C&S Conformance Test
CAN – Data Link Layer
Register Functionality Test Specification
CLASSIC and FD Frame Format
(Processor Interface Test Specification)

Author(s) and Organization(s):	Frank Hofmann C&S group
Reviewed by:	FF C&S group
Revision:	4.1 D4
Date:	2020-03-26
Status:	Draft
Working document:	**confidential**
Document Scope: (Short description only)	Description of register functionality tests performed by C&S

Disclaimer

This test specification as released by C&S group GmbH is intended for the purpose of information only. The use of material contained in this test specification requires written agreement with C&S group GmbH. C&S group GmbH will not be liable for any unauthorized use of this test specification. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

Revision History

Date	Version	changes	Author
26.02.2007	V2.1 R03	Initial Version	CB
21.03.2007	V3 R00 D01	Formatierung…	AM
12.08	V3 R00 D02; V3 R00 D03;	Ausarbeitung der Test Fälle	FH
19.02.09	V3 R00 D04;	Vorlage auf GmbH umgestellt	FH
20.11.2015	V3 R01 d00	Enhancement of the CAN FD Protocol	FH
17.02.2016	V4 R00 d01	Update to FD Frame Format only	FH
20.12.2017	V4 R01 d03	Reading Classic Frame Format	FH
26.03.2020	V4 R01 d04	Disclaimer added	CW

1 Introduction

1.1 General

The Register Functionality tests supplement the ISO test cases specified in ISO 16845. All tests are normally executed for all available message buffers of the CAN implementation.

The Processor Interface Specification Version 4 supports the testing of devices, which are able to operate in Classic CAN Mode and in FD enabled Mode.

Testing of special features is handled as part of the Chapter Miscellaneous. The documented cases can be performed, if the feature is supported by the device.

Please have a look to the device documentation about supported features.

1.2 Definitions

1.2.1 Abbreviations

CAN	Controller Area Network
CAN FD	Controller Area Network Fast Data Rate
UT	Upper Tester
LT	Lower Tester
IUT	Implementation Under Test
PI	Processor Interface
RF	Register Functionality
SOF	Start Of Frame
ID	Identifier
RTR	Remote Transmission Request
IDE	Identifier Extension
DLC	Data Length Code
ACK	Acknowledge
EOF	End Of Frame
SOT	Start Of Test
EOT	End Of Test frame
d_SOT	Data Start Of Test frame
d_EOT	Data End Of Test frame
CC_CC	CAN Mode is Classic only
FD_CC	CAN Mode is FD with Classic Protocol only
FD_FD	CAN Mode is FD with all features of the CAN FD Protocol Enhancements

1.2.2 CAN-Frames

See ISO 11898-1

1.2.3 Error / Status handling

Each test case must be able to support a return value. The returned value is used as part of the Data Field of the EOT Message which is transmitted at the end of each test sequence.

Error Code definition:

- Only the case <0> = NO_ERROR is defined. NO_ERROR is the as default returned value.
- Every different value can be an ERROR Code or Status Code and is defined individually by the test cases.

- The interpretation of the returned value if the value is an Error Code or a Status Code is part of the usage of each test case and the used script preparation.

E.g. using as Status Code:

The configured test script defines that the Device must send the observed flag <STUFF_ERROR> as Status Code:

The test case defines the following Status Codes:

```
0x01 = BIT_ERROR
…
0x08 = STUFF_ERROR
```

Test script:

```
#define = STUFF_ERROR 0x08
…
LowerTesterReceivesEOT(STUFF_ERROR, 0)
```

The observation of the STUFF_ERROR Flag is flagged and the Upper Tester sends this information as part of the Data Field of the EOT Message.

The handling can be used to optimize the analyzing process, because sometimes the used test function is not able to allow breakpoints at critical test steps. In such cases it is easier to flag a Status Code to the developer, with the information of e.g. <STUFF_ERROR> observed.

The same handling can although be used to flag Errors like a <COMPARE_ERROR>. But this is a result which should not happen.

1.2.4 Test Groups

1.2.4.1 Test Mode CC_CC

General Setup:

Device Mode	Classic CAN Mode
Message Length	$DLC = \in \{0 \dots 15\} = \{0,1,2,3,4,5,6,7,8,9 \dots 15\}$
Mailbox structure	8 Byte per Mailbox
Implementation details	The DUT shall be implemented to use a mailbox structure which supports up to 8 Byte per Message Buffer.

1.2.4.2 Test Mode FD_CC

General Setup:

Device Mode	Classic CAN Mode with FD Mode enabled
Message Length	$DLC = \in \{0 \dots 15\} = \{0,1,2,3,4,5,6,7,8,9 \dots 15\}$
BRS	$BRS = \in \{0\}$
FDF	$FDF = \in \{0\}$
Mailbox structure	8 Byte per Mailbox
Implementation details	The DUT shall be implemented to use a mailbox structure which supports up to 64 Byte per Message Buffer. The Data Length is restricted to Classic Mode settings.

