BS ISO 17458-3:2013

BSI Standards Publication

Road vehicles — FlexRay communications system

Part 3: Data link layer conformance test specification

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of ISO 17458-3:2013.

The UK participation in its preparation was entrusted to Technical Committee AUE/16, Electrical and electronic equipment.

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.

ISBN 978 0 580 76714 2

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 31 January 2013.

Amendments issued since publication

Date	Text affected

BS ISO 17458-3:2013

INTERNATIONAL STANDARD

ISO 17458-3

First edition
2013-02-01

Road vehicles— FlexRay communications system —

Part 3: Data link layer conformance test specification

Véhicules routiers — Système de communications FlexRay —

Partie 3: Spécification d'essai de conformité de la couche de liaison de données

Reference number
ISO 17458-3:2013(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword ... **iv**

Introduction ... **v**

1 Scope ... **1**

2 Normative references ... **1**

3 Terms, definitions, symbols and abbreviated terms ... **1**
3.1 Terms and definitions ... **1**
3.2 Symbols ... **1**
3.3 Abbreviated terms ... **2**
3.4 Functions ... **4**

4 Conventions ... **4**

5 Document overview ... **5**

6 General ... **6**
6.1 Test architecture ... **6**
6.2 Test implementation ... **7**
6.3 Internal RxDelay ... **9**
6.4 Analog delays ... **9**
6.5 Accepted deviations ... **10**
6.6 Testability requirements ... **11**
6.7 Test execution ... **14**

7 Conformance test cases ... **16**
7.1 General statements and test case structure ... **16**
7.2 Receive data ... **23**
7.3 CHI ... **125**
7.4 Clock synchronisation ... **440**
7.5 Wakeup ... **530**
7.6 Startup ... **568**
7.7 Miscellaneous ... **672**
7.8 Optional TT-E feature ... **820**
7.9 Preambles ... **846**

8 Configuration ... **854**
8.1 Basic configuration ... **854**
8.2 Standard modifications ... **859**

9 Static test cases ... **861**
9.1 Electrical interface ... **861**
9.2 Protocol parameter ... **861**

Annex A (normative) **Technical data for the electrical interface of a FlexRay Communication Controller V3.0** ... **862**

Annex B (normative) **Technical data for the protocol parameter of a FlexRay Communication Controller V3.0** ... **864**

Bibliography ... **865**

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 17458-3 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 17458 consists of the following parts, under the general title *Road vehicles — FlexRay communications system*:

— *Part 1: General information and use case definition*

— *Part 2: Data link layer specification*

— *Part 3: Data link layer conformance test specification*

— *Part 4: Electrical physical layer specification*

— *Part 5: Electrical physical layer conformance test specification*

Introduction

The FlexRay communications system is an automotive focused high speed network and was developed with several main objectives which were defined beyond the capabilities of established standardized bus systems like CAN and some other proprietary bus systems. Some of the basic characteristics of the FlexRay protocol are synchronous and asynchronous frame transfer, guaranteed frame latency and jitter during synchronous transfer, prioritization of frames during asynchronous transfer, single or multi-master clock synchronization, time synchronization across multiple networks, error detection and signalling, and scalable fault tolerance.

The FlexRay communications system is defined for advanced automotive control applications. It serves as a communication infrastructure for future generation high-speed control applications in vehicles by providing:

— A message exchange service that provides deterministic cycle based message transport;

— Synchronization service that provides a common time base to all nodes;

— Start-up service that provides an autonomous start-up procedure;

— Error management service that provides error handling and error signalling;

— Wakeup service that addresses the power management needs.

Since start of development the automotive industry world wide supported the specification development. The FlexRay communications system has been successfully implemented in production vehicles today.

The ISO 17458 series specifies the use cases, the communication protocol and physical layer requirements of an in-vehicle communication network called "FlexRay communications system".

This part of ISO 17458 has been established in order to define the protocol conformance test case requirements.

To achieve this, it is based on the Open Systems Interconnection (OSI) Basic Reference Model specified in ISO/IEC 7498-1 and ISO/IEC 10731, which structures communication systems into seven layers. When mapped on this model, the protocol and physical layer requirements specified by ISO 17458 are broken into:

— Diagnostic services (layer 7), specified in ISO 14229-1 [14], ISO 14229-4 [16];

— Presentation layer (layer 6), vehicle manufacturer specific;

— Session layer services (layer 5), specified in ISO°14229-2 [15];

— Transport layer services (layer 4), specified in ISO 10681-2 [5];

— Network layer services (layer 3), specified in ISO 10681-2 [5];

— Data link layer (layer 2), specified in ISO 17458-2, ISO 17458-3;

— Physical layer (layer 1), specified in ISO 17458-4, ISO 17458-5;

in accordance with Table 1.

Table 1 — FlexRay communications system specifications applicable to the OSI layers

Applicability	**OSI 7 layers**	ISO 17458 **FlexRay communications system**	**Vehicle manufacturer enhanced diagnostics**
Seven layer according to ISO 7498-1 and ISO/IEC 10731	Application (layer 7)	vehicle manufacturer specific	ISO 14229-1, ISO 14229-4
	Presentation (layer 6)	vehicle manufacturer specific	vehicle manufacturer specific
	Session (layer 5)	vehicle manufacturer specific	ISO 14229-2
	Transport (layer 4)	vehicle manufacturer specific	ISO 10681-2
	Network (layer 3)	vehicle manufacturer specific	
	Data link (layer 2)	ISO 17458-2, ISO 17458-3	
	Physical (layer 1)	ISO 17458-4, ISO 17458-5	

Table 1 shows ISO 17458 Parts 2 – 5 being the common standards for the OSI layers 1 and 2 for the FlexRay communications system and the vehicle manufacturer enhanced diagnostics.

The FlexRay communications system column shows vehicle manufacturer specific definitions for OSI layers 3 – 7.

The vehicle manufacturer enhanced diagnostics column shows application layer services covered by ISO 14229-4 which have been defined in compliance with diagnostic services established in ISO 14229-1, but are not limited to use only with them. ISO 14229-4 is also compatible with most diagnostic services defined in national standards or vehicle manufacturer's specifications. The presentation layer is defined vehicle manufacturer specific. The session layer services are covered by ISO 14229-2. The transport protocol and network layer services are specified in ISO 10681.

INTERNATIONAL STANDARD **ISO 17458-3:2013(E)**

Road vehicles — FlexRay communications system — Part 3: Data link layer conformance test specification

1 Scope

This part of ISO 17458 specifies the FlexRay protocol conformance test. This test verifies the conformance of FlexRay communication controllers with respect to ISO 17458-2.

Some testability requirements are given in 6.2.2.3 and 6.6 and are applicable for FlexRay communication controllers to pass the conformance test

2 Normative references

The following referenced documents are indispensable for the application 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 17458-2, *Road vehicles — FlexRay communications system — Part 2: Data link layer specification*

ISO 17458-4, *Road vehicles — FlexRay communications system — Part 4: Electrical physical layer specification*

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions defined in ISO 17458-2 and ISO 17458-4 apply.

3.2 Symbols

Δ	delta
$\in$	Element, lower-case epsilon
ξ	Xsi
µT	microtick
σT	*gdSampleClockPeriod* (= Sampletick)
t_{RC}	modification of cycle length due to calculated rate correction (equal to *zRateCorrection*, used in figures of "Clock synchronisation")
t_{OC}	modification of cycle length due to calculated offset correction (equal to *zOffsetCorrection*, used in figures of "Clock synchronisation")

φ_{Rx}, φ_{Tx} analogue delays (see 6.4)

ξ, ξ_{IUT} allowed deviation from theoretical results due to LT-IUT jitter (see 6.5)

3.3 Abbreviated terms

BC basic configuration

BD bus driver

BSS byte start sequence

CAS collision avoidance symbol

CC communication controller

CE communication element

CHI controller host interface

CHIRP channel idle recognition point

CRC cyclic redundancy code

DC dual channel

DTS dynamic trailing sequence

FES frame end sequence

FIFO first in first out

FPGA field programmable gate array

FSS frame start sequence

ID identifier

IP intellectual property

IUT implementation under test

LT lower tester

MT macrotick

MTS media access test symbol

NIT network idle time

NM network management

PE protocol engine

POC protocol operation control

RTL register transfer level

RxD	receive data signal from bus driver
SC	single channel
TE	test execution
TSS	transmission start sequence
TT-D	time triggered distributed
TT-E	time triggered external
TxD	transmit data signal from CC
TxEN	transmit data enable not signal from CC
UT	upper tester
WUDOP	wakeup during operation pattern
WUP	wakeup pattern
WUS	wakeup symbol

POC states:

C	*POC:config*
CSCC	*POC:coldstart consistency check*
CSCR	*POC:coldstart collision resolution*
CSG	*POC:coldstart gap*
CSJ	*POC:coldstart join*
CSL	*POC:coldstart listen*
DC	*POC:default config*
H	*POC:halt*
ICC	*POC:integration consistency check*
ICSC	*POC:integration coldstart check*
IL	*POC:integration listen*
IS	*POC:initialize schedule*
NA	*POC:normal active*
NP	*POC:normal passive*
R	*POC:ready*

3.4 Functions

TruncateTowardsZero: function returns the integer part of a number without its fractional digits (= sign(x) * floor(|x|))

4 Conventions

ISO 17458 are based on the conventions specified in the OSI Service Conventions (ISO/IEC 10731) as they apply for physical and data link layer (protocol).

5 Document overview

Figure 1 depicts the FlexRay document reference according to OSI model.

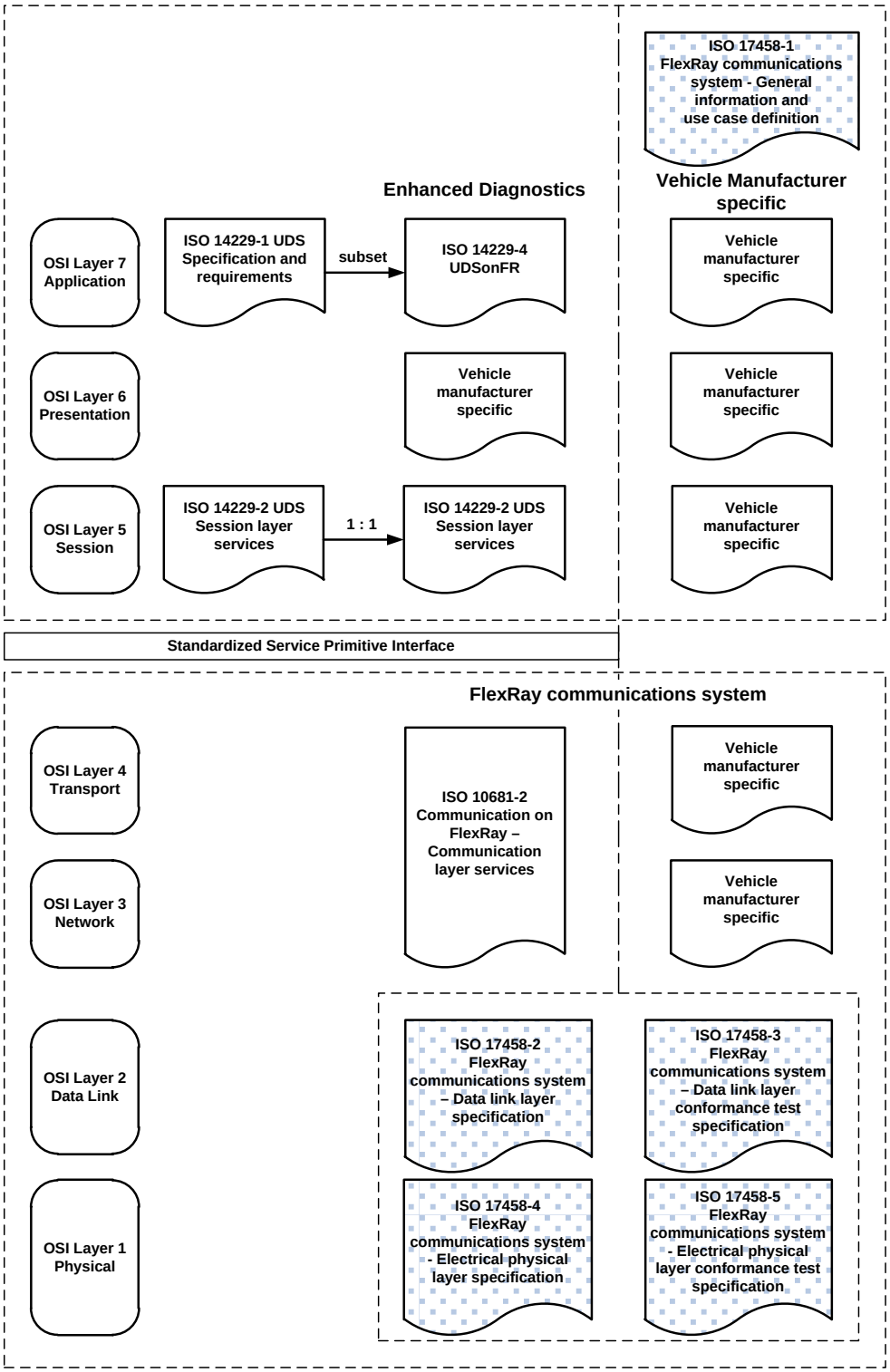


Figure 1 — FlexRay document reference according to OSI model

6 General

6.1 Test architecture

This part of ISO 17458 is based on a test architecture as shown in Figure 2, which follows the ISO 9646 standard. The implementation under test (IUT) is the FlexRay CC. The upper tester (UT) is connected to the FlexRay controller host interface (CHI) of the IUT and the CHI is device specific. The lower tester (LT) is connected to the FlexRay physical layer interface of the IUT and ISO 17458-4 describes this interface. The test coordination procedure controls the UT and the LT.

In a hardware-based test environment (see 6.2.2), the FlexRay CC can be an “embedded” FlexRay CC meaning that CC is embedded in a microcontroller. In this case, the CHI (i.e., the upper tester interface) is between the embedded FlexRay CC and the microcontroller and in order to get access to the CHI, the upper tester may be partly distributed to the microcontroller.

The test architecture shown in Figure 2 is suitable for testing one FlexRay CC.

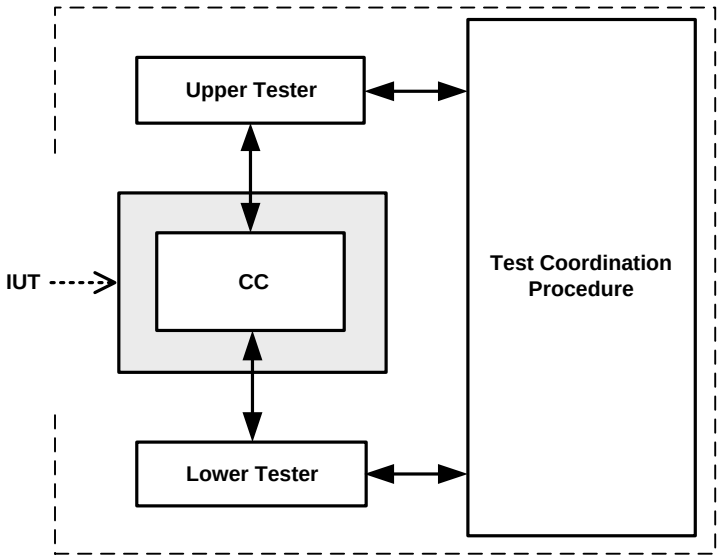


Figure 2 — Standard test architecture

There are some optional test cases in this part of ISO 17458 which tests the optional TT-E feature by using a pair of FlexRay CCs, namely a source CC and a sink CC, which are connected via the time gateway interface. Here, the IUT is the pair of connected FlexRay CCs. To test the TT-E feature, the test architecture as shown in Figure 3 is proposed. The upper tester and lower tester are connected to both FlexRay CCs and the test coordination procedure controls the upper and lower tester.

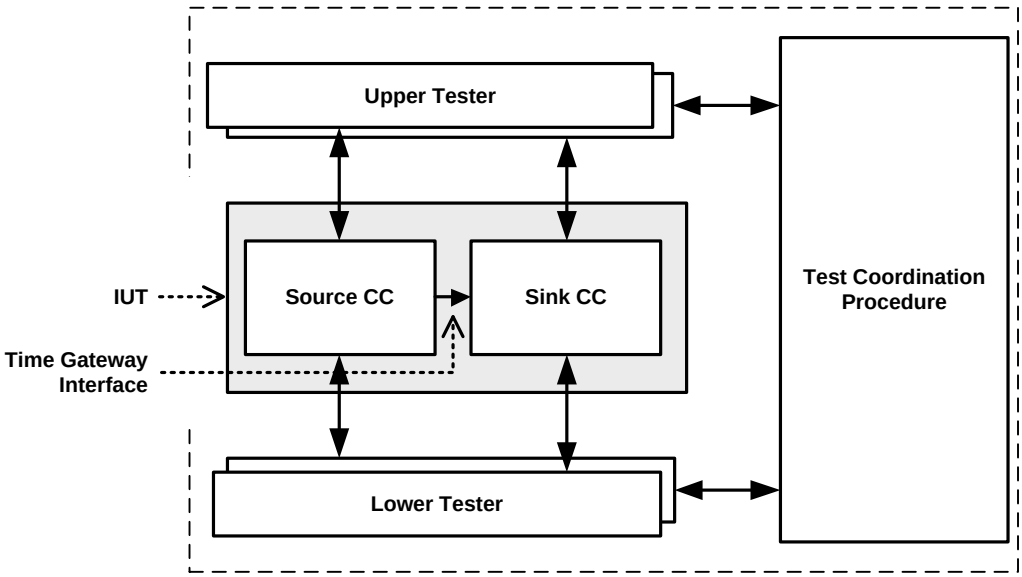


Figure 3 — Test architecture for the TT-E option

6.2 Test implementation

6.2.1 General

The test cases and the proposed test architecture of this specification can be either implemented in a hardware-based environment or in a simulation-based environment. In the following, both environments are described.

6.2.2 Hardware-based test implementation

6.2.2.1 IUT

In a hardware-based test implementation, the IUT is a physical device. The IUT can be either a standalone FlexRay CC, an embedded FlexRay CC, or a FlexRay CC programmed in an FPGA. For testing the optional TT-E feature, the IUT consists of a pair of connected FlexRay CCs.

6.2.2.2 Lower tester

The electrical characteristics of the lower tester shall follow the electrical characteristics of the interface between the FlexRay CC and the FlexRay Bus Driver on the Bus Driver side. This interface is described in ISO 17458-4. The requirements on the lower tester in a hardware-based test environment, including the electrical characteristics, are listed in Table 2. It is not advised to use any extra circuits (e.g., level shifters) between the IUT's FlexRay CC and the lower tester.

Table 2 — Requirements on the lower tester in a hardware-based test environment

Description	Relevant signal	Parameter name used in ISO 17458-4	Min	Max	Unit
Input capacitance	TxD, TxEN	*C_BDTxD*	-	10	pF
Threshold for detecting logical high	TxD, TxEN	*uBDLogic_1*	-	60	%
Threshold for detecting logical low	TxD, TxEN	*uBDLogic_0*	40	-	%
Voltage reference for logical high and low	TxD, TxEN, RxD	*uVDIG*	same as IUT		V
Sample Rate	TxD, TxEN	N/A	160	-	MHz
Asymmetry	RxD	see ISO 17458-4: measured at 50 % *uVDIG* and 25 pF load	-	2	ns
Sum of rise and fall time @ 15 pF load	RxD	*dBDRxDR15* + *dBDRxDF15*	-	13	ns
Difference of rise and fall time @ 15 pF load	RxD	| *dBDRxDR15* – *dBDRxDF15* |	-	5	ns
Sum of rise and fall time @ 25 pF load	RxD	*dBDRxDR25* + *dBDRxDF25*	-	16,5	ns
Difference of rise and fall time @ 25 pF load	RxD	| *dBDRxDR25* – *dBDRxDF25* |	-	5	ns
Frequency of FlexRay clock, provided to IUT	clk	N/A	-	80	MHz
Precision of FlexRay clock, provided to IUT	clk	N/A	-	500	ppm

6.2.2.3 Clock synchronisation

Several test cases require the synchronisation between the IUT and the test environment such that random clock deviations can be excluded and the occurrence time of a sampletick can be determined within one sampletick accuracy.

Therefore it is required that the LT provides a clock signal (called “FlexRay clock”) to the IUT. The LT shall use the FlexRay clock as a basis for FlexRay bus stimuli of the RxD signal and for sampling the TxD and TxEN signals. The IUT shall also use the FlexRay clock for FlexRay transmission and reception. Existing PLLs in the IUT need to be bypassed or programmed not to multiply. If the IUT uses an own clock source or an active PLL, then there might be the risk of failing some test cases.

Some requirements on the FlexRay clock signal are listed in Table 2. In addition, those frequencies shall be supported, which can be derived from 80 MHz by integer division. The FlexRay clock shall run continuously in order to be able to test embedded FlexRay CCs, where the FlexRay clock signal is also used to for clocking the host microcontroller.

6.2.3 Simulation-based test implementation

In a simulation-based test implementation, the IUT is described in a hardware description language typically on register transfer level (RTL), e.g., in SystemVerilog code or VHDL code. The IUT is a FlexRay CC, including message buffers and FIFO. For testing the optional TT-E feature, the IUT consists of a pair of connected FlexRay CCs. The test environment (upper tester, lower tester, and test coordination procedure) and the test cases exist as software only. Executing a test case in a simulation-based test implementation means to load all necessary parts (IUT, test environment, test case) in an RTL simulator and then to run the simulation.

In the simulation-based test implementation, the clocks of the IUT and the LT have no deviation from the nominal frequency and also have no jitter. Therefore, no requirements on the clock synchronisation between the LT and the IUT (as listed in 6.2.2.3 for the hardware-based test implementation) are given for the

simulation-based test implementation. However, ξ_{IUT} and ξ (see 6.5) have to be considered in the simulation-based test implementation.

6.3 Internal RxDelay

The parameter *adInternalRxDelay* is implementation specific and has an allowed range between 1 and 4 sampleticks ISO 17458-4. All basic configurations (see clause 8) assume *adInternalRxDelay* to be 4 sampleticks. In order to compensate between the implementation specific value of *adInternalRxDelay* and the assumed value of 4 sampleticks, a delay compensation shall be integrated in the RxD signal between the LT and the IUT. The delay compensation shall have a delay of 4-dt [sampleticks], and dt is the IUT's actual value of *adInternalRxDelay*. The delay compensation shall be applied in the hardware-based test implementation and in the simulation-based test implementation. Figure 4 gives an example how to implement the delay compensation in a hardware-based test implementation.

Please note that in the test cases, the time interval between IUT's frame and LT's frame are measured at the test points marked in Figure 4. Therefore the delay compensation is not to be considered in test steps like *"It is verified (LT) that the interval between the IUT's frames in slot 1 and LT's frame slot 2 is x µT..."*.

Figure 4 depicts the proposal for compensation of *adInternalRxDelay* in a hardware-based test implementation.

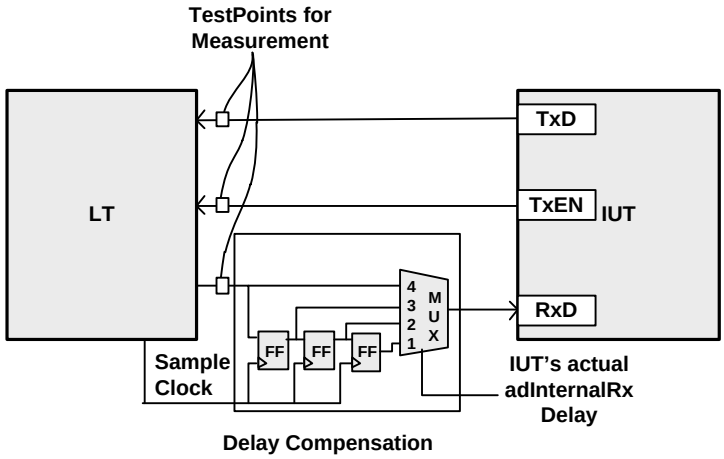


Figure 4 — Proposal for compensation of adInternalRxDelay in a hardware-based test implementation

6.4 Analog delays

Analog delays of the IUT appear in the signal paths between the physical pads of the device and the flip flops of the FlexRay CC. In ISO 17458-4, the analog delays are captured in the following parameters: *dCCRxD01*, *dCCRxD10*, *dCCTxD01*, *dCCTxD10*, *dCCTxEN10* or *dCCTxEN01*. In this conformance test specification, the following two analog delays are defined: φ_{Rx} is the analog delay on the reception path, and φ_{Tx} is the analog delay on the transmission path. In the hardware-based test implementation, the analog delays are determined as follows:

— $\varphi_{Rx} \in [\ 0;\ \max\{dCCRxD01, dCCRxD10\}\ ]$

— $\varphi_{Tx} \in [\ 0;\ \max\{dCCTxD01, dCCTxD10, dCCTxEN10, dCCTxEN01\}\ ]$

Here, the parameters *dCCRxD01*, *dCCRxD10*, *dCCTxD01*, *dCCTxD10*, *dCCTxEN10* or *dCCTxEN01* have the maximal values as defined in ISO 17458-4.

In the simulation-based test environment, there are no physical pads but only the FlexRay CC itself simulated. Therefore, the analog delays can be set to 0 here:

— $\varphi_{Rx} = 0$

— $\varphi_{Tx} = 0$

6.5 Accepted deviations

Despite the synchronisation of the IUT and the test environment described in 6.5 there is a remaining deviation between the LT and the IUT. A sample point at the LT does not necessarily cause a sample point at the IUT at the same moment since the clock signal stemming from the test environment can be delayed within the IUT due to the clock tree. Due to this we have to assume that the IUT's sample point occurs anywhere within two sample points of the LT and could also have a small amount of jitter.

Figure 5 depicts the shift of IUT sampletick relative to LT's signal.

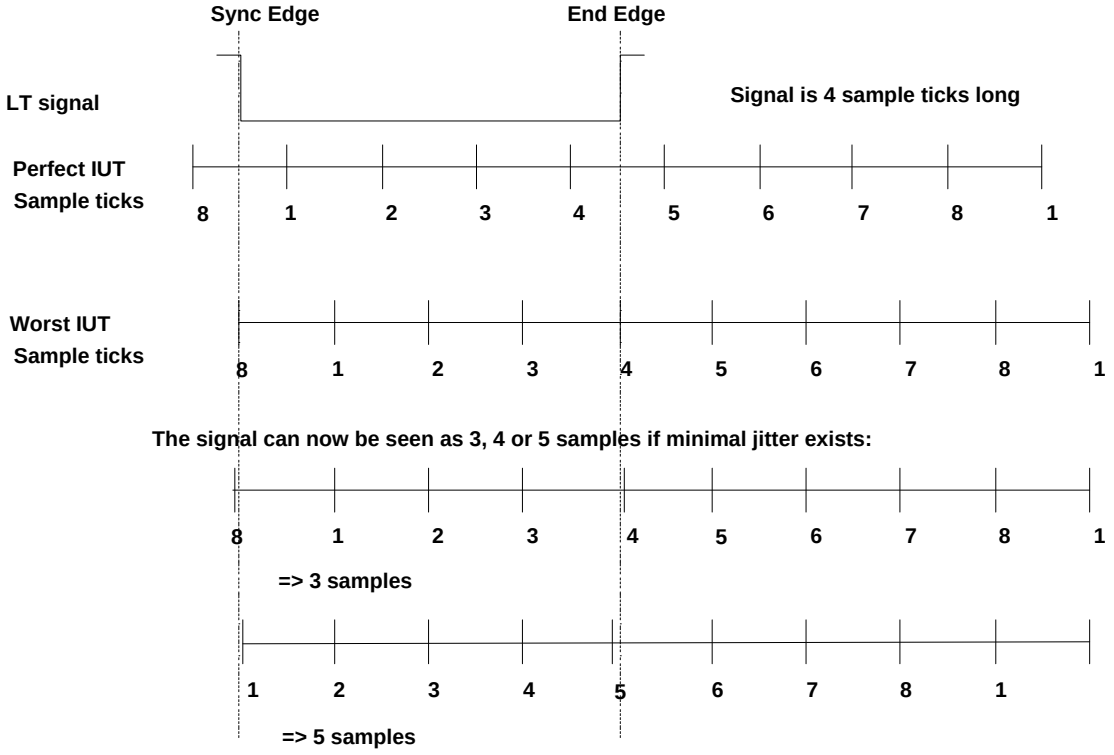


Figure 5 — Shift of IUT sampletick relative to LT's signal

Figure 5 shows that in the worst case this can lead to a situation where the IUT samples directly on the edges of the LT's signal. In that case even a small amount of jitter can cause a 4 sample signal of the LT to be seen as a 5 sample signal (line 3 of Figure 5) or a 3 sample signal (line 4 of Figure 5) by the IUT. This of course has consequences for the precision test cases can achieve and also on the results of the IUT's clock synchronisation algorithm.

The imprecision seen in Figure 5 prevents exact testing of the bit strobing point. Sending a 4 sample long pattern can, as has been shown, either be seen at the bit strobing point (sample point 5) or not. Similarly a 5

sample long pattern could not be seen at the bit strobing point. Thus 4 and 5 sample long LOW patterns can not be used to verify if bit strobing correctly happens at sample 5.

Similar the clock synchronisation mechanism of FlexRay can be disturbed by this sample point deviations and cause a slight de-synchronisation of IUT and test environment. Figure 5 shows that a sync edge which arrives at the same time as a sampletick can either be seen already by this sampletick (line 4) or only by the next one (line 3). If this happens to the FSS/BSS sync edge, the time reference point of the frame shifts (see ISO 17458-2 Figure "Time reference point definitions"). If due to jitter this time reference point shifts into different directions in consecutive cycles, the IUT will erroneously determine the need to correct its macrotick clock rate and phase (=offset). This means that all higher level events (e.g. slot starts, cycle starts, ...) that depend on macroticks can be slightly unaligned between IUT and test environment. This has to be taken into account in the test cases.

To address the deviations introduced by the unaligned FlexRay clock correction mechanism, ξ_{IUT} is introduced, with the allowed range of:

$$\xi_{IUT} \in [-6;5]\mu T$$

ξ_{IUT} expresses the range of deviation that can be introduced in signals coming from the IUT to the LT as well as the status variables related to clock correction sent from the IUT to the UT. The maximum tolerances for IUT signals and clock correction related variables as seen by the LT and UT shall be limited to. The test environment and setup shall minimize those tolerances as much as possible. Please note that ξ_{IUT} is only a valid deviation if the FlexRay clock correction is involved in the measurement in question. For example it is not to be applied to tests of the decoder or to tests within the WAKEUP period.

In addition, the LT introduces a deviation when measuring due to the granularity of measurements as well as some inherent inexactness. Thus every time measurement of the IUT by the LT is assumed to have a possible deviation of:

$$\xi \in [-1;1]\mu T$$

ξ is to be added as tolerance to each traffic measurement at the IUT's interface to the LT.

6.6 Testability requirements

6.6.1 General

Since ISO 17458-2 lacks some requirements which are necessary for defining precise test cases (e.g., CHI delays), this subclause specifies such testability requirements.

NOTE subclause 6.2.2.3 also gives a testability requirement in a hardware-based test implementation. A FlexRay CC, which does not meet the testability requirements, runs the risk of failing the test cases.

6.6.2 CHI delay constraints

ISO 17458-2 states for the controller host interface (CHI) that *"due to implementation constraints the CHI may add product specific delays for data or control signals exchanged between the host and the protocol engine"*. The maximum value of some CHI delays are to be known for testing a FlexRay CC. Therefore, Table 3 specifies the necessary maximum values and the maximum values are based on the time t_0 at which the event / the variable change happens in the protocol engine. It is assumed that if a host requests one of these values from the CHI anywhere between (t_0 + Max. delay) and the earliest visibility time of the next change of the value, the host will receive the correct value.

NOTE the delay values in Table 3 do not include the time required to transfer the data from the CHI to the host. They define the earliest point in time when a host might access the data and is guaranteed to receive the updated value in

response. It is expected that the CHI transfers the requested data immediately with no delay[1)] except that inherent in the CHI-host interface (e.g. due to round trip times, bandwidth restrictions, parallel ongoing transfers, ...). Should the underlying value changes during the transfer (e.g. during the transfer of payload a new frame arrives for the slot) it is the task of the CHI to ensure the atomicity of the transfer.

In order to be able to pass the test cases of this specification, the CHI delays of a FlexRay CC shall follow the maximum values and it is highly recommended that the delays are much below the maximum values.

Table 3 — Maximum delay allowed to occur in the CHI

Parameter as defined in ISO 17458-2	Earliest visible	Maximum delay in CHI	Remarks
CHI commands like RUN, ALL_SLOTS…	NA	2 000 µT after command submission	Thus for testing purposes it can not be assumed that the PE will react according to the new state until the delay has passed.
Protocol operation control status	earliest allowed state change time	500 µT after latest allowed state change time in PE	Several state changes are defined to happen sometime within the NIT. For these earliest and latest allowed state change time differ (typically to beginning and end of NIT).
Wakeup and startup status	at state change	500 µT after state change	---
vMacrotick	at macrotick	½ MT, minimum 20 µT, after macrotick	The repeated consecutive reads are not guaranteed, i.e. reading at macrotick x and then immediately again does not guarantee a result of x or x + 1.
vCycleCounter	at cycle start	500 µT after cycle start	---
vSlotCounter A/B	at slot start or segment start	50 µT after PE slot counter change	---
vInterimRateCorrection vInterimOffsetCorrection	at NIT start	500 µT after end of cycle	---
various error indicators	at event occurrence	sync frame overflow: 500 µT after slot or segment boundary; *pLatestTx*: 500 µT after dynamic segment end; transmission across boundary: 500 µT after slot or segment boundary	---
Synchronisation frame status	start of NIT	10 MT after the start of offset correction phase	for non TT-E coldstart nodes
Startup frame status	1 µT before cycle start	10 MT after start of cycle	for TT-E coldstart nodes only
Startup frame status	start of NIT	10 MT after the start of offset correction phase	---
Symbol window status vSS	at symbol window end	500 µT after symbol window end	---
NIT status vSS	at NIT end (= start of next cycle)	500 µT after start of cycle	---

1) However if there is a transfer delay greater than a given verification window, the access time shall be considered by the UT.

Parameter as defined in ISO 17458-2	Earliest visible	Maximum delay in CHI	Remarks
Aggregated channel status	at each slot and segment end	500 µT after each boundary	This could mean that it is not checkable for empty dynamic slots. But this is not necessary for the conformance test as that only checks this value during the NIT.
Dynamic segment status	at dynamic segment end	500 µT after dynamic segment end	---
Transmit buffer status	at slot end / segment end	2 * *gdStaticSlot* for static slots or 2 000 µT for dynamic slots, counting from the slot boundary at the end of the respective slot	Definition is to keep it in accordance with payload.
Slot status data	at slot end / segment end	2 * *gdStaticSlot* for static slots or 2 000 µT for dynamic slots, counting from the slot boundary at the end of the respective slot	---
Frame contents data	at complete (valid!) frame arrival	2 * *gdStaticSlot* for frames received in the static segment or 2 000 µT for frames received in the dynamic segment, counting from the slot boundary following the valid frame	2 000 µT is provided for the dynamic segment where slots can become as short as 2 MT, the duration of 2 static slots is used to scale the available time according to the payload length used. Counting from the end of the slot means, that e.g. the payload of a valid frame arriving in slot 1 could be accessed the earliest after the end of slot 3.
Queued receive buffers	at slot and segment end on complete arrival of a valid frame	2 000 µT after complete arrival of a valid frame	---
Interrupt service	at event occurrence	100 µT after event occurrence	---
Accrued network management vector	at each update (see ISO 17458-2)	500 µT after each update (see ISO 17458-2)	---

6.6.3 Consequences for interpretation of test steps

The upper limits of the CHI delays of Table 3 are used in the test cases explicitly to determine the earliest point in time a change of a CHI parameter can be checked / verified by the UT and the earliest point in time the LT can expect the IUT to show appropriate behaviour due to commands from the UT.

In several test cases, esp. in the preambles, the UT issues several CHI commands in short sequence, mostly to configure the IUT. In order to reduce the runtime of the test cases, the upper limits of the CHI delays are not explicitly stated here, but it is assumed that the UT takes care of the CHI delays and delays any further commands to the IUT until the first one was executed.

In the beginning of each test case the IUT is reset by the UT. In this situation it is the responsibility of the test environment to ensure that the IUT has completed its reset before further actions are performed.

6.6.4 Special handling of vMacrotick counter

The delay specification for *vMacrotick* given above results in a 20 µT window for reading the counter in case the macrotick length is configured as 40 µT. But additionally the IUT's macrotick can shift around the LT's macrotick by up to ξ_{IUT} µT with $\xi_{IUT} \in [-6;5]$ (for definition of ξ_{IUT} see 6.5). As such the validity window shrinks

by 11 µT, requiring the UT to hit a 9 µT window to read the *vMacrotick* counter correctly in all circumstances, a non-trivial task.

To ease this problem, additional information about the delay and the jitter for accessing the *vMacrotick* counter shall be provided. For example, the additional information could be "the *vMacrotick* counter is valid at the CHI from 5 µT after the IUT's macrotick to 2 µT after the next IUT's macrotick". For a macrotick length of 40 µT this results in a validity window of 37 µT, which, after reduction for ξ_{IUT}, provides the UT with a 26 µT window.

6.7 Test execution

6.7.1 Single channel CC

For a single channel CC all test cases with "SC" applicability and all test cases "SC, DC" applicability shall be executed in two runs: one run with *pChannels* = A and one run with *pChannels* = B. Subclause 7.1 gives more details on the "applicability" label per test case.

Total number of test cases for a single channel CC: 391 (this is the sum of 5 "SC" test cases and 386 "SC, DC" test cases).

6.7.2 Dual channel CC

For a dual channel CC, all test cases with "DC" applicability and all test cases with "SC, DC" applicability shall be executed in one run with *pChannels* = A&B. In addition, all test cases with "SC" applicability and all test cases with "SC, DC" applicability shall be executed in two runs: one run with *pChannels* = A and one run with *pChannels* = B.

Total number of test cases for a dual channel CC: 420 (this is the sum of 5 "SC" test cases, 386 "SC, DC" test cases, and 29 "DC" test cases).

6.7.3 Optional TT-E feature

TT-E stands for time triggered external synchronisation method. To test the optional TT-E feature requires a pair of CCs, which are connected by the time gateway interface. One CC is the time gateway source, the other CC is the time gateway sink. See clause 6 and Figure 3. Each CC can be either a single channel or a dual channel implementation. If a pair of connected CCs claims to support the TT-E option, all test cases with "TT-E" applicability shall be executed.

Total number of test cases for the TT-E option: 12

All other test cases shall be executed for each of the two CCs according to 6.7.1 and 6.7.2.

Depending on whether the source and sink CC is a single channel or a dual channel CC, Table 4 gives the required runs of the TT-E test cases and the corresponding set "$pChannels_{Source,Sink}$"of configurations of the parameter *pChannels* for both CCs. For example, only row 1 in Table 4 has to be considered if both CCs are single channel CCs, only row 4 has to be considered of both CCs are dual channel CCs. Notation used for the set $pChannels_{Source,Sink}$:*"(pChannels for Source:CC, pChannels for Sink:CC)"*. For example *"(A,A&B)"* means that the source CC is set to single channel mode with *pChannels*=A and the sink CC is set to dual channel mode with *pChannels*=A&B.

Table 4 — Configurations of pChannels for both CCs, depending on whether each CC (Source and Sink) is a single channel or a dual channel one

Row	Source CC	Sink CC	Number of runs of the "TT-E" tests and the set of configurations for pChannels
1	single channel	single channel	Four runs with $pChannels_{Source,Sink}$ = {(A,A),(A,B),(B,A),(B,B)}
2	single channel	dual channel	Six runs with $pChannels_{Source,Sink}$ = {(A,A),(A,B),(B,A),(B,B),(A,A&B),(B,A&B)}
3	dual channel	single channel	Six runs with $pChannels_{Source,Sink}$ = {(A,A),(A,B),(B,A),(B,B),(A&B,A),(A&B,B)}
4	dual channel	dual channel	Nine runs with $pChannels_{Source,Sink}$ = {(A,A),(A,B),(B,A),(B,B),(A,A&B),(B,A&B),(A&B,A),(A&B,B),(A&B,A&B)}

NOTE The following statement is valid for both CCs (source and sink CC): If the CC is a dual channel CC and it is set to single channel mode (*pChannels*=A or *pChannels*=B), the not configured channel shall be tested for inactivity. If *pChannels*=A, channel B is "not configured"; if *pChannels*=B, channel A is "not configured."

6.7.4 Requirements on clock synchronisation and wakeup

The following shall be considered in dual channel test execution (i.e., if *pChannels* = A&B):

— All test cases specified in 7.4 shall be executed in three instances: in instance 1 the LT simulates its frames on channel A, in instance 2 the LT simulates its frames on channel B, and in instance 3 the LT simulates its frames on both channels.

— All test cases specified in 7.5 shall be executed twice, first with *pWakeupChannel* = A and second with *pWakeupChannel* = B.

Exceptions to this and additional instructions in dual channel test executions are specified per test case and at the beginning of 7.4 and 7.5.

6.7.5 Basic configurations

All test cases, which are to be executed according to 6.7.1 through 6.7.3, shall be executed with all five basic configurations 1a, 1b, 2a, 2b, and 3. The five basic configurations represent the combination of mandatory bit rates and microtick lengths.

Table 5 holds the simplified five basic configurations and Clause 8 lists the basic configurations in detail with all relevant parameters.

Table 5 — Simplified basic configurations: bit rates and microtick lengths

Basic Configuration	1a	1b	2a	2b	3
Bit Rate [Mbit/s]	10	10	5	5	2,5
pdMicrotick [µs]	0,025	0,0125	0,025	0,05	0,05

6.7.6 Modifications, variants, instances

A test case can have modifications of the basic configurations, variants, and instances (see 7.1 for more details). For each test case, all defined modifications, variants, and instances shall be executed.

7 Conformance test cases

7.1 General statements and test case structure

7.1.1 General statements

All following general statements are valid for all test cases, unless otherwise specified in the individual test case.

— **Single channel test execution:**
A test case is in “single channel test execution” if it is executed with *pChannels*=A or *pChannels*=B. In single channel test execution, the stimulus and all channel dependent verifications shall be applied only for the channel configured by *pChannels*. If the IUT is a dual channel CC, the “not configured” channel shall be tested for inactivity. If *pChannels*=A, channel B is “not configured”; if *pChannels*=B, channel A is “not configured”.

— **Dual channel test execution:**
A test case is in “dual channel test execution” if it is executed with *pChannels*=A&B. In dual channel test execution, the stimulus shall be applied to both channels symmetrically and all channel dependent verifications shall be applied for both channels.
The phrase “For dual channel test execution <instructions>” means that the <instructions> shall be considered if the test case is executed with *pChannels*=A&B. Examples: “For dual channel test execution it is verified (LT) that the non-wakeup channel stays idle throughout the entire test.”, “For dual channel test execution the test has to be performed in three instances.”

— **Respective channel(s)” and “available channel(s):**
The phrases “respective channel(s)” and “available channel(s)” reflect the channel(s) on which the LT applies its stimuli.

— **Reset:**
A statement similar to “The UT resets the IUT” means that the UT resets the IUT and waits at least for the implementation specific delay to allow the IUT finishing the reset process. After reset, the IUT is in POC:default config.

— **Multiple communication elements in a slot:**
If the LT simulates multiple communication elements, e.g. frames and symbols, within a single slot, those communication elements have to be separated by at least 11 * *gdBit* in order to enable the IUT to differentiate them.

— **Default frame content:**
The default frame contents for frames simulated by the LT or transmitted by the IUT are defined as follows unless otherwise specified in the test case or preamble. After the execution of the preamble, the LT continues with the simulation of valid frames in the used slots of the preamble, unless otherwise specified.

Table 6 defines the default frame contents.

Table 6 — Default frame contents

Default frame contents		
Segment	static	dynamic
Reserved bit	0	0
Payload preamble indicator	0	0
Null frame indicator	1	1
Sync frame indicator	0	0
Startup frame indicator	0	0
Payload length	*gPayloadLengthStatic*	1
Payload data on channel A	all bytes set to 0x00	byte 0: 0xCC and byte 1: 0x33
Payload data on channel B	all bytes set to 0x00	byte 0: 0xCC and byte 1: 0x33
Cycle Count	current cycle count	

— **Check / verify:**
Each test case consists of a preamble, a test execution, and a postamble (see 7.2). The phrase “It is checked that ...” is a “check statement” and it is used in the preamble and postamble. The phrase “It is verified that ...” is a “verify statement” and it is used in test execution. If a “check statement” fails, the setup state for the testing part cannot be reached as required. Therefore the test cannot be performed as specified and the test case is not passed. If a “verify statement” fails the test case is failed.

— **Pass criteria:**
The intention of the section “Pass criteria” in the test cases is to summarize the “verify statements” of the test execution. The section does not imply additional “check statements”or “verify statements”.

— **Implicit checks**:
Implicit checks can be optionally performed in all test cases during preamble, test execution and postamble, even if these checks are not specified in the test case. If an implicit check fails, the test case is not prevented from being passed, but it can be treated as a hint that there might be an unintended behaviour within the IUT.
Although the implicit checks are not specified in very detail within this conformance test specification, they need to be implemented conform to the data link layer specification.
Implicit checks include (there can be more implicit checks):

- It is continuously checked (LT) that the IUT transmits nothing (i.e., TxD=high and TxEN=high on any channels supported by the IUT) between reset (as defined in ISO 17458-2) and the first time the UT issues the CHI_RUN command or the CHI_WAKEUP command.

- It is checked (LT) that the IUT transmits no communication elements in not assigned slots.

- If interrupt requests are globally enabled and interrupt requests for (one ore more) specific interrupt source(s) are enabled, then it is checked that whenever the interrupt requests are generated, the interrupt status indication flags for the enabled source(s) are set. This implicit check shall be performed according to ISO 17458-2 subclause "Interrupt service".

— **Not passed or failed test cases:**
Once a test case is not passed or has failed during conformance testing, the test result (failed or not passed) can only be modified after re-running the test case if:

  — the implementation of the test case has been modified so that the implementation follows the specification of the test case, or

  — the implementation of the test environment has been modified, or

  — the IUT has been modified, or

  — the specification of the test case has been modified and this modification has been accepted. Afterwards, the implementation of the test case shall be modified according to the modification of the test case specification.

— **Channel status:**
All channel specific status variables (e.g. flags, indicators, counters etc.) have to be verified on a per-channel basis for the active channel(s).

— **Aggregated channel status:**
Unless otherwise specified, in test cases which use the aggregated channel status for verification purposes, the aggregated channel status has to be cleared either in the NIT of every cycle (respectively after its verification in the NIT) except, for test cases in 7.2.2 test purpose at the end of the last static slot of every cycle.

— **Slot Status:**
To verify the slot status for an specific slot on a channel a receive buffer shall be configured by the UT in the IUT, if a buffer configuration is not explicitly stated in the test case.

— **Message buffers for transmission of frames:**
If a frame shall be transmitted in a specific slot, then a message buffer shall be configured for this slot even if the message buffer is not explicitly configured in the test case.

— **Message buffers for reception of frames:**
If a frame shall be received in a specific slot, then a message buffer shall be configured for this slot even if the message buffer is not explicitly configured in the test case.

— **Tolerance window:**
A statement similar to “The LT starts to simulate startup frames *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN” defines a tolerance window of ±0,25 * *gdCycle* during which the LT has to start its simulation. The exact point in time at which the LT begins its simulation is is not relevant for the correct test execution and therefor left free for the test implementer.

— **ξ and ξ_{IUT}:**
In a statement like “It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 12 and slot 1 / cycle 13 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.“, the term “ξ + ξ_{IUT}“ defines a tolerance of [-7; 6] µT. I.e. the time interval between the two frames sent by the IUT, which is measured by the LT is allowed to deviate by [-7; 6] µT.
In contrast, if ξ_{IUT} is used in combination with a verification window, like “[0; 100] µT + ξ_{IUT} µT“, this means that the verification window from the point of view of the test environment has to be reduced accordingly, because of the allowed deviation of IUTs clock and test environments clock. This leads for the given example “[0; 100] µT + ξ_{IUT} µT” to an allowed window of [5; 94] µT in test environments view.

— **Verification window:**
A statement similar to “It is verified (UT) that the IUT is in the *POC:initialize schedule* state *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.” defines a verification window during which the UT shall do the specified verification once (here: verification that IUT is in the *POC:initialize schedule*). The exact point in time within the verification window at which the UT does the verification is irrelevant for the correct test execution and therefor is left free to the test implementation.

A verification window is also be defined by statements like “In cycle 8, within interval tAW after t0 it shall be verified ...” with “$t_{AW} \in$ [0, 500] µT”. In this case, the verification window starts at t0 and ends at t0+500 µT.

— The clock deviation (ξ_{IUT}) has to be considered in all verification windows (for LT and UT) by reducing the windows accordingly.

— **Detecting low simultaneously:**
A phrase like "It is verified (LT) that the TxEN output and TxD output become low simultaneously" allows a time tolerance of 1 σT for detecting low at both outputs due to allowed fall times at the outputs and logic thresholds of the LT. Let t0_TxEN be the time when low of the TxEN output is detected by the LT and let t0_TxD be the time when low of the TxD output is detected by the LT. Then the tolerance of 1σT means that the absolute difference between t0_TxEN and t0_TxD shall be at most 1σT, i.e., |t0_TxEN - t0_TxD| <= 1 σT.

— **Clock of the LT:**
Unless otherwise specified, the LT does not perform a clock correction but statically applies its communication elements as specified in the test cases.
Depending on the startup scenario of the test case either LT or IUT starts the schedule. If LT initiates the communication the IUT aligns with the given schedule. However, if IUT starts the communication the LT is aware of the expected IUT's start of schedule.
In some test cases LT's time is shifted by starting the next cycle earlier or later. In these test cases the order of test steps is important for the referenced time: test steps before the instruction to shift the schedule refer to the LT time before the shift, test steps after this instruction refer to the LT time after the shift. Statements to shift the simulated frames, however, do not influence LT's clock.

— **Protocol related points in time:**
Although the LT does not perform clock correction the clock correction algorithm in the IUT makes sure that the clocks of the LT and the IUT are synchronized. Therefore, for the verification timings relative to the cycle start, to the NIT and to similar protocol-related points in time LT's schedule shall be taken as the references point. The allowed clock deviation between IUT and LT, $\boldsymbol{\xi_{IUT}}$, has to be considered for the verification timing by shortening the verification windows accordingly.
However, in some testcases LT's schedule is shifted in order to test the FlexRay clock correction algorithm. In these test cases IUT's schedule and LT's schedule differ for a number of cycles. To fit the requested verification timings in these cycles, valid verification windows are specified in both IUT's time and in LT's time. Due to simplification of the test description these verification windows may differ in start point or in size, but are valid by all means.

— **Execution time of test step:**
Each single test step shall be executed at the given time or within the given time window. If now timing is specified for a test step, the test step shall be executed immediately after completion of the previous test step.

— **LT transmission timing**:
Frames and symbols transmitted by the LT shall start at the according action point of the given slot or segment, unless otherwise specified. I.e. a statement like “... the LT simulates a frame in slot 1 ...” means that the LT starts the transmission at the action point of communication slot 1. The same principle applies to symbols transmitted in the symbol window.

— **Parameter value vs. preamble and test case:**
If the value of a parameter is defined in the preamble (Preamble I, II, III, IV) and in the test case itself (e.g., in the Configuration section of the test case), then the value as defined in the test case itself always supersedes the value as defined in the preamble.

— **Sampletick / Microtick / Macrotick:**
The abbreviation “σT” stands for sampletick and “σT” is equivalent to the parameter *gdSampleClockPeriod*. The abbreviation “MT” stands for macrotick and “MT” is equivalent to the parameter *gdMacrotick*. The abbreviation “µT” stands for microtick and “µT” is equivalent to the parameter *pdMicrotick*.

7.1.2 Test case structure

Each test case is structured by eight titles.

In the following, details and examples for each of these titles (ordered list) are given.

— **Test name**

Descriptive name of the test.

— **Test purpose**

Short description of which requirement of the FlexRay Protocol Specification is tested.

— **Applicability**

The following three applicability classes are available:

- SC: Single channel test case, i.e. applicable if pChannels = A or pChannels = B
- DC: Dual channel test case, i.e., applicable if pChannels = A&B
- TT-E: Time triggered external test case., i.e., applicable if the pair of CCs (source and sink CCs) offers the TT-E option

NOTE A test case might be applicable to more than one class, e.g., an applicability of “SC,DC” means that the test case is applicable if *pChannels* = A, *pChannels* = B, or *pChannels* = A&B.

— **Configuration**

The configuration section addresses the configuration of the parameters. If parameters need to be modified for the test execution, the modified parameter values are explicitly stated in the test case within a modification table. If modifications are used in a test case, the test case has to be executed for all modifications.

Different modification tables to basic configurations:

Table 7 defines one modification, valid for all basic configurations.

Table 7 — One modification, valid for all basic configurations

Parameter	Modification
pKeySlotID	2

Table 8 defines the set of modifications (numbered with Roman letters I, II, III, …), valid for all basic configurations.

Table 8 — Set of modifications (numbered with Roman letters I, II, III, …), valid for all basic configurations

Parameter	Modification		
	I	**II**	**III**
pAllowPassiveToActive	1	16	31
pAllowHaltDueToClock	false		

Table 9 defines the set of modifications, depending on the basic configurations.

Table 9 — Set of modifications, depending on the basic configurations

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdStaticSlot [MT]	42	40	68
gNumberOfStaticSlots	70	37	21
gPayloadLengthStatic [two-byte word]	1		
gdNIT [MT]	49	17	33

— **Preamble (setup state)**

In this subclause the necessary actions are defined to drive the IUT from the stable state (i.e. reset state) to the initial state for test execution. Four preambles (I, II, III, and IV) are defined in 7.8.4.4 Test purpose with IUT as following coldstart node, as leading coldstart node, as integrating node and as time gateway source / sink node. In some test cases, the preambles are modified slightly.

— **Test execution**

Test execution is organized in steps of taking actions by UT or LT and the verification of correct IUT's reaction and IUT's behaviour .

Dependency on modifications:

If parameter modifications are used and the execution of a test step depends on a certain modification, this dependency is marked by roman letters in brackets, i.e., “(I)”, (II)” or “(I,II)”.

EXAMPLE

Line 1: 1. In cycle 9, the LT simulates its frame in slot 1 with a payload length in the frame header equal to

Line 2: (I) *gPayloadLengthStatic* + 1

Line 3: (II, III) *gPayloadLengthStatic*

In this example, the test step 1 (lines 1-3) depends on the parameter modifications, i.e., line 2 is to be executed for modification I, line 3 is to be executed for modifications II and III.

Variants:

If the execution of a test step depends on a certain variant, this dependency is marked by lower Latin letters in brackets, i.e., “(a)”, “(b)”, “(c)”, etc. If variants are used in a test case, the test case has to be executed for all variants.

Line 1: 1. In cycle 16, 500 µT after cycle start and before NIT, it is verified (UT) that the counter *vAllowPassiveToActive* = 0 (*pAllowPassiveToActive* – 1) and that

Line 2: (a, b) the IUT stays in the *POC:normal passive* state (*vPOC!State* = NORMAL_PASSIVE) and

Line 3: (c) the IUT has returned to the *POC:normal active* state (*vPOC!State* = NORMAL_ACTIVE) and has set the *vPOC!ErrorMode* to ACTIVE.

In this example, the test step 1 (lines 1-3) depends on the variants, i.e., line 2 is to be executed for variants (a) and (b), line 3 for variant (c).

Instances:

In some test case, instances are used. In the following, two examples for instances are given.

EXAMPLE 1:

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

EXAMPLE 2:

Test instance 1:

— In cycles 7 to 9, the LT simulates its startup frame in slot 2.

— In cycle 7, it is verified (UT) that *vPOC!CHIHaltRequest* = false.

— The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

— …

Test instance 2:

— In cycles 7 to 9, the LT simulates its startup frame in slot 2.

— In cycle 7, it is verified (UT) that *vPOC!CHIHaltRequest* = false.

— At the beginning of cycle 8, the UT issues the CHI command DEFERRED_HALT.

— etc.

A test case shall be executed for all given instances.

Notes on modifications, variants, and instances:

In some test cases, modifications, variants, and instances are used in combinations in a nested way, e.g., two modifications (I and II) and three variants (a), (b), and (c) which gives in total 2 * 3 = 6 test executions for this example.

— **Postamble**

Description of the test steps defining how to drive the IUT into a final definite state.

— **Pass criteria**

Short description of the test pass criteria.

7.2 Receive data

7.2.1 Static segment

7.2.1.1 TSS

— **Test purpose**

Verify recognition of the transmission start sequence TSS in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications in Table 10, Table 11 and Table 12.

Table 10 — Modification to basic configuration 1a and 1b for static segment – receive data

Parameter	Modification to basic configurations					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	77	97	105	77	97	105
gdTSSTransmitter [gdBit]	1[a]	11	15	1[a]	11	15
pDecodingCorrection [µT]	12	52	68	24	104	136
pMacroInitialOffset[A,B] [MT]	5	6		5	6	
pMicroInitialOffset[A,B] [µT]	23		7	46		14

[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation

Table 11 — Modification to basic configuration 2a and 2b for static segment – receive data

Parameter	Modification to basic configurations					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	70	80	84	70	80	84
gdTSSTransmitter [gdBit]	1[a]	6	8	1[a]	6	8
pDecodingCorrection [µT]	24	64	80	12	32	40
pMacroInitialOffset[A,B] [MT]	4		5	4		5
pMicroInitialOffset[A,B] [µT]	46	6	70	23	3	35

[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation

Table 12 — Modification to basic configuration 3 for static segment – receive data

Parameter	Modification to basic configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	66	72	74
gdTSSTransmitter [gdBit]	1[a]	4	5
pDecodingCorrection [µT]	24	48	56
pMacroInitialOffset[A,B] [MT]	4	5	
pMicroInitialOffset[A,B] [µT]	6	22	14

[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) The LT simulates its frame in slot 1 with a TSS length of
3 / 8 [*gdBit*] in cycle 8,
1 [*gdBit*] in cycle 9,
gdTSSTransmitter [*gdBit*] in cycle 10,
gdTSSTransmitter + 2 + 3 / 8 [*gdBit*] in cycle 11, and
gdTSSTransmitter + 3 [*gdBit*] in cycle 12.

2) In the NIT of cycle 8, it is verified (UT) that the syntax error flag / indicator is true and that the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status.

3) In the NIT of cycles 9, 10 and 11, it is verified (UT) that the syntax error flag / indicator is false and that the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status.

4) In the NIT of cycle 12, it is verified (UT) that the syntax error flag / indicator is true and that the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the valid frame flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.2 Reserved bit

— **Test purpose**

Verify reception of the reserved bit in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame in slot 1 with the reserved bit set to '0' and it is verified (UT) that the reserved bit *vRF!Header!Reserved* is '0' in the frame contents data in the NIT of this cycle.

2) In cycle 9, the LT simulates its frame in slot 1 with the reserved bit set to '1' and it is verified (UT) that the reserved bit *vRF!Header!Reserved* is '1' in the frame contents data in the NIT of this cycle.

3) In cycle 10, the LT simulates its frame in slot 1 with the reserved bit set to '0' and it is verified (UT) that the reserved bit *vRF!Header!Reserved* is '0' in the frame contents data in the NIT of this cycle.

Figure 6 depicts the reserved bit.

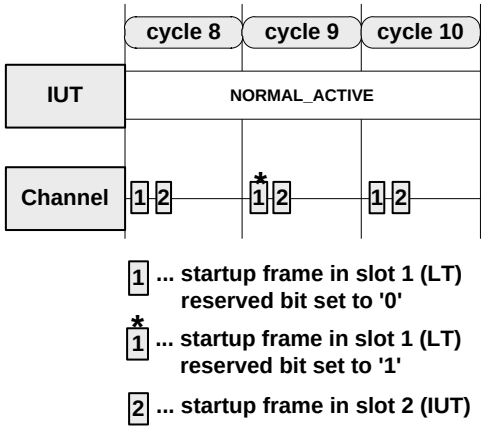


Figure 6 — Reserved bit

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The reserved bit *vRF!Header!Reserved* in the frame contents data is set accordingly.

7.2.1.3 Payload preamble indicator

— **Test purpose**

Verify reception of the payload preamble indicator in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame in slot 1 with the payload preamble indicator set to '0' and the null frame indicator set to '1'. It is verified (UT) that the payload preamble indicator *vRF!Header!PPIndicator* is '0' in the frame contents data in the NIT of this cycle. It is also verified (UT) that the valid frame flag / indicator is true and the content error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates its frame in slot 1 with the payload preamble indicator set to '1' and the null frame indicator set to '1'. It is verified (UT) that the payload preamble indicator *vRF!Header!PPIndicator* is '1' in the frame contents data in the NIT of this cycle. It is also verified (UT) that the valid frame flag / indicator is true and the content error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates its frame in slot 1 with the payload preamble indicator set to '0' and the null frame indicator set to '1'. It is verified (UT) that the payload preamble indicator *vRF!Header!PPIndicator* is '0' in the frame contents data in the NIT of this cycle. It is also verified (UT) that the valid frame flag / indicator is true and the content error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 10.

4) In cycle 11, the LT simulates its frame in slot 1 with the payload preamble indicator set to '1', the null frame indicator set to '0' and all payload bytes set to 0x00. It is verified (UT) that the valid frame flag / indicator is false and the content error flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 10.

Figure 7 depicts the payload preamble indicator.

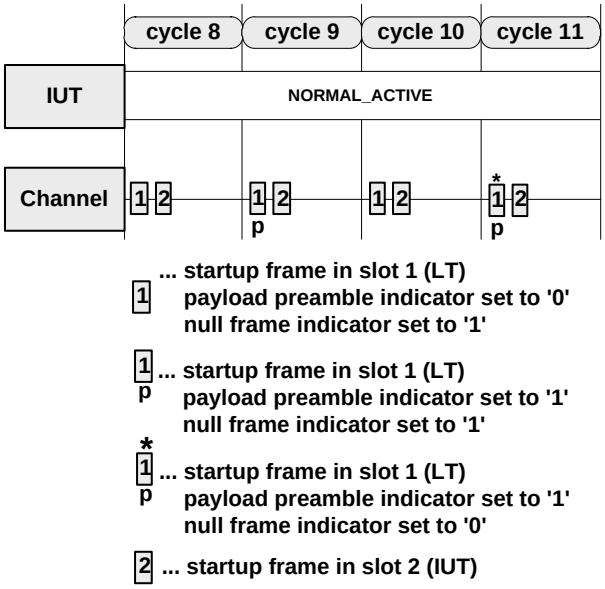


Figure 7 — Payload preamble indicator

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The payload preamble indicator *vRF!Header!PPIndicator* in the frame contents data is set accordingly and additionally the content error flag / indicator and the valid frame flag / indicator are set accordingly in the slot status data and in the aggregated channel status. The IUT does not update the frame contents data upon reception of an invalid frame.

7.2.1.4 Null frame indicator

— **Test purpose**

Verify reception of the null frame indicator in the static segment. Verify that the IUT does not update frame contents data upon reception of a null frame.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— Test execution

1) In cycle 8, the LT simulates its frame in slot 1 with the null frame indicator set to '1'. It is verified (UT) that the flag *vSS!NFIndicator* is '1' in the slot status data of slot 1 in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

2) In cycle 9, the LT simulates its frame in slot 1 with the null frame indicator set to '0' and all payload bytes set to 0x00. It is verified (UT) that the flag *vSS!NFIndicator* is '0' in the slot status data of slot 1 in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

3) In cycle 10, the LT simulates its frame in slot 1 with the null frame indicator set to '1'. It is verified (UT) that the flag *vSS!NFIndicator* is '1' in the slot status data of slot 1 in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 10.

Figure 8 depicts the null frame indicator.

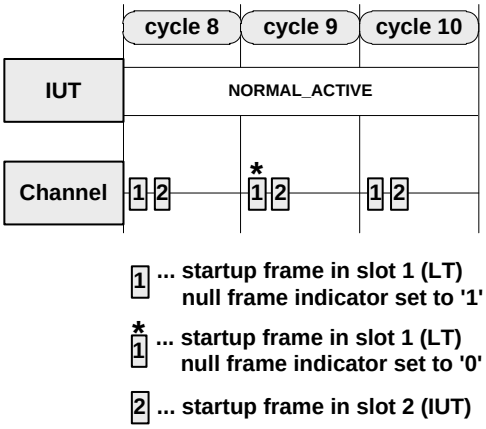


Figure 8 — Null frame indicator

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The flag *vSS!NFIndicator* in the slot status data is set accordingly. The IUT does not update frame contents data upon reception of a null frame.

7.2.1.5 Sync frame indicator

— Test purpose

Verify reception of the sync frame indicator in the static segment and the correct error indication in case of the reception of a startup frame having its sync frame indicator set to '0'.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications in Table 13.

Table 13 — Modification to basic configuration for static segment – sync frame indicator

Parameter	Modification
gMaxWithoutClockCorrectionFatal	15
gMaxWithoutClockCorrectionPassive	15

— **Preamble (setup state)**

Preamble I.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

Depending on the considered channel(s) the respective received or transmitted sync frames lists shall be verified in the test execution.

1) In cycle 8, the LT simulates its startup frame with the sync frame indicator set to '1' and matching header CRC.

2) In the symbol window of cycle 8, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2 (the IUT's startup frame ID) and the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 1 (the LT's startup frame ID).

3) In the NIT of cycle 8, it is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* in the frame contents data is '1' and the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status.

4) The UT clears the aggregated channel status after its verification.

5) In cycle 8, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick / pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2 the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 1 and the number of valid sync frames received and transmitted in the even and in the odd communication cycles is 2.

6) In cycle 9, the LT simulates its startup frame with the sync frame indicator set to '0' and matching header CRC.

7) In the symbol window of cycle 9, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2 and the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 1.

8) In the NIT of cycle 9, it is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* in the frame contents data is '1' (invalid frame received and therefore not updated) and the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status. The UT clears the aggregated channel status after its verification.

9) In cycle 9, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick / pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2, the received or transmitted sync frames list for even cycle contains the frame ID 1, the

received or transmitted sync frames list for odd cycle does not contain the frame ID 1, the number of valid sync frames received and transmitted in the even communication cycle is 2 and the number of valid sync frames received and transmitted in the odd communication cycle is 1.

10) In cycle 10, the LT simulates a static frame instead of a startup frame in slot 1.

11) In the symbol window of cycle 10, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2, the received or transmitted sync frames list for even cycle contains the frame ID 1 and the received or transmitted sync frames list for odd cycle does not contain the frame ID 1.

12) In the NIT of cycle 10, it is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* in the frame contents data is '0' and the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status. The UT clears the aggregated channel status after its verification.

13) In cycle 10, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2, the received or transmitted sync frames lists for even and for odd cycles do not contain the frame ID 1 and the number of valid sync frames received and transmitted in the even and in the odd communication cycles is 1.

14) In cycle 11, the LT simulates its startup frame with the sync frame indicator set to '1' and matching header CRC.

15) In the symbol window of cycle 10, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2 and the received or transmitted sync frames lists for even and for odd cycles do not contain the frame ID 1.

16) In the NIT of cycle 11, it is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* in the frame contents data is '1' and the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status. The UT clears the aggregated channel status after its verification.

17) In cycle 11, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2, the received or transmitted sync frames list for odd cycle contains the frame ID 1, the received or transmitted sync frames list for even cycle does not contain the frame ID 1, the number of valid sync frames received and transmitted in the even communication cycle is 1 and the number of valid sync frames received and transmitted in the odd communication cycle is 2.

18) In cycle 12, the LT simulates its startup frame with the sync frame indicator set to '1' and matching header CRC.

19) In the symbol window of cycle 12, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2, the received or transmitted sync frames list for odd cycle contains the frame ID 1, the received or transmitted sync frames list for even cycle does not contain the frame ID 1, the number of valid sync frames received and transmitted in the even communication cycle is 1 and the number of valid sync frames received and transmitted in the odd communication cycle is 2.

20) In the NIT of cycle 12, it is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* in the frame contents data is '1' and the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status.The UT clears the aggregated channel status after its verification.

21) In cycle 12, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames lists for even and for odd cycles contain the frame ID 2, the received or transmitted sync frames lists for even and for odd cycles contain the

frame ID 1 and the number of valid sync frames received and transmitted in the even and in the odd communication cycles is 2.

Figure 9 depicts the sync frame indicator.

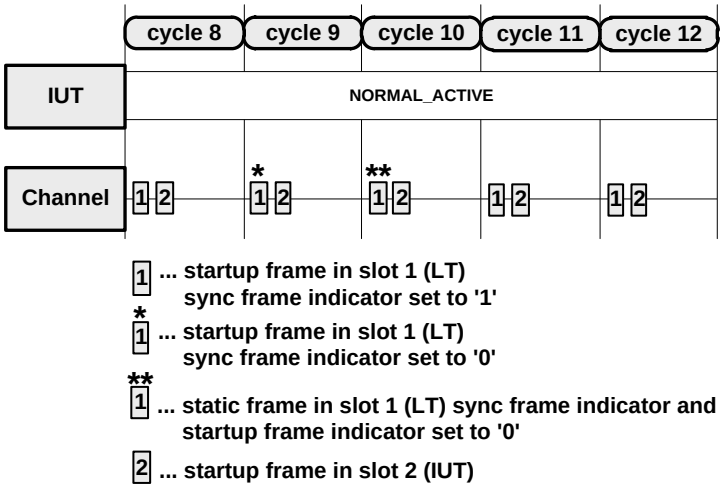


Figure 9 — Sync frame indicator

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The content error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status, the number of sync frames, the received or transmitted sync frames lists and the sync frame indicator *vRF!Header!SyFIndicator* in the frame contents data are set accordingly.

7.2.1.6 Startup frame indicator

— **Test purpose**

Verify reception of the startup frame indicator in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its startup frame with the startup frame indicator set to '1', the sync frame indicator set to '1' and matching header CRC. It is verified (UT) that the startup frame indicator *vRF!Header!SuFIndicator* in the frame contents data is '1', the valid frame flag / indicator is true and

the content error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates its startup frame with the startup frame indicator set to '0', the sync frame indicator set to '1' and matching header CRC. It is verified (UT) that the startup frame indicator *vRF!Header!SuFIndicator* in the frame contents data is '0', the valid frame flag / indicator is true and the content error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates its startup frame with the startup frame indicator set to '1', the sync frame indicator set to '1' and matching header CRC. It is verified (UT) that the startup frame indicator *vRF!Header!SuFIndicator* in the frame contents data is '1', the valid frame flag / indicator is true and the content error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

Figure 10 depicts the startup frame indicator.

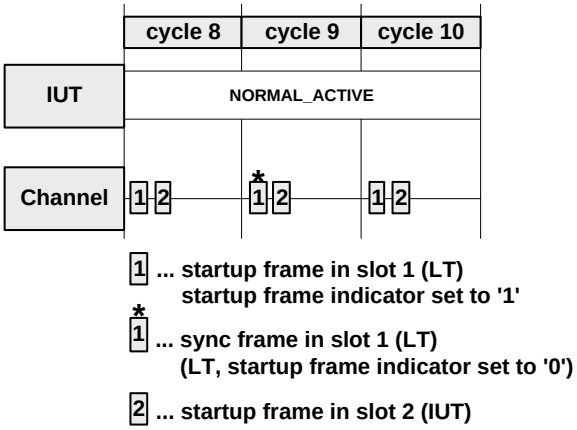


Figure 10 — Startup frame indicator

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The content error flag / indicator in the slot status data and in the aggregated channel status and the startup frame indicator *vRF!Header!SuFIndicator* in the frame contents data are set accordingly.

7.2.1.7 Frame ID

— Test purpose

Verify reception of the frame ID in the static segment. Verify that the IUT does not update frame contents data upon reception of invalid frames.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 14.

Table 14 — Modification to basic configuration for static segment – frame ID

Parameter	Modification
gMaxWithoutClockCorrectionFatal	3

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame with the frame ID set to 1 and matching header CRC. It is verified (UT) that the frame ID *vRF!Header!FrameID* is 1 in the frame contents data, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle. It is also verified (UT) that the remainder of the frame contents data of slot 1 in the IUT corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

2) In cycle 9, the LT simulates its frame with the frame ID set to 0, matching header CRC and with the payload 0x55 in byte 0 and 0xAA in byte 1. It is verified (UT) that the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 in the IUT is not updated and corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

3) In cycle 10, the LT simulates its frame with the frame ID set to 2, matching header CRC and with the payload 0x55 in byte 0 and 0xAA in byte 1. It is verified (UT) that the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 in the IUT is not updated and corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

4) In cycle 11, the LT simulates its frame with the frame ID set to 1, matching header CRC and with the payload 0x55 in byte 0 and 0xAA in byte 1. It is verified (UT) that the frame ID *vRF!Header!FrameID* is 1 in the frame contents data, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle. It is also verified (UT) that the remainder of the frame contents data of slot 1 in the IUT corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 11.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The content error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the frame ID *vRF!Header!FrameID* in the frame contents data are set accordingly. The IUT does not update frame contents data upon reception of invalid frames.

7.2.1.8 Payload length

— Test purpose

Verify reception of payload length in the static segment.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 15.

Table 15 — Modification to basic configurations for static segment – payload length

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdStaticSlot [MT]	274			271			331		
gNumberOfStaticSlots	10			5			4		
gPayloadLengthStatic [two-byte word]	0	64	127	0	64	127	0	64	76
gdNIT [MT]	249			141			137		
gdActionPointOffset [MT]	4			4			3		
pMacroInitialOffset[A] [MT]	6			5			5		
pMacroInitialOffset[B] [MT]	6			5			5		
gMaxWithoutClockCorrectionFatal	3			3			3		

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates its frame in slot 1 with a payload length in the frame header equal to *gPayloadLengthStatic.* It is verified (UT) that the length field *vRF!Header!Length* in the frame contents data equals *gPayloadLengthStatic*, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates its frame in slot 1 with a payload length in the frame header equal to:

   (I, II) *gPayloadLengthStatic* + 1.

   (III) *gPayloadLengthStatic.*

3) In the NIT of cycle 9, it is verified (UT)

   (I, II) that the frame contents data of slot 1 was not updated, that the content error flag / indicator is true and that the valid frame flag / indicator is false.

(III) that the frame contents data was updated, that the content error flag / indicator is false and that the valid frame flag / indicator is true

in the slot status data of slot 1 and in the aggregated channel status.

4) In cycle 10, the LT simulates its frame in slot 1 with a payload length in the frame header equal to

(II, III) *gPayloadLengthStatic* – 1.

(I) *gPayloadLengthStatic*.

5) In the NIT of cycle 10, it is verified (UT)

(II, III) that the frame contents data of slot 1 was not updated, that the content error flag / indicator is true and that the valid frame flag / indicator is false

(I) that the frame contents data of slot 1 was not updated, that the content error flag / indicator is false and that the valid frame flag / indicator is true

in the slot status data of slot 1 and in the aggregated channel status.

6) In cycle 11, the LT simulates its frame in slot 1 with a payload length in the frame header equal to *gPayloadLengthStatic*. It is verified (UT) that the length field *vRF!Header!Length* in the frame contents data equals *gPayloadLengthStatic,* the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The content error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the length field *vRF!Header!Length* in the frame contents data are set accordingly.

7.2.1.9 Cycle count

— **Test purpose**

Verify reception of the cycle counter in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame in slot 1 with the cycle counter set to 8. It is verified (UT) that the cycle counter *vRF!Header!CycleCount* is 8 in the frame contents data of slot 1, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates its frame in slot 1 with the cycle counter set to 7. It is verified (UT) that the cycle counter *vRF!Header!CycleCount* is 8 (holds the value from the previous cycle) in the frame contents data of slot 1, the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates its frame in slot 1 with the cycle counter set to 10. It is verified (UT) that the cycle counter *vRF!Header!CycleCount* is 10 in the frame contents data of slot 1, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The content error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the cycle counter *vRF!Header!CycleCount* in the frame contents data are set accordingly.

7.2.1.10 Header CRC

— **Test purpose**

Verify reception of the header CRC in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame in slot 1 with a correct header CRC. It is verified (UT) that the header CRC *vRF!Header!HeaderCRC* in the frame contents data equals the header CRC send by the LT in slot 1 of cycle 8, and that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

2) In cycle 9, the LT simulates its frame in slot 1 with the payload 0x55 in byte 0 and 0xAA in byte 1 and wrong header CRC (bit flip in the LSB of the header CRC). It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

3) In cycle 10, the LT simulates its frame in slot 1 with a correct header CRC. It is verified (UT) that the header CRC *vRF!Header!HeaderCRC* in the frame contents data equals the header CRC send by the LT in slot 1 of cycle 10, and that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1and in the aggregated channel status in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 10.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the header CRC *vRF!Header!HeaderCRC* in the frame contents data are set accordingly. If the header CRC of a received frame is wrong, the syntax error flag / indicator has to be true and the frame contents data shall not be updated.

7.2.1.11 FSS / BSS

— **Test purpose**

Verify recognition of frame start sequence FSS and first byte start sequence BSS in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

The LT simulates an additional frame in static slot 3 with the reserved bit, the payload preamble indicator, the null frame indicator, the sync frame indicator and the startup frame indicator set to '0'.

1) In cycle 8, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 1-0. It is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 1-1-0. It is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 1-1-1-0. It is verified (UT) that the syntax error flag / indicator is false that the valid frame flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status in the NIT of the cycle.

4) In cycle 11, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 1-1-1-1-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

5) In cycle 12, the LT simulates an additional frame in static slot 3 with FSS / BSS as a single bit set to 0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

6) In cycle 13, the LT simulates an additional frame in static slot 3 with FSS / BSS as a single bit set to 1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

7) In cycle 14, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 0-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

8) In cycle 15, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 1-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

9) In cycle 16, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 1-0-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

10) In cycle 17, the LT simulates an additional frame in static slot 3 with FSS / BSS being the bit sequence 1-1-0-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status. The valid frame flag is set accordingly in the slot status data.

7.2.1.12 Payload

— **Test purpose**

Verify reception of payload data in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 16.

Table 16 — Modification to basic configurations for static segment – payload

Parameter	Modification to Basic Configurations														
	1a & 1b					2a & 2b					3				
	I	II	III	IV	V	I	II	III	IV	V	I	II	III	IV	V
gdStaticSlot [MT]	274					271					331				
gNumberOfStaticSlots	10					5					4				
gPayloadLengthStatic [two-byte word]	0	1	32	64	127	0	1	32	64	127	0	1	32	64	76
gdNIT [MT]	249					141					137				
gdActionPointOffset [MT]	4					4					3				
pMacroInitialOffset[A,B] [MT]	6					5					5				

— **Preamble (setup state)**

Preamble I.

Deviating from the preamble the LT's payload data is 0x01 in byte 0, 0x02 in byte 1, ..., 0xFE in byte 253. According to the applied payload length *gPayloadLengthStatic*, the LT has to accommodate its frames with respect to the payload length in the frame header and the number of payload bytes in the payload. E.g., if *gPayloadLengthStatic* = 32 two-byte words, the LT simulates its frames with the payload length set to 32 and the payload comprising byte 0 up to byte 63.

