

BS ISO 16845-1:2016



BSI Standards Publication

Road vehicles — Controller area network (CAN) conformance test plan

Part 1: Data link layer and physical
signalling

bsi.

AVA Documents
Document Sharing Hub

National foreword

This British Standard is the UK implementation of ISO 16845-1:2016.

The UK participation in its preparation was entrusted to Technical Committee AUE/16, Data Communication (Road Vehicles).

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2016.
Published by BSI Standards Limited 2016

ISBN 978 0 580 88009 4

ICS 43.040.15

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 November 2016.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

Road vehicles — Controller area network (CAN) conformance test plan —

Part 1: Data link layer and physical signalling

Véhicules routiers — Plan d'essai de conformité du gestionnaire de réseau de communication (CAN) —

Partie 1: Couche liaison de données et signalisation physique





COPYRIGHT PROTECTED DOCUMENT

© ISO 2016, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword	vii
Introduction	viii
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 Abbreviated terms	3
5 Global overview	4
5.1 Scope of test plan	4
5.2 Architecture of test plan	4
5.3 Organization	5
5.3.1 General organization	5
5.3.2 Test case organization	6
5.3.3 Hierarchical structure of tests	7
6 LT parameters	8
6.1 Overview	8
6.2 Description of parameters	8
6.2.1 Communication parameters	8
6.2.2 Application parameters	9
6.2.3 Bit rate configuration parameter variation for bit timing tests	10
7 Test type 1, received frame	10
7.1 Test class 1, valid frame format	10
7.1.1 Identifier and number of data test in base format	10
7.1.2 Identifier and number of data test in extended format	11
7.1.3 Reception after arbitration lost	12
7.1.4 Acceptance of non-nominal bit in base format frame	13
7.1.5 Acceptance of non-nominal bit in extended format frame	13
7.1.6 Protocol exception behaviour on non-nominal bit	14
7.1.7 Minimum time for bus idle after protocol exception handling	15
7.1.8 DLC greater than 8	15
7.1.9 Absent bus idle — Valid frame reception	16
7.1.10 Stuff acceptance test in base format frame	16
7.1.11 Stuff acceptance test in extended format frame	17
7.1.12 Message validation	18
7.2 Test class 2, error detection	19
7.2.1 Bit error in data frame	19
7.2.2 Stuff error for basic frame	19
7.2.3 Stuff error for extended frame	20
7.2.4 Stuff error for FD frame payload bytes	21
7.2.5 CRC error	22
7.2.6 Combination of CRC error and form error	23
7.2.7 Form error in data frame at “CRC delimiter” bit position	24
7.2.8 Form error at fixed stuff bit in FD frames	24
7.2.9 Form error in data frame at “ACK delimiter” bit position	25
7.2.10 Form error in data frame at “EOF”	25
7.2.11 Message non-validation	26
7.3 Test class 3, error frame management	26
7.3.1 Error flag longer than 6 bits	26
7.3.2 Data frame starting on the third bit of intermission field	27
7.3.3 Bit error in error flag	27
7.3.4 Form error in error delimiter	28
7.4 Test class 4, overload frame management	28
7.4.1 MAC overload generation during intermission field	28

7.4.2	Last bit of EOF	29
7.4.3	Eighth bit of an error and overload delimiter	29
7.4.4	Bit error in overload flag	30
7.4.5	Form error in overload delimiter	30
7.4.6	MAC overload generation during intermission field following an error frame	31
7.4.7	MAC overload generation during intermission field following an overload frame	31
7.5	Test class 5, passive error state class	32
7.5.1	Passive error flag completion test 1	32
7.5.2	Data frame acceptance after passive error frame transmission	33
7.5.3	Acceptance of 7 consecutive dominant bits after passive error flag	33
7.5.4	Passive state unchanged on further errors	34
7.5.5	Passive error flag completion — Test case 2	34
7.5.6	Form error in passive error delimiter	35
7.5.7	Transition from active to passive ERROR FLAG	35
7.6	Test class 6, error counter management	36
7.6.1	REC increment on bit error in active error flag	36
7.6.2	REC increment on bit error in overload flag	37
7.6.3	REC increment when active error flag is longer than 13 bits	37
7.6.4	REC increment when overload flag is longer than 13 bits	38
7.6.5	REC increment on bit error in the ACK field	38
7.6.6	REC increment on form error in CRC delimiter	38
7.6.7	REC increment on form error in ACK delimiter	39
7.6.8	REC increment on form error in EOF field	39
7.6.9	REC increment on stuff error	40
7.6.10	REC increment on CRC error	41
7.6.11	REC increment on dominant bit after end of error flag	41
7.6.12	REC increment on form error in error delimiter	42
7.6.13	REC increment on form error in overload delimiter	42
7.6.14	REC decrement on valid frame reception	43
7.6.15	REC decrement on valid frame reception during passive state	43
7.6.16	REC non-increment on last bit of EOF field	44
7.6.17	REC non-increment on 13-bit length overload flag	44
7.6.18	REC non-increment on 13-bit length error flag	45
7.6.19	REC non-increment on last bit of error delimiter	45
7.6.20	REC non-increment on last bit of overload delimiter	46
7.6.21	REC non-decrement on transmission	46
7.6.22	REC increment on form error at fixed stuff bit in FD frames	47
7.6.23	REC non-increment on protocol exception in FD frames	47
7.7	Test class 7, bit timing Classical CAN frame format	48
7.7.1	Sample point test	48
7.7.2	Hard synchronization on SOF reception	49
7.7.3	Synchronization when $e > 0$ and $e \leq \text{SJW}(N)$	49
7.7.4	Synchronization when $e > 0$ and $e > \text{SJW}(N)$	50
7.7.5	Synchronization when $e < 0$ and $ e \leq \text{SJW}(N)$	50
7.7.6	Synchronization when $e < 0$ and $ e > \text{SJW}(N)$	51
7.7.7	Glitch filtering test on positive phase error	51
7.7.8	Glitch filtering test on negative phase error	52
7.7.9	Glitch filtering test in idle state	53
7.7.10	Non-Synchronization after a dominant sampled bit	54
7.7.11	Synchronization when $e < 0$ and $ e \leq \text{SJW}(N)$ at “ACK” bit position	55
7.8	Test class 8, bit timing CAN FD frame format	55
7.8.1	Sample point test	55
7.8.2	Hard synchronization on “res” bit	58
7.8.3	Synchronization when $e > 0$ and $e \leq \text{SJW}(D)$	59
7.8.4	Synchronization when $e > 0$ and $e > \text{SJW}(D)$	61
7.8.5	Synchronization when $e < 0$ and $ e \leq \text{SJW}$	63
7.8.6	Synchronization when $e < 0$ and $ e > \text{SJW}$	65

7.8.7	Glitch filtering test on positive phase error.....	67
7.8.8	Glitch filtering test on negative phase error.....	69
7.8.9	No synchronization after a dominant sampled bit.....	71
8	Test type 2, transmitted frame.....	73
8.1	Test class 1, valid frame format.....	73
8.1.1	Identifier and number of data bytes test in base format.....	73
8.1.2	Identifier and number of data bytes test in extended format.....	73
8.1.3	Arbitration in base format frame.....	74
8.1.4	Arbitration in extended format frame test.....	75
8.1.5	Message validation.....	76
8.1.6	Stuff bit generation capability in base format frame.....	76
8.1.7	Stuff bit generation capability in extended frame.....	77
8.1.8	Transmission on the third bit of intermission field after arbitration lost.....	78
8.2	Test class 2, error detection.....	79
8.2.1	Bit error test in base format frame.....	79
8.2.2	Bit error in extended format frame.....	80
8.2.3	Stuff error test in base format frame.....	81
8.2.4	Stuff error test in extended frame format.....	81
8.2.5	Form error test.....	82
8.2.6	Acknowledgement error.....	83
8.2.7	Form field tolerance test for FD frame format.....	84
8.2.8	Bit error at stuff bit position for FD frame payload bytes.....	84
8.3	Test class 3, error frame management.....	85
8.3.1	Error flag longer than 6 bit.....	85
8.3.2	Transmission on the third bit of intermission field after error frame.....	85
8.3.3	Bit error in error flag.....	86
8.3.4	Form error in error delimiter.....	86
8.4	Test class 4, overload frame management.....	87
8.4.1	MAC overload generation in intermission field.....	87
8.4.2	Eighth bit of an error and overload delimiter.....	88
8.4.3	Transmission on the third bit of intermission after overload frame.....	88
8.4.4	Bit error in overload flag.....	89
8.4.5	Form error in overload delimiter.....	89
8.5	Test class 5, passive error state and bus-off.....	90
8.5.1	Acceptance of active error flag overwriting passive error flag.....	90
8.5.2	Frame acceptance after passive error frame transmission.....	90
8.5.3	Acceptance of 7 consecutive dominant bits after passive error flag.....	91
8.5.4	Reception of a frame during suspend transmission.....	92
8.5.5	Transmission of a frame after suspend transmission — Test case 1.....	92
8.5.6	Transmission of a frame after suspend transmission — Test case 2.....	93
8.5.7	Transmission of a frame after suspend transmission — Test case 3.....	93
8.5.8	Transmission of a frame without suspend transmission.....	93
8.5.9	No transmission of a frame between the third bit of intermission field and eighth bit of suspend transmission.....	94
8.5.10	Bus-off state.....	94
8.5.11	Bus-off recovery.....	95
8.5.12	Completion condition for a passive error flag.....	96
8.5.13	Form error in passive error delimiter.....	96
8.5.14	Maximum recovery time after a corrupted frame.....	97
8.5.15	Transition from active to passive ERROR FLAG.....	97
8.6	Test class 6, error counter management.....	98
8.6.1	TEC increment on bit error during active error flag.....	98
8.6.2	TEC increment on bit error during overload flag.....	99
8.6.3	TEC increment when active error flag is followed by dominant bits.....	99
8.6.4	TEC increment when passive error flag is followed by dominant bits.....	100
8.6.5	TEC increment when overload flag is followed by dominant bits.....	100
8.6.6	TEC increment on bit error in data frame.....	101
8.6.7	TEC increment on form error in a frame.....	101