1.2.4.3 Test Mode FD_FD

General Setup:

Device Mode	FD CAN Mode
Message Length	DLC $= \in \{0 \dots 8\} = \{0,1,2,3,4,5,6,7,8\}$ DLC $= \in \{9 \dots 15\} = \{12,16,20,24,32,48,64\}$
BRS	$\in = \{0,1\}$
FDF	$\in = \{0,1\}$
Mailbox structure	If not other documented, Message Objects shall use the as max defined Payload size of 64 Byte per Message Object.
<u>If supported only</u> The Message Ram configuration can be changed to support smaller Message Object configurations with a payload size of smaller than 64 Byte.	$\in = \{8,12,16,20,24,32,48,64\}$ By Byte

1.2.4.4 Message Ram organization

The FD Frame format supports up to 64 Byte of payload. This requires a change of the used Classic Message Ram configuration of 8 byte to a storage size of 64 byte to support full FD handling.

Some of the Devices will assign a second Classic Message Object. The Message object is used as Slave Object and allows the storing of the extended payload parts. Other Nodes will rearrange their Message Ram. This results in a new Message Ram organization.

The here used configuration shall support the as maximum supported message configuration.

For Devices where it is possible to adjust the Message Ram organization to smaller settings, a special group of tests shall be performed.

2 Receive Messages

2.1 Receive into single message buffer

2.1.1 Receive into single message buffer (1 – x)

2.1.1.1 Base Format ID / polling

Test Description:

Check that a receive message buffer changes register values linked to the tested message buffer exclusively. All frames have to be received without any errors. The receive interrupt is disabled. This test has to be performed for all possible receive message buffers. This test has to be performed with all possible DLC configurations.

2.1.1.2 Base Format ID / Interrupt

Test Description:

Check that a receive message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be received without any errors. The receive interrupt generation is checked also.

This test has to be performed for all possible receive message buffers.

2.1.1.3 Extended Format ID / Polling

Test Description:

Check that a receive message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be received without any errors. The receive interrupt is disabled.

This test has to be performed for all possible receive message buffers.

2.1.1.4 Extended Format ID / Interrupt

Test Description:

Check that a receive message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be received without any errors. The receive interrupt generation is checked also.

This test has to be performed for all possible receive message buffers.

2.1.2 Overrun handling (1 – x)

2.1.2.1 Overrun handling (1 – x); Polling

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overrun handling. Only the first received frame has to be stored in the message buffer.

This test has to be performed for all possible receive message buffers.

This test can only be performed, if the DUT supports Overrun handling.

Buffer configurations in Fifo Mode shall not be used.

2.1.2.2 Overrun handling (1 – x);Interrupt

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overrun handling. Only the first received frame has to be stored in the message buffer.

- This test has to be performed for all possible receive message buffers.
- This test can only be performed, if the DUT supports Overrun handling.
- This test can only be performed, if the DUT supports Overrun Interrupt handling.
  Comment: Interrupt handling must not

Buffer configurations in Fifo Mode shall not be used.

2.1.2.3 Overrun handling (1 – x), remote flag set / Polling

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overrun handling. Only the first received frame has to be stored in the message buffer.

This test has to be performed for all possible receive message buffers.

This test can only be performed, if the DUT supports Overrun handling.

Buffer configurations in Fifo Mode shall not be used.

2.1.2.4 Overrun handling (1 – x), remote flag set / Interrupt

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overrun handling. Only the first received frame has to be stored in the message buffer.

This test has to be performed for all possible receive message buffers.

This test can only be performed, if the DUT supports Overrun handling.

Buffer configurations in Fifo Mode shall not be used.

2.1.3 Overwrite handling (1 – x)

2.1.3.1 Overwrite handling (1 – x) / Polling

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overwrite handling. The last received frame has to be stored in the message buffer.

This test has to be performed for all possible receive message buffers.

Buffer configurations in Fifo Mode shall not be used.

2.1.3.2 Overwrite handling (1 – x) / Interrupt

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overwrite handling. The last received frame has to be stored in the message buffer. The overwrite interrupt generation is checked also.

This test has to be performed for all possible receive message buffers.

Buffer configurations in Fifo Mode shall not be used.

2.1.3.3 Overwrite handling (1 – x), remote flag set / Polling

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overwrite handling. The last received frame has to be stored in the message buffer.

This test has to be performed for all possible receive message buffers.

Buffer configurations in Fifo Mode shall not be used.

2.1.3.4 Overwrite handling (1 – x), remote flag set / Interrupt

Test Description:

Check that a receive message buffer which receives more than one frame without reading out the message buffer performs an overwrite handling. The last received frame has to be stored in the message buffer. The overwrite interrupt generation is checked also.

This test has to be performed for all possible receive message buffers.

Buffer configurations in Fifo Mode shall not be used.

2.2 Receive into multiple message buffers

2.2.1 Receive Buffer / Buffer Order

2.2.1.1 Receive into multiple message buffers – even message buffers

Test Description:

Checks that several received frames are stored in the correct message buffer order when more than one message buffer is configured for reception.

Applicable for single message buffers only.

2.2.1.2 Receive into multiple message buffers – odd message buffers

Test Description:

Checks that several received frames are stored in the correct message buffer order when more than one message buffer is configured for reception.