— **Test execution**

1) In the NIT of cycle 8, it is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status and that the received payload data *vRF!Payload* in the frame contents data equals the one transmitted by the LT. If the IUT is not able to receive the whole payload of the frame due to size restrictions of receive buffers, the received bytes – starting with byte 0 – are compared to the transmitted ones. Bytes at the end of the frame exceeding the buffer size may be dropped.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status and the received payload in the frame contents data equals the payload transmitted by the LT.

7.2.1.13 BSS

— **Test purpose**

Verify recognition of byte start sequence BSS in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

The LT simulates an additional frame in static slot 3 with the reserved bit, the payload preamble indicator, the null frame indicator, the sync frame indicator and the startup frame indicator set to '0'.

1) In cycle 8, the LT simulates an additional frame in static slot 3 with:

   (a) the BSS after the frame ID in the header segment,

   (b) the BSS after the first data byte Data 0 in the payload segment and

   (c) the BSS after the first CRC byte in the trailer segment

   being the bit sequence 1-0. It is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates an additional frame in static slot 3 with:

   (a) the BSS after the frame ID in the header segment,

   (b) the BSS after the first data byte Data 0 in the payload segment and

   (c) the BSS after the first CRC byte in the trailer segment

   being the bit sequence 0-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

3) In cycle 10, the LT simulates an additional frame in static slot 3 with:

   (a) the BSS after the frame ID in the header segment,

   (b) the BSS after the first data byte Data 0 in the payload segment and

   (c) the BSS after the first CRC byte in the trailer segment

   being the bit sequence 0-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

4) In cycle 11, the LT simulates an additional frame in static slot 3 with:

   (a) the BSS after the frame ID in the header segment,

   (b) the BSS after the first data byte Data 0 in the payload segment and

   (c) the BSS after the first CRC byte in the trailer segment

   being the bit sequence 1-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status. The valid frame flag is set accordingly in the slot status data.

7.2.1.14 Frame CRC

— **Test purpose**

Verify reception of frame CRC in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 17.

Table 17 — Modification to basic configuration for static segment – frame CRC

Parameter	Modification
gMaxWithoutClockCorrectionFatal	3

— **Preamble (setup state)**

Preamble I.

Deviating from the Preamble the LT's payload data is 0xAA in byte 0 and 0x55 in byte 1 and 0x00 in all other bytes.

— **Test execution**

1) In cycle 8, the LT simulates its frame in slot 1 correctly. It is verified (UT) that the syntax error flag / indicator is false and that the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates its frame in slot 1 with a bit flip in the first bit of the payload data within the frame without adaption of the frame CRC. It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated

channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

3) In cycle 10, the LT simulates its frame in slot 1 with the payload 0x55 in byte 0, 0xAA in byte 1 and 0x00 in all other bytes and a bit flip in the first bit of the frame CRC within the frame. In the NIT of this cycle it is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

4) In cycle 11, the LT simulates its frame in slot 1 correctly. It is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 11.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the valid frame flag / indicator are set accordingly in the slot status data and in the aggregated channel status. If the frame CRC of a received frame is wrong, the syntax error flag / indicator has to be true, the valid frame flag / indicator has to be false and the frame contents data shall not be updated.

7.2.1.15 FES

— **Test purpose**

Verify recognition of frame end sequence FES in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame in slot 1 with FES being the bit sequence 0-1. It is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

2) In cycle 9, the LT simulates its frame in slot 1 with FES being the bit sequence 1-0. It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 8.

3) In cycle 10, the LT simulates its frame in slot 1 with FES being the bit sequence 0-0. It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 10.

4) In cycle 11, the LT simulates its frame in slot 1 with FES being the bit sequence 1-1. It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot 1 corresponds to the contents of the frame as simulated by the LT in slot 1 of cycle 10.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the valid frame flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.16 MTS

— **Test purpose**

Verify recognition of an MTS in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame in static slot 1. 500 µT after end of dynamic segment in this cycle and before cycle end , it is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.

2) In cycle 9, the LT simulates a valid MTS in static slot 1. 500 µT after end of dynamic segment in this cycle and before cycle end , it is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.

3) In cycle 10, the LT simulates its frame in static slot 1. 500 µT after end of dynamic segment in this cycle and before cycle end, it is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the flag *vSS!ValidMTS* in the symbol window status are set accordingly. If an MTS is received within the static segment, the syntax error flag / indicator has to be true, the valid frame flag / indicator has to be false and the flag *vSS!ValidMTS* has to be false.

7.2.1.17 WUP

— Test purpose

Verify the reception of a WUP in the static segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates its frame in static slot 1. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status in the NIT of the cycle.

2) In cycle 9, the LT simulates a valid WUP (high-low-high-low-high phase) in static slot 1. The low phase has a length of *gdWakeupRxLow* [*gdBit*], the high phase a length of *gdWakeupRxIdle* [*gdBit*]. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status in the NIT of the cycle.

3) In cycle 10, the LT simulates its frame in static slot 1. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 1 and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status in the NIT of the cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator in the slot status data and in the aggregated channel status, the wakeup status *vPOC!WakeupStatus* in the protocol operation control status and the wakeup pattern received indicator in the wakeup and startup status are set accordingly.

7.2.1.18 Complete valid frame

— Test purpose

Verify frame reception in the static segment during *POC:normal active* state and *POC:normal passive* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 18.

Table 18 — Modification to basic configurations for static segment – complete valid frame

Parameter	Modification
gMaxWithoutClockCorrectionPassive	2
gMaxWithoutClockCorrectionFatal	3
pAllowHaltDueToClock	false

— Preamble (setup state)

Preamble I.

Deviating from the Preamble the LT simulates an additional non-sync frame in the static slot 3.

— Test execution

1) In cycle 8, the LT simulates its frame in slot 1 correctly. It is verified (UT) that the valid frame flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9 to 13, the LT stops transmission of its startup frame in slot 1 and continues sending the frame in slot 3.

3) In cycle 12, 500 µT after cycles start and before start of slot 3, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

4) In the NIT of cycle 12 it is verified (UT) that the valid frame flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator is set accordingly in the slot status data and in the aggregated channel status.

7.2.1.19 Complete valid frame too late

— Test purpose

Verify reception of a delayed frame crossing slot boundaries in the static segment and the associated slot status indication.

— Applicability

SC, DC.

— Configuration

All basic configurations.

$n_{Id} = 3$

— Preamble (setup state)

Preamble II.

— Test execution

The LT simulates an additional frame in static slot n_{Id} with the reserved bit, the payload preamble indicator, the null frame indicator, the sync frame indicator and the startup frame indicator set to '0' and all payload bytes set to 0x00.

1) In cycle 7, the LT simulates its frame in slot n_{Id} correctly.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false and the content error flag / indicator is false in the slot status data of slot n_{Id} and in the aggregated channel status in the NIT of the cycle.
It is verified (UT) that the boundary violation flag / indicator is false in the slot status data of slot n_{Id} and slot $n_{Id} + 1$ and in the aggregated channel status in the NIT of the cycle.

2) In cycle 8, the LT simulates its frame in slot n_{Id} delayed such that the last byte of the frame CRC starts at the boundary to slot $n_{Id} + 1$ and that the frame causes a boundary violation.
It is verified (UT) that the valid frame flag is false, the syntax error flag is false and the content error flag is false in the slot status data of slot n_{Id} and slot $n_{Id} + 1$ in the NIT of the cycle.
It is verified (UT) that the boundary violation flag / indicator is true in the slot status data of slot n_{Id} and slot $n_{Id} + 1$ and in the aggregated channel status in the NIT of the cycle.
It is verified (UT) that the valid frame indicator is true, the syntax error indicator is false, the content error indicator is false and the additional communication indicator is false in the aggregated channel status in the NIT of the cycle.

3) In cycle 9, the LT simulates its frame in slot n_{Id} delayed such that the frame ends 5 * *gdBit* before the boundary to slot $n_{Id} + 1$ causing a boundary violation during channel idle recognition.
It is verified (UT) that the valid frame flag is true, the syntax error flag is false and the content error flag is false in the slot status data of slot n_{Id} in the NIT of the cycle.
It is verified (UT) that the valid frame flag is false, the syntax error flag is false and the content error flag is false in the slot status data of slot $n_{Id} + 1$ in the NIT of the cycle.
It is verified (UT) that the boundary violation flag / indicator is true in the slot status data of slot n_{Id} and slot $n_{Id} + 1$ and in the aggregated channel status in the NIT of the cycle.
It is verified (UT) that the valid frame indicator is true, the syntax error indicator is false, the content error indicator is false and the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.

4) In cycle 10, the LT simulates its frame in slot n_{Id} correctly.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false and the content error flag / indicator is false in the slot status data of slot n_{Id} and in the aggregated channel status in the NIT of the cycle.
It is verified (UT) that the boundary violation flag / indicator is false in the slot status data of slot n_{Id} and slot n_{Id} + 1 and in the aggregated channel status in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator, the boundary violation flag / indicator, the syntax error flag / indicator and the content error flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.20 Complete valid frame too late reaching dynamic segment

— **Test purpose**

Verify reception of a delayed frame crossing the boundary between static and dynamic segment and the associated slot status indication.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{Id} = *gNumberOfStaticSlots*.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

The LT simulates an additional frame in static slot n_{Id} with the reserved bit, the payload preamble indicator, the null frame indicator, the sync frame indicator and the startup frame indicator set to '0' and all payload bytes set to 0x00.

1) In cycle 7, the LT simulates its frame in slot n_{Id} correctly.
It is verified (UT) that the valid frame flag is true in the slot status data of slot n_{Id} and that the boundary violation flag / indicator is false in the slot status data of slot n_{Id} and slot n_{Id} +1 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 8, the LT simulates its frame in slot n_{Id} delayed such that the last byte of the frame CRC starts at the boundary to slot n_{Id} + 1 and that the frame causes a boundary violation, i.e. at the beginning of the dynamic segment.
It is verified (UT) that the valid frame flag is false in the slot status data of slot n_{Id} and slot n_{Id} +1 and that the boundary violation flag / indicator is true in the slot status data of slot n_{Id} and slot n_{Id} +1 and in the aggregated channel status in the NIT of the cycle.

3) In cycle 9, the LT simulates its frame in slot n_{Id} delayed such that the frame ends 5 * *gdBit* before the boundary to slot n_{Id} + 1 causing a violation of the boundary between the static segment and the dynamic segment during channel idle recognition.
It is verified (UT) that the valid frame flag is true in the slot status data of slot n_{Id} and that the boundary violation flag / indicator is true in the slot status data of slot n_{Id} and slot n_{Id} +1 and in the aggregated channel status in the NIT of the cycle.

4) In cycle 10, the LT simulates its frame in slot n_{Id} correctly.
It is verified (UT) that the valid frame flag is true in the slot status data of slot n_{Id} and that the boundary violation flag / indicator is false in the slot status data of slot n_{Id} and slot n_{Id} +1 and in the aggregated channel status in the NIT of the cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator and the boundary violation flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.21 More than one frame within one slot

— Test purpose

Verify reception of more than one frame in one static slot.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 19.

Table 19 — Modification to basic configurations for static segment – more than one frame within one slot

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdStaticSlot [MT]	42	40	68
gNumberOfStaticSlots	70	37	21
gPayloadLengthStatic [two-byte word]	1		
gdNIT [MT]	49	17	33

— Preamble (setup state)

Preamble III.

Deviating from the Preamble the LT simulates additional frames in static slot 3. Reserved bit, payload preamble indicator, sync frame indicator and startup frame indicator of those frames are set to '0', null frame indicator to '1'. The payload of the LT's default frame (frame 1) in slot 3 is 0xCC in byte 0 and 0x33 in byte 1, the payload of the LT's second frame (frame 2) 0x99 in byte 0 and 0x66 in byte 1 and the payload of the LT's third frame (frame 3) 0xEE in byte 0 and 0x11 in byte 1.

— **Test execution**

1) In cycle 9, the LT simulates the default frame in slot 3.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is false in the aggregated channel status in the NIT of the cycle.

2) In cycle 10, the LT simulates frame 1 and frame 2 in slot 3. The inter-frame gap is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is true, the content error flag / indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 10.

3) In cycle 11, the LT simulates frame 1 with the frame ID set to 0 followed by frame 2 in slot 3. The inter-frame gap is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is true and the boundary violation flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 2 as simulated by the LT in slot 3 of cycle 11.

4) In cycle 12, the LT simulates frame 1 followed by frame 2 with the frame ID set to 0 in slot 3. The inter-frame gap is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is true, the content error flag / indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 12.

5) In cycle 13, the LT simulates frame 1 and frame 2 in slot 3 followed by frame 3 starting in slot 3 and ending in slot 4 (the first byte of the payload data starts at the boundary to slot 4). The inter-frame gap between frame 1 and frame 2 is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is true, the content error flag / indicator is false and the boundary violation flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 13.

6) In cycle 14, the LT simulates frame 1 with the frame ID set to 0 and frame 2 in slot 3 followed by frame 3 starting in slot 3 and ending in slot 4 (the first byte of the payload data starts at the boundary to slot 4). The inter-frame gap between frame 1 and frame 2 is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is true and the boundary violation flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 2 as simulated by the LT in slot 3 of cycle 14.

7) In cycle 15, the LT simulates frame 1 followed by frame 2 with the frame ID set to 0 in slot 3 followed by frame 3 starting in slot 3 and ending in slot 4 (the first byte of the payload data starts at the boundary to slot 4). The inter-frame gap between frame 1 and frame 2 is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is true, the content error flag / indicator is false and the boundary violation flag / indicator is true in the slot status

data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 14.

8) In cycle 16, the LT simulates frame 1 in slot 3 followed by frame 2 starting in slot 3 and ending in slot 4 (the first byte of the payload data starts at the boundary to slot 4).
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is false and the boundary violation flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 14.

9) In cycle 17, the LT simulates frame 1 with header CRC error (bit flip in the LSB of the header CRC) in slot 3 followed by frame 2 starting in slot 3 and ending in slot 4 (the first byte of the payload data starts at the boundary to slot 4). The payload data of frame 1 is 0x0F in byte 0 and 0xF0 in byte 1.
It is verified (UT) that the valid frame flag is false, the syntax error flag / indicator is true, the content error flag / indicator is false and the boundary violation flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is false in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 16.

10) In cycle 18, the LT simulates frame 1 with the frame ID set to 0 in slot 3 followed by frame 2 starting in slot 3 and ending in slot 4 (the first byte of the payload data starts at the boundary to slot 4). The payload data of frame 1 is 0x0F in byte 0 and 0xF0 in byte 1.
It is verified (UT) that the valid frame flag is false, the syntax error flag / indicator is false, the content error flag / indicator is true and the boundary violation flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is false in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 16.

11) In cycle 19, the LT simulates frame 1 with the frame ID set to 0 followed by frame 2 with a header CRC error (bit flip in the LSB of the header CRC) in slot 3 followed by frame 3 starting in slot 3 and ending in slot 4 (the first byte of the payload data starts at the boundary to slot 4). The inter-frame gap between frame 1 and frame 2 is 2 * *cChannelIdleDelimiter*. The payload data of frame 1 and frame 2 is 0x0F in byte 0 and 0xF0 in byte 1.
It is verified (UT) that the valid frame flag is false, the syntax error flag / indicator is true, the content error flag / indicator is true and the boundary violation flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is false in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 16.

12) In cycle 20, the LT simulates frame 1 with the frame ID set to 0 followed by frame 2 with a header CRC error (bit flip in the LSB of the header CRC) in slot 3. The inter-frame gap is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag is false, the syntax error flag / indicator is true, the content error flag / indicator is true and the boundary violation flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is false in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 16.

13) In cycle 21, the LT simulates frame 1 with the frame ID set to 0, frame 2 and a low phase of 1 *gdBit* ending 2 * *cChannelIdleDelimiter* before the end of slot 3. The inter-frame gap is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is true, the

content error flag / indicator is true and the boundary violation flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 2 as simulated by the LT in slot 3 of cycle 21.

14) In cycle 22, the LT simulates frame 1 with the frame ID set to 0, frame 2 and a low phase of 1 *gdBit* ending 0,5 * *cChannelIdleDelimiter* before the end of slot 3. The inter-frame gap is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is true, the content error flag / indicator is true and the boundary violation flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is true in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 2 as simulated by the LT in slot 3 of cycle 22.

15) In cycle 23, the LT simulates the default frame in slot 3 correctly.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the additional communication indicator is false in the aggregated channel status in the NIT of the cycle.
It is also verified (UT) that the frame contents data corresponds to the contents of frame 1 as simulated by the LT in slot 3 of cycle 23.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator, the syntax error flag / indicator, the content error flag / indicator and the boundary violation flag / indicator in the slot status data and in the aggregated channel status and the additional communication indicator in the aggregated channel status are set accordingly. The frame contents data corresponds to the last received valid frame.

7.2.1.22 Complete frame with glitch

— **Test purpose**

Verify reception of a frame with glitches in the static segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble III.

Deviating from the preamble the payload data is 0xFF in byte 0 and 0x00 in byte 1.

The strobe point is located in the middle of a bit cell. Due to the quantization error the strobe point is actually a strobe window which starts at (*cStrobeOffset* – 1) * *gdSampleClockPeriod* and ends at *cStrobeOffset* * *gdSampleClockPeriod* relative to the beginning of the bit cell. Only if the bit value in the whole strobe window is altered by the glitch, the test may verify the glitch impact. If the bit value is only

partly modified in the strobe window, the test may not provide a defined result. The frame may be valid or invalid.

— **Test execution**

Test instance 1 – glitch is a low phase of length *gdSampleClockPeriod*:

1) In cycle 9, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

2) Starting with cycle 10, the LT simulates its startup frame in slot 2 with a glitch shifted over the frame's payload byte 0 in steps of *gdSampleClockPeriod* starting at the rising edge of the first BSS of the payload segment. The glitch does not pass the majority voting. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of every cycle.

3) In the last cycle, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

Test instance 2 – glitch is a high phase of length *gdSampleClockPeriod*:

1) In cycle 9, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

2) Starting with cycle 10, the LT simulates its startup frame in slot 2 with a glitch shifted over the frame's payload byte 1 in steps of *gdSampleClockPeriod* starting at the falling edge of the second BSS of the payload segment. The glitch does not pass the majority voting. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of every cycle.

3) In the last cycle, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

Test instance 3 – glitch is a low phase of length 4 * *gdSampleClockPeriod*:

1) In cycle 9, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

2) Starting with cycle 10, the LT simulates its startup frame in slot 2 with a glitch shifted over payload byte 0 in steps of 4 * *gdSampleClockPeriod* starting at –6 * *gdSampleClockPeriod* before the falling edge of the first BSS in the payload segment. The glitch passes the majority voting.

3) In the NIT of every cycle , it is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status, if the glitch is not located in the middle of a bit cell and, therefore, does not alter the bit value in the strobe window, and that the valid frame flag is false and the syntax error flag is true in the slot status data of slot 2 and that the syntax error indicator is true in the aggregated channel status, if the glitch is located in the middle of the bit cell, i.e. the glitch covers the strobe window, and alters the bit value.

4) In the last cycle, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

Figure 11 depicts the complete frame with glitch – test instance 3.

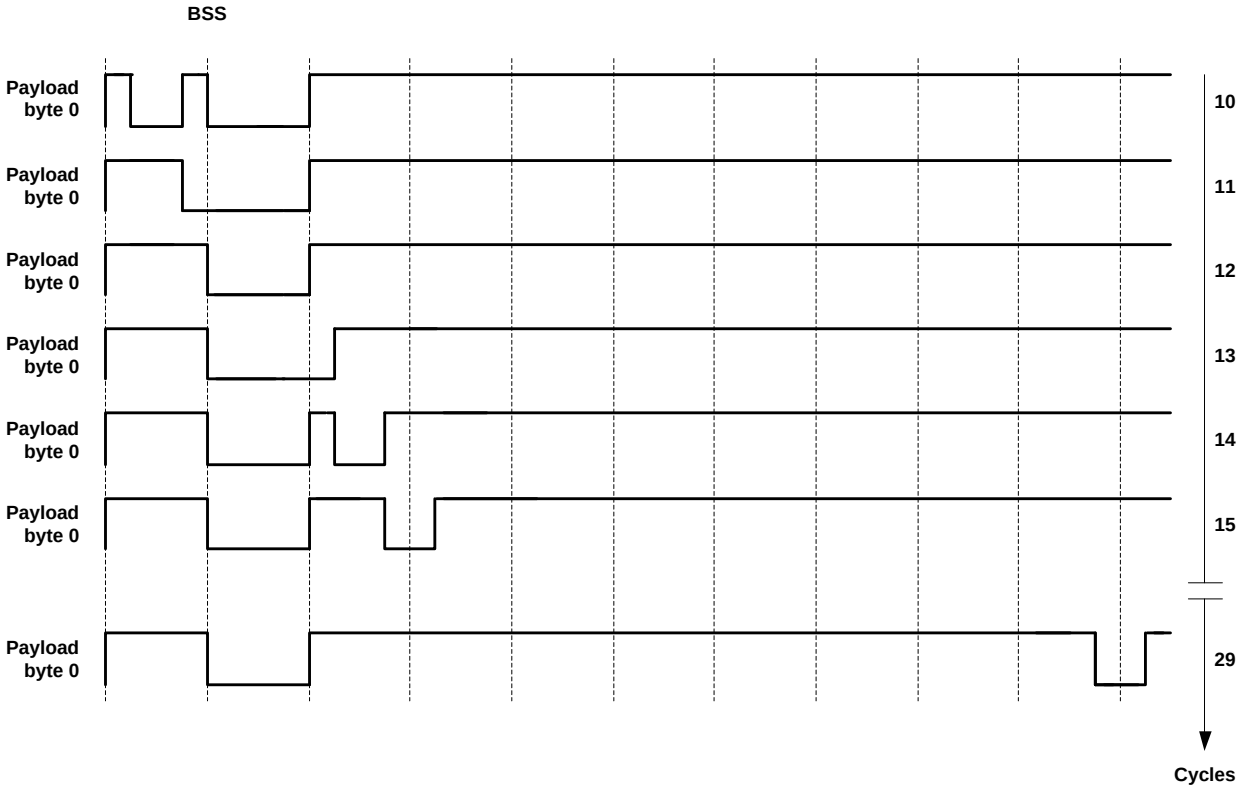


Figure 11 — Complete frame with glitch – test instance 3

Test instance 4 – glitch is a high phase of length 4 * *gdSampleClockPeriod*:

1) In cycle 9, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

2) Starting with cycle 10, the LT simulates its startup frame in slot 2 with a glitch shifted over payload byte 1 in steps of 4 * *gdSampleClockPeriod* starting at –2 * *gdSampleClockPeriod* before the falling edge of the first BSS in the payload segment.
The glitch passes the majority voting. It is verified (UT)

   (a) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of every cycle if the glitch is not located in the middle of a bit cell and, therefore, does not alter the bit value in the strobe window, and

   (b) that the valid frame flag is false and the syntax error flag is true in the slot status data of slot 2 and that the syntax error indicator is true in the aggregated channel status in the NIT of every cycle if the glitch is located in the middle of the bit cell, i.e. the glitch covers the strobe window, and alters the bit value.

3) In the last cycle, the LT simulates its startup frame in slot 2 correctly. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

Figure 12 depicts the complete frame with glitch – test instance 4.

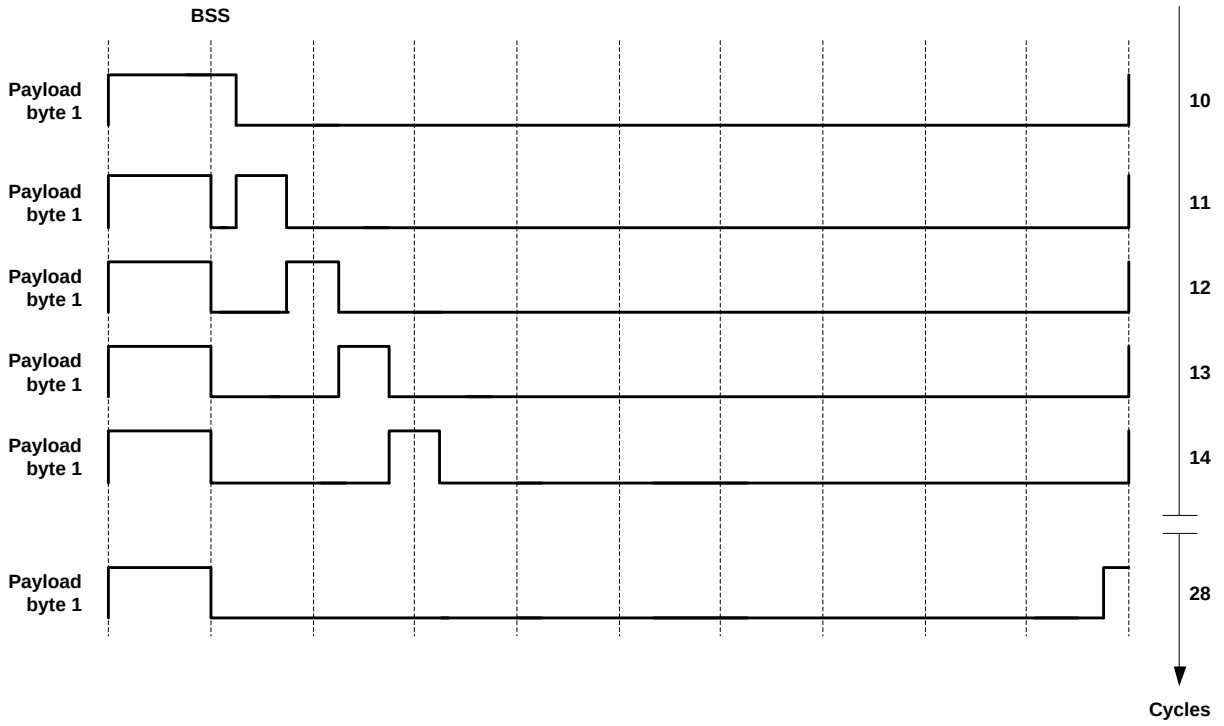


Figure 12 — Complete frame with glitch – test instance 4

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator and the syntax error flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.23 Noise

— Test purpose

Verify recognition of noise in a static slot.

— Applicability

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates idle in slot 3. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

2) In cycle 10, the LT simulates noise (low phase of length *gdSampleClockPeriod*) in slot 3. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

3) In cycle 11, the LT simulates noise (low phase of length 2 * *gdSampleClockPeriod*) in slot 3. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

4) In cycle 12, the LT simulates noise (low phase of length 3 * *gdSampleClockPeriod*) in slot 3. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

5) In cycle 13, the LT simulates noise (low phase of length (*cStrobeOffset* + 1) * *gdSampleClockPeriod*) in slot 3. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

6) In cycle 14, the LT simulates noise (low phase of length (*gdTSSTransmitter* + 2) * *gdBit*) in slot 3. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

7) In cycle 15, the LT simulates idle in slot 3. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot 3 and in the aggregated channel status and that the valid frame flag is false in the slot status data of slot 3 in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator in the slot status data and in the aggregated channel status and the valid frame flag in the slot status data are set accordingly.

7.2.1.24 Channel idle delimiter

— Test purpose

Verify the length of the channel idle delimiter. The IUT shall recognize channel idle if the physical layer stays idle for at least *cChannelIdleDelimiter*. Thus, the IUT shall indicate a single disturbance, if a short low phase is followed by a valid frame before the IUT may recognize channel idle. And, the IUT shall indicate a disturbance and a valid frame, if a short low phase is followed by a valid frame after the IUT recognizes channel idle.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates a low phase of length *gdBit* followed by a high phase of length *cChannelIdleDelimiter* – 1 [*gdBit*] before the startup frame in slot 1. It is verified (UT) that the valid frame flag / indicator is false and the syntax error flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates a low phase of length *gdBit* followed by a high phase of length *cChannelIdleDelimiter* [*gdBit*] before the startup frame in slot 1. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is true in the slot status data of slot 1 and in the aggregated channel status in the NIT of the cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator and the syntax error flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.25 Channel idle after transmission

— Test purpose

Verify the length of the channel idle delimiter. The IUT shall be able to detect an incoming communication element earliest *cChannelIdleDelimiter* after the end of its own transmission.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 20.

Table 20 — Modification to basic configurations for static segment – channel idle after transmission

Parameter	Modification		
	I	II	III
gdIgnoreAfterTx [gdBit]	0	7	15

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a low phase of length *gdBit* after a period of *cChannelIdleDelimiter* – 1 [*gdBit*] past the last bit of the FES in the IUT's frame in slot 2. It is verified (UT) that the syntax error flag / indicator is false in the transmit buffer status of slot 2 and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates a low phase of length *gdBit* after a period of *cChannelIdleDelimiter* [*gdBit*] past the last bit of the FES in the IUT's frame in slot 2. It is verified (UT) that

   (I, II) the syntax error flag / indicator is true and

   (III) the syntax error flag / indicator is false

   in the transmit buffer status of slot 2 and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates a low phase of length *gdBit* after a period of 3 *gdBit* and a second low phase of length *gdBit* after a period of *cChannelIdleDelimiter* [*gdBit*] past the last bit of the FES in the IUT's frame in slot 2. It is verified (UT) that

   (I) the syntax error flag / indicator is false,

   (II) the syntax error flag / indicator is true and

   (III) the syntax error flag / indicator is false

   in the transmit buffer status of slot 2 and in the aggregated channel status in the NIT of the cycle.

4) In the NIT of cycle 11, it is verified (UT) that the boundary violation flag / indicator is false in the transmit buffer status of slot 2 and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the boundary violation flag / indicator are set accordingly in the transmit buffer status and in the aggregated channel status.

7.2.1.26 Falling edge within payload bit

— **Test purpose**

Verify that a falling edge within a payload bit does not cause bit re-synchronisation of the sample counter in the IUT.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble III.

Deviating from the Preamble the payload data of the LT's startup frame is 0xAA in byte 0 and 0x55 in byte 1 in slot 2 on the active channel(s).

— **Test execution**

In cycle 9, the LT simulates its startup frame in slot 2 with the samples of bit MSB in the first payload byte set to 1111 1111, of bit MSB–1 set to 1110 0001 and of bit MSB–2 set to 1111 1100. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT does not interpret a falling edge in a payload bit as bit synchronisation edge. The IUT receives the LT's startup frame correctly. The valid frame flag / indicator and the syntax error flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.27 Channel specific slot status indication

— **Test purpose**

Verify correct behaviour and independence of the channel specific slot status indication.

— **Applicability**

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 21.

Table 21 — Modification to basic configurations for static segment – channel specific slot status indication

Parameter	Modification
gMaxWithoutClockCorrectionPassive	10
gMaxWithoutClockCorrectionFatal	10

— **Preamble (setup state)**

Preamble I.

Deviating from the preamble the LT's payload data is 0x01 in byte 0 and 0x02 in byte 1 on channel A, 0x10 in byte 0 and 0x20 in byte 1 on channel B.

This test has to be performed in two instances.

For the instance 1 two receive buffers are configured for reception in slot 1, buffer n for channel A and buffer m for channel B. The slot status data for channel A and B are stored in the buffers n and m respectively.

For the instance 2 one receive buffer is configured for reception in slot 1 for both channels. The slot status data for channel A and for channel B are stored in this buffer.

— **Test execution**

1) In cycle 8, the LT simulates its frames correctly.

2) In the NIT of cycle 8, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel A and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

3) In cycle 9, the LT simulates its frame on channel A correctly but the frame on channel B with the frame ID set to 0 and adapted header CRC and frame CRC.

4) In the NIT of cycle 9, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = true, *vSS!BViolation* = false, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel A and
the syntax error indicator is false, the content error indicator is true, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

5) In cycle 10, the LT simulates its frame on channel B correctly but the frame on channel A with the frame ID set to 0 and adapted header CRC and frame CRC.

6) In the NIT of cycle 10, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = true, *vSS!BViolation* = false, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is false, the content error indicator is true, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false on channel A and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

7) In cycle 11, the LT simulates its frame on channel A correctly and no frame but a low phase of length 2 MT starting 1 MT before the slot boundary to slot 2 on channel B.

8) In the NIT of cycle 11, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = true, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel A and
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is true, the valid frame indicator is false and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

9) In cycle 12, the LT simulates its frame on channel B correctly and no frame but a low phase of length 2 MT starting 1 MT before the slot boundary to slot 2 on channel A.

10) In the NIT of cycle 12, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = true, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is true, the valid frame indicator is false and the additional communication indicator is false on channel A and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

11) In cycle 13, the LT simulates its frame on channel A correctly and its frame on channel B with a header CRC error (bit flip in the LSB of the header CRC).

12) In the NIT of cycle 13, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A,
vSS!SyntaxError = true, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel B,
and that the aggregated channel status provides:

the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel A and
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

13) In cycle 14, the LT simulates its frame on channel B correctly and its frame on channel A with a header CRC error (bit flip in the LSB of the header CRC).

14) In the NIT of cycle 14, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = true, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false on channel A and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

15) In cycle 15, the LT simulates its frames on both channels correctly and an additional low phase of length 2 MT starting 1 MT before the slot boundary to slot 2 on channel B.

16) In the NIT of cycle 15, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = true, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel A and
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is true, the valid frame indicator is true and the additional communication indicator is true on channel B.
The UT clears the aggregated channel status.

17) In cycle 16, the LT simulates its frames on both channels correctly and an additional low phase of length 2 MT starting 1 MT before the slot boundary to slot 2 on channel A.

18) In the NIT of cycle 16, it is verified (UT) that the slot status data of slot 1 is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = true, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is true, the valid frame indicator is true and the additional communication indicator is true on channel A and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

19) In cycle 17, the LT simulates idle in slot 1 on both channels.

20) In the NIT of cycle 17, it is verified (UT) that the slot status data is:
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel A,
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' for channel B,
and that the aggregated channel status provides:
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false on channel A and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false on channel B.
The UT clears the aggregated channel status.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag *vSS!SyntaxError*, the content error flag *vSS!ContentError*, the boundary violation flag *vSS!BViolation*, the valid frame flag *vSS!ValidFrame* and the null frame indicator flag *vSS!NFIndicator* in the slot status data are set accordingly. The syntax error indicator, the content error indicator, the valid frame indicator and the additional communication indicator in the aggregated channel status are set accordingly. A receive buffer configured for reception on both channels is capable to store the slot status data for each channel separately.

7.2.1.28 Modified BSS

— Test purpose

Verify the bit alignment at the falling edge of BSS and the correctness of bit strobing offset.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

Preamble III.

Deviating from the preamble the payload data is 0x50 in byte 0 and 0x00 in byte 1.

— Test execution

1) In cycle 9, the LT simulates its startup frame in slot 2 with the first high bit in byte 0 shifted by 5 / 16 *gdBit* to the frame start.

2) In cycle 9, it is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates its startup frame in slot 2 with a stretched high phase of length 10 / 8 *gdBit* in the first BSS of the payload segment and the first high bit in byte 0 shifted by 5 / 16 *gdBit* to the frame start.

4) In cycle 10, it is verified (UT) that the valid frame flag is false and the syntax error flag is true in the slot status data of slot 2 and that the valid frame indicator is true and the syntax error indicator is true in the aggregated channel status in the NIT of this cycle. Since the delayed falling edge of the BSS is used for bit alignment and the second payload bit is shifted to the frame start, a 1 is strobed instead of a 0 in the first payload bit and a 0 instead of a 1 in the second payload bit. This causes a frame CRC error.

5) In cycle 11, the LT simulates its startup frame in slot 2 with the first high bit in byte 0 shifted by 5 / 16 *gdBit* to the frame start.

6) In cycle 11, it is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slot 2 and in the aggregated channel status in the NIT of the cycle.

Figure 13 depicts the modified BSS.

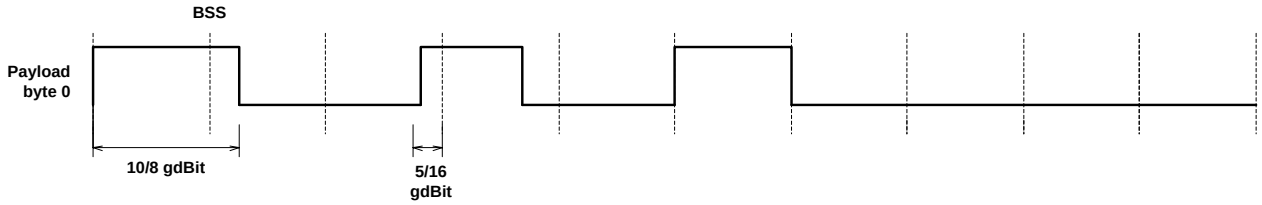


Figure 13 — Modified BSS

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator and the syntax error flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.1.29 Startup frame status for startup node

— **Test purpose**

Verify correct number of startup pairs for a startup node.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) In cycle 8, the LT simulates a startup frame in slot 3, a sync frame in slot 4 and a startup frame in slot 5.

2) In the symbol window of cycle 8, it is verified (UT) that the number of channel aligned startup pairs received or transmitted during the previous double cycle is 2 in the startup frame status (*vStartupPairs* = 2).

3) In cycle 9, the LT simulates a startup frame in slot 3, a startup frame in slot 4 and a sync frame in slot 5.

4) In the symbol window of cycle 9, it is verified (UT) that *vStartupPairs* = 2 in the startup frame status.

5) In cycle 9, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick / pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

6) In cycle 10, the LT simulates a sync frame in slot 3, a sync frame in slot 4 and a startup frame in slot 5.

7) In the symbol window of cycle 10, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

8) In cycle 10, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick / pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

9) In cycle 11, the LT simulates a sync frame in slot 3, a startup frame in slot 4 and a sync frame in slot 5.

10) In the symbol window of cycle 11, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

11) In cycle 11, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick / pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 2 in the startup frame status.

Figure 14 depicts the startup frame status for startup node.

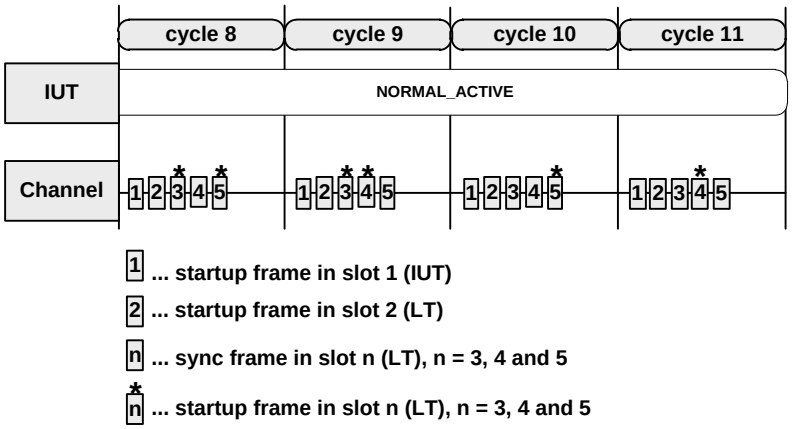


Figure 14 — Startup frame status for startup node

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The number of channel aligned startup pairs received or transmitted during the previous double cycle in the startup frame status is set accordingly.

7.2.1.30 Startup frame status for integrating node

— **Test purpose**

Verify correct number of startup pairs for an integrating node.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) In cycle 8, the LT simulates a startup frame in slot 3, a sync frame in slot 4 and a startup frame in slot 5.

2) In the symbol window of cycle 8, it is verified (UT) that the number of channel aligned startup pairs received or transmitted during the previous double cycle is 2 in the startup frame status (*vStartupPairs* = 2).

3) In cycle 9, the LT simulates a startup frame in slot 3, a startup frame in slot 4 and a sync frame in slot 5.

4) In the symbol window of cycle 9, it is verified (UT) that *vStartupPairs* = 2 in the startup frame status.

5) In cycle 9, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

6) In cycle 10, the LT simulates a sync frame in slot 3, a sync frame in slot 4 and a startup frame in slot 5.

7) In the symbol window of cycle 10, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

8) In cycle 10, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

9) In cycle 11, the LT simulates a sync frame in slot 3, a startup frame in slot 4 and a sync frame in slot 5.

10) In the symbol window of cycle 11, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

11) In cycle 11, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 2 in the startup frame status.

Figure 15 depicts the startup frame status for integrating node.

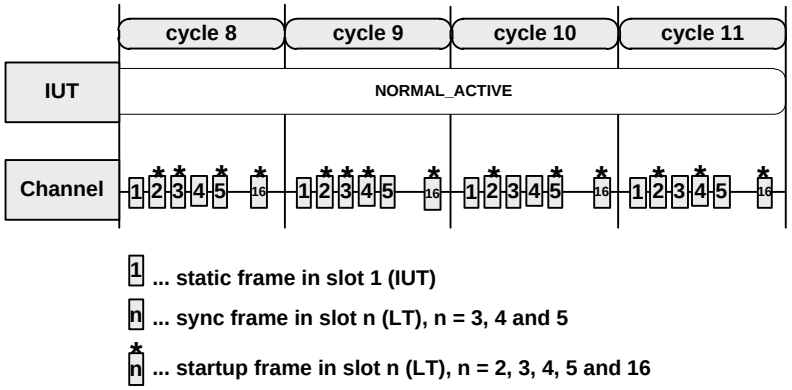


Figure 15 — Startup frame status for integrating node

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The number of channel aligned startup pairs received or transmitted during the previous double cycle in the startup frame status is set accordingly.

7.2.1.31 Slot status indication and slot status updated indicator

— Test purpose

Verify correct behaviour of slot status indication and slot status updated indicator.

— Applicability

SC.

— Configuration

All basic configurations using the modifications as listed in Table 22.

Table 22 — Modification to basic configurations for static segment – slot status indication and slot status updated indicator

Parameter	Modification
gMaxWithoutClockCorrectionPassive	10
gMaxWithoutClockCorrectionFatal	10

— Preamble (setup state)

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame correctly on the active channel.

2) In the dynamic segment of cycle 8, it is verified (UT) that
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true and *vSS!NFIndicator* = '1' in the slot status data of slot 1,
the slot status updated indicator is true in the slot status data of slot 1 and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true and the additional communication indicator is false in the aggregated channel status.
The UT clears the aggregated channel status and the slot status updated indicator in the slot status data of slot 1.

3) In the NIT of cycle 8, it is verified (UT) that the slot status updated indicator is false in the slot status data of slot 1.

4) In cycle 9, the LT simulates its frame with the frame ID set to 0 and adapted header CRC and frame CRC on the active channel.

5) In the dynamic segment of cycle 9, it is verified (UT) that
vSS!SyntaxError = false, *vSS!ContentError* = true, *vSS!BViolation* = false, *vSS!ValidFrame* = false and *vSS!NFIndicator* = '0' in the slot status data of slot 1
the slot status updated indicator is true in the slot status data of slot 1 and
the syntax error indicator is false, the content error indicator is true, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false in the aggregated channel status.
The UT clears the aggregated channel status and the slot status updated indicator in the slot status data of slot 1.

6) In the NIT of cycle 9, it is verified (UT) that the slot status updated indicator is false in the slot status data of slot 1.

7) In cycle 10, the LT simulates no frame but a low phase of length 2 MT starting 1 MT before the slot boundary to slot 2 on the active channel.

8) In the dynamic segment of cycle 10, it is verified (UT) that
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = true, *vSS!ValidFrame* = false and *vSS!NFIndicator* = '0' in the slot status data of slot 1,
the slot status updated indicator is true in the slot status data of slot 1 and
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is true, the valid frame indicator is false and the additional communication indicator is false in the aggregated channel status.
The UT clears the aggregated channel status and the slot status updated indicator in the slot status data of slot 1.

9) In the NIT of cycle 10, it is verified (UT) that the slot status updated indicator is false in the slot status data of slot 1.

10) In cycle 11, the LT simulates its frame with a header CRC error (bit flip in the LSB of the header CRC) on the active channel.

11) In the dynamic segment of cycle 11, it is verified (UT) that
vSS!SyntaxError = true, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = false and *vSS!NFIndicator* = '0' in the slot status data of slot 1,
the slot status updated indicator is true in the slot status data of slot 1 and
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false in the aggregated channel status.

The UT clears the aggregated channel status and the slot status updated indicator in the slot status data of slot 1.

12) In the NIT of cycle 11, it is verified (UT) that the slot status updated indicator is false in the slot status data of slot 1.

13) In cycle 12, the LT simulates its frame correctly and an additional low phase of length 2 MT starting 1 MT before the slot boundary to slot 2 on the active channel.

14) In the dynamic segment of cycle 12, it is verified (UT) that
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = true, *vSS!ValidFrame* = true and *vSS!NFIndicator* = '1' in the slot status data of slot 1,
the slot status updated indicator is true in the slot status data of slot 1 and
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is true, the valid frame indicator is true and the additional communication indicator is true in the aggregated channel status.
The UT clears the aggregated channel status and the slot status updated indicator in the slot status data of slot 1.

15) In the NIT of cycle 12, it is verified (UT) that the slot status updated indicator is false in the slot status data of slot 1.

16) In cycle 13, the LT simulates idle in slot 1.

17) In the dynamic segment of cycle 13, it is verified (UT) that
vSS!SyntaxError = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = false, *vSS!NFIndicator* = '0' in the slot status data of slot 1,
the slot status updated indicator is true in the slot status data of slot 1 and
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is false and the additional communication indicator is false in the aggregated channel status.
The UT clears the aggregated channel status and the slot status updated indicator in the slot status data of slot 1.

18) In the NIT of cycle 13, it is verified (UT) that the slot status updated indicator is false in the slot status data of slot 1.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag *vSS!SyntaxError*, the content error flag *vSS!ContentError*, the boundary violation flag *vSS!BViolation*, the valid frame flag *vSS!ValidFrame* and the null frame indicator flag *vSS!NFIndicator* in the slot status data and the slot status updated indicator in the message buffer status data are set accordingly. The syntax error indicator, the content error indicator, the valid frame indicator and the additional communication indicator in the aggregated channel status are set accordingly.

7.2.1.32 Channel alignment startup frame status for startup node

— **Test purpose**

Verify correct number of channel aligned startup pairs for a startup node.

— **Applicability**

DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) In cycle 8, the LT simulates a startup frame in slot 3 on channel A, a sync frame in slot 3 on channel B and a sync frame in slot 4 on both channels.

2) In the symbol window of cycle 8, it is verified (UT) that the number of channel aligned startup pairs received or transmitted during the previous double cycle is 2 in the startup frame status (*vStartupPairs* = 2).

3) In cycle 9, the LT simulates a startup frame in slot 3 on both channels and a sync frame in slot 4 on both channels.

4) In the symbol window of cycle 9, it is verified (UT) that *vStartupPairs* = 2 in the startup frame status.

5) In cycle 9, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

6) In cycle 10, the LT simulates a startup frame in slot 3 on both channels and a startup frame in slot 4 on both channels.

7) In the symbol window of cycle 10, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

8) In cycle 10, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

9) In cycle 11, the LT simulates a startup frame in slot 3 on channel B, a sync frame in slot 3 on channel A and a startup frame in slot 4 on both channels.

10) In the symbol window of cycle 11, it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

11) In cycle 11, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 4 in the startup frame status.

12) In cycle 12, the LT simulates a startup frame in slot 3 on both channels and a sync frame in slot 4 on both channels.

13) In the symbol window of cycle 12, it is verified (UT) that *vStartupPairs* = 4 in the startup frame status.

14) In cycle 12, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that *vStartupPairs* = 4 in the startup frame status.

15) In cycle 13, the LT simulates a startup frame in slot 3 on both channels and a sync frame in slot 4 on both channels.

16) In the symbol window of cycle 13, it is verified (UT) that *vStartupPairs* = 4 in the startup frame status.

17) In cycle 13, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT it is verified (UT) that *vStartupPairs* = 3 in the startup frame status.

Figure 16 depicts the channel aligned startup frame status for startup node.

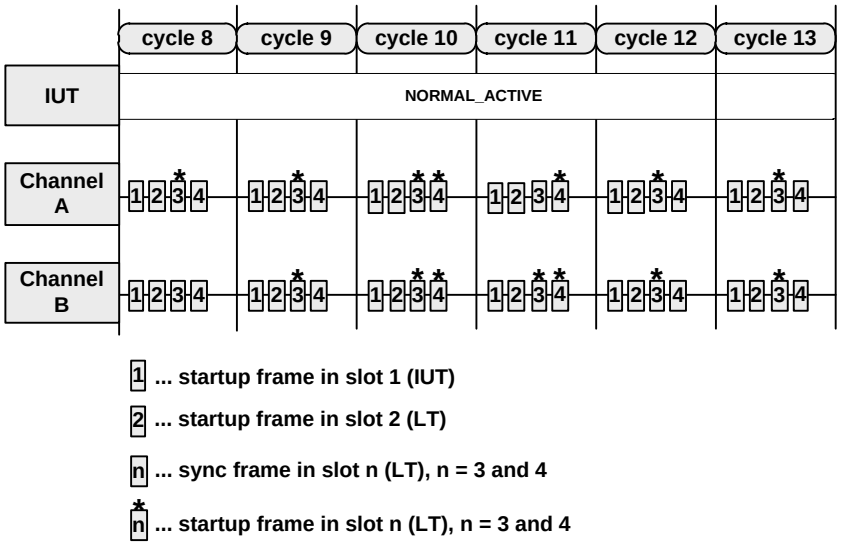


Figure 16 — Channel aligned startup frame status for startup node

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The number of channel aligned startup pairs received or transmitted during the previous double cycle in the startup frame status is set accordingly.

7.2.1.33 Receive valid frame within a transmission slot

— Test purpose

Verify correct handling of a received frame within a transmission slot. It is verified that the valid frame flag *vSS!ValidFrame* of the slot status data is false and the received frame is not considered for the accrued network management vector.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 23 and Table 24.

Table 23 — Modification to basic configurations 1a and 1b for static segment – receive valid frame within a transmission slot

Parameter	Modification to Basic Configurations			
	1a		1b	
	I	II	I	II
gdActionPointOffset [MT]	4	63	4	63
gdStaticSlot [MT]	399		399	
gdSymbolWindow [MT]	0			
gMacroPerCycle [MT]	2 037		2 037	
gNumberOfMinislots	0		0	
gNumberOfStaticSlots	5		5	
gPayloadLengthStatic [two-byte word]	1		1	
gdCycle [µs]	2 037		2 037	
gdMacrotick [µs]	1		1	
gdNIT [MT]	42		42	
pdListenTimeout [µT]	163 058		326 116	
pLatestTx [Minislot]	0		0	
pMicroPerCycle [µT]	81 480		162 960	
pOffsetCorrectionOut [µT]	153		226	
pOffsetCorrectionStart [MT]	2 019		2 019	
pRateCorrectionOut [µT]	49		98	
gNetworkManagementVectorLength [byte]	1		1	

Table 24 — Modification to basic configurations 2a, 2b and 3 for static segment – receive valid frame within a transmission slot

Parameter	Modification to Basic Configurations					
	2a		2b		3	
	I	II	I	II	I	II
gdActionPointOffset [MT]	3	63	3	63	3	63
gdStaticSlot [MT]	664		664		660	
gdSymbolWindow [MT]	0					
gMacroPerCycle [MT]	3 341		3 361		3 341	
gNumberOfMinislots	0		0		0	
gNumberOfStaticSlots	5		5		5	
gPayloadLengthStatic [two-byte word]	1		1		1	
gdCycle [µs]	6 682		6 722		6 682	
gdMacrotick [µs]	2		2		2	
gdNIT [MT]	21		41		41	
pdListenTimeout [µT]	534 882		269 042		267 442	
pLatestTx [Minislot]	0		0		0	
pMicroPerCycle [µT]	267 280		134 440		133 640	
pOffsetCorrectionOut [µT]	154		117		118	
pOffsetCorrectionStart [MT]	3 328		3 344		3 329	
pRateCorrectionOut [µT]	161		81		81	
gNetworkManagementVectorLength [byte]	1		1		1	

— **Preamble (setup state)**

Preamble II.

The IUT is configured to transmit a frame in the static slot 2 of cycle 7.

— **Test execution**

1) In cycle 7 the LT simulates a frame in slot 2

   (I) 200 MT after slot start with the payload preamble indicator set to 1 and a payload of 0x3333.

   (II) 10 MT after slot start with the payload preamble indicator set to 1 and a payload of 0x3333.

2) In the NIT of cycle 7, it is verified (UT) that the valid frame flag *vSS!ValidFrame* is false and the syntax error flag *vSS!SyntaxError* is true in the slot status data of slot 2.

3) In cycle 8, 500 µT after the cycle start and before the NIT, it is verified (UT) that the IUT provides an accrued network management vector of 0x00.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends a frame in a slot and receives within the same slot a frame. It is verified that the valid frame flag *vSS!ValidFrame* of the slot status is false and that the syntax error flag *vSS!SyntaxError* of the slot status is true and that the content of the IUT's network management vector is unaltered.

7.2.1.34 Aggregated channel status

— **Test purpose**

Verify the correct behaviour of the aggregated channel status.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 25.

Table 25 — Modification to basic configurations for static segment – aggregated channel status

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdStaticSlot [MT]	42	40	68
gNumberOfStaticSlots	70	37	21
gPayloadLengthStatic [two-byte word]	1		
gdNIT [MT]	49	17	33

— **Preamble (setup state)**

Preamble III.

Deviating from the Preamble the LT simulates additional frames in static slot 3 in cycle 9, slot 4 in cycle 10, slot 5 in cycle 11, slot 6 in cycle 12 and slot 7 in cycle 13. Reserved bit, payload preamble indicator, sync frame indicator and startup frame indicator of those frames are set to '0', null frame indicator to '1'. The IUT transmits an additional frame in slot 8 in cycle 14.

— **Test execution**

The aggregated channel status in not cleared in every cycle in the NIT as opposed to the general statements, for clearing the aggregated channel status the statements in the test execution shall be followed.

1) In cycle 8 the aggregated channel status is cleared (UT) on the respective channel (s).

2) In cycle 9, the LT simulates the default frame in slot 3 on the respective channel (s). It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is false, the transmission conflict indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 3 and that the aggregated channel status provides:

the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true, the transmission conflict indicator is false and the additional communication indicator is false and in the NIT of the cycle on the respective channel (s).

3) In cycle 10, the LT simulates frame 1 and frame 2 in slot 4 on the respective channel (s). The inter-frame gap is 2 * *cChannelIdleDelimiter*.
It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is true, the content error flag / indicator is false, the transmission conflict indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 4 and
that the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is true, the transmission conflict indicator is false and the additional communication indicator is true in the NIT of the cycle on the respective channel (s).

4) In cycle 11, the LT simulates the frame with the frame ID set to 0 in slot 5 and adapted header CRC and frame CRC on the respective channel (s). It is verified (UT) that the valid frame flag / indicator is false, the syntax error flag / indicator is false, the content error flag / indicator is true, the transmission conflict indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 5 and
that the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is true, the boundary violation indicator is false, the valid frame indicator is true, the transmission conflict indicator is false and the additional communication indicator is true in the NIT of the cycle on the respective channel (s).

5) In cycle 12 the LT simulates its frame on with a header CRC error (bit flip in the LSB of the header CRC) in slot 6 on the respective channel (s). It is verified (UT) that the valid frame flag / indicator is false, the syntax error flag / indicator is true, the content error flag / indicator is false, the transmission conflict indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 6 and
hat the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is true, the boundary violation indicator is false, the valid frame indicator is true, the transmission conflict indicator is false and the additional communication indicator is true in the NIT of the cycle on the respective channel (s).

6) In cycle 13, the LT simulates a correct frames in slot 7 and an additional low phase of length 2 MT starting 1 MT before the slot boundary to slot 8 on the respective channel (s). It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is false, the transmission conflict indicator is false and the boundary violation flag / indicator is true in the slot status data of slot 7 and hat the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is true, the boundary violation indicator is true, the valid frame indicator is true, the transmission conflict indicator is false and the additional communication indicator is true in the NIT of the cycle on the respective channel (s).

7) In cycle 14, the LT simulates additional low phase of length *gdActionpointOffset* + 2MT starting 1MT after the slot boundary to slot 8 on the respective channel (s). The IUT transmits a frame in slot 8 at the action point offset on the respective channel (s).
It is verified (UT) that the valid frame flag / indicator is false, the syntax error flag / indicator is false, the content error flag / indicator is false, the transmission conflict indicator is true and the boundary violation flag / indicator is false in the slot status data of slot 8 and
that the aggregated channel status provides:
the syntax error indicator is true, the content error indicator is true, the boundary violation indicator is true, the valid frame indicator is true, the transmission conflict indicator is true and the additional communication indicator is true in the NIT of the cycle on the respective channel (s).

8) In cycle 15 after slot 20 and before slot 30 the aggregated channel status is cleared (UT) on the respective channel (s).

9) In the NIT of cycle 15, it is verified that the aggregated channel status provides:
the syntax error indicator is false, the content error indicator is false, the boundary violation indicator is false, the valid frame indicator is false, the transmission conflict indicator is false and the additional communication indicator is false on the respective channel (s).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator, the syntax error flag / indicator, the content error flag / indicator, transmission conflict indicator and the boundary violation flag / indicator in the slot status data and in the aggregated channel status and the additional communication indicator in the aggregated channel status are set accordingly

7.2.2 Dynamic segment

7.2.2.1 TSS

— Test purpose

Verify recognition of the transmission start sequence TSS in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 26, Table 27 and Table 28.

Table 26 — Modification to basic configurations 1a and 1b for dynamic segment – TSS

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	77	97	105	77	97	105
gdTSSTransmitter [gdBit]	1[a]	11	15	1[a]	11	15
pDecodingCorrection [µT]	12	52	68	24	104	136
pMacroInitialOffset[A,B] [MT]	5	6		5	6	
pMicroInitialOffset[A,B] [µT]	23		7	46		14

[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation.

Table 27 — Modification to basic configurations 2a and 2b for dynamic segment – TSS

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	70	80	84	70	80	84
gdTSSTransmitter [gdBit]	1[a]	6	8	1[a]	6	8
pDecodingCorrection [µT]	24	64	80	12	32	40
pMacroInitialOffset[A,B] [MT]	4		5	4		5
pMicroInitialOffset[A,B] [µT]	46	6	70	23	3	35
[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation.						

Table 28 — Modification to basic configuration 3 for dynamic segment – TSS

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	66	72	74
gdTSSTransmitter [gdBit]	1[a]	4	5
pDecodingCorrection [µT]	24	48	56
pMacroInitialOffset[A,B] [MT]	4	5	
pMicroInitialOffset[A,B] [µT]	6	22	14
[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation.			

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) The LT simulates a frame in dynamic slot n_{dyn} with a TSS length of

   — 3 / 8 [*gdBit*] in cycle 8,

   — 1 [*gdBit*] in cycle 9,

   — *gdTSSTransmitter* [*gdBit*] in cycle 10,

   — *gdTSSTransmitter* + 2 + 3 / 8 [*gdBit*] in cycle 11, and

   — *gdTSSTransmitter* + 3 [*gdBit*] in cycle 12.

2) In the NIT of the cycle 8, it is verified (UT) that the syntax error flag / indicator is true and that the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status.

3) In the NIT of cycles 9, 10 and 11, it is verified (UT) that the syntax error flag / indicator is false and that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status.

4) In the NIT of the cycle 12, it is verified (UT) that the syntax error flag / indicator is true and that the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the valid frame flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.2.2 Reserved bit

— **Test purpose**

Verify reception of the reserved bit in the dynamic segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with the reserved bit set to '0' and it is verified (UT) that the reserved bit *vRF!Header!Reserved* is '0' in the frame contents data in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the reserved bit set to '1' and it is verified (UT) that the reserved bit *vRF!Header!Reserved* is '1' in the frame contents data in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the reserved bit set to '0' and it is verified (UT) that the reserved bit *vRF!Header!Reserved* is '0' in the frame contents data in the NIT of this cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The reserved bit *vRF!Header!Reserved* in the frame contents data is set accordingly.

7.2.2.3 Payload preamble indicator

— Test purpose

Verify reception of the payload preamble indicator in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with the payload preamble indicator set to '0' and it is verified (UT) that the payload preamble indicator *vRF!Header!PPIndicator* is '0' in the frame contents data in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the payload preamble indicator set to '1' and it is verified (UT) that the payload preamble indicator *vRF!Header!PPIndicator* is '1' in the frame contents data in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the payload preamble indicator set to '0' and it is verified (UT) that the payload preamble indicator *vRF!Header!PPIndicator* is '0' in the frame contents data in the NIT of this cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The payload preamble indicator *vRF!Header!PPIndicator* in the frame contents data is set accordingly.

7.2.2.4 Null frame indicator

— Test purpose

Verify reception of the null frame indicator in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with the null frame indicator set to '1'. It is verified (UT) that the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot n_{dyn} in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 8.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the null frame indicator set to '0' and all payload bytes set to 0x00. It is verified (UT) that the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status and that the flag *vSS!NFIndicator* is '0' in the slot status data of slot n_{dyn} in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 8.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the null frame indicator set to '1'. It is verified (UT) that the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot n_{dyn} in the NIT of this cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 10.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The flag *vSS!NFIndicator* in the slot status data is set accordingly. The content error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status is set accordingly. The IUT does not update frame contents data upon reception of a frame with content error.

7.2.2.5 Sync frame Indicator

— **Test purpose**

Verify reception of the sync frame indicator in the dynamic segment. Sync frame reception in the dynamic segment should cause a content error.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with the sync frame indicator set to '1' and matching header CRC.
It is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* is '0' (holds its initial value) in the frame contents data of slot n_{dyn} and that the content error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the sync frame indicator set to '0' and matching header CRC.
It is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* is '0' in the frame contents data of slot n_{dyn} and that the content error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the sync frame indicator set to '1' and matching header CRC.
It is verified (UT) that the sync frame indicator *vRF!Header!SyFIndicator* is '0' (holds the value from the previous cycle) in the frame contents data of slot n_{dyn} and that the content error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The content error flag / indicator in the slot status data and in the aggregated channel status and the sync frame indicator *vRF!Header!SyFIndicator* in the frame contents data are set accordingly.

7.2.2.6 Startup frame indicator

— **Test purpose**

Verify reception of the startup frame indicator in the dynamic segment. Startup frame reception in the dynamic segment should cause a content error.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with the startup frame indicator set to '1' and matching header CRC. It is verified (UT) that the startup frame indicator *vRF!Header!SuFIndicator* is '0' (holds its initial value) in the frame contents data of slot n_{dyn} and the content error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the startup frame indicator set to '0' and matching header CRC. It is verified (UT) that the startup frame indicator *vRF!Header!SuFIndicator* is '0' in the frame contents data of slot n_{dyn} and the content error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the startup frame indicator set to '1' and matching header CRC. It is verified (UT) that the startup frame indicator *vRF!Header!SuFIndicator* is '0' (holds the value from the previous cycle) in the frame contents data of slot n_{dyn} and the content error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The content error flag / indicator in the slot status data and in the aggregated channel status and the startup frame indicator *vRF!Header!SuFIndicator* in the frame contents data are set accordingly.

7.2.2.7 Frame ID

— **Test purpose**

Verify reception of the frame ID in the dynamic segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} and matching header CRC.
It is verified (UT) that the frame ID *vRF!Header!FrameID* is n_{dyn} in the frame contents data of slot n_{dyn}, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to 0 and matching header CRC. It is verified (UT) that the frame ID *vRF!Header!FrameID* is n_{dyn} (holds the value from cycle 8) in the frame contents data of slot n_{dyn}, the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} + 1 and matching header CRC. It is verified (UT) that the frame ID *vRF!Header!FrameID* is n_{dyn} (holds the value from cycle 8) in the frame contents data of slot n_{dyn}, the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

4) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} and matching header CRC. It is verified (UT) that the frame ID *vRF!Header!FrameID* is n_{dyn} in the frame contents data of slot n_{dyn}, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The content error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the frame ID *vRF!Header!FrameID* in the frame contents data are set accordingly.

7.2.2.8 Payload length

— Test purpose

Verify reception of payload length in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with a payload length equal to 1. It is verified (UT) that the length field *vRF!Header!Length* equals 1 in the frame contents data of slot n_{dyn} and that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with a payload length equal to *cPayloadLengthMax* or 76 two-byte word for Basic Configuration 3. It is verified (UT) that the length field *vRF!Header!Length* in the frame contents data of slot n_{dyn} equals *cPayloadLengthMax* (Basic Configuration 1a, 1b, 2a, 2b) or 76 (Basic Configuration 3) two-byte word and that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with a payload length equal to ceil(*cPayloadLengthMax* / 2). It is verified (UT) that the length field *vRF!Header!Length* in the frame contents data of slot n_{dyn} equals ceil(*cPayloadLengthMax* / 2) and that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator in the slot status data and in the aggregated channel status and the length field *vRF!Header!Length* in the frame contents data are set accordingly.

7.2.2.9 Cycle count

— **Test purpose**

Verify reception of the cycle counter in the dynamic segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with the cycle counter set to 8. It is verified (UT) that the cycle counter *vRF!Header!CycleCount* is 8 in the frame contents data of slot n_{dyn}, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the cycle counter set to 7. It is verified (UT) that the cycle counter *vRF!Header!CycleCount* is 8 (holds the value from the previous cycle) in the frame contents data of slot n_{dyn}, the content error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the cycle counter set to 10. It is verified (UT) that the cycle counter *vRF!Header!CycleCount* is 10 in the frame contents data of slot n_{dyn}, the content error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The content error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the cycle counter *vRF!Header!CycleCount* in the frame contents data are set accordingly.

7.2.2.10 Header CRC

— Test purpose

Verify reception of the header CRC in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with a correct header CRC. It is verified (UT) that the header CRC *vRF!Header!HeaderCRC* in the frame contents data of slot n_{dyn} equals the header CRC sent by the LT in slot n_{dyn} of cycle 8 and that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with a wrong header CRC (bit flip in the LSB of the header CRC). It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with a correct header CRC. It is verified (UT) that the header CRC *vRF!Header!HeaderCRC* in the frame contents data of slot n_{dyn} equals the header CRC sent by the LT in slot n_{dyn} of cycle 10 and that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of this cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator in the slot status data and in the aggregated channel status and the header CRC *vRF!Header!HeaderCRC* in the frame contents data are set accordingly.

7.2.2.11 FSS / BSS

— Test purpose

Verify recognition of frame start sequence FSS and first byte start sequence BSS in the dynamic segment.

— Applicability

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1

— **Preamble (setup state)**

Preamble I.

— **Test execution**

The LT simulates an additional frame in dynamic slot n_{dyn} with the reserved bit, the payload preamble indicator, the sync frame indicator and the startup frame indicator set to '0' and the null frame indicator set to '1'.

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 1-0. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 1-1-0. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 1-1-1-0. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

4) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 1-1-1-1-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

5) In cycle 12, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS as a single bit set to 0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

6) In cycle 13, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS as a single bit set to 1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

7) In cycle 14, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 0-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

8) In cycle 15, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 1-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

9) In cycle 16, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 1-0-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

10) In cycle 17, the LT simulates a frame in dynamic slot n_{dyn} with FSS / BSS being the bit sequence 1-1-0-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status.

7.2.2.12 Payload

— Test purpose

Verify reception of payload data in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

The LT's payload data in slot n_{dyn} is 0x01 in byte 0, 0x02 in byte 1, ..., 0xFE in byte 253.

— Preamble (setup state)

Preamble I.

— Test execution

If the IUT is not able to receive the whole payload of the frame due to size restrictions of receive buffers, the received bytes – starting with byte 0 – are compared to the transmitted ones. Bytes at the end of the frame exceeding the buffer size may be dropped.

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with a payload of 0 two-byte words. In the NIT of the cycle it is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with a payload of 1 two-byte words. In the NIT of the cycle it is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status and that the received payload data *vRF!Payload* in the frame contents data equals the one transmitted by the LT.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with a payload of 32 two-byte words. In the NIT of the cycle it is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status and that the received payload data *vRF!Payload* in the frame contents data equals the one transmitted by the LT.

4) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} with a payload of 64 two-byte words. In the NIT of the cycle it is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status and that the received payload data *vRF!Payload* in the frame contents data equals the one transmitted by the LT.

5) In cycle 12, the LT simulates a frame in dynamic slot n_{dyn} with a payload of 127 (Basic Configuration 1a, 1b, 2a, 2b) or 76 (Basic Configuration 3) two-byte word. In the NIT of the cycle it is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status and that the received payload data *vRF!Payload* in the frame contents data equals the one transmitted by the LT.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status and the received payload equals the payload transmitted by the LT.

7.2.2.13 BSS

— Test purpose

Verify recognition of byte start sequence BSS in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates a frame in dynamic slot n_{dyn} with:

   (a) the BSS after the frame ID in the header segment,

   (b) the BSS after the first data byte Data 0 in the payload segment and

   (c) the BSS after the first CRC byte in the trailer segment

   being the bit sequence 1-0. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot ndyn and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with:

   (a) the BSS after the frame ID in the header segment,

   (b) the BSS after the first data byte Data 0 in the payload segment and

   (c) the BSS after the first CRC byte in the trailer segment

   being the bit sequence 0-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot ndyn and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with:

   (a) the BSS after the frame ID in the header segment,

   (b) the BSS after the first data byte Data 0 in the payload segment and

(c) the BSS after the first CRC byte in the trailer segment

being the bit sequence 0-0. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot ndyn and in the aggregated channel status in the NIT of the cycle.

4) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} with:

(a) the BSS after the frame ID in the header segment,

(b) the BSS after the first data byte Data 0 in the payload segment and

(c) the BSS after the first CRC byte in the trailer segment

being the bit sequence 1-1. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot ndyn and in the aggregated channel status in the NIT of the cycle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status.

7.2.2.14 Frame CRC

— Test purpose

Verify reception of frame CRC in the dynamic segment.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates its frame in dynamic slot n_{dyn} correctly. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates its frame in dynamic slot n_{dyn} with a bit flip in the first bit of the payload data within the frame without adaption of the frame CRC. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates its frame in dynamic slot n_{dyn} with a bit flip in the first bit of the frame CRC within the frame. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

4) In cycle 11, the LT simulates its frame in dynamic slot n_{dyn} correctly. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status.

7.2.2.15 FES

— **Test purpose**

Verify recognition of frame end sequence FES in the dynamic segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frame in dynamic slot n_{dyn} with FES being the bit sequence 0-1. It is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 8.

2) In cycle 9, the LT simulates its frame in dynamic slot n_{dyn} with FES being the bit sequence 1-0. It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 8.

3) In cycle 10, the LT simulates its frame in dynamic slot n_{dyn} with FES being the bit sequence 0-0. It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 10.

4) In cycle 11, the LT simulates its frame in dynamic slot n_{dyn} with FES being the bit sequence 1-1. It is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 10.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and valid frame flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.2.16 DTS

— **Test purpose**

Verify recognition of the dynamic trailing sequence DTS in the dynamic segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit a dynamic frame in dynamic slot n_{dyn} + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates dynamic frames in slots n_{dyn} and n_{dyn} + 2. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slots n_{dyn} and n_{dyn} + 2 and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmits its frame in dynamic slot n_{dyn} + 1 correctly.

2) In cycle 9, the LT simulates dynamic frames in slots n_{dyn} and n_{dyn} + 2 with an incorrect DTS ending at the next but one minislot action point following the FES, i.e. with an DTS elongated by 1 *gdMinislot* in comparison to the DTS in cycle 8. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slots n_{dyn} and n_{dyn} + 2 and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmits its frame in dynamic slot n_{dyn} + 1 correctly and that the IUT's transmission starts 1 *gdMinislot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT later than in cycle 8.

3) In cycle 10, the LT simulates dynamic frames in slots n_{dyn} and n_{dyn} + 2 with an incorrect DTS ending 3 *gdBit* before the minislot start following the minislot action point after the FES, i.e. with an DTS elongated by ((*gdMinislot* – *gdMinislotActionPointOffset*) * *gdMacrotick* / *gdBit* – 3) [*gdBit*] in comparison to the DTS in cycle 8. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slots n_{dyn} and n_{dyn} + 2 and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmits its frame in dynamic slot n_{dyn} + 1 correctly and that the IUT's transmission starts as in cycle 8 with a deviation of 0 + ξ + ξ_{IUT} µT.

4) In cycle 11, the LT simulates dynamic frames in slots n_{dyn} and n_{dyn} + 2 with an incorrect DTS ending 3 *gdBit* after the minislot start following the minislot action point after the FES, i.e. with an DTS elongated by ((*gdMinislot* – *gdMinislotActionPointOffset*) * *gdMacrotick* / *gdBit* + 3) *gdBit* in comparison to the DTS in cycle 8. It is verified (UT) that the valid frame flag / indicator is true and the

syntax error flag / indicator is false in the slot status data of slots n_{dyn} and n_{dyn} + 2 and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmits its frame in dynamic slot n_{dyn} + 1 correctly and that the IUT's transmission starts 1 *gdMinislot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT later than in cycle 8.

5) In cycle 12, the LT simulates dynamic frames in slots n_{dyn} and n_{dyn} + 2. It is verified (UT) that the valid frame flag / indicator is true and the syntax error flag / indicator is false in the slot status data of slots n_{dyn} and n_{dyn} + 2 and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmits its frame in dynamic slot n_{dyn} + 1 correctly.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator and the syntax error flag / indicator are set accordingly in the slot status data and in the aggregated channel status.

7.2.2.17 MTS

— **Test purpose**

Verify the recognition of an MTS in the dynamic segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a valid frame in dynamic slot n_{dyn}. 500 µT after start of the NIT in this cycle and before the next dynamic segment, it is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.

2) In cycle 9, the LT simulates a valid MTS (low phase of length *gdTSSTransmitter* + *cdCAS*) in dynamic slot n_{dyn} starting at the minislot action point. 500 µT after start of the NIT in this cycle and before the next dynamic segment, it is verified (UT) that the syntax error flag / indicator is true and the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.

3) In cycle 10, the LT simulates a valid frame in dynamic slot n_{dyn}. 500 µT after start of the NIT in this cycle and before the next dynamic segment, it is verified (UT) that the syntax error flag / indicator is false and the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the valid frame flag / indicator in the slot status data and in the aggregated channel status and the flag *vSS!ValidMTS* in the symbol window status are set accordingly.

7.2.2.18 WUP

— **Test purpose**

Verify the recognition of a WUP in the dynamic segment.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a valid frame in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* and that the wakeup pattern received indicator is false in the wakeup and startup status in the NIT of the cycle.

2) In cycle 9, the LT simulates a valid WUP (consisting of 2 WUS) starting at the minislot action point of dynamic slot n_{dyn}. The low phase has a length of *gdWakeupRxLow* [*gdBit*], the high phase a length of *gdWakeupRxIdle* [*gdBit*]. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* and that the wakeup pattern received indicator is false in the wakeup and startup status in the NIT of the cycle.

3) In cycle 10, the LT simulates a valid frame in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* and that the wakeup pattern received indicator is false in the wakeup and startup status in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator in the slot status data and in the aggregated channel status, the wakeup status *vPOC!WakeupStatus* and the wakeup pattern received indicator are set accordingly.

7.2.2.19 Complete valid frame

— Test purpose

Verify frame reception in the dynamic segment.

— Applicability

SC, DC.

— Configuration

Table 29 defines the modification to basic configurations for *dynamic segment* – complete valid frame.

Table 29 — Modification to basic configurations for dynamic segment – complete valid frame

Parameter	Modification
gMaxWithoutClockCorrectionPassive	2
gMaxWithoutClockCorrectionFatal	3
pAllowHaltDueToClock	false

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8 to 13, the LT simulates a frame in dynamic slot n_{dyn}. It is verified (UT) that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9 to 13, the LT stops transmission of its startup frame in slot 1.

3) In cycle 12, 500 µT after cycles start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = NORMAL_PASSIVE).

4) In the NIT of cycle 12 it is verified (UT) that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator is set accordingly in the slot status data and in the aggregated channel status.

7.2.2.20 Complete valid frame too late reaching symbol window

— Test purpose

Verify frame reception in the dynamic segment with prolonged DTS or too long frame payload reaching the symbol window.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 30.

Table 30 — Modification to basic configurations for dynamic segment – complete valid frame too late reaching symbol window

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
pLatestTx [Minislot]	215	135	136
pPayloadLengthDynMax [two-byte word]	8		

n_{dyn} = *gNumberOfStaticSlots* + *pLatestTx.*

— Preamble (setup state)

Preamble III.

— Test execution

The LT simulates its frame in slot n_{dyn} with the payload data 0x01 in byte 0, 0x02 in byte 1, ..., 0x20 in byte 31 and the reserved bit, the payload preamble indicator, the sync frame indicator and the startup frame indicator set to '0', and the null frame indicator set to '1'.

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} holding 8 two-byte words of payload and a DTS ending at the action point of minislot *gNumberOfMinislots – gdDynamicSlotIdlePhase*.
   500 µT after start of the NIT in this cycle and before the next dynamic segment, it is verified (UT)
   that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status,
   that the syntax error flag / indicator is false in the slot status data of slot n_{dyn}, in the symbol window status and in the aggregated channel status and
   that the boundary violation flag / indicator is false in the slot status data of slot n_{dyn}, in the symbol window status and in the aggregated channel status.
   The UT clears the aggregated channel status after its verification.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} holding 16 two-byte words of payload, i.e. the frame crosses the boundary to the symbol window and ends within the symbol window.
   500 µT after start of the NIT in this cycle and before the next dynamic segment, it is verified (UT)
   that the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status,
   that the syntax error flag / indicator is false in the slot status data of slot n_{dyn} , in the symbol window status and in the aggregated channel status and
   that the boundary violation flag / indicator is true in the slot status data of slot n_{dyn}, in the symbol window status and in the aggregated channel status.
   It is also verified (UT) that the frame contents data corresponds to the contents of the frame as

simulated by the LT in slot n_{dyn} of the last cycle.
The UT clears the aggregated channel status after its verification.

3) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} holding 8 two-byte words of payload with a DTS ending *gdMinislot* – 5 * gdBit after the slot start of the last minislot causing a violation of the boundary between dynamic segment and symbol window during channel idle recognition.
500 µT after start of the NIT in this cycle and before the next dynamic segment, it is verified (UT)
that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status,
that the syntax error flag / indicator is false (frame decoding finished in the dynamic segment) in the slot status data of slot n_{dyn}, in the symbol window status and in the aggregated channel status and
that the boundary violation flag / indicator is true in the slot status data of slot n_{dyn}, in the symbol window status and in the aggregated channel status.
It is also verified (UT) that the frame contents data corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 11.
The UT clears the aggregated channel status after its verification.

4) In cycle 12, the LT simulates a frame in dynamic slot n_{dyn} holding 8 two-byte words of payload and a DTS ending at the action point of minislot *gNumberOfMinislots* – *gdDynamicSlotIdlePhase*.
500 µT after start of the NIT in this cycle and before the next dynamic segment, it is verified (UT)
that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status,
that the syntax error flag / indicator is false in the slot status data of slot n_{dyn}, in the symbol window status and in the aggregated channel status and
that the boundary violation flag / indicator is false in the slot status data of slot n_{dyn} , in the symbol window status and in the aggregated channel status.
It is also verified (UT) that the frame contents data corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 12.
The UT clears the aggregated channel status after its verification.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator in the slot status data and in the aggregated channel status, the boundary violation flag / indicator and the syntax error flag / indicator in the slot status data, in the symbol window status and in the aggregated channel status are set accordingly. The IUT shall not update the frame contents data of the active buffer, if the valid frame flag in the slot status data is false.

7.2.2.21 Complete valid frame too late reaching NIT

— **Test purpose**

Verify frame reception in the dynamic segment with prolonged DTS or too long frame payload reaching the network idle time.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 31.