8.6.8	TEC increment on acknowledgement error	102
8.6.9	TEC increment on form error in error delimiter	103
8.6.10	TEC increment on form error in overload delimiter	103
8.6.11	TEC decrement on successful frame transmission for $TEC < 128$	104
8.6.12	TEC decrement on successful frame transmission for $TEC > 127$	104
8.6.13	TEC non-increment on 13-bit long overload flag	105
8.6.14	TEC non-increment on 13-bit long error flag	105
8.6.15	TEC non-increment on form error at last bit of overload delimiter	106
8.6.16	TEC non-increment on form error at last bit of error delimiter	106
8.6.17	TEC non-increment on acknowledgement error in passive state	107
8.6.18	TEC increment after acknowledgement error in passive state	107
8.6.19	TEC non-increment on stuff error during arbitration	108
8.6.20	TEC non-decrement on transmission while arbitration lost	108
8.6.21	TEC non-increment after arbitration lost and error	109
8.7	Test class 7, bit timing	109
8.7.1	Sample point test	109
8.7.2	Hard synchronization on SOF reception before sample point	110
8.7.3	Hard synchronization on SOF reception after sample point	111
8.7.4	Synchronization when $e < 0$ and $ e \leq SJW(N)$	111
8.7.5	Synchronization for $e < 0$ and $ e > SJW(N)$	112
8.7.6	Glitch filtering test on negative phase error	113
8.7.7	Non-synchronization on dominant bit transmission	113
8.7.8	Synchronization before information processing time	114
8.7.9	Synchronization after sample point while sending a dominant bit	114
8.8	Test class 8, bit timing CAN FD frame format	115
8.8.1	Sample point test	115
8.8.2	Secondary sample point test	118
8.8.3	No synchronization within data phase bits when $e < 0$; $ e \leq SJW(D)$	121
8.8.4	Glitch filtering test on negative phase error within FD frame bits	123
8.8.5	No synchronization on dominant bit transmission in FD frames	124
9	Test type 3, bi-directional frame	125
9.1	Test class 1, valid frame format	125
9.2	Test class 2, error detection	125
9.3	Test class 3, active error frame management	125
9.4	Test class 4, overload frame management	125
9.5	Test class 5, passive-error state and bus-off	125
9.6	Test class 6, error counter management	126
9.6.1	REC unaffected when increasing TEC	126
9.6.2	TEC unaffected when increasing REC	126

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/TC 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

This first edition of ISO 16845-1 cancels and replaces ISO 16845:2004, which has been technically revised.

A list of all parts in the ISO 16845 series can be found on the ISO website.

Introduction

ISO 16845 was first published in 2004 to provide the methodology and abstract test suite necessary for checking the conformance of any CAN implementation of the CAN specified in ISO 11898-1.

Road vehicles — Controller area network (CAN) conformance test plan —

Part 1: Data link layer and physical signalling

1 Scope

This document specifies the conformance test plan for the CAN data link layer and the physical signalling as standardized in ISO 11898-1. This includes the Classical CAN protocols as well as the CAN FD protocols.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 11898-1:2015, *Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling*

ISO/IEC 9646-1, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 1: General concepts*

ISO/IEC 9646-2, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 2: Abstract Test Suite specification*

ISO/IEC 9646-4, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 4: Test realization*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 11898-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <https://www.iso.org/obp/>

3.1

bit rate prescaler

BRP

minimum time quantum used for a TQ in CAN Bit time configuration

3.2

conformance testing

applying the *test plan* (3.17) to an IUT

3.3

default state

state of the IUT

Note 1 to entry: The default state is characterized by the default value presented in 5.3.2.5.

3.4
dominant
represents the logical 0

3.5
dominant state
CAN bus is in dominant state when at least one CAN node drives a dominant value on the line

3.6
elementary test
repetitions of the test case for several values of the parameter to test

3.7
end of frame
last field of a data or remote frame before the intermission field

3.8
idle state
CAN bus is in idle state when no frame is started after intermission field

3.9
lower tester
supervises the *test suite* ([3.18](#))

3.10
REC passive state
device is in the passive state because the value of the REC has reached the error passive limit

3.11
recessive
represents the logical 1

3.12
recessive state
CAN bus is in the recessive state when no CAN node drives a dominant value on the line

3.13
TEC passive state
device is in the passive state because the value of the TEC has reached the error passive limit

3.14
test case
each test case is defined by a specific number and a particular name in the *test suite* ([3.18](#))

3.15
test class
each *test type* ([3.19](#)) is divided in eight test classes

3.16
test frame
CAN frames containing the test pattern specified in this document

3.17
test plan
specific application of the «OSI conformance testing general concepts» standard

3.18
test suite
checks the behaviour of the IUT for particular parameters of ISO 11898-1

3.19

test type

defines the direction of the *test frames* (3.16)

EXAMPLE Behaviour of the IUT if receiving and/or transmitting messages.

3.20

upper tester

acts as a user of the IUT

4 Abbreviated terms

All abbreviated terms in this document are written in upper case letters.

CTRL	Control field of CAN frame + SRR/RTR + IDE bit
CBFF: CTRL	= RTR, IDE, FDF, DLC merged together as 7 bit hexadecimal value
FBFF: CTRL	= RRS, IDE, FDF, res, BRS, ESI, DLC merged together as 10 bit hexadecimal value
CEFF: CTRL	= SRR, IDE, RTR, FDF, r0, DLC merged together as 9 bit hexadecimal value
FEFF: CTRL	= SRR, IDE, RRS, FDF, res, BRS, ESI, DLC merged together as 11 bit hexadecimal value
IPT	information processing time
LT	lower tester
NTQ(D)	number of time quantum in data bit rate
NTQ(N)	number of time quantum in nominal bit rate
Phase_Seg1(D)	Phase Segment 1 (Phase_Seg1) for data phase bit rate
Phase_Seg1(N)	Phase Segment 1 (Phase_Seg1) for nominal bit rate
Phase_Seg2(D)	Phase Segment 2 (Phase_Seg2) for data phase bit rate
Phase_Seg2(N)	Phase Segment 2 (Phase_Seg2) for nominal bit rate
Prop_Seg(D)	propagation segment (Prop_Seg) for data phase bit rate
Prop_Seg(N)	propagation segment (Prop_Seg) for nominal bit rate
Sampling_Point(D)	Sync_Seg(D) + Prop_Seg(D) + Phase_Seg1(D)
Sampling_Point(N)	Sync_Seg(N) + Prop_Seg(N) + Phase_Seg1(N)
SJW(D)	synchronization jump width (SJW) for data phase bit rate
SJW(N)	synchronization jump width (SJW) for nominal bit rate
Sync_Seg(D)	synchronization segment (Sync_Seg) for data phase bit rate
Sync_Seg(N)	synchronization segment (Sync_Seg) for nominal bit rate
TP	test plan

TQ(D)	time quantum in data bit rate
TQ(N)	time quantum in nominal bit rate
UT	upper tester

5 Global overview

5.1 Scope of test plan

ISO 9646-1, ISO 9646-2 and ISO 9646-4 define the methodology and the abstract test suite necessary to check the conformance of any CAN implementation to ISO 11898-1. The architecture of the TP is as shown in [Figure 1](#).

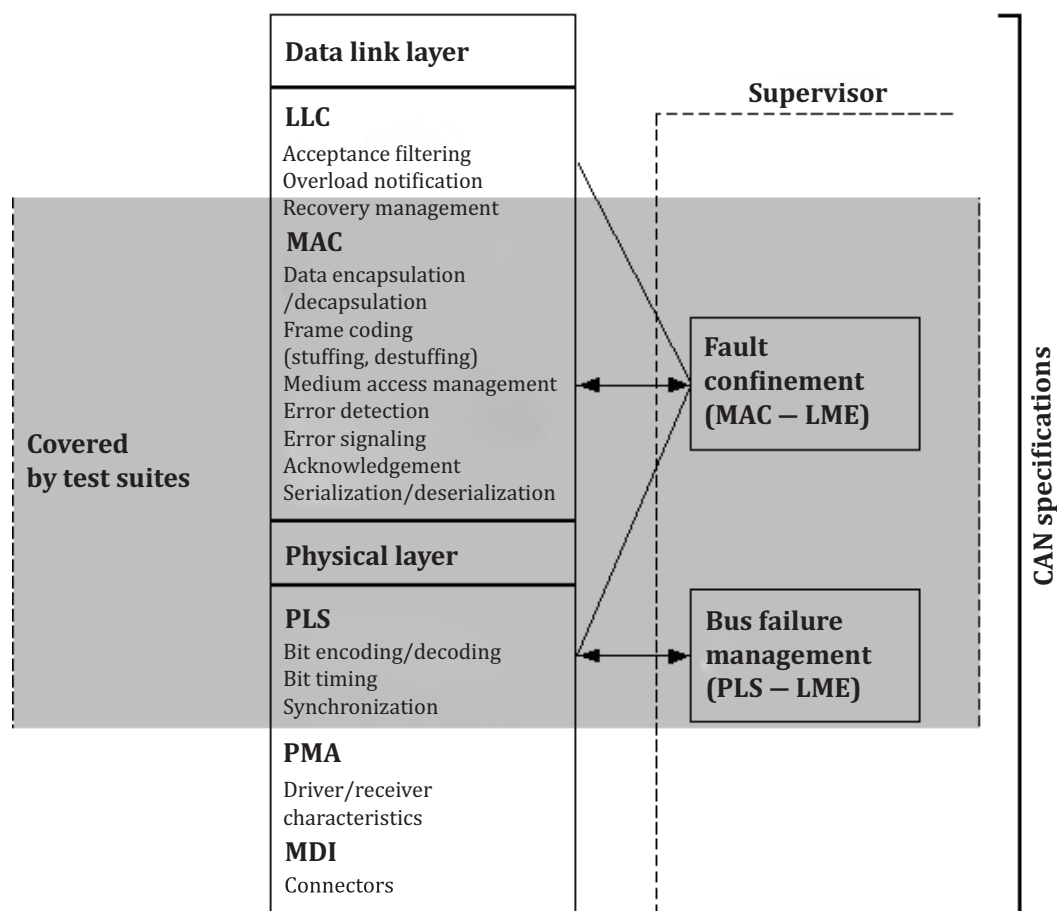


Figure 1 — Architecture of the test plan

5.2 Architecture of test plan

This methodology and the associated abstract test suites will be hereafter referred to as test plan (TP).

The TP is a specific application of the «OSI conformance testing general concepts» ISO 9646-1 and is restricted to the single party testing mode. Since the upper service boundary of a CAN implementation is not standardized and in some cases may not be observed and controlled [due to an application specific behaviour embedded in this implementation, for example, CAN SLIO (serial linked input/output)], the TP will rely either on the «coordinated test method» or the «remote test method».

