640]	
[PRS_E2E_00641]	Mapping Profile 1 to State Machine
[PRS_E2E_00644]	
[PRS_E2E_00645]	
[PRS_E2E_00646]	
[PRS_E2E_00647]	
[PRS_E2E_00648]	
[PRS_E2E_00649]	
[PRS_E2E_00650]	
[PRS_E2E_00651]	
[PRS_E2E_00652]	
[PRS_E2E_00653]	
[PRS_E2E_00654]	
[PRS_E2E_00655]	
[PRS_E2E_00656]	
[PRS_E2E_00657]	
[PRS_E2E_00658]	
[PRS_E2E_00659]	
[PRS_E2E_00660]	
[PRS_E2E_00661]	
[PRS_E2E_00662]	
[PRS_E2E_00663]	
[PRS_E2E_00664]	
[PRS_E2E_00665]	
[PRS_E2E_00666]	
[PRS_E2E_00667]	
[PRS_E2E_00668]	
[PRS_E2E_00669]	
[PRS_E2E_00670]	Mapping Profile 2 to State Machine
[PRS_E2E_00673]	
[PRS_E2E_00675]	
[PRS_E2E_00676]	
[PRS_E2E_00677]	
[PRS_E2E_00678]	
[PRS_E2E_00679]	
[PRS_E2E_00680]	

▽

△

Number	Heading
[PRS_E2E_00681]	
[PRS_E2E_00682]	
[PRS_E2E_00683]	
[PRS_E2E_00684]	
[PRS_E2E_00685]	
[PRS_E2E_00686]	
[PRS_E2E_00687]	
[PRS_E2E_00688]	
[PRS_E2E_00689]	
[PRS_E2E_00690]	
[PRS_E2E_00691]	
[PRS_E2E_00692]	
[PRS_E2E_00693]	
[PRS_E2E_00694]	
[PRS_E2E_00695]	
[PRS_E2E_00696]	
[PRS_E2E_00697]	
[PRS_E2E_00698]	
[PRS_E2E_00699]	
[PRS_E2E_00700]	
[PRS_E2E_00701]	
[PRS_E2E_00702]	
[PRS_E2E_00703]	
[PRS_E2E_00704]	
[PRS_E2E_00705]	
[PRS_E2E_00706]	
[PRS_E2E_00707]	
[PRS_E2E_00708]	
[PRS_E2E_00709]	
[PRS_E2E_00710]	
[PRS_E2E_00711]	
[PRS_E2E_00712]	
[PRS_E2E_00713]	
[PRS_E2E_00714]	
[PRS_E2E_00715]	
[PRS_E2E_00716]	
[PRS_E2E_00717]	
[PRS_E2E_00718]	
[PRS_E2E_00719]	

▽

△

Number	Heading
[PRS_E2E_00720]	
[PRS_E2E_00721]	
[PRS_E2E_00722]	
[PRS_E2E_00723]	
[PRS_E2E_00724]	
[PRS_E2E_00725]	
[PRS_E2E_00726]	
[PRS_E2E_00727]	
[PRS_E2E_00728]	
[PRS_E2E_00729]	
[PRS_E2E_00730]	
[PRS_E2E_00731]	
[PRS_E2E_00732]	
[PRS_E2E_00733]	
[PRS_E2E_00734]	
[PRS_E2E_00735]	
[PRS_E2E_00736]	
[PRS_E2E_00737]	
[PRS_E2E_00738]	Mapping Profile 44 to State Machine
[PRS_E2E_00739]	
[PRS_E2E_00740]	
[PRS_E2E_00741]	
[PRS_E2E_00742]	
[PRS_E2E_00743]	
[PRS_E2E_00744]	
[PRS_E2E_00745]	
[PRS_E2E_00746]	
[PRS_E2E_00747]	
[PRS_E2E_00748]	
[PRS_E2E_00749]	
[PRS_E2E_00750]	
[PRS_E2E_00751]	
[PRS_E2E_00752]	
[PRS_E2E_00753]	
[PRS_E2E_00756]	
[PRS_E2E_00757]	
[PRS_E2E_00758]	
[PRS_E2E_00759]	
[PRS_E2E_00760]	