Applicable for single message buffers only.

2.2.1.3 Receive into multiple message buffers – all message buffers

Test Description:

Checks that several received frames are stored in the correct message buffer order when more than one message buffer is configured for reception.

Applicable for single message buffers only.

2.2.2 Receive Buffer / Single FIFO message buffer

2.2.2.1 Base Format ID / Polling

Test Description:

Check that the FIFO message buffer changes register values linked to the tested message buffer exclusively. All frames have to be received without any errors. The receive interrupt is disabled. The test has to be performed with all possible DLC configurations.

2.2.2.2 Base Format ID / Interrupt

Test Description:

Check that the FIFO message buffer changes register values linked to the tested message buffer exclusively. All frames have to be received without any errors. The receive interrupt is enabled. The test has to be performed with all possible DLC configurations.

2.2.2.3 Extended Format ID / Polling

Test Description:

Check that the FIFO message buffer changes register values linked to the tested message buffer exclusively. All frames have to be received without any errors. The receive interrupt is disabled. The test has to be performed with all possible DLC configurations.

2.2.2.4 Extended Format ID / Interrupt

Test Description:

Check that the FIFO message buffer changes register values linked to the tested message buffer exclusively. All frames have to be received without any errors. The receive interrupt is enabled. The test has to be performed with all possible DLC configurations.

2.3 Message Buffer Filter

2.3.1 Message filtering / standard ID (1 – x)

Test Description:

The mask is build depending on the mask-architecture of the IUT. If there are multiple mask-architectures possible the test has to be executed for every mask-architecture.

Checks that a receive message buffer linked to the mask(s) only accept frames matching the mask criteria.

Concept:

- The IUT updates the message object and their filter configuration.
- After this, the IUT transmits as part of a synchronization frame the message with the ID 0x444.

The Test System starts to transmit two frames. First frame with

ID = 0x000 and a second frame with the matching frame id.

At the end of the sequence, the IUT starts to transmit the synchronization frame.

2.3.2 Message filtering / extended ID (1 – x)

Test Description:

The mask is build depending on the mask-architecture of the IUT. If there are multiple mask-architectures possible the test has to be executed for every mask-architecture.

Checks that a receive message buffer linked to the mask(s) only accept frames matching the mask criteria.

Concept: The IUT transmits before a sequence starts a synchronization frame with the ID 0x00000444. After this the IUT has finished the mask update. The Test System starts to transmit two frames. First frame with

ID = 0x0000000 and a second frame with the matching frame id.

At the end of the sequence, the IUT starts to transmit the synchronization frame.

2.5 Timestamp tests for receive message buffers

2.5.1 Timestamp for a single receive message buffer on SOF (1 – x)

Test Description:

Check that a timestamp for a receive message buffer is updated on reception of a valid frame. The timestamp value must be captured at the start of frame bit (SOF) of the received frame.

This test has to be executed for every possible message buffer.

The timestamp shall be configured to use Bit count.

2.5.2 Timestamp for a single receive message buffer on EOF (1 – x)

Test Description:

Check that a timestamp for a receive message buffer is updated on reception of a valid frame. The timestamp value must be captured at the end of frame bit (EOF) of the received frame.

This test has to be executed for every possible message buffer.

2.6 Message Acceptance Filter Combination

2.6.1 Message Acceptance Filter Combination (1 – x)

Test Description:

This test must be configured in a custom way, because there are different possible constructions.

- An IUT has several global masks. (Can be handled, is local mask, see chapter 2.3)
- An IUT can split the mask into a low and high mask.
- An IUT has local and global masks. For this both masks must be tested as separate masks. After this, a second test verifies the combination of local and global masks.
- Different filter mask can be assigned to one message object.

3 Transmit Messages

3.1 Transmit from single message buffer

3.1.1 Transmit from single message buffer / standard identifier (1 – x) / Polling

Test Description:

Check that a transmit message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be transmitted without any errors. The transmit interrupt is disabled.

This test has to be performed for all possible transmit message buffers.

3.1.2 Transmit from single message buffer / standard identifier (1 – x) / Interrupt

Test Description:

Check that a transmit message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be transmitted without any errors. The transmit interrupt is enabled.

This test has to be performed for all possible transmit message buffers.

3.1.3 Transmit from single message buffer / extended identifier (1 – x) / Polling

Test Description:

Check that a transmit message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be transmitted without any errors. The transmit interrupt is disabled.

This test has to be performed for all possible transmit message buffers.

3.1.4 Transmit from single message buffer / extended identifier (1 – x) / Interrupt

Test Description:

Check that a transmit message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be transmitted without any errors. The transmit interrupt is enabled.

This test has to be performed for all possible transmit message buffers.

3.2 Transmit from multiple message buffers

Requirement on how to setup:

- The IUT uses an internal arbitrary order of how to handle the transmission requests of each message buffer.
- Each Message Buffer shall be configured with a different message IDs in increased order. (see picture below).
- The transmit requests for all Messages Buffers shall be set at the same time. This allows the internal arbitrary system to select the first configured message (lowest message id) to be send as first.