Table 31 — Modification to basic configurations for dynamic segment – complete valid frame too late reaching NIT

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
pLatestTx [Minislot]	215	135	136
pPayloadLengthDynMax [two-byte word]	8		
gdSymbolWindow [MT]	0		
gdNIT [MT]	54	35	35

n_{dyn} = *gNumberOfStaticSlots* + *pLatestTx*

— **Preamble (setup state)**

Preamble III.

— **Test execution**

The LT simulates its frame in slot n_{dyn} with the payload data 0x01 in byte 0, 0x02 in byte 1, ..., 0x20 in byte 31 and the reserved bit, the payload preamble indicator, the sync frame indicator and the startup frame indicator set to '0', and the null frame indicator set to '1'.

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} holding 8 two-byte words of payload and a DTS ending at the action point of minislot *gNumberOfMinislots – gdDynamicSlotIdlePhase*.
500 µT after start of the next cycle and before the dynamic segment, it is verified (UT)
that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status,
that the syntax error flag / indicator is false in the slot status data of slot n_{dyn}, in the NIT status and in the aggregated channel status and
that the boundary violation flag / indicator is false in the slot status data of slot n_{dyn} , in the NIT status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} holding 16 two-byte words of payload, i.e. the frame crosses the boundary to the NIT and ends within the NIT.
500 µT after start of the next cycle and before the dynamic segment, it is verified (UT)
that the valid frame flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status,
that the syntax error flag / indicator is false in the slot status data of slot n_{dyn}, in the NIT status and in the aggregated channel status and
that the boundary violation flag / indicator is true in the slot status data of slot n_{dyn}, in the NIT status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

3) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} holding 8 two-byte words of payload with a DTS ending *gdMinislot* – 5 * gdBit after the slot start of the last minislot causing a violation of the boundary between the dynamic segment and the NIT during channel idle recognition.
500 µT after start of the next cycle and before the dynamic segment, it is verified (UT)
that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status,
that the syntax error flag / indicator is false in the slot status data of slot n_{dyn}, in the NIT status and in the aggregated channel status and
that the boundary violation flag / indicator is true in the slot status data of slot n_{dyn}, in the NIT status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

4) In cycle 12, the LT simulates a frame in dynamic slot n_{dyn} holding 8 two-byte words of payload and a DTS ending at the action point of minislot *gNumberOfMinislots* – *gdDynamicSlotIdlePhase*.

5) In cycle 13 within 500 µT after start and before the dynamic segment, it is verified (UT)
that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status,
that the syntax error flag / indicator is false in the slot status data of slot n_{dyn}, in the NIT status and in the aggregated channel status and
that the boundary violation flag / indicator is false in the slot status data of slot n_{dyn} , in the NIT status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator in the slot status data and in the aggregated channel status, the boundary violation flag / indicator and the syntax error flag / indicator in the slot status data, in the NIT status and in the aggregated channel status are set accordingly.

7.2.2.22 Noise

— Test purpose

Verify recognition of noise in a dynamic slot.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, the LT simulates idle in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of dynamic slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.
The UT clears the aggregated channel status after its verification.

2) In cycle 10, the LT simulates noise (low phase of length *gdSampleClockPeriod*) in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of dynamic slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.
The UT clears the aggregated channel status after its verification.

3) In cycle 11, the LT simulates noise (low phase of length 2 * *gdSampleClockPeriod*) in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of dynamic slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.
The UT clears the aggregated channel status after its verification.

4) In cycle 12, the LT simulates noise (low phase of length 3 * *gdSampleClockPeriod*) in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of dynamic slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.
The UT clears the aggregated channel status after its verification.

5) In cycle 13, the LT simulates noise (low phase of length (*cStrobeOffset* + 1) * *gdSampleClockPeriod*) in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of dynamic slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.
The UT clears the aggregated channel status after its verification.

6) In cycle 14, the LT simulates noise (low phase of length (*gdTSSTransmitter* + 2) * *gdBit*) in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is true in the slot status data of dynamic slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.
The UT clears the aggregated channel status after its verification.

7) In cycle 15, the LT simulates idle in dynamic slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the slot status data of dynamic slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.
The UT clears the aggregated channel status after its verification.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator is set accordingly in the slot status data and in the aggregated channel status.

7.2.2.23 Channel idle after transmission

— Test purpose

Verify the length of the channel idle delimiter. The IUT shall be able to detect an incoming communication element earliest *cChannelIdleDelimiter* after the end of its own transmission.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 32.

Table 32 — Modification to basic configurations for dynamic segment – channel idle after transmission

Parameter	Modification		
	I	II	III
gdIgnoreAfterTx [gdBit]	0	7	15

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit a frame in the dynamic slot n_{dyn} with the payload 0xAA in byte 1 and 0x55 in byte 2.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates a low phase of length *gdBit* after a period of *cChannelIdleDelimiter* – 1 [*gdBit*] past the last bit of the DTS in the IUT's frame in slot n_{dyn}. It is verified (UT) that the syntax error flag / indicator is false in the transmit buffer status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates a low phase of length *gdBit* after a period of *cChannelIdleDelimiter* [*gdBit*] past the last bit of the DTS in the IUT's frame in slot n_{dyn}. It is verified (UT) that

   (I, II) the syntax error flag / indicator is true and

   (III) the syntax error flag / indicator is false

   in the transmit buffer status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

3) In cycle 10, the LT simulates a low phase of length *gdBit* after a period of 3 *gdBit* and a second low phase of length *gdBit* after a period of *cChannelIdleDelimiter* [*gdBit*] past the last bit of the DTS in the IUT's frame in slot n_{dyn}.

4) In cycle 10, the LT simulates an additional frame in dynamic slot n_{dyn} + 1.

5) In the NIT of cycle 10, it is verified (UT) that

   (I) the syntax error flag / indicator is false,

   (II) the syntax error flag / indicator is true and

   (III) the syntax error flag / indicator is false

   in the transmit buffer status of slot n_{dyn} and in the aggregated channel status, that the frame transmitted indicator is true in the transmit buffer status. In addition, it is verified (UT) that the valid frame flag is true in the slot status data of slot n_{dyn} + 1 that the frame contents data of slot n_{dyn} + 1corresponds to the contents of the frame as simulated by the LT.

6) In cycle 11, the LT simulates an additional frame in dynamic slot n_{dyn} + 1.

7) In the NIT of cycle 11, it is verified (UT) the boundary violation flag / indicator is false in the transmit buffer status of slot n_{dyn} and in the aggregated channel status and that the valid frame flag is true in the slot status data of slot n_{dyn} + 1.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator in the transmit buffer status and in the aggregated channel status, the frame transmitted indicator in the transmit buffer status, and the valid frame flag in the slot status data a are set accordingly and the received frame contents are as expected.

7.2.2.24 Dynamic frame starting in the dynamic segment offset

— Test purpose

Verify correct reception of a dynamic frame starting in the dynamic segment offset.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 33.

Table 33 — Modification to basic configurations for dynamic segment – dynamic frame starting in the dynamic segment offset

Parameter	Modification to Basic Configurations			
	1a & 1b	**2a**	**2b**	**3**
gdActionPointOffset [MT]	7	6	6	6
gdStaticSlot [MT]	38	35	37	61
gNumberOfStaticSlots	78	42	40	23
gdNIT [MT]	22	24	14	55
pMacroInitialOffset[A,B] [MT]	9	7	7	8

n_{dyn} = *gNumberOfStaticSlots* + 1.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates an additional frame in dynamic slot n_{dyn}. It is verified (UT) that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 9, the LT simulates an additional frame in dynamic slot n_{dyn} 2 MT after the end of the last static slot. It is verified (UT) that the valid frame flag / indicator is true, the syntax error flag / indicator is false, the content error flag / indicator is false and the boundary violation flag / indicator is false in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is also verified (UT) that the frame contents data of slot n_{dyn} corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 9.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid fame flag / indicator, the syntax error flag / indicator, the content error flag / indicator, and the boundary violation flag / indicator in the slot status data and in the aggregated channel status are set accordingly and the received frame contents are as expected.

7.2.2.25 Noise before transmitted dynamic frame

— Test purpose

Verify correct handling of noise before transmitting a frame in the dynamic segment. Noise on the bus occurs before the frame transmission in the same dynamic slot.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit a frame in the dynamic slot n_{dyn} with a payload of 0xAA in byte 1 and 0x55 in byte 2.

— Preamble (setup state)

Preamble I.

— Test execution

1) In slot n_{dyn} of cycle 8, 5 *gdBit* before the IUT begins its frame transmission the LT simulates a low phase of length 1 *gdBit*.

2) In slot n_{dyn} + 1 of cycle 8, the LT simulates a frame with a payload length of 1 two-byte word and the payload data 0x55 in byte 1 and 0xAA in byte 2.

3) In the NIT of cycle 8, it is verified (UT) that the frame transmitted indicator is true, the transmission conflict flag is true and the syntax error flag is true in the transmit buffer status of slot n_{dyn}. It is also verified (UT) that the valid frame flag is true in the slot status data of slot n_{dyn} + 1 and that the frame contents data of slot n_{dyn} + 1 corresponds to the contents of the frame as simulated by the LT in slot n_{dyn} + 1 of cycle 8.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The frame transmitted indicator, the transmission conflict flag and the syntax error flag in the transmit buffer status, the valid frame flag in the slot status data and the frame contents data are set accordingly.

7.2.2.26 Independent nature of the dynamic segment

— Test purpose

Verify the independent nature of the dynamic segment for dual channel devices. It is possible that a frame transmission that is attempted on both channels may be successful only on one channel. It is also possible that a frame successfully transmitted on both channels but at different points in time on the different channels.

— Applicability

DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to send a dynamic frame on both channels in slots n_{dyn} + 1 and *gNumberOfStaticSlots* + *pLatestTx*. One transmit buffer per channel is set up for transmission in slot n_{dyn} + 1 for cycle 9 and one transmit buffer per channel is set up for transmission in slot *gNumberOfStaticSlots* + *pLatestTx* for cycle 10.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on channel A.

2) In the dynamic segment of cycle 9, it is verified (LT) that the IUT transmits its frame in slot n_{dyn} + 1 correctly (starting at the minislot action point + ξ + ξ_{IUT} µT of minislot n_{end} + 1 for channel A and minislot 2 for channel B).

3) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on both channels is n_{dyn} + 1 in the dynamic segment status.

4) In cycle 10, the LT simulates a frame in dynamic slot *gNumberOfStaticSlots* + *pLatestTx* – 1 with the frame ID set to *gNumberOfStaticSlots* + *pLatestTx* – 1 on channel B.

5) In the dynamic segment of cycle 10, it is verified (LT) that the IUT transmits its frame in slot *gNumberOfStaticSlots* + *pLatestTx* correctly on channel A and that the IUT does not transmit a frame on channel B in the dynamic segment.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on channel A is *gNumberOfStaticSlots* + *pLatestTx* and that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on channel B is 0 in the dynamic segment status.

Figure 17 depicts the independent nature of the dynamic segment.

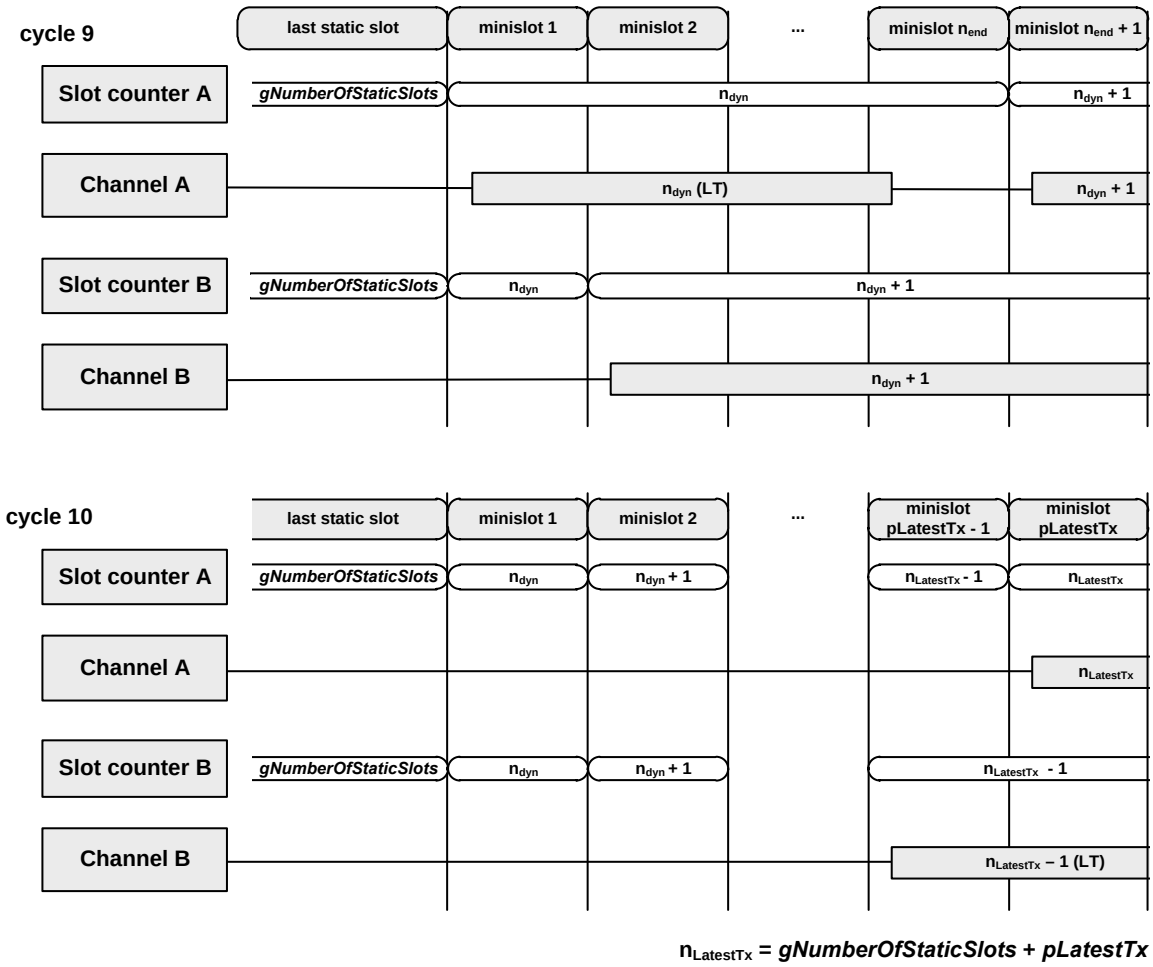


Figure 17 — Independent nature of the dynamic segment

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames in the dynamic segment as expected and sets *vLastDynTxSlot* accordingly.

7.2.3 Symbol window

7.2.3.1 MTS

— Test purpose

Verify the reception of an MTS in the symbol window.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 34, Table 35 and Table 36.

Table 34 — Modification to basic configurations 1a and 1b for symbol window – MTS

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	77	97	105	77	97	105
gdTSSTransmitter [gdBit]	1[a]	11	15	1[a]	11	15
pDecodingCorrection [µT]	12	52	68	24	104	136
pMacroInitialOffset[A,B] [MT]	5	6		5	6	
pMicroInitialOffset[A,B] [µT]	23		7	46		14

[a] *gdTSSTransmitter* = 1 *[gdBit]* which is an intentional protocol constraints violation.

Table 35 — Modification to basic configurations 2a and 2b for symbol window – MTS

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	70	80	84	70	80	84
gdTSSTransmitter [gdBit]	1[a]	6	8	1[a]	6	8
pDecodingCorrection [µT]	24	64	80	12	32	40
pMacroInitialOffset[A,B] [MT]	4		5	4		5
pMicroInitialOffset[A,B] [µT]	46	6	70	23	3	35

[a] *gdTSSTransmitter* = 1 *[gdBit]* which is an intentional protocol constraints violation.

Table 36 — Modification to basic configuration 3 for symbol window – MTS

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	66	72	74
gdTSSTransmitter [gdBit]	1[a]	4	5
pDecodingCorrection [µT]	24	48	56
pMacroInitialOffset[A,B] [MT]	4	5	
pMicroInitialOffset[A,B] [µT]	6	22	14

[a] *gdTSSTransmitter* = 1 *[gdBit]* which is an intentional protocol constraints violation.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

2) In cycle 9, the LT simulates an MTS of length *cdCASRxLowMin* in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is true in the symbol window status.
The UT clears the aggregated channel status after its verification.

3) In cycle 10, the LT simulates an MTS of length *cdCASRxLowMin* – 1 in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

4) In cycle 11, the LT simulates an MTS of length *gdCASRxLowMax* in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is true in the symbol window status.
The UT clears the aggregated channel status after its verification.

5) In cycle 12, the LT simulates an MTS of length *gdCASRxLowMax* + 1 in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is true in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

6) In cycle 13, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator in the symbol window status and in the aggregated channel status and the flag *vSS!ValidMTS* in the symbol window status are set accordingly.

7.2.3.2 Two MTS within symbol window

— **Test purpose**

Verify the reception of two media access test symbols in the symbol window.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

2) In cycle 9, the LT simulates two valid media access test symbols of length *gdTSSTransmitter* + *cdCAS* in the symbol window. The gap between the media access test symbols is 2 * *cChannelIdleDelimiter*. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is true in the symbol window status.
The UT clears the aggregated channel status after its verification.

3) In cycle 10, the LT simulates two valid media access test symbols of length *gdTSSTransmitter* + *cdCAS* with the first media access test symbol starting *gdTSSTransmitter* before the beginning of the symbol window and crossing the boundary between dynamic segment and symbol window and the second media access test symbol staying within the symbol window. The gap between the media access test symbols is 2 * *cChannelIdleDelimiter*. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is true in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is true in the symbol window status.
The UT clears the aggregated channel status after its verification.

4) In cycle 11, the LT simulates two valid media access test symbols of length *gdTSSTransmitter* + *cdCAS* with the first media access test symbol staying within the symbol window and the second media access test symbol starting *gdTSSTransmitter* before the end of the symbol window and crossing the boundary between symbol window and NIT. The gap between the media access test symbols is 2 * *cChannelIdleDelimiter*. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false and the

boundary violation flag / indicator is true in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is true in the symbol window status.
The UT clears the aggregated channel status after its verification.

5) In cycle 12, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator and the boundary violation flag / indicator in the symbol window status and in the aggregated channel status and the flag *vSS!ValidMTS* in the symbol window status are set accordingly.

7.2.3.3 Complete syntactically correct frame within symbol window

— Test purpose

Verify correct handling of a frame received in the symbol window.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 37.

Table 37 — Modification to basic configuration 3 for symbol window – complete syntactically correct frame within symbol window

Parameter	Modification to Basic Configuration 3
gdSymbolWindow [MT]	31
gNumberOfMinislots	143
gdNIT [MT]	18
pLatestTx [Minislot]	66

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

2) In the symbol window of cycle 9, starting at the symbol window action point the LT simulates a frame with a payload length of 1 two-byte word with all bytes set to 0x00, the frame ID set to 1, the reserved bit, the payload preamble indicator, the sync frame indicator and the startup frame indicator set to '0' and the null frame indicator set to '1'.
The UT clears the aggregated channel status after its verification.

3) 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is true and the boundary violation flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

4) In cycle 10, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator in the symbol window status and in the aggregated channel status and the flag *vSS!ValidMTS* in the symbol window status are set accordingly.

7.2.3.4 Complete syntactically correct frame reaching NIT

— **Test purpose**

Verify correct handling of a frame received in the symbol window crossing the boundary to the NIT.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the symbol window and in the NIT. 500 µT after start of the next cycle and before the symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status, in the NIT status and in the aggregated channel status and that the boundary violation flag / indicator is false in the symbol window status, in the NIT status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

2) In cycle 9, the LT simulates a frame with a payload length of 1 two-byte word with all bytes set to 0x00, the frame ID set to 1, the reserved bit, the payload preamble indicator, the sync frame indicator and the startup frame indicator set to '0' and the null frame indicator set to '1' starting 4 MT for the basic configurations 1a, 1b, 2a and 2b and 14 MT for the basic configuration 3 before the end of the symbol window and ending within the NIT, i.e. a frame crossing the boundary between the symbol

window and the NIT.
500 µT after start of the next cycle and before the symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status, in the NIT status and in the aggregated channel status, that the boundary violation flag / indicator is true in the symbol window status, in the NIT status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

3) In cycle 10, the LT simulates a frame with a payload length of 1 two-byte word with all bytes set to 0x00, the frame ID set to 1, the reserved bit, the payload preamble indicator, the sync frame indicator and the startup frame indicator set to '0' and the null frame indicator set to '1' starting within the symbol window and ending 5 * *gdBit* before the boundary to the NIT, i.e. the frame causes a violation of the boundary between the symbol window and the NIT during channel idle recognition.
500 µT after start of the next cycle and before the symbol window, it is verified (UT) that the syntax error flag / indicator is true in the symbol window status and in the aggregated channel status, that the syntax error flag is false in the NIT status, that the boundary violation flag / indicator is true in the symbol window status, in the NIT status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

4) In cycle 11, the LT simulates idle in the symbol window and the NIT. 500 µT after start of the next cycle and before the symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status, in the NIT status and in the aggregated channel status and that the boundary violation flag / indicator is false in the symbol window status, in the NIT status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the boundary violation flag / indicator are set accordingly in the symbol window status, in the NIT status and in the aggregated channel status.

7.2.3.5 Wakeup decoding

— **Test purpose**

Verify correct handling of wakeup pattern the reception in the symbol window. The wakeup pattern begins in dynamic segment or in symbol window and ends in symbol window or in NIT.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 38.

Table 38 — Modification to basic configurations for symbol window – wakeup decoding

Parameter	Modification
gMaxWithoutClockCorrectionPassive	2
gMaxWithoutClockCorrectionFatal	3
pAllowHaltDueToClock	false

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is false in the symbol window status, that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

2) In the symbol window of cycle 9, starting at the symbol window action point, the LT simulates a sequence of low phase – idle phase – low phase (see Figure 18 a). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the idle phase a length of *gdWakeupRxIdle* [*gdBit*]. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is true (first low phase of length *gdWakeupRxLow* interpreted as valid MTS) for Basic Configuration 1a and 1b and false for Basic Configuration 2a, 2b and 3 in the symbol window status, that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is true in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

3) In the dynamic segment of cycle 10, starting *gdWakeupRxIdle* / 2 [*gdBit*] before the end of the dynamic segment, the LT simulates a sequence of low phase – idle phase – low phase (see Figure 18 b). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the idle phase a length of *gdWakeupRxIdle* [*gdBit*]. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is true (second low phase of length *gdWakeupRxLow* interpreted as valid MTS) for Basic Configuration 1a and 1b and false for Basic Configuration 2a, 2b and 3 in the symbol window status, that the boundary violation flag / indicator is true in the symbol window status and in the aggregated channel status, that the syntax error flag / indicator is false for Basic Configuration 1a and 1b and true for Basic Configurations 2a, 2b and 3 in the symbol window status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is true in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

4) In the symbol window of cycle 11, starting 5 / 2 * *gdWakeupRxLow* [*gdBit*] + 2 * *gdWakeupRxIdle* [*gdBit*] before the end of the symbol window, the LT simulates a sequence of a low phase – idle phase – low phase – idle phase – low phase (see Figure 18 c). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the idle phase a length of *gdWakeupRxIdle* [*gdBit*]. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is true (first low phase of length *gdWakeupRxLow* interpreted as valid MTS) for Basic Configuration 1a and 1b and false for Basic Configuration 2a, 2b and 3 in the symbol window status, that the boundary violation flag / indicator is true in the symbol window status, in the NIT status and in the aggregated channel status, that the syntax error flag is false in the symbol window status, that the syntax error flag / indicator is false for Basic Configuration 1a and 1b and true for Basic Configurations 2a, 2b and 3 in the NIT status and in the aggregated channel status, that the wakeup status

vPOC!WakeupStatus is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is true in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

5) In cycle 12, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is false in the symbol window status, that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

6) In cycle 13 to 21, the LT stops transmission of its startup frame in slot 1.

7) In cycle 17, 500 µT after cycles start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = NORMAL_PASSIVE).

8) In cycle 17, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is false in the symbol window status, that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

9) In the symbol window of cycle 18, starting at the symbol window action point, the LT simulates a sequence of low phase – idle phase – low phase (see Figure 18 a). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the idle phase a length of *gdWakeupRxIdle* [*gdBit*]. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is true (first low phase of length *gdWakeupRxLow* interpreted as valid MTS) for Basic Configuration 1a and 1b and false for Basic Configuration 2a, 2b and 3 in the symbol window status, that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is true in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

10) In the dynamic segment of cycle 19, starting *gdWakeupRxIdle* / 2 [*gdBit*] before the end of the dynamic segment, the LT simulates a sequence of low phase – idle phase – low phase (see Figure 18 b). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the idle phase a length of *gdWakeupRxIdle* [*gdBit*]. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is true (second low phase of length *gdWakeupRxLow* interpreted as valid MTS) for Basic Configuration 1a and 1b and false for Basic Configuration 2a, 2b and 3 in the symbol window status, that the boundary violation flag / indicator is true in the symbol window status and in the aggregated channel status, that the syntax error flag / indicator is false for Basic Configuration 1a and 1b and true for Basic Configurations 2a, 2b and 3 in the symbol window status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is true in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

11) In the symbol window of cycle 20, starting 5 / 2 * *gdWakeupRxLow* [*gdBit*] + 2 * *gdWakeupRxIdle* [*gdBit*] before the end of the symbol window, the LT simulates a sequence of a low phase – idle phase – low phase – idle phase – low phase (see Figure 18 c). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the idle phase a length of *gdWakeupRxIdle* [*gdBit*]. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is true (first low phase of length *gdWakeupRxLow* interpreted as valid MTS) for Basic Configuration 1a and 1b and false for Basic Configuration 2a, 2b and 3 in the symbol window status, that the boundary violation flag / indicator is true in the symbol window status, in the NIT status and in the aggregated channel status, that the syntax error flag is false in the symbol window status, that the syntax error flag / indicator is false for Basic Configuration 1a and 1b and true for Basic Configurations 2a, 2b and

3 in the NIT status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is true in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

12) In cycle 21, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the flag *vSS!ValidMTS* is false in the symbol window status, that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status.
The UT clears the aggregated channel status after its verification.

Figure 18 depicts the Wakeup decoding – different scenarios of WUP reception in the symbol window.

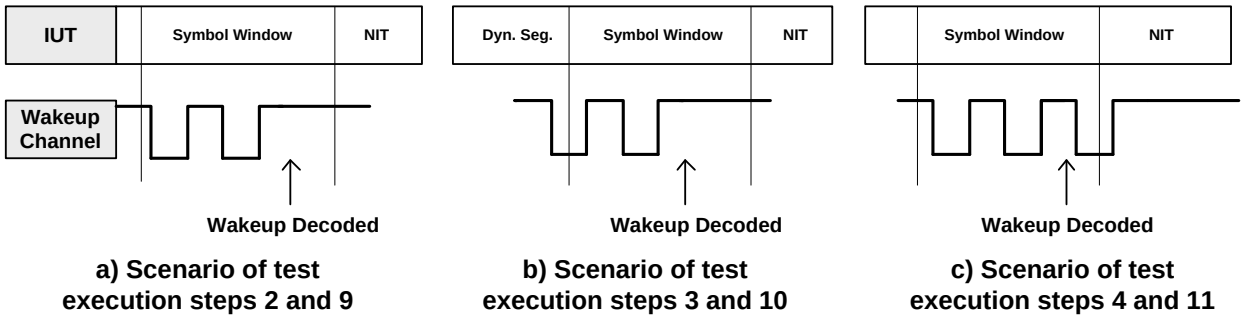


Figure 18 — Wakeup decoding – different scenarios of WUP reception in the symbol window

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator, the boundary violation flag / indicator and the flag *vSS!ValidMTS* in the symbol window status, in the aggregated channel status and in the NIT status, the wakeup status *vPOC!WakeupStatus* and the wakeup pattern received indicator are set accordingly.

7.2.3.6 Noise

— Test purpose

Verify correct handling of noise in the symbol window.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

2) In cycle 9, the LT simulates noise (low phase of length *gdSampleClockPeriod*) in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

3) In cycle 10, the LT simulates noise (low phase of length 2 * *gdSampleClockPeriod*) in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

4) In cycle 11, the LT simulates noise (low phase of length 3 * *gdSampleClockPeriod*) in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

5) In cycle 12, the LT simulates noise (low phase of length (*cStrobeOffset* + 1) * *gdSampleClockPeriod*) in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

6) In cycle 13, the LT simulates noise (low phase of length (*gdTSSTransmitter* + 2) * *gdBit*) in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

7) In cycle 14, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false in the symbol window status and in the aggregated channel status.
The UT clears the aggregated channel status after its verification.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator is set accordingly in the symbol window status and in the aggregated channel status.

7.2.3.7 Three MTS within symbol window

— Test purpose

Verify the reception of three media access test symbols in the symbol window with one symbol crossing symbol window boundary. The IUT shall set the syntax error flag / indicator, the boundary violation flag / indicator and the flag *vSS!ValidMTS* for that receive scenario.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 39.

Table 39 — Modification to basic configuration 3 for symbol window – three MTS within symbol window

Parameter	Modification to Basic Configuration 3
gdSymbolWindow [MT]	32
gNumberOfMinislots	143
gdNIT [MT]	17
pLatestTx [Minislot]	66

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that syntax error flag / indicator is false and the boundary violation flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

2) In cycle 9, the LT simulates three valid media access test symbols of length *gdTSSTransmitter* + *cdCAS* with the first media access test symbol starting *gdTSSTransmitter* before the beginning of the symbol window and crossing the boundary between dynamic segment and symbol window and the other media access test symbols staying within the symbol window. The gap between the media access test symbols is 2 * *cChannelIdleDelimiter*. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is true in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is true in the symbol window status.
The UT clears the aggregated channel status after its verification.

3) In cycle 10, the LT simulates three valid media access test symbols of length *gdTSSTransmitter* + *cdCAS* with the first two media access test symbols staying within the symbol window and the third media access test symbol starting *gdTSSTransmitter* before the end of the symbol window and crossing the boundary between symbol window and NIT. The gap between the media access test symbols is 2 * *cChannelIdleDelimiter*. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the boundary violation flag / indicator is true in the symbol window status and in the aggregated channel status and that the syntax error flag is false and the

flag *vSS!ValidMTS* is true in the symbol window status.
The UT clears the aggregated channel status after its verification.

4) In cycle 11, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that syntax error flag / indicator is false and the boundary violation flag / indicator is false in the symbol window status and in the aggregated channel status and that the flag *vSS!ValidMTS* is false in the symbol window status.
The UT clears the aggregated channel status after its verification.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator and the boundary violation flag / indicator in the symbol window status and in the aggregated channel status and the flag *vSS!ValidMTS* in the symbol window status are set accordingly.

7.2.3.8 WUDOP reception

— Test purpose

Verify correct handling of WUDOP reception with and without noise. The IUT shall not accept the first low phase of WUDOP, if the idle phase ahead was too short due to noise. The IUT shall accept the WUDOP when wakeup is decoded in the symbol window or in the NIT.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 40, Table 41 and Table 42.

Table 40 — Modification to basic configurations 1a and 1b for symbol window – WUDOP reception

Parameter	Modification to Basic Configurations			
	1a		1b	
	I	II	I	II
gMaxWithoutClockCorrectionPassive	1			
gMaxWithoutClockCorrectionFatal	3			
gdSymbolWindow [MT]	40	34	40	34
gdSymbolWindowActionPointOffset [MT]	2[a]	2[a]	2[a]	2[a]
gdNIT [MT]	14	20	14	20
[a] Intentional protocol constraints violation.				

Table 41 — Modification to basic configurations 2a and 2b for symbol window – WUDOP reception

Parameter	Modification to Basic Configurations			
	2a		2b	
	I	II	I	II
gMaxWithoutClockCorrectionPassive	1			
gMaxWithoutClockCorrectionFatal	3			
gdSymbolWindow [MT]	23	17	23	17
gdSymbolWindowActionPointOffset [MT]	1[a]	1[a]	1[a]	1[a]
gdNIT [MT]	12	18	12	18
[a] Intentional protocol constraints violation.				

Table 42 — Modification to basic configuration 3 for symbol window – WUDOP reception

Parameter	Modification to Basic Configuration 3	
	I	II
gMaxWithoutClockCorrectionPassive	1	
gMaxWithoutClockCorrectionFatal	3	
gdSymbolWindow [MT]	24	17
gdSymbolWindowActionPointOffset [MT]	1[a]	1[a]
gdNIT [MT]	11	18
[a] Intentional protocol constraints violation.		

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the symbol window. 500 µT after start of the NIT in this cycle and before the next symbol window, it is verified (UT) that the wakeup pattern received indicator is false in the wakeup and startup status on the available channel(s).

2) In the symbol window of cycle 9, starting at the symbol window action point, the LT simulates:

   (a) a valid WUDOP symbol (low – high – low – high – low phase, each low or high with a length of gdWakeupTxActive [gdBit]) on the available channel(s).

   (b) a low – high – low phase, each with a length of gdWakeupTxActive [gdBit] on the available channel(s).

3) In cycle 9, (3 * *gdWakeupTxActive* + *gdWakeupRxIdle*) * *gdBit* / *pdMicrotick* + 500 µT after the symbol window action point, it is verified (UT) that the wakeup pattern received indicator is true on the available channel(s) in the wakeup and startup status. The UT resets the wakeup pattern received indicator.

4) In the dynamic segment of cycle 10, starting 1 macrotick before the end of the dynamic segment, the LT simulates noise (a low phase of length 3 [*gdBit*]) on the available channel(s).

5) In the symbol window of cycle 10, starting at the symbol window action point, the LT simulates:

   (a) a valid WUDOP symbol on the available channel(s).

   (b) a low – high – low phase, each with a length of *gdWakeupTxActive* [*gdBit*] on the available channel(s).

6) In cycle 10, (5 * *gdWakeupTxActive* + *gdWakeupRxIdle*) * *gdBit* / *pdMicrotick* + 500 µT after the symbol window action point, it is verified (UT) that the wakeup pattern received indicator

   (a) is true on the available channel(s) in the wakeup and startup status. The UT resets the wakeup pattern received indicator.

   (b) is false on the available channel(s) in the wakeup and startup status.

7) In cycle 11 to 13, the LT stops transmission of its startup frame in slot 1.

8) In cycle 12, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

9) In the symbol window of cycle 12, starting at the symbol window action point, the LT simulates:

   (a) a valid WUDOP symbol on the available channel(s).

   (b) a low – high – low phase, each with a length of *gdWakeupTxActive* [*gdBit*] on the available channel(s).

10) In cycle 12, (3 * *gdWakeupTxActive* + *gdWakeupRxIdle*) * *gdBit* / *pdMicrotick* + 500 µT after the symbol window action point, it is verified (UT) that the wakeup pattern received indicator is true on the available channel(s) in the wakeup and startup status. The UT resets the wakeup pattern received indicator.

11) In the dynamic segment of cycle 13, starting 1 macrotick before the end of the dynamic segment, the LT simulates noise (a low phase of length 3 [*gdBit*]) on the available channel(s).

12) In the symbol window of cycle 13, starting at the symbol window action point, the LT simulates:

   (a) a valid WUDOP symbol on the available channel(s).

   (b) a low – high – low phase, each with a length of *gdWakeupTxActive* [*gdBit*] on the available channel(s).

13) In cycle 13, (5 * *gdWakeupTxActive* + *gdWakeupRxIdle*) * *gdBit* / *pdMicrotick* + 500 µT after the symbol window action point, it is verified (UT) that the wakeup pattern received indicator:

   (a) is true on the available channel(s) in the wakeup and startup status.

   (b) is false on the available channel(s) in the wakeup and startup status.

Figure 19 depicts the WUDOP reception – different scenarios of WUDOP reception with and without noise.

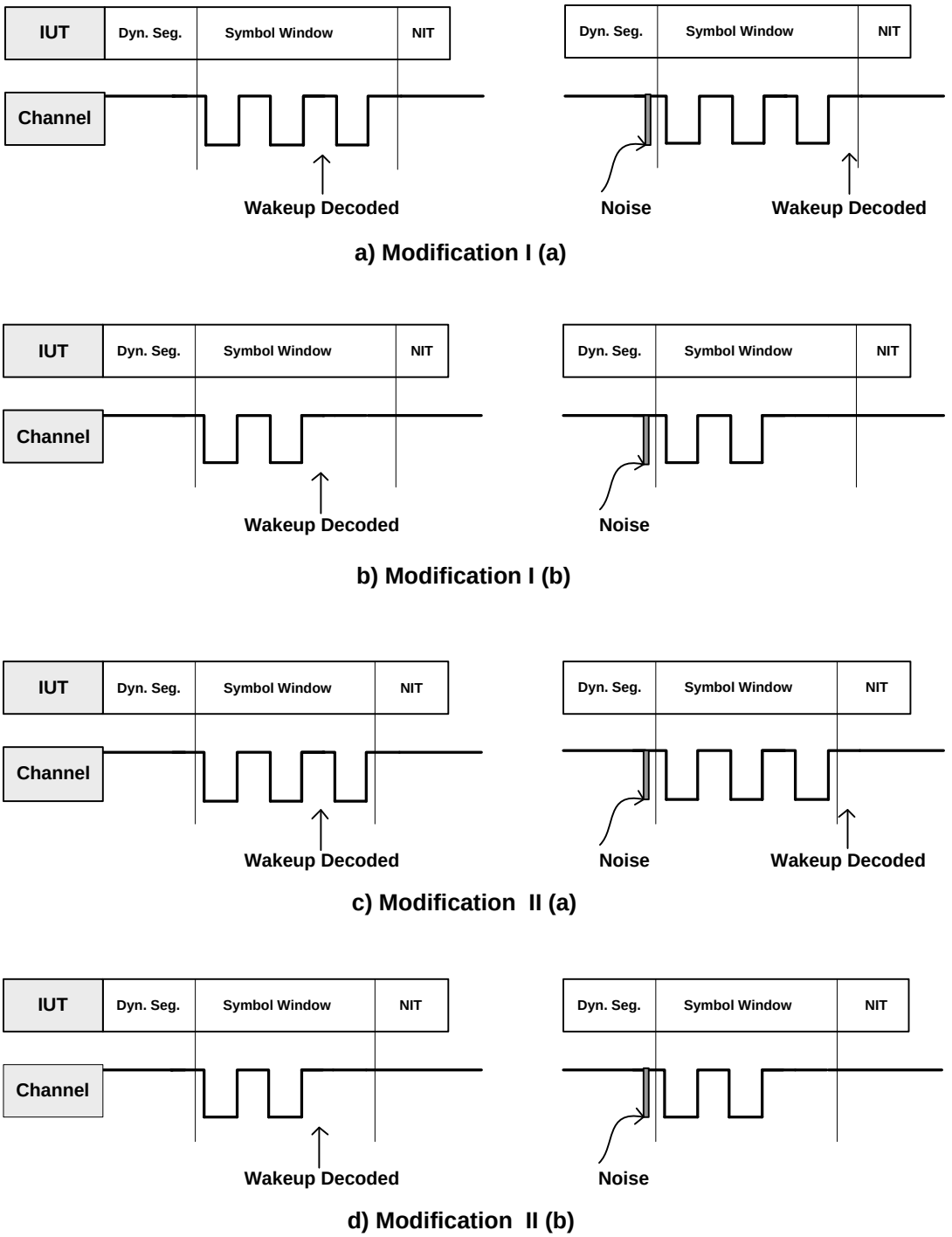


Figure 19 — WUDOP reception – different scenarios of WUDOP reception with and without noise

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The wakeup pattern received indicator in the wakeup pattern received status is set accordingly.

7.2.4 Network idle time

7.2.4.1 Complete syntactically correct frame within NIT

— **Test purpose**

Verify correct handling of a frame received during network idle time.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 43.

Table 43 — Modification to basic configurations for network idle time – complete syntactically correct frame within NIT

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
gdSymbolWindow [MT]	0	0	0
gdNIT [MT]	54	35	35

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

2) In the NIT of cycle 9, the LT simulates a frame with a payload length of 1 two-byte word with all bytes set to 0x00, the frame ID set to 1, the reserved bit, the payload preamble indicator, the sync frame indicator and the startup frame indicator set to '0' and the null frame indicator set to '1'.

3) 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status.

4) In cycle 10, the LT simulates idle in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag / indicator is set accordingly in the NIT status and in the aggregated channel status.

7.2.4.2 Complete syntactically correct frame reaching static segment

— Test purpose

Verify correct handling of a frame received during network idle time crossing the boundary to the static segment of the following cycle.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 44.

Table 44 — Modification to basic configurations for network idle time – complete syntactically correct frame reaching static segment

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
gdSymbolWindow [MT]	0	0	0
gNumberOfMinislots	219	140	142
gdNIT [MT]	54	35	56
pLatestTx [Minislot]	188	101	65

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates idle in the NIT.

2) In slot 4 of cycle 9, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 1, in the NIT status and in the aggregated channel status.

3) In cycle 9, the LT simulates its frame from slot 1 / cycle 10 earlier, i.e. the LT starts the frame 1 MT before the end of the NIT causing a boundary violation between the NIT of cycle 9 and the static segment of cycle 10.

4) In slot 4 of cycle 10, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is true in the slot status data of slot 1, in the NIT status, and in the aggregated channel status.

5) In cycle 10, the LT simulates its frame from slot 1 / cycle 11 earlier, i.e. the LT simulates the frame in the NIT such that the frame ends 5 * *gdBit* before the static segment start. Thus, the LT's frame causes a violation of the boundary between NIT and static segment during channel idle recognition.

6) In slot 4 of cycle 11, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status and that the boundary violation flag / indicator is true in the slot status data of slot 1, in the NIT status, and in the aggregated channel status.

7) In cycle 11, the LT simulates idle in the NIT.

8) In slot 4 of cycle 12, it is verified (UT) that the syntax error flag / indicator is false and the boundary violation flag / indicator is false in the slot status data of slot 1, in the NIT status, and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the boundary violation flag / indicator are set accordingly in the NIT status, in the slot status data and in the aggregated channel status.

7.2.4.3 MTS

— **Test purpose**

Verify correct handling of an MTS received during network idle time.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

2) In cycle 9, the LT simulates a valid MTS of length *gdTSSTransmitter* + *cdCAS* in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status.

3) In cycle 10, the LT simulates idle in the NIT.

4) In cycle 11, 500 µT after cycle start and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator is set accordingly in the NIT status and in the aggregated channel status.

7.2.4.4 Wakeup decoding

— Test purpose

Verify correct handling of wakeup pattern reception during network idle time. The wakeup pattern begins in the symbol window or in the NIT and ends in the NIT or in the static segment of the next cycle.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 45.

Table 45 — Modification to basic configurations for network idle time – Wakeup decoding

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
gdSymbolWindow [MT]	20	13	17
gdNIT [MT]	34	22	18

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates idle in the NIT.

2) In cycle 9, 500 µT after cycle start and before the symbol window, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation control status and that the wakeup pattern received indicator is false in the wakeup and startup status.

3) In cycle 9, starting *gdWakeupRxLow* [*gdBit*] + 4 [*gdBit*] before the end of the symbol window, the LT simulates a sequence of low phase – idle phase – low phase (see Figure 20 a). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the high phase a length of *gdWakeupRxIdle* [*gdBit*].

4) In cycle 10, 500 µT after cycle start and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status, that the boundary violation flag / indicator is true in the symbol window status, in the NIT status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation status and that the wakeup pattern received indicator is true in the wakeup and startup status.

5) In cycle 10, starting 4 MT after the beginning of the NIT, the LT simulates a sequence of low phase – idle phase – low phase (see Figure 20 b). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the high phase a length of *gdWakeupRxIdle* [*gdBit*].

6) In cycle 11, 500 µT after cycle start and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation status and that the wakeup pattern received indicator is true in the wakeup and startup status.

7) In cycle 11, starting 5 / 2 * *gdWakeupRxLow* [*gdBit*] + 2 * *gdWakeupRxIdle* [*gdBit*] before cycle end, the LT simulates a sequence of a low phase – idle phase – low phase – idle phase – low phase (see

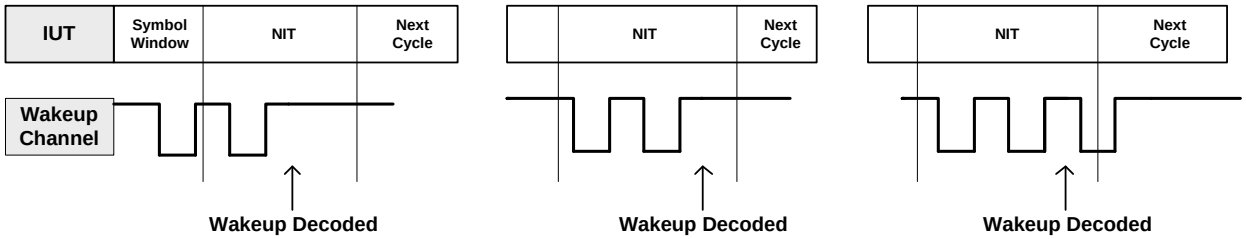
Figure 20 c). The low phase has a length of *gdWakeupRxLow* [*gdBit*], the high phase a length of *gdWakeupRxIdle* [*gdBit*].

8) In cycle 12, after the end of static segment and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status, that the boundary violation flag / indicator is true in the NIT status, in the slot status data of slot 1 and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation status and that the wakeup pattern received indicator is true in the wakeup and startup status.

9) In cycle 12, the LT simulates idle in the NIT.

10) In cycle 13, 500 µT after cycle start and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status, that the wakeup status *vPOC!WakeupStatus* is *UNDEFINED* in the protocol operation status and that the wakeup pattern received indicator is false in the wakeup and startup status.

Figure 20 depicts the wakeup decoding – different scenarios of WUP reception in the NIT.



a) Scenario of test execution steps 3

b) Scenario of test execution steps 5

c) Scenario of test execution steps 7

Figure 20 — Wakeup decoding – different scenarios of WUP reception in the NIT

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator and the boundary violation flag / indicator in the symbol window status and in the aggregated channel status, the wakeup status *vPOC!WakeupStatus* and the wakeup pattern received indicator are set accordingly.

7.2.4.5 Noise

— **Test purpose**

Verify correct handling of noise during network idle time.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates idle in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

2) In cycle 9, the LT simulates noise (low phase of length *gdSampleClockPeriod*) in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

3) In cycle 10, the LT simulates noise (low phase of length 2 * *gdSampleClockPeriod*) in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

4) In cycle 11, the LT simulates noise (low phase of length 3 * *gdSampleClockPeriod*) in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

5) In cycle 12, the LT simulates noise (low phase of length (*cStrobeOffset* + 1) * *gdSampleClockPeriod*) in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status.

6) In cycle 13, the LT simulates noise (low phase of length (*gdTSSTransmitter* + 2) * *gdBit*) in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status.

7) In cycle 14, the LT simulates noise (low phase of length (*gdTSSTransmitter* + 3) * *gdBit*) in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is true in the NIT status and in the aggregated channel status.

8) In cycle 15, the LT simulates idle in the NIT. 500 µT after start of the next cycle and before the NIT, it is verified (UT) that the syntax error flag / indicator is false in the NIT status and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag / indicator is set accordingly in the NIT status and in the aggregated channel status.

7.3 CHI

7.3.1 Timing related configuration data

7.3.1.1 pMicroPerCycle

— Test purpose

Verify correct handling of the parameter *pMicroPerCycle*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 46, Table 47 and Table 48.

Table 46 — Modification to basic configurations 1a and 1b for timing related configuration data – pMicroPerCycle

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	3	4		2	4	
gdStaticSlot [MT]	17	20	35	15	20	35
gdSymbolWindow [MT]	0	40		0	40	
gdSymbolWindowActionPointOffset [MT]	3	4		2	4	
gMacroPerCycle [MT]	92	300	16 000	52	300	16 000
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	3	12	450	2	12	455
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	92	300	16 000	52	300	16 000
gdNIT [MT]	41	20	210	22	20	35
pClusterDriftDamping [µT]	0	2		0	2	
pdListenTimeout [µT]	7 366	24 016	1 280 770	8 326	48 030	2 561 538
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	5	6		4	6	
pMicroPerCycle [µT]	3 680	12 000	640 000	4 160	24 000	1 280 000
pOffsetCorrectionOut [µT]	120	153		200	226	
pOffsetCorrectionStart [MT]	85	290	15 980	46	295	15 995
pRateCorrectionOut [µT]	3	8	385	3	15	769

Table 47 — Modification to basic configurations 2a and 2b for timing related configuration data – pMicroPerCycle

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	2	3	2	3	
gdStaticSlot [MT]	13	15	33	14	32	33
gdSymbolWindow [MT]	0		23	0	23	
gdSymbolWindowActionPointOffset [MT]	1	2	3	2	3	
gMacroPerCycle [MT]	47	150	8 000	85	300	8 000
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	2	8	241	3	8	240
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	94	300	16 000	170	600	16 000
gdNIT [MT]	21	30	24	43	21	57
pClusterDriftDamping [µT]	0	2		0	2	
pdListenTimeout [µT]	7 526	24 016	1 280 770	6 806	24 016	640 386
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	3	4	3	4	
pMicroPerCycle [µT]	3 760	12 000	640 000	3 400	12 000	320 000
pOffsetCorrectionOut [µT]	120	154		100	117	
pOffsetCorrectionStart [MT]	41	141	7 990	70	290	7 970
pRateCorrectionOut [µT]	3	8	385	3	8	193

Table 48 — Modification to basic configuration 3 for timing related configuration data – pMicroPerCycle

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	2	3	
gdStaticSlot [MT]	24	26	58
gdSymbolWindow [MT]	0	24	
gdSymbolWindowActionPointOffset [MT]	2	3	
gMacroPerCycle [MT]	210	300	8 000
gNumberOfMinislots	0		
gNumberOfStaticSlots	7	9	136
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	420	600	16 000
gdNIT [MT]	42		88
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	16 812	24 016	640 386
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	4	5	
pMicroPerCycle [µT]	8 400	12 000	320 000
pOffsetCorrectionOut [µT]	100	118	
pOffsetCorrectionStart [MT]	200	290	7 950
pRateCorrectionOut [µT]	6	8	193

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) It is verified (LT) that the IUT transmits its frame in slot 1 of cycle 7 correctly. I.e., the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0'.

2) It is verified (LT) that the interval between the IUT's startup frames in cycle 8 and cycle 9 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits frames correctly. The intervals between the IUT's frames are correct.

7.3.1.2 gMacroPerCycle

— Test purpose

Verify correct handling of the parameter *gMacroPerCycle*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 49, Table 50 and Table 51.

Table 49 — Modification to basic configurations 1a and 1b for timing related configuration data – gMacroPerCycle

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	4		1	4	
gdStaticSlot [MT]	5	35		6	35	
gdSymbolWindow [MT]	0	40		0	40	
gdSymbolWindowActionPointOffset [MT]	1	4		1	4	
gMacroPerCycle [MT]	17	2 666	16 000	20	2 666	16 000
gNumberOfMinislots	0	121	600	0	121	600
gNumberOfStaticSlots	2	42	300	2	42	300
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	102	2 666	16 000	60	2 666	16 000
gdMacrotick [µs]	6	1		3	1	
gdNIT [MT]	7	67	60	8	67	60
pClusterDriftDamping [µT]	2			2		
pdListenTimeout [µT]	8 166	213 410	1 280 770	9 606	426 818	2 561 538
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	6		2	6	
pMicroInitialOffset[A,B] [µT]	183	23		126	46	
pMicroPerCycle [µT]	4 080	106 640	640 000	4 800	213 280	1 280 000
pOffsetCorrectionOut [µT]	153			226		
pOffsetCorrectionStart [MT]	16	2 650	15 950	18	2 650	15 950
pRateCorrectionOut [µT]	3	65	385	3	129	769

Table 50 — Modification to basic configurations 2a and 2b for timing related configuration data – gMacroPerCycle

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	3		1	3	
gdStaticSlot [MT]	5	33		5	33	
gdSymbolWindow [MT]	0	23		0	23	
gdSymbolWindowActionPointOffset [MT]	1	3		1	3	
gMacroPerCycle [MT]	17	2 666	8 000	28	2 666	8 000
gNumberOfMinislots	0	140	335	0	144	335
gNumberOfStaticSlots	2	50	170	2	49	170
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	102	5 332	16 000	168	5 332	16 000
gdMacrotick [µs]	6	2		6	2	
gdNIT [MT]	7	13	22	16	18	22
pClusterDriftDamping [µT]	0	2		0	2	
pdListenTimeout [µT]	8 166	426 818	1 280 770	6 726	213 410	640 386
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	4		2	4	
pMicroInitialOffset[A,B] [µT]	166	6		83	3	
pMicroPerCycle [µT]	4 080	213 280	640 000	3 360	106 640	320 000
pOffsetCorrectionOut [µT]	154			117		
pOffsetCorrectionStart [MT]	16	2 660	7 990	25	2 650	7 990
pRateCorrectionOut [µT]	3	129	385	3	65	193

Table 51 — Modification to basic configuration 3 for timing related configuration data – gMacroPerCycle

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	3	
gdStaticSlot [MT]	8	58	
gdSymbolWindow [MT]	0	24	
gdSymbolWindowActionPointOffset [MT]	1	3	
gMacroPerCycle [MT]	70	2 666	8 000
gNumberOfMinislots	0	152	209
gNumberOfStaticSlots	6	27	112
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	420	5 332	16 000
gdMacrotick [µs]	6	2	
gdNIT [MT]	16	12	17
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	16 812	213 410	640 386
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	2	5	
pMicroInitialOffset[A,B] [µT]	62	22	
pMicroPerCycle [µT]	8 400	106 640	320 000
pOffsetCorrectionOut [µT]	120	118	
pOffsetCorrectionStart [MT]	64	2 659	7 990
pRateCorrectionOut [µT]	6	65	193

— **Preamble (setup state)**

Preamble II.

— **Test execution**

The time interval during which a respective value of *vMacrotick* may be read has to be specified by the IUT vendor.

1) It is verified (LT) that the IUT transmits its frame in slot 1 of cycle 7 correctly. I.e., the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0'.

2) In cycle 7, it is verified (UT) that

   (a) *vMacrotick* = 0 in the first macrotick of the cycle and

   (b) *vMacrotick* = *gMacroPerCycle* – 1 in the last macrotick of the cycle.

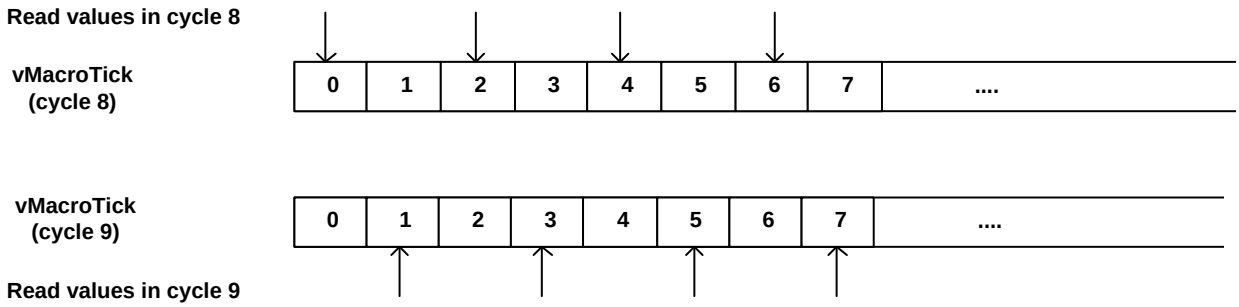
3) In cycle (8 + o) mod *gCycleCountMax*, it is verified (UT) that *vMacrotick* is set to k_n in macrotick k_n with $k_n = i * n + o$ and $k_n < gMacroPerCycle$ ($i \in [1; gMacroPerCycle]$, n = 0, 1, 2,... and o = 0 and is incremented by 1 in each cycle until o equals i–1).

NOTE 1 The step width i should be chosen as 1. In case of stringent timing requirements the step width i may be set to a value greater than 1.

4) It is verified (LT) that the interval between the IUT's startup frames in cycle (8 + i) mod *gCycleCountMax* and cycle (9 + i) mod *gCycleCountMax* is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

NOTE the time interval in which read access to *vMacrotick* shows the corresponding macrotick value of the protocol engine relies on IUT vendor specific information. See 6.6.4 for further information.

Figure 21 depicts the *gMacroPerCycle* – verification of *vMacrotick* with step width i = 2.



With i = 2, the macrotick counter values 0, 2, 4,… can be verified in cycle 8 and the remaining values in cycle 9.

Figure 21 — gMacroPerCycle – verification of vMacrotick with step width i = 2

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits frames correctly. The intervals between the IUT's frames are correct. The macrotick counter counts correctly.

7.3.1.3 gNumberOfStaticSlots

— Test purpose

Verify correct handling of the parameter *gNumberOfStaticSlots*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 52, Table 53 and Table 54.

Table 52 — Modification to basic configurations 1a and 1b for timing related configuration data – gNumberOfStaticSlots

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	3		1	2		1
gdStaticSlot [MT]	17		13	15		13
gdSymbolWindow [MT]	40			40		
gdSymbolWindowActionPointOffset [MT]	3		1	2		1
gMacroPerCycle [MT]	88	8 758	13 353	84	7 734	13 353
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	2	512	1 023	2	512	1 023
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	88	8 758	13 353	84	7 734	13 353
gdNIT [MT]	14			14		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	7 046	701 062	1 068 882	13 450	1 238 184	2 137 764
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	5		3	4		3
pMicroPerCycle [µT]	3 520	350 320	534 120	6 720	618 720	1 068 240
pOffsetCorrectionOut [µT]	153		45	226		45
pOffsetCorrectionStart [MT]	75	8 750	13 343	74	7 721	13 343
pRateCorrectionOut [µT]	3	211	321	5	372	642

Table 53 — Modification to basic configurations 2a and 2b for timing related configuration data – gNumberOfStaticSlots

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	2		1	3		1
gdStaticSlot [MT]	14		12	16	15	13
gdSymbolWindow [MT]	0			23		
gdSymbolWindowActionPointOffset [MT]	2		1	3		1
gMacroPerCycle [MT]	49	7 189	8 000	85	3 882	8 000
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	2	512	665	2	256	612
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	98	14 378	16 000	170	7 764	16 000
gdNIT [MT]	21		20	30	19	21
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	7 846	1 150 932	1 280 770	6 806	310 748	640 386
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	3		2	4		2
pMicroPerCycle [µT]	3 920	575 120	640 000	3 400	155 280	320 000
pOffsetCorrectionOut [µT]	154		45	117		45
pOffsetCorrectionStart [MT]	42	7 169	7 998	56	3 864	7 997
pRateCorrectionOut [µT]	3	346	385	3	94	193

Table 54 — Modification to basic configuration 3 for timing related configuration data – gNumberOfStaticSlots

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	3		1
gdStaticSlot [MT]	25		22
gdSymbolWindow [MT]	24		
gdSymbolWindowActionPointOffset [MT]	3		1
gMacroPerCycle [MT]	210	3 233	8 000
gNumberOfMinislots	0		
gNumberOfStaticSlots	2	127	361
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	420	6 466	16 000
gdNIT [MT]	136	34	
pClusterDriftDamping [µT]	0		
pdListenTimeout [µT]	16 812	258 796	640 386
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	5		3
pMicroPerCycle [µT]	8 400	129 320	320 000
pOffsetCorrectionOut [µT]	118		45
pOffsetCorrectionStart [MT]	95	3 225	7 995
pRateCorrectionOut [µT]	6	78	193

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) It is verified (LT) that the IUT transmits its frame in slot 1 of cycle 7 correctly. I.e., the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0'.

2) In cycle 7, the LT simulates an additional frame in the last static slot *gNumberOfStaticSlots*. The frame ID is set to *gNumberOfStaticSlots*, the cycle count to 7, the payload length to 0, the sync frame indicator and the startup frame indicator, the null frame indicator and the payload preamble indicator are set to '0'.

3) In cycle 7, 50 µT after each slot start and after the NIT start, it is verified (UT) that the slot counters for available channels (*vSlotCounter_A* and / or *vSlotCounter_B*) are incremented correctly in the IUT, i.e. that the slot counters are 1, 2,..., *gNumberOfStaticSlots* in the static slot 1, 2,..., *gNumberOfStaticSlots* of the cycle, and that the slot counters are 0 in the NIT.

4) In the NIT of cycle 7, after the last slot counter comparison, it is verified (UT) that the IUT has received the frame correctly, which was sent by the LT in the last static slot *gNumberOfStaticSlots*.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits and receives frames correctly. The IUT's slot counters count correctly.

7.3.1.4 gdStaticSlot

— **Test purpose**

Verify correct handling of the parameter *gdStaticSlot*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 55, Table 56 and Table 57.

Table 55 — Modification to basic configurations 1a and 1b for timing related configuration data – gdStaticSlot

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1		4	1		4
gdStaticSlot [MT]	3	13	399	5	13	399
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	19	90	2 037	25	81	2 037
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	3		5	3		5
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	114	90	2 037	75	81	2 037
gdMacrotick [µs]	6	1		3	1	
gdNIT [MT]	10	51	42	10	42	
pClusterDriftDamping [µT]	0		2	0		2
pdListenTimeout [µT]	9 126	7 206	163 058	12 008	12 968	326 116
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	3	6	2	3	6
pMicroInitialOffset[A,B] [µT]	183	23		126	46	
pMicroPerCycle [µT]	4 560	3 600	81 480	6 000	6 480	162 960
pOffsetCorrectionOut [µT]	153	45	153	226	45	226
pOffsetCorrectionStart [MT]	18	73	2 019	20	73	2 019
pRateCorrectionOut [µT]	3		49	4		98

Table 56 — Modification to basic configurations 2a and 2b for timing related configuration data – gdStaticSlot

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	2	3	1		3
gdStaticSlot [MT]	5	14	664	5	13	664
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	22	63	3 341	29	100	3 361
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	3		5	3		5
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	132	126	6 682	174	200	6 722
gdMacrotick [µs]	6	2		6	2	
gdNIT [MT]	7	21		14	61	41
pClusterDriftDamping [µT]	0		2	0		2
pdListenTimeout [µT]	10 568	10 088	534 882	6 966	8 006	269 042
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	3	4	2		4
pMicroInitialOffset[A,B] [µT]	166	6		83	3	
pMicroPerCycle [µT]	5 280	5 040	267 280	3 480	4 000	134 440
pOffsetCorrectionOut [µT]	154			117	45	117
pOffsetCorrectionStart [MT]	21	57	3 328	27	73	3 344
pRateCorrectionOut [µT]	4	4	161	3		81

Table 57 — Modification to basic configuration 3 for timing related configuration data – gdStaticSlot

Parameter	Modification to Basic Configuration 3		
	I	**II**	**III**
gdActionPointOffset [MT]	1		3
gdStaticSlot [MT]	8	22	660
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	70	210	3 341
gNumberOfMinislots	0		
gNumberOfStaticSlots	7		5
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	420	420	6 682
gdMacrotick [µs]	6	2	
gdNIT [MT]	14	56	41
pClusterDriftDamping [µT]	0		2
pdListenTimeout [µT]	16 812	16 812	267 442
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	2	3	5
pMicroInitialOffset[A,B] [µT]	62	22	
pMicroPerCycle [µT]	8 400	8 400	133 640
pOffsetCorrectionOut [µT]	118	45	118
pOffsetCorrectionStart [MT]	66	206	3 329
pRateCorrectionOut [µT]	6		81

The IUT is configured to transmit an additional frame in slot 3.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, it is verified (LT) that the IUT transmits its frames in slot 1 and slot 3 correctly, i.e. the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0' in slot 1 and the frame ID is set to 3, the cycle count to 7, the payload length to 0, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0' in slot 3. It is also verified (LT) that the interval between both frames equals 2 * *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ µT.

2) In cycle 7, 50 µT after each slot start and after the NIT start, it is verified (UT) that the slot counters for available channels (*vSlotCounter_A* and / or *vSlotCounter_B*) are incremented correctly in the IUT, i.e. that the slot counters are 1, 2, ..., *gNumberOfStaticSlots* in the static slot 1, 2, ..., *gNumberOfStaticSlots* of the cycle, and that the slot counters are 0 in the NIT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames correctly. The intervals between the IUT's frames are correct. The IUT's slot counters count correctly.

7.3.1.5 gdActionPointOffset

— **Test purpose**

Verify correct handling of the parameter *gdActionPointOffset*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 58, Table 59 and Table 60.

Table 58 — Modification to basic configurations 1a and 1b for timing related configuration data – gdActionPointOffset

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	32	63	1	32	63
gdStaticSlot [MT]	150			150		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	655			655		
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	4			4		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	655			655		
gdNIT [MT]	55			55		
pdAcceptedStartupRange [µT]	281			403		
pClusterDriftDamping [µT]	0	2		0	2	
pdListenTimeout [µT]	52 432			104 864		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	3	34	65	3	34	65
pMicroPerCycle [µT]	26 200			52 400		
pOffsetCorrectionOut [µT]	45	153		45	226	
pOffsetCorrectionStart [MT]	645			645		
pRateCorrectionOut [µT]	16			32		

Table 59 — Modification to basic configurations 2a and 2b for timing related configuration data – gdActionPointOffset

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	32	63	1	32	63
gdStaticSlot [MT]	150			150		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	656			656		
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	4			4		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	1 312			1 312		
gdNIT [MT]	56			56		
pdAcceptedStartupRange [µT]	283			221		
pClusterDriftDamping [µT]	0	2		0	2	
pdListenTimeout [µT]	105 024			52 512		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	33	64	2	33	64
pMicroPerCycle [µT]	52 480			26 240		
pOffsetCorrectionOut [µT]	45	154		45	117	
pOffsetCorrectionStart [MT]	646			646		
pRateCorrectionOut [µT]	32			16		

Table 60 — Modification to basic configuration 3 for timing related configuration data – gdActionPointOffset

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	32	63
gdStaticSlot [MT]	150		
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	656		
gNumberOfMinislots	0		
gNumberOfStaticSlots	4		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	1 312		
gdNIT [MT]	56		
pdAcceptedStartupRange [µT]	224		
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	52 512		
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3	34	65
pMicroPerCycle [µT]	26 240		
pOffsetCorrectionOut [µT]	45	118	
pOffsetCorrectionStart [MT]	646		
pRateCorrectionOut [µT]	16		

The IUT is configured to transmit an additional frame in slot 3.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, it is verified (LT) that the IUT transmits its frames in slot 1 and slot 3 correctly, i.e. the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0' in slot 1 and the frame ID is set to 3, the cycle count to 7, the payload length to 0, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0' in slot 3. It is also verified (LT) that the interval between the IUT's frame in slot 1 and the LT's frame in slot 2 equals *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT and that the interval between the LT's frame in slot 2 and the IUT's frame in slot 3 equals *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT.

2) In cycle 7, 50 µT after each slot start and after the NIT start, it is verified (UT) that the slot counters for available channels (*vSlotCounter_A* and / or *vSlotCounter_B*) are incremented correctly in the IUT, i.e. that the slot counters are 1, 2, ..., *gNumberOfStaticSlots* in the static slot 1, 2, ..., *gNumberOfStaticSlots* of the cycle, and that the slot counters are 0 in the NIT.

3) In cycle 8, the LT simulates an additional frame in static slot 4 starting *gdActionPointOffset* + 2 *gdBit* before the action point of slot 4.
It is verified (UT) that the valid frame flag is false, the syntax error flag is false, the content error flag is false and the boundary violation flag is true in the slot status data of slot 4 in the NIT of the cycle.

4) In cycle 9, the LT simulates an additional frame in static slot 4 starting *gdActionPointOffset* – *gdBit* before the action point of slot 4.
It is verified (UT) that the valid frame flag is true, the syntax error flag is false, the content error flag is false and the boundary violation flag is false in the slot status data of slot 4 in the NIT of the cycle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames correctly. The intervals between the IUT's frames and the LT's frame are correct. The IUT's slot counters count correctly.

7.3.1.6 gdMinislot

— **Test purpose**

Verify correct handling of the parameter *gdMinislot*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 61, Table 62 and Table 63.

Table 61 — Modification to basic configurations 1a and 1b for timing related configuration data – gdMinislot

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdMinislot [MT]	2	32	63	3	32	63
gdMinislotActionPointOffset [MT]	1	4		1	3	
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	165	1 360	2 600	205	1 365	2 600
gNumberOfMinislots	40			40		
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	990	1 360	2 600	615	2 730	2 600
gdMacrotick [µs]	6	1		3	2	1
gdNIT [MT]	12	10		12	14	9
pClusterDriftDamping [µT]	2			0	2	
pdListenTimeout [µT]	79 248	108 866	208 126	98 460	437 064	416 250
pLatestTx [Minislot]	37	30		36	30	
pMacroInitialOffset[A,B] [MT]	5	6		5		6
pMicroInitialOffset[A,B] [µT]	183	23		126	46	
pMicroPerCycle [µT]	39 600	54 400	104 000	49 200	218 400	208 000
pOffsetCorrectionStart [MT]	163	1 352	2 592	200	1 360	2 595
pRateCorrectionOut [µT]	24	33	63	30	132	125
pPayloadLengthDynMax [two-byte word]	2	127		2	127	

Table 62 — Modification to basic configurations 2a and 2b for timing related configuration data – gdMinislot

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdMinislot [MT]	2	32	63	2	32	63
gdMinislotActionPointOffset [MT]	1	4		1	3	
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	160	1 360	2 600	160	1 360	2 600
gNumberOfMinislots	40			40		
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	960	2 720	5 200	960	2 720	5 200
gdMacrotick [µs]	6	2		6	2	
gdNIT [MT]	12	14		12	14	
pClusterDriftDamping [µT]	2			0	2	
pdListenTimeout [µT]	76 848	217 732	416 250	38 424	108 866	208 126
pLatestTx [Minislot]	36	30		36	30	
pMacroInitialOffset[A,B] [MT]	4			4		
pMicroInitialOffset[A,B] [µT]	166	6		83	3	
pMicroPerCycle [µT]	38 400	108 800	208 000	19 200	54 400	104 000
pOffsetCorrectionStart [MT]	155	1 355	2 595	155	1 350	2 590
pRateCorrectionOut [µT]	24	66	125	12	33	63
pPayloadLengthDynMax [two-byte word]	2	127		2	127	

Table 63 — Modification to basic configuration 3 for timing related configuration data – gdMinislot

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdMinislot [MT]	2	32	63
gdMinislotActionPointOffset [MT]	1	3	
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	210	1 410	2 650
gNumberOfMinislots	40		
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	1 260	2 820	5 300
gdMacrotick [µs]	6	2	
gdNIT [MT]	12	14	
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	50 432	112 868	212 128
pLatestTx [Minislot]	30	20	
pMacroInitialOffset[A,B] [MT]	4	5	
pMicroInitialOffset[A,B] [µT]	62	22	
pMicroPerCycle [µT]	25 200	56 400	106 000
pOffsetCorrectionStart [MT]	200	1 400	2 640
pRateCorrectionOut [µT]	16	34	64
pPayloadLengthDynMax [two-byte word]	2	127	

The IUT is configured to transmit two additional frames in dynamic slot 3 and dynamic slot 5. The payload length of dynamic frames is 2 two-byte words, the payload 0x01 in byte 0, 0x02 in byte 1, ..., 0x04 in byte 3 for the frame in slot 3 and 0x05 in byte 0, 0x06 in byte 1, ..., 0x08 in byte 3 for the frame in slot 5.

The interval between the dynamic frames in slot 3 and slot 5 is:

$\Delta t(3, 5) = (\text{ceil}(($ *gdTSSTransmitter* + *cdFSS* + 50 + 2 * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 2) * (*gdMinislot* * *gdMacrotick*) / *pdMicrotick* + ξ µT

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, it is verified (LT) that the IUT transmits its frames in slots 1, slot 3 and slot 5 correctly, i.e. the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0' in slot 1 and the frame ID is set to 3 and 5, the cycle count to 7, the payload length to 2 two-byte words, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0' in slot 3 and slot 5. The payload of the dynamic frames in slot 5 and slot 3 is as given in the configuration. It is also verified (LT) that the interval between the IUT's frames in slot 3 and slot 5 equals:

Table 64 defines the modification to test execution basic configurations 1a and 1b for timing related configuration data – *gdMinislot*.

Table 64 — Modification to test execution basic configurations 1a and 1b for timing related configuration data – gdMinislot

Interval	Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
Δt(3,5) [µT]	2 400 + *ξξ*	5 120 + *ξξ*	10 080 + *ξ*	3 600 + *ξ*	20 480 + *ξ*	20 160 + *ξ*

Table 65 defines the modification to test execution basic configurations 2a and 2b for timing related configuration data – *gdMinislot*.

Table 65 — Modification to test execution basic configurations 2a and 2b for timing related configuration data – gdMinislot

Interval	Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
Δt(3,5) [µT]	2 880 + *ξ*	10 240 + *ξ*	20 160 + *ξ*	1 440 + *ξ*	5 120 + *ξ*	10 080 + *ξ*

Table 66 defines the modification to test execution basic configuration 3 for timing related configuration data – *gdMinislot*.

Table 66 — Modification to test execution basic configuration 3 for timing related configuration data – gdMinislot

Interval	Basic Configuration 3		
	I	II	III
Δt(3,5) [µT]	1 920 + *ξ*	5 120 + *ξ*	10 080 + *ξ*

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames correctly. The intervals between the IUT's frames are correct.

7.3.1.7 gNumberOfMinislots

— **Test purpose**

Verify correct handling of the parameter *gNumberOfMinislots*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 67, Table 68 and Table 69.

Table 67 — Modification to basic configurations 1a and 1b for timing related configuration data – gNumberOfMinislots

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	4	3	1	4		1
gdDynamicSlotIdlePhase [Minislot]	0	1		0	1	
gdMinislot [MT]	32	7	4	32	6	3
gdMinislotActionPointOffset [MT]	4	3	1	2		1
gdStaticSlot [MT]	35		13	35		13
gdSymbolWindow [MT]	40		0	40		0
gdSymbolWindowActionPointOffset [MT]	4	3	1	2		1
gMacroPerCycle [MT]	188	7 292	16 000	190	6 270	16 000
gNumberOfMinislots	2	1 024	3 992	2	1 024	5 322
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	188	7 292	16 000	190	6 270	16 000
gdNIT [MT]	14		6	14		8
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	15 050	583 712	1 280 770	30 420	1 003 804	2 561 538
pLatestTx [Minislot]	1	957	3 925	1	957	5 233
pMacroInitialOffset[A,B] [MT]	6	5	3	6		3
pMicroPerCycle [µT]	7 520	291 680	640 000	15 200	501 600	1 280 000
pOffsetCorrectionOut [µT]	153		45	226		45
pOffsetCorrectionStart [MT]	175	7 280	15 997	180	6 260	15 996
pRateCorrectionOut [µT]	5	176	385	10	302	769
pPayloadLengthDynMax [two-byte word]	4	127		4	127	

Table 68 — Modification to basic configurations 2a and 2b for timing related configuration data – gNumberOfMinislots

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	2		1	3	2	1
gdDynamicSlotIdlePhase [Minislot]	0	1		0	1	
gdMinislot [MT]	32	5	3	32	5	3
gdMinislotActionPointOffset [MT]	2		1	3	2	1
gdStaticSlot [MT]	33		12	33		13
gdSymbolWindow [MT]	23		0	23		0
gdSymbolWindowActionPointOffset [MT]	2		1	3	2	1
gMacroPerCycle [MT]	165	5 221	8 000	165	5 221	8 000
gNumberOfMinislots	2	1 024	2 657	2	1 024	2 656
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	330	10 442	16 000	330	10 442	16 000
gdNIT [MT]	12		5	12		6
pClusterDriftDamping [µT]	2		0	0		
pdListenTimeout [µT]	26 416	835 862	1 280 770	13208	417 932	640 386
pLatestTx [Minislot]	1	957	2 568	1	957	2 567
pMacroInitialOffset[A,B] [MT]	3		2	4	3	2
pMicroPerCycle [µT]	13 200	417 680	640 000	6600	208 840	320 000
pOffsetCorrectionOut [µT]	154		45	117	100	45
pOffsetCorrectionStart [MT]	160	5 211	7 998	155	5 211	7 997
pRateCorrectionOut [µT]	8	251	385	4	126	193
pPayloadLengthDynMax [two-byte word]	4	127		4	127	

Table 69 — Modification to basic configuration 3 for timing related configuration data – gNumberOfMinislots

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	3	2	1
gdDynamicSlotIdlePhase [Minislot]	0	1	2
gdMinislot [MT]	32	5	3
gdMinislotActionPointOffset [MT]	3	2	1
gdStaticSlot [MT]	58	58	22
gdSymbolWindow [MT]	24	24	0
gdSymbolWindowActionPointOffset [MT]	3	2	1
gMacroPerCycle [MT]	215	5 271	8 000
gNumberOfMinislots	2	1 024	2 650
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	430	10 542	16 000
gdNIT [MT]	11		6
pClusterDriftDamping [µT]	2	0	
pdListenTimeout [µT]	17 212	421 934	640 386
pLatestTx [Minislot]	1	891	2472
pMacroInitialOffset[A,B] [MT]	5	4	3
pMicroPerCycle [µT]	8 600	210 840	320 000
pOffsetCorrectionOut [µT]	118	100	45
pOffsetCorrectionStart [MT]	207	5 262	7 997
pRateCorrectionOut [µT]	6	127	193
pPayloadLengthDynMax [two-byte word]	3	127	

In Modification I and II, the IUT is configured to transmit a frame in dynamic slot *gNumberOfStaticSlots* + *pLatestTx* of cycle 7 and a receive buffer is set up for reception in slot *gNumberOfStaticSlots* + *pLatestTx*.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

The time interval during which a respective value of *vSlotCounter* may be read has to be specified by the IUT vendor.

1) In cycle 7, it is verified (LT) that the IUT transmits its frame in slot 1 correctly, i.e. the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0' and

   (I, II) that the IUT transmits a frame in slot *gNumberOfStaticSlots* + *pLatestTx* correctly, i.e. the frame starts (*gNumberOfStaticSlots* * *gdStaticSlot* – *gdActionPointOffset* + (*pLatestTx* – 1) * *gdMinislot* + *gdActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT, after the first falling edge at the beginning of the IUT's frame in slot 1, the frame ID is set to *gNumberOfStaticSlots* + *pLatestTx*, the cycle count to 7, the payload length set accordingly, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0' and

   (III) that the IUT does not transmit a dynamic frame.

2) In cycle (8 + o) mod *gCycleCountMax*, it is verified (UT) that *vSlotCounter* = k_n in slot k_n with k_n = i * n + o and *gNumberOfStaticSlots* < k_n < min(*gNumberOfStaticSlots* + *gNumberOfMinislots; cSlotIDMax)* (i $\in$ [1; *gMacroPerCycle*], n = 0, 1, 2,... and o = 0 and is incremented by 1 in each cycle until o equals i–1).
   It is also verified that the slot counters are 0 (I, II) after the last minislot at the end of the dynamic segment or (III) after the minislot during which the slot counter reached *cSlotIDMax*.

3) (I, II) In cycle (8 + i) mod *gCycleCountMax*, the LT simulates an additional dynamic frame in slot *gNumberOfStaticSlots* + *pLatestTx*.
   It is verified (UT), 2 000 µT after the end of the slot, that the frame contents data corresponds to the contents of the frame as simulated by the LT in slot *gNumberOfStaticSlots* + *pLatestTx*.