▽

△

Number	Heading
[PRS_E2E_00761]	
[PRS_E2E_00762]	
[PRS_E2E_00763]	
[PRS_E2E_00764]	
[PRS_E2E_00765]	
[PRS_E2E_00766]	
[PRS_E2E_00767]	
[PRS_E2E_00768]	Draft
[PRS_E2E_00769]	Draft
[PRS_E2E_00770]	Draft
[PRS_E2E_00771]	Draft
[PRS_E2E_00772]	
[PRS_E2E_00773]	
[PRS_E2E_00774]	
[PRS_E2E_00775]	
[PRS_E2E_00776]	
[PRS_E2E_00777]	
[PRS_E2E_00778]	
[PRS_E2E_00779]	
[PRS_E2E_00780]	
[PRS_E2E_00781]	
[PRS_E2E_00783]	
[PRS_E2E_00784]	
[PRS_E2E_00785]	
[PRS_E2E_00787]	
[PRS_E2E_00788]	
[PRS_E2E_00789]	
[PRS_E2E_00790]	
[PRS_E2E_00791]	
[PRS_E2E_00792]	
[PRS_E2E_00793]	
[PRS_E2E_00794]	
[PRS_E2E_00795]	
[PRS_E2E_00796]	
[PRS_E2E_00799]	
[PRS_E2E_00800]	
[PRS_E2E_00801]	
[PRS_E2E_00802]	
[PRS_E2E_00803]	

▽

△

Number	Heading
[PRS_E2E_00804]	
[PRS_E2E_00805]	
[PRS_E2E_00806]	
[PRS_E2E_00807]	
[PRS_E2E_00808]	
[PRS_E2E_00809]	
[PRS_E2E_00810]	
[PRS_E2E_00811]	Draft
[PRS_E2E_00812]	Draft
[PRS_E2E_00813]	Draft
[PRS_E2E_00814]	Draft
[PRS_E2E_00815]	
[PRS_E2E_00816]	
[PRS_E2E_00817]	
[PRS_E2E_00818]	
[PRS_E2E_00819]	
[PRS_E2E_00820]	
[PRS_E2E_00821]	
[PRS_E2E_00822]	
[PRS_E2E_00823]	
[PRS_E2E_00824]	
[PRS_E2E_00825]	
[PRS_E2E_00826]	Mapping Profile 4m to State-Machine
[PRS_E2E_00827]	Mapping Profile 7m to State-Machine
[PRS_E2E_00828]	
[PRS_E2E_00829]	
[PRS_E2E_00830]	
[PRS_E2E_00831]	
[PRS_E2E_00832]	
[PRS_E2E_00834]	
[PRS_E2E_00835]	
[PRS_E2E_00836]	
[PRS_E2E_00837]	
[PRS_E2E_00840]	
[PRS_E2E_00841]	
[PRS_E2E_00842]	
[PRS_E2E_00843]	
[PRS_E2E_00844]	
[PRS_E2E_00848]	

▽

△

Number	Heading
[PRS_E2E_00849]	
[PRS_E2E_00850]	Mapping Profile 8 to State Machine
[PRS_E2E_UC_-00055]	
[PRS_E2E_UC_-00062]	
[PRS_E2E_UC_-00063]	
[PRS_E2E_UC_-00071]	
[PRS_E2E_UC_-00072]	
[PRS_E2E_UC_-00073]	
[PRS_E2E_UC_-00171]	
[PRS_E2E_UC_-00173]	
[PRS_E2E_UC_-00235]	
[PRS_E2E_UC_-00238]	
[PRS_E2E_UC_-00239]	
[PRS_E2E_UC_-00308]	
[PRS_E2E_UC_-00317]	
[PRS_E2E_UC_-00320]	
[PRS_E2E_UC_-00321]	
[PRS_E2E_UC_-00325]	
[PRS_E2E_UC_-00328]	
[PRS_E2E_UC_-00606]	
[PRS_E2E_UC_-00743]	
[PRS_E2E_UC_-00754]	
[PRS_E2E_UC_-00786]	

▽

△

Number	Heading
[PRS_E2E_UC_-00797]	
[PRS_E2E_USE_-00741]	