  If this is not applicable, because of each message buffer must be triggered individual. Set the transmit request flag as fast as possible.

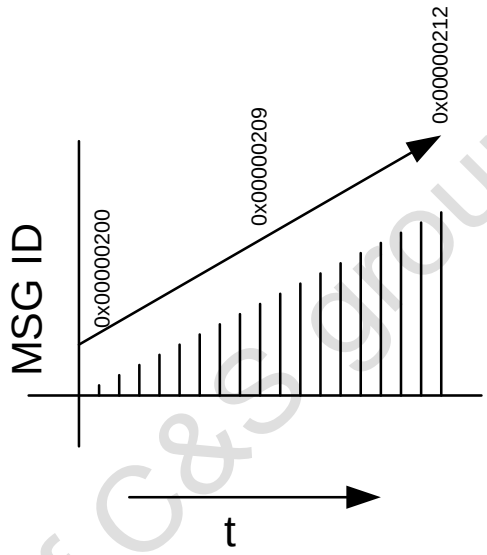


Figure 1 Expected transmission order

3.2.1 Transmit message buffer order

3.2.1.1 Message buffer priority

Test Description:

Check that multiple transmit requests on different message buffers are executed in the correct order.

One test has to be performed with transmit order given by message buffer number.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.1.2 Identifier priority

Test Description:

Check that a multiple transmit requests from different message buffers are executed in the correct order.

One test has to be performed with transmit order given by identifier

The test is applicable, if the device supports the functionality of Message Buffer ID priority.

3.2.2 Transmit message buffer order and arbitration lost

3.2.2.1 Message Buffer Order

Test Description:

Check that multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any loss of arbitration during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.2.2 ID Priority Order

Test Description:

Check that multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any loss of arbitration during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable, if the device supports the functionality of Message Buffer ID priority.

3.2.3 Transmit message buffer order and bus - errors

3.2.3.1 Message Buffer order

Test Description:

Check that a multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any bus error occurrence during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.3.2 Id Priority Order

Test Description:

Check that a multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any bus error occurrence during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable if the device supports the functionality of Message Buffer ID priority.

3.2.4 Transmit message Buffer order within FIFO Buffered Mode

3.2.4.1 Transmit message Buffer order

Test Description:

Check that multiple transmit requests on different message buffers are executed in the correct order.

One test has to be performed with transmit order given by message buffer number.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.4.2 Transmit message buffer order and arbitration lost

Test Description:

Check that multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any loss of arbitration during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.4.3 Transmit message buffer order and bus - errors

Test Description:

Check that a multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any bus error occurrence during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.5 Transmit message Buffer order within Queue Buffered Mode

3.2.5.1 Transmit message Buffer order

Test Description:

Check that multiple transmit requests on different message buffers are executed in the correct order.

One test has to be performed with transmit order given by message buffer number.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.5.2 Transmit message buffer order and arbitration lost

Test Description:

Check that multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any loss of arbitration during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.2.5.3 Transmit message buffer order and bus - errors

Test Description:

Check that a multiple transmit requests from different message buffers are executed in the correct order.

The transmit order must be independent of any bus error occurrence during the test sequence

One test has to be performed with transmit order given by message buffer no.

The test is applicable, if the device supports the functionality of Message Buffer priority.

3.3 Stop transmission

3.3.1 Abort transmission (1 – x)

Test Description:

Check that a message buffer which is transmitting a frame never interrupts the transmission when an abort is requested until the transmit sequence is completed.

This test has to be performed for all possible transmit message buffers.

3.3.2 Abort transmission during arbitration lost (1 – x)

Test Description:

Check that a message buffer which is transmitting a frame never interrupts the transmission when an abort is requested until the transmit sequence is completed. The transmit sequence is completed after arbitration is lost.

This test has to be performed for all possible transmit message buffers.

3.3.3 Abort transmission during bus - errors (1 – x)

Test Description:

Check that a message buffer which is transmitting a frame never interrupts the transmission when an abort is requested until the transmit sequence is completed. The transmit sequence is completed after an error frame has been sent.

This test has to be performed for all possible transmit message buffers.

3.4 Timestamp tests for transmit message buffers

3.4.1 Timestamp for a single transmit message buffer on SOF (1 – x)

Test Description:

This Test verifies the behaviour of the transmit time stamp, measured at SOF.

The application must be implemented to use 2 Mailboxes. This is necessary to get the fastest performance of the IUT's CAN core.

This test has to be performed for all possible transmit message buffers.

3.4.2 Timestamp for a single transmit message buffer on EOF (1 – x)

Test Description:

This Test verifies the behaviour of the transmit timestamp, measured at EOF.

The application must be implemented to use 2 Mailboxes. This is necessary to get the fastest performance of the IUT's CAN core.

This test has to be performed for all possible transmit message buffers.

4 Remote Message Handling

4.1 Remote frame reception

4.1.1 Remote frame reception / standard identifier

4.1.1.1 Remote frame reception / standard identifier (1 – x) / Polling

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be received without any errors.

This test has to be performed for all possible receive message buffers.