Depending on the test method applied, the TP will involve up to the following three test functions:

- a lower tester (LT) operating in a way similar to the CAN implementation to be tested (IUT), running test suite and granting test verdict;
- an upper tester (UT) acting as user of the IUT (IUT dependant);
- a test management protocol between the IUT and the LT. The protocol consists in test coordination procedures.

The last two functions are only applicable to the coordinated test procedure.

During test execution, the LT can observe and control the standardized lower service boundary of the IUT (PCO) through the two service primitives provided by the CAN physical signalling sub-layer: PLS-Data.indicate and PLS-Data.request in most cases.

The environment that implements the TP is described in [Figure 2](#).

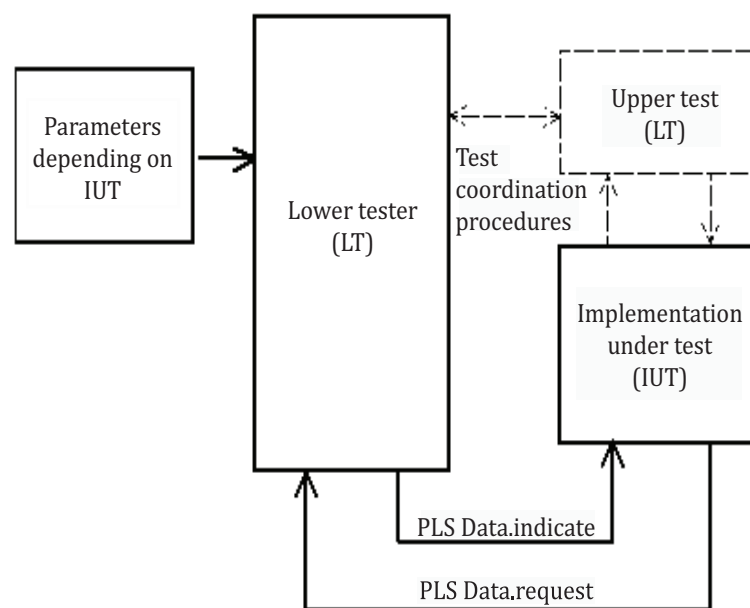


Figure 2 — CAN conformance TP environment

Using the network interface, the LT indicates to the UT the actions to be performed and the UT provides the LT with information concerning the internal behaviour of the IUT.

In order to allow the LT and the UT to communicate, it is necessary to define some test coordination procedures between them. These procedures use the network to the exclusion of any other physical link. They are used to set up the UT and to verify the test results.

5.3 Organization

5.3.1 General organization

The LT verifies if the IUT complies with the MAC, LLC and PLS sub-layers of ISO 11898-1. The LT points out differences between what is expected from the standard and the actual behaviour of the IUT.

The abstract test suites of the TP are independent to one another. Each abstract test suite checks the behaviour of the IUT for a particular parameter of ISO 11898-1. Each test case may be executed one after another in any order or alone.

Test cases requiring variations of individual parameters (identifier, number of data, etc.) should be repeated for each value of the parameter. Each repetition is named elementary test. A test case including different elementary tests is valid only if all tests pass.

5.3.2 Test case organization

5.3.2.1 General

Each elementary test is made of the following three states:

- set-up state;
- test state;
- verification state.

At the PCO, these states involve exchanges of valid sequences of PLS service primitives [CAN frame(s)] or invalid sequences of PLS primitives (invalid CAN frames or noise).

Before the first elementary test is started, the IUT has to be initialized into the default state.

5.3.2.2 Set-up state

The set-up state is the state in which the IUT has to be before entering the test state.

5.3.2.3 Test state

This is the part of the elementary test in which the parameter or protocol feature is actually checked. This state needs one or several exchanges or frames. These frames are named test frames.

5.3.2.4 Verification state

Verification state is made of the data reading frames which verify that the data have been handled in accordance with ISO 11898-1. These data should be checked.

For tests belonging to classes 1 to 6, the LT should be able to detect the correct value of the bit. For bit timing tests (classes 7 and 8), the LT should be able to detect a faulty synchronization of one time quantum.

For tests belonging to class 6, a readable error counter should be used for verification. In case the error counter value is not readable, the test should be applied by driving the IUT to the next error state by additional bus errors. The state change at expected position indicates the correct error counting up to this state change.

5.3.2.5 Default state

The default state is characterized by the following default value:

- both REC and TEC should be equal to 0;
- no pending transmission should be present;
- IUT should be in idle state;
- PLS-Data indicate and PLS-Data request should be recessive.

After the end of each elementary test, the default state should be applied.

5.3.3 Hierarchical structure of tests

5.3.3.1 Overview

All the tests defined in the test plan are grouped into categories in order to aid planning, development, understanding or execution of each test case. There are three levels of categories:

- test types;
- test classes;
- test cases.

5.3.3.2 Test types

The types define the direction of the frames. The three types are as follows.

- **Type 1: Received frame type**

It includes all the tests evaluating the behaviour of the IUT for data frames and remote frames received by the IUT.

- **Type 2: Transmitted frame type**

It includes all the tests evaluating the behaviour of the IUT for data frames and remote frames transmitted by the IUT.

- **Type 3: Bi-directional frame type**

It includes all the tests with data frames or remote frames both received and transmitted by the IUT.

5.3.3.3 Test classes

Each of the three test types defined in [5.3.3.2](#) is divided in eight classes grouping tests by topic regarding to ISO 11898-1. These eight classes are as follows.

- **Class 1: Valid frame format class**

This class includes the tests involving only error free data or remote frames.

- **Class 2: Error detection class**

This class includes the tests corrupting data or remote frames. These tests check the correct error detection by the IUT.

- **Class 3: Active error frame management class**

This class includes the tests verifying the IUT correct management of error-free and of corrupted active error frames.

- **Class 4: Overload frame management class**

This class includes the tests verifying the IUT correct management of error free and of corrupted overload frames.

- **Class 5: Passive error state and bus-off class**

This class includes the tests verifying the IUT behaviour during passive error state and bus-off state.

- **Class 6: Error counters management class**

This class includes the tests verifying the correct management of the TEC and REC by the IUT in both active and passive error state.

— **Class 7: Bit timing Classical CAN frame format**

This class includes the tests verifying the correct management of bit timing by the IUT. This class should only be applied to components performing only recessive to dominant edge synchronization (if the recessive to dominant edge synchronization exists, it should be disabled).

— **Class 8: Bit timing CAN FD frame format**

This class includes the tests verifying the correct management of bit timing by the IUT. This class should only be applied to components performing only recessive to dominant edge synchronization (if the recessive to dominant edge synchronization exists, it should be disabled).

5.3.3.4 Test cases

Any basic entry of the test list is intended to check a particular parameter of ISO 11898-1 in the IUT.

Each test case is defined by a specific number and a particular name in order to differentiate the test cases and to easily summarize the goal of the test case. Some test cases may be subdivided into elementary tests which are repetitions of the test case for several values of the parameter to test.

6 LT parameters

6.1 Overview

ISO 11898-1 allows several IUT implementations. Consequently, the user should provide the LT with parameters in order to indicate which kind of IUT is going to be tested.

These parameters can be classified in two categories.

— **Communication parameters**

This category defines which tests can be executed for the IUT, and which test method will be applied.

— **Application parameters**

This category defines the features of the frames used for each test case selected according to the previous parameters.

6.2 Description of parameters

6.2.1 Communication parameters

6.2.1.1 Categories of communication parameter

These parameters are subdivided in three categories:

- implementation parameters;
- timing parameters;
- other parameters.

In this document, they are always written in upper case letters.

6.2.1.2 Implementation classes

Some parameters depending on the IUT should be specified by the user in order to allow the LT to match the IUT. These parameters are as follows.

a) CAN_VERSION

This parameter indicates the version implemented in the IUT. It can take three values.

- Classical CAN

A Classical CAN node that cannot receive or transmit frames in CAN FD frame format and it disturbs them by error treatment.

- CAN FD tolerant

A Classical CAN compatible node that cannot receive or transmit frames in CAN FD frame format, but does not disturb them.

- CAN FD enabled

A CAN node that is able to receive or transmit in CAN FD frame format as well as in Classical CAN frame format.

b) OPEN/SPECIFIC

This parameter indicates if the IUT is open regarding to the application layers, or if it includes a specific application. It can take two values.

- OPEN

Open IUT allowing the test coordination procedure to be implemented in an UT. These IUT are tested with the «coordinated test method» of ISO/IEC 9646-1.

- SPECIFIC

IUT that can be tested only with the help of a specific configuration procedure. These IUT are tested with the «remote test method» of ISO/IEC 9646-1.

6.2.1.3 Timing parameters

The LT also needs some timing parameters to be in accordance with the IUT and the UT characteristics. These parameters are as follows.

a) TIMEOUT

This parameter indicates the minimum duration time for which the LT should wait in order to respect the three following conditions:

- the UT should have enough time to put the IUT into the set-up state;
- the IUT should have enough time to transmit a response frame after a remote frame;
- the LT should consider an optional additional waiting time after the end of the minimum bus-off recovery sequence before the IUT enters error-active state again.

b) Bit timing parameter as defined in ISO 11898-1.

6.2.2 Application parameters

Except the test cases for which a particular profile of data is defined by the TP, the ID, DLC and content of the data used during the test cases should be chosen by the user and is not relevant for test purpose.

CAN FD format frames are configured with BRS = 1 and ESI = 0 if not defined by the TP.

6.2.3 Bit rate configuration parameter variation for bit timing tests

The configuration range for the number of time quanta and the resultant sampling point is IUT-specific. A variation of bit rate configuration setups used for bit timing tests shall be done according the IUT's individual configuration range.

For Classical CAN configurations:

- at least one configuration for minimum programmable number of TQ(N)s (ISO 11898-1:2015, Table 8);
- at least one configuration for maximum programmable number of TQ(N)s;
- at least one configuration for a programmable number of TQ(N)s in the middle between MIN and MAX number of TQ(N)s.

For FD enabled configuration:

- test with shared prescaler (different number of TQs for nominal and data bit rate) and if supported, test with separate prescalers (constant number of TQs for nominal and data bit rate and different TQ time);
- at least one configuration for minimum possible number of TQ(D)s;
- at least one configuration for maximum possible number of TQ(D)s;
- at least one configuration for a possible number of TQ(D)s in the middle of MIN and MAX.