4) (III) It is verified (LT) that the interval between the IUT's frame in slot 1 of cycle (8 + i) mod *gCycleCountMax* and cycle (9 + i) mod *gCycleCountMax* is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames and receives the frames in slot *gNumberOfStaticSlots* + *pLatestTx* of cycle (8 + i) mod *gCycleCountMax* correctly. The IUT's slot counters count correctly.

7.3.1.8 pLatestTx

— Test purpose

Verify correct handling of the parameter *pLatestTx*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 70, Table 71 and Table 72.

Table 70 — Modification to basic configurations 1a and 1b for timing related configuration data – pLatestTx

Parameter	Modification to Basic Configuration					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1			1		
gdDynamicSlotIdlePhase [Minislot]	1			1		
gdMinislot [MT]	4			3		
gdMinislotActionPointOffset [MT]	1			1		
gdStaticSlot [MT]	13			13		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	16 000			16 000		
gNumberOfMinislots	3 992			5 323		
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	16 000			16 000		
gdNIT [MT]	6			5		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	1 280 770			2 561 538		
pLatestTx [Minislot]	1	1 024	3 988	1	1 024	5 318
pMacroInitialOffset[A,B] [MT]	3			3		
pMicroPerCycle [µT]	640 000			1 280 000		
pOffsetCorrectionOut [µT]	45			45		
pOffsetCorrectionStart [MT]	15 997			15 997		
pRateCorrectionOut [µT]	385			769		
pPayloadLengthDynMax [two-byte word]	0			0		

Table 71 — Modification to basic configurations 2a and 2b for timing related configuration data – pLatestTx

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1			1		
gdDynamicSlotIdlePhase [Minislot]	1			1		
gdMinislot [MT]	3			3		
gdMinislotActionPointOffset [MT]	1			1		
gdStaticSlot [MT]	13			13		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	8 000			8 000		
gNumberOfMinislots	2 656			2 656		
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	16 000			16 000		
gdNIT [MT]	6			6		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	1 280 770			640 386		
pLatestTx [Minislot]	1	1 024	2 651	1	1 024	2 651
pMacroInitialOffset[A,B] [MT]	2			2		
pMicroPerCycle [µT]	640 000			320 000		
pOffsetCorrectionOut [µT]	45			45		
pOffsetCorrectionStart [MT]	7 997			7 997		
pRateCorrectionOut [µT]	385			193		
pPayloadLengthDynMax [two-byte word]	0			0		

Table 72 — Modification to basic configuration 3 for timing related configuration data – pLatestTx

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1		
gdDynamicSlotIdlePhase [Minislot]	2		
gdMinislot [MT]	3		
gdMinislotActionPointOffset [MT]	1		
gdStaticSlot [MT]	22		
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	8 000		
gNumberOfMinislots	2 650		
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	16 000		
gdNIT [MT]	6		
pClusterDriftDamping [µT]	0		
pdListenTimeout [µT]	640 386		
pLatestTx [Minislot]	1	1 024	2 642
pMacroInitialOffset[A,B] [MT]	3		
pMicroPerCycle [µT]	320 000		
pOffsetCorrectionOut [µT]	45		
pOffsetCorrectionStart [MT]	7 997		
pRateCorrectionOut [µT]	193		
pPayloadLengthDynMax [two-byte word]	0		

The IUT is configured to transmit frames in dynamic slot *gNumberOfStaticSlots* + *pLatestTx* for Modification I and II and in dynamic slot n_{dyn} of cycle 7 and dynamic slot n_{dyn}+1 of cycle 8 for Modification III. The payload length of dynamic frames is 0 bytes unless otherwise specified.

Table 73 defines auxiliary variables.

Table 73 — Auxiliary variables

Auxiliary variable	Basic Configuration			
	1a	**1b**	**2a & 2b**	**3**
m	58	58	29	14
n_{dyn}	104	158	72	152

— Preamble (setup state)

Preamble II.

— Test execution

1) In cycle 7, it is verified (LT) that the IUT transmits its frame in slot 1 correctly, i.e. the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0'.

   (I, II) It is also verified (LT) that the IUT transmits its frame in slot *gNumberOfStaticSlots* + *pLatestTx* correctly, i.e. and the frame ID is set to *gNumberOfStaticSlots* + *pLatestTx*, the cycle count to 7, the payload length to 0, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0'.

   (III) In cycle 7, the LT simulates m additional dynamic frames, each with a payload of 127 two-byte words in dynamic slots 3 to m+2. The payload is 0x01 in byte 0, 0x02 in byte 1, ..., and 0xFE in byte 253. It is also verified (LT) that the IUT transmits its frame in slot n_{dyn} correctly, i.e. the frame ID is set to n_{dyn}, the cycle count to 7, the payload length to 0, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0'.

2) (III) In cycle 8, the LT simulates m additional dynamic frames, each with a payload of 127 two-byte words in dynamic slots 3 to m+2. The payload is 0x01 in byte 0, 0x02 in byte 1, ..., and 0xFE in byte 253. It is verified (LT) that the IUT does not transmit a frame in the dynamic segment.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames correctly.

7.3.1.9 gdMinislotActionPointOffset

— Test purpose

Verify correct handling of the parameter *gdMinislotActionPointOffset*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 74, Table 75 and Table 76.

Table 74 — Modification to basic configurations 1a and 1b for timing related configuration data – gdMinislotActionPointOffset

Parameter	Modification to Basic Configuration					
	1a			1b		
	I	II	III	I	II	III
gdMinislot [MT]	63			63		
gdMinislotActionPointOffset [MT]	1	15	31	1	15	31
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	1 400			1 400		
gNumberOfMinislots	20			20		
gNumberOfStaticSlots	3			3		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	1 400			1 400		
gdNIT [MT]	32	35		32	35	
pClusterDriftDamping [µT]	0	2		0	2	
pdListenTimeout [µT]	112 068			224 136		
pLatestTx [Minislot]	2			2		
pMicroPerCycle [µT]	56 000			112 000		
pOffsetCorrectionStart [MT]	1 390			1 390		
pRateCorrectionOut [µT]	34			68		

Table 75 — Modification to basic configurations 2a and 2b for timing related configuration data – gdMinislotActionPointOffset

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdMinislot [MT]	63			63		
gdMinislotActionPointOffset [MT]	1	15	31	1	15	31
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	1 375			1 375		
gNumberOfMinislots	20			20		
gNumberOfStaticSlots	3			3		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	2 750			2 750		
gdNIT [MT]	14	16		14	16	
pClusterDriftDamping [µT]	0	2		0	2	
pdListenTimeout [µT]	220 134			110 068		
pLatestTx [Minislot]	2			2		
pMicroPerCycle [µT]	110 000			55 000		
pOffsetCorrectionStart [MT]	1 370			1 365		
pRateCorrectionOut [µT]	67			34		

Table 76 — Modification to basic configuration 3 for timing related configuration data – gdMinislotActionPointOffset

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdMinislot [MT]	63		
gdMinislotActionPointOffset [MT]	1	15	31
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	1 450		
gNumberOfMinislots	20		
gNumberOfStaticSlots	3		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	2 900		
gdNIT [MT]	14	16	
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	116 070		
pLatestTx [Minislot]	2		
pMicroPerCycle [µT]	58 000		
pOffsetCorrectionStart [MT]	1 440		
pRateCorrectionOut [µT]	35		

The IUT is configured to transmit two additional frames in static slot 3 and in dynamic slot 4.

The interval between the IUT's frame in slot 4 and the LT's frame in slot 5 is:

$\Delta t(4, 5) = (\text{ceil}((gdTSSTransmitter + cdFSS + 50 + 1 * 20 + 30 + cdFES + 1) * gdBit / (gdMinislot * gdMacrotick)) + gdDynamicSlotIdlePhase + 1) * (gdMinislot * gdMacrotick) / pdMicrotick + \xi\ \mu T + \xi_{IUT}\ \mu T$

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, the LT simulates a frame in dynamic slot 5 and it is verified (LT) that
the IUT transmits its frames in slot 1, slot 3 and slot 4 correctly, i.e. the frame ID is set to 1, the cycle count to 7, the payload length to 0, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0' in slot 1 and the frame ID is set to 3 and 4, the cycle count to 7, the payload length to 0 and 1, the payload to 0x0102 (slot 4 only), the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0' in slot 3 and slot 4,
the interval between the IUT's frame in slot 4 and the LT's frame in slot 5 is $\Delta t(4,5)$ and the interval between the IUT's frames in slots 3 and slot 4 is $\Delta t(3,4)$:

Table 77 defines the modification to test execution basic configuration of timing related configuration data – *gdMinislot*.

Table 77 — Modification to test execution basic configuration of timing related configuration data – gdMinislot

Interval	Modification		
	I	II	III
Δt(3,4) [µT]	*gdStaticSlot * gdMacrotick / pdMicrotick + ξ*	*(gdStaticSlot + gdMinislotActionPointOffset – gdActionPointOffset) * gdMacrotick / pdMicrotick + ξ*	

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames correctly. The intervals between the IUT's frames in slot 3 and slot 4 and the intervals between the IUT's frame in slot 4 and the LT's frame in slot 5 are correct.

7.3.1.10 gdDynamicSlotIdlePhase

— **Test purpose**

Verify the correct handling of the parameter *gdDynamicSlotIdlePhase*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 78, Table 79 and Table 80.

Table 78 — Modification to basic configurations 1a and 1b for timing related configuration data – gdDynamicSlotIdlePhase

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gMacroPerCycle [MT]	2 095			2 095		
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	2 095			2 095		
pClusterDriftDamping [µT]	0	2		2		
pdListenTimeout [µT]	167 702			335 402		
pMicroPerCycle [µT]	83 800			167 600		
pOffsetCorrectionStart [MT]	2 085			2 085		
pRateCorrectionOut [µT]	51			101		
pPayloadLengthDynMax [two-byte word]	1			1		

Table 79 — Modification to basic configurations 2a and 2b for timing related configuration data – gdDynamicSlotIdlePhase

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gMacroPerCycle [MT]	1 081			1 081		
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	2 162			2 162		
pClusterDriftDamping [µT]	2			0	2	
pdListenTimeout [µT]	173 064			86 532		
pMicroPerCycle [µT]	86 480			43 240		
pOffsetCorrectionStart [MT]	1 075			1 074		
pRateCorrectionOut [µT]	52			26		
pPayloadLengthDynMax [two-byte word]	1			1		

Table 80 — Modification to basic configuration 3 for timing related configuration data – gdDynamicSlotIdlePhase

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2
gMacroPerCycle [MT]	1 166		
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	2 332		
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	93 336		
pMicroPerCycle [µT]	46 640		
pOffsetCorrectionStart [MT]	1 159		
pRateCorrectionOut [µT]	28		
pPayloadLengthDynMax [two-byte word]	1		

The IUT is configured to receive frames in slot 4 and to transmit additional frames in dynamic slots 3 and 5.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, the LT simulates an additional frame in dynamic slot 4.
It is verified (LT) that the IUT transmits its frames in slots 1, 3 and slot 5 correctly, i.e. the frame ID, the cycle count, the payload length, the sync frame indicator, the startup frame indicator, the null frame indicator and the payload preamble indicator are set accordingly.
It is verified (LT) that the interval between the IUT's frames in slot 3 and slot 5 is:
$\Delta t(3, 5) = 2 * (\text{ceil}((gdTSSTransmitter + cdFSS + 50 + 1 * 20 + 30 + cdFES + 1) * gdBit / (gdMinislot * gdMacrotick)) + gdDynamicSlotIdlePhase + 1) * (gdMinislot * gdMacrotick) / pdMicrotick + \xi\ \mu T$

2) In the NIT of cycle 7, it is verified (UT) that the frame contents data of slot 4 corresponds to the contents of the frame as simulated by the LT in slot 4 of cycle 7.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames correctly and receives the LT's frame in slot 4 correctly.

7.3.1.11 gdSymbolWindow

— Test purpose

Verify correct handling of the parameter *gdSymbolWindow*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 81.

Table 81 — Modification to basic configurations for timing related configuration data – gdSymbolWindow

Parameter	Modification to Basic Configuration								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdSymbolWindow [MT]	15	64	161	11	64	162	16	64	145
gNumberOfMinislots	219	216	206	140	134	120	145	139	128
gdNIT [MT]	39	17	10	24	13	13	19	13	9
pClusterDriftDamping [µT]	0			0			0		
pLatestTx [Minislot]	188	185	175	101	95	81	68	62	51
pOffsetCorrectionStart [MT]	4 990		4 992	2 491			2 492		

— Preamble (setup state)

Preamble I.

— Test execution

1) The UT requests the IUT to transmit an MTS in the symbol window of cycle 8.

2) It is verified (LT) that the IUT transmits a frame in slot 2 of cycle 8 starting at t_{AP} = (*gdStaticSlot* + *gdActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + *ξ* + ξ_{IUT} µT after the cycle start.

3) It is verified (LT) that the IUT transmits a valid MTS of length (*gdTSSTransmitter* + *cdCAS*) * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT starting at the symbol window action point t_{SWAP} = (*gNumberOfStaticSlots* * *gdStaticSlot* + *gNumberOfMinislots* * *gdMinislot* + *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + *ξ* + ξ_{IUT} µT after the cycle start.

4) In cycle 9, the LT starts to simulate an MTS 1 *gdBit* after the symbol window start.
In the static segment of cycle 10, it is verified (UT) that the flag *vSS!ValidMTS* is true in the symbol window status and that the boundary violation flag / indicator is false in the symbol window status, the NIT status and the aggregated channel status.
The UT clears the aggregated channel status after its verification.

5) In cycle 10, the LT starts to simulate an MTS 2 *gdBit* before the symbol window start.
In the static segment of cycle 11, it is verified (UT) that the flag *vSS!ValidMTS* is false in the symbol

window status, the boundary violation flag / indicator is true in the symbol window status and the aggregated channel status and that the boundary violation flag is false in the NIT status.
The UT clears the aggregated channel status after its verification.

6) In cycle 11, the LT starts to simulate an MTS *gdTSSTransmitter* + *cdCAS* + *cChannelIdleDelimiter* + 2 *gdBit* before the symbol window end.
In the static segment of cycle 12, it is verified (UT) that the flag *vSS!ValidMTS* is true in the symbol window status and that the boundary violation flag / indicator is false in the symbol window status, the NIT status and the aggregated channel status.
The UT clears the aggregated channel status after its verification.

7) In cycle 12, the LT starts to simulate an MTS *gdTSSTransmitter* + *cdCAS* + *cChannelIdleDelimiter* – 1 *gdBit* before the symbol window end.
In the static segment of cycle 13, it is verified (UT) that the flag *vSS!ValidMTS* is true in the symbol window status and that the boundary violation flag / indicator is true in the symbol window status, the NIT status and the aggregated channel status.
The UT clears the aggregated channel status after its verification.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits the MTS correctly. The measurement of the symbol window duration via the boundary violation indication is in accordance with the configuration parameter *gdSymbolWindow* ± 1 *gdBit*.

7.3.1.12 pLatestTx violation

— Test purpose

Verify correct transmission abortion in the dynamic segment to ensure that the CHIRP is detected during the dynamic slot idle phase.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 82.

Table 82 — Modification to basic configurations for timing related configuration data – pLatestTx violation

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdDynamicSlotIdlePhase [Minislot]	2	2	2
pLatestTx [Minislot]	215	136	139
pPayloadLengthDynMax [two-byte word]	4	2	2

n_{dyn} = *gNumberOfStaticSlots* + *pLatestTx*.

The IUT is configured to transmit a frame in the dynamic slot n_{dyn}.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In the static segment of cycle 7, the IUT is configured to transmit a dynamic frame in slot n_{dyn}, with a payload length of

   (a) 127 two-byte words,

   (b) *pPayloadLengthDynMax* +1 two-byte words,

   (c) *pPayloadLengthDynMax* two-byte words and

   (d) 0 two-byte word.

2) (a, b) In cycle 7, it is verified (LT) that the IUT starts the transmission of its frame in slot n_{dyn}, and aborts the frame transmission at the action point of minislot *gNumberOfMinislots* – *gdDynamicSlotIdlePhase*. It is also verified (LT) that TxD becomes high at the action point of minislot *gNumberOfMinislots* – *gdDynamicSlotIdlePhase* and TxEN becomes high 1 *gdBit* after TxD.
   (c, d) In cycle 7, it is verified (LT) that the IUT transmits its frame in slot n_{dyn} correctly.

3) In the NIT of cycle 7, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*), that *vLastDynTxSlot* is n_{dyn} and that

   (a, b) the *pLatestTx* violation status indicator is true and

   (c, d) the *pLatestTx* violation status indicator is false.
   The UT resets the *pLatestTx* violation status indicator after its verification and verifies that this indicator has been reset.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts ongoing dynamic frame transmission in minislot *gNumberOfMinislots* – *gdDynamicSlotIdlePhase*, sets the *pLatestTx* violation status indicator and *vLastDynTxSlot* accordingly and stays in *POC:normal active* state.

7.3.1.13 gdSymbolWindowActionPointOffset

— **Test purpose**

Verify the correct handling of the parameter *gdSymbolWindowActionPointOffset*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 83.

Table 83 — Modification to basic configurations for timing related configuration data – gdSymbolWindowActionPointOffset

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdActionPointOffset [MT]	1			1			1		
gdMinislotActionPointOffset [MT]	1			1			1		
gdSymbolWindow [MT]	157			142			143		
gdSymbolWindowActionPointOffset [MT]	1	31	63	1	31	63	1	31	63
gNumberOfMinislots	206			123			128		
pClusterDriftDamping [µT]	0			0			0		
pLatestTx [Minislot]	175			84			51		
pMacroInitialOffset[A,B] [MT]	3			2			3		

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) The UT requests the IUT to transmit an MTS in the symbol window of cycle 7 and a WUDOP in the symbol window of cycle 9.

2) In cycle 7 it is verified (LT) that the IUT transmits an MTS starting at (*gdStaticSlot* * *gNumberOfStaticSlots* + *gdMinislot* * *gNumberOfMinislots* + *gdSymbolWindowActionPointOffset*) * (*gdMacrotick* / *pdMicrotick*) + ξ + ξ_{IUT}.

3) In cycle 8 the LT simulates an MTS starting at *gdStaticSlot* * *gNumberOfStaticSlots* + *gdMinislot* * *gNumberOfMinislots* + *gdSymbolWindowActionPointOffset*. It is verified that the interval between the LT's MTS in cycle 8 and the IUT's MTS in cycle 7 equals *pMicroPerCycle* + ξ + ξ_{IUT} µT.

4) In cycle 9 it is verified (LT) that the IUT transmits a WUDOP starting at (*gdStaticSlot* * *gNumberOfStaticSlots* + *gdMinislot* * *gNumberOfMinislots* + *gdSymbolWindowActionPointOffset*) * (*gdMacrotick* / *pdMicrotick*) + ξ + ξ_{IUT}. It is verified that the interval between the LT's MTS in cycle 8 and the IUT's WUDOP in cycle 9 equals *pMicroPerCycle* + ξ + ξ_{IUT} µT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its MTS and WUDOP correctly starting at the symbol window action point.

7.3.2 Protocol operation related configuration data

7.3.2.1 gColdstartAttempts

— Test purpose

Verify the correct handling of the *gColdstartAttempts* parameter.

This test is performed in test case 7.6.3.1.

7.3.2.2 gListenNoise

— Test purpose

Verify the correct handling of the *gListenNoise* parameter.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 84.

Table 84 — Modification to basic configurations for protocol operation related configuration data – gListenNoise

Parameter	Modification		
	I	II	III
gListenNoise	2	9	16

The IUT is configured as coldstart node sending startup frames in slot 1.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— Test execution

1) The LT simulates permanent noise (low phase) on the available channel(s) before the UT initiates the IUT's startup procedure.

2) The UT initiates the startup procedure with the CHI command RUN.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

4) The LT stops noise simulation on the available channel(s) earliest 2 500 µT and latest *pdListenTimeout* / 2 – 2 500 µT after the CHI command RUN.

5) 100 *gdBit* after the end of the LT's noise, the LT simulates a CAS of length *gdTSSTransmitter* + *cdCAS* on the available channel(s).

6) *pdListenTimeout* / 2 after the end of the LT's CAS, the LT simulates permanent noise (low phase) of length (*gListenNoise* – 1) * *pdListenTimeout* on the available channel(s).

7) It is verified (LT) that the IUT starts CAS transmission (*cVotingDelay* + *cStrobeOffset* +*cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* +*cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the LT finished CAS simulation.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT starts CAS transmission (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* +*cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the LT finished CAS simulation.

7.3.2.3 pdListenTimeout

— Test purpose

Verify the correct interpretation of the *pdListenTimeout* parameter.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 85, Table 86 and Table 87.

Table 85 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pdListenTimeout

Parameter	Modification to Basic Configuration					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	4	3		1	2	
gdMinislotActionPointOffset [MT]	4	3		1	2	
gdStaticSlot [MT]	19	17	17	13	15	15
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	84	4 673	16 000	46	4 655	16 000
gNumberOfMinislots	0	512	1 770	0	512	1 772
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003		0,0015	0,0003		0,0015
gdCycle [µs]	84	4 673	16 000	46	4 655	16 000
gdNIT [MT]	46	31	36	20	17	22
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	6 726	374 066	1 283 846	7 366	745 248	2 567 692
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	6	5		3	4	
pMicroPerCycle [µT]	3 360	186 920	640 000	3 680	372 400	1 280 000
pOffsetCorrectionOut [µT]	153			45	226	
pOffsetCorrectionStart [MT]	76	4 665	15 992	45	4 650	15 990
pRateCorrectionOut [µT]	3	113	1 923	3	224	3 846

Table 86 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pdListenTimeout

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	2		1	2	
gdMinislotActionPointOffset [MT]	1	2		1	2	
gdStaticSlot [MT]	12	14		13	15	
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	44	3 630	8 000	65	3 649	8 000
gNumberOfMinislots	0	512	1 135	0	512	1 130
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003		0,0015	0,0004	0,0003	0,0015
gdCycle [µs]	88	7 260	16 000	130	7 298	16 000
gdNIT [MT]	20	18	27	39	35	60
pdListenTimeout [µT]	7 046	581 150	1 283 846	5 206	292 096	641 924
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	3		2	3	
pMicroPerCycle [µT]	3 520	290 400	640 000	2 600	145 960	320 000
pOffsetCorrectionOut [µT]	45	154		45	100	
pOffsetCorrectionStart [MT]	40	3 625	7 995	60	3 643	7 994
pRateCorrectionOut [µT]	3	175	1 923	3	88	962

Table 87 — Modification to basic configuration 3 for protocol operation related configuration data – pdListenTimeout

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	2	
gdMinislotActionPointOffset [MT]	1	2	
gdStaticSlot [MT]	22	24	24
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	83	3 667	8 000
gNumberOfMinislots	0	512	1 130
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gClockDeviationMax	0,0009	0,0003	0,0015
gdCycle [µs]	166	7 334	16 000
gdNIT [MT]	39	35	42
pClusterDriftDamping [µT]	0		
pdListenTimeout [µT]	6 652	293 538	641 924
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3	4	
pMicroPerCycle [µT]	3 320	146 680	320 000
pOffsetCorrectionOut [µT]	45	100	
pOffsetCorrectionStart [MT]	80	3 661	7 993
pRateCorrectionOut [µT]	6	89	962

The IUT is configured as coldstart node sending startup frames in slot 1.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— **Test execution**

1) The LT generates permanent noise (low phase) on the available channel(s) before the UT initiates the IUT's startup procedure.

2) The UT initiates the startup procedure with the CHI command RUN.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

4) The LT stops noise generation earliest 2 500 µT and latest *pdListenTimeout* / 2 – 2 500 µT after the CHI command RUN on the available channel(s). The noise shall be a multiple of *gdBit*.

5) It is verified (LT) that the IUT starts CAS transmission (*cVotingDelay* + *cStrobeOffset* +*cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the LT stopped noise generation.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT starts CAS transmission (*cVotingDelay* + *cStrobeOffset* +*cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter – 1)* * *gdBit* / *pdMicrotick* + *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the LT stopped noise generation.

7.3.2.4 pMacroInitialOffset

— **Test purpose**

Verify correct interpretation of the parameters *pMacroInitialOffset[A]* and *pMacroInitialOffset[B]*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 88, Table 89 and Table 90.

Table 88 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pMacroInitialOffset

Parameter	Modification to Basic Configuration					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	32	63	1	32	63
gdCASRxLowMax [gdBit]	95	97	105	95	97	105
gdMinislot [MT]	9		16	9		15
gdMinislotActionPointOffset [MT]	4		6	4		6
gdStaticSlot [MT]	150			150		
gdSymbolWindow [MT]	0			0		
gdTSSTransmitter [gdBit]	10	11	15	10	11	15
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	33		32	33		32
gPayloadLengthStatic [two-byte word]	0			0		
gdNIT [MT]	50		200	50		200
pdAcceptedStartupRange [µT]	281		561	403		1 044
pClusterDriftDamping [µT]	0			0		
pDecodingCorrection [µT]	48	52	68	96	104	136
pDelayCompensation[A,B] [µT]	5		100	10		185
pKeySlotID	3			3		
pKeySlotUsedForStartup	false			false		
pKeySlotUsedForSync	false			false		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	3	34	68	3	34	68
pMicroInitialOffset[A,B] [µT]	27	23	32	54	46	79
pOffsetCorrectionOut [µT]	45	153	350	45	226	620
pOffsetCorrectionStart [MT]	4 990		4 982	4 990		4 980

Table 89 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pMacroInitialOffset

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	32	63	1	32	63
gdCASRxLowMax [gdBit]	80	84		80	84	
gdMinislot [MT]	7		8	7		10
gdMinislotActionPointOffset [MT]	3			3		4
gdStaticSlot [MT]	150			150		
gdSymbolWindow [MT]	0			0		
gdTSSTransmitter [gdBit]	6	8		6	8	
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	16			16		
gPayloadLengthStatic [two-byte word]	0			0		
gdNIT [MT]	100			100		
pdAcceptedStartupRange [µT]	283		544	221		311
pClusterDriftDamping [µT]	0			0		
pDecodingCorrection [µT]	64	80		32	40	
pDelayCompensation[A,B] [µT]	10		85	5		45
pKeySlotID	3			3		
pKeySlotUsedForStartup	false			false		
pKeySlotUsedForSync	false			false		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	34	66	2	34	66
pMicroInitialOffset[A,B] [µT]	6	70	75	3	35	
pOffsetCorrectionOut [µT]	45	154	350	45	117	80
pOffsetCorrectionStart [MT]	2 491			2 491		2 480

Table 90 — Modification to basic configuration 3 for protocol operation related configuration data – pMacroInitialOffset

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	32	63
gdCASRxLowMax [gdBit]	72		74
gdMinislot [MT]	7		
gdMinislotActionPointOffset [MT]	3		
gdStaticSlot [MT]	150		
gdSymbolWindow [MT]	0		
gdTSSTransmitter [gdBit]	4		5
gNumberOfMinislots	0		
gNumberOfStaticSlots	16		
gPayloadLengthStatic [two-byte word]	8		
gdNIT [MT]	100		
pdAcceptedStartupRange [µT]	234		
pClusterDriftDamping [µT]	0		
pDecodingCorrection [µT]	48		56
pDelayCompensation[A,B] [µT]	10		26
pKeySlotID	3		
pKeySlotUsedForStartup	false		
pKeySlotUsedForSync	false		
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3	34	66
pMicroInitialOffset[A,B] [µT]	22		38
pOffsetCorrectionOut [µT]	45	118	
pOffsetCorrectionStart [MT]	2 492	2 492	2 492

The IUT is an integrating node and is configured to send static frames in slot 3 on the available channel(s) for all instances.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

— **Test execution**

1) The LT begins to simulate startup frames in slots 1 and 2 of cycles 0 to 6 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) In cycle 4 before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

3) In cycle 6 before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

4) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

5) In cycle 6, it is verified (LT) that the interval between the IUT's frame in slot 3 and the LT's frame in slot 2 is *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* – *(pDelayCompensation[Ch]* – *10 / pSamplesPerMicrotick)* + ξ + ξ_{IUT} µT on the respective channel.

Figure 22 depicts the *pMacroInitialOffset*.

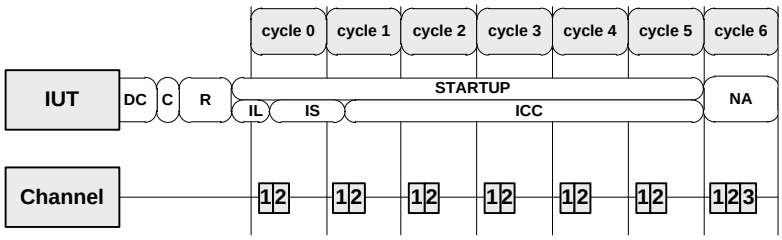


Figure 22 — pMacroInitialOffset

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs appropriate offset correction according to the value of parameters *pMacroInitialOffset[A]* and *pMacroInitialOffset[B]*.

7.3.2.5 pRateCorrectionOut during startup (1)

— Test purpose

Verify correct interpretation of the parameter *pRateCorrectionOut* when the maximum positive drift is exceeded during integration.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 91, Table 92 and Table 93.

Table 91 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pRateCorrectionOut during startup (1)

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	4			3		
gdMinislotActionPointOffset [MT]	4			3		
gdStaticSlot [MT]	21	19		17		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	84	8 320	16 000	56	8 320	16 000
gNumberOfMinislots	0	914	1 769	0	915	1 770
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003	0,0015		0,0003	0,0015	
gdCycle [µs]	84	8 320	16 000	56	8 320	16 000
gdNIT [MT]	42	56	41	22	51	36
pClusterDriftDamping [µT]	2			2		
pdListenTimeout [µT]	6 726	667 600	1 283 846	8 966	1 335 200	2 567 692
pKeySlotID	2			2		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	6			5		
pMicroPerCycle [µT]	3 360	332 800	640 000	4 480	665 600	1 280 000
pOffsetCorrectionOut [µT]	153			226		
pOffsetCorrectionStart [MT]	62	8 300	15 990	45	8 300	15 990
pRateCorrectionOut [µT]	3	1 000	1 923	3	2 000	3 846

Table 92 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pRateCorrectionOut during startup (1)

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	2			3		
gdMinislotActionPointOffset [MT]	2			3		
gdStaticSlot [MT]	14			16		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	49	4 160	8 000	73	4 160	8 000
gNumberOfMinislots	0	585	1 135	0	585	1 130
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003	0,0015		0,0004	0,0015	
gdCycle [µs]	98	8 320	16 000	146	8 320	16 000
gdNIT [MT]	21	37	27	41	33	58
pClusterDriftDamping [µT]	2			2		
pdListenTimeout [µT]	7 846	667 600	1 283 846	5 846	333 800	641 924
pKeySlotID	2			2		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	3			4		
pMicroPerCycle [µT]	3 920	332 800	640 000	2 920	166 400	320 000
pOffsetCorrectionOut [µT]	154	160		117		
pOffsetCorrectionStart [MT]	45	4 155	7 995	62	4 150	7 990
pRateCorrectionOut [µT]	3	1 000	1 923	3	500	962

Table 93 — Modification to basic configuration 2 for protocol operation related configuration data – pRateCorrectionOut during startup (1)

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	3	
gdMinislotActionPointOffset [MT]	1	3	
gdStaticSlot [MT]	22	26	
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	83	4 160	8 000
gNumberOfMinislots	0	581	1 130
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gClockDeviationMax	0,0009	0,0015	0,0015
gdCycle [µs]	166	8 320	16 000
gdNIT [MT]	39	41	38
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	6 652	333 800	641 924
pKeySlotID	2		
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3	5	
pMicroPerCycle [µT]	3 320	166 400	320 000
pOffsetCorrectionOut [µT]	45	130	
pOffsetCorrectionStart [MT]	80	4 150	7 990
pRateCorrectionOut [µT]	6	500	962

The IUT is configured as coldstart node sending startup frames in slot 2 on the available channel(s) for all instances.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT simulates a CAS *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) The LT simulates startup frames in slot 1 of cycles 0 to 7. The LT's cycle length is (*pMicroPerCycle* + (*pRateCorrectionOut* + 1)) * *pSamplesPerMicrotick* + 1 σT.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,7 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

6) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2,7 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

7) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 5,7 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 6,7 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

9) It is verified (LT) that the IUT does not send any frames.

Figure 23 depicts the *pRateCorrectionOut* during startup (1).

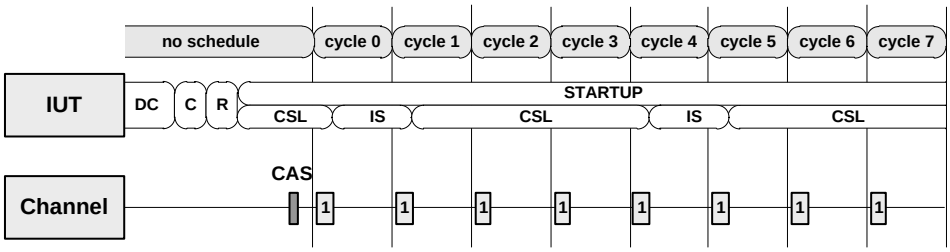


Figure 23 — pRateCorrectionOut during startup (1)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT does not perform a startup. The IUT does not accept the corresponding odd startup frames of the LT because the LT's cycle is too long.

7.3.2.6 pRateCorrectionOut during startup (2)

— Test purpose

Verify correct interpretation of the parameter *pRateCorrectionOut* when the maximum positive drift is not exceeded during integration.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 94, Table 95 and Table 96.

Table 94 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pRateCorrectionOut during startup (2)

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	4			3		
gdMinislotActionPointOffset [MT]	4			3		
gdStaticSlot [MT]	21	19		17		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	84	8 320	16 000	56	8 320	16 000
gNumberOfMinislots	0	914	1 769	0	915	1 770
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003	0,0015		0,0003	0,0015	
gdCycle [µs]	84	8 320	16 000	56	8 320	16 000
gdNIT [MT]	42	56	41	22	51	36
pClusterDriftDamping [µT]	2			2		
pdListenTimeout [µT]	6 726	667 600	1 283 846	8 966	1 335 200	2 567 692
pKeySlotID	2			2		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	6			5		
pMicroPerCycle [µT]	3 360	332 800	640 000	4480	665 600	1 280 000
pOffsetCorrectionOut [µT]	153			226		
pOffsetCorrectionStart [MT]	62	8 300	15 990	45	8 300	15 990
pRateCorrectionOut [µT]	3	1 000	1 923	3	2 000	3 846

Table 95 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pRateCorrectionOut during startup (2)

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	2			3		
gdMinislotActionPointOffset [MT]	2			3		
gdStaticSlot [MT]	14			16		
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	49	4 160	8 000	73	4 160	8 000
gNumberOfMinislots	0	585	1 135	0	585	1 130
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003	0,0015		0,0004	0,0015	
gdCycle [µs]	98	8 320	16 000	146	8 320	16 000
gdNIT [MT]	21	37	27	41	33	58
pClusterDriftDamping [µT]	2			2		
pdListenTimeout [µT]	7 846	667 600	1 283 846	5 846	333 800	641 924
pKeySlotID	2			2		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	3			4		
pMicroPerCycle [µT]	3 920	332 800	640 000	2 920	166 400	320 000
pOffsetCorrectionOut [µT]	154	160		117		
pOffsetCorrectionStart [MT]	45	4 155	7 995	62	4 150	7 990
pRateCorrectionOut [µT]	3	1 000	1 923	3	500	962

Table 96 — Modification to basic configuration 3 for protocol operation related configuration data – pRateCorrectionOut during startup (2)

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	3	
gdMinislotActionPointOffset [MT]	1	3	
gdStaticSlot [MT]	22	26	
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	83	4 160	8 000
gNumberOfMinislots	0	581	1 130
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gClockDeviationMax	0,0009	0,0015	0,0015
gdCycle [µs]	166	8 320	16 000
gdNIT [MT]	39	41	38
pClusterDriftDamping [µT]	0	2	
pdListenTimeout [µT]	6 652	333 800	641 924
pKeySlotID	2		
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3	5	
pMicroPerCycle [µT]	3 320	166 400	320 000
pOffsetCorrectionOut [µT]	45	130	
pOffsetCorrectionStart [MT]	80	4 150	7 990
pRateCorrectionOut [µT]	6	500	962

The IUT is configured as coldstart node sending startup frames in slot 2.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT simulates a CAS *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) The LT simulates startup frames in slot 1 of cycles 0 to 7. The LT's cycle length is (*pMicroPerCycle* + (*pRateCorrectionOut* – 1)) * *pSamplesPerMicrotick* – 1 σT.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,7 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

6) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,7 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

7) In cycle 4, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

8) In cycle 7, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 24 depicts the *pRateCorrectionOut* during startup (2).

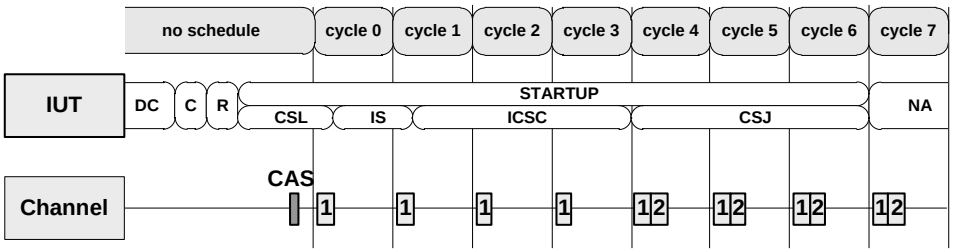


Figure 24 — pRateCorrectionOut during startup (2)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the startup accordingly. The IUT accepts the corresponding odd startup frames of the LT because the LT's cycle length is within the range determined by *pRateCorrectionOut*.

7.3.2.7 pMicroInitialOffset[A]

— Test purpose

Verify correct interpretation of the parameter *pMicroInitialOffset[A]* and correct initialization of the clock synchronisation.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 97, Table 98 and Table 99.

Table 97 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pMicroInitialOffset[A]

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	97		95	97		
gdMinislot [MT]	9			9		
gdTSSTransmitter [gdBit]	11		10	11		
gMacroPerCycle [MT]	2 655			5 000		
gNumberOfStaticSlots	18			85		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	2 655		15 930	5 000		15 000
gdMacrotick [µs]	1		6	1		3
gdNIT [MT]	14			14		
pdAcceptedStartupRange [µT]	281			403		1 108
pClusterDriftDamping [µT]	0			0		
pDecodingCorrection [µT]	52		48	104		
pDelayCompensation[A] [µT]	28	5		56	7	137
pdListenTimeout [µT]	212 528		1 275 166	800 482		2 401 442
pKeySlotID	3			3		
pKeySlotUsedForStartup	false			false		
pKeySlotUsedForSync	false			false		
pMacroInitialOffset[A] [MT]	6		5	6		
pMacroInitialOffset[B] [MT]	6		5	6		5
pMicroInitialOffset[A] [µT]	0	23	187	0	46	239
pMicroInitialOffset[B] [µT]	23		187	46		126
pMicroPerCycle [µT]	106 200		637 200	400 000		1 200 000
pOffsetCorrectionOut [µT]	153		400	226		800
pOffsetCorrectionStart [MT]	2 647			4 990		
pRateCorrectionOut [µT]	64		383	241		721

Table 98 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pMicroInitialOffset[A]

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	80			80		
gdMinislot [MT]	7			7		
gdTSSTransmitter [gdBit]	6			6		
gMacroPerCycle [MT]	2 500			2 500		
gNumberOfStaticSlots	45			45		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	5 000		15 000	5 000		15 000
gdMacrotick [µs]	2		6	2		6
gdNIT [MT]	12			12		
pdAcceptedStartupRange [µT]	283		663	221		411
pClusterDriftDamping [µT]	0			0		
pDecodingCorrection [µT]	64			32		
pDelayCompensation[A] [µT]	16	10		8	5	
pdListenTimeout [µT]	400 242		1 200 722	200 122		600 362
pKeySlotID	3			3		
pKeySlotUsedForStartup	false			false		
pKeySlotUsedForSync	false			false		
pMacroInitialOffset[A] [MT]	4			4		
pMacroInitialOffset[B] [MT]	4			4		
pMicroInitialOffset[A] [µT]	0	6	166	0	3	83
pMicroInitialOffset[B] [µT]	6		166	3		83
pMicroPerCycle [µT]	200 000		600 000	100 000		300 000
pOffsetCorrectionOut [µT]	154		400	117		350
pOffsetCorrectionStart [MT]	2 491			2 491		
pRateCorrectionOut [µT]	121		361	61		181

Table 99 — Modification to basic configuration 3 for protocol operation related configuration data – pMicroInitialOffset[A]

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	72		
gdMinislot [MT]	8	7	
gdTSSTransmitter [gdBit]	4		
gMacroPerCycle [MT]	2 500		
gNumberOfStaticSlots	25		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	5 000		15 000
gdMacrotick [µs]	2		6
gdNIT [MT]	40	11	
pdAcceptedStartupRange [µT]	288	224	334
pClusterDriftDamping [µT]	0		
pDecodingCorrection [µT]	48		
pDelayCompensation[A] [µT]	32	10	
pdListenTimeout [µT]	200 122		600 362
pKeySlotID	3		
pKeySlotUsedForStartup	false		
pKeySlotUsedForSync	false		
pMacroInitialOffset[A] [MT]	5		4
pMacroInitialOffset[B] [MT]	5		4
pMicroInitialOffset[A] [µT]	0	22	62
pMicroInitialOffset[B] [µT]	22		62
pMicroPerCycle [µT]	100 000		300 000
pOffsetCorrectionOut [µT]	150	118	
pOffsetCorrectionStart [MT]	2 492		
pRateCorrectionOut [µT]	61		181

The IUT is an integrating node configured to send a static frame in slot 3 on channel A. In case of SC applicability the IUT shall set *pChannels* to A only.

— **Preamble (setup state)**

1) The UT resets the IUT (vPOC!State = DEFAULT_CONFIG).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

— **Test execution**

1) The LT begins to simulate startup frames in slots 1 and 2 of cycles 0 to 6 on channel A 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

3) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

4) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

5) In cycle 6, it is verified (LT) that the interval between the IUT's frame in slot 3 and the LT's frame in slot 2 is for

   (II) *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT

   (I) and (III) *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* – (*pDelayCompensation[A]* – 10 / *pSamplesPerMicrotick*) + ξ + ξ_{IUT} µT

Figure 25 depicts the *pMicroInitialOffset[A]*.

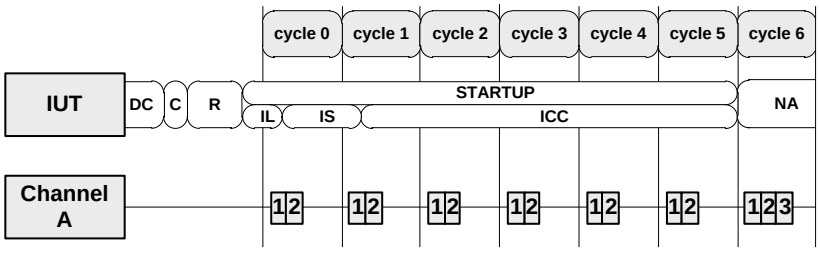


Figure 25 — pMicroInitialOffset[A]

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs appropriate offset and rate correction and sends its frame with the correct offset from the LT's frame.

7.3.2.8 pMicroInitialOffset[B]

— **Test purpose**

Verify correct interpretation of the parameter *pMicroInitialOffset[B]* and correct initialization of the clock synchronisation.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 100, Table 101 and Table 102.

Table 100 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pMicroInitialOffset[B]

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	97		95	97		
gdMinislot [MT]	9			9		
gdTSSTransmitter [gdBit]	11		10	11		
gMacroPerCycle [MT]	2 655			5 000		
gNumberOfStaticSlots	18			85		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	2 655		15 930	5 000		15 000
gdMacrotick [µs]	1		6	1		3
gdNIT [MT]	14			14		
pdAcceptedStartupRange [µT]	281			403		1 108
pClusterDriftDamping [µT]	0			0		
pDecodingCorrection [µT]	52		48	104		
pDelayCompensation[B] [µT]	28	5		56	10	137
pdListenTimeout [µT]	212 528		1 275 166	800 482		2 401 442
pKeySlotID	3			3		
pKeySlotUsedForStartup	false			false		
pKeySlotUsedForSync	false			false		
pMacroInitialOffset[A] [MT]	6		5	6		5
pMacroInitialOffset[B] [MT]	6		5	6		
pMicroInitialOffset[A] [µT]	23		187	46		126
pMicroInitialOffset[B] [µT]	0	23	187	0	46	239
pMicroPerCycle [µT]	106 200		637 200	400 000		1 200 000
pOffsetCorrectionOut [µT]	153		400	226		800
pOffsetCorrectionStart [MT]	2 647			4 990		
pRateCorrectionOut [µT]	64		383	241		721

Table 101 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pMicroInitialOffset[B]

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	80			80		
gdMinislot [MT]	7			7		
gdTSSTransmitter [gdBit]	6			6		
gMacroPerCycle [MT]	2 500			2 500		
gNumberOfStaticSlots	45			45		
gPayloadLengthStatic [two-byte word]	0			0		
gdCycle [µs]	5 000		15 000	5 000		15 000
gdMacrotick [µs]	2		6	2		6
gdNIT [MT]	12			12		
pdAcceptedStartupRange [µT]	283		663	221		411
pClusterDriftDamping [µT]	0			0		
pDecodingCorrection [µT]	64			32		
pDelayCompensation[B] [µT]	16	10		8	5	
pdListenTimeout [µT]	400 242		1 200 722	200 122		600 362
pKeySlotID	3			3		
pKeySlotUsedForStartup	false			false		
pKeySlotUsedForSync	false			false		
pMacroInitialOffset[A] [MT]	4			4		
pMacroInitialOffset[B] [MT]	4			4		
pMicroInitialOffset[A] [µT]	6		166	3		83
pMicroInitialOffset[B] [µT]	0	6	166	0	3	83
pMicroPerCycle [µT]	200 000		600 000	100 000		300 000
pOffsetCorrectionOut [µT]	154		400	117		350
pOffsetCorrectionStart [MT]	2 491			2 491		
pRateCorrectionOut [µT]	121		361	61		181

Table 102 — Modification to basic configuration 3 for protocol operation related configuration data – pMicroInitialOffset[B]

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	72		
gdMinislot [MT]	8	7	
gdTSSTransmitter [gdBit]	4		
gMacroPerCycle [MT]	2 500		
gNumberOfStaticSlots	25		
gPayloadLengthStatic [two-byte word]	0		
gdCycle [µs]	5 000		15 000
gdMacrotick [µs]	2		6
gdNIT [MT]	40	11	
pdAcceptedStartupRange [µT]	288	224	334
pClusterDriftDamping [µT]	0		
pDecodingCorrection [µT]	48		
pDelayCompensation[B] [µT]	32	10	
pdListenTimeout [µT]	200 122		600 362
pKeySlotID	3		
pKeySlotUsedForStartup	false		
pKeySlotUsedForSync	false		
pMacroInitialOffset[A] [MT]	5		4
pMacroInitialOffset[B] [MT]	5		4
pMicroInitialOffset[A] [µT]	22		62
pMicroInitialOffset[B] [µT]	0	22	62
pMicroPerCycle [µT]	100 000		300 000
pOffsetCorrectionOut [µT]	150	118	
pOffsetCorrectionStart [MT]	2 492		
pRateCorrectionOut [µT]	61		181

The IUT is an integrating node and is configured to send a static frame in slot 3 on channel B. In case of SC applicability the IUT shall set *pChannels* to B only.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = READY).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = STARTUP and *vPOC!StartupState* = INTEGRATION_LISTEN) 2 500 µT after the CHI command RUN.

— **Test execution**

1) The LT begins to simulate startup frames in slots 1 and 2 of cycles 0 to 6 on channel B 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

3) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

4) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

5) In cycle 6, it is verified (LT) that the interval between the IUT's frame in slot 3 and the LT's frame in slot 2 is for

   (II) *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT

   (I) and (III) *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* – (*pDelayCompensation[B]* – 10 / *pSamplesPerMicrotick*) + ξ + ξ_{IUT} µT

Figure 26 depicts the *pMicroInitialOffset[B]*.

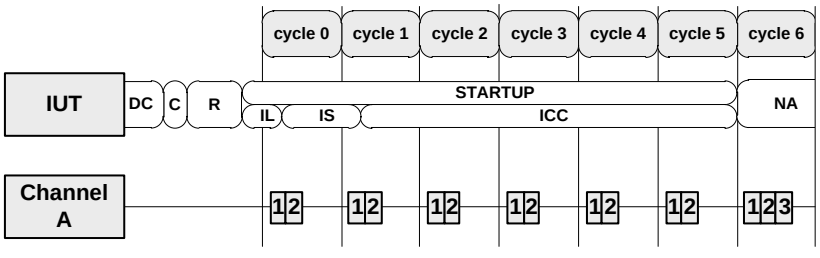


Figure 26 — pMicroInitialOffset[B]

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs appropriate offset and rate correction and sends its frame with the correct offset from the LT's frame.

7.3.2.9 pdAcceptedStartupRange

— **Test purpose**

Verify the correct handling of the parameter *pdAcceptedStartupRange*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 103, Table 104 and Table 105. The test has to be performed repeatedly for each of the Δt's specified below.

Table 103 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pdAcceptedStartupRange

Parameter	Modification to Basic Configurations			
	1a		1b	
	I	II	I	II
gdActionPointOffset [MT]	4	63	4	63
gdStaticSlot [MT]	35	160	35	160
gdSymbolWindow [MT]	40	0	40	0
gMacroPerCycle [MT]	5 000	16 000	5 000	16 000
gNumberOfStaticSlots	85		85	
gPayloadLengthStatic [two-byte word]	0		0	
gClockDeviationMax	0,0003	0,0015	0,0003	0,0015
gdCycle [µs]	5 000	16 000	5 000	16 000
gdNIT [MT]	14	370	14	370
pdAcceptedStartupRange [µT]	79	1 608	79	1 329
pClusterDriftDamping [µT]	0	2	0	2
pdListenTimeout [µT]	400 242	1 283 846	800 482	2 567 692
pKeySlotID	3		3	
pKeySlotUsedForStartup	false		false	
pKeySlotUsedForSync	false		false	
pMacroInitialOffset[A,B] [MT]	6	65	6	65
pMicroPerCycle [µT]	200 000	640 000	4 00000	1 280 000
pOffsetCorrectionOut [µT]	153	2 523	226	2 523
pOffsetCorrectionStart [MT]	4 990	15 800	4 990	15 800
pRateCorrectionOut [µT]	121	1 923	241	3 846

Table 104 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pdAcceptedStartupRange

Parameter	Modification to Basic Configurations			
	2a		2b	
	I	II	I	II
gdActionPointOffset [MT]	3	63	3	63
gdStaticSlot [MT]	33	150	33	150
gdSymbolWindow [MT]	23	0	23	0
gMacroPerCycle [MT]	2 500	8 000	2 500	8 000
gNumberOfStaticSlots	45		45	
gPayloadLengthStatic [two-byte word]	0		0	
gClockDeviationMax	0,0003	0,0015	0,0003	0,0015
gdCycle [µs]	5 000	16 000	5 000	16 000
gdNIT [MT]	12	210	12	210
pdAcceptedStartupRange [µT]	80	814	79	1 093
pClusterDriftDamping [µT]	0		0	2
pdListenTimeout [µT]	400 242	1 283 846	200 122	641 924
pKeySlotID	3		3	
pKeySlotUsedForStartup	false		false	
pKeySlotUsedForSync	false		false	
pMacroInitialOffset[A,B] [MT]	4	64	4	64
pMicroPerCycle [µT]	200 000	640 000	100 000	320 000
pOffsetCorrectionOut [µT]	163	2 523	117	962
pOffsetCorrectionStart [MT]	2 491	7 950	2 491	7 950
pRateCorrectionOut [µT]	121	1 923	61	962

Table 105 — Modification to basic configuration 3 for protocol operation related configuration data – pdAcceptedStartupRange

Parameter	Modification to Basic Configuration 3	
	I	II
gdActionPointOffset [MT]	3	63
gdStaticSlot [MT]	58	150
gdSymbolWindow [MT]	24	0
gMacroPerCycle [MT]	2 500	8 000
gNumberOfStaticSlots	25	45
gPayloadLengthStatic [two-byte word]	0	
gClockDeviationMax	0,0003	0,0015
gdCycle [µs]	5 000	16 000
gdNIT [MT]	11	175
pdAcceptedStartupRange [µT]	79	557
pClusterDriftDamping [µT]	0	2
pdListenTimeout [µT]	200 122	641 924
pKeySlotID	3	
pKeySlotUsedForStartup	false	
pKeySlotUsedForSync	false	
pMacroInitialOffset[A,B] [MT]	5	65
pMicroPerCycle [µT]	100 000	320 000
pOffsetCorrectionOut [µT]	118	962
pOffsetCorrectionStart [MT]	2 492	7 950
pRateCorrectionOut [µT]	61	962

Table 106 defines the deviation for all variants.

Table 106 — Deviation for all variants

Variant	Deviation
	Δt [µT]
(a)	*pdAcceptedStartupRange* – 2
(b)	– (*pdAcceptedStartupRange* – 2)
(c)	*pdAcceptedStartupRange* + 1 + 2
(d)	– (*pdAcceptedStartupRange* + 1 + 2)

The IUT is an integrating node and is configured to send a static frame in slot 3.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

— **Test execution**

1) The LT begins to simulate startup frames in slots 1 and 2 of cycles 0 and 1 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT shifts its startup frames by Δt in cycle 2 and by $\Delta t / 2$ in cycle 3. After cycle 3 the LT does not shift its startup frames any more.

3) It is verified (UT) that the IUT is

   (a, b) in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) in cycle 3, 500 µT after cycle start end before cycle end,
   and

   (c, d) in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) in cycle 3, 500 µT after cycle start and before cycle end, and in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) in the NIT of cycle 4.

4) (a, b) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 27 depicts the *pdAcceptedStartupRange* – variants (a) and (b).

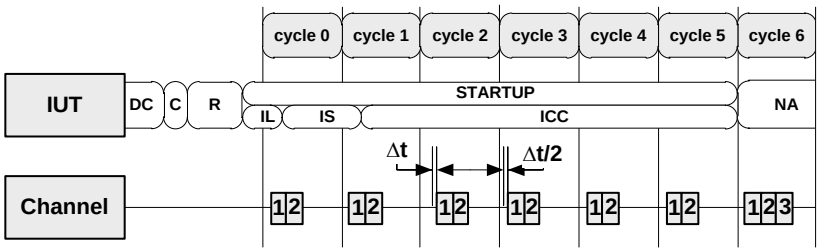


Figure 27 — pdAcceptedStartupRange – variants (a) and (b)

Figure 28 depicts the *pdAcceptedStartupRange* – variants (c) and (d).

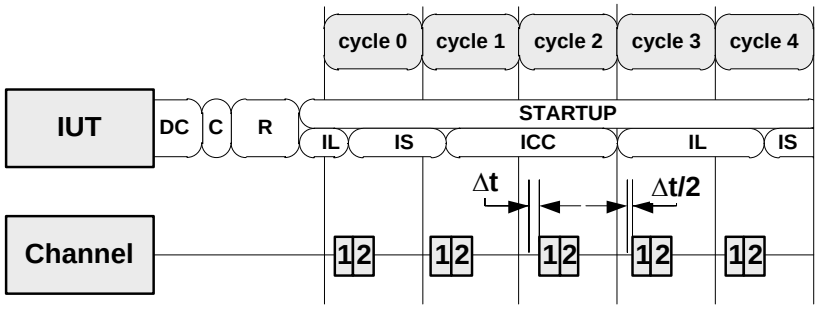


Figure 28 — pdAcceptedStartupRange – variants (c) and (d)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT shows the appropriate startup behaviour according to the value of the parameter *pdAcceptedStartupRange*.

7.3.2.10 pChannels

— Test purpose

Verify correct behaviour of the IUT according to the parameter *pChannels*. The IUT shall transmit frames and symbols only on the channel(s) enabled via *pChannels*.

— Applicability

SC, DC.

Modification III is only applicable for dual channel test execution.

— Configuration

All basic configurations using the modifications as listed in Table 107..

Table 107 — Modification to basic configurations for protocol operation related configuration data – pChannels

Parameter	Modification		
	I	II	III
pChannels	A	B	A & B

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to send additional frames in slot 3 and in slot ndyn on all available channel(s).

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In the static segment of cycle 7, the UT requests manual (one-shot) MTS transmission on all available channel(s).

2) In cycle 7, it is verified (LT) that the IUT transmits its frames and the MTS only on channel(s) *pChannels*. The frame contents of the frames in slots 3 and n_{dyn} are: The frame ID is set to 3 (for the frame in slot 3) and n_{dyn} (for the frame in slot n_{dyn}), the cycle count to 7, the payload length to *gPayloadLengthStatic* (for the frame in slot 3) and 1 (for the frame in slot n_{dyn}), the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0' and the payload is set to the default payload. The frame CRC is correct and complies to the channel on which the respective frame is transmitted by the IUT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames and the MTS on channel(s) *pChannels* correctly.

7.3.2.11 pWakeupChannel

— **Test purpose**

Verify channel assignment according to *pWakeupChannel*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 108.

Table 108 — Modification to basic configurations for protocol operation related configuration data – pWakeupChannel

Parameter	Modification	
	I	II
pWakeupChannel	A	B

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT issues the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) It is verified (LT) that the IUT transmits a WUP as configured on channel *pWakeupChannel* *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT after the CHI command WAKEUP.

4) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *TRANSMITTED* 2 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT after the CHI command WAKEUP.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a WUP as configured (*pWakeupPattern* wakeup symbols each consisting of a low phase of length *gdWakeupTxActive* followed by an idle phase of length *gdWakeupTxIdle*) on channel *pWakeupChannel*.

7.3.2.12 pKeySlotOnlyEnabled (1)

— **Test purpose**

Verify correct behaviour of the IUT according to the parameter *pKeySlotOnlyEnabled* = true. The IUT shall transmit frames only in the slot *pKeySlotID* as long as the key slot only mode is enabled. Verify de-activated MTS transmission while the key slot only mode is enabled (*pKeySlotOnlyEnabled* = true).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 109.

Table 109 — Modification to basic configurations for protocol operation related configuration data – pKeySlotOnlyEnabled (1)

Parameter	Modification
pKeySlotOnlyEnabled	true

The IUT is configured as coldstart node sending startup frames in slot 1 and transmitting static frames in slot 3.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

— **Test execution**

1) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

2) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

3) The UT requests the IUT to send an MTS in cycle 6 on the available channel(s).

4) It is verified (UT) that *pKeySlotOnlyEnabled* = true and *vPOC!SlotMode* = *KEYSLOT*.

5) The UT initiates the startup procedure with the CHI command RUN (*vPOC!State* = *STARTUP*).

6) The IUT is the leading coldstart node and transmits startup frames in slot 1.

7) The LT simulates the following coldstart node. The LT simulates startup frames in slot 2 of cycle 4 to 10.

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

9) In the symbol window of cycle 6, it is verified (LT), that the IUT does not transmit an MTS.

10) It is verified (LT) that the IUT does not transmit a static frame in slot 3 of cycles 0 to 7.

11) In slot 1 of cycle 7, the UT disables key slot only mode via CHI command ALL_SLOTS.

12) It is verified (UT) that *vPOC!SlotMode* = *ALL_PENDING* 2 500 µT after the CHI command ALL_SLOTS within cycle 7.

13) In cycle 8, 500 µT after cycle start and before cycle end, it is verified (UT) that *vPOC!SlotMode* = *ALL*.

14) In cycle 8, it is verified (LT) that the IUT transmits its startup frame in slot 1 and the static frame in slot 3 on the available channel(s) correctly, i.e. the frame ID is set to 1, the cycle count to 8, the payload length to *gPayloadLengthStatic*, the payload to the default payload, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1' and the payload preamble indicator is set to '0' in slot 1 and the frame ID is set to 3, the cycle count to 8, the payload length to *gPayloadLengthStatic*, the payload to the default payload, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0' in slot 3.

15) In cycle 9, the UT tries to change the value of *pKeySlotOnlyEnabled* by setting *pKeySlotOnlyEnabled* = false.

16) In cycle 10, 500 µT after cycle start and before cycle end, it is verified (UT) that *pKeySlotOnlyEnabled* = true. It is also verified (LT) that the IUT transmits its startup frame in slot 1 and the static frame in slot 3 on the available channel(s) correctly

Figure 29 depicts the *pKeySlotOnlyEnabled (1)*.

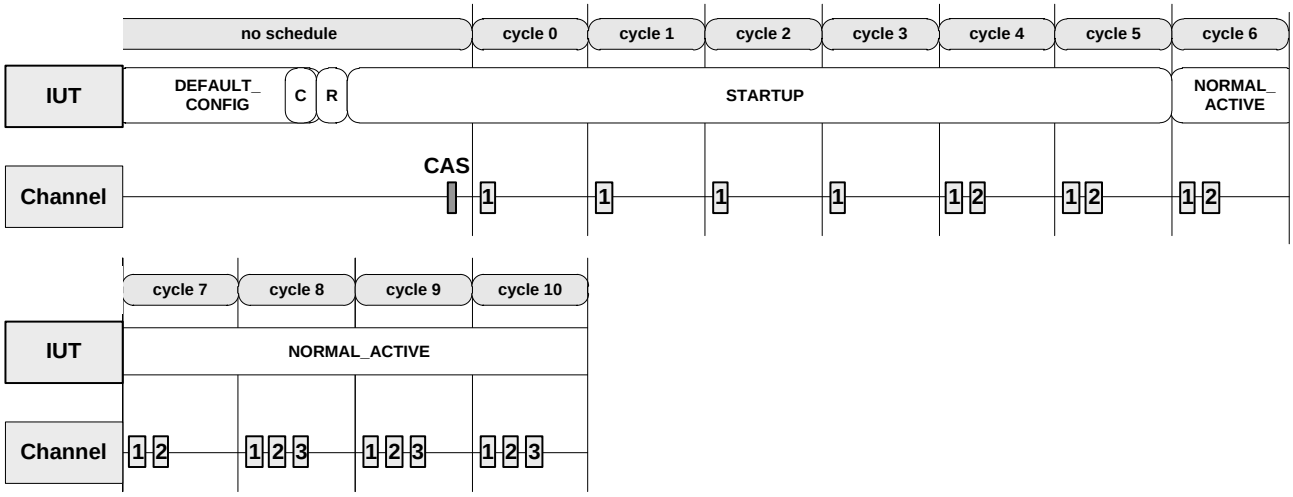


Figure 29 — pKeySlotOnlyEnabled (1)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The slot mode can be altered as specified. The IUT transmits frames only in the slot *pKeySlotID* as long as the key slot only mode is enabled. The IUT does not transmit an MTS while the key slot only mode is enabled.

7.3.2.13 pKeySlotOnlyEnabled (2)

— Test purpose

Verify correct behaviour of the IUT according to *pKeySlotOnlyEnabled* = false. The IUT shall transmit frames in all slots assigned to the IUT after successful startup.

— Applicability

SC, DC.

— Configuration

All basic configurations.

The IUT is configured as coldstart node sending startup frames in slot 1 and static frames in slot 3.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

— Test execution

1) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

2) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

3) After 2 500 µT it is verified (UT) that *pKeySlotOnlyEnabled* = false and *vPOC!SlotMode* = *ALL*.

4) The UT initiates the startup procedure with the CHI command RUN (*vPOC!State* = *STARTUP*).

5) The IUT is the leading coldstart node and transmits startup frames in slot 1.

6) The LT simulates the following coldstart node. The LT simulates startup frames in slot 2 of cycles 4 to 6.

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

8) In cycle 6, it is verified (LT) that the IUT transmits its static frame in slot 3 of cycle 6 on the available channel(s) correctly, i.e. the frame ID is set to 3, the cycle count to 6, the payload length to *gPayloadLengthStatic*, the payload to the default payload, the null frame indicator is set to '1', the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0'.

Figure 30 depicts the *pKeySlotOnlyEnabled (2)*.

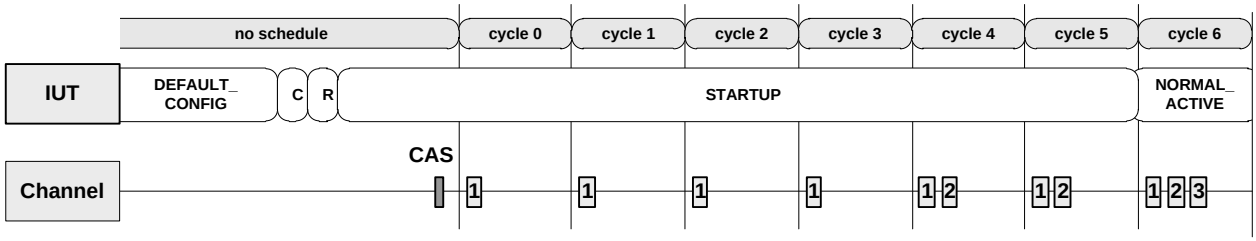


Figure 30 — pKeySlotOnlyEnabled (2)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits frames in all slots assigned to the IUT after successful startup.

7.3.2.14 pKeySlotID

— Test purpose

Verify correct behaviour of the IUT according to the parameter *pKeySlotID*. The IUT shall transmit its startup frame in slot *pKeySlotID*.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 110, Table 111 and Table 112.

Table 110 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pKeySlotID

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1			1		
gdStaticSlot [MT]	15			15		
gdSymbolWindow [MT]	40			40		
gdSymbolWindowActionPointOffset [MT]	1			1		
gMacroPerCycle [MT]	15 400			15 400		
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	1 023			1 023		
gPayloadLengthStatic [two-byte word]	1			1		
gdCycle [µs]	15 400			15 400		
gdNIT [MT]	15			15		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	1 232 740			2 465 480		
pKeySlotID	1	512	1 023	1	512	1 023
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	3			3		
pMicroPerCycle [µT]	616 000			1 232 000		
pOffsetCorrectionOut [µT]	45			45		
pOffsetCorrectionStart [MT]	15 397			15 397		
pRateCorrectionOut [µT]	370			740		

Table 111 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pKeySlotID

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1			1		
gdStaticSlot [MT]	14			15		
gdSymbolWindow [MT]	0			23		
gdSymbolWindowActionPointOffset [MT]	1			1		
gMacroPerCycle [MT]	8 000			8 000		
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	570			530		
gPayloadLengthStatic [two-byte word]	1			1		
gdCycle [µs]	16 000			16 000		
gdNIT [MT]	20			27		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	1 280 770			640 386		
pKeySlotID	1	256	570	1	256	530
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2			2		
pMicroPerCycle [µT]	640 000			320 000		
pOffsetCorrectionOut [µT]	45			45		
pOffsetCorrectionStart [MT]	7 998			7 997		
pRateCorrectionOut [µT]	385			193		

Table 112 — Modification to basic configuration 3 for protocol operation related configuration data – pKeySlotID

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1		
gdStaticSlot [MT]	26		
gdSymbolWindow [MT]	24		
gdSymbolWindowActionPointOffset [MT]	1		
gMacroPerCycle [MT]	8 000		
gNumberOfMinislots	0		
gNumberOfStaticSlots	306		
gPayloadLengthStatic [two-byte word]	1		
gdCycle [µs]	16 000		
gdNIT [MT]	20		
pClusterDriftDamping [µT]	0		
pdListenTimeout [µT]	640 386		
pKeySlotID	1	127	306
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3		
pMicroPerCycle [µT]	320 000		
pOffsetCorrectionOut [µT]	45		
pOffsetCorrectionStart [MT]	7 995		
pRateCorrectionOut [µT]	193		

— **Preamble (setup state)**

Preamble II.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) In cycle 7, it is verified (LT) that the IUT transmits its frame in slot *pKeySlotID* on the available channel(s) correctly, i.e. the frame ID is set to *pKeySlotID*, the cycle count to 7, the payload length to 1, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1', the payload preamble indicator is set to '0' and the payload is set to the default payload.

2) In cycle 7, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames list(s) for the odd cycle on the available channel(s) contain(s) the IUT's startup frame ID.

3) In cycle 8, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames list(s) for the even cycle on the available channel(s) contain(s) the IUT's startup frame ID.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its startup frame in the configured slot *pKeySlotID*.

7.3.2.15 pKeySlotUsedForSync

— Test purpose

Verify correct behaviour of the IUT according to *pKeySlotUsedForSync*. The IUT shall transmit its frame in slot *pKeySlotID* with the sync frame indicator set to '1' or '0' depending on whether the parameter *pKeySlotUsedForSync* is set to true or false.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 113.

Table 113 — Modification to basic configurations for protocol operation related configuration data – pKeySlotUsedForSync

Parameter	Modification		
	I	II	III
pKeySlotUsedForStartup	true	false	false
pKeySlotUsedForSync	true	true	false
pKeySlotID	1		

— Preamble (setup state)

Preamble II for Modification I.

Preamble III for Modification II and Modification III.

Step 4 of Preamble III is replaced by the UT, which configures the IUT to transmit a sync frame in slot 1.

— Test execution

It is verified (LT) that the IUT transmits its frame in slot *pKeySlotID* with

(I) the sync frame indicator set to '1' in cycle 7,

(II) the sync frame indicator set to '1' in cycle 9 and

(III) the sync frame indicator set to '0' in cycle 9.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frame in slot *pKeySlotID* with the sync frame indicator set to '1' if *pKeySlotUsedForSync* = true and to '0' if *pKeySlotUsedForSync* = false.

7.3.2.16 pKeySlotUsedForStartup

— Test purpose

Verify correct behaviour of the IUT according to *pKeySlotUsedForStartup*. The IUT shall transmit its frame in slot *pKeySlotID* with the startup frame indicator set to '1' or '0' depending on whether the parameter *pKeySlotUsedForStartup* is set to true or false.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 114.

Table 114 — Modification to basic configurations for protocol operation related configuration data – pKeySlotUsedForStartup

Parameter	Modification	
	I	II
pKeySlotUsedForStartup	true	false
pKeySlotUsedForSync	true	
pKeySlotID	2	1

— Preamble (setup state)

Preamble I for Modification I.

Preamble III for Modification II.

Step 4 of Preamble III is replaced by the UT, which configures the IUT to transmit a sync frame in slot 1.

— Test execution

It is verified (LT) that the IUT transmits its frame in slot *pKeySlotID* with

(I) the startup frame indicator set to '1' in cycle 8 and

(II) the startup frame indicator set to '0' in cycle 9.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frame in slot *pKeySlotID* with the startup frame indicator set to '1' if *pKeySlotUsedForStartup* = true and set to '0' if *pKeySlotUsedForStartup* = false.

7.3.2.17 pDecodingCorrection

— **Test purpose**

Verify correct behaviour of the IUT according to the parameter *pDecodingCorrection*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 115, Table 116 and Table 117.

Table 115 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pDecodingCorrection

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	77	97	105	77	97	105
gdTSSTransmitter [gdBit]	1[a]	11	15	1[a]	11	15
pDecodingCorrection [µT]	12	52	68	24	104	136
pMacroInitialOffset[A,B] [MT]	5	6		5	6	
pMicroInitialOffset[A,B] [µT]	23		7	46		14
[a] gdTSSTransmitter = 1 [gdBit] which is an intentional protocol constraints violation.						

Table 116 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pDecodingCorrection

Parameter	Modification to Basic Configuration					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	70	80	84	70	80	84
gdTSSTransmitter [gdBit]	1[a]	6	8	1[a]	6	8
pDecodingCorrection [µT]	24	64	80	12	32	40
pMacroInitialOffset[A,B] [MT]	4		5	4		5
pMicroInitialOffset[A,B] [µT]	46	6	70	23	3	35
[a] gdTSSTransmitter = 1 [gdBit] which is an intentional protocol constraints violation.						

Table 117 — Modification to basic configuration 3 for protocol operation related configuration data – pDecodingCorrection

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	66	72	74
gdTSSTransmitter [gdBit]	1[a]	4	5
pDecodingCorrection [µT]	24	48	56
pMacroInitialOffset[A,B] [MT]	4	5	
pMicroInitialOffset[A,B] [µT]	6	22	14
[a] gdTSSTransmitter = 1 [gdBit] which is an intentional protocol constraints violation.			

— **Preamble (setup state)**

Preamble III.

— **Test execution**

In cycle 10, it is verified (LT) that the interval between the LT's frame in slot 2 and the IUT's frame in slot 1 is *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT on the available channel(s).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The interval between the LT's frame in slot 2 and the IUT's frame in slot 1 is *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT on the available channel(s).

7.3.2.18 pDelayCompensation

— **Test purpose**

Verify correct behaviour of the IUT according to the parameter *pDelayCompensation[Ch]*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 118, Table 119 and Table 120.

Table 118 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pDelayCompensation

Parameter	Modification to Basic Configurations			
	1a		1b	
	I	II	I	II
gdActionPointOffset [MT]	7		7	
gdMinislot [MT]	18		18	
gdMinislotActionPointOffset [MT]	7		7	
gdStaticSlot [MT]	44		44	
gdSymbolWindow [MT]	0		0	
gNumberOfMinislots	67		67	
gdNIT [MT]	54		54	
pdAcceptedStartupRange [µT]	669		1 179	
pDelayCompensation[A,B] [µT]	5	105	10	211
pLatestTx [Minislot]	0		0	
pMacroInitialOffset[A,B] [MT]	9	11	9	11
pMicroInitialOffset[A,B] [µT]	23	3	46	5
pOffsetCorrectionOut [µT]	388		700	
pOffsetCorrectionStart [MT]	4 980		4 980	

Table 119 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pDelayCompensation

Parameter	Modification to Basic Configurations			
	2a		2b	
	I	II	I	II
gdActionPointOffset [MT]	4		5	
gdMinislot [MT]	10		12	
gdMinislotActionPointOffset [MT]	4		5	
gdStaticSlot [MT]	35		37	
gdSymbolWindow [MT]	0		0	
gNumberOfMinislots	87		65	
gdNIT [MT]	55		55	
pdAcceptedStartupRange [µT]	679		419	
pDelayCompensation[A,B] [µT]	10	111	5	55
pLatestTx [Minislot]	0		0	
pMacroInitialOffset[A,B] [MT]	5	7	6	8
pMicroInitialOffset[A,B] [µT]	6	65	3	33
pOffsetCorrectionOut [µT]	395		255	
pOffsetCorrectionStart [MT]	2 491		2 485	

Table 120 — Modification to basic configuration 3 for protocol operation related configuration data – pDelayCompensation

Parameter	Modification to Basic Configuration 3	
	I	II
gdActionPointOffset [MT]	5	
gdMinislot [MT]	12	
gdMinislotActionPointOffset [MT]	5	
gdStaticSlot [MT]	63	
gdSymbolWindow [MT]	0	
gNumberOfMinislots	75	
gdNIT [MT]	25	
pdAcceptedStartupRange [µT]	429	
pDelayCompensation[A,B] [µT]	10	61
pLatestTx [Minislot]	0	
pMacroInitialOffset[A,B] [MT]	7	8
pMicroInitialOffset[A,B] [µT]	22	11
pOffsetCorrectionOut [µT]	260	
pOffsetCorrectionStart [MT]	2 486	

— **Preamble (setup state)**

Preamble III.

— **Test execution**

In cycle 9 and 10, it is verified (LT) that the interval between the LT's frame in slot 2 and the IUT's frame in slot 1 is *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + *pDelayCompensation[Ch]* - 10 / *pSamplesPerMicrotick* + ξ + ξ_{IUT} µT on the available channel(s).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT compensate the shifts of the LT's frames according to *pDelayCompensation[Ch]*.

7.3.2.19 pKeySlotOnlyEnabled (3)

— Test purpose

Verify correct behaviour of the IUT according to *pKeySlotOnlyEnabled* = true. The IUT is an integrating node and does not send startup frames nor sync frames.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 121.

Table 121 — Modification to basic configurations for protocol operation related configuration data – pKeySlotOnlyEnabled (3)

Parameter	Modification		
	I	II	III
pKeySlotID	0	3	n_{dyn}[a]
pKeySlotUsedForStartup	false		
pKeySlotUsedForSync	false		
pKeySlotOnlyEnabled	true		
[a] pKeySlotID = ndyn which is an intentional protocol constraints violation.			

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit frames in slots 1 and n_{dyn} + 1.

— Preamble (setup state)

Preamble III.

Step 12 of Preamble III is replaced by the following statement: "In cycle 8, it is checked (LT) that the IUT does not transmit a static frame in slot 1."

— Test execution

1) In cycle 9, it is verified (UT) that *pKeySlotOnlyEnabled* = true and *vPOC!SlotMode* = *KEYSLOT*.

2) In cycle 9, it is verified (LT) that the IUT

   (I, III) does not transmit any frame and

   (II) solely transmits its null frame in slot 3 correctly, i.e. the frame ID is set to 3, the cycle count to 9, the null frame indicator is set to '0', all payload bytes are 0x00, the sync frame indicator, the startup frame indicator and the payload preamble indicator are set to '0'.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT shall indicate *pKeySlotOnlyEnabled* = true and *vPOC!SlotMode* = *KEYSLOT* and transmit a single frame in the static slot *pKeySlotID* (Modification II), and shall not transmit frames if *pKeySlotID* is 0 or a SlotId of the dynamic segment (Modifications I and III).

7.3.2.20 pdListenTimeout and startup frame reception

— **Test purpose**

Verify the correct interpretation of the *pdListenTimeout* parameter and the handling of a received startup frame while the IUT is in *POC:coldstart listen* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 122, Table 123 and Table 124.

Table 122 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pdListenTimeout and startup frame reception

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	4	3		1	2	
gdMinislotActionPointOffset [MT]	4	3		1	2	
gdStaticSlot [MT]	19	17	17	13	15	15
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	84	4 673	16 000	46	4 655	16 000
gNumberOfMinislots	0	512	1 770	0	512	1 772
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003		0,0015	0,0003		0,0015
gdCycle [µs]	84	4 673	16 000	46	4 655	16 000
gdNIT [MT]	46	31	36	20	17	22
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	6 726	374 066	1 283 846	7 366	745 248	2 567 692
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	6	5		3	4	
pMicroPerCycle [µT]	3 360	186 920	640 000	3 680	372 400	1 280 000
pOffsetCorrectionOut [µT]	153			45	226	
pOffsetCorrectionStart [MT]	76	4 665	15 992	45	4 650	15 990
pRateCorrectionOut [µT]	3	113	1 923	3	224	3 846

Table 123 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pdListenTimeout and startup frame reception

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	2		1	2	
gdMinislotActionPointOffset [MT]	1	2		1	2	
gdStaticSlot [MT]	12	14		13	15	
gdSymbolWindow [MT]	0			0		
gMacroPerCycle [MT]	44	3 630	8 000	65	3 649	8 000
gNumberOfMinislots	0	512	1 135	0	512	1 130
gNumberOfStaticSlots	2			2		
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0003		0,0015	0,0004	0,0003	0,0015
gdCycle [µs]	88	7 260	16 000	130	7 298	16 000
gdNIT [MT]	20	18	27	39	35	60
pdListenTimeout [µT]	7 046	581 150	1 283 846	5 206	292 096	641 924
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2	3		2	3	
pMicroPerCycle [µT]	3 520	290 400	640 000	2 600	145 960	320 000
pOffsetCorrectionOut [µT]	45	154		45	100	
pOffsetCorrectionStart [MT]	40	3 625	7 995	60	3 643	7 994
pRateCorrectionOut [µT]	3	175	1 923	3	88	962

Table 124 — Modification to basic configuration 3 for protocol operation related configuration data – pdListenTimeout and startup frame reception

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	2	
gdMinislotActionPointOffset [MT]	1	2	
gdStaticSlot [MT]	22	24	24
gdSymbolWindow [MT]	0		
gMacroPerCycle [MT]	83	3 667	8 000
gNumberOfMinislots	0	512	1 130
gNumberOfStaticSlots	2		
gPayloadLengthStatic [two-byte word]	0		
gClockDeviationMax	0,0009	0,0003	0,0015
gdCycle [µs]	166	7 334	16 000
gdNIT [MT]	39	35	42
pClusterDriftDamping [µT]	0		
pdListenTimeout [µT]	6 652	293 538	641 924
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3	4	
pMicroPerCycle [µT]	3 320	146 680	320 000
pOffsetCorrectionOut [µT]	45	100	
pOffsetCorrectionStart [MT]	80	3 661	7 993
pRateCorrectionOut [µT]	6	89	962

The IUT is configured as coldstart node sending startup frames in slot 1.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— **Test execution**

1) The LT generates permanent noise (low phase) on the available channel(s) before the UT initiates the IUT's startup procedure.