4.1.1.2 Remote frame reception / standard identifier (1 – x) / Interrupt

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

The generation of a receive Interrupt is checked.

All frames have to be received without any errors.

This test has to be performed for all possible receive message buffers

4.1.1.3 Remote frame reception / standard identifier / Polling FIFO Mode

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be received without any errors.

This test has to be performed for all possible buffers in FIFO Mode.

4.1.1.4 Remote frame reception / standard identifier / Interrupt FIFO Mode

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

The generation of a receive Interrupt is checked.

All frames have to be received without any errors.

This test has to be performed for all possible buffers in FIFO Mode.

4.1.2 Remote frame reception / extended identifier

4.1.2.1 Remote frame reception / extended identifier (1 – x) / Polling

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be received without any errors.

This test has to be performed for all possible receive message buffers.

4.1.2.2 Remote frame reception / extended identifier (1 – x) / Interrupt

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

The generation of a receive Interrupt is checked.

All frames have to be received without any errors.

This test has to be performed for all possible receive message buffers.

4.1.2.3 Remote frame reception / extended identifier / Polling FIFO Mode

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be received without any errors.

This test has to be performed for all possible buffers in FIFO Mode.

4.1.2.4 Remote frame reception / extended identifier / Interrupt FIFO Mode

Test Description:

The purpose of this test is to verify that a receive message buffer changes register values linked to the tested message buffer exclusively.

The generation of a receive Interrupt is checked.

All frames have to be received without any errors.

This test has to be performed for all possible buffers in FIFO Mode.

4.2 Remote frame transmission

4.2.1 Remote frame transmission / standard identifier

4.2.1.1 Remote frame transmission / standard identifier (1 – x) / Polling

Test Description:

The purpose of this test is to verify that a transmit message buffer changes register values linked to the tested message buffer exclusively.

The transmit interrupt generation is checked also. All frames have to be transmitted without any errors.

This test has to be performed for all possible transmit message buffers.

4.2.1.2 Remote frame transmission / standard identifier (1 – x) / Interrupt

Test Description:

The purpose of this test is to verify that a transmit message buffer changes register values linked to the tested message buffer exclusively.

The transmit interrupt generation is checked also. All frames have to be transmitted without any errors.

This test has to be performed for all possible transmit message buffers.

4.2.2 Remote frame transmission / extended identifier

4.2.2.1 Remote frame from single message buffer / extended identifier (1 – x) / Polling

Test Description:

The purpose of this test is to verify that a transmit message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be transmitted without any errors. The transmit interrupt is disabled.

This test has to be performed for all possible transmit message buffers.

4.2.2.2 Remote Frame from single message buffer / extended identifier (1 – x) / Interrupt

Test Description:

The purpose of this test is to verify that a transmit message buffer changes register values linked to the tested message buffer exclusively.

All frames have to be transmitted without any errors. The transmit interrupt is enabled.

This test has to be performed for all possible transmit message buffers.

4.3 Auto Answer Mode

4.3.1 Automatic remote frame reply / standard identifier (1 – x)

Test Description:

The purpose of this test is to verify that a message buffer automatically answers the reception of a remote frame without setting the transmission start bit by software.

All frames have to be transmitted and received without any errors.

This test has to be performed for all possible transmit message buffers.

4.3.2 Automatic remote frame reply / extended identifier (1 – x)

Test Description:

The purpose of this test is to verify that a message buffer automatically answers the reception of a remote frame without setting the transmission start bit by software.

All frames have to be transmitted and received without any errors.

This test has to be performed for all possible transmit message buffers.

4.4 Remote frame handling with a single message buffer

4.4.1 Remote frame transmission and reception in the same message buffer / std. id. (1 – x)

Test Description:

The purpose of this test is to verify that a Remote frame transmission and reply can be handled with only on message buffer. All frames have to be transmitted and received without any errors.

This test has to be performed for all possible transmit message buffers.

4.4.2 Remote frame transmission and reception in the same message buffer / ext. id. (1 – x)

Test Description:

The purpose of this test is to verify that a Remote frame transmission and reply can be handled with only on message buffer. All frames have to be transmitted and received without any errors.

This test has to be performed for all possible transmit message buffers.

5 Error signalling

5.1 Error signalling during reception

5.1.1 Signalling 'Form Error'

Test Description:

Check that a CAN node, which is receiving a frame with formal errors detecting the FORM Error and write it into the LEC Field. There are three test case to perform

5.1.2 Signalling 'CRC Error'

Test Description:

Check that a CAN node, which is receiving a frame with a CRC error detecting the CRC Error and write it into the LEC Field.

5.1.3 Signalling 'Stuff Error'

Test Description:

Check that a CAN node, which is receiving a frame with a Stuff error detecting the Stuff Error and write it into the LEC Field.

5.2 Error signalling during transmission

5.2.1 Signalling 'Bit Error'

Test Description:

Check that a CAN node, which is transmitting a frame with a Bit0 error detecting the Bit0 Error and write it into the LEC Field.