If enabled, the transmitter delay compensation could be implemented with the following two options for configuring the delay compensation time in TQ(D)s:

- the IUT may use a preconfigured fix value to describe the SSP position;
- the IUT can do an internal-edge to external-edge measurement to determine the real delay plus a preconfigured fix offset to describe the SSP position relative to the delay measurement.

There shall be one elementary test to perform for each possible way of configuration of delay compensation (fix programmed or automatically measured).

“Delay compensation” + “programmed offset” shall lead to a value similar like “applied delay” + Sync_Seg(D) + Prop_Seg(D) + Phase_Seg1(D), so that the SSP evaluate the delayed bit at the same position inside the delayed bit as the programmed sampling point describe it for a non-delayed bit.

For test cases using TDC, the bit timings need to be selected so that $[\text{Sync_Seg}(N) + \text{Prop_Seg}(N) + \text{Phase_Seg1}(N)] > 2$ data bit times.

7 Test type 1, received frame

7.1 Test class 1, valid frame format

7.1.1 Identifier and number of data test in base format

[Table 1](#) specifies the test of the correct reception of frames in base frame format.

Table 1 — Identifier and number of data test in base format

Item	Description	
Purpose	This test verifies the behaviour of the IUT when receiving a correct data frame with different identifiers and different numbers of data bytes in base format frame.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	ID DLC FDF = 0	ID DLC FDF = 1
Elementary test cases	The CAN ID shall be element of: $\in [000_h, 7FF_h]$ Different CAN IDs are used for test. #1 CAN ID = 555 _h #2 CAN ID = 2AA _h #3 CAN ID = 000 _h #4 CAN ID = 7FF _h #5 CAN ID = a random value Tested number of data bytes: $\in [0, 8]$ Number of tests: 45	The CAN ID will be element of: $\in [000_h, 7FF_h]$ Different CAN IDs are used for test. #1 CAN ID = 555 _h #2 CAN ID = 2AA _h #3 CAN ID = 000 _h #4 CAN ID = 7FF _h #5 CAN ID = a random value Tested number of data bytes: $\in [0, 8] \cup [12] \cup [16] \cup [20] \cup [24] \cup [32] \cup [48] \cup [64]$ Number of tests: 80
Set-up	The IUT is left in the default state.	
Execution	The test system sends a frame with ID and DLC as specified in elementary test cases definition.	
Response	The IUT shall not generate any error flag during the test. The IUT shall acknowledge the test frame. The data received by the IUT during the test state should match the data sent in the test frame.	
NOTE	An implementation with limited payload capabilities will be tested in range of their payload capabilities.	

7.1.2 Identifier and number of data test in extended format

[Table 2](#) specifies the test of the correct reception of frames in extended frame format.

Table 2 — Identifier and number of data test in extended format

Item	Description	
Purpose	This test verifies the behaviour of the IUT when receiving a correct data frame with different identifiers and different numbers of data bytes in extended format frame.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	ID DLC FDF = 0	ID DLC FDF = 1

Table 2 (continued)

Item	Description	
Elementary test cases	The CAN ID shall be element of: $\in [00000000_h, 1FFFFFFF_h]$ Different CAN IDs are used for test. #1 CAN ID = 15555555_h #2 CAN ID = 0AAAAAAA_h #3 CAN ID = 00000000_h #4 CAN ID = 1FFFFFFF_h #5 CAN ID = random value Tested number of data bytes: $\in [0, 8]$ Number of tests: 45	The CAN ID shall be element of: $\in [00000000_h, 1FFFFFFF_h]$ Different CAN IDs are used for test. #1 CAN ID = 15555555_h #2 CAN ID = 0AAAAAAA_h #3 CAN ID = 00000000_h #4 CAN ID = 1FFFFFFF_h #5 CAN ID = random value Tested number of data bytes: $\in [0, 8] \cup [12] \cup [16] \cup [20] \cup [24] \cup [32] \cup [48] \cup [64]$ Number of tests: 80
Set-up	The IUT is left in the default state.	
Execution	The test system sends a frame with ID and DLC as specified in elementary test cases definition.	
Response	The IUT shall not generate any error flag during the test. The IUT shall acknowledge the test frame. The data received by the IUT during the test state shall match the data sent in the test frame.	
NOTE	An implementation with limited ID range may not be able to receive the frame. An implementation with limited payload capabilities will be tested in range of their payload capabilities.	

7.1.3 Reception after arbitration lost

[Table 3](#) specifies the test of the reception of arbitration winning frame, while the IUT loses the arbitration.

Table 3 — Reception after arbitration lost

Item	Description	
Purpose	This test verifies the capability of the IUT to manage the reception of arbitration winning frame, while the IUT loses the arbitration.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	ID all bit = 1 IDE SRR (in case of IDE = 1) FDF = 0 DLC = 0 RTR = 1	ID all bit = 1 IDE SRR (in case of IDE = 1) FDF DLC = 0 RTR = 1

Table 3 (continued)

Item	Description							
Elementary test cases		LT frame format	IUT frame format	Bit for arb. lost		LT frame format	IUT frame format	Bit for arb. lost
	#1	CBFF	CBFF	RTR				
	#2	CBFF	CEFF	SRR	#1	CBFF	FBFF	LSB base ID
	#3	CBFF	CEFF	IDE	#2	FBFF	CBFF	RTR
	#4	CEFF	CBFF	LSB base ID	#3	CEFF	FEFF	LSB ID ext.
	#5	CEFF	CEFF	LSB ID ext.	#4	FEFF	CEFF	RTR
	#6	CEFF	CEFF	RTR				
Set-up	The IUT is left in the default state.							
Execution	The LT causes the IUT to transmit a frame according to "IUT frame format" in elementary test cases. Then, the LT forces the bit described at "bit for arbitration lost" in elementary test cases to dominant state and continues to send a valid frame according to elementary test cases.							
Response	The IUT shall become the receiver when sampling the dominant bit sent by the LT. The frame received by the IUT shall match the frame sent by the LT. As soon as the bus is idle again, the IUT shall restart the transmission of the frame. The IUT shall not generate any error flag during the test. The content of the frame shall match the LT request.							
NOTE	An implementation with limited ID range may not be able to transmit/receive the frame.							

7.1.4 Acceptance of non-nominal bit in base format frame

Table 4 specifies the test of an IUT that accepts the non-nominal bit value in a valid base format frame.

Table 4 — Acceptance of non-nominal in base format frame

Item	Description			
Purpose	The purpose of this test is to verify that the IUT accepts the non-nominal value of bit described in test variables in a valid base format frame.			
CAN_VERSION	Classical CAN		CAN FD enabled	
Test variables	FDF		RRS FDF = 1	
Elementary test cases	Test #1	FDF 1	Test #1	RRS 1
Set-up	The IUT is left in the default state.			
Execution	A single test frame is used for the elementary test cases.			
Response	The IUT shall not generate any error flag in this test frame. The IUT shall acknowledge the test frame. The data received by the IUT during the test state shall match the data sent in the test frame.			

7.1.5 Acceptance of non-nominal bit in extended format frame

Table 5 specifies the test of an IUT that accepts the non-nominal bit value in a valid extended frame.

Table 5 — Acceptance of non-nominal in extended format frame

Item	Description									
Purpose	The purpose of this test is to verify that the IUT accepts the non-nominal value of bit described in test variables in a valid extended format frame.									
CAN_VERSION	Classical CAN				CAN FD Tolerant			CAN FD enabled		
Test variables	SRR FDF r0				SRR r0 FDF = 0			SRR RRS FDF = 1		
Elementary test cases	TEST	SRR	r0	FDF	TEST	SRR	r0	TEST	SRR	RRS
	#1	1	1	1	#1	1	1	#1	1	1
	#2	1	1	0	#2	0	1	#2	0	1
	#3	1	0	1	#3	0	0	#3	0	0
	#4	0	1	1						
	#5	0	1	0						
	#6	0	0	1						
	#7	0	0	0						
Set-up	The IUT is left in the default state.									
Execution	A single test frame is used for each of the elementary tests.									
Response	The IUT shall not generate any error flag during the test.									
	The IUT shall acknowledge the test frame. The data received by the IUT during the test state shall match the data sent in the test frame.									
NOTE	An implementation with limited ID range may not be able to receive the frame.									

7.1.6 Protocol exception behaviour on non-nominal bit

[Table 6](#) specifies the test of an IUT that switches to protocol exception on non-nominal bit value

Table 6 — Protocol exception behaviour

Item	Description									
Purpose	The purpose of this test is to verify that the IUT switches to protocol exception on non-nominal values of the bits described in test variables.									
CAN_VERSION	CAN FD Tolerant					CAN FD enabled				
Test variables	FDF = 1					FDF = 1				
	DLC					“res” bit = 1				
	Data: All data byte with the same value					DLC				
	Bit rate ratio between nominal and data bit rate					Data: All data byte with the same value				
						Bit rate ratio between nominal and data bit rate				
Elementary test cases	Test	Format	DLC	data	bit rate ratio	Test	Format	DLC	data	bit rate ratio
	#1	FBFF	A _h ;	AA _h ;	1:2	#1	FBFF	A _h ;	AA _h ;	1:2
	#2	FBFF	F _h ;	FF _h ;	1:8	#2	FBFF	F _h ;	FF _h ;	1:8
	#3	CBFF ^a	F _h ;	FF _h ;	—	Protocol exception handling shall be enabled.				
^a Valid classical frame with non-nominal FDF bit.										

Table 6 (continued)

Item	Description
Set-up	The IUT is left in the default state.
Execution	A single test frame is used for the elementary test, followed immediately by a valid Classical CAN frame.
Response	The IUT shall not generate any error flag in this test frame. The IUT shall not acknowledge the test frame. A following data frame in classical frame format received by the IUT during the test state shall match the data sent in the test frame.
^a Valid classical frame with non-nominal FDF bit.	

7.1.7 Minimum time for bus idle after protocol exception handling

[Table 7](#) specifies the test of the correct reception behaviour after protocol exception (IDLE detection).