Table B.2: Added Traceables in R20-11

B.2.5 Changed Traceables in R20-11

Number	Heading
[PRS_E2E_00070]	
[PRS_E2E_00082]	
[PRS_E2E_00117]	
[PRS_E2E_00118]	
[PRS_E2E_00119]	
[PRS_E2E_00120]	
[PRS_E2E_00125]	
[PRS_E2E_00126]	
[PRS_E2E_00134]	
[PRS_E2E_00150]	
[PRS_E2E_00195]	
[PRS_E2E_00218]	
[PRS_E2E_00219]	
[PRS_E2E_00299]	
[PRS_E2E_00318]	
[PRS_E2E_00319]	
[PRS_E2E_00322]	
[PRS_E2E_00323]	
[PRS_E2E_00324]	
[PRS_E2E_00330]	
[PRS_E2E_00345]	
[PRS_E2E_00372]	
[PRS_E2E_00394]	
[PRS_E2E_00401]	
[PRS_E2E_00421]	
[PRS_E2E_00479]	
[PRS_E2E_00480]	
[PRS_E2E_00484]	

▽

△

Number	Heading
[PRS_E2E_00485]	
[PRS_E2E_00503]	
[PRS_E2E_00522]	
[PRS_E2E_00527]	
[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_00598]	Mapping Profile 1 to State Machine
[PRS_E2E_00599]	Mapping Profile 2 to State Machine
[PRS_E2E_00600]	Mapping Profile 4 to State Machine
[PRS_E2E_00601]	Mapping Profile 5 to State Machine
[PRS_E2E_00602]	Mapping Profile 6 to State Machine
[PRS_E2E_00603]	Mapping Profile 7 to State Machine
[PRS_E2E_00604]	Mapping Profile 11 to State Machine
[PRS_E2E_00605]	Mapping Profile 22 to State Machine
[PRS_E2E_00608]	
[PRS_E2E_00609]	
[PRS_E2E_00610]	
[PRS_E2E_00611]	
[PRS_E2E_00612]	
[PRS_E2E_00613]	
[PRS_E2E_00614]	
[PRS_E2E_00615]	
[PRS_E2E_00616]	
[PRS_E2E_00617]	
[PRS_E2E_00618]	
[PRS_E2E_00619]	
[PRS_E2E_00620]	
[PRS_E2E_00621]	
[PRS_E2E_00622]	
[PRS_E2E_00623]	
[PRS_E2E_00624]	
[PRS_E2E_00625]	
[PRS_E2E_00626]	

▽

△

Number	Heading
[PRS_E2E_00627]	
[PRS_E2E_00628]	
[PRS_E2E_00629]	
[PRS_E2E_00630]	
[PRS_E2E_00631]	
[PRS_E2E_00632]	
[PRS_E2E_00633]	
[PRS_E2E_00634]	
[PRS_E2E_00635]	
[PRS_E2E_00636]	
[PRS_E2E_00637]	
[PRS_E2E_00638]	
[PRS_E2E_00639]	
[PRS_E2E_UC_-00051]	
[PRS_E2E_UC_-00061]	
[PRS_E2E_UC_-00170]	
[PRS_E2E_UC_-00316]	
[PRS_E2E_UC_-00351]	
[PRS_E2E_UC_-00466]	

Table B.3: Changed Traceables in R20-11

B.2.6 Deleted Traceables in R20-11

Number	Heading
[PRS_E2E_00217]	
[PRS_E2E_00221]	
[PRS_E2E_00227]	
[PRS_E2E_00228]	
[PRS_E2E_00307]	
[PRS_E2E_00584]	
[PRS_E2E_00585]	

▽

△

Number	Heading
[PRS_E2E_00586]	
[PRS_E2E_00587]	
[PRS_E2E_UC_-00237]	
[PRS_E2E_USE_-00235]	
[PRS_E2E_USE_-00236]	
[PRS_E2E_USE_-00237]	
[PRS_E2E_USE_-00321]	
[PRS_E2E_USE_-00325]	
[PRS_E2E_USE_-00606]	

Table B.4: Deleted Traceables in R20-11

B.3 Constraint History R21-11

B.3.1 Added Constraints in R21-11

none

B.3.2 Changed Constraints in R21-11

none

B.3.3 Deleted Constraints in R21-11

Number	Heading
[constr_3176]	Value range of `windowSizeValid`
[constr_3177]	Dependency between `maxErrorStateValid`, `maxErrorStateInit` and `maxErrorStateInvalid`
[constr_3178]	Dependency between `minOkStateValid`, `minOkStateInit` and `minOkStateInvalid`