5.2.2 Signalling 'Bit Error', arbitration field

Test Description:

Check that a CAN node, which is transmitting a frame with a Bit1 error detecting the Bit1 Error and write it into the LEC Field.

5.2.3 Signalling ‘Acknowledge Error’

Test Description:

Check that a CAN node, which is transmitting a frame with an ACK error detecting the ACK Error and write it into the LEC Field.

6 Controller state signalling

6.1 Status change due to REC

6.1.1 Transition from ‘Error Active’ over ‘Error Warning’ to ‘Error Passive’

Test Description:

The purpose of this test is to verify that a CAN node which is receiving a frame with more than one bit error changes its status to warning and then to error passive. For each status change an interrupt has to be generated (if implemented). The REC value must equal 96 at warning level and 128 at error passive state.

6.1.2 Transition from ‘Error Passive’ to ‘Error Active’

Test Description:

The purpose of this test is to verify that a CAN node which is in Error Passive state recovers to Error Active when REC becomes ≤ 127.

6.2 Status change due to TEC

6.2.1 Transition from ‘Error Active’ over ‘Error Warning’ to ‘Error Passive’

Test Description:

The purpose of this test is to verify that a CAN node which is transmitting a frame with more than one bit error changes its status to warning and then to error passive. For each status change an interrupt has to be generated (if implemented). The TEC value must equal 96 at warning level and 128 at error passive state.

6.2.2 Transition from ‘Error Passive’ to ‘Error Active’

Test Description:

The purpose of this test is to verify that a CAN node which is in Error Passive state recovers to Error Active when TEC becomes ≤ 127.

6.3 Bus Off state

6.3.1 Entering ‘Bus Off’ state and recovery sequence

Test Description:

The purpose of this test is to verify that a CAN node which is transmitting a frame with more than one bit error changes its status to Bus Off state. For the status change an interrupt has to be generated (if implemented). The TEC value must equal 256 at Bus Off state.

7 CAN Power Mode Tests

7.1 Sleep mode Tests

Sleep Mode is the behaviour of the DUT to go into a low Power State. The DUT shall switch into the Low Power State without disturbing the CAN Bus Communication in case of normal communication or errors which are detected on the CAN Bus.

In case of the DUT cannot reached this state in a direct way, the test can be set as not applicable. The DUT must be set to Halt Mode and after this, the Environment allows the Switch into Low Power State.

7.1.1 Entering sleep mode during bus idle

7.1.1.1 Entering sleep mode by setting sleep request during bus idle

Test Description:

Check that a CAN node in idle state which is requested for sleep mode enters this state immediately.

7.1.2 Entering sleep mode during reception

7.1.2.1 Entering sleep mode during reception

Test Description:

Check that a CAN node in idle state which is requested for sleep mode enters this state only if the reception sequence is finished.

7.1.2.2 Entering sleep mode during reception and bus errors

Test Description:

Check that a CAN node in idle state which is requested for sleep mode enters this state only if the reception sequence is finished.

7.1.3 Entering sleep mode during transmission

7.1.3.1 Entering sleep mode during transmission

Test Description:

Check that a CAN node which is requested for Sleep mode enters this state only if the transmission sequence is finished.

7.1.3.2 Entering sleep mode during transmission and arbitration lost

Test Description:

Check that a CAN node which is requested for Sleep mode enters this state only if the transmission and reception sequence is finished.

7.1.3.3 Entering sleep mode during transmission and bus errors

Test Description:

Check that a CAN node which is requested for Sleep mode enters this state only if the error handling sequence is finished.

7.1.4 Leaving sleep mode

7.1.4.1 Leaving sleep mode by resetting sleep request

Test Description:

Check that a CAN node in Sleep mode which is requested for leaving Sleep mode enters normal mode immediately.

7.1.4.2 Leaving sleep mode by auto wake up mode

Test Description:

Check that a CAN node in Sleep mode enters normal mode after detecting a dominant bus value and checking 11 recessive bits on the bus.

7.2 Halt/Init mode tests

Halt Mode is the behaviour of the DUT to stop the CAN Core and setting the CAN Core to Init or Halt State. The DUT shall switch into the Halt State without disturbing the CAN Bus Communication in case of normal communication or errors which are detected on the CAN Bus.

7.2.1 Entering Halt/Init mode during bus idle

7.2.1.1 Entering Halt/Init mode by setting Halt/Init mode request during bus idle

Test Description:

Check that a CAN node in idle state which is requested for Halt mode enters this state immediately.

7.2.2 Entering Halt/Init mode during reception

7.2.2.1 Entering Halt/Init mode during reception

Test Description:

Check that a CAN node which is requested for Halt mode enters this state only if the reception sequence is finished.

7.2.2.2 Entering Halt/Init mode during reception and bus errors

Test Description:

Check that a CAN node which is requested for Halt mode enters this state only if the reception sequence is finished.

7.2.3 Entering Halt/Init mode during transmission

7.2.3.1 Entering Halt/Init mode during transmission

Test Description:

Check that a CAN node which is requested for Halt mode enters this state only if the transmission sequence is finished.