Table 7 — Minimum time for bus idle after protocol exception

Item	Description
Purpose	This test verifies the behaviour of an IUT in protocol exception state when receiving frames separated by different times for inter-frame space.
CAN_VERSION	CAN FD tolerant CAN FD enabled
Test variables	Intermission field length Intermission field length Protocol exception handling shall be enabled
Elementary test cases	#1 The second frame starts after the third intermission bit + 1 bit time after the first frame. #2 The second frame starts after the third intermission bit of the first frame. #3 The second frame starts after the second intermission bit of the first frame followed by a third frame starts after the third intermission bit of the previous frame.
Set-up	The IUT is left in the default state.
Execution	The LT send a frame with non-nominal bit in control field causing protocol exception behaviour. The LT send a valid classical frame according to elementary test cases.
Response	The IUT shall not generate any error flag during the test. The IUT shall only acknowledge the last test frame in each test sequence.

7.1.8 DLC greater than 8

[Table 8](#) specifies the test of correct reception of classical frame with a DLC field greater than 8.

Table 8 — DLC greater than 8

Item	Description
Purpose	This test verifies the behaviour of the IUT when receiving a correct classical frame with a DLC greater than 8.
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled
Test variables	DLC FDF = 0

Table 8 (continued)

Item	Description
Elementary test cases	There are seven elementary tests, for which $DLC \in [9_h, F_h]$. DLC #1 9_h #2 A_h #3 B_h #4 C_h #5 D_h #6 E_h #7 F_h
Set-up	The IUT is left in the default state.
Execution	A single test frame is used for each of the elementary tests.
Response	The IUT shall not generate any error flag during the test. The IUT shall acknowledge the test frame. The data and DLC received by the IUT during the test state shall match the data and DLC sent in the test frame.

7.1.9 Absent bus idle — Valid frame reception

[Table 9](#) specifies the test of the correct reception of two consecutive frames which are not separated by a bus idle state.

Table 9 — Absent bus idle — Valid frame reception

Item	Description
Purpose	This test verifies the behaviour of the IUT when receiving two consecutive frames not separated by a bus idle state.
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled CAN FD enabled
Test variables	Intermission field length FDF = 0 Intermission field length FDF = 1
Elementary test cases	#1 The second frame starts after the second intermission bit of the first frame. #2 The second frame starts after the third intermission bit of the first frame.
Set-up	The IUT is left in the default state.
Execution	Two different test frames are used for each of the two elementary tests.
Response	The IUT shall not generate any error flag during the test. The IUT shall acknowledge the test frames.

7.1.10 Stuff acceptance test in base format frame

[Table 10](#) specifies the test of the correct reception of frames in base frame format with particular data containing critical stuffing bit profiles in the different fields of the frame.

Table 10 — Stuff acceptance test in base format frame

Item	Description											
Purpose	This test verifies the behaviour of the IUT when receiving a correct base format frame with particular data containing critical stuffing bit profiles in the different fields of the frame according to test variables.											
CAN_VERSION	Classical CAN				CAN FD tolerant CAN FD enabled				CAN FD enabled			
Test variables	ID				ID				ID			
	RTR				RTR				RRS			
	FDF				DLC				BRS			
	DLC				DATA				ESI			
	DATA								DLC DATA			
Elementary test cases	CBFF				CBFF				FBFF			
	ID	CTRL	DATA		ID	CTRL	DATA		ID	CTRL	DATA	
	#1	078 _h	08 _h	01 _h , all others E1 _h	#1	078 _h	08 _h	01 _h , all others E1 _h	#1	078 _h	0AE _h	F8 _h all others 78 _h
	#2	41F _h	01 _h	00 _h	#2	41F _h	01 _h	00 _h	#2	47C _h	0A8 _h	all bytes 3C _h
	#3	707 _h	1F _h	all bytes 0F _h	#3	707 _h	0F _h	all bytes 0F _h	#3	41E _h	0BE _h	all bytes 1E _h
	#4	360 _h	10 _h	—	#4	360 _h	00 _h	—	#4	20F _h	09F _h	all bytes 0F _h
	#5	730 _h	10 _h	—	#5	730 _h	00 _h	—	#5	107 _h	28F _h	all bytes 87 _h
	#6	47F _h	01 _h	1F _h	#6	47F _h	01 _h	1F _h	#6	7C3 _h	083 _h	all bytes C3 _h
	#7	758 _h	00 _h	—	#7	758 _h	00 _h	—	#7	3E1 _h	0A3 _h	all bytes E1 _h
	#8	777 _h	01 _h	1F _h	#8	777 _h	01 _h	1F _h	#8	1F0 _h	0A1 _h	all bytes F0 _h
	#9	7EF _h	42 _h	—	#9	7Ef _h	42 _h	—	#9	000 _h	0A0 _h	—
#10	3EA _h	5F _h	—	#10	3Ea _h	4F _h	—	#10	7FF _h	0B0 _h	—	
Set-up	The IUT is left in the default state.											
Execution	A single test frame is used for each elementary test.											
Response	The IUT shall not generate any error flag during the test.											
	The IUT shall acknowledge the test frame.											
	The data received by the IUT during the test state shall match the data sent in the test frame.											

7.1.11 Stuff acceptance test in extended format frame

[Table 11](#) specifies the test of the correct reception of frames in extended frame with particular data containing critical stuffing bit profiles in the different fields of the frame.

Table 11 — Stuff acceptance test in extended format frame

Item	Description										
Purpose	This test verifies the behaviour of the IUT when receiving a correct extended frame with particular data containing critical stuffing bit profiles in the different fields of the frame according to test variables.										
CAN_VERSION	Classical CAN				CAN FD tolerant CAN FD enabled				CAN FD enabled		
Test variables	ID				ID				ID		
	SRR				SRR				SRR		
	RTR				RTR				RRS		
	FDF				DLC				BRS		
	r0				DATA				ESI		
	DLC				FDF = 0				DLC		
	DATA								DATA FDF = 1		
Elementary test cases	CEFF				CEFF				FEFF		
	ID	CTRL	DATA		ID	CTRL	DATA		ID	CTRL	DATA
	#1	07C30F0F _h	188 _h	all bytes 3C _h	#1	07C30F0F _h	188 _h	all bytes 3C _h	#1	01E38787 _h	6AE _h F8 _h
	#2	07C0F0F0 _h	181 _h	00 _h	#2	07C0F0F0 _h	181 _h	00 _h			all others 78 _h
	#3	01E31717 _h	19F _h	all bytes 0F _h	#3	01E31717 _h	19F _h	all bytes 0F _h	#2	11F3C3C3 _h	2A8 _h all bytes 3C _h
	#4	01E00FF0 _h	1BC _h	1F _h , 0F _h , E0 _h , F0 _h , 7F _h , E0 _h , FF _h , 20 _h	#4	01E00FF0 _h	19C _h	1F _h , 0F _h , E0 _h , F0 _h , 7F _h , E0 _h , FF _h , 20 _h	#3	1079C1E1 _h	6BE _h all bytes 1E _h
	#5	1FB80000 _h	181 _h	A0 _h	#5	1FB80000 _h	181 _h	A0 _h	#4	083DF0F0 _h	69F _h all bytes 0F _h
	#6	00BC540F _h	1E0 _h	—	#6	00BC540F _h	1C0 _h	—	#5	041EF878 _h	68F _h all bytes 87 _h
	#7	155D5557 _h	1FF _h	—	#7	155D5557 _h	1DF _h	—	#6	1F0C3C3C _h	683 _h all bytes C3 _h
									#7	0F861E1E _h	6A3 _h all bytes E1 _h
									#8	07C30F0F _h	6A1 _h all bytes F0 _h
Set-up	The IUT is left in the default state.										
Execution	A single test frame is used for each of the elementary tests.										
Response	The IUT shall not generate any error flag during the test.										
	The IUT shall acknowledge the test frame.										
	The data received by the IUT during the test state shall match the data sent in the test frame.										
NOTE	An ID limited implementation may not receive the frame.										

7.1.12 Message validation

[Table 12](#) specifies the test of the point of time at which a message is taken to be valid.

Table 12 — Message validation

Item	Description	
Purpose	The purpose of this test is to verify the point of time at which a message is taken to be valid by the IUT.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	EOF FDF = 0	EOF FDF = 1
Elementary test cases	Number of elementary tests: 1 #1 The last bit of the EOF is forced to dominant state.	
Set-up	The IUT is left in the default state.	
Execution	A single test frame is used for the elementary test.	
Response	The IUT shall not generate any error flag during the test. The IUT shall acknowledge the test frame. The IUT shall generate an overload frame. The data received by the IUT during the test state shall match the data sent in the test frame.	

7.2 Test class 2, error detection

7.2.1 Bit error in data frame

[Table 13](#) specifies the test of an IUT that detects a bit error in data frame when the dominant ACK slot is forced to recessive state by LT.

Table 13 — Bit error in classical data frame

Item	Description	
Purpose	This test verifies that the IUT detects a bit error when the dominant ACK slot is forced to recessive state by LT.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	ACK Slot FDF = 0	ACK Slot FDF = 1
Elementary test cases	Number of elementary test cases: 1 #1 The dominant acknowledgement bit sent by the IUT is forced to recessive state.	
Set-up	The IUT is left in the default state.	
Execution	A single test frame is used for the elementary test.	
Response	The IUT shall generate an active error frame starting at the bit position following the bit error.	
NOTE	The receiver will generate an ACK immediately after CRC delimiter.	

7.2.2 Stuff error for basic frame

[Table 14](#) specifies the test of a stuff error whenever it receives 6 consecutive bits of the same value until the position of the CRC delimiter in a base format frame.

Table 14 — Stuff error for basic frame

Item	Description												
Purpose	This test verifies that the IUT detects a stuff error whenever it receives 6 consecutive bits of the same value until the position of the CRC delimiter in a base format frame.												
CAN_VERSION	Classical CAN				CAN FD tolerant CAN FD enabled				CAN FD enabled				
Test variables	ID RTR FDF DLC DATA				ID RTR DLC DATA				ID RTR BRS ESI DLC DATA byte 0 defined, all others = 55 _h				
Elementary test cases	All 63 stuff bits within the defined frames will be tested.				All 59 stuff bits within the defined frames will be tested.				All 40 stuff bits within the defined frames will be tested.				
		CBFF				CBFF				FBFF			
		ID	CTRL	DATA		ID	CTRL	DATA		ID	CTRL	DATA	
	#1	078 _h	08 _h	01 _h , all others E1 _h	#1	078 _h	08 _h	01 _h , all others E1 _h	#1	078 _h	0AE _h	F8 _h , #2 47C _h	0A8 _h 3C _h
	#2	41F _h	01 _h	00 _h	#2	41F _h	01 _h	00 _h	#3	41E _h	0BE _h	1E _h	
	#3	707 _h	1F _h	all byte 0F _h	#3	707 _h	0F _h	all byte 87 _h	#4	20F _h	09F _h	0F _h	
	#4	360 _h	10 _h	—	#4	360 _h	00 _h	—	#5	107 _h	28F _h	87 _h	
	#5	730 _h	10 _h	—	#5	730 _h	00 _h	—	#6	7C3 _h	083 _h	C3 _h	
	#6	47F _h	01 _h	1F _h	#6	47F _h	01 _h	1F _h	#7	3E1 _h	0A3 _h	E1 _h	
	#7	758 _h	00 _h	—	#7	758 _h	00 _h	—	#8	1F0 _h	0A1 _h	F0 _h	
	#8	777 _h	01 _h	1F _h	#8	777 _h	01 _h	1F _h	#9	000 _h	0A0 _h	—	
	#9	7EF _h	42 _h	—	#9	7EF _h	42 _h	—	#10	7FF _h	0B0 _h	—	
#10	3EA _h	5F _h	—	#10	3EA _h	4F _h	—						
Set-up	The IUT is left in the default state.												
Execution	A single test frame is used for each elementary test. The LT forces one of the stuff bits to its complement.												
Response	The IUT shall generate an active error frame starting at the bit position following the stuff error.												