▽

△

Number	Heading
[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`

Table B.5: Deleted Constraints in R21-11

B.3.4 Added Traceables in R21-11

Number	Heading
[PRS_E2E_00507]	
[PRS_E2E_01107]	
[PRS_E2E_01154]	
[PRS_E2E_01155]	
[PRS_E2E_01156]	
[PRS_E2E_01157]	
[PRS_E2E_01159]	
[PRS_E2E_01160]	
[PRS_E2E_01161]	
[PRS_E2E_01162]	
[PRS_E2E_01163]	
[PRS_E2E_01164]	
[PRS_E2E_01165]	
[PRS_E2E_01166]	
[PRS_E2E_01167]	
[PRS_E2E_01169]	
[PRS_E2E_01170]	
[PRS_E2E_01171]	
[PRS_E2E_01172]	
[PRS_E2E_01173]	
[PRS_E2E_01174]	
[PRS_E2E_01175]	
[PRS_E2E_01176]	

▽

△

Number	Heading
[PRS_E2E_01177]	
[PRS_E2E_01178]	
[PRS_E2E_01179]	
[PRS_E2E_01180]	
[PRS_E2E_01181]	Draft
[PRS_E2E_01182]	Draft
[PRS_E2E_01183]	Draft
[PRS_E2E_01184]	Draft
[PRS_E2E_01185]	
[PRS_E2E_01186]	
[PRS_E2E_01187]	
[PRS_E2E_01188]	
[PRS_E2E_01189]	
[PRS_E2E_01190]	
[PRS_E2E_01191]	
[PRS_E2E_01192]	
[PRS_E2E_01193]	
[PRS_E2E_01194]	
[PRS_E2E_01195]	
[PRS_E2E_01196]	
[PRS_E2E_01197]	
[PRS_E2E_01198]	
[PRS_E2E_01199]	
[PRS_E2E_01200]	
[PRS_E2E_01201]	
[PRS_E2E_01202]	
[PRS_E2E_01203]	
[PRS_E2E_02355]	
[PRS_E2E_02356]	
[PRS_E2E_02357]	
[PRS_E2E_02358]	
[PRS_E2E_02359]	
[PRS_E2E_02360]	
[PRS_E2E_02361]	
[PRS_E2E_02362]	
[PRS_E2E_02363]	
[PRS_E2E_02364]	
[PRS_E2E_02365]	
[PRS_E2E_02366]	

▽

△

Number	Heading
[PRS_E2E_02367]	
[PRS_E2E_02368]	
[PRS_E2E_02369]	
[PRS_E2E_02376]	
[PRS_E2E_02615]	
[PRS_E2E_02616]	
[PRS_E2E_02617]	
[PRS_E2E_02618]	
[PRS_E2E_-CONSTR_03176]	Value range of `windowSizeValid`
[PRS_E2E_-CONSTR_03177]	Dependency between `maxErrorStateValid`
[PRS_E2E_-CONSTR_03178]	Dependency between `minOkStateValid`, `minOkStateInit` and `minOkStateInvalid`
[PRS_E2E_-CONSTR_03179]	Dependency between `minOkStateInit`, `maxErrorStateInit` and `WindowSizeInit`
[PRS_E2E_-CONSTR_03180]	Dependency between `minOkStateValid`, `maxErrorStateValid` and `windowSizeValid`
[PRS_E2E_-CONSTR_03181]	Dependency between `minOkStateInvalid`, `maxErrorStateInvalid` and `WindowSizeInvalid`
[PRS_E2E_-CONSTR_06300]	MaxDeltaCounter for Client-Server Communication
[PRS_E2E_-CONSTR_06301]	Dependency between `windowSizeInvalid` and `windowSizeValid`
[PRS_E2E_-CONSTR_06302]	Dependency between `windowSizeInit` and `windowSizeValid`
[PRS_E2E_UC_-01158]	
[PRS_E2E_UC_-01168]	

Table B.6: Added Traceables in R21-11

B.3.5 Changed Traceables in R21-11

Number	Heading
[PRS_E2E_00329]	
[PRS_E2E_00355]	
[PRS_E2E_00356]	

▽

△

Number	Heading
[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_00372]	
[PRS_E2E_00376]	
[PRS_E2E_00394]	
[PRS_E2E_00404]	
[PRS_E2E_00405]	
[PRS_E2E_00407]	
[PRS_E2E_00409]	
[PRS_E2E_00412]	
[PRS_E2E_00413]	
[PRS_E2E_00414]	
[PRS_E2E_00416]	
[PRS_E2E_00424]	
[PRS_E2E_00425]	
[PRS_E2E_00426]	
[PRS_E2E_00428]	
[PRS_E2E_00429]	
[PRS_E2E_00431]	
[PRS_E2E_00432]	
[PRS_E2E_00433]	
[PRS_E2E_00434]	
[PRS_E2E_00436]	
[PRS_E2E_00469]	
[PRS_E2E_00470]	
[PRS_E2E_00479]	
[PRS_E2E_00486]	
[PRS_E2E_00487]	