7.2.3.2 Entering Halt/Init mode during transmission and arbitration lost

Test Description:

Check that a CAN node which is requested for Halt mode enters this state only if the transmission and reception sequence is finished

7.2.3.3 Entering Halt/Init mode during transmission and bus errors

Test Description:

Check that a CAN node which is requested for Halt mode enters this state only if the error handling sequence is finished.

7.2.4 Leaving Halt/Init Mode

7.2.4.1 Leaving Halt/Init mode by resetting Halt/Init mode request

Test Description:

Check that a CAN node in Halt mode which is requested for leaving Halt mode enters normal mode immediately.

8 Miscellaneous

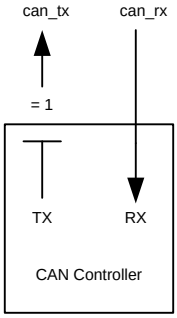
The in the following chapter listed test cases can be performed, if the related functionality is supported by the device.

8.1 Test mode tests

8.1.1 Listen mode test

Test Description:

Check that a receive message buffer changes register values linked to the tested buffer exclusively, during the CAN node is in Listen Mode.



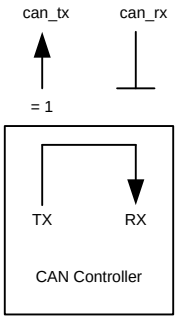
Listen Only Mode

8.2 Loop back mode tests

8.2.1 Loop back mode (Internal)

Test Description:

Check that a CAN node which is entered to Loop Back Mode (internal) is able to receive and transmit frames. The CAN node shall not send to or receive messages from extern.



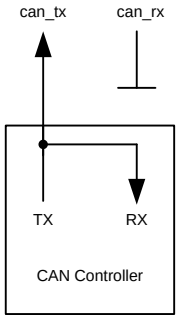
Internal Loop Back Mode

8.2.2 Loop back mode (External)

Test Description:

Check that a CAN node which is entered to Loop Back Mode (external) is not able to receive messages from extern.

Check that a CAN node which is entered to Loop Back Mode (external) is able to transmit messages to the CAN Bus.



External Loop Back Mode

8.3 Error counter tests

8.3.1 Write error counters

8.3.1.1 Write REC Counter

Test Description: (applicable, if the feature is supported)

Check that the CAN Node is able to write the REC Error counter.

8.3.1.2 Write TEC

Test Description: (applicable, if the feature is supported)

Check that the CAN Node is able to write the TEC Error counter.

8.3.2 Reset error counters

Test Description: (applicable, if the feature is supported)

Check that it is able to reset the Error counter.

8.4 Signaling overload frame interrupt

Test Description:

Check that a CAN node generates an Overload Interrupt when LT sends a frame which where the Overload flag is set.

8.5 Disable automatic retransmission for message buffer x

Test Description:

tbd

8.6 Frame counter tests

8.6.1 Count each foreign frame

Test Description:

Check that a CAN node whose frame counter are activated increment the frame counter each time a foreign frame has be received.

8.6.2 Count each valid frame

Test Description:

Check that a CAN node whose frame counter are activated increment the frame counter each time a valid frame has be received.

8.6.3 Count each transmitted frame

Test Description:

Check that a CAN node whose frame counter are activated increment the frame counter each time a a frame has been transmitted successfully by the node.

8.6.4 Frame counter receive overflow without interrupt

Test Description:

Check that a CAN node whose frame counter are activated for increment each time a valid frame has be received and receives frames over the capacity of the frame counter, performs a frame counter receive overflow handling.

8.6.5 Frame counter receive overflow with interrupt

Test Description:

Check that a CAN node whose frame counter are activated for increment each time a valid frame has be received and receives frames over the capacity of the frame counter, performs a frame counter receive overflow handling. The frame counter Interrupt is checked also.

8.7 Auto Bus on

Test Description:

Check that a CAN node which is in Bus Off State is able to immediately set the CAN node to normal state. The CAN node has to keep care about the Bus Off recovery time.

Applicable, if Auto Bus On Control is available.

8.8 Universal counter mode

Applicable, if feature Universal Counter mode is available.

8.8.1 Reload counter by a valid reception of fixed message buffer

Test Description:

Check that a CAN node which is receiving frames and whose universal counter mode is set to Watchdog Mode hold the reload value.

8.8.2 Auto transmit mode

Test Description:

Check that a CAN node who's Universal counter Mode is set on Auto Transmit Mode sends a message from message buffer 11 after reaching the counter value 0x0000. After transmitting the message the Universal counter has to be reloaded on the reload value.

8.8.3 Counter increment due to error frame on the CAN bus

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented if there is an error frame on the CAN bus" increment the counter only if there is an error frame on the bus.

8.8.4 Counter increment due to overload frame on the CAN bus

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented if there is an overload frame on the CAN bus" increment the counter only if there is an overload frame on the bus.

8.8.5 Counter increment: arbitration lost during transmission

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented, if there is an arbitration lost during transmission" increment the counter only if there is an arbitration lost during transmission.

8.8.6 Counter increment: Transmission aborted

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented: transmission aborted" increment the counter only if there is an aborted transmission.