2) The UT initiates the startup procedure with the CHI command RUN.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

4) The LT stops noise generation at t0 (earliest 2 500 µT and latest *pdListenTimeout* / 2 – 2 500 µT after the CHI command RUN) on the available channel(s).

5) The LT starts to simulate a startup frame with frame ID set to 2 (null frame indicator, payload preamble indicator and reserved bit are set to '0', sync frame indicator and startup frame indicator are set to '1', the payload length and the cycle count are 0) at t1 = t0 +

   (a) *cChannelIdleDelimiter* * *gdBit* / *pdMicrotick* µT,

   (b) *pdListenTimeout* / 2 µT and

   (c) *pdListenTimeout* – 1 + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* µT.

6) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) at t1 + *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + 500 µT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) after startup frame reception during coldstart listen.

7.3.2.21 pOffsetCorrectionOut

— **Test purpose**

Verify the correct handling of the parameter *pOffsetCorrectionOut*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 125, Table 126 and Table 127.

Table 125 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pOffsetCorrectionOut

Parameter	Modification to Basic Configurations			
	1a		1b	
	I	II	I	II
gdActionPointOffset [MT]	4	63	4	63
gdStaticSlot [MT]	150		150	
gdSymbolWindow [MT]	0		0	
gMacroPerCycle [MT]	680	720	680	750
gNumberOfMinislots	0		0	
gNumberOfStaticSlots	4		4	
gPayloadLengthStatic [two-byte word]	0		0	
gdCycle [µs]	680	4 320	680	2 250
gdMacrotick [µs]	1	6	1	3
gdNIT [MT]	80	120	80	150
gExternOffsetCorrection [µs]	0,35	0,35	0,175	0,175
pAllowHaltDueToClock	false		false	
pdAcceptedStartupRange [µT]	281	294	403	416
pClusterDriftDamping [µT]	0	2	0	2
pdListenTimeout [µT]	54 434	345 808	108 866	360 218
pLatestTx [Minislot]	0		0	
pMacroInitialOffset[A,B] [MT]	6	64	6	64
pMicroInitialOffset[A,B] [µT]	23	183	46	126
pMicroPerCycle [µT]	27 200	172 800	54 400	180 000
pOffsetCorrectionOut [µT]	15	15 077	15	15 008
pOffsetCorrectionStart [MT]	670	640	620	670
pRateCorrectionOut [µT]	17	104	33	109
pExternOffsetCorrection [µT]	14			

Table 126 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pOffsetCorrectionOut

Parameter	Modification to Basic Configurations			
	2a		2b	
	I	II	I	II
gdActionPointOffset [MT]	3	63	3	63
gdStaticSlot [MT]	150		150	
gdSymbolWindow [MT]	0		0	
gMacroPerCycle [MT]	680	720	680	720
gNumberOfMinislots	0		0	
gNumberOfStaticSlots	4		4	
gPayloadLengthStatic [two-byte word]	0		0	
gdCycle [µs]	1 360	4 320	1 360	4 320
gdMacrotick [µs]	2	6	2	6
gdNIT [MT]	80	120	80	120
gExternOffsetCorrection [µs]	0,35	0,35	0,35	0,35
pAllowHaltDueToClock	false		false	
pdAcceptedStartupRange [µT]	283	297	221	228
pClusterDriftDamping [µT]	0	2	0	2
pdListenTimeout [µT]	108 866	345 808	54 434	172 904
pLatestTx [Minislot]	0		0	
pMacroInitialOffset[A,B] [MT]	4	64	4	64
pMicroInitialOffset[A,B] [µT]	6	166	3	83
pMicroPerCycle [µT]	54 400	172 800	27 200	86 400
pOffsetCorrectionOut [µT]	15	15 080	15	7 504
pOffsetCorrectionStart [MT]	650	640	650	640
pRateCorrectionOut [µT]	33	104	17	61
pExternOffsetCorrection [µT]	14		7	

Table 127 — Modification to basic configuration 3 for protocol operation related configuration data – pOffsetCorrectionOut

Parameter	Modification to Basic Configuration 3	
	I	II
gdActionPointOffset [MT]	3	63
gdStaticSlot [MT]	150	
gdSymbolWindow [MT]	0	
gMacroPerCycle [MT]	680	720
gNumberOfMinislots	0	
gNumberOfStaticSlots	4	
gPayloadLengthStatic [two-byte word]	0	
gdCycle [µs]	1 360	4 320
gdMacrotick [µs]	2	6
gdNIT [MT]	80	120
gExternOffsetCorrection [µs]	0,35	0,35
pAllowHaltDueToClock	false	
pdAcceptedStartupRange [µT]	224	230
pClusterDriftDamping [µT]	0	2
pdListenTimeout [µT]	54 434	172 904
pLatestTx [Minislot]	0	
pMacroInitialOffset[A,B] [MT]	5	64
pMicroInitialOffset[A,B] [µT]	22	62
pMicroPerCycle [µT]	27 200	86 400
pOffsetCorrectionOut [µT]	15	7 450
pOffsetCorrectionStart [MT]	650	620
pRateCorrectionOut [µT]	17	52
pExternOffsetCorrection [µT]	7	

Table 128 defines the deviation for all variants.

Table 128 — Deviation for all variants

Variant	Deviation	
	Δt [µT]	***vExternOffsetControl***
(a)	*pOffsetCorrectionOut* – 10 µT	0
(b)	*pOffsetCorrectionOut* – 10 µT	1
(c)	*pOffsetCorrectionOut* + 10 µT	0
(d)	*pOffsetCorrectionOut* + 10 µT	–1
(e)	– (*pOffsetCorrectionOut* – 10) µT	0
(f)	– (pOffsetCorrectionOut – 10) µT	–1
(g)	– (*pOffsetCorrectionOut* + 10) µT	0
(h)	– (*pOffsetCorrectionOut* + 10) µT	1

— **Preamble (setup state)**

Preamble III.

The LT simulates its startup frame in slot 4 instead of slot 16.

— **Test execution**

1) Beginning with cycle 8, the LT shifts the start of cycle 8 and all following cycles by Δt.

2) In cycle 9, 500 µT after cycle start and before NIT the UT sets *vExternOffsetControl* as defined in the configuration to apply the external offset correction in the NIT of cycle 9.

3) In cycle 9, 500 µT after cycle start and before NIT (in terms of LT this approximates to 500 µT + |Δt| and |Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and that the IUT is in the

   (a, b, e, f) *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*)

   (c, d, g, h) *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*)

4) It is verified (LT) that the interval between the IUT's frames in cycle 9 and cycle 10 is

   (a, e) *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT

   (b) *pMicroPerCycle* + Δt + *pExternOffsetCorrection* + ξ + ξ_{IUT} µT

   (f) *pMicroPerCycle* + Δt – *pExternOffsetCorrection* + ξ + ξ_{IUT} µT

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT shall calculate the offset correction value correctly and shall enter the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) if the calculated offset exceeds the limit given by parameter *pOffsetCorrectionOut*. When the calculated offset correction value is inside the limit, the IUT shall apply the summation of the calculated offset correction and the external offset correction values.

7.3.2.22 pRateCorrectionOut

— **Test purpose**

Verify rate correction limits when external rate correction is applied.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 129, Table 130 and Table 131.

Table 129 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pRateCorrectionOut

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	4	63	1	4	63
gdStaticSlot [MT]	13	35	150	13	35	150
gdSymbolWindow [MT]	0	40	0	0	40	0
gdSymbolWindowActionPointOffset [MT]	1	4		1	4	
gMacroPerCycle [MT]	300	5 000	16 000	250	5 000	16 000
gNumberOfMinislots	0	219	348	0	219	348
gNumberOfStaticSlots	16	85		16	85	
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,0008	0,0003	0,0015	0,00049	0,0003	0,0015
gdCycle [µs]	300	5 000	16 000	250	5 000	16 000
gdNIT [MT]	92	14	59	42	14	59
gExternRateCorrection [µs]	0,35			0,175		
pAllowHaltDueToClock	false			false		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	24 040	400 242	1 283 846	40 040	800 482	2 567 692
pExternRateCorrection [µT]	14			14		
pLatestTx [Minislot]	0	188		0	188	
pMacroInitialOffset[A,B] [MT]	3	6	65	3	6	65
pMicroPerCycle [µT]	12 000	200 000	640 000	20 000	400 000	1 280 000
pOffsetCorrectionOut [µT]	45	153		45	260	
pOffsetCorrectionStart [MT]	295	4 990	15 992	245	4 990	15 992
pRateCorrectionOut [µT]	20	121		20	241	

Table 130 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pRateCorrectionOut

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1	4	63	1	4	63
gdStaticSlot [MT]	13	33	150	13	33	150
gdSymbolWindow [MT]	0	22	0	0	22	0
gdSymbolWindowActionPointOffset [MT]	1	3		1	3	
gMacroPerCycle [MT]	250	2 500	8 000	265	2 500	8 000
gNumberOfMinislots	0	140	160	0	140	160
gNumberOfStaticSlots	16	45		16	45	
gPayloadLengthStatic [two-byte word]	0			0		
gClockDeviationMax	0,00049	0,0003	0,0015	0,0009	0,0003	0,0015
gdCycle [µs]	500	5 000	16 000	530	5 000	16 000
gdNIT [MT]	42	12	70	57	12	70
gExternRateCorrection [µs]	0,35			0,35		
pAllowHaltDueToClock	false			false		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	40 040	400 242	1 283 846	21 240	200 122	641 924
pExternRateCorrection [µT]	14			14[a]		
pLatestTx [Minislot]	0	101		0	101	
pMacroInitialOffset[A,B] [MT]	2	5	64	2	5	64
pMicroPerCycle [µT]	20 000	200 000	640 000	10 600	100 000	320 000
pOffsetCorrectionOut [µT]	45	163	154	45	117	
pOffsetCorrectionStart [MT]	245	2 491	7 995	260	2 491	7 990
pRateCorrectionOut [µT]	20	121		20	61	

[a] *pExternRateCorrection* = 14 µT is an intentional protocol constraints violation

Table 131 — Modification to basic configuration 3 for protocol operation related configuration data – pRateCorrectionOut

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1	4	63
gdStaticSlot [MT]	22	58	150
gdSymbolWindow [MT]	0	23	0
gdSymbolWindowActionPointOffset [MT]	1	3	
gMacroPerCycle [MT]	400	2 500	8 000
gNumberOfMinislots	0	145	580
gNumberOfStaticSlots	16	25	
gPayloadLengthStatic [two-byte word]	0		
gClockDeviationMax	0,0006	0,0003	0,0015
gdCycle [µs]	800	5 000	16 000
gdNIT [MT]	48	11	130
gExternRateCorrection [µs]	0,35		
pAllowHaltDueToClock	false		
pClusterDriftDamping [µT]	0		
pdListenTimeout [µT]	32 040	200 122	641 924
pExternRateCorrection [µT]	14[a]		
pLatestTx [Minislot]	0	68	
pMacroInitialOffset[A,B] [MT]	3	6	65
pMicroPerCycle [µT]	16 000	100 000	320 000
pOffsetCorrectionOut [µT]	45	118	
pOffsetCorrectionStart [MT]	395	2 492	7 990
pRateCorrectionOut [µT]	20	61	

[a] *pExternRateCorrection* = 14 µT is an intentional protocol constraints violation

Table 132 defines the deviation for all variants.

Table 132 — Deviation for all variants

Variant	Deviation	
	Δt [µT]	***vExternRateControl***
(a)	*pRateCorrectionOut* – 10 µT	0
(b)	*pRateCorrectionOut* – 10 µT	1
(c)	*pRateCorrectionOut* + 10 µT	0
(d)	*pRateCorrectionOut* + 10 µT	–1
(e)	– (*pRateCorrectionOut* – 10) µT	0
(f)	– (*pRateCorrectionOut* – 10) µT	–1
(g)	– (*pRateCorrectionOut* + 10) µT	0
(h)	– (*pRateCorrectionOut* + 10) µT	1

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) At the beginning of cycle 9, the LT shifts its cycle start by Δt, i.e. all following cycles are shifted.

2) In cycle 9, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| and |Δt| before NIT) the UT sets *vExternRateControl* as defined in the configuration to apply external rate correction in cycle 10.

3) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT after cycle start and *pExternalRateCorrectionOut* µT before NIT), it is verified (UT) that the rate correction value *vInterimRateCorrection* = Δt + ξ_{IUT} µT.
It is also verified (UT) that the IUT is

(a, d, e, h) in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*)

(b, c, f, g) in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT's rate correction does not exceed ± *pRateCorrectionOut* although external rate correction is applied.

7.3.2.23 pSecondKeySlotID and pTwoKeySlotMode

— **Test purpose**

Verify correct behaviour of the IUT if *pTwoKeySlotMode* is set to true. The IUT shall transmit its second startup frame in slot *pSecondKeySlotID* and shall set the rate and offset correction values to 0.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 133, Table 134 and Table 135.

Table 133 — Modification to basic configurations 1a and 1b for protocol operation related configuration data – pSecondKeySlotID and pTwoKeySlotMode

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1			1		
gdStaticSlot [MT]	15			15		
gdSymbolWindow [MT]	40			40		
gdSymbolWindowActionPointOffset [MT]	1			1		
gMacroPerCycle [MT]	15 400			15 400		
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	1 023			1 023		
gPayloadLengthStatic [two-byte word]	1			1		
gdCycle [µs]	15 400			15 400		
gdNIT [MT]	15			15		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	1 232 006			2 464 006		
pKeySlotID	2			2		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	3			3		
pMicroPerCycle [µT]	616 000			1 232 000		
pOffsetCorrectionOut [µT]	15			15		
pOffsetCorrectionStart [MT]	15 397			15 397		
pRateCorrectionOut [µT]	3			3		
pKeySlotOnlyEnabled	true			true		
pSecondKeySlotID	1	512	1 023	1	512	1 023
pTwoKeySlotMode	true			true		

Table 134 — Modification to basic configurations 2a and 2b for protocol operation related configuration data – pSecondKeySlotID and pTwoKeySlotMode

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	1			1		
gdStaticSlot [MT]	14			15		
gdSymbolWindow [MT]	0			23		
gdSymbolWindowActionPointOffset [MT]	1			1		
gMacroPerCycle [MT]	8 000			8 000		
gNumberOfMinislots	0			0		
gNumberOfStaticSlots	570			530		
gPayloadLengthStatic [two-byte word]	1			1		
gdCycle [µs]	16 000			16 000		
gdNIT [MT]	20			27		
pClusterDriftDamping [µT]	0			0		
pdListenTimeout [µT]	1 280 006			640 006		
pKeySlotID	2			2		
pLatestTx [Minislot]	0			0		
pMacroInitialOffset[A,B] [MT]	2			2		
pMicroPerCycle [µT]	640 000			320 000		
pOffsetCorrectionOut [µT]	15			15		
pOffsetCorrectionStart [MT]	7 998			7 997		
pRateCorrectionOut [µT]	3			3		
pKeySlotOnlyEnabled	true			true		
pSecondKeySlotID	1	256	570	1	256	530
pTwoKeySlotMode	true			true		

Table 135 — Modification to basic configuration 3 for protocol operation related configuration data – pSecondKeySlotID and pTwoKeySlotMode

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	1		
gdStaticSlot [MT]	26		
gdSymbolWindow [MT]	24		
gdSymbolWindowActionPointOffset [MT]	1		
gMacroPerCycle [MT]	8 000		
gNumberOfMinislots	0		
gNumberOfStaticSlots	306		
gPayloadLengthStatic [two-byte word]	1		
gdCycle [µs]	16 000		
gdNIT [MT]	20		
pClusterDriftDamping [µT]	0		
pdListenTimeout [µT]	640 012		
pKeySlotID	2		
pLatestTx [Minislot]	0		
pMacroInitialOffset[A,B] [MT]	3		
pMicroPerCycle [µT]	320 000		
pOffsetCorrectionOut [µT]	15		
pOffsetCorrectionStart [MT]	7 995		
pRateCorrectionOut [µT]	3		
pKeySlotOnlyEnabled	true		
pSecondKeySlotID	1	127	306
pTwoKeySlotMode	true		

The IUT is configured to transmit frames in slots *pKeySlotID* and *pSecondKeySlotID.*

— **Preamble (setup state)**

1) The UT resets the IUT.

2) The UT sets the IUT into *POC:config* with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state (*vPOC!State* = *READY*) with the CHI command CONFIG_COMPLETE.

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure of the IUT with the CHI command RUN.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) It is verified (LT) that the IUT sends a CAS (low phase of length *gdTSSTransmitter* + *cdCAS*) starting *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the CHI command RUN.

2) In cycles 0 to 5, it is verified (LT) that the IUT transmits its frames in slot *pKeySlotID* and *pSecondKeySlotID* on the available channel(s) correctly, i.e. the frame ID is set to *pKeySlotID* (first startup frame) and *pSecondKeySlotID* (second startup frame) respectively, the cycle count to the respective cycle number, the payload length to *gPayloadLengthStatic*, the sync frame indicator, the startup frame indicator are set to '1' and the null frame indicator is set to '0', the payload preamble indicator is set to '0' and the payload bytes are all 0x00.

3) In cycle 6, within 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized *(vPOC!State* = *NORMAL_ACTIVE*).

4) In cycles 6 and 7, it is verified (LT) that the IUT transmits its frames in slot *pKeySlotID* and *pSecondKeySlotID* on the available channel(s) correctly, i.e. the frame ID is set to *pKeySlotID* (first startup frame) and *pSecondKeySlotID* (second startup frame) respectively, the cycle count to the respective cycle number, the payload length to *gPayloadLengthStatic*, the sync frame indicator, the startup frame indicator and the null frame indicator are set to '1', the payload preamble indicator is set to '0' and the payload is the default payload.

5) In cycle 7, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames list(s) for the odd cycle on the available channel(s) contain(s) the IUT's startup frame IDs (*pKeySlotID* and *pSecondKeySlotID*). It is also verified (UT) that the number of channel aligned startup pairs received or transmitted during the previous double cycle is 2 in the startup frame status (*vStartupPairs* = 2).

6) In cycle 8, 500 µT after the cycle start and before the NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 and that the offset correction value is *vInterimOffsetCorrection* = 0.

7) In cycle 8, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames list(s) for the even cycle on the available channel(s) contain(s) the IUT's startup frame IDs (*pKeySlotID* and *pSecondKeySlotID*).

8) In cycle 9, 500 µT after the cycle start and before the NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 and that the offset correction value is *vInterimOffsetCorrection* = 0.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT shall transmit its startup frames in slots *pKeySlotID* and *pSecondKeySlotID* and set rate and offset correction values are as expected.

7.3.2.24 pKeySlotID = 0

— Test purpose

Verify correct behaviour of the IUT according to the parameter *pKeySlotID*. The IUT shall transmit its frame in slot *pKeySlotID* and shall not transmit a frame when *pKeySlotID* is set to 0.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 136.

Table 136 — Modification to basic configurations for protocol operation related configuration data – pKeySlotID = 0

Parameter	Modification	
	I	II
pKeySlotID	0	3

— Preamble (setup state)

Preamble III.

— Test execution

In cycle 9, it is verified (LT) that the IUT transmits a static frame with the null frame indicator set to '1' holding the respective payload in slot 1 and

(I) does not transmit any further frames.

(II) a static frame in slot *pKeySlotID* with the sync frame indicator, the startup frame indicator and the payload preamble indicator set to '0'.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frame in slot *pKeySlotID* except *pKeySlotID* is set to 0.

7.3.2.25 External rate and offset correction (TTL Mode)

— Test purpose

Verify correct behaviour of the IUT if *pTwoKeySlotMode* is set to true and the minimum external rate and offset correction values are applied.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 137.

Table 137 — Modification to basic configurations for protocol operation related configuration data – external rate and offset correction (TTL Mode)

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
pExternOffsetCorrection [µT]	1	1	1	1	1
pExternRateCorrection [µT]	1	1	1	1	1
pdListenTimeout [µT]	400 006	800 006	400 006	200 006	200 012
pOffsetCorrectionOut [µT]	15	15	15	15	15
pRateCorrectionOut [µT]	3	3	3	3	6
pKeySlotOnlyEnabled	true	true	true	true	true
pSecondKeySlotID	2	2	2	2	2
pTwoKeySlotMode	true	true	true	true	true

The IUT is configured to transmit frames in slots *pKeySlotID* and *pSecondKeySlotID*.

The test has to be executed repeatedly with *vExternOffsetControl* and *vExternRateControl* set to (a) -1 and (b) +1.

— **Preamble (setup state)**

1) The UT resets the IUT.

2) The UT sets the IUT into *POC:config* with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state (*vPOC!State* = *READY*) with the CHI command CONFIG_COMPLETE.

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure of the IUT with the CHI command RUN.

7) In cycle 6, within 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is synchronized *(vPOC!State* = *NORMAL_ACTIVE*).

— **Test execution**

1) In slot 1 of cycle 6, the IUT sets *vExternOffsetControl* to the respective value.

2) In cycle 7, it is verified (LT) that the interval between the IUT's frames in cycle 6 and 7 is *pMicroPerCycle* * *pSamplesPerMicrotick* / *pdMicrotick* ± σT.

3) In cycle 8, it is verified (LT) that the interval between the IUT's frames in cycle 7 and 8 is (*pMicroPerCycle* + *vExternOffsetControl* * *pExternOffsetCorrection*) * *pSamplesPerMicrotick* / *pdMicrotick* ± σT.

4) In slot 1 of cycle 9 the IUT sets *vExternOffsetControl* to the 0 and sets the *vExternRateControl* to the respective value.

5) In cycle 11, it is verified (LT) that the interval between the IUT's frames in cycle 10 and 11 is (*pMicroPerCycle* + *vExternOffsetControl * pExternRateCorrection*) * *pSamplesPerMicrotick* / *pdMicrotick* ± σT.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its startup frames in slots *pKeySlotID* and *pSecondKeySlotID* and applies the external rate and offset correction correctly.

7.3.3 Frame related configuration data

7.3.3.1 gPayloadLengthStatic

— Test purpose

Verify correct behaviour of the IUT according to the parameter *gPayloadLengthStatic*. The IUT shall transmit frames with a payload of length *gPayloadLengthStatic* two-byte words and with the payload length in the header set to *gPayloadLengthStatic*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 138.

Table 138 — Modification to basic configurations 1a and 1b for frame related configuration data – gPayloadLengthStatic

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdStaticSlot [MT]	274			271			331		
gNumberOfStaticSlots	10			5			4		
gPayloadLengthStatic [two-byte word]	0	1	127	0	1	127	0	1	76
gdNIT [MT]	249			142			137		

For *gPayloadLengthStatic* = 1 two-byte word the payload data is 0x01 in byte 0 and 0x02 in byte 1, for *gPayloadLengthStatic* = 127 two-byte words the payload data is 0x01 in byte 0, 0x02 in byte 1, ..., 0xFE in byte 253 and for *gPayloadLengthStatic* = 76 two-byte words the payload data is 0x01 in byte 0, 0x02 in byte 1, ..., 0x98 in byte 151.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

In cycle 7, it is verified (LT) that the IUT transmits its frame in slot 1 with a payload of size *gPayloadLengthStatic* two-byte words, that the payload length in the header is set to *gPayloadLengthStatic* and

(I) that the frame contains no payload data,

(II) that the payload data is 0x01 in byte 0 and 0x02 in byte 1

(III) that the payload data is 0x01 in byte 0, 0x02 in byte 1, ..., 0xFE in byte 253 for Basic Configuration 1a, 1b, 2a and 2b and that the payload data is 0x01 in byte 0, 0x02 in byte 1, ..., 0x98 in byte 151 for Basic Configuration 3.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frame with a payload of size *gPayloadLengthStatic* correctly and the payload length in the header is set accordingly.

7.3.3.2 gdTSSTransmitter

— **Test purpose**

Verify correct handling of the parameter *gdTSSTransmitter*. The IUT shall transmit frames with a TSS of length *gdTSSTransmitter* and an MTS of length *gdTSSTransmitter* + *cdCAS*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 139, Table 140 and Table 141.

Table 139 — Modification to basic configurations 1a and 1b for frame related configuration data – gdTSSTransmitter

Parameter	Modification to Basic Configuration					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	77	97	105	77	97	105
gdTSSTransmitter [gdBit]	1[a]	11	15	1[a]	11	15
pDecodingCorrection [µT]	12	52	68	24	104	136
pMacroInitialOffset[A,B] [MT]	5	6		5	6	
pMicroInitialOffset[A,B] [µT]	23		7	46		14
[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation.						

Table 140 — Modification to basic configurations 2a and 2b for frame related configuration data – gdTSSTransmitter

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	70	80	84	70	80	84
gdTSSTransmitter [gdBit]	1[a]	6	8	1[a]	6	8
pDecodingCorrection [µT]	24	64	80	12	32	40
pMacroInitialOffset[A,B] [MT]	4		5	4		5
pMicroInitialOffset[A,B] [µT]	46	6	70	23	3	35
[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation.						

Table 141 — Modification to basic configuration 3 for frame related configuration data – gdTSSTransmitter

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	66	72	74
gdTSSTransmitter [gdBit]	1[a]	4	5
pDecodingCorrection [µT]	24	48	56
pMacroInitialOffset[A,B] [MT]	4	5	
pMicroInitialOffset[A,B] [µT]	6	22	14
[a] *gdTSSTransmitter* = 1 [*gdBit*] which is an intentional protocol constraints violation.			

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to send a startup frame in (static) slot 1, a regular frame in (static) slot 15 and a regular frame in (dynamic) slot n_{dyn}.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) The UT requests the IUT to send an MTS in the symbol window of cycle 7.

2) In cycle 7, the UT requests the IUT to send regular frames in slot 15 and n_{dyn}.

3) In cycle 7, it is verified (LT) that the IUT transmits its frames in slot 1, slot 15, and slot n_{dyn} with a TSS of length *gdTSSTransmitter * gdBit * pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT and the MTS with a length of (*gdTSSTransmitter* + *cdCAS*) * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames with a TSS of length *gdTSSTransmitter * gdBit * pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT and the MTS with a length of (*gdTSSTransmitter* + *cdCAS*) * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT.

7.3.4 Symbol related configuration data

7.3.4.1 gdWakeupTxIdle

— Test purpose

Verify the correct interpretation of *gdWakeupTxIdle* in the WUS encoding process. The IUT shall transmit a WUP comprising *pWakeupPattern* low-idle phases of length *gdWakeupTxActive* and *gdWakeupTxIdle* respectively.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) It is verified (LT) that the IUT transmits a WUP consisting of *pWakeupPattern* consecutive WUS, each consisting of a low phase of length (*gdWakeupTxActive* + *cdStaggerDelay*) * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT and an idle phase of length (*gdWakeupTxIdle* – *cdStaggerDelay*) * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT on TxD port of channel *pWakeupChannel*. It is also verified (LT) that the TxEN output and the TxD output on channel *pWakeupChannel* become low simultaneously. The TxEN output stays high for a period of *gdWakeupTxIdle* * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT between two low phases.

4) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *TRANSMITTED* *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick* + 2 500 µT after the CHI command WAKEUP.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits a WUP comprising *pWakeupPattern* WUS. Each WUS comprises a low phase and a high phase of expected duration at TxD and TxEN output ports of IUT. Both signals shall switch to low simultaneously. TxD returns to high delayed by *cdStaggerDelay*, compared to TxEN.

7.3.4.2 gdWakeupTxActive

— Test purpose

Verify the correct interpretation of *gdWakeupTxActive* in the WUS encoding process. The IUT shall transmit a WUP comprising *pWakeupPattern* low-high phases of length *gdWakeupTxActive* and *gdWakeupTxIdle* respectively.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) It is verified (LT) that the IUT transmits a WUP consisting of *pWakeupPattern* consecutive WUS, each consisting of a low phase of length (*gdWakeupTxActive* + *cdStaggerDelay*) * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT and an idle phase of length (*gdWakeupTxIdle* – *cdStaggerDelay*) * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT on TxD port of channel *pWakeupChannel.* It is also verified (LT) that the TxEN output and the TxD output on channel *pWakeupChannel* become low simultaneously and the TxEN output stays low for a period of *gdWakeupTxActive* * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT.

4) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *TRANSMITTED pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick* + 2 500 µT after the CHI command WAKEUP.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits a WUP comprising *pWakeupPattern* WUS. Each WUS comprises a low phase and a high phase of expected duration at TxD and TxEN output ports of IUT. Both signals shall switch to low simultaneously. TxD returns to high delayed by *cdStaggerDelay*, compared to TxEN.

7.3.4.3 pWakeupPattern

— Test purpose

Verify the correct interpretation of *pWakeupPattern* in the WUP encoding process. The IUT shall transmit a WUP comprising *pWakeupPattern* low phases.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 142.

Table 142 — Modification to basic configurations for symbol related configuration data – pWakeupPattern

Parameter	Modification			
	I	II	III	IV
pWakeupPattern	2	30	31	63

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) It is verified (LT) that the IUT transmits a WUP consisting of *pWakeupPattern* consecutive WUS, each consisting of a low-high phase on channel *pWakeupChannel.* It is also verified (LT) that the TxEN output and the TxD output on channel *pWakeupChannel* become low simultaneously and that the TxD port becomes high *cdStaggerDelay* after TxEN port.

4) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *TRANSMITTED pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick* + 2 500 µT after the CHI command WAKEUP.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a WUP comprising *pWakeupPattern* low phases.

7.3.4.4 pWakeupPattern < 2

— **Test purpose**

Verify the correct interpretation of *pWakeupPattern* in the POC process if the parameter is set to 0 or 1.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 143.

Table 143 — Modification to basic configurations for symbol related configuration data – pWakeupPattern < 2

Parameter	Modification
pWakeupPattern	2

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) It is verified (LT) that the IUT transmits a WUP consisting of *pWakeupPattern* consecutive WUS, each consisting of a low-high phase on channel *pWakeupChannel*.

4) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *TRANSMITTED* *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick* + 2 500 µT after the CHI command WAKEUP.

5) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*) and configures the IUT with

   (a) *pWakeupPattern* = 0.

   (b) *pWakeupPattern* = 1.

6) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

7) It is verified (UT) that IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *TRANSMITTED* 2 500 µT after the CHI command CONFIG_COMPLETE.

8) The UT triggers the wakeup process with the CHI command WAKEUP.

9) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *UNDEFINED* 2 500 µT after the CHI command WAKEUP.

10) It is verified (LT) that the IUT does not transmit any communication element *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT after the CHI command WAKEUP.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT does not transmit a WUP if *pWakeupPattern* < 2 and updates the *vPOC!WakeupStatus* correctly.

7.3.4.5 Intentional prevention of CAS reception

— Test purpose

Verify correct handling of the parameter *gdCASRxLowMax [gdBit].* The IUT shall not accept any CAS if *gdCASRxLowMax = 28 (gdCASRxLowMax* < *cdCASRxLowMin)*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 144.

Table 144 — Modification to basic configurations for symbol related configuration data – intentional prevention of CAS reception

Parameter	Modification
gdCASRxLowMax [gdBit]	28
gListenNoise	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT generates two valid CAS symbols (low phases of length *cdCAS* during the *POC:coldstart listen* state) on the available channel(s), the first one after 1 / 2 * *pdListenTimeout* and the second one after 3 / 2 * *pdListenTimeout* after the CHI command RUN.

4) It is verified (LT) that the IUT sends a CAS *pdListenTimeout* * *gListenNoise* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

5) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

Figure 31 depicts the intentional prevention of CAS reception.

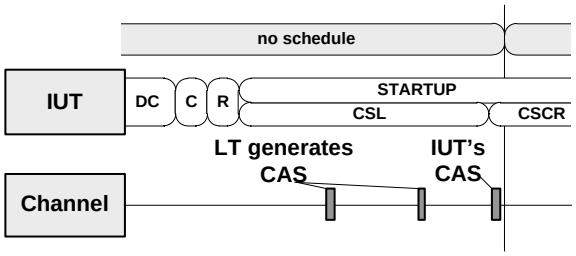


Figure 31 — Intentional prevention of CAS reception

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT shall not accept any CAS symbol and handle the received valid CAS symbols as noise, if *gdCASRxLowMax* < *cdCASRxLowMin*. The IUT shall enter the *POC:coldstart collision resolution* state after the noise timer is expired.

7.3.5 Control of protocol operation control

7.3.5.1 POC:default config

7.3.5.1.1 Command CONFIG

— Test purpose

Verify the correct behaviour of IUT regarding default values and write access to protocol configuration data in *POC:default config* state. Verify correct behaviour after CHI command CONFIG in *POC:default config* state. Verify that the IUT does neither activates the TxD output nor the TxEN output after either a CHI RUN command or a CHI WAKEUP command if the UT doesn't change the default configuration.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

— Test execution

It shall be verified (LT) that the TxEN output and the TxD output are high during the whole test execution.

1) After 2 500 µT it is verified (UT) that the IUT has entered the POC:default config state
   - (*vPOC!State* = *DEFAULT_CONFIG*) and that
   - *vPOC!Freeze* = false,
   - *vPOC!CHIHaltRequest* = false,
   - *vPOC!CHIReadyRequest* = false,
   - *vPOC!ColdstartNoise* = false,
   - *vPOC!SlotMode* = *KEYSLOT*,
   - *vPOC!ErrorMode* = *ACTIVE*,
   - *vPOC!WakeupStatus* = *UNDEFINED,*
   - *vPOC!StartupState* = *UNDEFINED,*
   - *vClockCorrectionFailed* = 0, and
   - *vAllowPassiveToActive* = 0.
2) It is verified (UT) that the IUT holds the default configuration data, i.e.

- all[2)] buffers (including FIFO buffers) are configured such that they can neither transmit nor receive,
- no slot is assigned for transmission or reception,
- the payload data valid flag of all message buffers is set to false,
- *pKeySlotID* and *pSecondKeySlotID* is set to 0,
- *pKeySlotUsedForStartup* is set to false,
- *pTwoKeySlotMode* is set to false
- *pWakeupPattern* is set to 0 and
- *pExternalSync* (if applicable) is set to false.

All other protocol configuration data, message buffer configuration data and queued receive buffer configuration data are set to implementation dependent predefined initialization values.

NOTE the initialization values for each individual configuration in the default configuration must be described in the documentation of the implementation.

It is also verified that the UT does not have write access to any protocol configuration data in *POC:default config* state 2 500 µT after the IUT has entered the *POC:default config* state.

3) The UT issues the CHI command CONFIG.

4) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT initiates the startup procedure with the CHI command RUN.

8) After 3 * *pdListenTimeout* µT it is verified (UT) that the IUT is still in the *POC:integration listen* state (*vPOC!State* = *STARTUP*).

9) The UT sets the IUT into the *POC:ready* state with the CHI command IMMEDIATE_READY..

10) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*),

11) The UT issues the CHI command WAKEUP.

12) After 2 500 µT it is verified (UT) that the IUT is still in the *POC:ready* state (*vPOC!State* = *READY*).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

---

2) In case the implementation allows a dynamic allocation of message buffers or FIFOs, it is verified that no message buffer or FIFO buffer is enabled.

— **Pass criteria**

The IUT provides the correct default protocol configuration data and the protocol configuration data is read-only in the *POC:default config* state. The protocol state changes to *POC:config* after issuing the CHI command CONFIG. The IUT does not activates the TxD output and the TxEN output i.e. the level of the TxD output and TxEN output remain high during the whole test execution.

7.3.5.1.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:default config* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:default config* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:default config* state (*vPOC!State* = *DEFAULT_CONFIG)* and that *vPOC!Freeze* = false.

3) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) The UT issues the CHI command FREEZE.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *DEFAULT_CONFIG)* and that *vPOC!Freeze* = true.

4) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.1.3 Command DEFERRED_HALT

— Test purpose

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:default config* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:default config* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:default config* state (*vPOC!State* = *DEFAULT_CONFIG)* and *vPOC!Freeze* = false.

3) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) The UT issues the CHI command DEFERRED_HALT.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT)* and *vPOC!CHIHaltRequest* = false and *vPOC!CHIReadyRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions and *vPOC!CHIHaltRequest* and *vPOC!CHIReadyRequest* are set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.2 POC:config

7.3.5.2.1 Command CONFIG_COMPLETE

— Test purpose

Verify the correct behaviour of IUT regarding write access to the protocol configuration data in *POC:config* state. Verify correct behaviour after CHI command CONFIG_COMPLETE in *POC:config* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:config* to *POC:ready*, initiated by the CHI command CONFIG_COMPLETE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 145.

Table 145 — Modification to basic configurations for POC:config – command CONFIG_COMPLETE

Parameter	Modification
pKeySlotOnlyEnabled	true

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

5) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

— **Test execution**

1) The UT configures the IUT with the given configuration.

2) It is verified (UT) that the IUT holds the provided configuration data.

3) It is verified (UT) that the IUT did not generate any interrupt request and no interrupt status indications are set.

4) The UT issues the CHI command CONFIG_COMPLETE.

5) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*) and that the slot mode is *vPOC!SlotMode* = *KEYSLOT*.

6) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

All given protocol configurations are written to the IUT properly. The IUT performs the expected POC state transitions and the slot mode is set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:ready*.

7.3.5.2.2 Command FREEZE

— Test purpose

Verify correct behaviour after CHI command FREEZE in *POC:config* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:config* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) The UT issues the CHI command FREEZE.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *CONFIG)* and *vPOC!Freeze* = true.

4) It is verified (UT) that the IUT did not generate any interrupt request.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions and *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on POC state transition to the *POC:halt* state.

7.3.5.2.3 Command DEFERRED_HALT

— Test purpose

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:config* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:config* to *POC:halt*, initiated by the CHI command DEFERED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) The UT issues the CHI command DEFERRED_HALT.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and
vPOC!CHIHaltRequest = false and *vPOC!CHIReadyRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and the UT reads out the expected values for the variables
vPOC!CHIHaltRequest and *vPOC!CHIReadyRequest*. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.3 POC:ready

7.3.5.3.1 Command CONFIG

— **Test purpose**

Verify correct behaviour after CHI command CONFIG in *POC:ready* state. Verify read and write access to protocol configuration data in *POC:ready* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

— **Test execution**

1) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*) and *vPOC!CHIHaltRequest* = false and *vPOC!CHIReadyRequest* = false.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected POC state transitions and *vPOC!CHIHaltRequest and vPOC!CHIReadyRequest* are set accordingly. The protocol configuration data may be read and shall not written while the IUT is in the *POC:ready* state.

7.3.5.3.2 Command RUN

— **Test purpose**

Verify correct behaviour after CHI command RUN in *POC:ready* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:ready* to *POC:normal active*, initiated by the CHI command RUN). Verify that the integration of a node starts only with frames that have an even cycle counter.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration. The IUT is configured to send a startup frame in slot 1.

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests. The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

— **Test execution**

1) The LT simulates startup frames in slot 2 and slot 3 of cycles 1 to 9.

2) The UT issues the CHI command RUN in slot 4 of cycle 1.

3) After 2 500 µT it is verified (UT) that

   — the IUT has entered the startup state (*vPOC!State* = *STARTUP*) and that *vPOC!CHIReadyRequest* = false,

   — *vPOC!CHIHaltRequest* = false,

   — *vPOC!ErrorMode* = *ACTIVE* and *vPOC!ColdstartNoise* = false.

4) In cycle 8, before the start of the NIT it is verified (UT) that the IUT has not generated any interrupt request for the interrupt source POC state transition.

5) In cycle 9, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT has entered the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*), that *vClockCorrectionFailed* = 0 and that *vPOC!StartupState* = *UNDEFINED*. It is also verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected POC state transitions and sets *vPOC!CHIReadyRequest*, *vPOC!CHIHaltRequest*, *vPOC!ErrorMode*, *vPOC!ColdstartNoise*, *vClockCorrectionFailed*, and *vPOC!StartupState* accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:normal active*.

7.3.5.3.3 Command WAKEUP

— Test purpose

Verify correct behaviour after CHI command WAKEUP in *POC:ready* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup send* to *POC:ready*).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration. The UT issues the CHI command CONFIG_COMPLETE.

5) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

6) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests. The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

— Test execution

1) The UT issues the CHI command WAKEUP.

2) After 2 500 µT it is verified (UT) that

   — the IUT is in a wakeup state (*vPOC!State* = *WAKEUP*) and that *vPOC!CHIReadyRequest* = false,

   — *vPOC!CHIHaltRequest* = false and *vPOC!WakeupStatus* = *UNDEFINED*.

   It is also verified (UT) that the IUT has not generated any interrupt request for the interrupt source POC state transition.

3) (*pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick* + 2 500) µT after the CHI command WAKEUP it is verified (UT) that
   the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* is set to *TRANSMITTED*.
   It is also verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected POC state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:ready*.

7.3.5.3.4 Command FREEZE

— Test purpose

Verify correct behaviour after CHI command FREEZE in *POC:ready* state. Verify that no frame is processed in *POC:ready* state and in *POC:halt* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:ready* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 146.

Table 146 — Modification to basic configurations for for symbol related configuration data – command FREEZE

Parameter	Modification
pKeySlotID	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The LT simulates a frame with the frame ID set to 1, the cycle count to 0 and the payload length to 1 holding a payload of 0xAA in byte 0 and 0x55 in byte 1 on both channels. Reserved bit, payload preamble indicator, sync frame indicator and startup frame indicator are set to '0'. The null frame indicator is set to '1'.

2) It is verified (UT) that the receive buffer assigned to slot 1 is not updated, i.e. the receive buffer holds its initial contents, and that the valid frame indicator signals no reception of a valid frame in the

aggregated channel status of the IUT *gdCycle* after the frame end. The IUT does not process the LT's frame.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

4) The UT issues the CHI command FREEZE.

5) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *READY)* and *vPOC!Freeze* = true.

6) It is verified (UT) that the IUT did not generate any interrupt request. The LT simulates a frame with the frame ID set to 1, the cycle count to 0 and the payload length to 1 holding a payload of 0x55 in byte 0 and 0xAA in byte 1 on both channels after the IUT entered *POC:halt* state. Reserved bit, payload preamble indicator, sync frame indicator and startup frame indicator are set to '0'. The null frame indicator is set to '1'.

7) It is verified (UT) that the receive buffer assigned to slot 1 is not updated, i.e. the receive buffer holds its initial contents, and that the valid frame indicator signals no reception of a valid frame in the aggregated channel status of the IUT *gdCycle* after the frame end. The IUT does not process the LT's frame.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!Freeze* is set accordingly. The IUT does not process any frames while in *POC:ready* state or in *POC:halt* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.3.5 Command DEFERRED_HALT

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:ready* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:ready* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) The UT issues the CHI command DEFERRED_HALT.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT)* and that *vPOC!CHIHaltRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!CHIHaltRequest* is set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.3.6 Command ALLOW_COLDSTART

— **Test purpose**

Verify correct behaviour after CHI command ALLOW_COLDSTART in *POC:ready* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

— **Test execution**

1) The UT issues the CHI command ALLOW_COLDSTART.

2) After 2 500 µT the UT issues the CHI command RUN.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected POC state transitions accordingly.

7.3.5.4 POC:startup states

The test cases of this subclause have been replaced by separate test cases for all startup states in 7.3.5.11 through 7.3.5.19.

7.3.5.5 POC:normal active

7.3.5.5.1 Command IMMEDIATE_READY and received WUDOP

— **Test purpose**

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:normal active* state. The IUT shall set the *vColdstartInhibit* flag on return to the *POC:ready* state from the *POC:normal active* state. Verify read and write access to protocol configuration data in *POC:normal active* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal active* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 147.

Table 147 — Modification to basic configurations for POC:normal active – command IMMEDIATE_READY and received WUDOP

Parameter	Modification	
	I	II
pKeySlotOnlyEnabled	false	true

— **Preamble (setup state)**

Preamble II.

The preamble is supplemented by the UT, which enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 7, the LT simulates its startup frame in slot 2.

2) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

4) In the symbol window of cycle 7, starting at the symbol window action point (referred as t0 later in this test), the LT simulates a WUDOP of Low-High-Low-High-Low phases. The low phase has a length of *gdWakeupTxActive* [*gdBit*], the high phase a length of *gdWakeupTxActive* [*gdBit*].

5) At t1 = t0 + 1,5 * *gdWakeupTxActive* * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxActive* * *gdBit* / *pdMicrotick* µT during the first high phase of the LT's WUDOP, the UT issues the CHI command IMMEDIATE_READY.

6) At t1 + 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*), that *vPOC!CHIHaltRequest* = false and that the slot mode is (I) *vPOC!SlotMode* = *ALL* and (II) *vPOC!SlotMode* = *KEYSLOT*.

   It is verified (UT) that the IUT did not generate any interrupt request.

7) It is verified (UT) that the wakeup pattern received indicator in the wakeup and startup status is set at t0 + 3 * *gdWakeupTxActive* + *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* + 500 µT.

8) The UT issues the CHI command RUN 2 * *gdCycle* after the IUT returned to the *POC:ready* state.

9) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

Figure 32 depicts the command IMMEDIATE_READY and received WUDOP – command issued in *POC:normal active*.

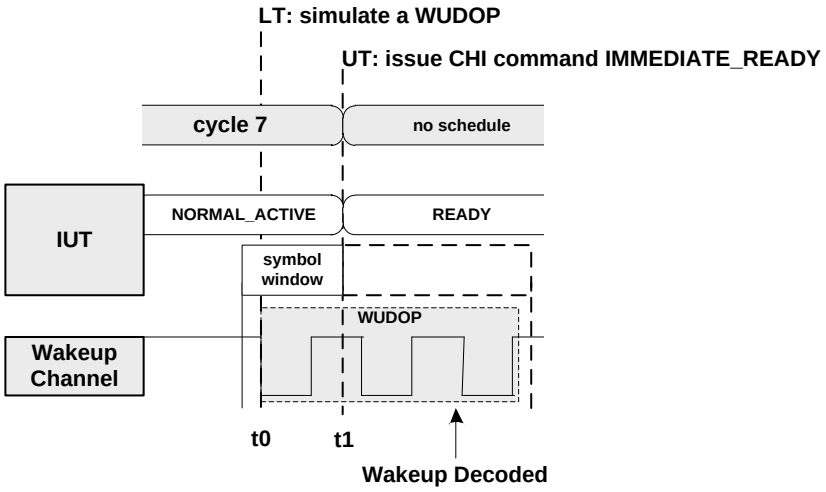


Figure 32 — Command IMMEDIATE_READY and received WUDOP – command issued in POC:normal active

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions and the slot mode is set accordingly. The IUT enters the integration path after the *vColdstartInhibit* flag was set on the return to the *POC:ready* state from the *POC:normal active* state. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:normal active* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.5.2 Command DEFERRED_HALT - Command CLEAR_DEFERRED

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:normal active* state. Verify the correct behaviour after CHI command CLEAR_DEFERRED in *POC:normal active* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 148.

Table 148 — Modification to basic configurations for POC:normal active – command DEFERRED_HALT - Command CLEAR_DEFERRED

Parameter	Modification
gMaxWithoutClockCorrectionPassive	2
gMaxWithoutClockCorrectionFatal	3

— **Preamble (setup state)**

Preamble II.

The preamble is supplemented by the UT, which enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would stay in state *POC:normal active* without DEFERRED_HALT command:

1) In cycles 7 to 9, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIHaltRequest* = false.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

4) At the beginning of cycle 8, the UT issues the CHI command DEFERRED_HALT.

5) After 2 500 µT it is verified (UT) that *vPOC!CHIHaltRequest* = true.

6) In cycle 9, 500 µT after cycle start, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIHaltRequest* = true.

7) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

Test instance 2 – IUT would leave state *POC:normal active* to *POC:halt* (DEFERRED_HALT) without CLEAR_DEFERRED command:

1) In cycles 7 to 9, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIHaltRequest* = false.

3) At the beginning of cycle 8, the UT issues the CHI command DEFERRED_HALT.

4) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that *vPOC!CHIHaltRequest* = true.

5) 5 000 µT after the CHI command DEFERRED_HALT and latest 3 000 µT before NIT, the UT issues the CHI command CLEAR_DEFERRED.

6) 2 500 µT after the CHI command CLEAR_DEFERRED it is verified (UT) that *vPOC!CHIHaltRequest* = false.

7) In cycle 9, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

8) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

Test instance 3 – IUT would leave state *POC:normal active* to *POC:normal passive* without DEFERRED_HALT command:

1) In cycles 7, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIHaltRequest* = false.

3) In cycles 8 to 11, the LT stops transmission of its startup frame in slot 2.

4) In cycle 11, 500 µT after cycle start and latest 2 000 µT before NIT, the UT issues the CHI command DEFERRED_HALT.

5) In cycle 12, 500 µT after cycle start, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIHaltRequest* = true.

Test instance 4 – IUT would leave state *POC:normal active* to *POC:halt* (DEFERRED_HALT command) without DEFERRED_READY command:

1) In cycles 7 to 8, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIHaltRequest* = false and *vPOC!CHIReadyRequest* = false.

3) At the beginning of cycle 8, the UT issues the CHI command DEFERRED_HALT.

4) After 2 500 µT it is verified (UT) that *vPOC!CHIHaltRequest* = true and *vPOC!CHIReadyRequest* = false.

5) 5 000 µT after the CHI command DEFERRED_HALT and latest 3 000 µT before NIT, the UT issues the CHI command DEFERRED_READY.

6) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that *vPOC!CHIHaltRequest* = false and *vPOC!CHIReadyRequest* = true.

7) Within 500 µT and 2 500 µT after cycle end of cycle 8, it is verified (UT) that the IUT is in the *POC:ready* state (*vPOC!State* = *READY*).

8) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!CHIHaltRequest* is set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state (test instance 1).

7.3.5.5.3 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:normal active* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal active* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble II.

The preamble is supplemented by the UT, which enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) It is verified (UT) that *vPOC!Freeze* = false.

2) In the symbol window of cycle 7, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command FREEZE.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *NORMAL_ACTIVE* and *vPOC!Freeze* = true).

4) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions and *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.5.4 Command ALL_SLOTS

— Test purpose

Verify correct behaviour after CHI command ALL_SLOTS in *POC:normal active* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 149.

Table 149 — Modification to basic configurations for POC:normal active – command ALL_SLOTS

Parameter	Modification	
	I	II
pKeySlotOnlyEnabled	false	true

n_{dyn} = *gNumberOfStaticSlots* + 1

— Preamble (setup state)

Preamble II.

Additionally the IUT is configured to transmit frames in static slot 3 and in dynamic slot n_{dyn}.

— Test execution

1) In cycles 7 to 11, the LT simulates its startup frame in slot 2.

2) It is verified (UT) that (I) *vPOC!SlotMode* = *ALL* or (II) *vPOC!SlotMode* = *KEYSLOT* at the beginning of cycle 7.

3) It is verified (LT) that the IUT (I) sends frames in slot 1, slot 3 and slot n_{dyn} or (II) only sends its frame in slot 1 in cycle 7 and cycle 8.

4) In cycle 9, at least 2 500 µT before the start of the NIT, the UT issues the CHI command ALL_SLOTS.

5) After 2 500 µT it is verified (UT) that the slot mode (I) has not changed (*vPOC!SlotMode* = *ALL)* or (II) changed to *vPOC!SlotMode* = *ALL_PENDING*.

6) In cycle 10, 500 µT after cycle start and before cycle end, it is verified (UT) that the slot mode has changed to *vPOC!SlotMode* = *ALL*.

7) In cycle 10, it is verified (LT) that the IUT transmits frames in slot 1, slot 3 and slot n_{dyn}.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames correctly. The slot mode is changed accordingly.

7.3.5.5.5 Command DEFERRED_READY - Command CLEAR_DEFERRED

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_READY in *POC:normal active* state. Verify the correct behaviour after CHI command CLEAR_DEFERRED in *POC:normal active* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal active* to *POC:ready*, initiated by the CHI command DEFERED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 150.

Table 150 — Modification to basic configurations for POC:normal active – command DEFERRED_READY - Command CLEAR_DEFERRED

Parameter	Modification
gMaxWithoutClockCorrectionPassive	2
gMaxWithoutClockCorrectionFatal	3

— **Preamble (setup state)**

Preamble II.

The preamble is supplemented by the UT, which enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would stay in state *POC:normal active* without DEFERRED_READY command:

1) In cycle 7 to 9, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIReadyRequest* = false.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

4) At the beginning of cycle 8, the UT issues the CHI command DEFERRED_READY.

5) 2 500 µT after the CHI command DEFERRED_READY, it is verified (UT) that *vPOC!CHIReadyRequest* = true.

6) In cycle 9, 500 µT after the cycle start, it is verified (UT) that

   — the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*),

   — that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and

   — that *vPOC!CHIReadyRequest* = true.

7) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

Test instance 2 – IUT would leave state *POC:normal active* to *POC:ready* (DEFERRED_READY) without CLEAR_DEFERRED command:

1) In cycles 7 to 9, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIReadyRequest* = false.

3) At the beginning of cycle 8, the UT issues the CHI command DEFERRED_READY.

4) 2 500 µT after the CHI command DEFERRED_READY, it is verified (UT) that *vPOC!CHIReadyRequest* = true.

5) 5 000 µT after the CHI command DEFERRED_READY and latest 3 000 µT before NIT, the UT issues the CHI command CLEAR_DEFERRED.

6) 2 500 µT after the CHI command CLEAR_DEFERRED it is verified (UT) that *vPOC!CHIReadyRequest* = false.

7) In cycle 9, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Test instance 3 – IUT would leave state *POC:normal active* to *POC:normal passive* without DEFERRED_READY command:

1) In cycles 7, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIReadyRequest* = false.

3) In cycles 8 to 11, the LT stops transmission of its startup frame in slot 2.

4) In cycle 11, 500 µT after cycle start and latest 2 000 µT before NIT, the UT issues the CHI command DEFERRED_READY.

5) In cycle 12, 500 µT after cycle start, it is verified (UT)

   — that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*),

   — that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and

   — that *vPOC!CHIReadyRequest* = true.

Test instance 4 – IUT would leave state *POC:normal active* to *POC:ready* (DEFERRED_READY command) without DEFERRED_HALT command:

1) In cycles 7 to 8, the LT simulates its startup frame in slot 2.

2) In cycle 7, it is verified (UT) that *vPOC!CHIHaltRequest* = false and *vPOC!CHIReadyRequest* = false.

3) At the beginning of cycle 8, the UT issues the CHI command DEFERRED_READY.

4) After 2 500 µT it is verified (UT) that *vPOC!CHIHaltRequest* = false and *vPOC!CHIReadyRequest* = true.

5) 5 000 µT after the CHI command DEFERRED_READY and latest 3 000 µT before NIT, the UT issues the CHI command DEFERRED_HALT.

6) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that *vPOC!CHIHaltRequest* = true and *vPOC!CHIReadyRequest* = false.

7) Within 500 µT and 2 500 µT after cycle end of cycle 8, it is verified (UT) that the IUT is in the *POC:halt* state (*vPOC!State* = *HALT*).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions and *vPOC!CHIReadyRequest* is set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state (test instance 1).

7.3.5.6 POC:normal passive

7.3.5.6.1 Command IMMEDIATE_READY and received WUDOP

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:normal passive* state. The IUT shall set the *vColdstartInhibit* flag on return to the *POC:ready* state from the *POC:normal passive* state. Verify read and write access to protocol configuration data in *POC:normal passive* state. Verify de-activated MTS and WUDOP transmission in *POC:normal passive* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal passive* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 151.

Table 151 — Modification to basic configurations for command POC:normal active – IMMEDIATE_READY and received WUDOP

Parameter	Modification	
	I	II
gMaxWithoutClockCorrectionPassive	2	
gMaxWithoutClockCorrectionFatal	3	
pKeySlotOnlyEnabled	false	true

— **Preamble (setup state)**

Preamble II.

The preamble is followed by:

1) In cycle 7, the LT simulates its startup frame in slot 2.

2) Beginning with cycle 8, the LT stops startup frame simulation.

3) In cycle 12, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

4) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) (a) The UT requests the IUT to transmit a WUDOP on *pWakeupChannel*.
(b) The UT requests the IUT to transmit an MTS on *pWakeupChannel*.

2) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) (a) In the symbol window of cycle 12, it is verified (LT), that the IUT does not transmit a WUDOP.
(b) In the symbol window of cycle 12, it is verified (LT), that the IUT does not transmit an MTS.

4) Beginning with cycle 13, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

5) In the symbol window of cycle 13, starting at t0 = symbol window action point, the LT simulates a WUDOP of Low-High-Low-High-Low phases. The low phase has a length of *gdWakeupTxActive* [*gdBit*], the high phase of length of *gdWakeupTxActive* [*gdBit*].

6) At t1 = t0 + 1,5 * *gdWakeupTxActive* * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxActive* * *gdBit* / *pdMicrotick* µT during the first high phase of the LT's WUDOP, the UT issues the CHI command IMMEDIATE_READY.

7) At t1 + 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY* and *vPOC!StartupState* = *UNDEFINED*), that *vPOC!CHIHaltRequest* = false, *POC!CHIReadyRequest* = false and that the slot mode is (I) *vPOC!SlotMode* = *ALL* or (II) *vPOC!SlotMode* = *KEYSLOT*.

8) It is verified (UT) that the wakeup pattern received indicator in the wakeup and startup status is set at t0 + 3 * *gdWakeupTxActive* + *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* + 500 µT.

9) It is verified (UT) that the IUT did not generate any interrupt request.

10) The UT issues the CHI command RUN 2 500 µT after the IUT returned to the *POC:ready* state.

11) 2 500 µT after the CHI command RUN, it is verified (UT) that the IUT has entered the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

Figure 33 depicts the command IMMEDIATE_READY and received WUDOP – command issued in *POC:normal passive*.

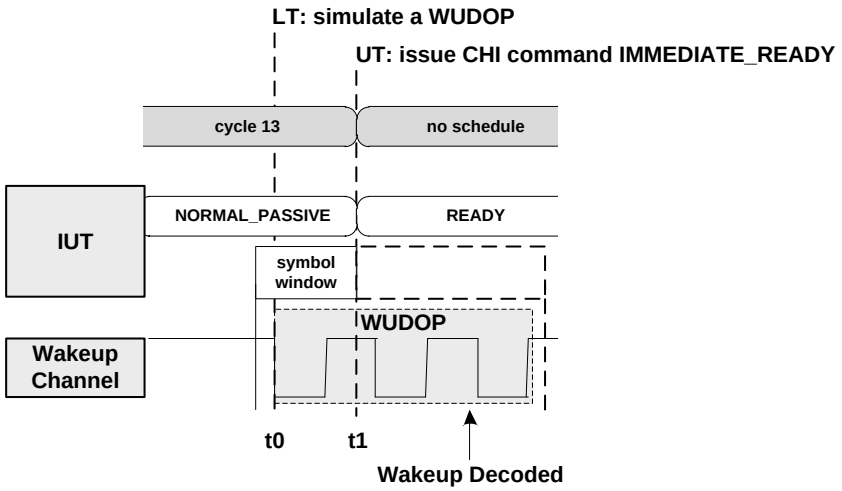


Figure 33 — Command IMMEDIATE_READY and received WUDOP – command issued in POC:normal passive

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions and the slot mode is set accordingly. The IUT enters the integration path after the *vColdstartInhibit* flag was set on the return to the *POC:ready* state from the *POC:normal passive* state. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:normal passive* state. The IUT does not transmit an MTS and a WUDOP while in *POC:normal passive* state. The IUT sets the wakeup pattern received indicator. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.6.2 Command DEFERRED_HALT - Command CLEAR_DEFERRED

— Test purpose

Verify correct behaviour after CHI command HALT in *POC:normal passive* state. Verify the correct behaviour after CHI command CLEAR_DEFERRED in *POC:normal passive* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal passive* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 152.

Table 152 — Modification to basic configurations for POC:normal active – command DEFERRED_HALT - Command CLEAR_DEFERRED

Parameter	Modification
pAllowHaltDueToClock	false
pAllowPassiveToActive	1

— **Preamble (setup state)**

Preamble II.

The preamble is followed by:

1) In cycle 7, the LT simulates its startup frame in slot 2.

2) Beginning with cycle 8, the LT stops startup frame simulation.

3) In cycle 12, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that *vPOC!CHIHaltRequest* = false.

4) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would stay in state *POC:normal passive* without DEFERRED_HALT command:

1) In cycle 13, 500 µT after cycle start and latest 1 000 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT.

2) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that *vPOC!CHIHaltRequest* = true.

3) In cycle 14, 500 µT after cycle start, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIHaltRequest* = true.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

Test instance 2 – IUT would leave state *POC:normal passive* to *POC:halt* (DEFERRED_HALT) without CLEAR_DEFERRED command:

1) At the beginning of cycle 13, the UT issues the CHI command DEFERRED_HALT.

2) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that *vPOC!CHIHaltRequest* = true.

3) 5 000 µT after the CHI command DEFERRED_HALT and latest 3 000 µT before NIT, the UT issues the CHI command CLEAR_DEFERRED.

4) 2 500 µT after the CHI command CLEAR_DEFERRED it is verified (UT) that *vPOC!CHIHaltRequest* = false.

5) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

6) The UT sets the IUT into *POC:halt* state with the command FREEZE.

Test instance 3 – IUT would leave state *POC:normal passive* to *POC:normal active* without DEFERRED_HALT command:

1) In cycle 14, the LT resumes startup frame simulation.

2) In cycle 15, after cycle start and latest 5500 µT before NIT, the UT issues the CHI command DEFERRED_HALT.

3) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that *vPOC!CHIHaltRequest* = true.

4) In cycle 16, 500 µT after cycle start, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIHaltRequest* = true.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!CHIHaltRequest* is set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state (test instance 1).

7.3.5.6.3 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:normal passive* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal passive* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 153.

Table 153 — Modification to basic configurations for POC:normal active – command FREEZE

Parameter	Modification
gMaxWithoutClockCorrectionPassive	2
gMaxWithoutClockCorrectionFatal	3

— **Preamble (setup state)**

Preamble II.

The preamble is followed by:

1) In cycle 7, the LT simulates its startup frame in slot 2.

2) Beginning with cycle 8, the LT stops startup frame simulation.

3) In cycle 12, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

4) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) The UT issues the CHI command FREEZE.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *NORMAL_PASSIVE* and *vPOC!Freeze* = true).

4) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.6.4 Command ALL_SLOTS

— **Test purpose**

Verify correct behaviour after CHI command ALL_SLOTS in *POC:normal passive* state. Veriry correct behaviour on POC state transition from *POC:normal passive* to *POC:normal active* with *pKeySlotOnlyEnabled* = *true*. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal passive* to *POC:normal active* and from *POC:normal active* to *POC:normal passive*).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 154.

Table 154 — Modification to basic configurations for POC:normal active – command ALL_SLOTS

Parameter	Modification
gMaxWithoutClockCorrectionPassive	2
gMaxWithoutClockCorrectionFatal	5
pKeySlotOnlyEnabled	true
pAllowPassiveToActive	1

n_{dyn} = *gNumberOfStaticSlots + 1*

— **Preamble (setup state)**

Preamble II.

The IUT is configured to transmit additionally frames in static slot 3 and in dynamic slot n_{dyn}.

The preamble is followed by:

1) In cycle 7, the LT simulates its startup frame in slot 2.

2) In cycles 7 to 11, it is checked (LT) that the IUT only sends its frame in slot 1.

3) Beginning with cycle 8, the LT stops startup frame simulation.

4) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

5) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

6) In cycle 12, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that *vPOC!SlotMode* = *KEYSLOT*.

— **Test execution**

Test instance 1 – IUT transmits all frames when it is back in *POC:normal active*:

1) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

2) At the beginning of cycle 13, the UT issues the CHI command ALL_SLOTS.

3) After 2 500 µT it is verified (UT) that the slot mode changes to *vPOC!SlotMode* = *ALL_PENDING*.

4) In cycle 14, 500 µT after cycle start and before cycle end, it is verified (UT) that the slot mode changes to *vPOC!SlotMode* = *ALL*.

5) In cycle 14, after the verification and at least 2 500 µT before the start of the NIT, the UT issues the CHI command ALL_SLOTS.

6) After 2 500 µT it is verified (UT) that the slot mode is still *vPOC!SlotMode* = *ALL*.

7) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

8) In cycles 14 to 16, the LT simulates its startup frame in slot 2.

9) In cycle 16, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

10) In cycle 16, it is verified (LT) that the IUT transmits frames in slot 1, slot 3 and slot n_{dyn}.

11) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

Test instance 2 – IUT transmits only its sync frame when it is back in *POC:normal active*:

1) In cycle 14, 500 µT after cycle start and before cycle end, it is verified (UT) that the slot mode is *vPOC!SlotMode = KEYSLOT*.

2) In cycles 14 to 16, the LT simulates its startup frame in slot 2.

3) In cycle 16, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State = NORMAL_ACTIVE*).

4) In cycle 16, it is verified (LT) that the IUT transmits only the frame in slot 1.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames correctly. The slot mode is changed accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:normal active* and *POC:normal passive* state (test instance 1).

7.3.5.6.5 Command DEFERRED_READY - Command CLEAR_DEFERRED

— Test purpose

Verify correct behaviour after CHI command DEFERRED_READY in *POC:normal passive* state. Verify correct behaviour after CHI command CLEAR_DEFERRED in *POC:normal passive* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:normal passive* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 155.

Table 155 — Modification to basic configurations for POC:normal active – command DEFERRED_READY - command CLEAR_DEFERRED

Parameter	Modification	
	I	II
pAllowHaltDueToClock	false	
pAllowPassiveToActive	1	
gMaxWithoutClockCorrectionPassive	1	
gMaxWithoutClockCorrectionFatal	3	
pKeySlotOnlyEnabled	fasle	true

— Preamble (setup state)

Preamble I.

The preamble is followed by:

1) Beginning with cycle 8, the LT stops startup frame simulation.

2) In cycle 10, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that *vPOC!CHIReadyRequest* = false.

3) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would stay in state *POC:normal passive* without DEFERRED_READY command:

1) In cycle 10, 500 µT after cycle start and latest 1 000 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

2) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that *vPOC!CHIReadyRequest* = true and that

   (I) *vPOC!SlotMode* = *ALL* or

   (II) *vPOC!SlotMode* = *KEYSLOT*.

3) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = true and that

   (I) *vPOC!SlotMode* = *ALL* or

   (II) *vPOC!SlotMode* = *KEYSLOT*.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

Test instance 2 – IUT would leave state *POC:normal passive* to *POC:ready* (DEFERRED_READY) without CLEAR_DEFERRED command:

1) In cycle 10, 500 µT after cycle start and latest 1 000 µT after cycle start, the UT issues the CHI command DEFERRED_READY.

2) 2 500 µT after the CHI command DEFERRED_READY, it is verified (UT) that *vPOC!CHIReadyRequest* = true.

3) 5 000 µT after the CHI command DEFERRED_READY and latest 3 000 µT before NIT, the UT issues the CHI command CLEAR_DEFERRED.

4) 2 500 µT after the CHI command CLEAR_DEFERRED it is verified (UT) that *vPOC!CHIReadyRequest* = false.

5) In cycle 11, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that *vPOC!CHIReadyRequest* = false.

Test instance 3 – IUT would leave state *POC:normal passive* to *POC:normal active* without DEFERRED_READY command:

1) Beginning with cycle 12, the LT resumes startup frame simulation.

2) In cycle 13, after cycle start and latest 5500 µT before NIT, the UT issues the CHI command DEFERRED_READY.

3) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that *vPOC!CHIReadyRequest* = true.

4) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = true.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions and *vPOC!CHIReadyRequest* is set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state (test instance 1).

7.3.5.7 POC:wakeup listen

7.3.5.7.1 Command IMMEDIATE_READY

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in POC:wakeup send – state. The IUT shall set the *vColdstartInhibit* flag on return to the *POC:ready* state from the *POC:wakeup listen* state. Verify read and write access to protocol configuration data in *POC:wakeup listen* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup listen* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 156.

Table 156 — Modification to basic configurations for POC:wakeup send – command IMMEDIATE_READY

Parameter	Modification	
	I	II
pKeySlotOnlyEnabled	false	true

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

8) The UT issues the CHI command WAKEUP 2 500 µT after the CHI command ALLOW_COLDSTART.

9) After 2 500 µT it is checked (UT) that the IUT is in a wakeup state (*vPOC!State* = *WAKEUP*).

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

2) 5 000 µT after the CHT command WAKEUP the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY 5 000 µT.

3) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*), that *vPOC!CHIHaltRequest* = false, *vPOC!CHIReadyRequest* = false and that the slot mode is

   (I) *vPOC!SlotMode* = *ALL* or

   (II) *vPOC!SlotMode* = *KEYSLOT*.

4) It is verified (UT) that the IUT did not generate any interrupt request.

5) The UT issues the CHI command RUN 2 * *gdCycle* after the IUT returned to the *POC:ready* state.

6) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions and the slot mode is set accordingly. The IUT enters the integration path after the *vColdstartInhibit* flag was set on the return to the *POC:ready* state from the *POC:wakeup listen* state. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:wakeup listen* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.7.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:wakeup listen* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup listen* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT issues the CHI command WAKEUP.

8) After 2 500 µT it is checked (UT) that the IUT has entered a wakeup state (*vPOC!State* = *WAKEUP*).

9) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command FREEZE right after the verification of the wakeup state.

2) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *WAKEUP* and *vPOC!Freeze* = true).

3) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.7.3 Command DEFERRED_READY and correct wakeup decoding

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_READY in *POC:wakeup listen* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup listen* to *POC:ready*, initiated by the CHI command DEFERRED_READY). The IUT shall accept the simulated WUP on channel *pWakeupChannel*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT issues the CHI command WAKEUP at t0.

8) After 2 500 µT it is checked (UT) that the IUT is in a wakeup state (*vPOC!State* = *WAKEUP*).

9) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources

2) The LT starts to generate a WUP on channel *pWakeupChannel* with a low phase of *gdWakeupTxActive*, an idle phase of length *gdWakeupTxIdle* and a low phase of length *gdWakeupTxActive* earliest 2 500 µT and latest 0,95 * *pdListenTimeout* µT after the CHI command WAKEUP at t1.

3) (a) The UT issues the CHI command DEFERRED_READY 0,5 * *pdListenTimeout* µT after the CHI command WAKEUP.
(b) At t2 = t1 + (*gdWakeupTxActive* + 0,5 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* µT during the first idle phase of the LT's WUP, the UT issues the CHI command DEFERRED_READY.
(c) At t2 = t1 + (*gdWakeupTxActive* + 0,5 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* µT during the first idle phase of the LT's WUP, the UT issues the CHI command IMMEDIATE_READY.

4) 2 500 µT after the CHI command DEFERRED_READY or IMMEDIATE_READY, it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that *vPOC!CHIHaltRequest* = false and that *vPOC!CHIReadyRequest* = false.

5) It is verified (UT) that the wakeup pattern received indicator for channel *pWakeupChannel* in the wakeup and startup status is set and, for dual channel test execution, that the wakeup pattern received indicator for the non-wakeup channel in the wakeup and startup status is reset after a period of 0,95 * *pdListenTimeout* + (2 * *gdWakeupTxActive* + 2 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT past the CHI command WAKEUP.

6) (a,b) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.
(c) It is verified (UT) that the IUT did not generate any interrupt request.

Figure 34 depicts the command DEFERRED_READY and correct wakeup decoding – *pWakeupChannel* = A.

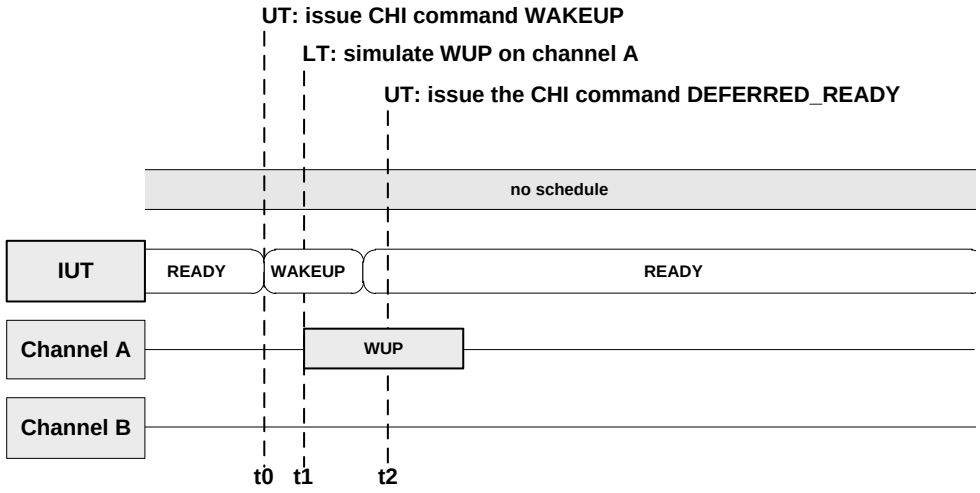


Figure 34 — Command DEFERRED_READY and correct wakeup decoding – pWakeupChannel = A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions and the wakeup pattern received indicator is set accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.7.4 Command DEFERRED_HALT

— Test purpose

Verify correct behaviour after CHI command DEFERRED_HALT in POC:wakeup listen state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup listen* to *POC:halt*, initiated by the CHI command DEFERED_HALT).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command CONFIG_COMPLETE.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

7) The UT issues the CHI command WAKEUP.

8) After 2 500 µT it is checked (UT) that the IUT has entered a wakeup state (*vPOC!State* = *WAKEUP*).

9) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT 0,5 * *pdListenTimeout* µT after the CHI command WAKEUP. After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state(*vPOC!State* = *HALT*).

2) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.8 POC:halt

7.3.5.8.1 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:halt* state. Verify the correct behaviour of IUT regarding write access to the protocol configuration data in *POC:halt* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command DEFERRED_HALT.

— **Test execution**

1) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT)* and that *vPOC!Freeze* = false.

2) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT issues the CHI command FREEZE.

4) After 2 500 µT it is verified (UT) that the IUT is still in the *POC:halt* state (*vPOC!State* = *HALT)* and *vPOC!Freeze* = true.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!Freeze* is set accordingly. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:halt* state.

7.3.5.8.2 Command DEFAULT_CONFIG

— **Test purpose**

The IUT shall not increment the macrotick counter and the cycle counter in *POC:halt* state. Verify correct behaviour after CHI command DEFAULT_CONFIG in *POC:halt* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT issues the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the IUT with the given configuration.

5) The UT issues the CHI command DEFERRED_HALT.

6) After 2 500 µT it is checked (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT)*.

— **Test execution**

1) The LT simulate startup frames with frame ID 2 and cycle count 2 and 3.

2) It is verified (UT) that the macrotick counter *vMacrotick* and the cycle counter *vCycleCounter* stay constant, *gdStaticSlot*, 2 * *gdStaticSlot* and 2 * *gdCycle* after the end of the LT's startup frames.

3) The UT issues the CHI command DEFAULT_CONFIG.

4) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:default config* state

   — (*vPOC!State* = *DEFAULT_CONFIG*) and that

   — *vPOC!Freeze* = false,

   — *vPOC!CHIHaltRequest* = false,

   — *vPOC!CHIReadyRequest* = false,

   — *vPOC! coldstartNoise* = false,

   — *vPOC!SlotMode* = *KEYSLOT*,

   — *vPOC!ErrorMode* = *ACTIVE*,

   — *vPOC!WakeupStatus* = *UNDEFINED,*

   — *vPOC!StartupState* = *UNDEFINED,*

   — *vClockCorrectionFailed* = 0 and

   — *vAllowPassiveToActive* = 0

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions and *vPOC!Freeze*, *vPOC!CHIHaltRequest*, *vPOC!CHIReadyRequest , vPOC!ColdstartNoise*, *vPOC!SlotMode*, *vPOC!ErrorMode, vPOC!WakeupStatus, POC!StartupState*, *vClockCorrectionFailed* and *vAllowPassiveToActive* are set accordingly. The macrotick counter and the cycle counter are not incremented during *POC:halt* state.

7.3.5.9 POC:wakeup send

7.3.5.9.1 Command IMMEDIATE_READY

— **Test purpose**

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:wakeup send* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:wakeup send* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup send* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 157.

Table 157 — Modification to basic configurations for POC:wakeup send – command IMMEDIATE_READY

Parameter	Modification
pWakeupPattern	24

Table 158 defines the two different buffer configuration data sets for *POC:wakeup send* – command IMMEDIATE_READY.

Table 158 — Two different buffer configuration data sets for POC:wakeup send – command IMMEDIATE_READY

Buffer Configuration Data Class 1	Set 1	Set 2
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

   (b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

   (c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) It is verified (UT), *pdListenTimeout* + 500 µT after the CHI command WAKEUP, that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

4) The UT configures MB with the buffer configuration data set 2.

5) By reading the configuration data of MB, it is verified (UT) that

   (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

   (b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

   (c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

6) During the 14th low phase of the IUT's WUP, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command IMMEDIATE_READY.

7) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) and that the wakeup status is undefined (*vPOC!WakeupStatus* = *UNDEFINED*) 2 500 µT after the CHI command IMMEDIATE_READY. It is also verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:wakeup send* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.9.2 Command FREEZE

— Test purpose

Verify correct behaviour after CHI command FREEZE in *POC:wakeup send* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup send* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) During the first idle phase of the IUT's WUP, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command FREEZE.

4) It is verified (UT) that the IUT is in *POC:halt* state (*vPOC!State* = *WAKEUP* and *vPOC!Freeze* = true) and that the wakeup status is undefined (*vPOC!WakeupStatus* = *UNDEFINED)* 2 500 µT after the CHI command FREEZE. It is verified (UT) that the IUT did not generate any interrupt request.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.9.3 Command DEFERRED_READY – Command CLEAR_DEFERRED

— Test purpose

Verify correct behaviour after CHI command DEFERRED_READY in *POC:wakeup send* state. Verify correct behaviour of the CHI command CLEAR_DEFERRED in *POC:wakeup send* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup send* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

Test instance 1 – DEFERRED_READY command during IUT's WUP transmission:

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIReadyRequest* = false, 2 500 µT after the CHI command WAKEUP.

3) During the first idle phase of the IUT's WUP the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

4) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIReadyRequest* = true, 2 500 µT after the CHI command DEFERRED_READY.

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!CHIReadyRequest* = true and *vPOC!WakeupStatus* = *TRANSMITTED* (2 500 + *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick*) µT after the CHI command WAKEUP.

6) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

Test instance 2 – DEFERRED_READY and CLEAR_DEFERRED command during IUT's WUP transmission:

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIReadyRequest* = false, 2 500 µT after the CHI command WAKEUP.

3) During the first idle phase of the IUT's WUP the UT issues the CHI command DEFERRED_READY.

4) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIReadyRequest* = true and issues the command CLEAR_DEFERRED, 2 500 µT after the CHI command DEFERRED_READY.

5) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIReadyRequest* = false, 2 500 µT after the CHI command CLEAR_DEFERRED.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* and *vPOC!CHIReadyRequest* = false (2 500 + *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick*) µT after the CHI command WAKEUP.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.9.4 Command DEFERRED_HALT – Command CLEAR_DEFERRED

— Test purpose

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:wakeup send* state. Verify correct behaviour of the CHI command CLEAR_DEFERRED in *POC:wakeup send* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup send* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – DEFERRED_HALT and CLEAR_DEFERRED command during *POC:wakeup send*:

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIHaltRequest* = false, 2 500 µT after the CHI command WAKEUP.

3) During the first idle phase of the IUT's WUP, the UT issues the command DEFERRED_HALT.

4) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIHaltRequest* = true and issues the command CLEAR_DEFERRED, 2 500 µT after the CHI command DEFERRED_HALT.

5) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) and *vPOC!CHIHaltRequest* = false, 2 500 µT after the CHI command CLEAR_DEFERRED.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* and *vPOC!CHIHaltRequest* = false (2 500 + *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick*) µT after the CHI command WAKEUP.

7) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

Test instance 2 – DEFERRED_HALT command during *POC:wakeup send*:

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) During the first idle phase of the IUT's WUP the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command DEFERRED_HALT.

4) It is verified (UT) that the IUT is in state *POC:halt* (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = true and *vPOC!WakeupStatus* = *TRANSMITTED* (2 500 + *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick*) µT after the CHI command WAKEUP.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.10 POC:wakeup detect

7.3.5.10.1 Command IMMEDIATE_READY

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:wakeup detect* state.

Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:wakeup detect* state and the reset of *vPOC!WakeupStatus* after CHI command WAKEUP. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup detect* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

Table 159 defines the two different buffer configuration data sets for *POC:wakeup detect* – command IMMEDIATE_READY.

Table 159 — Two different buffer configuration data sets for POC:wakeup detect – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

(b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

(c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP (*vPOC!State* = *WAKEUP* and *vPOC!WakeupStatus* = *UNDEFINED*). During the first idle phase of the IUT's WUP, the LT generates a low phase of length *cdWakeupMaxCollision* + 1 *gdBit* on the wakeup channel *pWakeupChannel* (t1 is the start of the high phase).

2) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT configures MB with the buffer configuration data set 2.

4) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

5) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY at (t1 + *pdListenTimeout* / 2) µT.

6) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) and that the wakeup status *vPOC!WakeupStatus* = *COLLISION_UNKNOWN* 2 500 µT after the CHI command IMMEDIATE_READY. It is also verified (UT) that the IUT did not generate any interrupt request.

7) The UT triggers the wakeup process with the CHI command WAKEUP.

8) It is verified (UT) that the IUT is in a wakeup state (*vPOC!State* = *WAKEUP*) and that the wakeup status *vPOC!WakeupStatus* = *UNDEFINED* 2 500 µT after the CHI command WAKEUP.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions and resets *vPOC!WakeupStatus* to *UNDEFINED*. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:wakeup detect* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.10.2 Command FREEZE

— Test purpose

Verify correct behaviour after CHI command FREEZE in *POC:wakeup detect* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup detect* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP (*vPOC!State* = *WAKEUP* and *vPOC!WakeupStatus* = *UNDEFINED*).

2) During the first idle phase of the IUT's WUP, the LT generates a low phase of length *cdWakeupMaxCollision* + 1 *gdBit* on the wakeup channel *pWakeupChannel* (t1 is the start of the high phase).

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command FREEZE at (t1 + 2 500) µT and latest at (t1 + *pdListenTimeout* / 2) µT.

4) It is verified (UT) 2 500 µT after the CHI command FREEZE that the IUT is in *POC:halt* state (*vPOC!State* = *WAKEUP* and *vPOC!Freeze* = true) and that the wakeup status *vPOC!WakeupStatus* = *COLLISION_UNKNOWN*.

5) It is verified (UT) that the IUT did not generate any interrupt request.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.10.3 Command DEFERRED_READY

— Test purpose

Verify correct behaviour after CHI command DEFERRED_READY in *POC:wakeup detect* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup detect* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP (*vPOC!State* = *WAKEUP* and *vPOC!WakeupStatus* = *UNDEFINED*).

2) During the first idle phase of the IUT's WUP, the LT generates a low phase of length *cdWakeupMaxCollision* + 1 *gdBit* on the wakeup channel *pWakeupChannel* (t1 is the start of the high phase).

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY at (t1 + 2 500) µT and latest at (t1 + *pdListenTimeout* / 2) µT.

4) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) and that the wakeup status
vPOC!WakeupStatus = *COLLISION_UNKNOWN* 2 500 µT after the CHI command DEFERRED_READY.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.10.4 Command DEFERRED_HALT

— Test purpose

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:wakeup detect* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:wakeup detect* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP (*vPOC!State* = *WAKEUP* and *vPOC!WakeupStatus* = *UNDEFINED*).

2) During the first idle phase of the IUT's WUP, the LT generates a low phase of length *cdWakeupMaxCollision* + 1 *gdBit* on the wakeup channel *pWakeupChannel* (t1 is the start of the high phase).

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT at (t1 + 2 500) µT and latest at (t1 + *pdListenTimeout* / 2) µT.

4) It is verified (UT) 2 500 µT after the CHI command DEFERRED_HALT that the IUT is in *POC:halt* state (*vPOC!State* = *HALT)* and that the wakeup status *vPOC!WakeupStatus* = *COLLISION_UNKNOWN*.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.11 POC:coldstart listen

7.3.5.11.1 Command IMMEDIATE_READY

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:coldstart listen* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:coldstart listen* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart listen* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

Table 160 defines the two different buffer configuration data sets for *POC:coldstart listen* – command IMMEDIATE_READY.

Table 160 — Two different buffer configuration data sets for POC:coldstart listen – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— Preamble (setup state)

1) The UT resets the IUT (vPOC!State = DEFAULT_CONFIG).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

5) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

(b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

(c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

6) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

7) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

8) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) 2 500 µT after the CHI command ALLOW_COLDSTART, the UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

4) The UT configures MB with the buffer configuration data set 2.

5) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

6) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command IMMEDIATE_READY (*pdListenTimeout* – 2 500) µT after the CHI command RUN.

7) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) 2 500 µT after the CHI command IMMEDIATE_READY.

8) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:coldstart listen* state. Write access to buffer configuration data is

as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.11.2 Command FREEZE

— Test purpose

Verify correct behaviour after CHI command FREEZE in *POC:coldstart listen* state. It is also verified that no interrupt is indicated after the state transition to the *POC:halt* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart listen* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

6) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) 2 500 µT after the CHI command ALLOW_COLDSTART, the UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command FREEZE 3 000 µT after the CHI command RUN.

4) It is verified (UT) that the IUT is in *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *COLDSTART_LISTEN* and *vPOC!Freeze* = true) 2 500 µT after the CHI command FREEZE.

5) It is verified (UT) that the IUT did not generate any interrupt request.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.11.3 Command DEFERRED_READY

— Test purpose

Verify correct behaviour after CHI command DEFERRED_READY in *POC:coldstart listen* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart listen* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

6) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) 2 500 µT after the CHI command ALLOW_COLDSTART, the UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command DEFERRED_READY 3 000 µT after the CHI command RUN.

4) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) 2 500 µT after the CHI command DEFERRED_READY.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.11.4 Command DEFERRED_HALT

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:coldstart listen* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart listen* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

6) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) 2 500 µT after the CHI command ALLOW_COLDSTART, the UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command DEFERRED_HALT 3 000 µT after the CHI command RUN.

4) It is verified (UT) that the IUT is in *POC:halt* state (*vPOC!State* = *HALT*) 2 500 µT after the CHI command DEFERRED_HALT.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.12 POC:coldstart collision resolution

7.3.5.12.1 Command IMMEDIATE_READY

— **Test purpose**

Verify correct behaviour after the command IMMEDIATE_READY in *POC:coldstart collision resolution* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:coldstart collision resolution* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart collision resolution* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

Table 161 defines the two different buffer configuration data sets for *POC:coldstart collision resolution* – command IMMEDIATE_READY.

Table 161 — Two different buffer configuration data sets for POC:coldstart collision resolution – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

(a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

(b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

(c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

7) The UT initiates the startup procedure with the CHI command RUN 2 500 µT after the CHI command ALLOW_COLDSTART.

8) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

9) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) It is verified (LT) that the IUT sends a CAS starting within [*pdListenTimeout*; *pdListenTimeout* + 2 000] µT after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) (2 500 + *pdListenTimeout*) µT after the CHI command RUN.

3) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data in *POC:coldstart collision resolution* state (2 500 + *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + *pdListenTimeout*) µT after the CHI command RUN.

4) The UT configures MB with the buffer configuration data set 2.

5) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

6) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command IMMEDIATE_READY after protocol and buffer configuration check.

7) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) 2 500 µT after the CHI command IMMEDIATE_READY.

8) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:coldstart collision resolution* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.12.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after the command FREEZE in *POC:coldstart collision resolution* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart collision resolution* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 500 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) It is verified (LT) that the IUT sends a CAS starting within [*pdListenTimeout*; *pdListenTimeout* + 2 000] µT after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) (2 500 + *pdListenTimeout*) µT after the CHI command RUN

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command FREEZE (2 500 + *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + *pdListenTimeout*) µT after the CHI command RUN.

4) It is verified (UT) that the IUT is in *POC:halt* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* and *vPOC!Freeze* = true) 2 500 µT after the CHI command FREEZE.

5) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.12.3 Command DEFERRED_READY - Command CLEAR_DEFERRED

— **Test purpose**

Verify correct behaviour after the command DEFERRED_READY and CLEAR_DEFERRED in *POC:coldstart collision resolution* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart collision resolution* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

4) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

5) The UT initiates the startup procedure with the CHI command RUN 2 500 µT after the CHI command ALLOW_COLDSTART.

6) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

7) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would leave state *POC:coldstart collision resolution* to *POC:ready* (DEFERRED_READY) without CLEAR_DEFERRED command:

1) It is verified (LT) that the IUT sends a CAS starting within [*pdListenTimeout*; *pdListenTimeout* + 2 000] µT after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* (*vPOC!State* = *STARTUP* , *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* and *vPOC!CHIReadyRequest* = false) (2 500 + *pdListenTimeout*) µT after the CHI command RUN

3) In cycle 0, 500 µT after cycle start, the UT issues the CHI command DEFERRED_READY.

4) It is verified (UT) that the IUT has set *vPOC!CHIReadyRequest* = true 2 500 µT after the CHI command DEFERRED_READY.

5) In cycle 0, 5 500 µT after cycle start, the UT issues the CHI command CLEAR_DEFERRED.

6) It is verified (UT) that the IUT has reset *vPOC!CHIReadyRequest* = false 2 500 µT after the CHI command CLEAR_DEFERRED.

7) In cycle 1, 500 µT after cycle start it is verified (UT) that the IUT is still in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* and *vPOC!CHIReadyRequest* = false).

Test instance 2 – IUT would stay in state *POC:coldstart collision resolution* without DEFERRED_READY command:

1) It is verified (LT) that the IUT sends a CAS starting within [*pdListenTimeout*; *pdListenTimeout* + 2 000] µT after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* (*vPOC!State* = *STARTUP* , *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* and *vPOC!CHIReadyRequest* = false) (2 500 + *pdListenTimeout*) µT after the CHI command RUN.

3) In cycle 0, 3 000 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

4) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that the IUT set *vPOC!CHIReadyRequest* = true. The LT starts to generate

   (a) nothing.

   (b, c) a header of a sync frame with slot ID 2 (null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the payload length is 1, the cycle count 1) plus the BSS preceding the frames payload section on (b) channel A and (c) channel B.

   (d, e) a CAS of length *cdCASRxLowMin* on (d) channel A and (e) channel B.

5) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*)

   (a) (500 + *pMicroPerCycle*) µT

   (b, c) (500 + 2 500 + (*gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick*) µT

   (d, e) (500 + 2 500 + *cdCASRxLowMin* * *gdBit* / *pdMicrotick*) µT

after the CHI command DEFERRED_READY.

6) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state (test instance 2).

7.3.5.12.4 Command DEFERRED_HALT - Command CLEAR_DEFERRED

— **Test purpose**

Verify correct behaviour after the command DEFERRED_HALT and CLEAR_DEFERRED in *POC:coldstart collision resolution* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart collision resolution* check to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

4) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 500 µT after the CHI command CONFIG_COMPLETE.

5) The UT initiates the startup procedure with the CHI command RUN 2 500 µT after the CHI command ALLOW_COLDSTART.

6) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

7) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would leave state *POC:coldstart collision resolution* to *POC:halt* (DEFERRED_HALT) without CLEAR_DEFERRED command:

1) It is verified (LT) that the IUT sends a CAS starting within [*pdListenTimeout*; *pdListenTimeout* + 2 000] µT after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* (*vPOC!State* = *STARTUP* , *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* and *vPOC!CHIHaltRequest* = false) (2 500 + *pdListenTimeout*) µT after the CHI command RUN

3) In cycle 0, 3 000 µT after cycle start, the UT issues the CHI command DEFERRED_HALT.

4) It is verified (UT) that the IUT has set *vPOC!CHIHaltRequest* = true 2 500 µT after the CHI command DEFERRED_HALT.

5) In cycle 0, 5500 µT after cycle start, the UT issues the CHI command CLEAR_DEFERRED.

6) It is verified (UT) that the IUT has reset *vPOC!CHIHaltRequest* = false 2 500 µT after the CHI command CLEAR_DEFERRED.

7) In cycle 1, 500 µT after cycle start it is verified (UT) that the IUT is still in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* and *vPOC!CHIHaltRequest* = false).

8) The UT sets the IUT into *POC:halt* state with the command FREEZE.

Test instance 2 – IUT would stay in state *POC:coldstart collision resolution* without DEFERRED_HALT command:

1) It is verified (LT) that the IUT sends a CAS starting within [*pdListenTimeout*; *pdListenTimeout* + 2 000] µT after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* (*vPOC!State* = *STARTUP* , *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* and *vPOC!CHIHaltRequest* = false) (2 500 + *pdListenTimeout*) µT after the CHI command RUN

3) In cycle 0, 2 500 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT.

4) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that the IUT set *vPOC!CHIHaltRequest* = true. The LT starts to generate

   (a) nothing.

   (b, c) a header of a sync frame with slot ID 2 (null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the payload length is 1, the cycle count 1) plus the BSS preceding the frames payload section on (b) channel A and (c) channel B.

   (d, e) a CAS of length *cdCASRxLowMin* on (d) channel A and (e) channel B.

5) It is verified (UT) that the IUT is in *POC:halt* state (*vPOC!State* = *HALT*)

   (a) (500 + *pMicroPerCycle*) µT

   (b, c) (500 + 2 500 + (*gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick*) µT

   (d, e) (500 + 2 500 + *cdCASRxLowMin* * *gdBit* / *pdMicrotick*) µT

   after the CHI command DEFERRED_HALT.

6) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state. (test instance 2)

7.3.5.13 POC:coldstart consistency check

7.3.5.13.1 Command IMMEDIATE_READY

— **Test purpose**

Verify correct behaviour after the command IMMEDIATE_READY in *POC:coldstart consistency check* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:coldstart consistency check* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart consistency check* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

Table 162 defines the two different buffer configuration data sets for *POC:coldstart consistency check* – command IMMEDIATE_READY.

Table 162 — Two different buffer configuration data sets for POC:coldstart consistency check – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

   (b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

   (c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

7) The UT initiates the startup procedure with the CHI command RUN.

8) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

9) It is checked (LT) that the IUT sends a CAS.

10) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

11) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

12) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

2) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 4.

3) In cycle 4, the LT simulates a startup frame in slot 2.

4) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

5) The UT configures MB with the buffer configuration data set 2.

6) By reading the configuration data of MB, it is verified (UT) that

   (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

   (b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

7) In the NIT of cycle 4 the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command IMMEDIATE_READY after protocol and buffer configuration check.

8) It is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) 2 500 µT after the CHI command IMMEDIATE_READY.

9) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:coldstart consistency check* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.13.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after the command FREEZE in *POC:coldstart consistency check* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart consistency check* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP, vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK* and *vPOC!Freeze* = false).

2) In cycle 4, the LT simulates a startup frame in slot 2.

3) In slot 3 of cycle 4, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command FREEZE.

4) It is verified (UT) that the IUT is in *POC:halt* state (*vPOC!State* = *STARTUP, vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK* and *vPOC!Freeze* = true*)* 2 500 µT after the CHI command FREEZE.

5) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.13.3 Command DEFERRED_READY - Command CLEAR_DEFERRED

— **Test purpose**

Verify correct behaviour after the command DEFERRED_READY and CLEAR_DEFERRED in *POC:coldstart consistency check* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart consistency check* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would leave state *POC:coldstart consistency check* to *POC:ready* (DEFERRED_READY) without CLEAR_DEFERRED command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*) and *vPOC!CHIReadyRequest* = false.

2) It is verified (LT) that the IUT sends startup frames in slot 1 of cycle 4 and 5.

3) In cycles 4 and 5, the LT simulates startup frames in slot 2.

4) In cycle 4, 1 000 µT after cycle start, the UT issues the command DEFERRED_READY.

5) In cycle 4, 2 500 µT after the CHI command DEFERRED_READY, it is verified (UT) that the IUT has set *vPOC!CHIReadyRequest* = true.

6) In cycle 4, 3 000 µT after the CHI command DEFERRED_READY, the UT issues the command CLEAR_DEFERRED.

7) In cycle 4, 2 500 µT after the CHI command CLEAR_DEFERRED, it is verified (UT) that the IUT has reset *vPOC!CHIReadyRequest* = false.

8) In cycle 5, 500 µT after cycle start, it is verified (UT) that the IUT is still in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*) and *vPOC!CHIReadyRequest* = false.

Test instance 2 – IUT would stay in state *POC:coldstart consistency check* without DEFERRED_READY command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*) and *vPOC!CHIReadyRequest* = false.

2) It is verified (LT) that the IUT sends startup frames in slot 1 of cycle 4 and 5.

3) In cycles 4 and 5, the LT simulates startup frames in slot 2.

4) In cycle 4, 1 000 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command DEFERRED_READY.

5) In cycle 4, 2 500 µT after the CHI command DEFERRED_READY, it is verified (UT) that the IUT has set *vPOC!CHIReadyRequest* = true.

6) In cycle 5, 500 µT after cycle start, it is verified (UT) that the IUT is in *POC:ready* state (*vPOC!State* = *READY*) and *vPOC!CHIReadyRequest* = true.

7) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state (test instance 2).

7.3.5.13.4 Command DEFERRED_HALT - Command CLEAR_DEFERRED

— **Test purpose**

Verify correct behaviour after the command DEFERRED_HALT and CLEAR_DEFERRED in *POC:coldstart consistency check* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart consistency check* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would leave state *POC:coldstart consistency check* to *POC:halt* (DEFERRED_HALT) without CLEAR_DEFERRED command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*) and *vPOC!CHIHaltRequest* = false.

2) It is verified (LT) that the IUT sends startup frames in slot 1 of cycle 4 and 5.

3) In cycles 4 and 5, the LT simulates startup frames in slot 2.

4) In cycle 4, 1 000 µT after cycle start, the UT issues the command DEFERRED_HALT.

5) In cycle 4, 2 500 µT after the CHI command DEFERRED_HALT, it is verified (UT) that the IUT has set *vPOC!CHIHaltRequest* = true.

6) In cycle 4, 3 000 µT after the CHI command DEFERRED_HALT, the UT issues the command CLEAR_DEFERRED.

7) In cycle 4, 2 500 µT after the CHI command CLEAR_DEFERRED, it is verified (UT) that the IUT has reset *vPOC!CHIHaltRequest* = false.

8) In cycle 5, 500 µT after cycle start, it is verified (UT) that that the IUT is still in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*) and *vPOC!CHIHaltRequest* = false.

9) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

Test instance 2 – IUT would stay in state *POC:coldstart consistency check* without DEFERRED_HALT command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*) and *vPOC!CHIHaltRequest* = false.

2) It is verified (LT) that the IUT sends startup frames in slot 1 of cycle 4 and 5.

3) In cycles 4 and 5, the LT simulates startup frames in slot 2.

4) In cycle 4, 1 000 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command DEFERRED_HALT.

5) In cycle 4, 2 500 µT after the CHI command DEFERRED_HALT, it is verified (UT) that the IUT has set *vPOC!CHIHaltRequest* = true.

6) In cycle 5, 500 µT after cycle start, it is verified (UT) that the IUT is in *POC:halt* state (*vPOC!State* = *HALT*) and *vPOC!CHIHaltRequest* = true.

7) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the expected state transitions. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state (test instance 2).

7.3.5.14 POC:coldstart gap

7.3.5.14.1 Command IMMEDIATE_READY

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:coldstart gap* state. Verify read and write access to configuration data (including buffer configuration) in *POC:coldstart gap* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart gap* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

Table 163 defines the two different buffer configuration data sets for *POC:coldstart gap* – command IMMEDIATE_READY.

Table 163 — Two different buffer configuration data sets for POC:coldstart gap – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

   (b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

   (c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

7) The UT initiates the startup procedure with the CHI command RUN.

8) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

9) It is checked (LT) that the IUT sends a CAS.

10) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

11) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

12) In cycle 4, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

13) It is checked (LT) that the IUT sends a startup frame in slot 1 of cycle 4.

14) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 5, 500 µT after cycle start and latest 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

2) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT configures MB with the buffer configuration data set 2.

4) By reading the configuration data of MB, it is verified (UT) that

   (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

5) In cycle 5, 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY.

6) 2 500 µT after the CHI command IMMEDIATE_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State = READY)* and that the startup state is undefined *(vPOC!StartupState = UNDEFINED*).

7) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions accordingly. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:coldstart gap* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.14.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:coldstart gap* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart gap* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

11) In cycle 4, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

12) It is checked (LT) that the IUT sends a startup frame in slot 1 of cycle 4.

13) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 5, 500 µT after cycle start and latest 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

2) In cycle 5, earliest 3 000 µT before NIT and latest 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.and issues the CHI command FREEZE.

3) 2 500 µT after the CHI command FREEZE it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *COLDSTART_GAP* and *vPOC!Freeze* = true).

4) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* and the *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.14.3 Command DEFERRED_READY

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_READY in *POC:coldstart gap* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart gap* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

11) In cycle 4, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

12) It is checked (LT) that the IUT sends a startup frame in slot 1 of cycle 4.

13) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 5, 500 µT after cycle start and latest 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

2) In cycle 5, earliest 3 000 µT before NIT and latest 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

3) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the state transition to *POC:ready* accordingly. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.14.4 Command DEFERRED_HALT

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:coldstart gap* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration coldstart* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 2 of cycles 0 to 3.

11) In cycle 4, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

12) It is checked (LT) that the IUT sends a startup frame in slot 2 of cycle 4.

13) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) In cycle 5, 500 µT after cycle start and latest 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

2) In cycle 5, earliest 3 000 µT before NIT and latest 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT.

3) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT performs the state transition to *POC:halt* accordingly and the *vPOC!CHIHaltRequest* is reset. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.15 POC:initialize schedule

7.3.5.15.1 Command IMMEDIATE_READY

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:initialize schedule* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:initialize schedule* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:initialize schedule* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

Table 164 defines the two different buffer configuration data sets for *POC:initialize schedule* – command IMMEDIATE_READY.

Table 164 — Two different buffer configuration data sets for POC:initialize schedule – command IMMEDIATE_READY

Buffer Configuration Data Class 1	Set 1	Set 2
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

   (b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

   (c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates startup frames in slot 2 of cycles 0 to 2.

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) In cycle 0, 500 µT after the start of slot 3 and 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

2) In cycle 0, 500 µT after the start of slot 3 it is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT configures MB with the buffer configuration data set 2.

4) By reading the configuration data of MB, it is verified (UT) that

   (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

   (b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

   (c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

5) In cycle 0, 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY.

6) 2 500 µT after the CHI command IMMEDIATE_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY)* and that the startup state is undefined *(vPOC!StartupState* = *UNDEFINED*).

7) It is verified (UT) that the IUT did not generate any interrupt request.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions accordingly. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:coldstart listen* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.15.2 Command FREEZE

— Test purpose

Verify correct behaviour after CHI command FREEZE in *POC:initialize schedule* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:initialize schedule* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 2.

9) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 0, 500 µT after the start of slot 3 and 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

2) In cycle 0, earliest 3 000 µT before NIT and latest 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command FREEZE.

3) 2 500 µT after the CHI command FREEZE it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *INITIALIZE_SCHEDULE* and *vPOC!Freeze* = true). It is also verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* and the *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.15.3 Command ALLOW_COLDSTART

— **Test purpose**

Verify correct behaviour after CHI command ALLOW_COLDSTART in *POC:initialize schedule* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates a startup frame in slot 2 of cycle 0.

— **Test execution**

1) In cycle 0, 500 µT after the start of slot 3 and 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

2) In cycle 0, 2 000 µT before NIT, the UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

3) In cycle 1, (500 + *pRateCorrectionOut*) µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT has entered the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters the *POC:coldstart listen* startup state.

7.3.5.15.4 Command DEFERRED_READY

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_READY in *POC:initialize schedule* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:initialize schedule* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 2.

9) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

2) In cycle 0, 500 µT after the start of slot 3 and 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

3) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the state transition to *POC:ready* accordingly and the *vPOC!CHIReadyRequest* is reset. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.15.5 Command DEFERRED_HALT

— Test purpose

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:initialize schedule* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:initialize schedule* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 2.

9) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) In cycle 0, 500 µT after the start of slot 3 and 3 000 µT before NIT, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

2) In cycle 0, earliest 3 000 µT before NIT and latest 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT.

3) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* accordingly and the *vPOC!CHIHaltRequest* is reset. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.16 POC:integration coldstart check

7.3.5.16.1 Command IMMEDIATE_READY

— **Test purpose**

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:integration coldstart check* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:integration coldstart check* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration coldstart check* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

Table 165 defines the Two different buffer configuration data sets for *POC:integration coldstart check* – command IMMEDIATE_READY.

Table 165 — Two different buffer configuration data sets for POC:integration coldstart check – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

(a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

(b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

(c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates startup frames in slot 2 of cycles 0 and 1.

10) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*)

2) In cycle 1, 500 µT after the start of slot 3, it is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT configures MB with the buffer configuration data set 2.

4) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

5) In cycle 2, 500 µT after cycle start and 2 500 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY.

6) 2 500 µT after the CHI command IMMEDIATE_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY)* and that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*).

7) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions accordingly. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:integration coldstart check* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.16.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:integration coldstart check* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration coldstart check* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 and 1.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

2) In cycle 2, 500 µT after cycle start and 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command FREEZE.

3) 2 500 µT after the CHI command FREEZE it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK* and *vPOC!Freeze* = true).

4) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* and the *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.16.3 Command DEFERRED_READY

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_READY in *POC:integration coldstart check* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration coldstart check* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 and 1.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT , it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

2) In cycle 2, 500 µT after cycle start and 2 500 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

3) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the state transition to *POC:ready* accordingly and the *vPOC!CHIReadyRequest* is not set. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.16.4 Command DEFERRED_HALT

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:integration coldstart check* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration coldstart check* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 and 1.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

2) In cycle 2, 500 µT after cycle start and 2 500 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT.

3) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* accordingly and the *vPOC!CHIHaltRequest* is not set. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.16.5 Command ALLOW_COLDSTART

— **Test purpose**

Verify correct behaviour after CHI command ALLOW_COLDSTART in *POC:integration coldstart check* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 and 1.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

2) In cycle 2, 500 µT after cycle start and 2 000 µT before NIT, the UT issues the CHI command ALLOW_COLDSTART.

3) In cycle 3, 500 µT after cycle start, it is verified (UT) that the IUT has entered the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters the *POC:coldstart listen* startup state.

7.3.5.17 POC:coldstart join

7.3.5.17.1 Command IMMEDIATE_READY

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:coldstart join* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:coldstart join* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart join* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

Table 166 defines the two different buffer configuration data sets for *POC:coldstart join* – command IMMEDIATE_READY.

Table 166 — Two different buffer configuration data sets for POC:coldstart join – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

   (b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

(c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates startup frames in slot 2 of cycles 0 to 4.

10) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

11) In cycle 2, 500 µT after the cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

12) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 4, 500 µT after cycle start , it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

2) In cycle 4, 500 µT after the cycle start, it is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT configures MB with the buffer configuration data set 2.

4) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

5) In cycle 5, 500 µT after cycle start and latest 2 500 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY.

6) 2 500 µT after the CHI command IMMEDIATE_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY)* and that the startup state is undefined *(vPOC!StartupState* = *UNDEFINED*).

7) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions accordingly. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:coldstart join* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.17.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:coldstart join* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart join* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 4.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) In cycle 2, 500 µT after the cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

2) In cycle 5, 500 µT after cycle start and latest 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command FREEZE.

3) 2 500 µT after the CHI command FREEZE it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *COLDSTART_JOIN* and *vPOC!Freeze* = true).

4) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* state and the *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.17.3 Command ALLOW_COLDSTART

— **Test purpose**

Verify correct behaviour after CHI command ALLOW_COLDSTART in *POC:coldstart join* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 4.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) In cycle 2, 500 µT after the cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

— Test execution

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

2) In cycle 4, 500 µT after cycle start and 2 000 µT before NIT, the UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

3) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters the *POC:coldstart listen* startup state.

7.3.5.17.4 Command DEFERRED_READY - Command CLEAR_DEFERRED

— Test purpose

Verify correct behaviour after CHI command DEFERRED_READY in *POC:coldstart join* state. Verify the correct behaviour after CHI command CLEAR_DEFERRED in *POC:coldstart join* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart join* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 5.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) In cycle 2, 500 µT after the cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would leave state *POC:coldstart join* to *POC:ready* (DEFERRED_READY) without CLEAR_DEFERRED command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*) and that *vPOC!CHIReadyRequest* = false.

2) In cycle 4, 1 000 µT after cycle start, the UT issues the CHI command DEFERRED_READY.

3) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that *vPOC!CHIReadyRequest* = true.

4) 5 000 µT after the CHI command DEFERRED_READY and latest 3 000 µT before NIT of cycle 4, the UT issues the CHI command CLEAR_DEFERRED.

5) 2 500 µT after the CHI command CLEAR_DEFERRED it is verified (UT) that *vPOC!CHIReadyRequest* = false.

6) In cycle 5, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*) and that *vPOC!CHIReadyRequest* = false.

Test instance 2 – IUT would stay in state *POC:coldstart join* without DEFERRED_READY command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*) and that *vPOC!CHIReadyRequest* = false.

2) In cycle 4, 1 000 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

3) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that *vPOC!CHIReadyRequest* = true.

4) In cycle 5, 500 µT after cycle start, it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = true.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions accordingly. The IUT performs the state transition to *POC:ready* accordingly and *vPOC!CHIReadyRequest* is set. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state (test instance 2).

7.3.5.17.5 Command DEFERRED_HALT - Command CLEAR_DEFERRED

— Test purpose

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:coldstart join* state. Verify the correct behaviour after CHI command CLEAR_DEFERRED in *POC:coldstart join* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:coldstart join* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 5.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) In cycle 2, 500 µT after the cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*).

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

Test instance 1 – IUT would leave state *POC:coldstart join* to *POC:halt* after DEFERRED_HALT and without CLEAR_DEFERRED command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*) and that *vPOC!CHIHaltRequest* = false.

2) In cycle 4, 1 000 µT after cycle start, the UT issues the CHI command DEFERRED_HALT.

3) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that *vPOC!CHIHaltRequest* = true.

4) 5 000 µT after the CHI command DEFERRED_HALT, the UT issues the CHI command CLEAR_DEFERRED.

5) 2 500 µT after the CHI command CLEAR_DEFERRED it is verified (UT) that *vPOC!CHIHaltRequest* = false.

6) In cycle 5, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*) and that *vPOC!CHIHaltRequest* = false.

7) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

Test instance 2 – IUT would stay in state *POC:coldstart join* without DEFERRED_HALT command:

1) In cycle 4, 500 µT after cycle start, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*) and that *vPOC!CHIHaltRequest* = false.

2) In cycle 4, 1 000 µT after cycle start, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT.

3) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that *vPOC!CHIHaltRequest* = true.

4) In cycle 5, 500 µT after cycle start, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = true.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transitions accordingly. The IUT performs the last state transition to *POC:halt* state accordingly and *vPOC!CHIHaltRequest* is set. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state (test instance 2).

7.3.5.18 POC:integration listen

7.3.5.18.1 Command IMMEDIATE_READY

— Test purpose

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:integration listen* state. Verify the correct behaviour of IUT regarding write access to the protocol configuration data in *POC:integration listen* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration listen* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) It is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

4) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY after the configuration verification.

5) 2 500 µT after the CHI command IMMEDIATE_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY)* and that the startup state is undefined *(vPOC!StartupState* = *UNDEFINED*). It is also verified (UT) that the IUT did not generate any interrupt request.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions accordingly. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:integration listen* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.18.2 Command FREEZE

— **Test purpose**

Verify correct behaviour after CHI command FREEZE in *POC:integration listen* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration listen* to *POC:halt*, initiated by the CHI command FREEZE).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command FREEZE 3 000 µT after the CHI command RUN.

4) 2 500 µT after the CHI command FREEZE it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *INTEGRATION_LISTEN* and *vPOC!Freeze* = true). It is also verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt*.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* and the *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.18.3 Command ALLOW_COLDSTART

— Test purpose

Verify correct behaviour after CHI command ALLOW_COLDSTART in *POC:integration listen* state.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) 3 000 µT after the CHI command RUN, the UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

4) 2 500 µT after the CHI command ALLOW_COLDSTART, it is verified (UT) that the IUT has entered the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters the *POC:coldstart listen* startup state.

7.3.5.18.4 Command DEFERRED_READY

— Test purpose

Verify correct behaviour after CHI command DEFERRED_READY in *POC:integration listen* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration listen* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— Applicability

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY 3 000 µT after the CHI command RUN.

4) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = false.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the state transition to *POC:ready* accordingly and the *vPOC!CHIReadyRequest* is reset. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.18.5 Command DEFERRED_HALT

— **Test purpose**

Verify correct behaviour after CHI command DE FERRED_HALT in *POC:integration listen* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration listen* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT 3 000 µT after the CHI command RUN.

4) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = false.

5) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* accordingly and the *vPOC!CHIHaltRequest* is reset. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.5.19 POC:integration consistency check

7.3.5.19.1 Command IMMEDIATE_READY

— **Test purpose**

Verify correct behaviour after CHI command IMMEDIATE_READY in *POC:integration consistency check* state. Verify the correct behaviour of IUT regarding write access to the protocol and buffer configuration data (Class 1 and Class 2) in *POC:integration consistency check* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration consistency check* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 167.

Table 167 — Modification to basic configurations for POC:integration consistency check – command IMMEDIATE_READY

Parameter	Modification
pKeySlotUsedForStartup	false

Table 168 defines the two different buffer configuration data sets for *POC:integration consistency check* – command IMMEDIATE_READY.

Table 168 — Two different buffer configuration data sets for POC:integration consistency check – command IMMEDIATE_READY

Buffer Configuration Data Class 1	**Set 1**	**Set 2**
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration and configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

4) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

   (b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

   (c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates startup frames in slot 2 of cycles 0 to 2.

10) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

11) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before 1 000 µT after the start of slot 3 , it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

2) In cycle 1, 500 µT after the start of slot 3, it is verified (UT) that the IUT holds the provided configuration data and that the UT does not have write access to any protocol configuration data.

3) The UT configures MB with the buffer configuration data set 2.

4) By reading the configuration data of MB, it is verified (UT) that

   (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

   (b) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

   (c) the Class 1 message buffer configuration data of MB matches set 1 and the Class 2 message buffer configuration data of MB matches set 2 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

5) In cycle 2, 500 µT after cycle start and 2 500 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command IMMEDIATE_READY.

6) 2 500 µT after the CHI command IMMEDIATE_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY)* and that the startup state is undefined *(vPOC!StartupState* = *UNDEFINED*).

7) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the expected state transitions accordingly. The protocol configuration data may be read and shall not be written while the IUT is in the *POC:integration consistency check* state. Write access to buffer configuration data is as configured in *POC:config* state. The IUT does not indicate an interrupt request on the POC state transition to *POC:ready*.

7.3.5.19.2 Command FREEZE

— Test purpose

Verify correct behaviour after CHI command FREEZE in *POC:integration consistency check* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration consistency check* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 169.

Table 169 — Modification to basic configurations for POC:integration consistency check – command FREEZE

Parameter	Modification
pKeySlotUsedForStartup	false

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 2.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

2) In cycle 2, 500 µT after the cycle start and before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the command FREEZE.

3) 2 500 µT after the CHI command FREEZE, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK* and *vPOC!Freeze* = true).

4) It is verified (UT) that the IUT did not generate any interrupt request.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the expected state transition and the *vPOC!Freeze* is set accordingly. The IUT does not indicate an interrupt request on the POC state transition to *POC:halt*.

7.3.5.19.3 Command DEFERRED_READY

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_READY in *POC:integration consistency check* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration consistency check* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 170.

Table 170 — Modification to basic configurations for POC:integration consistency check – command DEFERRED_READY

Parameter	Modification
pKeySlotUsedForStartup	false

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 2.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) and that *vPOC!CHIReadyRequest* = false.

2) In cycle 2, 500 µT after cycle start and 2 000 µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_READY.

3) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the state transition to *POC:ready* accordingly and the *vPOC!CHIReadyRequest* is reset. The IUT indicates an interrupt request on the POC state transition to *POC:ready* state.

7.3.5.19.4 Command DEFERRED_HALT

— **Test purpose**

Verify correct behaviour after CHI command DEFERRED_HALT in *POC:integration consistency check* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:integration consistency check* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 171.

Table 171 — Modification to basic configurations for POC:integration consistency check – command DEFERRED_HALT

Parameter	Modification
pKeySlotUsedForStartup	false

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 2.

9) In cycle 0, 500 µT after the start of slot 3 and before NIT, it is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) In cycle 1, 500 µT after the start of slot 3 and before NIT, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) and that *vPOC!CHIHaltRequest* = false.

2) In cycle 2, 500 µT after cycle start and 2 000µT before NIT, the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and issues the CHI command DEFERRED_HALT.

3) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = false.

4) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT performs the state transition to *POC:halt* accordingly and the *vPOC!CHIHaltRequest* is reset. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.3.6 Control of symbol transmission

7.3.6.1 MTS transmission

— Test purpose

Verify correct transmission of the media access test symbol MTS. Verify reset of slot counters in the symbol window.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

Preamble II.

— Test execution

For dual channel test execution the test has to be performed in three instances. The UT requests MTS transmission for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) In the static segment of cycle 7, the UT requests manual (one-shot) MTS transmission.

2) In cycle7, 50 µT after the start of the symbol window, it is verified (UT) that the slot counter *vSlotCounter* is 0 for all available channels.

3) In the symbol window of cycle 7, it is verified (LT) that the IUT transmits a valid MTS of length *gdTSSTransmitter* + *cdCAS* starting at the symbol window action point on the requested channel(s).

4) It is also verified (LT) that the TxEN output(s) and the TxD output(s) on the requested channel(s) become low simultaneously, that the TxEN output(s) stay low for a period of *gdTSSTransmitter* + *cdCAS,* the TxD output(s) stay low for a period of *gdTSSTransmitter* + *cdCAS* + *cdStaggerDelay* and the TxD and TxEN outputs become high after the claimed low phase.

5) In the symbol window of cycle 8, it is verified (LT) that all available channels are idle.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits the MTS on the requested channel(s) correctly. The IUT resets its slot counters to 0 in the symbol window.

7.3.6.2 WUDOP transmission

— Test purpose

Verify the correct transmission of the wakeup during operation pattern WUDOP.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The UT requests WUDOP transmission for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) In the static segment of cycle 7, the UT requests manual (one-shot) WUDOP transmission.

2) In the symbol window of cycle 7, it is verified (LT) that the IUT transmits a valid WUDOP on the requested channel(s) starting at (*gNumberOfStaticSlots* * *gdStaticSlot* + *gNumberOfMinislots* * *gdMinislot* + *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT after cycle start.
It is verified that the TxEN output(s) and the TxD output(s) on the requested channel(s) become low simultaneously. Further it is verified that TxEN output(s) stay low for (5 * *gdWakeupTxActive* + 1) * *gdBit* µs and that TxD shows Low-High-Low-High-Low phases with *gdWakeupTxActive* * *gdBit* µs length followed by a High phase of at least 1 * *gdBit* µs length.

3) In the symbol window of cycle 8, it is verified (LT) that all available channels are idle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits the WUDOP correctly.

7.3.6.3 Automatic WUDOP transmission

— **Test purpose**

Verify the correct automatic WUDOP transmission according to the configured values of Cycle_Offset and Cycle_Repetition.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 172.

Table 172 — The following repetition and offset parameters are used for control of symbol transmission – automatic WUDOP transmission request(s)

Parameter	Modifications of repetition and offset parameters				
	I	II	III	IV	V
Cycle_Repetition	1	32	40	50	64
Cycle_Offset	0	31	39	49	63

— **Preamble (setup state)**

Preamble II.

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The UT requests WUDOP transmission

for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) In static segment of cycle 7, the UT requests automatic WUDOP transmission according to Cycle_Repetition and Cycle_Offset.

2) In the symbol windows of the next 120 cycles, it is verified (LT) that if *vCycleCounter* mod Cycle_Repetition = Cycle_Offset
the IUT transmits a valid WUDOP on the WUDOP transmission channel(s), otherwise that all active channel(s) are idle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a WUDOP in the symbol window automatically if *vCycleCounter* mod Cycle_Repetition = Cycle_Offset.

7.3.6.4 Request of WUDOP transmission

— **Test purpose**

Verify the correct control of automatic and manual WUDOP transmission.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

Cycle_Repetition = 2.

Cycle_Offset = 0.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

Test steps 10 to 17 of the test execution only apply to dual channel test execution.

1) In cycle 7, the UT switches on automatic WUDOP transmission on all available channels.

2) In the symbol window of cycle 8, it is verified (LT) that the IUT transmits a valid WUDOP on all available channels.

3) In the symbol window of cycle 9, it is verified (LT) that all available channels are idle.

4) In the static segment of cycle 10, the UT requests manual (one-shot) WUDOP transmission on all available channels.

5) In the symbol window of cycle 10, it is verified (LT) that the IUT transmits a valid WUDOP on all available channels.

6) In the static segment of cycle 11, the UT requests manual (one-shot) WUDOP transmission on all available channels.

7) In the symbol window of cycle 11, it is verified (LT) that the IUT transmits a valid WUDOP on all available channels.

8) In the static segment of cycle 12, the UT switches off automatic WUDOP transmission on all available channels.

9) In the symbol window of cycles 12, 13 and 14, it is verified (LT) that all available channels are idle.

10) In the static segment of cycle 15, the UT switches on automatic WUDOP transmission on channel A.

11) In the symbol window of cycle 16, it is verified (LT) that the IUT transmits a valid WUDOP on channel A and that the channel B is idle.

12) In the symbol window of cycle 17, it is verified (LT) that both channels are idle.

13) In the static segment of cycle 18, the UT switches on automatic WUDOP transmission on channel B and switches off automatic WUDOP transmission on channel A.

14) In the symbol window of cycle 18, it is verified (LT) that the IUT transmits a valid WUDOP on channel B and that the channel A is idle.

15) In the symbol window of cycle 19, it is verified (LT) that both channels are idle.

16) In the static segment of cycle 20, the UT switches off automatic WUDOP transmission on channel B.

17) In the symbol window of cycles 20 and 21, it is verified (LT) that both channels are idle.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT shall transmit a WUDOP in the symbol window either automatically when *vCycleCounter* mod Cycle_Repetition = Cycle_Offset, after a manual request, or both.

7.3.6.5 Request for MTS and WUDOP transmission

— Test purpose

Verify correct transmission of the media access test symbol MTS in the case that the transmission of both media test symbol and wakeup pattern during operation have been requested by the host.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

Preamble II.

— Test execution

For dual channel test execution the test has to be performed in three instances. The UT requests MTS and WUDOP transmission for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) (a) In the static segment of cycle 7, the UT requests manual (one-shot) MTS transmission.
(b) In the static segment of cycle 7, the UT requests manual (one-shot) WUDOP transmission.

2) (a) 2 500 µT after the request for MTS transmission the UT requests a (one-shot) WUDOP transmission.
(b) 2 500 µT after the request for WUDOP transmission the UT requests a (one-shot) MTS transmission.

3) In the symbol window of cycle 7, it is verified (LT) that the IUT transmits a valid MTS of length *gdTSSTransmitter* + *cdCAS* starting at the symbol window action point on the requested channel(s).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits the MTS on the requested channel(s) correctly.

7.3.7 Control of external clock correction

7.3.7.1 External offset correction

— Test purpose

Verify correct behaviour of external offset correction. The IUT shall alter its offset correction according to the applied external offset correction.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 173, Table 174 and Table 175.

Table 173 — Modification to basic configurations 1a and 1b for control of external clock correction – external offset correction

Parameter	Modification to Basic Configuration									
	1a					1b				
	I	II	III	IV	V	I	II	III	IV	V
gMaxWithoutClockCorrectionFatal	1					1				
gMaxWithoutClockCorrectionPassive	1					1				
gExternOffsetCorrection [µs]	0,35	0,175	0	0,175	0,35	0,35	0,175	0	0,175	0,35
pAllowHaltDueToClock	false					false				
pClusterDriftDamping [µT]	0					0				
pExternOffsetCorrection [µT]	14	7	0	7	14	28	14	0	14	28
vExternOffsetControl	-1	-1	+1	+1	+1	-1	-1	-1	+1	+1

Table 174 — Modification to basic configurations 2a and 2b for control of external clock correction – external offset correction

Parameter	Modification to Basic Configuration									
	2a					2b				
	I	II	III	IV	V	I	II	III	IV	V
gMaxWithoutClockCorrectionFatal	1					1				
gMaxWithoutClockCorrectionPassive	1					1				
gExternOffsetCorrection [µs]	0,35	0,175	0	0,175	0,35	0,35	0,2	0	0,2	0,35
pAllowHaltDueToClock	false					false				
pClusterDriftDamping [µT]	0					0				
pExternOffsetCorrection [µT]	14	7	0	7	14	7	4	0	4	7
vExternOffsetControl	-1	-1	+1	+1	+1	-1	-1	-1	+1	+1

Table 175 — Modification to basic configuration 3 for control of external clock correction – external offset correction

Parameter	Modification to Basic Configuration 3				
	I	II	III	IV	V
gMaxWithoutClockCorrectionFatal	1				
gMaxWithoutClockCorrectionPassive	1				
gExternOffsetCorrection [µs]	0,35	0,2	0	0,2	0,35
pAllowHaltDueToClock	false				
pClusterDriftDamping [µT]	0				
pExternOffsetCorrection [µT]	7	4	0	4	7
vExternOffsetControl	-1	-1	+1	+1	+1

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates startup frames in slot 2 and in slot 3.

The preamble is supplemented by:

1) In cycle 4, the LT begins to simulate an additional startup frame in slot 3.

2) Starting in cycles 7 the LT continues simulating its startup frame in slot 2 and slot 3.

— **Test execution**

1) In cycle 9, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

2) In slot 1 of cycle 9, the UT applies the external offset correction to the IUT only for cycle 9.

3) At the end of cycle 9, the LT applies the same external offset correction.

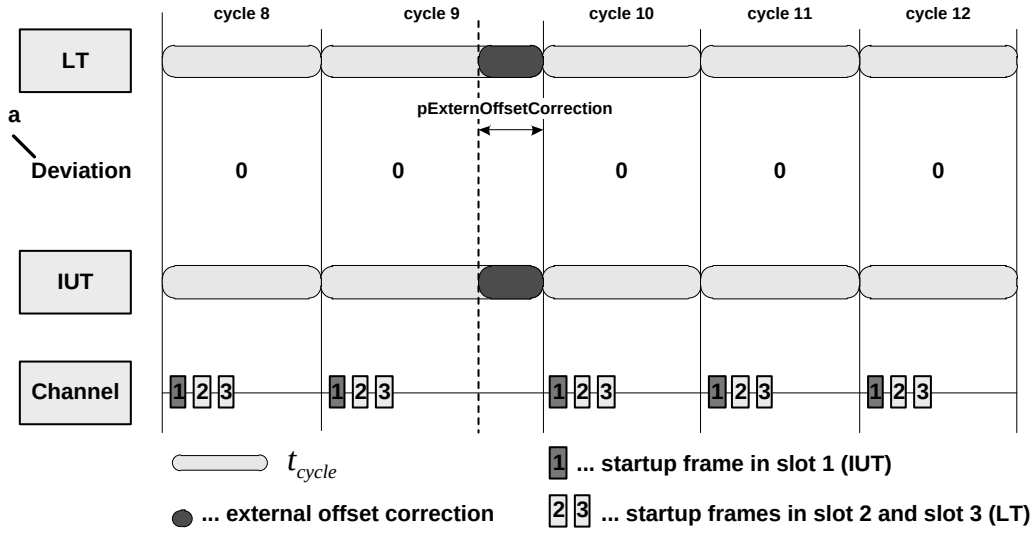
4) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

5) In cycle 10, it is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 9 and slot 1 / cycle 10 is *pMicroPerCycle* + *pExternOffsetCorrection* * *vExternOffsetControl* + ξ + ξ_{IUT} µT.

6) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is still synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

7) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

Figure 35 depicts the external offset correction.



a Expected deviation measured by the IUT.

Figure 35 — External offset correction

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs clock correction according to the applied external offset correction.

7.3.7.2 External rate correction

— **Test purpose**

Verify correct behaviour of external rate correction. The IUT shall alter its rate correction according to the applied external rate correction.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 176, Table 177 and Table 178.

Table 176 — Modification to basic configurations 1a and 1b for control of external clock correction – external rate correction

Parameter	Modification to Basic Configuration									
	1a					1b				
	I	II	III	IV	V	I	II	III	IV	V
gMaxWithoutClockCorrectionFatal	1					1				
gMaxWithoutClockCorrectionPassive	1					1				
gExternRateCorrection [µs]	0,35	0,175	0	0,175	0,35	0,35	0,175	0	0,175	0,35
pAllowHaltDueToClock	false					false				
pClusterDriftDamping [µT]	0					0				
pExternRateCorrection [µT]	14	7	0	7	14	28	14	0	14	28
vExternRateControl	-1	-1	+1	+1	+1	-1	-1	-1	+1	+1

Table 177 — Modification to basic configurations 2a and 2b for control of external clock correction – external rate correction

Parameter	Modification to Basic Configurations									
	2a					2b				
	I	II	III	IV	V	I	II	III	IV	V
gMaxWithoutClockCorrectionFatal	1					1				
gMaxWithoutClockCorrectionPassive	1					1				
gExternRateCorrection [µs]	0,35	0,175	0	0,175	0,35	0,35	0,2	0	0,2	0,35
pAllowHaltDueToClock	false					false				
pClusterDriftDamping [µT]	0					0				
pExternRateCorrection [µT]	14	7	0	7	14	7	4	0	4	7
vExternRateControl	-1	-1	+1	+1	+1	-1	-1	-1	+1	+1

Table 178 — Modification to basic configuration 3 for control of external clock correction – external rate correction

Parameter	Modification to Basic Configuration 3				
	I	**II**	**III**	**IV**	**V**
gMaxWithoutClockCorrectionFatal	1				
gMaxWithoutClockCorrectionPassive	1				
gExternRateCorrection [µs]	0,35	0,2	0	0,2	0,35
pAllowHaltDueToClock	false				
pClusterDriftDamping [µT]	0				
pExternRateCorrection [µT]	7	4	0	4	7
vExternRateControl	-1	-1	+1	+1	+1

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates startup frames in slot 2 and in slot 3.

The preamble is supplemented by:

1) In cycle 4, the LT begins to simulate the additional startup frame in slot 3.

2) Starting in cycles 7 the LT continues simulating its startup frame in slot 2 and slot 3.

— **Test execution**

1) In cycle 9, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

2) In slot 1 of cycle 9, the UT applies the external rate correction.

3) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

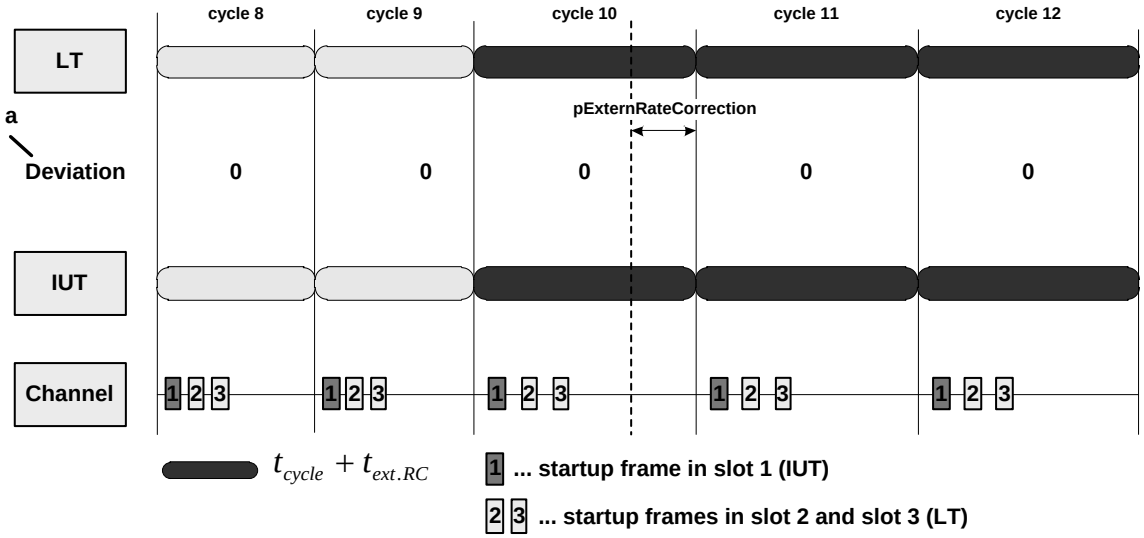
4) Starting with cycle 10, the LT applies the same external rate correction. The rate correction is distributed over the cycle according to the macrotick generation process as specified in the data link layer specification, which impacts the point in time the LT starts the frame simulation.

5) In cycle 11, it is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is *pMicroPerCycle* + *vExternRateControl* * *pExternRateCorrection* + ξ + ξ_{IUT} µT.

6) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = *vExternRateControl* * *pExternRateCorrection* + ξ_{IUT} µT.

7) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is still synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 36 depicts the external rate correction.



a Expected deviation measured by the IUT.

Figure 36 — External rate correction

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs clock correction according to the applied external rate correction.

7.3.8 Buffer configuration

7.3.8.1 Transmit buffer

7.3.8.1.1 Transmit buffer assignment on "A" via slot ID in the static segment

— **Test purpose**

Verify correct transmission of a static frame on channel A in all configured slots. The IUT shall transmit frames in the appropriate slots on channel A.

— **Applicability**

SC, DC.

All checks on channel B are not applicable for SC devices.

— **Configuration**

All basic configurations using the standard modification set 1 and the additional modification of Table 179.

Table 179 — Modification to basic configurations for transmit buffer – transmit buffer assignment on "A" via slot ID in the static segment

Parameter	Modification
pKeySlotID	0

Additionally to the static frame in slot 1, the IUT is configured to send static frames on channel A in slot ceil(*gNumberOfStaticSlots* / 2) and *gNumberOfStaticSlots*. These three slots are referred as configured slots later in this test. The payload byte 0 is 0x55, byte 1 is 0x55.

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, it is verified (LT) that
the IUT transmits its frames in the configured slots on channel A correctly (payload equals byte 0 0x55, byte 1 0x55),
the frame ID within the frame header equals the slot ID of the transmission slot and,
for dual channel test execution, the IUT does not transmit any communication element on channel B.

2) In the NIT of cycle 9, the UT modifies the payload in the transmit buffer to byte 0 0xAA, byte 1 0xAA and verifies that the payload was changed to byte 0 0xAA, byte 1 0xAA.

3) In cycle 10, it is verified (LT) that
the IUT transmits its frames in the configured slots on channel A correctly (payload equals byte 0 0xAA, byte 1 0xAA) and,
the frame ID within the frame header equals the slot ID of the transmission slot and,
for dual channel test execution, the IUT does not transmit any communication element on channel B.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.2 Transmit buffer assignment on "A" via cycle count in the static segment

— Test purpose

Verify correct transmission of frames on channel A in the configured slots and within the configured communication cycles. The IUT shall transmit frames in the appropriate slots and cycles on channel A.

— Applicability

SC, DC.

All checks on channel B are not applicable for SC devices.

— Configuration

All basic configurations using the modifications as listed in Table 180.

Table 180 — Modification to basic configurations for transmit buffer – transmit buffer assignment on "A" via cycle count in the static segment

Parameter	Modification
pKeySlotID	0

The IUT is configured to send static frames on channel A in slots 1 and 20 (these slots are referred as configured slots later in this test). One transmit buffer is set up for transmission in slot 1. The payload byte 0 is 0x55, byte 1 is 0x55. Two transmit buffers are set up for transmission in slot 20 with different payload data, the first one for cycle 9 (the payload byte 0 is 0x33, byte 1 is 0x33) the second one for cycle 10 (the payload byte 0 is 0xCC, byte 1 is 0xCC).

Table 181 defines the message buffer configuration.

Table 181 — Message buffer configuration for transmit buffer – transmit buffer assignment on "A" via cycle count in the static segment

Tx Buffer	SlotID	Cycle_Offset	Cycle_Repetition	Channel
B1	1	0	1	A
B2	20	9	64	A
B3	20	10	64	A

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycles 9 and 10, it is verified (LT) that
the IUT transmits its frame in slot 1 on channel A correctly,
for dual channel test execution, the IUT does not transmit any communication element on channel B and,
the cycle count in the frame header of every IUT frame equals the cycle number of the transmission cycle.

2) In cycle 9, it is verified (LT) that the IUT transmits its frame in slot 20 on channel A with the payload data as given in the configuration.

3) In cycle 10, it is verified (LT) that the IUT transmits its frame in slot 20 on channel A with the payload data as given in the configuration.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.3 Transmit buffer assignment on "B" via slot ID in the static segment

— Test purpose

Verify correct transmission of a static frame on channel B in all configured slots. The IUT shall transmit frames in the appropriate slots on channel B.

— Applicability

SC, DC.

All checks on channel A are not applicable for SC devices.

— Configuration

All basic configurations using the modifications as listed in Table 182.

Table 182 — Modification to basic configurations for transmit buffer – transmit buffer assignment on "B" via slot ID in the static segment

Parameter	Modification
pKeySlotID	0

Additionally to the static frame in slot 1, the IUT is configured to send static frames on channel B in slot ceil(*gNumberOfStaticSlots* / 2) and *gNumberOfStaticSlots*. These three slots are referred as configured slots later in this test. The payload byte 0 is 0x55, byte 1 is 0x55.

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, it is verified (LT) that
the IUT transmits its frames in the configured slots on channel B correctly (payload equals byte 0 0x55, byte 1 0x55) and,
the frame ID within the frame header equals the slot ID of the transmission slot and,
for dual channel test execution, the IUT does not transmit any communication element on channel A.

2) In the NIT of cycle 9, the UT modifies the payload in the transmit buffer to byte 0 0xAA, byte 1 0xAA and verifies that the payload was changed to byte 0 0xAA, byte 1 0xAA.

3) In cycle 10, it is verified (LT) that
the IUT transmits its frames in the configured slots on channel B correctly (payload equals byte 0 0xAA, byte 1 0xAA) and,
the frame ID within the frame header equals the slot ID of the transmission slot and,
for dual channel test execution, the IUT does not transmit any communication element on channel A.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.4 Transmit buffer assignment on "B" via cycle count in the static segment

— Test purpose

Verify correct transmission of frames on channel B in the configured slots and within the configured communication cycles. The IUT shall transmit frames in the appropriate slots and cycles on channel B.

— Applicability

SC, DC.

All checks on channel A are not applicable for SC devices.

— Configuration

All basic configurations using the modifications as listed in Table 183.

Table 183 — Modification to basic configurations for transmit buffer – transmit buffer assignment on "B" via cycle count in the static segment

Parameter	Modification
pKeySlotID	0

The IUT is configured to send static frames on channel B in slots 1 and 20 (these slots are referred as configured slots later in this test. One transmit buffer is set up for transmission in slot 1. The payload byte 0 is 0x55, byte 1 is 0x55. Two transmit buffers are set up for transmission in slot 20 with different payload data, the first one for cycle 9 (the payload byte 0 is 0x33, byte 1 is 0x33) the second one for cycle 10 (the payload byte 0 is 0xCC, byte 1 is 0xCC).

Table 184 defines the message buffer configuration.

Table 184 — Message buffer configuration for transmit buffer – transmit buffer assignment on "B" via cycle count in the static segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	1	0	1	B
B2	20	9	64	B
B3	20	10	64	B

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycles 9 and 10, it is verified (LT) that
the IUT transmits its frame in slot 1 on channel B correctly and,
the cycle count in the frame header of every IUT frame equals the cycle number of the transmission cycle and,
for dual channel test execution, the IUT does not transmit any communication element on channel A.

2) In cycle 9, it is verified (LT) that the IUT transmits its frame in slot 20 on channel B with the payload data as given in the configuration.

3) In cycle 10, it is verified (LT) that the IUT transmits its frame in slot 20 on channel B with the payload data as given in the configuration.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.5 Transmit buffer assignment on "A&B" via slot ID in the static segment

— Test purpose

Verify correct transmission of a frame on channel A and on channel B in all configured slots. The IUT shall transmit frames in the appropriate slots on channel A and channel B.

— Applicability

DC.

— Configuration

All Basic Configurations using the standard modification set 1 and the additional modification of Table 185.

Table 185 — Modification to basic configurations for transmit buffer – transmit buffer assignment on "A&B" via slot ID in the static segment

Parameter	Modification
pKeySlotID	0

Two transmit buffers are set up for transmission in slot 1 with different payload data, one for slot 1 / channel A (the payload byte 0 is 0x33, byte 1 is 0x33) and the other one for slot 1 / channel B (the payload byte 0 is 0xCC, byte 1 is 0xCC).

One transmit buffer is set up for transmission on both channels in the slot (ceil(*gNumberOfStaticSlots* / 2)), and one for transmission on both channels in the slot (*gNumberOfStaticSlots*). The payload byte 0 is 0x55, byte 1 is 0x55.

Table 186 defines the message buffer configuration.

Table 186 — Message buffer configuration for transmit buffer – transmit buffer assignment on "A&B" via slot ID in the static segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	1	0	1	A
B2	1	0	1	B
B3	ceil(*gNumberOfStaticSlots* / 2)	0	1	A&B
B4	*gNumberOfStaticSlots*	0	1	A&B

— **Preamble (setup state)**

Preamble III.

— **Test execution**

In cycle 9, it is verified (LT) that
the IUT transmits its frames in the configured slots on channel A and on channel B correctly and, the frame ID within the frame header equals the slot ID of the transmission slot.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.6 Transmit buffer assignment on "A&B" via cycle count in the static segment

— **Test purpose**

Verify correct frame transmission in the static segment on channel A and channel B in the configured slots and within the configured communication cycles and verify the slot assignment methods (Cycle-dependent slot assignment and Cycle-independent slot assignment). The IUT shall transmit frames in the appropriate slots and cycles on channel A and channel B.

— **Applicability**

DC.

— **Configuration**

All Basic Configurations using the standard modification set 1 and the additional modification of Table 187.

Table 187 — Modification to basic configurations for transmit buffer – transmit buffer assignment on "A&B" via cycle count in the static segment

Parameter	Modification
pKeySlotID	0

The IUT is configured to send static frames in slots 1 and 20. Two transmit buffers are set up for transmission in slot 1 on both channels with different payload data, the first one for cycle 9 (the payload byte 0 is 0x33, byte 1 is 0x33) the second one for cycle 10 (the payload byte 0 is 0xCC, byte 1 is 0xCC). Two transmit buffers are set up for transmission in slot 20 on channel A with different payload data, one for cycle 9 (the payload byte 0 is 0x11, byte 1 is 0x11) the other for cycle 10 (the payload byte 0 is 0xEE, byte 1 is 0xEE). Two transmit buffers are set up for transmission in slot 20 on channel B with different payload data, one for cycle 9 (the payload byte 0 is 0x55, byte 1 is 0x55) the other one for cycle 10 (the payload byte 0 is 0xAA, byte 1 is 0xAA).

Table 188 defines the message buffer configuration.

Table 188 — Message buffer configuration for transmit buffer assignment on "A&B" via cycle count in the static segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	1	9	64	A&B
B2	1	10	64	A&B
B3	20	9	64	A
B4	*20*	10	64	A
B5	20	9	64	B
B6	20	10	64	B

— **Preamble (setup state)**

Preamble III.

Deviating from the preamble the LT shall not check the frame transmission of IUT in slot 1 of cycle 8.

The test shall be executed with two different slot assignment variants:

(a) Cycle-dependent slot assignment: The IUT is configured for the new static segment behaviour that allows a node to transmit in a slot with a specific number in only some cycles.

(b) Cycle-independent slot assignment: The UT configures the IUT for static segment behaviour consistent with the previous versions of the protocol, where a node that transmits in a slot with a specific slot number in any cycle shall transmit either a non-null frame or a null frame in all slots with this number in all cycles.

— **Test execution**

1) In cycles 9 and 10, it is verified (LT) that the cycle count in the frame header of every IUT frame equals the cycle number of the transmission cycle.

2) In cycle 9 and 10, it is verified (LT) that
the IUT transmits its frame in slot 1 on both channels with the payload data as given in the configuration and,
the IUT transmits its frame in slot 20 on channel A with the payload data as given in the configuration and,
the IUT transmits its frame in slot 20 on channel B with the payload data as given in the configuration.

3) In cycle 11, it is verified (LT), according to the configured behaviour in the preamble (slot assignment variants), that
(a) the IUT does not transmit any frames on either channels.
(b) the IUT transmits only two null frames, one in slot 1 and the other in slot 20 on both channels.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames on both channels correctly in cycles 9 and 10. The IUT has the correct static segment behaviour in cycle 11 depending on its configuration.

7.3.8.1.7 Transmit buffer assignment on "A" via slot ID in the dynamic segment

— Test purpose

Verify correct transmission of a frame on channel A in all configured dynamic slots. The IUT shall transmit frames in the appropriate slots on channel A.

— Applicability

SC, DC.

All checks on channel B are not applicable for SC devices.

— Configuration

All basic configurations using the standard modification set 2.

The IUT is configured to send dynamic frames on channel A in slots 3, 512, 1023 and 2047. An additional transmit buffer is configured for slot 28 on Channel A, but the IUT is not requested to transmit a frame in this slot. The payload length of dynamic frames in slot 512 is 0 in cycle 7, 1 in cycle 8, 64 in cycle 9 and 127 in cycle 10. The payload of dynamic frames in slot 512 is 0xAA in every payload byte that might be hold in the dedicated transmit buffer. The payload of dynamic frames in slots 3, 1 023 and 2 047 is always 0x55 in byte 0 and 0x55 in byte 1.

Table 189 defines the message buffer configuration.

Table 189 — Message buffer configuration for transmit buffer assignment on "A" via slot ID in the dynamic segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	3	0	1	A
B2	512	0	1	A
B3	1 023	0	1	A
B4	2 047	0	1	A
B5[a]	28	0	1	A

[a] Payload data valid flag in buffer configuration data of B5 is set to false.

— Preamble (setup state)

Preamble II.

— Test execution

In cycles 7 to 10, it is verified (LT) that:

— the IUT does not transmit any frame in slot 28 on channel A and,

— the IUT transmits its frames in the configured slots (3, 512, 1 023 and 2 047) on channel A correctly and,

— the frame ID within the frame header equals the slot ID of the transmission slot and,

— for dual channel test execution, the IUT does not transmit any communication element in the dynamic segment on channel B.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.8 Transmit buffer assignment on "A" via cycle count in the dynamic segment

— Test purpose

Verify correct transmission of frames on channel A in the configured slots and within the configured communication cycles. The IUT shall transmit frames in the appropriate slots and cycles on channel A. Verify the indication of the last frame transmission in the dynamic segment.

— Applicability

SC, DC.

All checks on channel B are not applicable for SC devices.

— Configuration

All basic configurations using the standard modification set 2.

The IUT is configured to send a dynamic frames on channel A in slots 3 and 20. Two transmit buffers are set up for transmission in slot 3 with different payload data, the first one for cycle 7 (the payload byte 0 is 0x55, byte 1 is 0x55) the second one for cycle 8 (the payload byte 0 is 0xAA, byte 1 is 0xAA). Two transmit buffers are set up for transmission in slot 20 with different payload data, the first one for cycle 7 (the payload byte 0 is 0x33, byte 1 is 0x33) the second one for cycle 8 (the payload byte 0 is 0xCC, byte 1 is 0xCC).

Table 190 defines the message buffer configuration.

Table 190 — Message buffer configuration for transmit buffer assignment on "A" via cycle count in the dynamic segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	3	7	64	A
B2	3	8	64	A
B3	20	7	64	A
B4	20	8	64	A

— Preamble (setup state)

Preamble II.

— Test execution

1) In cycle 7, 500 µT after the cycle start and before the end of slot 20, it is verified (UT) that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on available channel(s) is 0 in the dynamic segment status.

2) In cycles 7 and 8, it is verified (LT) that
the IUT transmits its frame in slot 3 and in slot 20 on channel A correctly and,
the cycle count in the frame header of every IUT frame equals the cycle number of the transmission cycle and,
for dual channel test execution, the IUT does not transmit any communication element in the dynamic segment on channel B.

3) 500 µT after the start of NIT of the cycles 7 and 8, it is verified (UT) that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on channel A is 20 and that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on channel B (for dual channel test execution) is 0 in the dynamic segment status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration. The IUT indicates the slot during which the IUT transmitted its last dynamic frame in the dynamic segment status correctly.

7.3.8.1.9 Transmit buffer assignment on "B" via slot ID in the dynamic segment

— **Test purpose**

Verify correct transmission of a frame on channel B in all configured dynamic slots. The IUT shall transmit frames in the appropriate slots on channel B.

— **Applicability**

SC, DC.

All checks on channel A are not applicable for SC devices.

— **Configuration**

All basic configurations using the standard modification set 2.

The IUT is configured to send dynamic frames on channel B in slots 3, 512, 1 023 and 2 047. An additional transmit buffer is configured for slot 28 on Channel B, but the IUT is not requested to transmit a frame in this slot. The payload length of dynamic frames in slot 512 is 0 in cycle 7, 1 in cycle 8, 64 in cycle 9 and 127 in cycle 10. The payload of dynamic frames in slot 512 is 0xAA in every payload byte that might be hold in the dedicated transmit buffer. The payload of dynamic frames in slots 3, 1 023 and 2 047 is always 0x55 in byte 0 and 0x55 in byte 1.

Table 191 defines the message buffer configuration.

Table 191 — Message buffer configuration for transmit buffer assignment on "B" via slot ID in the dynamic segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	3	0	1	B
B2	512	0	1	B
B3	1 023	0	1	B
B4	2 047	0	1	B
B5[a]	28	0	1	B
[a] Payload data valid flag in buffer configuration data of B5 is set to false				

— **Preamble (setup state)**

Preamble II.

— **Test execution**

In cycles 7 to 10, it is verified (LT) that

— the IUT dos not transmit any frame in slot 28 on channel B and,

— the IUT transmits its frames in the configured slots (3, 512, 1 023 and 2 047) on channel B correctly and,

— the frame ID within the frame header equals the slot ID of the transmission slot and,

— for dual channel test execution, the IUT does not transmit any communication element in the dynamic segment on channel A.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.10 Transmit buffer assignment on "B" via cycle count in the dynamic segment

— **Test purpose**

Verify correct transmission of frames on channel B in the configured slots and within the configured communication cycles. The IUT shall transmit frames in the appropriate slots and cycles on channel B. Verify the indication of the last frame transmission in the dynamic segment.

— **Applicability**

SC, DC.

All checks on channel A are not applicable for SC devices.

— **Configuration**

All basic configurations using the standard modification set 2.

The IUT is configured to send a dynamic frame on channel B in slots 3 and 20. Two transmit buffers are set up for transmission in slot 3 with different payload data, the first one for cycle 7 (the payload byte 0 is 0x55, byte 1 is 0x55), the second one for cycle 8 (the payload byte 0 is 0xAA, byte 1 is 0xAA). Two transmit buffers are set up for transmission in slot 20 with different payload data, the first one for cycle 7 (the payload byte 0 is 0x33, byte 1 is 0x33) the second one for cycle 8 (the payload byte 0 is 0xCC, byte 1 is 0xCC).

Table 192 defines the message buffer configuration.

Table 192 — Message buffer configuration for transmit buffer assignment on "B" via cycle count in the dynamic segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	3	7	64	B
B2	3	8	64	B
B3	20	7	64	B
B4	20	8	64	B

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, 500 µT after the start and before NIT, it is verified (UT) that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on available channel(s) is 0 in the dynamic segment status.

2) In cycles 7 and 8, it is verified (LT) that

   — the IUT transmits its frame in slot 3 and in slot 20 on channel B correctly and,

   — the cycle count in the frame header of every IUT frame equals the cycle number of the transmission cycle and,

   — for dual channel test execution, the IUT does not transmit any communication element in the dynamic segment on channel A.

3) 500 µT after the start of NIT of the cycles 7 and 8, it is verified (UT) that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on channel B is 20 and that the slot ID of the last frame transmission in the dynamic segment (*vLastDynTxSlot*) on channel A (for dual channel test execution) is 0 in the dynamic segment status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration. The IUT indicates the slot during which the IUT transmitted its last dynamic frame in the dynamic segment status correctly.

7.3.8.1.11 Message length

— Test purpose

Verify correct frame transmission by varying the available message data (*MessageLength*) in a transmit buffer. The IUT shall append padding bytes to the end of the frame during transmission in the static segment, if the available message data (*MessageLength*) in a transmit buffer is less than *gPayloadLengthStatic*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 193, Table 194 and Table 195.

Table 193 — Modification to basic configurations 1a and 1b for transmit buffer – message length

Parameter	Modification to Basic Configurations			
	1a		1b	
	I	II	I	II
gdStaticSlot [MT]	274		274	
gMacroPerCycle [MT]	8 950		8 950	
gNumberOfMinislots	500		500	
gNumberOfStaticSlots	16		16	
gPayloadLengthStatic [two-byte word]	127	1	127	1
gdCycle [µs]	8 950		8 950	
gdNIT [MT]	26		26	
pdListenTimeout [µT]	716 430		1 432 860	
pLatestTx [Minislot]	469		469	
pMicroPerCycle [µT]	358 000		716 000	
pOffsetCorrectionStart [MT]	8 940		8 940	
pRateCorrectionOut [µT]	215		430	

Table 194 — Modification to basic configurations 2a and 2b for transmit buffer – message length

Parameter	Modification to Basic Configurations			
	2a		2b	
	I	II	I	II
gdStaticSlot [MT]	271		271	
gMacroPerCycle [MT]	7 875		7 875	
gNumberOfMinislots	500		500	
gNumberOfStaticSlots	16		16	
gPayloadLengthStatic [two-byte word]	127	1	127	1
gdCycle [µs]	15 750		15 750	
gdNIT [MT]	16		16	
pdListenTimeout [µT]	1 260 758		630 380	
pLatestTx [Minislot]	461		461	
pMicroPerCycle [µT]	630 000		315 000	
pOffsetCorrectionStart [MT]	7 865		7 865	
pRateCorrectionOut [µT]	379		190	

Table 195 — Modification to basic configuration 3 for transmit buffer – message length

Parameter	Modification to Basic Configuration 3	
	I	II
gdStaticSlot [MT]	331	
gMacroPerCycle [MT]	8 000	
gNumberOfMinislots	380	
gNumberOfStaticSlots	16	
gPayloadLengthStatic [two-byte word]	76	1
gdCycle [µs]	16 000	
gdNIT [MT]	20	
pdListenTimeout [µT]	640 386	
pLatestTx [Minislot]	303	
pMicroPerCycle [µT]	320 000	
pOffsetCorrectionStart [MT]	7 990	
pRateCorrectionOut [µT]	193	

$MessageLength_1$ = 32 two-byte word or the implementation specific maximum length of available message data in two-byte word if less than 32 two-byte word.

$MessageLength_5$ = *gPayloadLengthStatic* two-byte word or the implementation specific maximum length of available message data in two-byte word if less than *gPayloadLengthStatic* two-byte word.

$MessageLength_{20}$ = 32 two-byte word or the implementation specific maximum length of available message data in two-byte word if less than 32 two-byte word.

$MessageLength_{21}$ = 127 two-byte word or the implementation specific maximum length of available message data in two-byte word if less than 127 two-byte word.

The IUT is configured to send frames in static slots 1 and 5 and in dynamic slots 20 and 21.

A transmit buffer is configured for slot 1 with *MessageLength* set to $MessageLength_1$, slot 5 with *MessageLength* set to $MessageLength_5$, slot 20 / cycle 9 with *MessageLength* set to $MessageLength_{20}$, slot 20 / cycle 10 with *MessageLength* set to 1 two-byte word and slot 21 with *MessageLength* set to $MessageLength_{21}$, respectively.

For dual channel test execution dedicated transmit buffers shall be assigned for each channel. The payload is initialized in a way that every payload byte contains an individual none-zero value in increasing order. The payload preamble indicator is always set to '0'.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the IUT sends its frames in slot 1, slot 5, slot 20 and slot 21. It is verified (LT) that the IUT's frame in slot 1 contains a payload of *gPayloadLengthStatic* two-byte word, the payload length in the frame header equals *gPayloadLengthStatic* and

   (I) the transmit buffer contents appear at the beginning of the frame's payload section followed by the padding pattern 0x0000,

   (II) the first two-byte word of the transmit buffer contents appears in the frame's payload section, the IUT's frame in slot 5 contains a payload of *gPayloadLengthStatic* two-byte word and the payload length in the frame header equals *gPayloadLengthStatic*, the IUT's frame in slot 20 contains a payload of $MessageLength_{20}$ two-byte word and the payload length in the frame header equals $MessageLength_{20}$ and the IUT's frame in slot 21 contains a payload of $MessageLength_{21}$ *t*wo-byte word and the payload length in the frame header equals $MessageLength_{21}$.

2) In cycle 10, the IUT sends its frames in slot 1, slot 5, slot 20 and slot 21. It is verified (LT) that the IUT's frame in slot 20 contains a payload of 1 two-byte word and the payload length in the frame header equals 1.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames correctly.

7.3.8.1.12 Transmit buffer assignment in the static and dynamic segment

— Test purpose

Verify correct transmission of frames in the configured slots and within the configured communication cycles.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 196.

Table 196 — Modification to basic configuration for transmit buffer – transmit buffer assignment in the static and dynamic segment

Parameter	Modification
gPayloadLengthStatic [two-byte word]	1

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to send frames in static slot 3 and in dynamic slot n_{dyn}.

Two transmit buffers are set up for transmission in slot 3 with different payload data, the first one for cycle 9 (the payload byte 0 is 0x33, byte 1 is 0x33), the second one for cycle 10 (the payload byte 0 is 0xCC, byte 1 is 0xCC).

Two transmit buffers are set up for transmission in slot n_{dyn} on one available channel with different payload data, the first one for cycle 9 (the payload byte 0 is 0x55, byte 1 is 0x55), the second one for cycle 10 (the payload byte 0 is 0xAA, byte 1 is 0xAA).