▽

△

Number	Heading
[PRS_E2E_00488]	
[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_00503]	
[PRS_E2E_00522]	
[PRS_E2E_00615]	
[PRS_E2E_00616]	
[PRS_E2E_00617]	
[PRS_E2E_00618]	
[PRS_E2E_00619]	
[PRS_E2E_00620]	
[PRS_E2E_00623]	
[PRS_E2E_00624]	
[PRS_E2E_00626]	
[PRS_E2E_00627]	
[PRS_E2E_00628]	
[PRS_E2E_00629]	
[PRS_E2E_00646]	
[PRS_E2E_00648]	
[PRS_E2E_00651]	
[PRS_E2E_00654]	
[PRS_E2E_00657]	
[PRS_E2E_00660]	
[PRS_E2E_00663]	
[PRS_E2E_00666]	
[PRS_E2E_00683]	
[PRS_E2E_00684]	
[PRS_E2E_00685]	

▽

△

Number	Heading
[PRS_E2E_00686]	
[PRS_E2E_00687]	
[PRS_E2E_00688]	
[PRS_E2E_00689]	
[PRS_E2E_00690]	
[PRS_E2E_00691]	
[PRS_E2E_00692]	
[PRS_E2E_00693]	
[PRS_E2E_00694]	
[PRS_E2E_00695]	
[PRS_E2E_00696]	
[PRS_E2E_00697]	
[PRS_E2E_00698]	
[PRS_E2E_00699]	
[PRS_E2E_00700]	
[PRS_E2E_00701]	
[PRS_E2E_00702]	
[PRS_E2E_00706]	
[PRS_E2E_00712]	
[PRS_E2E_00713]	
[PRS_E2E_00714]	
[PRS_E2E_00715]	
[PRS_E2E_00716]	
[PRS_E2E_00717]	
[PRS_E2E_00718]	
[PRS_E2E_00719]	
[PRS_E2E_00720]	
[PRS_E2E_00721]	
[PRS_E2E_00722]	
[PRS_E2E_00723]	
[PRS_E2E_00724]	
[PRS_E2E_00725]	
[PRS_E2E_00726]	
[PRS_E2E_00727]	
[PRS_E2E_00728]	
[PRS_E2E_00729]	
[PRS_E2E_00730]	
[PRS_E2E_00731]	
[PRS_E2E_00735]	

▽

△

Number	Heading
[PRS_E2E_00736]	
[PRS_E2E_00824]	
[PRS_E2E_00831]	
[PRS_E2E_00836]	
[PRS_E2E_00840]	
[PRS_E2E_UC_-00055]	
[PRS_E2E_UC_-00071]	
[PRS_E2E_UC_-00072]	
[PRS_E2E_UC_-00073]	
[PRS_E2E_UC_-00170]	
[PRS_E2E_UC_-00171]	
[PRS_E2E_UC_-00173]	
[PRS_E2E_UC_-00235]	
[PRS_E2E_UC_-00236]	
[PRS_E2E_UC_-00238]	
[PRS_E2E_UC_-00239]	
[PRS_E2E_UC_-00308]	
[PRS_E2E_UC_-00316]	
[PRS_E2E_UC_-00317]	
[PRS_E2E_UC_-00320]	
[PRS_E2E_UC_-00321]	
[PRS_E2E_UC_-00325]	
[PRS_E2E_UC_-00327]	
[PRS_E2E_UC_-00328]	

▽

△

Number	Heading
[PRS_E2E_UC_-00351]	
[PRS_E2E_UC_-00463]	
[PRS_E2E_UC_-00464]	
[PRS_E2E_UC_-00606]	
[PRS_E2E_UC_-00743]	
[PRS_E2E_UC_-00754]	
[PRS_E2E_UC_-00786]	
[PRS_E2E_UC_-00797]	
[PRS_E2E_USE_-00741]	

Table B.7: Changed Traceables in R21-11

B.3.6 Deleted Traceables in R21-11

Number	Heading
[PRS_E2E_0507]	

Table B.8: Deleted Traceables in R21-11