7.2.3 Stuff error for extended frame

[Table 15](#) specifies the test of a stuff error whenever it receives 6 consecutive bits of the same value until the position of the CRC delimiter in an extended frame.

Table 15 — Stuff error for extended frame

Item	Description																																																																																																																										
Purpose	This test verifies that the IUT detects a stuff error whenever it receives 6 consecutive bits of the same value until the position of the CRC delimiter in an extended frame.																																																																																																																										
CAN_VERSION	Classical CAN	CAN FD tolerant CAN FD enabled	CAN FD enabled																																																																																																																								
Test variables	basic ID SRR ID extension RTR FDF r0 DLC DATA	basic ID SRR ID extension RTR FDF = 0 r0 DLC DATA	basic ID SRR ID extension RRS FDF = 1 res BRS ESI DLC DATA byte 0 defined, all others = 55 _h																																																																																																																								
Elementary test cases	There are 90 elementary tests to perform. <table> <tr> <th></th><th>ID</th><th>CTRL</th><th>DATA</th></tr> <tr> <td>#1</td><td>07C30F0F_h</td><td>188_h</td><td>all 3C_h</td></tr> <tr> <td>#2</td><td>07C0F0F0_h</td><td>181_h</td><td>00_h</td></tr> <tr> <td>#3</td><td>01E31717_h</td><td>19F_h</td><td>all 0F_h</td></tr> <tr> <td>#4</td><td>01E00FF0_h</td><td>1BC_h</td><td>1F_h, 0F_h, E0_h, F0_h, 7F_h, E0_h, FF_h, 20_h</td></tr> <tr> <td>#5</td><td>1FB80000_h</td><td>181_h</td><td>A0_h</td></tr> <tr> <td>#6</td><td>00BC540F_h</td><td>1E0_h</td><td>—</td></tr> <tr> <td>#7</td><td>155D5557_h</td><td>1FF_h</td><td>—</td></tr> <tr> <td>#8</td><td>00000000_h</td><td>181_h</td><td>00_h</td></tr> </table>		ID	CTRL	DATA	#1	07C30F0F _h	188 _h	all 3C _h	#2	07C0F0F0 _h	181 _h	00 _h	#3	01E31717 _h	19F _h	all 0F _h	#4	01E00FF0 _h	1BC _h	1F _h , 0F _h , E0 _h , F0 _h , 7F _h , E0 _h , FF _h , 20 _h	#5	1FB80000 _h	181 _h	A0 _h	#6	00BC540F _h	1E0 _h	—	#7	155D5557 _h	1FF _h	—	#8	00000000 _h	181 _h	00 _h	There are 85 elementary tests to perform. <table> <tr> <th></th><th>ID</th><th>CTRL</th><th>DATA</th></tr> <tr> <td>#1</td><td>07C30F0F_h</td><td>188_h</td><td>all 3C_h</td></tr> <tr> <td>#2</td><td>07C0F0F0_h</td><td>181_h</td><td>00_h</td></tr> <tr> <td>#3</td><td>01E31717_h</td><td>19F_h</td><td>all 0F_h</td></tr> <tr> <td>#4</td><td>01E00FF0_h</td><td>19C_h</td><td>1F_h, 0F_h, E0_h, F0_h, 7F_h, E0_h, FF_h, 20_h</td></tr> <tr> <td>#5</td><td>1FB80000_h</td><td>181_h</td><td>A0_h</td></tr> <tr> <td>#6</td><td>00BC540F_h</td><td>1C0_h</td><td>—</td></tr> <tr> <td>#7</td><td>155D5557_h</td><td>1DF_h</td><td>—</td></tr> <tr> <td>#8</td><td>00000000_h</td><td>181_h</td><td>00_h</td></tr> </table>		ID	CTRL	DATA	#1	07C30F0F _h	188 _h	all 3C _h	#2	07C0F0F0 _h	181 _h	00 _h	#3	01E31717 _h	19F _h	all 0F _h	#4	01E00FF0 _h	19C _h	1F _h , 0F _h , E0 _h , F0 _h , 7F _h , E0 _h , FF _h , 20 _h	#5	1FB80000 _h	181 _h	A0 _h	#6	00BC540F _h	1C0 _h	—	#7	155D5557 _h	1DF _h	—	#8	00000000 _h	181 _h	00 _h	There are 88 elementary tests to perform. <table> <tr> <th></th><th>ID</th><th>CTRL</th><th>DATA</th></tr> <tr> <td>#1</td><td>01E38787_h</td><td>6AE_h</td><td>F8_h</td></tr> <tr> <td>#2</td><td>11F3C3C3_h</td><td>2A8_h</td><td>3C_h</td></tr> <tr> <td>#3</td><td>1079C1E1_h</td><td>6BE_h</td><td>1E</td></tr> <tr> <td>#4</td><td>083DF0F0_h</td><td>69F_h</td><td>0F_h</td></tr> <tr> <td>#5</td><td>041EF878_h</td><td>68F_h</td><td>87_h</td></tr> <tr> <td>#6</td><td>1F0C3C3C_h</td><td>683_h</td><td>C3_h</td></tr> <tr> <td>#7</td><td>0F861E1E_h</td><td>6A3_h</td><td>E1_h</td></tr> <tr> <td>#8</td><td>07C30F0F_h</td><td>6A1_h</td><td>F0_h</td></tr> <tr> <td>#9</td><td>01E38787_h</td><td>3A0_h</td><td>—</td></tr> <tr> <td>#10</td><td>11F3C3C3_h</td><td>380_h</td><td>—</td></tr> <tr> <td>#11</td><td>00000000_h</td><td>6B0_h</td><td>—</td></tr> </table>		ID	CTRL	DATA	#1	01E38787 _h	6AE _h	F8 _h	#2	11F3C3C3 _h	2A8 _h	3C _h	#3	1079C1E1 _h	6BE _h	1E	#4	083DF0F0 _h	69F _h	0F _h	#5	041EF878 _h	68F _h	87 _h	#6	1F0C3C3C _h	683 _h	C3 _h	#7	0F861E1E _h	6A3 _h	E1 _h	#8	07C30F0F _h	6A1 _h	F0 _h	#9	01E38787 _h	3A0 _h	—	#10	11F3C3C3 _h	380 _h	—	#11	00000000 _h	6B0 _h	—
	ID	CTRL	DATA																																																																																																																								
#1	07C30F0F _h	188 _h	all 3C _h																																																																																																																								
#2	07C0F0F0 _h	181 _h	00 _h																																																																																																																								
#3	01E31717 _h	19F _h	all 0F _h																																																																																																																								
#4	01E00FF0 _h	1BC _h	1F _h , 0F _h , E0 _h , F0 _h , 7F _h , E0 _h , FF _h , 20 _h																																																																																																																								
#5	1FB80000 _h	181 _h	A0 _h																																																																																																																								
#6	00BC540F _h	1E0 _h	—																																																																																																																								
#7	155D5557 _h	1FF _h	—																																																																																																																								
#8	00000000 _h	181 _h	00 _h																																																																																																																								
	ID	CTRL	DATA																																																																																																																								
#1	07C30F0F _h	188 _h	all 3C _h																																																																																																																								
#2	07C0F0F0 _h	181 _h	00 _h																																																																																																																								
#3	01E31717 _h	19F _h	all 0F _h																																																																																																																								
#4	01E00FF0 _h	19C _h	1F _h , 0F _h , E0 _h , F0 _h , 7F _h , E0 _h , FF _h , 20 _h																																																																																																																								
#5	1FB80000 _h	181 _h	A0 _h																																																																																																																								
#6	00BC540F _h	1C0 _h	—																																																																																																																								
#7	155D5557 _h	1DF _h	—																																																																																																																								
#8	00000000 _h	181 _h	00 _h																																																																																																																								
	ID	CTRL	DATA																																																																																																																								
#1	01E38787 _h	6AE _h	F8 _h																																																																																																																								
#2	11F3C3C3 _h	2A8 _h	3C _h																																																																																																																								
#3	1079C1E1 _h	6BE _h	1E																																																																																																																								
#4	083DF0F0 _h	69F _h	0F _h																																																																																																																								
#5	041EF878 _h	68F _h	87 _h																																																																																																																								
#6	1F0C3C3C _h	683 _h	C3 _h																																																																																																																								
#7	0F861E1E _h	6A3 _h	E1 _h																																																																																																																								
#8	07C30F0F _h	6A1 _h	F0 _h																																																																																																																								
#9	01E38787 _h	3A0 _h	—																																																																																																																								
#10	11F3C3C3 _h	380 _h	—																																																																																																																								
#11	00000000 _h	6B0 _h	—																																																																																																																								
Set-up	The IUT is left in the default state.																																																																																																																										
Execution	A single test frame is used for each elementary test. The LT forces one of the stuff bits to its complement.																																																																																																																										
Response	The IUT shall generate an active error frame starting at the bit position following the stuff error.																																																																																																																										

7.2.4 Stuff error for FD frame payload bytes

[Table 16](#) specifies the test of a stuff error whenever the IUT receives 6 consecutive bits of the same value until the position of the CRC in an FD format frame with maximum payload DLC = 15.