For dual channel test execution, two additional transmit buffers are set up for transmission in slot n_{dyn} on the second available channel with different payload data, the first one for cycle 9 (the payload byte 0 is 0x11, byte 1 is 0x11), the second one for cycle 10 (the payload byte 0 is 0xEE, byte 1 is 0xEE).

Table 197 defines the message buffer configuration.

Table 197 — Message buffer configuration for transmit buffer assignment in the static and dynamic segment

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Channel
B1	3	9	64	A / B / A&B
B2	3	10	64	A / B / A&B
B3	n_{dyn}	9	64	A / B / A
B4	n_{dyn}	10	64	A / B / A
B5[a]	n_{dyn}	9	64	- / - / B
B6[a]	n_{dyn}	10	64	- / - / B
[a] only applicable for dual channel test execution.				

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, it is verified (LT) that the IUT transmits its frame in slot 3 and slot n_{dyn} on respective channel(s) with the payload data as given in the configuration.

2) In cycle 10, it is verified (LT) that the IUT transmits its frame in slot 3 and slot n_{dyn} on respective channel(s) with the payload data as given in the configuration.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.13 Payload data valid flag

— **Test purpose**

Verify that the IUT only transmits frames of buffers whose payload data valid flag is true.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

Two transmit buffers are configured in IUT, one to send a frame in static slot 3 (the payload byte 0 is 0xAA, byte 1 is 0xBB) and one to send a frame in the dynamic slot n_{dyn} (the payload byte 0 is 0xDD, byte 1 is 0xCC) in single shot transmission mode.

Two transmit buffers are configured in IUT, one to send frames in static slot 4 (the payload byte 0 is 0xBB, byte 1 is 0xAA) and one to send frames in the dynamic slot (n_{dyn} + 1) (the payload byte 0 is 0xCC, byte 1 is 0xDD) in continuous transmission mode.

Payload data valid flag is set to false during the buffer configuration.

For dual channel test execution the buffer B2 is configured for transmission on channel A and the buffer B4 for transmission on channel B.

Table 198 defines the message buffer configuration.

Table 198 — Message buffer configuration for payload data valid flag

Tx Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Transmission mode
B1	3	0	1	single shot mode
B2	n_{dyn}	0	1	single shot mode
B3	4	0	1	continuous mode
B4	n_{dyn} + 1	0	1	continuous mode

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, it is verified (LT) that the IUT sends its frame in static slot 3 and 4 as null frames (the null frame indicator is set to '0', the payload preamble indicator is set to '0' and all payload bytes are 0x00), and that the IUT does not send in the dynamic slots n_{dyn} and (n_{dyn} + 1).

2) In the NIT of cycle 7, the UT sets the payload data valid flag for all 4 configured transmit buffers to true.

3) In cycle 8, it is verified (LT) that the IUT sends its frames in the static slots 3 and 4 and in the dynamic slots n_{dyn} and (n_{dyn} + 1) correctly, i.e. the null frame indicator is set to '1'.

4) In cycle 9, it is verified (LT) that the IUT sends its frame in static slot 3 as null frame (the null frame indicator is set to '0', the payload preamble indicator is set to '0' and all payload bytes are 0x00), that the IUT sends its frames in the static slot 4 and in the dynamic slot (n_{dyn} + 1) correctly (i.e. the null frame indicator is set to '1') and that the IUT does not send in the dynamic slot n_{dyn}.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames in cycles 7, 8 and 9 as expected.

7.3.8.1.14 Frame transmitted indicator

— **Test purpose**

Verify correct setting of the frame transmitted indicator (*vSS!FrameSent*).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

Two additional buffers are configured for transmission in static slot 3 and in the first dynamic slot n_{dyn} (for dual channel test execution on channel A) with all payload bytes set to 0x55.

The payload data valid flag in the transmit buffer configuration and the frame transmitted indicator in the transmit buffer status are both set to false during buffer configuration. The buffers are configured to operate in single shot transmission mode.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In cycle 7, it is verified (LT) that the IUT sends its frame in static slot 3 as null frames (the null frame indicator is set to '0', the payload preamble indicator is set to '0' and all payload bytes are 0x00), and that the IUT does not send in the dynamic slot n_{dyn}.

2) In the NIT of cycle 7, it is verified (UT) that the frame transmitted indicator in the transmit buffer status is false for both transmit buffers. The UT sets the payload data valid flag for both transmit buffers to true.

3) In cycle 8, it is verified (LT) that the IUT sends its frames in slot 3 and in slot n_{dyn} with the correct payload.

4) In the NIT of cycle 8, it is verified (UT) that the frame transmitted indicator in the transmit buffer status is true for both transmit buffers. The UT sets the frame transmitted indicator in the transmit buffer status for slot 3 and the payload data valid flag for both buffers to false.

5) In cycle 9, is verified (LT) that the IUT sends a null frame in slot 3 and no frame in slot n_{dyn}.

6) In the NIT of cycle 9, it is verified (UT) that the frame transmitted indicator in the transmit buffer status is false for slot 3 and true for slot n_{dyn}.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The frame transmitted indicator (*vSS!FrameSent*) in the transmit buffer status is set accordingly.

7.3.8.1.15 Active message buffer identification

— **Test purpose**

If the IUT allows that multiple buffers are configured for the same slot in a specific communication cycle, then verify that the IUT selects a unique buffer in a deterministic way which is predictable.

NOTE This test case is implementation-dependent and the execution steps 4 and 5 must be adapted according to the IUT's specification.

— **Applicability**

SC, DC.

Devices able to have two transmit / receive buffers for transmission / reception assigned to the same slot in a specific communication cycle.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

The UT configures 9 message buffers for reception and transmission. If the number of configurable buffers is less than 9, then this test shall be performed in several instances with buffers configured for the same slot placed into the same instance.

The buffers are referred as B1 to B9 later in this test. The transmit buffer B1 is configured for transmission of startup frames in static slot 2.

The payload data valid flag of all transmit buffers are set to false during buffer configuration.

Four transmit buffers are configured in the IUT for transmission in odd cycle. The buffers B2 and B3 are configured for the static slot 3 and continuous transmission mode. The buffers B4 and B5 are configured for the dynamic slot n_{dyn} and single shot transmission mode. The buffer number shall be applied in each nibble of each payload byte (i.e. payload of B2 is 0x22 in all bytes, payload of B3 is 0x33 in all bytes, ...). The payload data valid flags of all transmit buffers are set to true in the NIT of cycle 7.

Four receive buffers are configured in the IUT for reception in every cycle. The receive buffers B6 and B7 are configured for the static slot 3, the receive buffers B8 and B9 are configured for the dynamic slot n_{dyn}.

Table 199 defines the message buffer configuration.

Table 199 — Message buffer configuration for frame transmitted indicator

Buffer	SlotId	Cycle_Offset	Cycle_Repetition	Transmission mode
Tx B1	2	0	1	don't care
Tx B2	3	1	2	continuous mode
Tx B3	3	1	2	continuous mode
Tx B4	n_{dyn}	1	2	single shot mode
Tx B5	n_{dyn}	1	2	single shot mode
Rx B6	3	0	1	-
Rx B7	3	0	1	-
Rx B8	n_{dyn}	0	1	-
Rx B9	n_{dyn}	0	1	-

For dual channel test execution the test has to be performed repeatedly with the buffers B4, B5, B8, and B9 configured (a) for channel A and (b) for channel B.

— **Preamble (setup state)**

Preamble I.

In the NIT of cycle 7, the UT set the payload data valid flags of all transmit buffers to true.

— **Test execution**

1) In cycle 8, the LT transmits two additional valid frames with the default payload in static slot 3 and in dynamic slot n_{dyn}.

2) In the NIT of cycle 8, it is verified (UT) that the IUT has received the frames transmitted in slot 3 and in slot n_{dyn}, and has stored the frames in the correct receive buffers as specified in IUT's specification.

3) In cycle 9, it is verified (LT) that the IUT sends frames in slot 3 and in slot n_{dyn}, and that the payload of each frame matches the payload of correct transmit buffer as specified in IUT's specification.

4) In cycle 10, it is verified (LT) that the IUT sends a frame in the slot 2 only.

5) In cycle 11, it is verified (LT) that the IUT sends frames in slot 3 and in slot n_{dyn}, and that the payload of the frame in slot 3 matches the payload received in cycle 9. It is also verified (LT) that the received frame in the dynamic slot n_{dyn}, differs from the received frame in the same slot in cycle 9 and matches the contents of the other configured transmit buffer for this slot in IUT (if in slot n_{dyn} of cycle 9 the payload of received frame was 0x44 in all bytes, then the payload of received frame in slot n_{dyn} of cycle 11 is 0x55 in all bytes and vice versa).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT selects a unique buffer for transmission or reception in a deterministic way and as specified in IUT's specification.

7.3.8.1.16 Transmit buffer assignment via cycle repetition and cycle offset

— **Test purpose**

Verify correct transmission of frames in the configured slots and within the configured communication cycles. The IUT shall transmit frames in the appropriate slots and cycles.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

Eleven additional transmit buffers are configured in continuous transmission mode with the configurations and payloads given in the following table.

Table 200 defines the message buffer configuration.

Table 200 — Message buffer configuration for transmit buffer assignment via cycle repetition and cycle offset

Tx Buffer	SlotId	Payload	Cycle_ Repetition	Cycle_ Offset	Transmission in (cycle)
B1	2	0x1111	1	0	all cycles
B2	3	0x2222	2	1	odd cycles
B3	4	0x3333	4	3	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63
B4	5	0x4444	5	4	4, 9, 14, 19, 24, 29, 34, 39, 44, 49, 54, 59
B5	6	0x5555	8	6	6, 14, 22, 30, 38, 46, 54, 62
B6	7	0x6666	10	7	7, 17, 27, 37, 47, 57
B7	8	0x7777	16	10	10, 26, 42, 58
B8	9	0x8888	20	16	16, 36, 56
B9	10	0x9999	32	25	25, 57
B10	11	0xAAAA	40	30	30
B11	12	0xBBBB	50	48	48
B12	13	0xCCCC	64	63	63

NOTE 1 If the IUT supports less than 12 transmit buffers, this test case shall be executed multiple times until all configurations are tested.

NOTE 2 Deviating from the preamble the IUT's payload in slot 2 is as given in the configuration (0x1111).

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In each cycle (beginning with cycle 8), it is verified (LT) that the IUT transmits its frames correctly according to the given configuration (e.g. in cycle 8 IUT shall send one startup frame in slot 2, with the correct frame ID and payload according to the given configuration).

2) The step 1 is repeated until cycle 8 is reached after the first wraparound of cycle counter.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits its frames according to the buffer configuration of this test case and no transmission shall take place beyond this configuration.

7.3.8.1.17 Reception and transmission in the same static slot in different cycles

— **Test purpose**

Verify transmission and reception of frames in the same static slot.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

Two transmit buffers in IUT are set up for transmission in slot 5 with different payload data, the first one for cycle 8 (the payload byte 0 is 0x33, byte 1 is 0x33), the second one for cycle 10 (the payload byte 0 is 0xCC, byte 1 is 0xCC).

Two receive buffers in the IUT are assigned to slot 5, one for cycle 9 and the other one for cycle 11.

Table 201 defines the message buffer configuration.

Table 201 — Message buffer configuration for reception and transmission in the same static slot in different cycles

Buffer	SlotId	Cycle_Offset	Cycle_Repetition
Tx B1	5	8	64
Tx B2	5	10	64
Rx B3	5	9	64
Rx B4	5	11	64

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, it is verified (LT) that the IUT transmits its frame in slot 5 with the payload data as given in the configuration.

2) In cycle 9, the LT simulates an additional frame in slot 5 (the payload byte 0 is 0x01, byte 1 is 0x02).

3) In the NIT of cycle 9, it is verified (UT) that the receive buffer assigned to slot 5 / cycle 9 holds the frame contents (frame header and payload data) in the frame contents data as simulated by the LT in slot 5 / cycle 9.

4) In cycle 10, it is verified (LT) that the IUT transmits its frame in slot 5 with the payload data as given in the configuration.

5) In cycle 11, the LT simulates an additional frame in slot 5 (the payload byte 0 is 0x0E, byte 1 is 0x0D).

6) In the NIT of cycle 11, it is verified (UT) that the receive buffer assigned to slot 5 / cycle 11 holds the frame contents (frame header and payload data) in the frame contents data as simulated by the LT in slot 5 / cycle 11.

7) In cycle 12, it is verified (LT) that the IUT does not send a frame in slot 5.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits and receives the frames in slot 5 as given by the buffer configuration.

7.3.8.2 Receive buffer

7.3.8.2.1 Receive buffer configuration by slot and channel in the static segment

— **Test purpose**

Verify frame reception into the assigned receive buffer. The IUT shall identify the assigned receive buffer on a slot / channel basis. A receive buffer is assigned to both channels. If on both channels valid frames are received then the receive buffer shall store the frame that was received on channel A.

— **Applicability**

DC.

— **Configuration**

All basic configurations.

Deviating from the Preamble the LT's payload data is 0x01 in byte 0 and 0x02 in byte 1 on channel A, 0x10 in byte 0 and 0x20 in byte 1 on channel B. The LT simulates startup frames in slot 1 on both channels. A receive buffer in the IUT is assigned to slot 1.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

The test has to be performed repeatedly with the receive buffer assigned (a) to channel A, (b) to channel B and (c) to both channels.

1) In cycle 8, the LT simulates the frame in slot 1 on channel A starting at *gdActionPointOffset* – 1 MT and the frame in slot 1 on channel B starting at *gdActionPointOffset*.

2) In the NIT of cycle 8, it is verified (UT) that the receive buffer holds in frame contents data the contents (frame header and payload data) of

   (a) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   (b) the frame from slot 1, and the channel indicator signals reception on channel B (*vRF!Channel* = B).

   (c) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot 1.

3) In cycle 9, the LT simulates the frame in slot 1 on channel A starting at *gdActionPointOffset* and the frame on channel B starting at *gdActionPointOffset* – 1 MT.

4) In the NIT of cycle 9, it is verified (UT) that the receive buffer holds in frame contents data the contents (frame header and payload data) of

   (a) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   (b) the frame from slot 1, and the channel indicator signals reception on channel B (*vRF!Channel* = B).

   (c) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot 1.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag *vSS!SyntaxError*, the content error flag *vSS!ContentError*, the boundary violation flag *vSS!BViolation*, the valid frame flag *vSS!ValidFrame,* the flag *vSS!NFIndicator* in the slot status data and the channel indicator *vRF!Channel* in the frame contents data are set accordingly and the received frame contents are as expected.

7.3.8.2.2 Receive buffer configuration by slot, channel and cycle in the static segment

— Test purpose

Verify frame reception into assigned receive buffers depending on the cycle during which frames were received. The IUT shall identify the assigned receive buffer on a slot / channel / cycle basis. A receive buffer is assigned to both channels. If on both channels valid frames are received than the receive buffer shall store the frame that was received on channel A.

— Applicability

DC.

— Configuration

All basic configurations.

Deviating from the Preamble the LT's default payload data is 0x00 in byte 0 and 0x00 in byte 1. In cycle 8, the LT's payload data is 0x01 in byte 0 and 0x02 in byte 1 on channel A and 0x10 in byte 0 and 0x20 in byte 1 on channel B. In cycle 9, the LT's payload data is 0x0E in byte 0 and 0x0D in byte 1 on channel A and 0xE0 in byte 0 and 0xD0 in byte 1 on channel B. The LT simulates startup frames in slot 1 on both channels. One receive buffer in the IUT is assigned to slot 1 / cycle 8 and another one to slot 1 / cycle 9.

— Preamble

Preamble I.

— Test execution

The test has to be performed repeatedly with both receive buffers assigned (a) to channel A, (b) to channel B and (c) to both channels.

1) In cycle 8, the LT simulates the frame in slot 1 on channel A starting at *gdActionPointOffset* – 1 MT and the frame on channel B starting at *gdActionPointOffset*.

2) In the NIT of cycle 8, it is verified (UT) that the receive buffer assigned to slot 1 / cycle 9 was not updated (neither buffer contents nor buffer related status flags were altered in comparison to the initial buffer contents / flags) and that the receive buffer assigned to slot 1 / cycle 8 holds in frame contents data the contents (frame header and payload data) of

   (a) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   (b) the frame from slot 1, and the channel indicator signals reception on channel B (*vRF!Channel* = B).

   (c) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot 1.

3) In cycle 9, the LT simulates the frame in slot 1 on channel A starting at *gdActionPointOffset* and the frame on channel B starting at *gdActionPointOffset* – 1 MT.

4) In the NIT of cycle 9, it is verified (UT) that the receive buffer assigned to slot 1 / cycle 8 was not updated (neither buffer contents nor buffer related status flags were altered in comparison to the buffer contents / flags in the previous cycle) and that the receive buffer assigned to slot 1 / cycle 9 holds in frame contents data the contents (frame header and payload data) of

   (a) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   (b) the frame from slot 1, and the channel indicator signals reception on channel B (*vRF!Channel* = B).

   (c) the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A).

   It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot 1.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag *vSS!SyntaxError*, the content error flag *vSS!ContentError*, the boundary violation flag *vSS!BViolation*, the valid frame flag *vSS!ValidFrame*, the flag *vSS!NFIndicator* in the slot status data and the channel indicator *vRF!Channel* in the frame contents data are set accordingly and the received frame contents are as expected.

7.3.8.2.3 Receive buffer configuration with specific size in the static segment

— Test purpose

Verify frame reception into the assigned receive buffer configured to hold a specific number of two-byte words.

— Applicability

SC, DC.

Variant (a) for a implementation that is able to configure a buffer length of 1 two-byte words.

Variant (b) for a implementation that is able to configure a buffer length of 64 two-byte words.

Variant (c) for a implementation that is able to configure a buffer length of 127 two-byte words.

Variant (d) for a implementation that is not able to configure a buffer length at all

An implementation that is able to execute variant (b) shall also execute variant (a).

An implementation that is able to execute variant (c) shall also execute variant (b) and (a).

— Configuration

All basic configurations using the modifications as listed in Table 202.

Table 202 — Modification to basic configurations for receive buffer – receive buffer configuration with specific size in the static segment

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdStaticSlot [MT]	274	271	331
gNumberOfStaticSlots	10	5	4
gPayloadLengthStatic [two-byte word]	127	127	76
gdNIT [MT]	249	142	137

For *gPayloadLengthStatic* = 127 two-byte word the LT's payload data is 0x01 in byte 0, 0x02 in byte 1, ..., and 0xFE in byte 253 on channel A, 0xFE in byte 0, 0xFD in byte 1, ..., and 0x01 in byte 253 on channel B and for *gPayloadLengthStatic* = 76 two-byte word the LT's payload data is 0x01 in byte 0, 0x02 in byte 1, ..., and 0x98 in byte 151 on channel A, 0x98 in byte 0, 0x97 in byte 1, ..., and 0x01 in byte 151 on channel B.

The LT simulates startup frames in slot 1 on available channel(s). For each available channel a receive buffer is configured for slot 1.

— Preamble (setup state)

Preamble I.

— **Test execution**

The test has to be performed repeatedly with the size of the receive buffers set to (a) 1, (b) 64 and (c) 127 two-byte words.

1) In cycle 8, the LT simulates its startup frame in slot 1 with the given payload.

2) In the NIT of cycle 8, it is verified (UT) that the receive buffer assigned to slot 1 holds

   (a) the first two-byte word,

   (b) the first 64 two-byte words,

   (c) all two-byte words as defined in configuration modification and

   (d) - the first number of BufferSize two-byte word if *BufferSize* <= *gPayloadLengthStatic*,
   - all two-byte words as defined in configuration modification if *BufferSize* > *gPayloadLengthStatic*,

   of the payload of the startup frame received on the respective channel.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT provides the proper payload data to the UT. Each receive buffer holds an individually configurable number of two-byte words.

7.3.8.2.4 Receive buffer configuration by slot and channel in the dynamic segment

— **Test purpose**

Verify frame reception into the assigned receive buffer. The IUT shall identify the assigned receive buffer on a slot / channel basis.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

Deviating from the Preamble the LT's payload data (for the frame in slot n_{dyn}) is 0x01 in byte 0 and 0x02 in byte 1 on channel A, 0x10 in byte 0 and 0x20 in byte 1 on channel B. The LT simulates startup frames in slot 1 and dynamic frames in slot n_{dyn} on available channel(s). A receive buffer in the IUT is assigned to slot n_{dyn}.

For dual channel test execution the test has to be performed repeatedly with the receive buffer assigned (a) to channel A and (b) to channel B.

— **Preamble (setup state)**

Preamble I.

— Test execution

1) In cycle 8, the LT simulates dynamic frames in slot n_{dyn} on available channel(s).

2) In the NIT of cycle 8, it is verified (UT) that the receive buffer holds the contents (frame header and payload data) of the frame from slot n_{dyn}, and the channel indicator signals reception on the respective channel (for dual channel test execution (a) *vRF!Channel* = A, (b) *vRF!Channel* = B). It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true and the flag *vSS!NFIndicator* is '1' in the slot status data of slot n_{dyn}.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The syntax error flag *vSS!SyntaxError*, the content error flag *vSS!ContentError*, the boundary violation flag *vSS!BViolation*, the valid frame flag *vSS!ValidFrame* and the flag *vSS!NFIndicator* in the slot status data and the channel indicator *vRF!Channel* in the frame contents data are set accordingly and the received frame contents are as expected.

7.3.8.2.5 Receive buffer configuration by slot, channel and cycle in the dynamic segment

— Test purpose

Verify frame reception into assigned receive buffers depending on the cycle during which frames were received. The IUT shall identify the assigned receive buffer on a slot / channel / cycle basis.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

Deviating from the preamble the LT's payload data for slot n_{dyn} is 0x01 in byte 0 and 0x02 in byte 1 on channel A and 0x10 in byte 0 and 0x20 in byte 1 on channel B in cycle 8. In cycle 9, the LT's payload data for the slot n_{dyn} is 0x0E in byte 0 and 0x0D in byte 1 on channel A and 0xE0 in byte 0 and 0xD0 in byte 1 on channel B. The LT simulates startup frames in slot 1 and dynamic frames in slot n_{dyn} on available channel(s). One receive buffer in the IUT is assigned to slot n_{dyn} / cycle 8 and another one to slot n_{dyn} / cycle 9.

For dual channel test execution the test has to be performed repeatedly with both receive buffers assigned (a) to channel A and (b) to channel B.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates dynamic frames in slot n_{dyn} on available channel(s).

2) In the NIT of cycle 8, it is verified (UT) that the receive buffer assigned to slot n_{dyn} / cycle 9 was not updated (holds the initial contents) and that the receive buffer assigned to slot n_{dyn} / cycle 8 holds in

frame contents data the contents (frame header and payload data) of the frame from slot n_{dyn}, and the channel indicator signals reception on the respective channel (for dual channel test execution (a) *vRF!Channel* = A, (b) *vRF!Channel* = B).
It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot n_{dyn}.

3) In cycle 9, the LT simulates dynamic frames in slot n_{dyn} on available channel(s).

4) In the NIT of cycle 9, it is verified (UT) that the receive buffer assigned to slot n_{dyn} / cycle 8 was not updated (holds the contents from the previous cycle) and that the receive buffer assigned to slot n_{dyn} / cycle 9 holds in frame contents data the contents (frame header and payload data) of the frame from slot n_{dyn}, and the channel indicator signals reception on the respective channel (for dual channel test execution (a) *vRF!Channel* = A, (b) *vRF!Channel* = B).
It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true and that the flag *vSS!NFIndicator* is '1' in the slot status data of slot n_{dyn}.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag *vSS!SyntaxError*, the content error flag *vSS!ContentError*, the boundary violation flag *vSS!BViolation*, the valid frame flag *vSS!ValidFrame* and the null frame indicator flag *vSS!NFIndicator* in the slot status data and the channel indicator *vRF!Channel* in the frame contents data are set accordingly and the received frame contents are as expected.

7.3.8.2.6 Receive buffer assignment via cycle repetition and cycle offset

— **Test purpose**

Verify correct reception of frames in the buffers configured for specific slots within specific communication cycles. The IUT shall store the received frames in the appropriate buffer.

— **Applicability**

SC, DC.

— **Configuration**

All Basic Configurations.

Twelve receive buffers are configured in the IUT with the configurations given in the following table.

Table 203 defines the message buffer configuration.

Table 203 — FIFO admittance criteria for receive buffer assignment via cycle repetition and cycle offset

Rx Buffer	SlotId	Cycle_ Repetition	Cycle_Offset	Reception in (cycle)
B1	1	1	0	all cycles
B2	3	2	1	odd cycles
B3	4	4	3	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63
B4	5	5	4	4, 9, 14, 19, 24, 29, 34, 39, 44, 49, 54, 59
B5	6	8	6	6, 14, 22, 30, 38, 46, 54, 62
B6	7	10	7	7, 17, 27, 37, 47, 57
B7	8	16	10	10, 26, 42, 58
B8	9	20	16	16, 36, 56
B9	10	32	25	25, 57
B10	11	40	30	30
B11	12	50	48	48
B12	13	64	63	63

NOTE 1 If the device supports less than 12 receive buffers or the UT can not verify all buffers in the NIT of each cycle, this test case shall be executed multiple times until all configurations are tested.

NOTE 2 For dual channel test execution all 12 receive buffers shall be configured for reception of both channels (A&B).

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In each cycle (beginning with cycle 7), the LT simulates additional frames in all slots between 3 and 13. For all simulated frames (1 and 3 to 13) the first byte of payload is equal to cycle number the second byte equal to SlotId.

2) In the NIT of each cycle (beginning with cycle 7), it is verified (UT) that the IUT has only stored the frames from the configured slots for this communication cycle according to the given configuration (e.g., in cycle 7 IUT shall store only the frames from the slots 1, 3, 4 and 7).
The slot status update flag shall be set for all buffers configured to receive frames in this cycle and shall be reset for all other buffers.
The payload in the frame contents data of buffers shall be as expected (the first byte equal to cycle number and the second byte equal to SlotId according to the configuration table).
The UT clears the slot status update flags of active buffers in this cycle.

The step 1 and 2 are repeated until cycle 7 is reached after the first wraparound of cycle counter.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The slot status update flags and the payload in the frame contents data in the receive buffers are as expected.

7.3.8.2.7 Buffer updated flag in frame contents data

— Test purpose

Verify the flag that indicates that the buffer has been updated at some point during operation. This flag should be set when the buffer is updated as a result of reception of a valid, non-null frame.

— Applicability

SC, DC.

— Configuration

All basic configurations.

A receive buffer in the IUT is assigned to static slot 15.

— Preamble

Preamble I.

— Test execution

1) In cycle 8, 500 µT after cycle start and before static slot 15, it is verified (UT) that the flag that indicates that the buffer has been updated at some point during operation is '0' in the frame contents data of slot 15.

2) In cycle 8, the LT simulates an additional frame in slot 15 with the null frame indicator set to '0' and all payload bytes set to 0x00.

3) In the NIT of cycle 8, it is verified (UT) that the flag that indicates that the buffer has been updated at some point during operation is '0' in the frame contents data of slot 15.

4) In cycle 9, the LT simulates an additional frame in slot 15 with the null frame indicator set to '1'.

5) In the NIT of cycle 9, it is verified (UT) that the flag that indicates that the buffer has been updated at some point during operation is '1' in the frame contents data of slot 15.

6) In cycle 10, the LT simulates an additional frame in slot 15 with the null frame indicator set to '0' and all payload bytes set to 0x00.

7) In the NIT of cycle 10, it is verified (UT) that the flag that indicates that the buffer has been updated at some point during operation is '1' in the frame contents data of slot 15.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The flag that indicates that the buffer has been updated at some point during operation is set accordingly.

7.3.8.2.8 Receiving null and non-null frame in the same slot on different channels

— Test purpose

Verify that the a receive buffer configured for both channels contains the valid non-null frame received on either channel A or on channel B if on the other channel a valid null frame was received within the same slot.

— **Applicability**

DC.

— **Configuration**

All basic configurations.

The LT simulates startup frames in slot 1 on both channels. Deviating from the preamble the LT transmit in cycle 8 a non-null frame on channel A, a null-frame on channel B and in cycle 9, a null-frame on channel A and non-null frame on channel B.

One receive buffer in the IUT is configured for slot 1 / cycle 8 and another one for slot 1 / cycle 9.

— **Preamble**

Preamble I.

— **Test execution**

The test has to be performed repeatedly with both receive buffers assigned (a) to channel A, (b) to channel B and (c) to both channels.

1) In cycle 8, the LT simulates the frame in slot 1 on channel A starting at *gdActionPointOffset* and the frame on channel B starting at *gdActionPointOffset* – 1 MT.

2) In the NIT of cycle 8, it is verified (UT) that the receive buffer assigned to slot 1 / cycle 9 was not updated (neither buffer contents nor buffer related status flags were altered in comparison to the initial buffer contents / flags) and that the receive buffer assigned to slot 1 / cycle 8
(a, c) holds in frame contents data the contents (frame header and payload data) of the frame from slot 1, and the channel indicator signals reception on channel A (*vRF!Channel* = A) and *vRF!Header!NFIndicator* = 1.
(b) was not updated, *vSS!NFIndicator* = 0 in the slot status data.
It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, and the boundary violation flag *vSS!BViolation* is false, that the valid frame flag *vSS!ValidFrame* is true in the slot status data of slot 1.

3) In cycle 9, the LT simulates the frame in slot 1 on channel A starting at *gdActionPointOffset* – 1 MT and the frame on channel B starting at *gdActionPointOffset*.

4) In the NIT of cycle 9, it is verified (UT) that the receive buffer assigned to slot 1 / cycle 8 was not updated (neither buffer contents nor buffer related status flags were altered in comparison to the buffer contents / flags in the previous cycle) and that the receive buffer assigned to slot 1 / cycle 9
(a) was not updated, *vSS!NFIndicator* = 0 in the slot status data.
(b, c) holds in frame contents data the contents (frame header and payload data) of the frame from slot 1, and the channel indicator signals reception on channel B (*vRF!Channel* = B) *vRF!Header!NFIndicator* = 1.
It is also verified (UT) that the syntax error flag *vSS!SyntaxError* is false, the content error flag *vSS!ContentError* is false, the boundary violation flag *vSS!BViolation* is false and that the valid frame flag *vSS!ValidFrame* is true in the slot status data of slot 1.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The syntax error flag *vSS!SyntaxError*, the content error flag *vSS!ContentError*, the boundary violation flag *vSS!BViolation*, the valid frame flag *vSS!ValidFrame*, the flag *vSS!NFIndicator* in the slot status data

and the channel indicator *vRF!Channel* in the frame contents data are set accordingly and the received frame contents are as expected.

7.3.8.3 Queued receive buffer

7.3.8.3.1 FIFO frame validity admittance criteria

— Test purpose

Verify the FIFO frame validity admittance criteria. The behaviour on reception of a frame with thenull frame indicator set to '0' in the static segment is defined by AdmitNullFrame.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

A FIFO buffer is configured in the IUT to receive all valid non-null frames in static slot 5 and dynamic slot n_{dyn} (AdmitNullFrame = false, $Range1_{min}$ = 5, $Range1_{max}$ = 5, $Range2_{min}$ = n_{dyn}, $Range2_{max}$ = n_{dyn}). Write access to the admittance criteria is enabled during *POC:normal active* state. No non-queued receive buffers in the IUT are assigned for reception in static slot 5 and dynamic slot n_{dyn}.

The LT simulates an additional frame in static slot 5 and an additional frame in dynamic slot n_{dyn}.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT transmits its frame in slot 5 with the null frame indicator set to '1' and with the frame ID set to 5 and its frame in slot n_{dyn} with the null frame indicator set to '1'.

2) In the NIT of cycle 8, it is verified (UT) that the FIFO buffer holds the contents of the frames (frame contents data) as simulated by the LT in slot 5 and slot n_{dyn} of cycle 8.

3) In cycle 9, the LT transmits its frame in slot 5 with the null frame indicator set to '0' and with the frame ID set to 5 and its frame in slot n_{dyn} with the null frame indicator set to '0'.

4) In the NIT of cycle 9, it is verified (UT) that the FIFO buffer does not hold the contents of the frames as simulated by the LT in slot 5 and slot n_{dyn} of cycle 9.

5) In cycle 10, the LT transmits a frame in slot 5 with the null frame indicator set to '1' and with the frame ID set to 0 and its frame in slot n_{dyn} with the null frame indicator set to '1'.

6) In the NIT of cycle 10, it is verified (UT) that the FIFO buffer holds the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 10.

7) In the NIT of cycle 10, the UT configures AdmitNullFrame = true.

8) In cycle 11, the LT transmits its frame in slot 5 with the null frame indicator set to '1' and with the frame ID set to 5 and its frame in slot n_{dyn} with the null frame indicator set to '1'.

9) In the NIT of cycle 11, it is verified (UT) that the FIFO buffer holds the contents of the frames (frame contents data) as simulated by the LT in slot 5 and slot n_{dyn} of cycle 11.

10) In cycle 12, the LT transmits its frame in slot 5 with the null frame indicator set to '0' and with the frame ID set to 5 and its frame in slot n_{dyn} with the null frame indicator set to '0'.

11) In the NIT of cycle 12, it is verified (UT) that the FIFO buffer holds the contents of the frame (frame contents data) as simulated by the LT in slot 5 of cycle 12.

12) In cycle 13, the LT transmits a frame in slot 5 with the null frame indicator set to '0' and with the frame ID set to 0 and its frame in slot n_{dyn} with the null frame indicator set to '1'.

13) In the NIT of cycle 13, it is verified (UT) that the FIFO buffer holds the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 13.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs proper valid frame filtering.

7.3.8.3.2 FIFO channel admittance criteria

— **Test purpose**

Verify the FIFO channel admittance criteria. Only frames which pass the channel admittance criteria are admitted to the configured FIFO buffer.

— **Applicability**

DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

If the IUT supports the configuration of one single FIFO for both channels configuration (a) applies to this test case.

If the IUT does not support the configuration of one single FIFO for both channels configuration (b) applies to this test case.

(a) A FIFO buffer (referred as FIFO1 later in this test) is configured in the IUT to receive all valid non-null frames in static slot 5 and dynamic slot n_{dyn} ($Range1_{min}$ = 5, $Range1_{max}$ = 5, $Range2_{min}$ = n_{dyn}, $Range2_{max}$ = n_{dyn}) on channel A (a frame pass the channel admittance criteria only if the frame was received on channel A). Write access to the admittance criteria is enabled during *POC:normal active* state.

(b) One FIFO buffer (referred as FIFO1 later in this test) is configured in the IUT to receive all valid non-null frames in static slot 5 and dynamic slot n_{dyn} ($Range1_{min}$ = 5, $Range1_{max}$ = 5, $Range2_{min}$ = n_{dyn}, $Range2_{max}$ = n_{dyn}) on channel A (a frame pass the channel admittance criteria only if the frame was received on channel A). And another FIFO buffer (referred as FIFO2 later in this test) is configured in the IUT to receive all valid non-null frames in static slot 5 and dynamic slot n_{dyn} of cycle 10 (Cycle_Repetition = 64, Cycle_Offset = 10, $Range1_{min}$ = 5, $Range1_{max}$ = 5, $Range2_{min}$ = n_{dyn}, $Range2_{max}$ = n_{dyn}) on channel B (a frame pass the channel admittance criteria only if the frame was

received on channel B in cycle 10). Write access to the admittance criteria is enabled for both FIFOs during *POC:normal active* state.

No non-queued receive buffers in the IUT are assigned for reception in static slot 5 and dynamic slot n_{dyn}.

The LT simulates an additional frame in static slot 5 and an additional frame in dynamic slot n_{dyn} on channel A and on channel B correctly beginning with cycle 8.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In the NIT of cycle 8, it is verified (UT) that FIFO1 holds the contents of the frames (frame contents data) as simulated by the LT on channel A in slot 5 and slot n_{dyn} of cycle 8.

2) In the NIT of cycle 8, the UT configures FIFO1 in the IUT so that a frame passes the channel admittance criteria only if the frame was received on channel B.

3) In the NIT of cycle 9, it is verified (UT) that FIFO1 holds the contents of the frames (frame contents data) as simulated by the LT on channel B in slot 5 and slot n_{dyn} of cycle 9.

4) In the NIT of cycle 9, the UT configures

   (a) FIFO1 in the IUT so that a frame passes the channel admittance criteria regardless of whether the frame was received on channel A or on channel B.

   (b) FIFO1 in the IUT so that a frame passes the channel admittance criteria only if the frame was received on channel A.

5) In the NIT of cycle 10, it is verified (UT) that

   (a) FIFO1 holds the contents of the frames (frame contents data) as simulated by the LT on channel A and on channel B in slot 5 and slot n_{dyn} of cycle 10.

   (b) FIFO1 holds the contents of the frames (frame contents data) as simulated by the LT on channel A in slot 5 and slot n_{dyn} of cycle 10 and FIFO2 holds the contents of the frames (frame contents data) as simulated by the LT on channel B in slot 5 and slot n_{dyn} of cycle 10

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stores the frame content into the FIFO correctly and according to its admittance criteria.

7.3.8.3.3 Received frame placed into none-queued receive buffer / FIFO buffer

— **Test purpose**

Verify that if the received frame is not placed into a non-queued receive buffer, the frame is a candidate for admission into a FIFO buffer.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

A FIFO buffer is configured in the IUT to receive all valid non-null frames in static slot 5 and in the dynamic slot n_{dyn} ($Range1_{min}$ = 5, $Range1_{max}$ = 5, $Range2_{min}$ = n_{dyn}, $Range2_{max}$ = n_{dyn}) on available channel(s).Write access to the admittance criteria is enabled during *POC:normal active* state.

(a) No non-queued receive buffers in the IUT are assigned for reception in slot 5 and dynamic slot n_{dyn} on available channel(s).

(b) Non-queued receive buffers in the IUT are assigned for reception in static slot 5 and dynamic slot n_{dyn} on available channel(s).

The LT simulates an additional frame in static slot 5 and an additional frame in dynamic slot n_{dyn}.

NOTE For dual channel test execution and in case the IUT does not support the configuration of one single FIFO for both channels, two FIFO buffers shall be configured and the verification steps in the test execution shall apply to the respective FIFO.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates its frames in slot 5 and slot n_{dyn} with the null frame indicator set to '1'.

2) In the NIT of cycle 8, it is verified (UT) that

3) the FIFO buffer holds the contents of the frames (frame contents data) as simulated by the LT in slot 5 and slot n_{dyn} of cycle 8.

4) the FIFO buffer is empty and that the non-queued receive buffer(s) assigned to slot 5 holds the contents of the frame (frame contents data) from slot 5 and the non-queued receive buffer(s) assigned to slot n_{dyn} holds the contents of the frame from slot n_{dyn} as simulated by the LT in cycle 8.

5) In cycle 9, the LT simulates its frames in slot 5 and slot n_{dyn} with the null frame indicator set to '0' and all payload bytes set to 0x00.

6) In the NIT of cycle 9, it is verified (UT) that

   (a) the FIFO buffer does not hold the contents of the frames (frame contents data) as simulated by the LT in slot 5 and slot n_{dyn} of cycle 9.

   (b) the FIFO buffer is empty and that the non-queued receive buffer(s) assigned to slot 5 still holds the contents of the frame (frame contents data) as simulated by the LT in slot 5 of cycle 8. It is also verified (UT) that the non-queued receive buffer(s) assigned to slot n_{dyn} still holds the contents of the frame (frame contents data) as simulated by the LT in slot n_{dyn} of cycle 8. (The receive buffers have not been updated.)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The received frame is placed into the non-queued receive buffer or into the FIFO receive buffer accordingly.

7.3.8.3.4 FIFO cycle counter admittance criteria

— **Test purpose**

Verify FIFO cycle counter admittance criteria.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

A FIFO buffer is configured in the IUT to receive all valid non-null frames in static slot 5 and in the dynamic slot n_{dyn} in every even cycle (Cycle_Repetition = 2, Cycle_Offset = 0, $Range1_{min}$ = 5, $Range1_{max}$ = 5, $Range2_{min}$ = n_{dyn}, $Range2_{max}$ = n_{dyn}). Write access to the admittance criteria is enabled during *POC:normal active* state. No non-queued receive buffers in the IUT are assigned for reception in slot 5 and dynamic slot n_{dyn}.

The LT simulates an additional frame in static slot 5 and an additional frame in dynamic slot n_{dyn}.

NOTE For dual channel test execution and in case the IUT does not support the configuration of one single FIFO for both channels, two FIFO buffers shall be configured and the verification steps in the test execution shall apply to the respective FIFO.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT transmits its frames in slot 5 and slot n_{dyn} correctly.

2) In the NIT of cycle 8, it is verified (UT) that the FIFO buffer holds the contents of the frames (frame contents data) as simulated by the LT in slot 5 and slot n_{dyn} of cycle 8.

3) In cycle 9, the LT transmits its frames in slot 5 and slot n_{dyn} correctly.

4) In the NIT of cycle 9, it is verified (UT) that the FIFO buffer does not hold the contents of the frames as simulated by the LT in slot 5 and slot n_{dyn} of cycle 9.

5) In cycle 10, the LT transmits its frames in slot 5 and slot n_{dyn} correctly.

6) In the NIT of cycle 10, it is verified (UT) that the FIFO buffer holds the contents of the frames (frame contents data) as simulated by the LT in slot 5 and slot n_{dyn} of cycle 10.

7) In the NIT of cycle 10, the UT configures the IUT to receive valid non-null frames in static slot 5 and in the dynamic slot n_{dyn} in every odd cycle (Cycle_Repetition = 2, Cycle_Offset = 1).

8) In cycle 11, the LT transmits its frames in slot 5 and slot n_{dyn} correctly.

9) In the NIT of cycle 11, it is verified (UT) that the FIFO buffer holds the contents of the frames (frame contents data) as simulated by the LT in slot 5 and slot n_{dyn} of cycle 11.

10) In cycle 12, the LT transmits its frames in slot 5 and slot n_{dyn} correctly.

11) In the NIT of cycle 12, it is verified (UT) that the FIFO buffer does not hold the contents of the frames as simulated by the LT in slot 5 and slot n_{dyn} of cycle 12.

12) In cycle 13, the LT transmits its frames in slot 5 and slot n_{dyn} correctly.

13) In the NIT of cycle 13, it is verified (UT) that the FIFO buffer holds the contents of the frames as simulated by the LT in slot 5 and slot n_{dyn} of cycle 13.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs frame filtering according to the FIFO buffer configuration.

7.3.8.3.5 Non-queued receive buffer for both channels and FIFO buffer

— Test purpose

Verify frame reception into an assigned non-queued receive buffer and FIFO buffer. The first semantically valid frame received within the assigned slot is stored in the non-queued receive buffer.

— Applicability

DC.

— Configuration

All basic configurations.

FIFO buffer(s) is configured in the IUT to receive all valid non-null frames in static slot 5 on both channels. Write access to the admittance criteria is enabled during *POC:normal active* state. A non-queued receive buffer in the IUT is assigned to slot 5 for both channels.

The LT simulates an additional frame in static slot 5 with different payload data on channel A (the payload byte 0 is 0x55, byte 1 is 0x55) and on channel B (the payload byte 0 is 0xAA, byte 1 is 0xAA).

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates the frame in slot 5 on channel A starting at *gdActionPointOffset* – 1 MT and the frame in slot 5 on channel B starting at *gdActionPointOffset*.

2) In the NIT of cycle 8, it is verified (UT) that the non-queued receive buffer holds the contents (frame header and payload data) of the frame in the frame contents data as simulated by the LT on channel A in slot 5 of cycle 8. Additionally it is verified (UT) that the non-queued receive buffer holds the slot status for each channel separately, i.e. *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A and *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true,

vSS!NFIndicator = '1' for channel B. It is verified that the FIFO buffer holds no payload data (i.e. the payload data as simulated by the LT on channel B in slot 5 of cycle 8 is not stored in either buffers).

3) In cycle 9, the LT simulates the frame in slot 5 on channel B starting at *gdActionPointOffset* – 1 MT and the frame in slot 5 on channel A starting at *gdActionPointOffset*.

4) In the NIT of cycle 9, it is verified (UT) that the non-queued receive buffer holds the contents (frame header and payload data) of the frame in frame contents data as simulated by the LT on channel B in slot 5 of cycle 9. Additionally it is verified (UT) that the non-queued receive buffer holds the slot status for each channel separately, i.e. *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel A and *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1' for channel B. It is verified that the FIFO buffer holds no payload data (i.e. the payload data as simulated by the LT on channel A in slot 5 of cycle 9 is not stored in either buffers).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

A non-queued receive buffer assigned to a slot on both channels shall store the first semantically valid frame received within the assigned slot and is capable of storing the slot status data of both channels separately. A FIFO buffer assigned to the same slot shall not store the frame contents.

7.3.8.3.6 FIFO frame identifier admittance criteria

— Test purpose

Verify that the IUT is capable to admit only frames with frame identifiers belonging to a configurable set of frame identifiers into a FIFO buffer.

— Applicability

SC, DC.

— Configuration

All basic configurations.

A FIFO buffer is configured in the IUT to receive valid non-null frames in the static slots 3 and 4, as well as the frames in the last static and in the first dynamic slot in every cycle ($Range1_{min}$ = 3, $Range1_{max}$ = 4, $Range2_{min}$ = *gNumberOfStaticSlots*, $Range2_{max}$ = *gNumberOfStaticSlots* + 1). No non-queued receive buffer is configured in the IUT.

NOTE For dual channel test execution and in case the IUT does not support the configuration of one single FIFO for both channels, two FIFO buffers shall be configured and the verification steps in the test execution shall apply to the respective FIFO.

— Preamble (setup state)

Preamble II.

— Test execution

1) In cycle 7, the LT transmits valid frames in the static slots 3, 4, 5, in the last two static slots and in the first two dynamic slots. Each frame is transmitted with a payload of two bytes containing the slot number.

2) In the NIT of cycle 7, it is verified (UT) that the IUT has stored the correct frames into the FIFO buffer, i.e. the frame ID, the channel identifier and the payload of FIFO entries shall match the transmitted frames in the slots 3, 4, in the last static slot and in the first dynamic slot respectively, and the FIFO shall not contain any other entry.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT admits received frames into their FIFO, if the received frame identifier belongs to a configurable set of frame identifiers. All other received frames are not admitted into the FIFO.

7.3.8.3.7 Configuration of FIFO admittance criteria

— **Test purpose**

Verify the correct handling of host's write access to admittance criteria of a FIFO in different POC states.

The write access to the admittance criteria during *POC:normal active* state and *POC:normal passive* state shall be configurable in the *POC:config* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 204.

Table 204 — Modification to basic configuration for queued receive buffer – configuration of FIFO admittance criteria

Parameter	Modifications
pKeySlotID	1
pKeySlotUsedForSync	true
pKeySlotOnlyEnabled	true
gMaxWithoutClockCorrectionPassive	1

Table 205 defines two different FIFO admittance criteria sets, which are used to verify the UT's write access to IUT's FIFO configuration data in different POC states.

Table 205 — FIFO admittance criteria for configuration of FIFO admittance criteria

FIFO admittance criteria	Set 1	Set 2
FIFO frame validity admittance criteria	AdmitNullFrame = true	AdmitNullFrame = false
Pass the FIFO channel admittance criteria only if the frame was received on	channel A	channel B
FIFO frame identifier admittance criteria	$Range1_{min}$ = 1, $Range1_{max}$ = 2, $Range2_{min}$ = 3, $Range2_{max}$ = 4	$Range1_{min}$ = 5, $Range1_{max}$ = 6, $Range2_{min}$ = 7, $Range2_{max}$ = 8
FIFO cycle counter admittance criteria	Cycle_Repetition = 2, Cycle_Offset = 1	Cycle_Repetition = 4, Cycle_Offset = 3
Message identifier admittance criteria	AdmitWithoutMessageID = false	AdmitWithoutMessageID = true

— **Preamble (setup state)**

The IUT is configured as coldstart node sending startup frames in slot 1. The IUT takes the role of the following coldstart node, the LT the role of the leading coldstart node simulating startup frames in slot 3.

The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

— **Test execution**

1) The UT writes the configuration data of the FIFO buffer (in the following referenced as FIFO1) with the FIFO admittance criteria set 1.

2) The UT verifies that the write access in *vPOC!State* = *DEFAULT_CONFIG* is not successful, by reading the admittance criteria of FIFO1 and comparing them with set 1.

3) The UT sets the IUT into *POC:config* state with the CHI command CONFIG.

4) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

5) The UT configures FIFO1 with the admittance criteria set 2 and

   (a) disables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

   (b) enables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

6) The UT verifies that the write access to the FIFO admittance criteria in *vPOC!State* = *CONFIG* is successful, by reading the admittance criteria of FIFO1 and comparing them with set 2.

7) The UT configures the IUT with the given basic configuration.

8) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE.

9) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*). The UT

   (a) enables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

(b) disables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

10) The UT initiates the startup procedure with the CHI command RUN.

11) The LT represents the leading coldstart node in the cluster and simulates a CAS 0,5 * *pdListenTimeout* ± 0,1 * *pdListenTimeout* after the CHI command RUN.

12) In cycle 0, the UT

(a) enables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

(b) disables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

13) In slot 3 of cycles 0 to 6, the LT simulates a startup frame with the null frame indicator, the reserved bit and the payload preamble indicator set to '0', holding a payload of 0x00 in all payload bytes.

14) In cycle 7, within 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

15) In cycle 7, the UT

(a) enables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

(b) disables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

16) In the NIT of cycle 7, the UT configures FIFO1 with the admittance criteria set 1.

17) The UT verifies that the

(a) write access to the FIFO admittance criteria in *vPOC!State* = *NORMAL_ACTIVE* is not successful, by reading the admittance criteria of FIFO1 and comparing them with set 2. The admittance criteria matches set 2.

(b) write access to the FIFO admittance criteria in *vPOC!State* = *NORMAL_ACTIVE* is successful, by reading the admittance criteria of FIFO1 and comparing them with set 1. The admittance criteria matches set 1. The UT configures FIFO1 again with the FIFO admittance criteria configuration set 2.

18) In cycle 8, the LT stops startup frame simulation.

19) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

20) In cycle 10, the UT

(a) enables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

(b) disables the write access to the FIFO admittance criteria during *POC:normal active* state or *POC:normal passive* state.

21) In the NIT of cycle 10, the UT configures FIFO1 with the admittance criteria set 1.

22) The UT verifies that the

(a) write access to the FIFO admittance criteria in *vPOC!State* = *NORMAL_PASSIVE* is not successful, by reading the admittance criteria of FIFO1 and comparing them with set 2. The admittance criteria matches set 2.

(b) write access to the FIFO admittance criteria in *vPOC!State* = *NORMAL_PASSIVE* is successful, by reading the admittance criteria of FIFO1 and comparing them with set 1. The admittance criteria matches set 1.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs the expected state transitions and the write access to FIFO's admittance criteria buffer is as expected.

7.3.8.3.8 FIFO access after transition from normal passive

— Test purpose

Verify that the IUT is capable to provide the host with the complete slot status data and frame contents data of the oldest entry, that the host can remove the oldest entry, and that upon a transition from *POC:normal passive* state to either the *POC:halt* or *POC:ready* state the IUT continues to provide host access to FIFO.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 206.

Table 206 — Modification to basic configurations for queued receive buffer – FIFO access after transition from normal passive

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1

A FIFO buffer is configured only for one available channel in the IUT to receive valid frames in the static slots 1 and 3 in each cycle. This FIFO is the only receive buffer configured in the IUT.

— Preamble (setup state)

Preamble I.

The preamble is supplemented by the following statement: "In the NIT of cycle 7, the UT removes one entry from IUT's FIFO."

— Test execution

1) In cycle 8, the LT simulates a valid frame in slot 1 with startup and sync bit set to '0', and an additional valid frame in slot 3.

2) In the NIT of cycle 8, it is verified (UT) that the IUT has stored two frames into the FIFO buffer. It is verified (UT), by reading the first FIFO entry and removing it from FIFO, that the IUT has stored the frame transmitted in slot 1 with the slot status data containing *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1', and with the frame contents data containing *vRF!Header!Reserved* = '0', *vRF!Header!FrameID* = 1, *vRF!Header!CycleCount* = 8, *vRF!Header!PPIndicator* = '0', *vSS!NFIndicator* = '1', *vRF!Header!SyFIndicator* = '0', *vRF!Header!SuFIndicator* = '0', and the expected *vRF!Header!HeaderCRC*, *vRF!Header!Length*, *vRF!Payload* and *vRF!Channel*.
It is verified (UT) that the IUT has remaining one frame in the FIFO.
It is also verified (UT), by reading the first FIFO entry and removing it from FIFO, that this entry contains the frame transmitted in slot 3 with the slot status data containing *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1', and with the frame contents data containing *vRF!Header!Reserved* = '0', *vRF!Header!FrameID* = 3, *vRF!Header!CycleCount* = 8, *vRF!Header!PPIndicator* = '0', *vSS!NFIndicator* = '1', *vRF!Header!SyFIndicator* = '0', *vRF!Header!SuFIndicator* = '0', and the expected *vRF!Header!HeaderCRC*, *vRF!Header!Length*, *vRF!Payload* and *vRF!Channel*.
It is verified (UT) that the FIFO is empty.

3) In cycle 9, the LT transmits only one frame in slot 3.

4) In the NIT of cycle 9, it is verified (UT) that the IUT has stored one frame into the FIFO buffer.

5) In cycle 10, the LT transmits both frames correctly (frame in slot 1 with startup and sync bit set to '1').

6) 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

7) In the NIT of cycle 10, it is verified (UT) that the FIFO buffer contains three entries. By reading and removing the first entry, it is verified (UT) that this entry contains the frame transmitted in cycle 9 / slot 3 with the slot status data containing *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1', and with the frame contents data containing *vRF!Header!Reserved* = '0', *vRF!Header!FrameID* = 3, *vRF!Header!CycleCount* = 9, *vRF!Header!PPIndicator* = '0', *vSS!NFIndicator* = '1', *vRF!Header!SyFIndicator* = '0', *vRF!Header!SuFIndicator* = '0', and the expected *vRF!Header!HeaderCRC*, *vRF!Header!Length*, *vRF!Payload* and *vRF!Channel*.
It is verified (UT) that the IUT has still stored two remaining frames in the FIFO.

8) In the NIT of cycle 10, the UT issues the CHI command

   (a) IMMEDIATE_READY.

   (b) FREEZE

9) 2 500 µT after UT has issued the last CHI command, it is verified (UT) that the IUT has entered

   (a) the *POC:ready* state (*vPOC!State* = *READY*).

   (b) the *POC:halt* state (*vPOC!State* = *NORMAL_PASSIVE* and *vPOC!Freeze* = true). It is verified (UT) that the IUT has still two remaining frames in the FIFO.

10) It is verified (UT), by reading the first FIFO entry and removing it from the FIFO, that this entry contains the frame transmitted in cycle 10 / slot 1 with the slot status data containing *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1', and with the frame contents data containing *vRF!Header!Reserved* = '0', *vRF!Header!FrameID* = 1, *vRF!Header!CycleCount* = 10, *vRF!Header!PPIndicator* = '0', *vSS!NFIndicator* = '1', *vRF!Header!SyFIndicator* = '1', *vRF!Header!SuFIndicator* = '1', and the expected *vRF!Header!HeaderCRC*, *vRF!Header!Length*, *vRF!Payload* and *vRF!Channel*.
It is verified (UT) that the IUT has one remaining frame in the FIFO.
It is verified (UT), by reading the first FIFO entry and removing it from the FIFO, that this entry

contains the frame transmitted in cycle 10 / slot 3 with the slot status data containing *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1', and with the frame contents data containing *vRF!Header!Reserved* = '0', *vRF!Header!FrameID* = 3, *vRF!Header!CycleCount* = 10, *vRF!Header!PPIndicator* = '0', *vSS!NFIndicator* = '1', *vRF!Header!SyFIndicator* = '0', *vRF!Header!SuFIndicator* = '0', and the expected *vRF!Header!HeaderCRC*, *vRF!Header!Length*, *vRF!Payload* and *vRF!Channel*.
It is verified (UT) that the FIFO is empty.

— **Postamble**

(a) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

(b) No action required as IUT is already in *POC:halt.*

— **Pass criteria**

The number of FIFO entries and the content of entries are as expected.

7.3.8.3.9 FIFO access after transition from normal active

— **Test purpose**

Verify that the IUT is capable to provide the host with the complete slot status data and frame contents data of the oldest entry, that the host can remove the oldest entry, and that upon a transition from *POC:normal active* state to either the *POC:halt* or *POC:ready* state the IUT continues to provide host access to FIFO.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble II.

A FIFO buffer is configured in the IUT to receive valid frames in the static slots 2 and 3 in each cycle and on one available channel. This FIFO is the only receive buffer configured in the IUT.

The LT transmits an additional frame in the static slot 3.

— **Test execution**

1) In the NIT of cycle 6, it is verified (UT) that the IUT has stored two frames into the FIFO buffer. It is verified (UT) that the first entry contains the frame transmitted in slot 2 with the slot status data containing *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1', and with the frame contents data containing *vRF!Header!Reserved* = '0', *vRF!Header!FrameID* = 2, *vRF!Header!CycleCount* = 6, *vRF!Header!PPIndicator* = '0', *vSS!NFIndicator* = '1', *vRF!Header!SyFIndicator* = '1', *vRF!Header!SuFIndicator* = '1', and with the expected *vRF!Header!HeaderCRC*, *vRF!Header!Length*, *vRF!Payload* and *vRF!Channel*.

2) The UT removes one entry from IUT's FIFO and issues the CHI command

   (a) IMMEDIATE_READY.

   (b) FREEZE

3) After 2 500 µT it is verified (UT) that the IUT has entered

   (a) the *POC:ready* state (*vPOC!State* = *READY*).

   (b) the *POC:halt* state (*vPOC!State* = *NORMAL_ACTIVE* and *vPOC!Freeze* = true).

4) It is verified (UT) that the FIFO buffer has one entry containing the frame transmitted in slot 3 with the slot status data containing *vSS!SyntaxError* = false, *vSS!ContentError* = false, *vSS!BViolation* = false, *vSS!ValidFrame* = true, *vSS!NFIndicator* = '1', and with the frame contents data containing *vRF!Header!Reserved* = '0', *vRF!Header!FrameID* = 3, *vRF!Header!CycleCount* = 6, *vRF!Header!PPIndicator* = '0', *vSS!NFIndicator* = '1', *vRF!Header!SyFIndicator* = '0', *vRF!Header!SuFIndicator* = '0', and with the expected *vRF!Header!HeaderCRC*, *vRF!Header!Length*, *vRF!Payload* and *vRF!Channel*.

— Postamble

(a) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

(b) No action required as IUT is already in *POC:halt*.

— Pass criteria

The number of FIFO entries and their content are as expected and the UT has read access to FIFO in the states *POC:ready* and *POC:halt*.

7.3.8.3.10 Queued receive buffer performance

— Test purpose

Verify FIFO buffer performance requirements. The FIFO shall have a depth of at least eight entries. The size of each entry shall be greater than or equal of a non-queued receive buffer. An overrun indicator is provided in the CHI and set if an overrun condition is occurred.

— Applicability

SC.

— Configuration

All basic configurations using the modifications as listed in Table 207.

Table 207 — Modification to basic configurations for queued receive buffer – queued receive buffer performance

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
gdStaticSlot [MT]	274	274	271	271	535
gMacroPerCycle [MT]	4 826	4 826	3 729	3 729	6 459
gNumberOfStaticSlots	10	10	10	10	10
gPayloadLengthStatic [two-byte word]	127	127	127	127	127
gdCycle [µs]	4 826	4 826	7 458	7 458	12 918
gdNIT [MT]	75	75	16	16	70
pdListenTimeout [µT]	386 312	772 624	597 000	298 500	517 032
pMicroPerCycle [µT]	193 040	386 080	298 320	149 160	258 360
pOffsetCorrectionStart [MT]	4 752	4 752	3 714	3 714	6 390
pRateCorrectionOut [µT]	116	232	180	90	156

$BufferMessageLength_{FIFO}$ = the implementation specific maximum length of available message data in two-byte word.

A FIFO buffer is configured in the IUT to receive all valid non-null frames in all static slots between 3 and 10 ($Range1_{min}$ = 3, $Range1_{max}$ = 10).

The FIFO buffer width is $BufferMessageLength_{FIFO}$.

The size of the FIFO buffer is configured for at least eight entries.

Write access to the admittance criteria is disabled during *POC:normal active* state. No non-queued receive buffer in the IUT is assigned to the slots between 3 and 10.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In cycle 8, the LT simulates additional frames in all slots between 3 and 10. For all simulated frames (1 and 3 to 10) the first byte of payload is equal to cycle number the second byte equal to SlotId, all other bytes are set to 0x00.

2) In the NIT of cycle 8, it is verified (UT) that the IUT has stored eight entries in the FIFO and that the overrun indicator in the FIFO status information is not set. It is further verified (UT) that the IUT has stored the correct frames into the FIFO buffer by reading and removing the entries.
The frame ID, the channel identifier and the first (BufferMessageLengthFIFO * two-byte words) of the payload of the FIFO entries shall match the transmitted frames in the slots 3 to 10.

3) In cycle 9, the LT simulates additional frames in all slots between 3 and 10.

4) Beginning with cycle 10, the LT simulates an additional valid non-null frame in slot 3 only.

5) In the NIT of the cycle where the LT has transmitted the amount of frames necessary for causing the FIFO buffer to overrun, it is verified (UT) the FIFO overrun indicator is set in the FIFO status information.

6) In the NIT of the next cycle, it is verified (UT) the FIFO overrun indicator is still set in the FIFO status information. The UT resets (clears) the FIFO overrun indicator and verifies that the FIFO overrun indicator is cleared in the FIFO status information.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The FIFO buffer contains the received frames as expected. The IUT shall indicate a FIFO overrun correctly in the FIFO status information.

7.3.8.4 Buffer configuration data class 1 and class 2

— Test purpose

To verify the correct behaviour of IUT regarding write access to the configuration data class 1 and class 2 the UT enables / disables the write access to buffer configuration data class 1 and / or 2 and tries to change the buffer configuration data when IUT is in different POC states.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 208.

Table 208 — Modification to basic configurations for queued receive buffer – buffer configuration data class 1 and class 2

Parameter	Modification
pKeySlotOnlyEnabled	true
gMaxWithoutClockCorrectionPassive	1

Table 209 defines the two different buffer configuration data sets.

Table 209 — Message buffer configuration for buffer configuration data class 1 and class 2

Buffer Configuration Data Class 1	Set 1	Set 2
Type indication	TX	RX
Slot identifier	11	12
Channel identifier	A	B
Set of communication cycles	odd	even
Buffer Configuration Data Class 2		
Message length	1	2
Payload preamble indicator	1	0
Header CRC	0x7FF	0x000
Transmission mode indicator	single shot mode	continuous mode

— **Preamble (setup state)**

The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

— **Test execution**

1) The UT reads the configuration data of all[3)] message buffers and verifies that the message buffers can neither transmit nor receive.

2) The UT configures one message buffer (in the following referenced as MB) with the buffer configuration data set 1.

3) The UT verifies that the write access in *vPOC!State* = *DEFAULT_CONFIG* is not successful, by reading the configuration data of MB and comparing it with set 1.

4) The UT sets the IUT into *POC:config* state with the CHI command CONFIG.

5) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

6) The UT configures the IUT with the given basic configuration.

7) The UT configures MB with the buffer configuration data set 1.

8) The UT verifies that the write access in *vPOC!State* = *CONFIG* is successful, by reading the configuration data of MB and Comparing it with set 1.

9) The UT configures

   (a) No write access to either Class 1 or Class 2 message buffer configuration data while in states other than *POC:config*.

3) In case the implementation allows a dynamic allocation of message buffers, it is verified that no message buffer is enabled.

(b) Write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config*.

(c) Write access to Class 2 message buffer configuration data while in states other than *POC:default config* and no write access to Class 1 message buffer configuration data while in states other than *POC:config*.

10) The UT configures MB with the buffer configuration data set 2.

11) The UT verifies that the write access in *vPOC!State* = *CONFIG* is successful, by reading the configuration data of MB and Comparing it with set 2.

12) The UT configures the IUT to transmit a startup frame in slot 1.

13) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE.

14) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

15) The UT configures MB with the buffer configuration data set 1.

16) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 2 (the UT has no write access to either Class 1 or Class 2 message buffer configuration data).

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 1 (the UT has write access to both Class 1 and Class 2 message buffer configuration data).

(c) the Class 1 message buffer configuration data of MB matches set 2 and the Class 2 message buffer configuration data of MB matches set 1 (the UT has write access to Class 2 and no write access to Class 1 message buffer configuration data).

The UT configures MB with the buffer configuration data set 2.

17) The UT issues the CHI command WAKEUP.

18) After 2 500 µT it is verified (UT) that the IUT is in a wakeup state (*vPOC!State* = *WAKEUP*).

19) The UT configures MB with the buffer configuration data set 1.

20) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 2.

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 1.

(c) the Class 1 message buffer configuration data of MB matches set 2 and the Class 2 message buffer configuration data of MB matches set 1.

The UT configures MB with the buffer configuration data set 2.

21) The UT issues the CHI command IMMEDIATE_READY.

22) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*) and the UT issues the CHI command RUN.

23) After 2 500 µT it is verified (UT) that the IUT has entered a startup state (*vPOC!State* = *STARTUP*).

24) The UT configures MB with the buffer configuration data set 1.

25) By reading the configuration data of MB, it is verified (UT) that

    (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 2.

    (b) the Class 1 and Class 2 message buffer configuration data of MB matches set 1.

    (c) the Class 1 message buffer configuration data of MB matches set 2 and the Class 2 message buffer configuration data of MB matches set 1.

    The UT configures MB with the buffer configuration data set 2.

26) The LT represents the leading coldstart node and simulates a CAS latest (0,9 ± 0,1) * *pdListenTimeout* after the CHI command RUN.

27) In slot 2 of cycles 0 to 7, the LT simulates startup frames.

28) In cycle 7, within 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

29) In the NIT of cycle 7, the UT configures MB with the buffer configuration data set 1.

30) By reading the configuration data of MB, it is verified (UT) that

    (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 2.

    (b) the Class 1 and Class 2 message buffer configuration data of MB matches set 1.

    (c) the Class 1 message buffer configuration data of MB matches set 2 and the Class 2 message buffer configuration data of MB matches set 1.

    The UT configures MB with the buffer configuration data set 2.

31) With the begin of cycle 8, the LT stops startup frame simulation.

32) In cycle 10, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

33) The UT configures MB with the buffer configuration data set 1.

34) By reading the configuration data of MB, it is verified (UT) that

    (a) the Class 1 and Class 2 message buffer configuration data of MB matches set 2.

    (b) the Class 1 and Class 2 message buffer configuration data of MB matches set 1.

    (c) the Class 1 message buffer configuration data of MB matches set 2 and the Class 2 message buffer configuration data of MB matches set 1.

    The UT configures MB with the buffer configuration data set 2.

35) The UT issues the CHI command FREEZE.

36) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:halt* state, i.e. *vPOC!State* = *HALT* and *vPOC!Freeze* = true.

37) The UT configures MB with the buffer configuration data set 1.

38) By reading the configuration data of MB, it is verified (UT) that

(a) the Class 1 and Class 2 message buffer configuration data of MB matches set 2.

(b) the Class 1 and Class 2 message buffer configuration data of MB matches set 1.

(c) the Class 1 message buffer configuration data of MB matches set 2 and the Class 2 message buffer configuration data of MB matches set 1.

— Postamble

none.

— Pass criteria

The IUT performs the expected state transitions and the configuration data of the message buffer is as expected.

7.3.8.5 Slot status updated indicator

— Test purpose

Verify correct setting of the slot status updated indicator.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

Two buffers are configured for transmission in static slot 3 and in dynamic slot n_{dyn} with all payload bytes set to 0x55. Two additional receive buffers in the IUT are assigned to static slot 5 and to dynamic slot n_{dyn}+1.

The IUT is configured for write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config* and *POC:config*.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates its frames in slot 5 and slot n_{dyn}+1.

2) In cycle 8, 2 000 µT after slot n_{dyn}+1 and before the symbol window, it is verified (UT) that the slot status updated indicator is true in the transmit buffer status of slot 3 and slot n_{dyn}. It is also verified (UT) that the slot status updated indicator is true in the slot status data of slot 5 and slot n_{dyn}+1.

3) In the symbol window of cycle 8, the UT configures the *MessageLength* of the message buffers for slot 3 and slot n_{dyn} to 1 two-byte word.

4) In the NIT of cycle 8, it is verified (UT) that the slot status updated indicator is false in the message buffer status of slot 3 and slot n_{dyn}. It is also verified (UT) that the slot status updated indicator is true in the message buffer status of slot 5 and in the slot status data of slot n_{dyn}+1.

5) The UT sets the slot status updated indicator of all transmit and receive buffers to false and verifies that they are cleared and the UT sets the payload data valid flag for all configured transmit buffers to true. The header CRC for slot n_{dyn} is recalculated because of the modification of the MessageLength.

6) In cycle 9, the LT simulates its frames in slot 5 and slot n_{dyn}+1.

7) In the NIT of cycle 9, it is verified (UT) that the slot status updated indicator is true in the transmit buffer status of slot 3 and slot n_{dyn}. It is also verified (UT) that the slot status updated indicator is true in the slot status data of slot 5 and slot n_{dyn}+1.

8) In the static segment of cycle 10, the UT issues the command IMMEDIATE_READY.

9) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*). It is also verified (UT) that the slot status updated indicator is true in the transmit buffer status of slot 3 and slot n_{dyn} and that the slot status updated indicator is true in the slot status data of slot 5 and slot n_{dyn}+1.

10) The UT issues the command RUN.

11) After 2 500 µT it is verified (UT) that the IUT has entered a startup state (*vPOC!State* = *STARTUP*). It is also verified (UT) that the slot status updated indicator is false in the transmit buffer status of slot 3 and slot n_{dyn} and that the slot status updated indicator is false in the slot status data of slot 5 and slot n_{dyn}+1.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The slot status updated indicator in the transmit buffer status and slot status data is set accordingly.

7.3.8.6 Payload data valid flag

— **Test purpose**

Verify correct setting of the payload data valid flag in different states.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 210.

Table 210 — Modification to basic configurations for queued receive buffer – payload data valid flag

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1

n_{dyn} = *gNumberOfStaticSlots* + 1.

Two buffers are configured for transmission in static slot 3 and in dynamic slot n_{dyn} with all payload bytes set to 0x55 in continuous transmission mode. Two additional receive buffers in the IUT are assigned to static slot 5 and to dynamic slot n_{dyn}+1.

The IUT is configured for write access to both Class 1 and Class 2 message buffer configuration data while in states other than *POC:default config* and *POC:config*.

The payload data valid flag of the transmit buffers are set to false during the buffer configuration.

— **Preamble (setup state)**

Preamble I.

— **Test execution**

1) In the NIT of cycle 7, it is verified (UT) that the payload data valid flag is false for both receive buffers (slot 5 and slot n_{dyn}+1).

2) The UT sets the payload data valid flag for all transmit buffers to true.

3) In cycle 8, the LT simulates its frames in slot 5 and slot n_{dyn}+1 and continues to transmit its startup frame in slot 1.

4) In cycle 8, 2 000 µT after slot n_{dyn}+1 and before the symbol window, it is verified (UT) that the payload data valid flag is true for both transmit buffers (slot 3 and slot n_{dyn}) and for both receive buffers (slot 5 and slot n_{dyn}+1).

5) In the symbol window of cycle 8, the UT configures the *MessageLength* of the message buffers for slot 3 and slot n_{dyn} to 1 two-byte word. The header CRC for slot n_{dyn} is recalculated because of the modification of the MessageLength.

6) In the NIT of cycle 8, it is verified (UT) that the payload data valid flag is false for the transmit buffer of slot 3 and for the receive buffer of slot n_{dyn}. It is also verified (UT) that the payload data valid flag is true for the receive buffer of slot 5 and for the transmit buffer of slot n_{dyn}+1.

7) The UT sets the payload data valid flag for all transmit buffers to true.

8) In cycle 9, the LT simulates its frames in slot 5 and slot n_{dyn}+1 and continues to transmit its startup frame in slot 1.

9) In the symbol window of cycle 9, it is verified (UT) that the payload data valid flag is true for all configured buffers.

10) In the NIT of cycle 9, the UT issues the CHI command IMMEDIATE_READY.

11) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

12) 3 000 µT after the CHI command IMMEDIATE_READY, it is verified (UT) that the payload data valid flag is false for both transmit buffers (slot 3 and slot n_{dyn}). It is also verified (UT) that the payload data valid flag is true for both receive buffers (slot 5 and slot n_{dyn}+1).

13) The UT sets the payload data valid flag for all transmit buffers to true and issues the CHI command WAKEUP.

14) After 2 500 µT it is verified (UT) that the IUT is in a wakeup state (*vPOC!State* = *WAKEUP*).

15) 3 000 µT after the CHI command WAKEUP, it is verified (UT) that the payload data valid flag is true for both transmit buffers (slot 3 and slot n_{dyn}). It is also verified (UT) that the payload data valid flag is true for both receive buffers (slot 5 and slot n_{dyn}+1).

16) The UT issues the CHI command IMMEDIATE_READY.

17) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

18) 3 000 µT after the CHI command IMMEDIATE_READY, it is verified (UT) that the payload data valid flag is true for all configured buffers.

19) The UT issues the CHI command RUN.

20) After 2 500 µT it is verified (UT) that the IUT has entered a startup state (*vPOC!State* = *STARTUP*).

21) 3 000 µT after the CHI command RUN, it is verified (UT) that the payload data valid flag is true for both transmit buffers (slot 3 and slot n_{dyn}). It is also verified (UT) that the payload data valid flag is false for both receive buffers (slot 5 and slot n_{dyn}+1).

22) The UT issues the CHI command IMMEDIATE_READY.

23) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

24) 3 000 µT after the CHI command IMMEDIATE_READY, it is verified (UT) that the payload data valid flag is false for all configured buffers.

25) The UT sets the payload data valid flag for all transmit buffers to true and issues the command RUN.

26) After 2 500 µT it is verified (UT) that the IUT has entered a startup state (*vPOC!State* = *STARTUP*).

27) The LT represents the leading coldstart node and simulates a CAS latest (0,9 ± 0,1) * *pdListenTimeout* after the CHI command RUN.

28) In slot 1 of cycle 0 to 7, the LT simulates a startup frame with the null frame indicator, the reserved bit and the payload preamble indicator set to '0', holding a payload of 0x00 in all payload bytes.

29) Beginning with cycle 7, the LT simulates its frames only in slot 5 and slot n_{dyn}+1.

30) In cycle 7 within 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

31) In the NIT of cycle 7, it is verified (UT) that the payload data valid flag is true for all configured buffers.

32) Beginning with cycle 8, the LT stops startup frame simulation.

33) In cycle 10, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

34) The UT issues the CHI command IMMEDIATE_READY.

35) After 2 500 µT it is verified (UT) that the IUT has entered the *POC:ready* state (*vPOC!State* = *READY*).

36) 3 000 µT after the CHI command IMMEDIATE_READY, it is verified (UT) that the payload data valid flag is false for both transmit buffers (slot 3 and slot n_{dyn}). It is also verified (UT) that the payload data valid flag is true for both receive buffers (slot 5 and slot n_{dyn}+1).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The payload data valid flag is set accordingly.

7.3.9 Timer and interrupt service

7.3.9.1 Cycle count and macrotick timer (repetitive mode)

— Test purpose

Verify the functionality of the timer, the corresponding timer interrupt requests, and the corresponding interrupt status indication flags. It is verified that the IUT provides 2 timers that expire on a configured communication cycle count and macrotick in repetitive mode. It is also verified that an interrupt request is issued for the cycle start interrupt source with each cycle start.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{int_1} = 3 000 µs / *gdCycle* * *gMacroPerCycle*.

n_{int_2} = 4 000 µs / *gdCycle* * *gMacroPerCycle*.

m_{tol} = ceil(100 µT / (*gdMacrotick* / *pdMicrotick*)).

— Preamble (setup state)

Preamble II.

— Test execution

The IUT has to provide at least two timers that are further denoted in the test execution timer T1 and timer T2.

1) At the beginning of cycle 7 the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) In cycle 7 after clearing the interrupt indication flags the UT configures the timer T1 to cycle count 8, macrotick n_{int_1} and repetitive mode, activates it and enables interrupt requests for the corresponding interrupt source.

3) In cycle 7 after clearing the interrupt indication flags the UT configures the timer T2 to cycle count 9, macrotick n_{int_2} and repetitive mode, activates it and enables interrupt requests for the corresponding interrupt source.

4) In cycle 7 after clearing the interrupt indication flags the UT enables interrupt requests for the cycle start interrupt source.

5) In cycle 7, 2 500 µT after the activation of timer T1, timer T2, and enabling interrupt requests for the cycle start interrupt source it is verified (UT) that the interrupt status indication flags are not set for the timer T1 and T2 or for the cycle start.

6) In each cycle (n) mod (*gCycleCountMax* + 1) = 8 (n=[8;2 * (*gCycleCountMax* + 1)]), it is verified (UT) that the IUT generates an interrupt request for the interrupt source T1 within [0; 100] µT + ξ_{IUT} µT after

T1 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_1}; n_{int_1} + m_{tol}] and that the cycle counter *vCycleCounter* is 8 right after the interrupt assertion. The UT clears the corresponding interrupt status indication flag after its verification.

7) In each cycle (n) mod (*gCycleCountMax* + 1) = 9 (n=[8;2 * (*gCycleCountMax* + 1)]), it is verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after T2 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_2};n_{int_2} + m_{tol}] and that the cycle counter *vCycleCounter* is 9 right after the interrupt assertion. The UT clears the corresponding interrupt status indication flag after its verification.

8) In each cycle (n) mod (*gCycleCountMax* + 1) ≠ 8 (n=[8;2 * (*gCycleCountMax* + 1)]), it is verified (UT) that the IUT does not generate an interrupt request for the interrupt source T1.

9) In each cycle (n) mod (*gCycleCountMax* + 1) ≠ 9 (n=[8;2 * (*gCycleCountMax* + 1)]), it is verified (UT) that the IUT does not generate an interrupt request for the interrupt source T2.

10) In each cycle (n) mod (*gCycleCountMax* + 1) (n=[8;2 * (*gCycleCountMax* + 1)]), it is verified (UT) that the IUT generates an interrupt request for the cycle start interrupt source within [0; 100] µT + ξ_{IUT} µT after the cycle start, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [0; m_{tol}]. The UT clears the corresponding interrupt status indication flag after its verification.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT indicates an interrupt request at the configured cycle count and macrotick and at the cycle start, repetitive in every cycle. The interrupt status indication flags are set accordingly.

7.3.9.2 Cycle count and macrotick timer (non-repetitive mode)

— **Test purpose**

Verify the functionality of the timer, the corresponding timer interrupt requests, and the corresponding interrupt status indication flags. It is verified that the IUT provides 2 timers that expire on a configured communication cycle count and macrotick in non-repetitive mode.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

n_{int_1} = 3 000 µs / *gdCycle* * *gMacroPerCycle*.

n_{int_2} = 4 000 µs / *gdCycle* * *gMacroPerCycle*.

m_{tol} = ceil(100 µT / (*gdMacrotick* / *pdMicrotick*)).

— **Preamble (setup state)**

Preamble II.

— Test execution

The IUT has to provide at least two timers that are further denoted in the test execution timer T1 and timer T2.