8.8.7 Counter increment: Transmission successful

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented: transmission successful" increment the counter if there was a successful transmission.

8.8.8 Counter increment: receive message rejected message without errors but not matching identifier [B]

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented: receive message rejected message without errors but not matching identifier" increment the counter if there was a message which don't match the Identifier/mask criteria.

8.8.9 Counter increment: receive message lost message without errors but not stored cause unread data [C]

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented: receive message lost message without errors but not stored cause unread data" increment the counter if there was a message which is lost cause message buffer contains unread data.

8.8.10 Counter increment: successful reception (rejected/stored) [D]

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented: successful reception" increment the counter if there was a successful reception.

8.8.11 Counter increment: successful reception matching identifier [E]

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented: successful reception and matching identifier" increment the counter if there was a successful reception.

8.8.12 Counter increment: valid message on the CAN bus (transmission/reception) [F]

Test Description:

Check that a CAN node who's Universal counter Mode is set on "Counter is incremented: valid message on the CAN bus" increment the counter if there is a valid message on the CAN bus. This may be a reception or a transmission.

8.9 FIFO Mode

8.9.1 Receive FIFO

8.9.1.1 Blocking Mode

Test Description:

Applicable for Devices with FIFO buffer handling.

The CAN Node shall be able to receive CAN Messages up to the maximum available Message objects.

The Mode <Blocking> is the default mode of a FIFO Buffer implementation.

The test shall check the device handling regarding the FLAGs:

Warning:

Warning shall be flagged if the used FIFO Buffer is filled up to Warning Level.

The Warning Level shall be placed at $\frac{2}{3}$ of the available FIFO_BUFFER_SIZE.

The functionality is applicable if the handling is supported by the Device

BUFFER Full:

The FIFO Buffer FULL flag shall be flagged after reception of FIFO_BUFFER_SIZE Messages.

MSG Lost:

The FIFO Buffer MSG Lost flag shall be flagged if one Message more than the available FIFO_BUFFER_SIZE is received by the IUT.

According to the used Mode <Blocking Mode>, the last stored message must not be overwritten.

8.9.1.2 Overwrite Mode

Test Description:

Applicable for Devices with FIFO buffer handling. The Receive FIFO Buffer Mode can be changed to Overwrite Mode.

The CAN Node shall be able to receive CAN Messages up to the maximum available Message objects.

The test shall check the device handling regarding the FLAGs:

Warning:

Warning shall be flagged if the used FIFO Buffer is filled up to Warning Level.

The Warning Level shall be placed at $\frac{2}{3}$ of the available FIFO_BUFFER_SIZE.

The functionality is applicable if the handling is supported by the Device

BUFFER Full:

The FIFO Buffer FULL flag shall be flagged after reception of FIFO_BUFFER_SIZE Messages.

MSG Lost:

The FIFO Buffer MSG Lost flag shall be flagged if one Message more than the available FIFO_BUFFER_SIZE is received by the IUT.

According to the used Mode <Overwrite Mode>, the last stored message shall be overwritten by the as last received message.

8.9.2 Event FIFO

8.9.2.1 Event FIFO Flags

Test Description:

Applicable for Devices with FIFO buffer handling.

The CAN Node shall be able to transmit CAN Messages.

The Event Buffer shall log the transmitted TX Events.

The Event FIFO Buffer shall flag the Buffer states Watermark, Buffer FULL and Message LOST after transmitting the appropriate frames.

8.10 Timestamp

8.10.1 Timestamp Overflow Flag

Test Description:

Verify the Flags in case of the Timestamp

8.10.2 FIFO Buffer Watchdog Timeout guard (MCAN)

Test Description:

Observe the Fifo Buffer Flags, if a buffer timeout is flagged.

There should be no timeout flagged during the watchdog time.

9 FD Frame Format

Test in FD Mode with the new Features of CAN FD requires a special Application designed to fulfill the rules of CAN FD.

E.g. All Message Objects, data structures shall be able to handle up to 64 Byte per Mailbox.

The implementation of such tests is not yet implemented.

9.1 FD Receive Messages

Test Description:

Send of Messages in CAN FD Frame Format with IDE $= \in \{0,1\}$, DLC $= \in \{0 \dots 8\}$, DLC $= \in \{9 \dots 15\}$, BRS $= \in \{0,1\}$; $FDF = \in \{0,1\}$

9.2 FD Transmit Messages

Test Description:

Transmit Messages in CAN FD Frame Format with IDE $= \in \{0,1\}$, DLC $= \in \{0 \dots 8\}$, DLC $= \in \{9 \dots 15\}$, BRS $= \in \{0,1\}$; $FDF = \in \{0,1\}$

9.3 FD Time Stamp

Test Description:

A timestamp could be counted as part of the unit BIT.

9.4 FD Error Signaling

Test Description:

Stuff Count, CRC Error, …

9.5 FD Controller State Signaling

Test Description:

TEC / REC Error Counting based on CAN FD Frames.

9.6 FD Miscellaneous

Test Description:

The Miscellaneous test group contains tests which are not available on all devices or did not match to one of the other groups.