Table 16 — Stuff error for CAN FD enabled implementation

Item	Description																											
Purpose	This test verifies that the IUT detects a stuff error whenever it receives 6 consecutive bits of the same value until the position of the CRC delimiter in a base format frame.																											
CAN_VERSION	CAN FD enabled																											
Test variables	DATA byte 0–63 ID = 555 _h IDE = 0 DLC = 15 FDF = 1																											
Elementary test cases	All 1 008 stuff bits within the defined data bytes 1 to 63 will be tested. <table><thead><tr><th></th><th>Data Byte 0</th><th>Data bytes 1 – 63</th></tr></thead><tbody><tr><td>#1</td><td>10_h</td><td>78_h</td></tr><tr><td>#2</td><td>78_h</td><td>3C_h</td></tr><tr><td>#3</td><td>34_h</td><td>1E_h</td></tr><tr><td>#4</td><td>12_h</td><td>0F_h</td></tr><tr><td>#5</td><td>0F_h</td><td>87_h</td></tr><tr><td>#6</td><td>17_h</td><td>C3_h</td></tr><tr><td>#7</td><td>43_h</td><td>E1_h</td></tr><tr><td>#8</td><td>21_h</td><td>F0_h</td></tr></tbody></table>		Data Byte 0	Data bytes 1 – 63	#1	10 _h	78 _h	#2	78 _h	3C _h	#3	34 _h	1E _h	#4	12 _h	0F _h	#5	0F _h	87 _h	#6	17 _h	C3 _h	#7	43 _h	E1 _h	#8	21 _h	F0 _h
	Data Byte 0	Data bytes 1 – 63																										
#1	10 _h	78 _h																										
#2	78 _h	3C _h																										
#3	34 _h	1E _h																										
#4	12 _h	0F _h																										
#5	0F _h	87 _h																										
#6	17 _h	C3 _h																										
#7	43 _h	E1 _h																										
#8	21 _h	F0 _h																										
Set-up	The IUT is left in the default state.																											
Execution	A single test frame is used for each elementary test. In each elementary test, the LT forces one of the stuff bits to its complement.																											
Response	The IUT shall generate an active error frame starting at the bit position following the stuff error.																											

7.2.5 CRC error

[Table 17](#) specifies the test of an IUT that uses the specific CRC mechanism and detecting a CRC error, generates an error frame at the correct position and does not detect an error when monitoring a dominant bit at the ACK slot while sending a recessive one.

Table 17 — CRC error

Item	Description
Purpose	The purpose of this test is to verify — that the IUT uses the specific CRC mechanism according to frame format, — that the IUT detecting a CRC error and generates an error frame at the correct position, and — that the IUT does not detect an error when monitoring a dominant bit at the ACK slot while sending a recessive one.
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled
Test variables	CRC DLC - to cause different CRC types FDF = 1 SOF

Table 17 (continued)

Item	Description	
Elementary test cases	Number of elementary tests: 3	Number of elementary tests: 6
	#1 A dominant bit in the CRC field is changed in a recessive bit.	#1 and #2 A dominant bit in the CRC field is changed in a recessive bit for CRC-17 with DLC ≤ 10 (#1) and CRC-21 with DLC > 10 (#2) (test for CRC value).
	#2 A recessive bit in the CRC field is changed in a dominant bit.	#3 and #4 A recessive bit in the CRC field is changed in a dominant bit for CRC-17 with DLC ≤ 10 (#3) and CRC-21 with DLC > 10 (#4) (test for CRC value).
	#3 The dominant SOF bit in the frame is changed in a recessive one followed by an ID 001 _h .	#5 The test system sends a frame where two times a recessive stuff bit becomes a normal bit by losing one of the previous bits by synchronization issues while the CRC register is equal zero (test for stuff-counter).
Set-up	The IUT is left in the default state.	
Execution	A single test frame is used for each elementary test. The LT modifies the frame according to elementary test cases.	
Response	The IUT shall not acknowledge the test frame. The IUT shall generate an active error frame starting at the first bit position following the ACK delimiter.	The IUT shall not acknowledge the test frame. The IUT shall generate an active error frame starting at the fourth bit position following the CRC delimiter.

7.2.6 Combination of CRC error and form error

Table 18 specifies the test of an IUT that detects a CRC error and a form error on the CRC delimiter in the same frame.

Table 18 — Combination of CRC error and form error

Item	Description	
Purpose	The purpose of this test is to verify that an IUT detecting a CRC error and a form error on the CRC delimiter in the same frame generates only one single 6 bits long error flag starting on the bit following the CRC delimiter.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	CRC FDF = 0	CRC DLC - to cause different CRC types FDF = 1
Elementary test cases	Number of elementary tests: 1 #1 CRC (15)	Number of elementary tests: 2 #1 DLC ≤ 10 -> CRC (17) #2 DLC > 10 -> CRC (21)

Table 18 (continued)

Item	Description
Set-up	The IUT is left in the default state.
Execution	A single test frame is used for the elementary test. The LT generates a CAN frame with CRC error and form error at CRC delimiter according to elementary test cases.
Response	The IUT shall generate one active error frame starting at the bit position following the CRC delimiter.

7.2.7 Form error in data frame at “CRC delimiter” bit position

[Table 19](#) specifies the test of an IUT that detects a form error when the recessive bit of CRC delimiter is forced to dominant state.

Table 19 — Form error in data frame at “CRC delimiter” bit position

Item	Description
Purpose	This test verifies that the IUT detects a form error when the recessive bit of CRC delimiter is forced to dominant state by LT.
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled CAN FD enabled
Test variables	CRC delimiter FDF = 0 CRC delimiter FDF = 1
Elementary test cases	There is only one elementary test to perform. #1 CRC delimiter = 0
Set-up	The IUT is left in the default state.
Execution	A single test frame is used for the elementary test. The LT generates a CAN frame with form error at CRC delimiter according to elementary test cases.
Response	The IUT shall generate an active error frame at the bit position following the CRC delimiter.

7.2.8 Form error at fixed stuff bit in FD frames

[Table 20](#) specifies the test of an IUT that detects a form error when a fixed stuff bit did not match to the previous bit.

Table 20 — Form error at fixed stuff bit in FD frames

Item	Description
Purpose	This test verifies that the IUT detects a form error when a fixed stuff bit did not match to the previous bit.
CAN_VERSION	CAN FD enabled
Test variables	DLC - to cause different CRC types FDF = 1

Table 20 (continued)

Item	Description
Elementary test cases	There are 22 elementary tests to perform. Tests to perform on recessive stuff bits: #1 DLC ≤ 10 → CRC (17) field – (6 bits) #2 DLC > 10 → CRC (21) field – (7 bits) Tests to perform on dominant stuff bits: #3 DLC ≤ 10 → CRC (17) field – (6 bits) #4 DLC > 10 → CRC (21) field – (7 bits)
Set-up	The IUT is left in the default state.
Execution	The LT corrupts a fixed stuff bit according to elementary test cases.
Response	The IUT shall generate an error frame at the bit position following the stuff bit.

7.2.9 Form error in data frame at “ACK delimiter” bit position

[Table 21](#) specifies the test of an IUT that detects a form error when the recessive ACK delimiter is forced to dominant state.

Table 21 — Form error in data frame at “ACK delimiter” bit position

Item	Description								
Purpose	This test verifies that the IUT detects a form error when the recessive ACK delimiter is forced to dominant state by LT.								
CAN_VERSION	<table> <tr> <td>Classical CAN</td><td></td></tr> <tr> <td>CAN FD tolerant</td><td>CAN FD enabled</td></tr> <tr> <td>CAN FD enabled</td><td></td></tr> </table>	Classical CAN		CAN FD tolerant	CAN FD enabled	CAN FD enabled			
Classical CAN									
CAN FD tolerant	CAN FD enabled								
CAN FD enabled									
Test variables	<table> <tr> <td>ACK delimiter</td><td>ACK delimiter</td></tr> <tr> <td>ACK = 0</td><td>ACK1 = 0</td></tr> <tr> <td>FDF = 0</td><td>ACK2 = 0</td></tr> <tr> <td></td><td>FDF = 1</td></tr> </table>	ACK delimiter	ACK delimiter	ACK = 0	ACK1 = 0	FDF = 0	ACK2 = 0		FDF = 1
ACK delimiter	ACK delimiter								
ACK = 0	ACK1 = 0								
FDF = 0	ACK2 = 0								
	FDF = 1								
Elementary test cases	There is only one elementary test to perform. #1 ACK delimiter = 0								
Set-up	The IUT is left in the default state.								
Execution	A single test frame is used for the elementary test. The LT generates a CAN frame with form error at ACK delimiter according to elementary test cases.								
Response	The IUT shall generate an active error frame at the bit position following the ACK delimiter.								

7.2.10 Form error in data frame at “EOF”

[Table 22](#) specifies the test of an IUT that detects a form error when one of 6 first recessive bits of EOF is forced to dominant state.

Table 22 — Form error in data frame at “EOF”

Item	Description	
Purpose	This test verifies that the IUT detects a form error when one of 6 first recessive bits of EOF is forced to dominant state by LT.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	EOF FDF = 0	EOF FDF = 1
Elementary test cases	There are five elementary tests to perform: #1 to #5 corrupting the first until the fifth bit position.	
Set-up	The IUT is left in the default state.	
Execution	A single test frame is used for the elementary test. The LT generates a CAN frame with form error at EOF according to elementary test cases.	
Response	The IUT shall generate an active error frame at the bit position following the corrupted bit.	

7.2.11 Message non-validation

[Table 23](#) specifies the test of the point of time at which a message is still considered as non-valid by the IUT.

Table 23 — Message non-validation

Item	Description	
Purpose	The purpose of this test is to verify the point of time at which a message is still considered as non-valid by the IUT.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	EOF FDF = 0	EOF FDF = 1
Elementary test cases	There is only one elementary test to perform. #1 The sixth bit of the EOF is forced to dominant.	
Set-up	The IUT has to be initialized with data different from those used in the test frame.	
Execution	A single test frame is used for the elementary test. The LT generates a CAN frame with form error at EOF according to elementary test cases.	
Response	The IUT shall generate an active error frame. The data initialized during the set-up state shall remain unchanged. No frame reception shall be indicated to the upper layers of the IUT.	

7.3 Test class 3, error frame management

7.3.1 Error flag longer than 6 bits

[Table 24](#) specifies the test that the IUT tolerates up to 7 consecutive dominant bits after sending an active error flag.

Table 24 — Error flag longer than 6 bits

Item	Description	
Purpose	This test verifies that the IUT tolerates up to 7 consecutive dominant bits after sending an active error flag.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	FDF = 0	FDF = 1
Elementary test cases	Elementary tests to perform: #1 lengthening the error flag by 1 dominant bit; #2 lengthening the error flag by 4 dominant bits; #3 lengthening the error flag by 7 dominant bits.	
Set-up	The IUT is left in the default state.	
Execution	The LT causes the IUT to generate an error frame in data field. The LT lengthens the error flag generated by the IUT according to elementary test cases.	
Response	After sending the active error flag, the IUT sends recessive bits.	

7.3.2 Data frame starting on the third bit of intermission field

[Table 25](#) specifies the test of an IUT that accepts a frame starting after the second bit of the intermission following the error frame it has transmitted.

Table 25 — Data frame starting on the third bit of intermission field

Item	Description	
Purpose	The purpose of this test is to verify that an IUT accepts a frame starting after the second bit of the intermission following the error frame it has transmitted.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	FDF = 0	FDF = 1
Elementary test cases	There is one elementary test to perform. #1 Frame is started 2 bits after the end of the error delimiter.	
Set-up	The IUT is left in the default state.	
Execution	The LT causes the IUT to generate an error frame in data field. The LT sends a valid frame according to elementary test cases.	
Response	The IUT shall acknowledge the test frame in data field. The data received by the IUT during the test state shall match the data sent in the test frame.	

7.3.3 Bit error in error flag

[Table 26](#) specifies the test of a bit error in error flag when one of the 6 dominant bits of the error flag the IUT transmits is forced to recessive state.

Table 26 — Bit error in error flag

Item	Description	
Purpose	This test verifies that the IUT detects a bit error when one of the 6 dominant bits of the error flag it transmits is forced to recessive state by LT.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	FDF = 0	FDF = 1
Elementary test cases	There are three elementary tests to perform: #1 corrupting the first bit of the error flag; #2 corrupting the third bit of the error flag; #3 corrupting the sixth bit of the error flag.	
Set-up	The IUT is left in the default state.	
Execution	The LT causes the IUT to generate an error frame in data field. The LT forces one of the bits of the error frame generated by the IUT to recessive state according to elementary test cases.	
Response	The IUT shall restart with an active error frame at the bit position following the corrupted bit.	

7.3.4 Form error in error delimiter

[Table 27](#) specifies the test of a form error when receiving an invalid error delimiter.

Table 27 — Form error in error delimiter

Item	Description	
Purpose	This test verifies that the IUT detects a form error when receiving an invalid error delimiter.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	FDF = 0	FDF = 1
Elementary test cases	The LT replaces one of the 8 recessive bits of the error delimiter by a dominant bit. Elementary tests to perform: #1 corrupting the second bit of the error delimiter; #2 corrupting the fourth bit of the error delimiter; #3 corrupting the seventh bit of the error delimiter.	
Set-up	The IUT is left in the default state.	
Execution	The LT causes the IUT to generate an error frame in data field. The LT forces one of the bits of the error delimiter generated by the IUT to dominant state according to elementary test cases.	
Response	The IUT shall restart with an active error frame at the bit position following the replaced bit.	

7.4 Test class 4, overload frame management

7.4.1 MAC overload generation during intermission field

[Table 28](#) specifies the test of a MAC overload generation frame when detecting a dominant bit on one of the 2 first recessive bits of the intermission field.

Table 28 — MAC overload generation during intermission field

Item	Description	
Purpose	This test verifies that the IUT generates an overload frame when detecting a dominant bit on one of the 2 first recessive bits of the intermission field.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	FDF = 0	FDF = 1
Elementary test cases	There are two elementary tests to perform: #1 first bit of intermission; #2 second bit of intermission.	
Set-up	The IUT is left in the default state.	
Execution	One test frame is used for each of the two elementary tests. The LT forces one of the 2 first bits of the intermission field of the test frame to dominant state according to elementary test cases.	
Response	The IUT generates an overload frame at the bit position following the dominant bit.	

7.4.2 Last bit of EOF

[Table 29](#) specifies the test of a correct generated overload frame when detecting a dominant state on the last bit of EOF.

Table 29 — Last bit of EOF

Item	Description	
Purpose	This test verifies that the IUT generates an overload frame when detecting a dominant state on the last bit of EOF.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	EOF FDF = 0	EOF FDF = 1
Elementary test cases	There is one elementary test to perform. #1 Last bit of the EOF.	
Set-up	The IUT is left in the default state.	
Execution	The LT forces 1 bit of the EOF to a dominant state according to elementary test cases.	
Response	The IUT generates an overload frame at the bit position following the dominant bit.	

7.4.3 Eighth bit of an error and overload delimiter

[Table 30](#) specifies the test of a correct generated overload frame when detecting a dominant bit on the eighth bit of an error and overload delimiter it is transmitting.

Table 30 — Eighth bit of an error and overload delimiter

Item	Description	
Purpose	This test verifies that the IUT generates an overload frame when detecting a dominant bit on the eighth bit of an error and overload delimiter it is transmitting.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	Error delimiter Overload delimiter FDF = 0	Error delimiter Overload delimiter FDF = 1
Elementary test cases	There are two elementary tests to perform: #1 apply error at the eighth bit of the error delimiter; #2 apply error at the eighth bit of the overload delimiter.	
Set-up	The IUT is left in the default state.	
Execution	The LT causes the IUT to generate an error frame in data field or an overload frame after a data frame. The LT forces 1 bit to dominant state according to elementary test cases.	
Response	The IUT generates an overload frame starting at the bit position following the dominant bit forced by LT.	

7.4.4 Bit error in overload flag

[Table 31](#) specifies the test of a bit error when one of the 6 dominant bits of the overload flag the IUT transmits is forced to recessive state.

Table 31 — Bit error in overload flag

Item	Description	
Purpose	This test verifies that the IUT detects a bit error when one of the 6 dominant bits of the overload flag it transmits is forced to recessive state by LT.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	Overload flag FDF = 0	Overload flag FDF = 1
Elementary test cases	Elementary tests to perform: #1 corrupting the first bit of the overload flag; #2 corrupting the third bit of the overload flag; #3 corrupting the sixth bit of the overload flag.	
Set-up	The IUT is left in the default state.	
Execution	The LT causes the IUT to generate an overload frame after a data frame. The LT forces 1 bit of the overload flag to the recessive state according to elementary test cases.	
Response	The IUT shall generate an error frame at the bit position following the corrupted bit.	

7.4.5 Form error in overload delimiter

[Table 32](#) specifies the test of an IUT that detects a form error when receiving an invalid overload delimiter.

Table 32 — Form error in overload delimiter

Item	Description	
Purpose	This test verifies that the IUT detects a form error when receiving an invalid overload delimiter.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	Overload delimiter FDF = 0	Overload delimiter FDF = 1
Elementary test cases	The LT replaces one of the 8 recessive bits of the overload delimiter by a dominant bit. Elementary tests to perform: #1 corrupting the second bit of the overload delimiter; #2 corrupting the fourth bit of the overload delimiter; #3 corrupting the seventh bit of the overload delimiter.	
Set-up	The IUT is left in the default state.	
Execution	The LT causes the IUT to generate an overload frame after a data frame. The LT forces 1 bit of the overload delimiter to the dominant state according to elementary test cases.	
Response	The IUT generates an error frame starting at the bit position following the replaced bit.	

7.4.6 MAC overload generation during intermission field following an error frame

[Table 33](#) specifies the test of a correct generation of a MAC overload frame when detecting a dominant bit on one of the 2 first recessive bits of the intermission field.

Table 33 — MAC overload generation during intermission field following an error frame

Item	Description	
Purpose	This test verifies that the IUT generates an overload frame when detecting a dominant bit on one of the 2 first recessive bits of the intermission field.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	intermission field FDF = 0	intermission field FDF = 1
Elementary test cases	There are two elementary tests to perform” #1 intermission field bit 1 dominant; #2 intermission field bit 2 dominant.	
Set-up	The IUT is left in the default state.	
Execution	One test frame is used for each of the two elementary tests. The LT causes the IUT to generate an error frame in data field. The LT forces one of the 2 first bits of the intermission field after the previous error delimiter of the test frame to a dominant value according to elementary test cases.	
Response	The IUT generates an overload frame at the bit position following the dominant bit.	

7.4.7 MAC overload generation during intermission field following an overload frame

[Table 34](#) specifies the test of a correct generation of a MAC overload frame when detection a dominant bit on one of the 2 first recessive bits of the intermission field.

Table 34 — MAC overload generation during intermission field following an overload frame

Item	Description	
Purpose	This test verifies that the IUT generates an overload frame when detecting a dominant bit on one of the 2 first recessive bits of the intermission field.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	Intermission field of overload frame FDF = 0	Intermission field of overload frame FDF = 1
Elementary test cases	There are two elementary tests to perform: #1 intermission field bit 1 dominant; #2 intermission field bit 2 dominant.	
Set-up	The IUT is left in the default state.	
Execution	One test frame is used for each of the two elementary tests. The LT causes the IUT to generate an overload frame after a data frame. The LT forces one of the 2 first bits of the intermission field after the overload delimiter of the test frame to a dominant value.	
Response	The IUT generates an overload frame at the bit position following the dominant bit.	

7.5 Test class 5, passive error state class

7.5.1 Passive error flag completion test 1

[Table 35](#) specifies the test of an error passive IUT that considers the passive error flag as completed after the detection of 6 consecutive bits of the same value.

Table 35 — Passive error flag completion test 1

Item	Description	
Purpose	The purpose of this test is to verify that an error passive IUT considers the passive error flag as completed after the detection of 6 consecutive bits of the same value.	
CAN_VERSION	Classical CAN CAN FD tolerant CAN FD enabled	CAN FD enabled
Test variables	Passive error flag FDF = 0	Passive error flag FDF = 1
Elementary test cases	Elementary tests to perform: #1 superimposing the passive error flag by an active error flag starting at the first bit; #2 superimposing the passive error flag by an active error flag starting at the third bit; #3 superimposing the passive error flag by an active error flag starting at the sixth bit.	
Set-up	The IUT is set in passive state.	
Execution	The LT causes the IUT to generate a passive error frame in data field. During the passive error flag sent by the IUT, the LT sends an active error flag according to elementary test cases. At the end of the active error flag, the LT waits for (8 + 2) bit time before sending a valid test frame.	
Response	The IUT shall acknowledge the test frame.	