1) At the beginning of cycle 7 the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) In cycle 7 after clearing the interrupt indication flags the UT configures the timer T1 to cycle count 8, macrotick n_{int_1} and non-repetitive mode, activates it and enables interrupt requests for the corresponding interrupt source.

3) In cycle 7 after clearing the interrupt indication flags the UT configures the timer T2 to cycle count 9, macrotick n_{int_2} and non-repetitive mode, activates it and enables interrupt requests for the corresponding interrupt source.

4) In cycle 7, 2 500 µT after the activation of timer T1 and T2 it is verified (UT) that the interrupt status indication flags are not set for the timer T1 or T2.

5) In cycle 8, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T1 within [0; 100] µT + ξ_{IUT} µT after T1 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_1}; n_{int_1} + m_{tol}] and that the cycle counter *vCycleCounter* is 8 right after the interrupt assertion. The UT clears the corresponding interrupt status indication flag after its verification.

6) In cycle 9, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after T2 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_2}; n_{int_2} + m_{tol}] and that the cycle counter *vCycleCounter* is 9 right after the interrupt assertion. The UT clears the corresponding interrupt status indication flag after its verification.

7) In cycles 10 to cycle 9 after the cycle count wraparound, it is verified (UT) that the IUT does not generate an interrupt request for the interrupt sources T1 and T2 and the interrupt status indication flags for T1 and T2 are not set.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT indicates an interrupt at the configured cycle count and macrotick once. The interrupt status indication flags are set accordingly.

7.3.9.3 Macrotick only timer (repetitive mode)

— Test purpose

Verify the functionality of the timer, the corresponding timer interrupt requests, and the corresponding interrupt status indication flags. It is verified that the IUT provides 2 timers that expire on a configured macrotick in repetitive mode. It is also verified that no interrupt request is issued to the UT at expiration of the timer if interrupt requests for the interrupt source timer are disabled and that an interrupt request is issued to the UT at expiration of the timer if interrupt requests for the interrupt source timer are enabled.

— Applicability

SC, DC.

— **Configuration**

All basic configurations.

n_{int_1} = 3 000 µs / *gdCycle* * *gMacroPerCycle*.

n_{int_2} = 4 000 µs / *gdCycle* * *gMacroPerCycle*.

m_{tol} = ceil(100 µT / (*gdMacrotick* / *pdMicrotick*)).

— **Preamble (setup state)**

Preamble II.

— **Test execution**

The IUT has to provide at least two timers that are further denoted in the test execution timer T1 and timer T2.

1) In the NIT of cycle 7 the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and configures the timer T1 to macrotick n_{int_1} and repetitive mode, activates it, and disables interrupt requests for the corresponding interrupt source.

2) In the NIT of cycle 7, after clearing the interrupt indication flags and configuring T1, the UT configures the timer T2 to macrotick n_{int_2} and repetitive mode, activates it, and disables interrupt requests for the corresponding interrupt source.

3) 2 500 µT after the activation of timer T1 and T2 it is verified (UT) that the interrupt status indication flags are not set for the timer T1 and T2.

4) In cycle 8, it is verified (UT) that the IUT does not generate an interrupt request for the interrupt source T1 and T2 and the interrupt status indication flags for T1 and T2 are set within [0; 100] µT + ξ_{IUT} µT after the timers expire. The UT clears the corresponding interrupt status indication flags after its verification.

5) In cycle 9, at the start of the cycle and before the expiration of the first timer in this cycle the UT enables the interrupt requests for interrupt source T1 and T2.

6) In cycles 9 to cycle 10 after the cycle count wraparound, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T1 within [0; 100] µT + ξ_{IUT} µT after the timer T1 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_1}; n_{int_1} + m_{tol}] right after the interrupt assertion.
It is also verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after the timer T2 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_2}; n_{int_2} + m_{tol}] right after the interrupt assertion.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT indicates an interrupt request at the configured macrotick repetitive in every cycle only if interrupt requests are enabled for the corresponding interrupt source. The interrupt status indication flags are set accordingly.

7.3.9.4 Macrotick only timer (non-repetitive mode)

— Test purpose

Verify the functionality of the timer, the corresponding timer interrupt requests, and the corresponding interrupt status indication flags. It is verified that the IUT provides 2 timers that expire on a configured macrotick in non-repetitive mode.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{int_1} = 3 000 µs / *gdCycle* * *gMacroPerCycle*.

n_{int_2} = 4 000 µs / *gdCycle* * *gMacroPerCycle*.

m_{tol} = ceil(100 µT / (*gdMacrotick* / *pdMicrotick*)).

— Preamble (setup state)

Preamble II.

— Test execution

The IUT has to provide at least two timers that are further denoted in the test execution timer T1 and timer T2.

1) In the NIT of cycle 7 the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources and configures the timer T1 to macrotick n_{int_1} and non-repetitive mode, activates it, and enables interrupt requests for the corresponding interrupt source.

2) In the NIT of cycle 7, after clearing the interrupt indication flags and configuring T1, the UT configures the timer T2 to macrotick n_{int_2} and non-repetitive mode, activates it, and enables interrupt requests for the corresponding interrupt source.

3) 2 500 µT after the activation of timer T1 and T2 it is verified (UT) that the interrupt status indication flags are not set for the timer T1 and T2.

4) In cycle 8, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T1 within [0; 100] µT + ξ_{IUT} µT after the timer T1 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_1}; n_{int_1} + m_{tol}] right after the interrupt assertion.
It is also verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after the timer T2 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_2}; n_{int_2} + m_{tol}] right after the interrupt assertion. The UT clears the interrupt status indication flags for the interrupt source T1 and T2 after its verification.

5) In cycle 9 to cycle 9 after the cycle count wraparound, it is verified (UT) that the IUT does not generate an interrupt request for the interrupt sources T1 and T2.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT indicates an interrupt request for the timer interrupt sources at the configured macrotick once. The interrupt status indication flags are set accordingly.

7.3.9.5 Timer activation and deactivation

— **Test purpose**

Verify correct timer deactivation due to state transition and host request and that the timers can be configured and activated in *POC:normal active* state and in *POC:normal passive* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 211.

Table 211 — Modification to basic configurations for timer and interrupt service – timer activation and deactivation

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1
pAllowPassiveToActive	1
pAllowHaltDueToClock	false

n_{int_1} = 4 000 µs / *gdCycle* * *gMacroPerCycle*.

n_{int_2} = 2 000 µs / *gdCycle* * *gMacroPerCycle*.

m_{tol} = ceil(100 µT / (*gdMacrotick* / *pdMicrotick*)).

— **Preamble (setup state)**

Preamble II.

— **Test execution**

The IUT has to provide at least two timers that are further denoted in the test execution timer T1 and timer T2. This test has to be performed repeatedly with the following transitions between POC states:

Test instance 1 – Transition from *POC:normal active* to *POC:halt* state:

1) In cycles 7 and 8, the LT simulates its startup frame.

2) In cycle 7, the UT configures the timer T1 to cycle count 8, macrotick n_{int_1} and non-repetitive mode, activates it, enables interrupt requests for the corresponding interrupt source and issues the CHI command DEFERRED_HALT latest 2 000 µT before start of NIT.

3) In cycle 8, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:halt* state (*vPOC!State* = *HALT*).

4) It is verified (UT) that the IUT does not generate an interrupt request for the interrupt source T1 for a period of at least *gCycleCountMax * gdCycle*.

Test instance 2 – Transition from *POC:normal passive* to *POC:halt* state:

1) In cycle 7, the LT simulates its startup frame.

2) In cycles 8 and 9, the LT does not simulate its startup frame.

3) In cycle 10, the LT resumes startup frame simulation.

4) In cycle 10, 500 µT after the cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*). The UT configures the timer T1 to cycle count 11 and macrotick n_{int_1} and non-repetitive mode, activates it, enables interrupt requests for the corresponding interrupt source and issues the CHI command DEFERRED_HALT latest 2 000 µT before start of NIT.

5) In cycle 11, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:halt* state (*vPOC!State* = *HALT*).

6) It is verified (UT) that the IUT does not generate an interrupt request for the interrupt source T1 for a period of at least *gCycleCountMax * gdCycle*.

Test instance 3 – Transition from *POC:normal active* to *POC:ready* state:

1) In cycle 7, the LT simulates its startup frame.

2) In cycle 7, the UT configures the timer T1 to cycle count 8, macrotick n_{int_1} and non-repetitive mode, activates it, enables interrupt requests for the corresponding interrupt source and issues the CHI command IMMEDIATE_READY.

3) It is verified (UT) that the IUT is in the *POC:ready* state (*vPOC!State* = *READY*) 2 500 µT after the CHI command IMMEDIATE_READY.

4) It is verified (UT) that the IUT does not generate an interrupt request for the interrupt source T1 for a period of at least *gCycleCountMax * gdCycle*.

Test instance 4 – Transition from *POC:normal passive* to *POC:ready* state:

1) In cycle 7, the LT simulates its startup frame.

2) In cycles 8 and 9, the LT does not simulate its startup frame.

3) In cycle 10, the LT resumes startup frame simulation.

4) In cycle 10, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*). The UT configures the timer T1 to cycle count 11 and macrotick n_{int_1} and non-repetitive mode, activates it, enables interrupt requests for the corresponding interrupt source and issues the CHI command IMMEDIATE_READY.

5) It is verified (UT) that the IUT is in the *POC:ready* state (*vPOC!State* = *READY*) 2 500 µT after the CHI command IMMEDIATE_READY.

6) It is verified (UT) that the IUT does not generate an interrupt request for the interrupt source T1 for a period of at least *gCycleCountMax * gdCycle*.

Test instance 5 – Transition *POC:normal active* to *POC:normal passive* state and vice versa:

1) In cycle 7, the LT simulates its startup frame.

2) In cycle 7, the UT configures the timer T1 to cycle count 10, macrotick n_{int_1} and non-repetitive mode, activates it and enables interrupt requests for the corresponding interrupt source.

3) In cycles 8 and 9, the LT does not send its startup frame.

4) In cycles 10 to 14, the LT simulates its startup frame.

5) In cycle 10, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*)

6) In cycle 10, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T1 within [0; 100] µT + ξ_{IUT} µT after the timer T1 expires at the configured time. It is verified (UT) that the macrotick counter *vMacrotick* is in the range [n_{int_1}; n_{int_1} + m_{tol}] and that the cycle counter *vCycleCounter* is 10.

7) In cycle 10, after the interrupt assertion, the UT clears the corresponding interrupt status indication flag.

8) In cycle 11, the UT configures the timer T1 to cycle count 14, macrotick n_{int_1} and non-repetitive mode, activates it and enables interrupt requests for the corresponding interrupt source.

9) In cycle 14, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

10) In cycle 14, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T1 within [0; 100] µT + ξ_{IUT} µT after the timer T1 expires at the configured time. It is verified (UT) that the macrotick counter *vMacrotick* is in the range [n_{int_1}; n_{int_1} + m_{tol}] and that the cycle counter *vCycleCounter* is 14.

Test instance 6 – Timer deactivation in *POC:normal active* state:

1) In cycle 7, the LT simulates its startup frame.

2) In the dynamic segment of cycle 7, the UT configures the timer T2 to macrotick n_{int_2} and repetitive mode, activates it and enables interrupt requests for the corresponding interrupt source.

3) In cycle 8, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after the timer T2 expires at the configured time. It is verified (UT) that the macrotick counter *vMacrotick* is in the range [n_{int_2}; n_{int_2} + m_{tol}].

4) In the dynamic segment of cycle 8, the UT clears the corresponding interrupt status indication flag and deactivates the timer T2.

5) In cycle 9, it is verified (UT) that no interrupt request is generated for the interrupt source T2.

6) In the dynamic segment of cycle 9, the UT activates the timer T2.

7) In cycle 10, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after the timer T2 expires at the configured time. It is verified (UT) that the macrotick counter *vMacrotick* is in the range [n_{int_2}; n_{int_2} + m_{tol}].

Test instance 7 – Timer deactivation in *POC:normal passive* state:

1) In cycle 7, the LT simulates its startup frame.

2) Beginning with cycle 8, the LT does not simulate its startup frame.

3) In cycle 10, 500 µT after the cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

4) In the dynamic segment of cycle 10, the UT configures the timer T2 to macrotick n_{int_2} and repetitive mode, activates it and enables the corresponding interrupt request.

5) In cycle 11, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after the timer T2 expires at the configured time. It is verified (UT) that the macrotick counter *vMacrotick* is in the range [n_{int_2}; $n_{int\_2} + m_{tol}$].

6) In the dynamic segment of cycle 11, the UT clears the corresponding interrupt status indication flag and deactivates the timer T2.

7) In cycle 12, it is verified (UT) that no interrupt request is generated for the interrupt source T2.

8) In the dynamic segment of cycle 12, the UT activates the timer T2.

9) In cycle 13, it is verified (UT) that the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after the timer T2 expires at the configured time. It is verified (UT) that the macrotick counter *vMacrotick* is in the range [n_{int_2}; $n_{int\_2} + m_{tol}$].

— Postamble

Test instance 1 and 2: None.

Test instance 3 and 4: The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

Test instance 5, 6 and 7: The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

Any transition from the *POC:normal active* state or the *POC:normal passive* state to other states deactivates the timer. The transition from the *POC:normal active* state to the *POC:normal passive* state and vice versa does not deactivate the timer. The UT shall be able to configure and activate the IUT's timers in *POC:normal active* state and in *POC:normal passive* state.

7.3.9.6 Generation of interrupt requests for various interrupt sources

— Test purpose

Verify correct generation of interrupt requests for the minimum number of required interrupt sources of an implementation. It shall be possible for the host to enable / disable the generation of interrupt requests for each interrupt source individually. Further the host can globally disable all interrupt request generation regardless of the individual interrupt request generation status of each interrupt source.

It is also verified that the interrupt status indication flags remain set until reset under the control of the host.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 212.

Table 212 — Modification to basic configurations for timer and interrupt service – generation of interrupt requests for various interrupt sources

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1
pAllowPassiveToActive	1
pAllowHaltDueToClock	false

n_{int_1} = 4 000 µs / *gdCycle* * *gMacroPerCycle*.

n_{int_2} = 2 000 µs / *gdCycle* * *gMacroPerCycle*.

m_{tol} = ceil(100 µT / (*gdMacrotick* / *pdMicrotick*)).

A FIFO buffer (F1) with a depth of at least eight entries is configured only for one available channel in the IUT to receive valid non-null frames in static slots 4 in every cycle. No non-queued receive buffers in the IUT are assigned for reception in slot 4.

Interrupt requests for all mandatory interrupt sources are activated (two timers, cycle start, state transitions and FIFO buffer). According to the IUT documentation for at least one timer the dedicated interrupt request shall be used. The first timer T1 is configured to cycle count 8, macrotick n_{int_1} in non-repetitive mode and the second timer T2 to cycle count 8, macrotick n_{int_2} in non-repetitive mode. The level for notification via interrupt request of the FIFO buffer F1 is set to the smallest possible value > 0, depending on the IUTs possibility for configuration of the notification.

(a) Interrupt request generation is globally enabled; individual interrupt request generation is enabled for each interrupt source

(b) Interrupt request generation is globally disabled; individual interrupt request generation is enabled for each interrupt source

(c) Interrupt request generation is globally enabled; individual interrupt request generation is disabled for each interrupt source

(d) Interrupt request generation is globally disabled; individual interrupt request generation is disabled for each interrupt source

— Preamble (setup state)

Preamble II.

The preamble is supplemented by the following statement: "In the NIT of cycle 6, the UT clears the interrupt status indication flags for all interrupt sources."

— Test execution

1) In cycle 7, it is verified (UT) that

   (a) the IUT generates an interrupt request within [0; 100] µT + ξ_{IUT} µT after the start of the cycle (= interrupt source event), i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [0; m_{tol}] right after the interrupt assertion.

   (b,c,d) the IUT does not generate any interrupt request for the cycle start interrupt source.

   (a,b,c,d) it is also verified (UT) that the interrupt status indication flag for the cycle start interrupt source is set within [0; 100] µT after the interrupt source event occurs.

(a,b) the UT disables individual interrupt request generation for the interrupt source cycle start.

2) Beginning with cycle 8, the LT stops transmission of startup frames in slot 2.

3) In cycle 8, it is verified (UT) that

   (a) the IUT generates an interrupt request for the interrupt source T2 within [0; 100] µT + ξ_{IUT} µT after the timer T2 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_2}; n_{int_2} + m_{tol}] right after the interrupt assertion. It is also verified (UT) that the IUT generates an interrupt request for the interrupt source T1 within [0; 100] µT + ξ_{IUT} µT after the timer T1 expires at the configured time, i.e. it is verified (UT) that the macrotick counter *vMacrotick* is in the range of [n_{int_1}; n_{int_1} + m_{tol}] right after the interrupt assertion.

   (b,c,d) That the IUT does not generate any interrupt request for the interrupt sources T2 and T1.

   (a,b,c,d) It is also verified (UT) that the interrupt status indication flags for the interrupt sources T2 and T1 are set within [0; 100] µT after the interrupt source event occurs.

   (a,b) The UT disables individual interrupt request generation for the interrupt sources T2 and T1.

4) Beginning with cycle 10, the LT starts transmission of startup frames in slot 2 again.

5) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*). It is also verified (UT) that

   (a) the IUT generates an interrupt request for the interrupt source POC state transition at earliest start of the NIT in cycle 9 and latest 100µT after the start of cycle 10 (state transition to the *POC:normal passive* state).

   (b,c,d) The IUT does not generate an interrupt request for the interrupt source POC state transition.

   (a,b,c,d) It is also verified (UT) that the interrupt status indication flag for the interrupt source POC state transition is set within [0; 100] µT after the interrupt source event occurs.

   (a,b) The UT disables individual interrupt request generation for the interrupt source POC state transition.

6) Beginning with cycle 11, the LT simulates one additional frame in static slot 4.

7) In the NIT of the cycle, where the LT has transmitted the amount of frames necessary for causing the available FIFO resources to fall below the configured level it is verified that

   (a) the IUT has generated an interrupt request for the interrupt source FIFO buffer F1 below configured level and that the available resources of the FIFO is below the configured level and
   that available resources of the FIFO reduced by the last stored frame in the FIFO is above the configured level.

   (b,c,d) The IUT did not generate an interrupt request for the interrupt source FIFO buffer F1 below configured level.

8) One cycle after the available FIFO resources are fallen below the configured level it is verified that

(a,b,c,d) the interrupt status indication flags of the interrupt sources cycle start, timer T1 and T2, POC state transition and FIFO buffer F1 below configured level are still set. The UT clears all interrupt status indication flags and it is verified that the interrupt status indication flags of the interrupt sources cycle start, timer T1 and T2, POC state transition and FIFO buffer F1 below configured level are cleared.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT raises an interrupt request related to various interrupt sources only if the interrupt request is globally and individually enabled. The interrupt status indication flags are set irrespective of whether interrupt request generation is enabled or disabled. The interrupt status indication flags remain set until cleared under the control of the host.

7.3.10 Message ID and network management vector

7.3.10.1 Message ID

7.3.10.1.1 Message ID transmission

— **Test purpose**

Verify correct message ID transmission.

— **Applicability**

SC, DC.

— **Configuration**

All Basic Configurations.

n_{dyn} = *gNumberOfStaticSlots* + 2.

The dynamic payload length is 1 two-byte word and the payload is 0x1234.

The IUT transmits an additional frame in dynamic slot n_{dyn}.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) After slot n_{dyn} in cycle 7, the UT triggers the transmission of a frame in dynamic slot n_{dyn} with the payload preamble indicator set to '1'.

2) In cycle 8, it is verified (LT) that the IUT transmits its frame in dynamic slot n_{dyn} with the payload preamble indicator set to '1' and the payload set to 0x1234 (the most significant bit appears first on the FlexRay bus).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a dynamic frame with the payload preamble indicator set to '1' and the payload set accordingly.

7.3.10.1.2 Message ID reception and filtering

— **Test purpose**

Verify correct message ID reception and filtering. The behaviour on reception of a frame in the dynamic segment with the payload preamble indicator is '0' is defined by AdmitWithoutMessageID.

— **Applicability**

SC, DC.

— **Configuration**

All Basic Configurations.

n_{dyn} = *gNumberOfStaticSlots* + 2.

A receive FIFO is configured in the IUT to receive all valid frames in slot n_{dyn} and static slot 5 ($Range1_{min}$ = 5, $Range1_{max}$ = 5, $Range2_{min}$ = n_{dyn}, $Range2_{max}$ = n_{dyn}). The message ID filter of the FIFO buffer is set up in order to accept only frames in the dynamic segment with a message ID of 0x1234 (AdmitWithoutMessageID = false, MsgIDMask = 0x1FFF and MsgIDMatch = 0x1234). Write access to the admittance criteria is enabled during *POC:normal active* state.

No non-queued buffers in the IUT are assigned for reception in slot n_{dyn} and slot 5.

— **Preamble (setup state)**

Preamble II.

The LT simulates an additional frame in static slot 5 and an additional frame in dynamic slot n_{dyn}.

— **Test execution**

1) In cycle 7, the LT transmits an additional frame in slot n_{dyn} and slot 5 with the payload preamble indicator set to '0' and the payload set to 0x1234.

2) In the NIT of cycle 7, it is verified (UT) that the FIFO buffer holds the contents of the frame as simulated by the LT in slot 5 of cycle 7 and that the FIFO buffer does not hold the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 7.

3) In cycle 8, the LT transmits an additional frame in dynamic slot n_{dyn} and in slot 5 with the payload preamble indicator set to '1' and the payload set to 0x3234.

4) In the NIT of cycle 8, it is verified (UT) that the FIFO holds the contents of the frames as simulated by the LT in slot n_{dyn} and slot 5 of cycle 8.

5) In cycle 9, the LT transmits an additional frame in dynamic slot n_{dyn} and slot 5 with the payload preamble indicator set to '1' and the payload set to 0x1235.

6) In the NIT of cycle 9, it is verified (UT) that the FIFO buffer holds the contents of the frame as simulated by the LT in slot 5 of cycle 9 and that the FIFO buffer does not hold the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 9.

7) In the NIT of cycle 9, the UT configures AdmitWithoutMessageID = true.

8) In cycle 10, the LT transmits an additional frame in dynamic slot n_{dyn} and slot 5 with the payload preamble indicator set to '0' and the payload set to 0x1235.

9) In the NIT of cycle 10, it is verified (UT) that the FIFO buffer holds the contents of the frames as simulated by the LT in slot n_{dyn} and slot 5 of cycle 10.

10) In cycle 11, the LT transmits an additional frame in dynamic slot n_{dyn} and slot 5 with the payload preamble indicator set to '1' and the payload set to 0x3234.

11) In the NIT of cycle 11, it is verified (UT) that the FIFO buffer holds the contents of the frames as simulated by the LT in slot n_{dyn} and slot 5 of cycle 11.

12) In cycle 12, the LT transmits an additional frame in dynamic slot n_{dyn} and slot 5 with the payload preamble indicator set to '1' and the payload set to 0x1235.

13) In the NIT of cycle 12, it is verified (UT) that the FIFO buffer holds the contents of the frame as simulated by the LT in slot 5 of cycle 12 and that the FIFO buffer does not hold the contents of the frame as simulated by the LT in slot n_{dyn} of cycle 12.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT applies correct Message ID filtering to dynamic frames and does not apply Message ID filtering to static frames.

7.3.10.2 Network management vector

7.3.10.2.1 Network management vector transmission

— **Test purpose**

Verify correct value of the payload preamble indicator.

— **Applicability**

SC, DC.

— **Configuration**

All Basic Configurations.

The IUT is configured to transmit an additional frame in slot 3.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) In the NIT of cycle 7, the UT sets up the IUT to transmit an additional frame in slot 3 / cycle 8 with the payload preamble indicator set to '1' and the payload set to 0x1234.

2) In cycle 8, it is verified (LT) that the IUT transmits its frame in slot 3 with the payload preamble indicator set to '1' and the payload set to 0x1234 (byte 0 = 0x12, byte 1 = 0x34, the most significant bit of byte 0 appears first on the FlexRay bus).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a static frame with the payload preamble indicator and the payload set accordingly.

7.3.10.2.2 Network management vector length

— **Test purpose**

Verify correct reception of network management vector with different length *gNetworkManagementVectorLength*. Verify that the network management vector length may not be modified in *POC:normal active* state.

— **Applicability**

SC, DC.

— **Configuration**

A receive buffer in the IUT is assigned to slot 2. No receive buffer in the IUT is assigned for slot 3.

All basic configurations using the modifications as listed in Table 213.

Table 213 — Modification to basic configurations for network management vector – network management vector length

Parameter	Modification to all Basic Configurations		
	I	II	III
gNetworkManagementVectorLength [byte]	1	5	12
pNMVectorEarlyUpdate	false		
gMaxWithoutClockCorrectionPassive	1		
gMaxWithoutClockCorrectionFatal	1		
pAllowHaltDueToClock	false		
pAllowPassiveToActive	1		
gPayloadLengthStatic [two-byte word]	6		
payload in slot 2 [a]	0x0102 0x0304 0x0506 0x0708 0x090A 0xFBFC		
payload in slot 3 [b]	0x1020 0x3040 0x5060 0x7080 0x90A0 0xBFCF		
payload in slot ndyn	0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF		

[a] byte 0 = 0x01 on the FlexRay bus.

[b] byte 0 = 0x10 on the FlexRay bus.

n_{dyn} = *gNumberOfStaticSlots* + 1.

— **Preamble (setup state)**

Preamble II.

The LT simulates a startup frame in slot 2, a static frame in slot 3 and a dynamic frame in slot n_{dyn}.

— **Test execution**

1) In cycle 7, the LT simulates its frames in static slot 2, static slot 3 and dynamic slot n_{dyn} with the payload preamble indicator set to '1' and the payload as given in the configuration.

2) In cycle 8, 500 µT after the cycle start and before the NIT, it is verified (UT) that the IUT provides the network management vector of configured length *gNetworkManagementVectorLength*, i.e. the network management vector is 0x11 for *gNetworkManagementVectorLength* = 1 byte, 0x11 0x22 0x33 0x44 0x55 for *gNetworkManagementVectorLength* = 5 byte and 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xAA 0xFF 0xFF for *gNetworkManagementVectorLength* = 12 byte.

3) In cycle 8, the LT simulates its frames in static slots 2 and 3 with the payload preamble indicator set to '0' and the payload as given in the configuration and an additional frame in dynamic slot n_{dyn} with the payload preamble indicator set to '1' and the payload of 6 two-byte words set to 0xFFFF.

4) In cycle 9, 500 µT after the cycle start and before the NIT, it is verified (UT) that the IUT provides a network management vector with all bytes set to 0x00.

5) In cycles 10 and 11, the LT does not simulate any frame.

6) In cycle 12, 500 µT after the cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

7) In cycle 12, the LT simulates its frames in static slot 2, static slot 3 and dynamic slot n_{dyn} with the payload preamble indicator set to '1' and the payload as given in the configuration.

8) In cycle 13, 500 µT after the cycle start and before the NIT, it is verified (UT) that the IUT provides the network management vector of configured length *gNetworkManagementVectorLength*, i.e. the network management vector is 0x11 for *gNetworkManagementVectorLength* = 1 byte, 0x11 0x22 0x33 0x44 0x55 for *gNetworkManagementVectorLength* = 5 byte and 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xAA 0xFF 0xFF for *gNetworkManagementVectorLength* = 12 byte.

9) In cycle 13, the LT simulates its frames in static slots 2 and 3 with the payload preamble indicator set to '0' and the payload as given in the configuration and an additional frame in dynamic slot n_{dyn} with the payload preamble indicator set to '1' and the payload of 6 two-byte words set to 0xFFFF.

10) In cycle 14, 500 µT after the cycle start and before the NIT, it is verified (UT) that the IUT provides a network management vector with all bytes set to 0x00.

11) In cycle 14, it is verified (UT) that the configured value of *gNetworkManagementVectorLength* may not be altered in *POC:normal active* state.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

In *POC:normal active* state and in *POC:normal passive* state, the IUT provides the received network management vector to the UT in the subsequent cycle according to the configured network management vector length *gNetworkManagementVectorLength*. The network management vector length may not be modified in *POC:normal active* state.

7.3.10.2.3 Network management vector update time

— **Test purpose**

Verify correct reception and update of network management vector according to the configuration of *pNMVectorEarlyUpdate*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 214.

Table 214 — Modification to basic configurations for network management vector – network management vector update time

Parameter	Modification to all Basic Configurations	
	I	II
pNMVectorEarlyUpdate	true	false

— **Preamble (setup state)**

Preamble I.

— **Test execution**

$t_{AW} \in [1, 500]$ µT

1) In cycle 8, the LT simulates its frames in static slot 1 with the payload preamble indicator set to '1', the first payload word set to 0x0001 and all other bytes set to 0x00 and an additional frame in the last static slot with the payload preamble indicator set to '1', the first payload word set to 0x0002 and all other payload bytes set to 0x00.

2) I) In cycle 8, within *gdStaticSlot* * *gNumberOfStaticSlots* * *gdMacrotick* / *pdMicrotick* – t_{AW} after the cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x00.
II) In cycle 8, within *gMacroPerCycle* * *gdMacrotick* / *pdMicrotick* – t_{AW} after the cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x00.

3) I) In cycle 8, within *gdStaticSlot* * *gNumberOfStaticSlots* * *gdMacrotick* / *pdMicrotick* + 2 * *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + t_{AW} after cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x03.
II) In cycle 9, within 500 µT + t_{AW} after the cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x03.

4) In cycle 9, the LT simulates its frames in static slot 1 with the payload preamble indicator set to '1', the first payload word set to 0x00 0x04 and all other bytes set to 0x00 and an additional frame in the last static slot with the payload preamble indicator set to '1', the first payload word set to 0x00 0x08 and all other payload bytes set to 0x00.

5) I) In cycle 9, within *gdStaticSlot* * *gNumberOfStaticSlots* * *gdMacrotick* / *pdMicrotick* – t_{AW} after the cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x03.
II) In cycle 9, within *gMacroPerCycle* * *gdMacrotick* / *pdMicrotick* – t_{AW} after the cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x03.

6) I) In cycle 9, within *gdStaticSlot* * *gNumberOfStaticSlots* * *gdMacrotick* / *pdMicrotick* + 2 * *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + t_{AW} after cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x0C.
II) In cycle 10, within 500 µT + t_{AW} after the cycle start, it is verified (UT) that the IUT provides the network management vector 0x00 0x0C.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The CHI provides the previous value of the network management until the soonest allowed update time and provides the new value after the latest allowed update time according to the configuration of *pNMVectorEarlyUpdate*.

7.4 Clock synchronisation

7.4.1 General

For dual channel test execution, all test cases in this subclause shall be executed in three instances, except the test cases 7.4.2.1.1.5, 7.4.2.1.1.13, 7.4.2.1.1.18 to 7.4.2.1.1.22, 7.4.2.1.2.6 and 7.4.2.2.1.6. The LT simulates its frames for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels. The IUT is configured to send frames in *pKeySlotID* on both channels if not otherwise specified.

7.4.2 Synchronisation process

7.4.2.1 Offset correction

7.4.2.1.1 POC:normal active

7.4.2.1.1.1 No deviation (startup node)

— Test purpose

Verify correct behaviour of offset calculation with the IUT as startup node when there is no deviation.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 215.

Table 215 — Modification to basic configurations for POC:normal active – no deviation (startup node)

Parameter	Modification
pClusterDriftDamping [µT]	0

— Preamble (setup state)

Preamble II.

The test has to be performed in 14 instances. For instance 1 the LT simulates no additional sync frame, for instance 2 one additional sync frame, …, and for instance 14 thirteen additional sync frames in successive slots starting with slot 3.

— Test execution

1) In cycles 6, 7 and 8, the LT simulates its frames as within the preamble and begins to simulate additional frames.

2) In cycle 7, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and that *vClockCorrectionFailed* = 0.

3) In cycle 8, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

4) It is verified (LT) that the interval between the IUT's frame in slot 1 / cycle 7 and slot 1 / cycle 8 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

5) In cycle 8, 500 µT after cycle start and before NIT, it is verified (UT) that *vPOC!State* = *NORMAL_ACTIVE*.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs appropriate offset correction and is in *POC:normal active* state at the end of the test execution.

7.4.2.1.1.2 No deviation (integrating node)

— **Test purpose**

Verify correct behaviour of offset calculation with the IUT as integrating node when there is no deviation.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 216.

Table 216 — Modification to basic configurations for POC:normal active – no deviation (integrating node)

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false

— **Preamble (setup state)**

Preamble III.

The test has to be performed in 13 instances. For instance 1 the LT simulates one additional sync frame, for instance 2 two additional sync frames, …, and for instance 13, thirteen additional sync frames in successive slots starting with slot 3.

— **Test execution**

1) In cycles 8, 9 and 10, the LT simulates its frames as within the preamble and begins to simulate additional frames.

2) In cycle 9, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

3) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

4) It is verified (LT) that the interval between the IUT's frame in slot 1 / cycle 9 and slot 1 / cycle 10 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

5) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that *vPOC!State* = *NORMAL_ACTIVE*.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs appropriate offset correction and is in *POC:normal active* state at the end of the test execution.

7.4.2.1.1.3 Deviation inside bounds (startup node)

— **Test purpose**

Verify correct behaviour of offset correction with the IUT as startup node and with varying number of sync frames exhibiting a deviation inside clock correction bounds.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 217.

Table 217 — Modification to basic configurations for POC:normal active – deviation inside bounds (startup node)

Parameter	Modification
pClusterDriftDamping [µT]	0

Δt = 50 µT.

— **Preamble (setup state)**

Preamble II.

(a) The LT does not simulate additional sync frames.

(b) The test has to be performed in 13 instances. For instance 1 the LT simulates one additional sync frame, for instance 2 two additional sync frames, …, and for instance 13 thirteen additional sync frames in successive slots starting with slot 3.

— **Test execution**

1) In cycles 7 to 9, the LT simulates its frames as within the preamble and begins to simulate additional frames.

2) At the beginning of cycle 10, the LT shifts its cycle start by Δt, i.e. all following cycles are shifted.

3) In cycle 11, the LT

(a) starts its cycle still shifted by Δt and shortens its cycle at the end by TruncateTowardsZero(Δt / 2) µT (The LT still transmits its startup frame with the same offset from its cycle start.) and

(b) starts its cycle still shifted by Δt.

4) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT – Δt after cycle start and Δt before NIT), it is verified (UT) that the offset correction value is

(a) *vInterimOffsetCorrection* = TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT and

(b) *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

5) Starting with cycle 12, the LT proceeds to generate cycles with the nominal cycle length. It does not undo any remaining shift.

6) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the offset correction value is

(a) *vInterimOffsetCorrection* = TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT and

(b) *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is

(a) *pMicroPerCycle* + TruncateTowardsZero(Δt / 2) + ξ + ξ_{IUT} µT and

(b) *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT

due to the IUT's offset correction in cycle 11.

8) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

9) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 12 and slot 1 / cycle 13 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

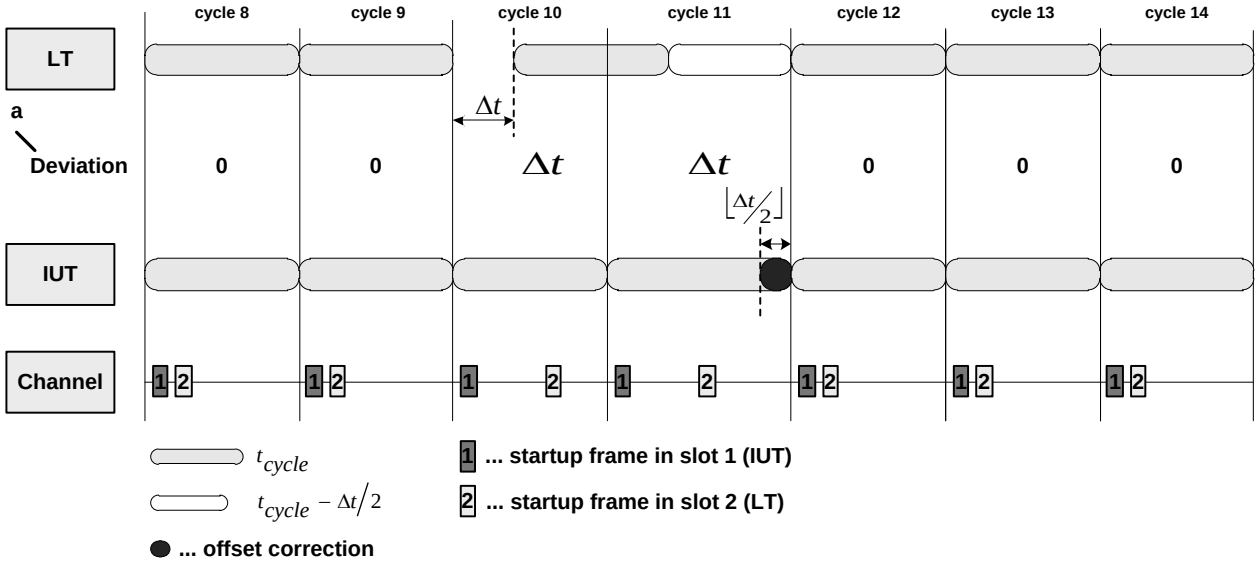
10) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

11) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 13 and slot 1 / cycle 14 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

12) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

13) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

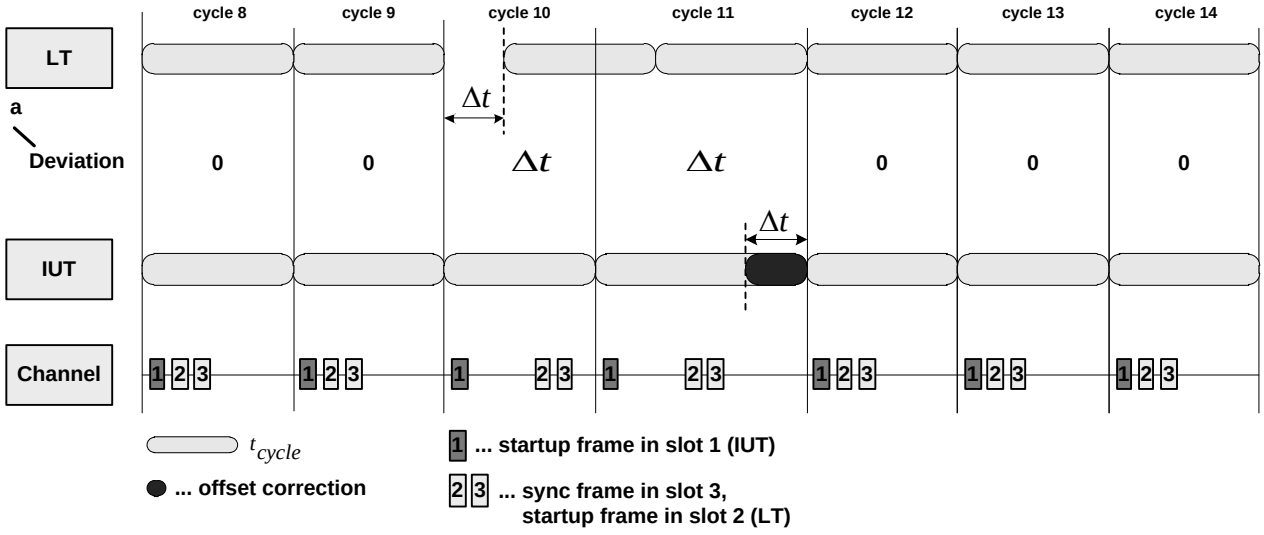
Figure 37 depicts the deviation inside bounds (startup node) – LT simulates startup frame in slot 2.



a Expected deviation measured by the IUT.

Figure 37 — Deviation inside bounds (startup node) – LT simulates startup frame in slot 2

Figure 38 depicts the deviation inside bounds (startup node) – LT simulates startup frame in slot 2 and sync frame in slot 3.



a Expected deviation measured by the IUT.

Figure 38 — Deviation inside bounds (startup node) – LT simulates startup frame in slot 2 and sync frame in slot 3

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate offset correction.

7.4.2.1.1.4 Deviation inside bounds (integrating node)

— **Test purpose**

Verify correct behaviour of offset correction with the IUT as integrating node and with varying number of sync frames exhibiting a deviation inside clock correction bounds.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 218.

Table 218 — Modification to basic configurations for POC:normal active – deviation inside bounds (integrating node)

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pClusterDriftDamping [µT]	0

$\Delta t = 50\ \mu T$.

— **Preamble (setup state)**

Preamble III.

The test has to be performed in 14 instances. For instance 1 the LT simulates no additional sync frame, for instance 2 one additional sync frame, …, and for instance 14 thirteen additional sync frames in successive slots starting with slot 3.

— **Test execution**

1) In cycle 9, the LT simulates its frames as within the preamble and begins to simulate additional frames.

2) At the beginning of cycle 10, the LT shifts its cycle start by Δt, i.e. all following cycles are shifted

3) In cycle 10 within [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick*] µT (in terms of the LT this corresponds to [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100 – Δt; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – Δt] µT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

4) In cycle 11 within [500; 600] µT after cycle start (in terms of the LT this corresponds to [500 – Δt; 600 – Δt] µT after cycle start) and in cycle 11 within [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick*] µT (in terms of the LT this corresponds to [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100 – Δt; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – Δt] µT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

5) In cycle 12 within [500; 600] µT after cycle start and in cycle 12 within [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick*] µT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

6) Starting with cycle 12, the LT's cycle and the IUT's cycle are synchronized again and the LT proceeds to simulate its frames.

7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT due to the IUT's offset correction in cycle 11.

8) In cycle 13 within [500; 600] µT after cycle start and in cycle 13 within [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick*] µT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

9) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 12 and slot 1 / cycle 13 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

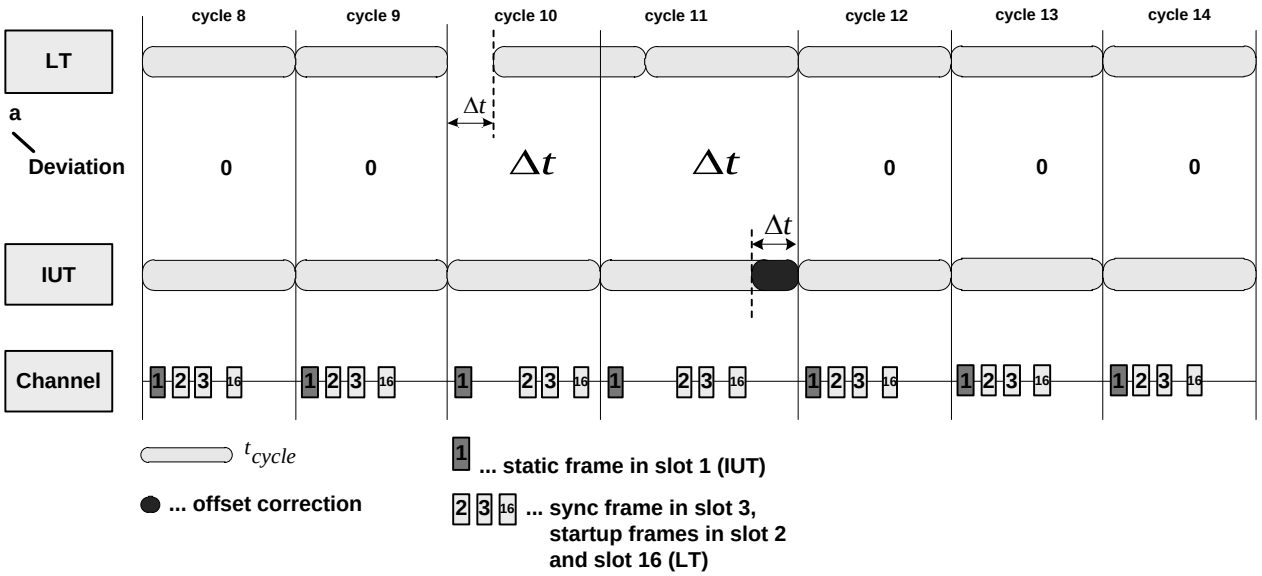
10) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

11) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 13 and slot 1 / cycle 14 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

12) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

13) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 39 depicts the deviation inside bounds (integrating node).



a Expected deviation measured by the IUT.

Figure 39 — Deviation inside bounds (integrating node)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate offset correction.

7.4.2.1.1.5 Different deviation on channel "A&B" inside bounds

— **Test purpose**

Verify correct behaviour of offset correction with the IUT as integrating node and with different deviation on channel A and channel B inside clock correction bounds.

— **Applicability**

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 219.

Table 219 — Modification to basic configurations for POC:normal active – different deviation on channel "A&B" inside bounds

Parameter	Modification
pKeySlotUsedForSync	false
pKeySlotUsedForStartup	false
pClusterDriftDamping [µT]	0

Δt = (a) -35, (b) -20, (c) 40 and (d) 70 µT.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

The test shall be performed four times with Δt = -35, -20, 40 and 70 µT.

1) The LT starts its cycle 10 earlier (Δt < 0) or later (Δt > 0) by ∆t on channel A and 2 * ∆t on channel B, i.e. all following cycles are shifted.

2) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |2 * Δt| after cycle start and |2 * Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT if Δt > 0 and *vInterimOffsetCorrection* = 2 * Δt + ξ_{IUT} µT if Δt < 0.

3) In cycle 12, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |2 * Δt| after cycle start and |2 * Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT if Δt > 0 and *vInterimOffsetCorrection* = 2 * Δt + ξ_{IUT} µT if Δt < 0 and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

4) Starting with cycle 12, the LT proceeds to simulate startup frames in slots 2 and 16.

5) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT if Δt > 0 and *pMicroPerCycle* + 2 * Δt + ξ + ξ_{IUT} µT if Δt < 0 due to the IUT's offset correction in cycle 11.

6) In cycle 13, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |2 * Δt| after cycle start and |2 * Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 12 and slot 1 / cycle 13 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

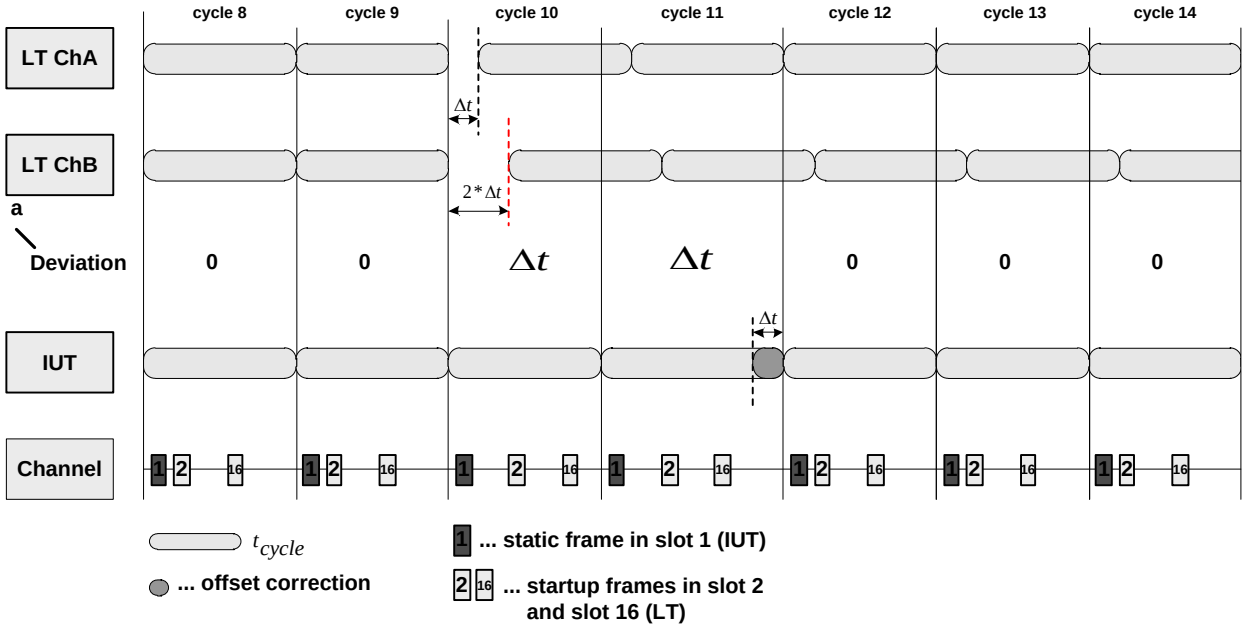
8) In cycle 14, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |2 * Δt| after cycle start and |2 * Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

9) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 13 and slot 1 / cycle 14 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

10) In cycle 15, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |2 * Δt| after cycle start and |2 * Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

11) In cycle 15, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |2 * Δt| after cycle start and |2 * Δt| before NIT), it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 40 depicts the different deviation on channel A&B inside bounds; Δt greater 0.



a Expected deviation measured by the IUT.

Figure 40 — Different deviation on channel A&B inside bounds; Δt greater 0

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate offset correction.

7.4.2.1.1.6 Different deviation inside bounds

— **Test purpose**

Verify correct behaviour of offset correction with the IUT as integrating node and with different deviation inside clock correction bounds.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 220.

Table 220 — Modification to basic configurations for POC:normal active – different deviation inside bounds

Parameter	Modification
pKeySlotUsedForSync	false
pKeySlotUsedForStartup	false
pClusterDriftDamping [µT]	0

Δt =-*pOffsetCorrectionOut*+7, -100, -35, -5, 5, 35, 100 and *pOffsetCorrectionOut*-7µT.

— **Preamble (setup state)**

Preamble III.

The IUT is an integrating node and transmits its frame (no sync frame) in slot 1. The LT simulates in cycle 9 startup frames in slot 2 and in slot 16.

— **Test execution**

The test shall be performed with a Δt of -*pOffsetCorrectionOut*+7, -100, -35, -5, 5, 35, 100 and *pOffsetCorrectionOut*-7 µT.

1) The LT starts its cycle 10 earlier (Δt < 0) or later (Δt > 0), i.e. all following cycles are shifted.

2) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

3) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

4) Starting with cycle 12, the LT's cycle and the IUT's cycle are synchronized again and the LT proceeds to simulate startup frames in slots 2 and 16.

5) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT due to the IUT's offset correction in cycle 11.

6) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 12 and slot 1 / cycle 13 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

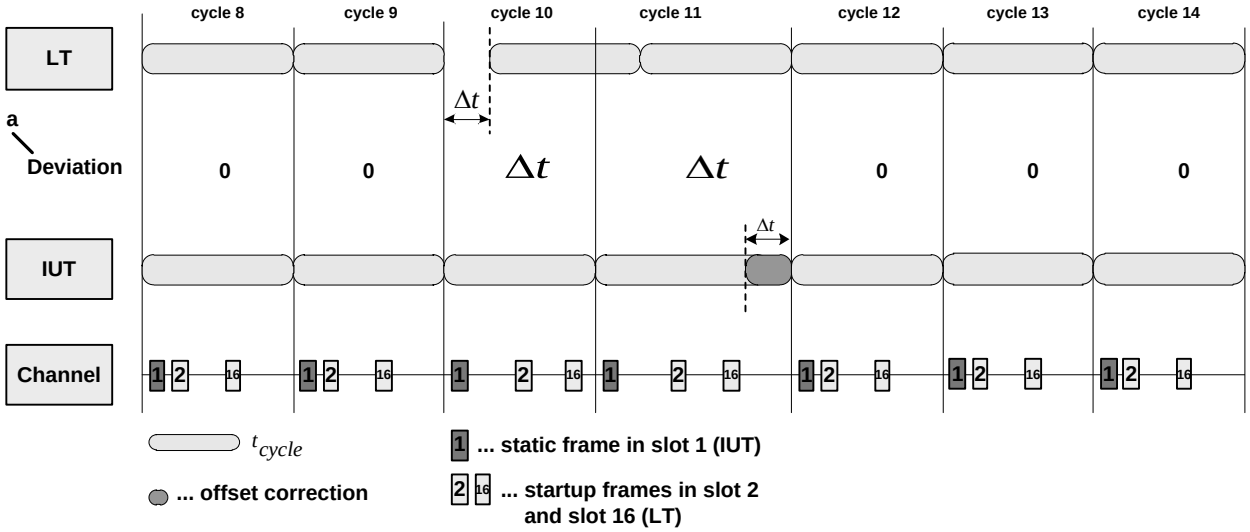
8) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

9) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 13 and slot 1 / cycle 14 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

10) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

11) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 41 depicts the different deviation inside bounds; Δt greater 0.



a Expected deviation measured by the IUT.

Figure 41 — Different deviation inside bounds; Δt greater 0

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate offset correction.

7.4.2.1.1.7 Deviation outside bounds (halt allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node and with deviation outside offset correction bounds. The transition to *POC:halt* state is allowed.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 221.

Table 221 — Modification to basic configurations for POC:normal active – deviation outside bounds (halt allowed)

Parameter	Modification
pOffsetCorrectionOut [µT]	80

— Preamble (setup state)

Preamble II.

The LT simulates two following coldstarters. The LT simulates in cycles 7 to 9 startup frames in slot 2 and in slot 3.

— Test execution

The test shall be performed with a Δt of -100, -87, 87 and 100 µT.

1) The LT starts its cycle 10 earlier (Δt < 0) or later (Δt > 0), i.e. all following cycles are shifted.

2) In cycle 11 of the LT, 500 µT + |Δt| after LT cycle start and |Δt| before LT NIT, it is verified (UT) that the deviation results in an equivalent offset correction value *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT in the IUT.

3) In cycle 11 of the LT, 500 µT + |Δt| after LT cycle start and |Δt| before LT NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the IUT has set the error mode *vPOC!ErrorMode* to COMM_HALT.

4) In cycle 11 of the LT, the LT simulates startup frames in slot 2 and 3.

5) In cycle 11 of the LT, it is verified (LT) that the IUT stops transmission.

Figure 42 depicts the deviation outside bounds (halt allowed); Δt greater 0.

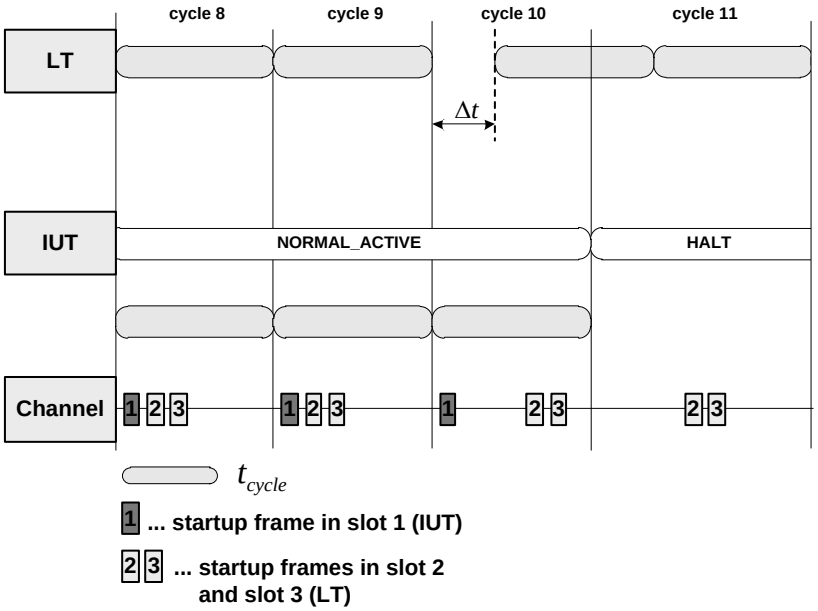


Figure 42 — Deviation outside bounds (halt allowed) ; Δt greater 0

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT recognizes and handles the offset correction violation correctly.

7.4.2.1.1.8 Deviation outside bounds (halt not allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as startup node and with deviation outside offset correction bounds. The transition to *POC:halt* state is not allowed.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 222.

Table 222 — Modification to basic configurations for POC:normal active – deviation outside bounds (halt not allowed)

Parameter	Modification
pOffsetCorrectionOut [µT]	80
pAllowHaltDueToClock	false

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstarters. The LT simulates in cycle 7 to 9 startup frames in slot 2 and in slot 3.

— **Test execution**

The test shall be performed with a Δt of -100, -87, 87 and 100 µT.

1) The LT starts its cycle 10 earlier (Δt < 0) or later (Δt > 0), i.e. all following cycles are shifted.

2) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the deviation results in an equivalent offset correction value *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT in the IUT.

3) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the IUT has set the error mode *vPOC!ErrorMode* to PASSIVE.

4) In cycle 11, the LT simulates startup frames in slots 2 and 3.

5) In cycle 11, it is verified (LT) that the IUT stops transmission.

Figure 43 depicts the deviation outside bounds (halt not allowed);Δt greater 0.

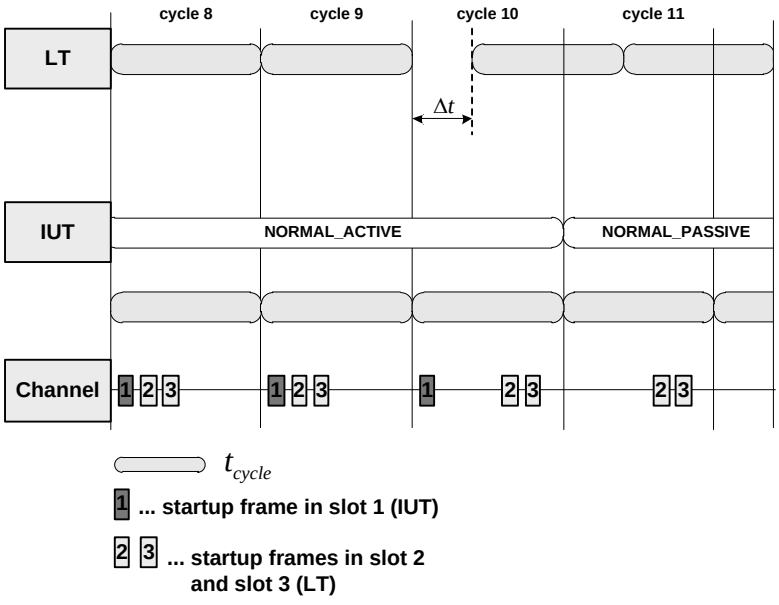


Figure 43 — Deviation outside bounds (halt not allowed); Δt greater 0

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the offset correction violation correctly.

7.4.2.1.1.9 Missing sync frames (sync node, halt allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node when the LT stops startup frame transmission. The transition to *POC:halt* state is allowed.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 223.

Table 223 — Modification to basic configurations for POC:normal active – missing sync frames (sync node, halt allowed)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15

— Preamble (setup state)

Preamble II.

The LT simulates two following coldstart nodes with startup frames in slot 2 and slot 3. The LT continues simulating the startup frames in slot 2 and slot 3 in cycles 6 and 7.

— Test execution

1) Beginning with cycle 8, the LT stops frame transmission.

2) In cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive*.

3) In cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that

   (a) the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE* if *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal* or

(b) the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT* if *gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*. It is also verified (UT) that the clock correction failed counter *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal*.

4) It is verified (LT) that the IUT stops frame transmission in cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive* and does not send anything till the test is finished at the end of cycle 9 + 2 * *gMaxWithoutClockCorrectionFatal*.

5) If *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*:

— In cycle 8 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal – 1*.

— In cycle 8 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT*.

Figure 44 depicts the missing sync frames (sync node, halt allowed) – gMaxWithoutClockCorrectionPassive = 1, gMaxWithoutClockCorrectionFatal = 4.

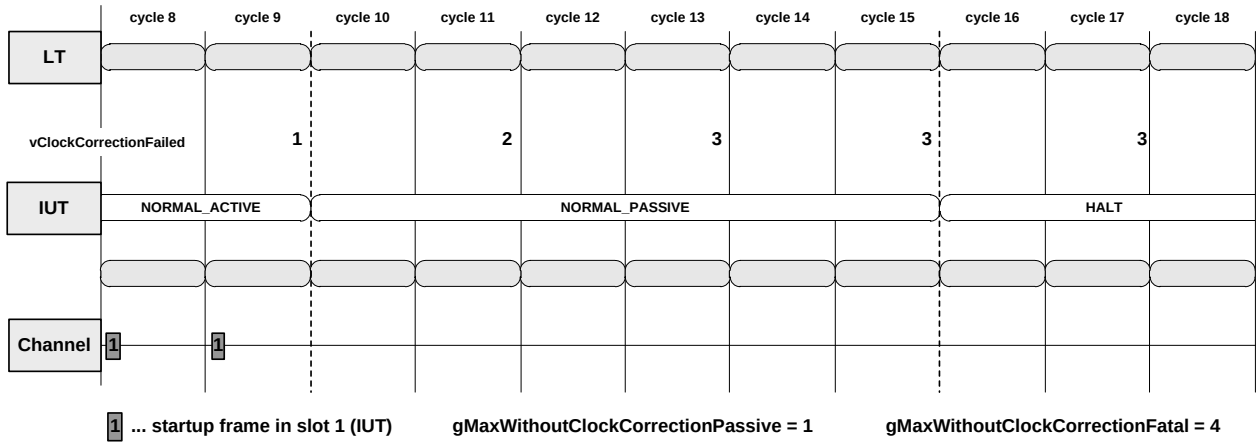


Figure 44 — Missing sync frames (sync node, halt allowed) – gMaxWithoutClockCorrectionPassive = 1, gMaxWithoutClockCorrectionFatal = 4

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.10 Missing sync frames (sync node, halt not allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node when the LT stops startup frame transmission. The transition to *POC:halt* state is not allowed.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 224.

Table 224 — Modification to basic configurations for POC:normal active – missing sync frames (sync node, halt not allowed)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15
pAllowHaltDueToClock	false					

— Preamble (setup state)

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates in cycle 6 and 7 startup frames in slot 2 and in slot 3.

— Test execution

1) In cycle 8, the LT stops frame transmission.

2) In cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive*.

3) In cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE*.

4) It is verified (LT) that the IUT stops frame transmission in cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive* and does not send anything till the test is finished at the end of cycle 9 + 2 * *gMaxWithoutClockCorrectionFatal*.

5) If *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*:
In cycle 8 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1.
In cycle 8 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT stays in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

6) In cycle 9 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is

   (a) *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1, if *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*, or

   (b) *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal*, if *gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*.

Figure 45 depicts the missing sync frames (sync node, halt not allowed).

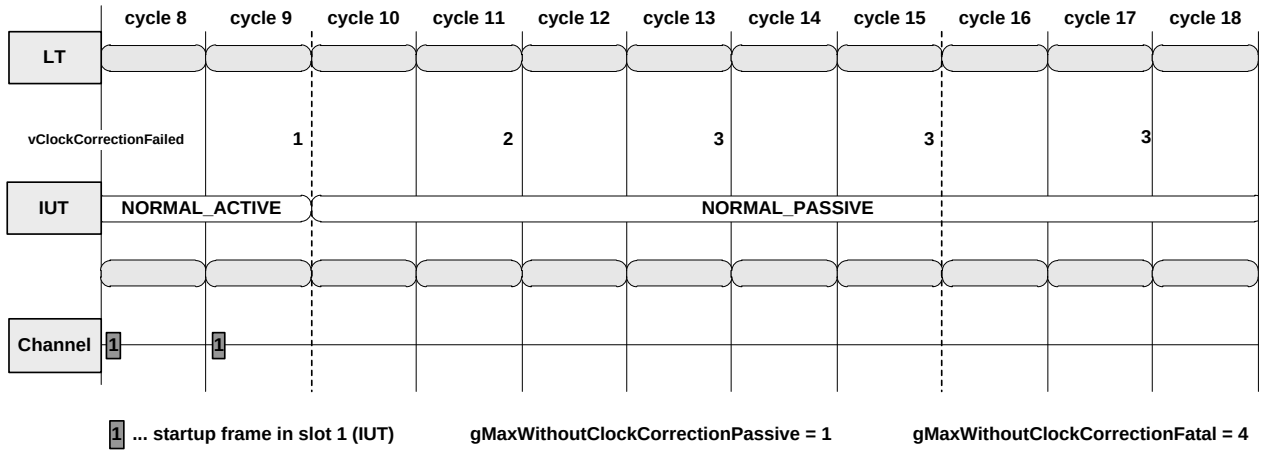


Figure 45 — Missing sync frames (sync node, halt not allowed)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.11 Missing sync frames (no sync node, halt allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as integrating node not sending a sync frame when the LT stops startup frame transmission. The transition to *POC:halt* state is allowed.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 225.

Table 225 — Modification to basic configurations for POC:normal active – missing sync frames (no sync node, halt allowed)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15
pKeySlotUsedForSync	false					
pKeySlotUsedForStartup	false					

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 10, the LT stops frame transmission.

2) In cycle 10 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive*.

3) In cycle 10 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that

   (a) if *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*:
   the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE* or

   (b) if *gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*:
   the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT*. It is also verified (UT) that the clock correction failed counter *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal*.

4) In cycle 10 + 2 * *gMaxWithoutClockCorrectionPassive*, it is verified (LT) that the IUT stops frame transmission and does not send anything in the following cycles.

5) If *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*:
In cycle 10 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1.
In cycle 10 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT*.

Figure 46 depicts the missing sync frames (no sync node, halt allowed).

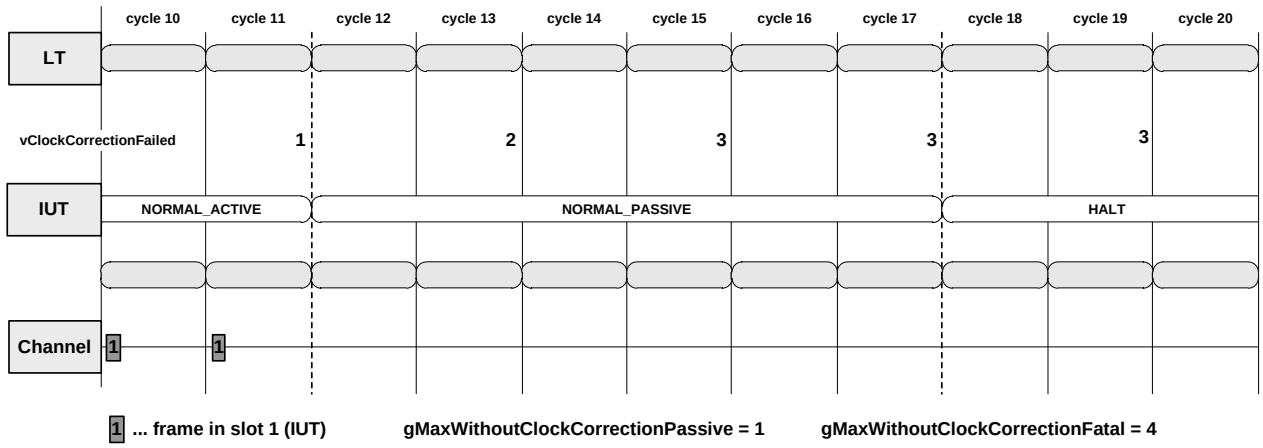


Figure 46 — Missing sync frames (no sync node, halt allowed)

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.12 Missing sync frames (no sync node, halt not allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as integrating node not sending a sync frame when the LT stops startup frame transmission. The transition to *POC:halt* state is not allowed.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 226.

Table 226 — Modification to basic configurations for POC:normal active – missing sync frames (no sync node, halt not allowed)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15
pAllowHaltDueToClock	false					
pKeySlotUsedForSync	false					
pKeySlotUsedForStartup	false					

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 10, the LT stops frame transmission.

2) For n = 12 to 10 + 2 * *gMaxWithoutClockCorrectionPassive*:
In every cycle n, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is incremented, i.e. *vClockCorrectionFailed*(n) = *vClockCorrectionFailed*(n–2) + 1 with *vClockCorrectionFailed*(10) = 0.

3) For n = 12 to 10 + 2 * (*gMaxWithoutClockCorrectionPassive* – 1):
In every cycle n, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the error mode is *vPOC!ErrorMode* = *ACTIVE*. It is also verified (LT) in every cycle that the IUT transmits its frame in slot 1.

4) In cycle 10 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE*.

5) In cycle 10 + 2 * *gMaxWithoutClockCorrectionPassive*, it is verified (LT) that the IUT stops frame transmission and does not send anything till the test is finished at the end of cycle 11 + 2 * *gMaxWithoutClockCorrectionFatal*.

6) If *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*:
In cycle 10 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1.
In cycle 10 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT stays in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

7) In cycle 11 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is:

   (a) *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1, if *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*, or

   (b) *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal*, if *gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*.

Figure 47 depicts the missing sync frames (no sync node, halt not allowed).

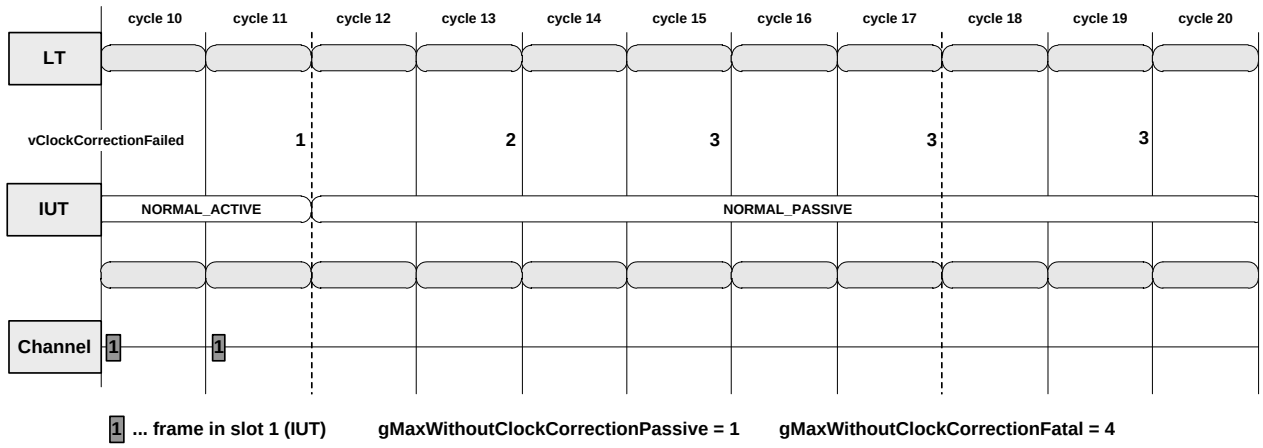


Figure 47 — Missing sync frames (no sync node, halt not allowed)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.13 Content error in sync frames (sync node, halt not allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as startup node when the LT simulates sync frames with content error on one channel. The transition to *POC:halt* state is not allowed. This test case verifies the cross channel sync frame check. A sync frame on one channel is only valid if no content error is present on the other channel.

— **Applicability**

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 227.

Table 227 — Modification to basic configurations for POC:normal active – content error in sync frames (sync node, halt not allowed)

Parameter	Modification					
	I	II	III	IV	V	VI
pAllowHaltDueToClock	false					
gMaxWithoutClockCorrectionPassive	1					
gMaxWithoutClockCorrectionFatal	1					

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstarters. The LT simulates startup frames in slot 2 and in slot 3.

— **Test execution**

This test has to be performed twice with LT startup frames exhibiting content error

(a) on channel A and

(b) on channel B.

1) In cycle 8, the LT starts to simulate startup frames in slot 2 and slot 3 exhibiting content error (cycle count = 0)

   (a) on channel A.

   (b) on channel B.

2) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 1.

3) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE*.

4) It is verified (LT) that the IUT stops frame transmission in cycle 10 and does not send anything in the following 2 cycles.

5) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT stays in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT recognizes and handles the sync frames with content error appropriately. The IUT enters the *POC:normal passive* state due to content errors on a single channel which invalidates also the sync frame on the other channel.

7.4.2.1.1.14 Deviation inside bounds (integrating node, single startup frame)

— **Test purpose**

Verify correct behaviour of offset correction with the IUT as integrating node and with a single startup frame within two succeeding cycles exhibiting a deviation inside clock correction bounds. The IUT shall stay synchronized and shall remain in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 228.

Table 228 — Modification to basic configurations for POC:normal active – deviation inside bounds (integrating node, single startup frame)

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pClusterDriftDamping [µT]	0

Δt = 50 µT.

— **Preamble (setup state)**

Preamble III.

The IUT is an integrating node and transmits its frame (no sync frame) in slot 1.

— **Test execution**

1) In cycle 10 and in cycle 11, the LT simulates only the startup frame in slot 2.

2) At the beginning of cycle 10, the LT shifts its cycle start by Δt, i.e. all following cycles are shifted.

3) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $\Delta t + \xi_{IUT}$ µT. It is also verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

4) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $\Delta t + \xi_{IUT}$ µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = $0 + \xi_{IUT}$ µT. It is also verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

5) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + $\Delta t + \xi + \xi_{IUT}$ µT due to the IUT's offset correction in cycle 11.

6) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $0 + \xi_{IUT}$ µT.

7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 12 and slot 1 / cycle 13 is *pMicroPerCycle* + $\xi + \xi_{IUT}$ µT.

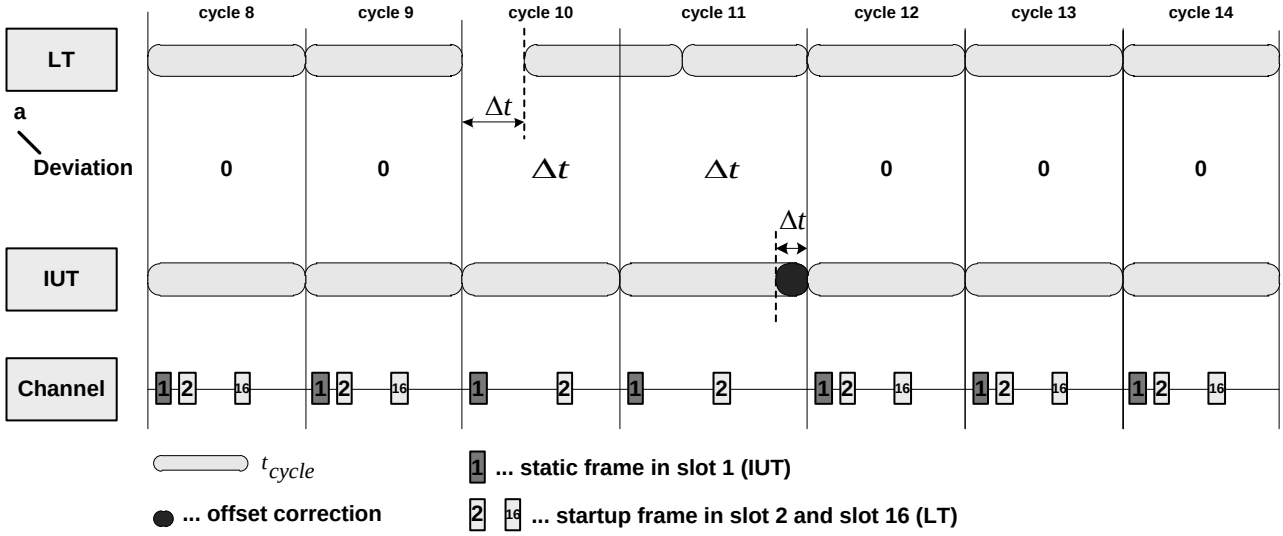
8) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $0 + \xi_{IUT}$ µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = $0 + \xi_{IUT}$ µT.

9) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 13 and slot 1 / cycle 14 is *pMicroPerCycle* + $\xi + \xi_{IUT}$ µT.

10) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $0 + \xi_{IUT}$ µT.

11) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 48 depicts the deviation inside bounds (integrating node, single startup frame).



a Expected deviation measured by the IUT.

Figure 48 — Deviation inside bounds (integrating node, single startup frame)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT stays synchronized and performs appropriate offset correction.

7.4.2.1.1.15 gSyncFrameIDCountMax

— Test purpose

Verify correct behaviour of offset correction with the IUT as integrating node and with different values for the parameter *gSyncFrameIDCountMax*. The IUT shall calculate its offset correction value over a number of *gSyncFrameIDCountMax* sync / startup frames.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 229 and Table 230.

Table 229 — Modification to basic configurations for POC:normal active – gSyncFrameIDCountMax

Parameter	Modification to Basic Configurations														
	1a & 1b					2a & 2b					3				
	I	II	III	IV	V	I	II	III	IV	V	I	II	III	IV	V
gSyncFrameIDCountMax	2	3	7	8	15	2	3	7	8	15	2	3	7	8	15
pKeySlotUsedForSync	false														
pKeySlotUsedForStartup	false														
pClusterDriftDamping [µT]	0														

Table 230 — Shift of the frames by the LT in cycles 10 and 11 for gSyncFrameIDCountMax

Slot Nr.	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Δt [µT]	8	16	32	48	48	64	104	104	104	104	104	104	112	112	112

— **Preamble (setup state)**

Preamble III.

The IUT is an integrating node and transmits its frame (no sync frame) in slot 1. The LT simulates 13 sync frames in adjacent slots starting with slot 3.

Beginning with cycle 8, the LT begins to simulate additional frames.

— **Test execution**

1) In cycle 10 and in cycle 11, the LT deviates its frames according to the above configuration, i.e. the LT shifts its frames by Δt(slot n) (n = 2, 3, ..., 16) within the LT's cycle.

2) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is:

   (I) *vInterimOffsetCorrection* = 12 + ξ_{IUT} µT,

   (II) *vInterimOffsetCorrection* = 16 + ξ_{IUT} µT,

   (III) *vInterimOffsetCorrection* = 40 + ξ_{IUT} µT,

   (IV) *vInterimOffsetCorrection* = 48 + ξ_{IUT} µT and

   (V) *vInterimOffsetCorrection* = 72 + ξ_{IUT} µT.

3) At the beginning of cycle 12, the LT shifts its cycle start by *vInterimOffsetCorrection*. In the LT's cycle 12, the LT simulates its frames without deviation.

4) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT uses *gSyncFrameIDCountMax* sync / startup frames for the calculation of the offset correction value.

7.4.2.1.1.16 Missing sync frame and deviation outside bounds (halt allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node and with deviation outside offset correction bounds. The transition to *POC:halt* state is allowed.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 231.

Table 231 — modification to basic configurations for POC:normal active – missing sync frame and deviation outside bounds (halt allowed)

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
gMaxWithoutClockCorrectionPassive	4		
gMaxWithoutClockCorrectionFatal	4		
pOffsetCorrectionOut [µT]	50		
pClusterDriftDamping [µT]	0		

Repeat this test case with a Δt of -110, and 110 for every basic configuration µT.

— Preamble (setup state)

Preamble II.

— Test execution

1) In cycle 8, the LT does not simulate its startup frame.

2) The LT starts its cycle 9 earlier (Δt < 0) or later (Δt > 0), i.e. all following cycles are shifted.

3) In cycle 10 of the LT, 500 µT + |Δt| after LT cycle start and |Δt| before LT NIT, it is verified (UT) that the deviation results in an equivalent offset correction value *vInterimOffsetCorrection* = Δt / 2 + ξ_{IUT} µT in the IUT. It is also verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

4) In cycle 10 of the LT, 500 µT + |Δt| after LT cycle start and |Δt| before LT NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the IUT has set the error mode *vPOC!ErrorMode* to *COMM_HALT*.

5) In cycle 10 of the LT, it is verified (LT) that the IUT does not transmit anything and that the macrotick counter *vMacrotick* stays constant in the IUT, i.e., the macrotick counter *vMacrotick* holds the same value 0,5 * *gdMacrotick* after LT cycle start, 0,25 * *gdCycle* after LT cycle start, 0,5 * *gdCycle* after LT cycle start and 0,75 * *gdCycle* after LT cycle start.

Figure 49 depicts the missing sync frame and deviation outside bounds (halt allowed); Δt greater 0.

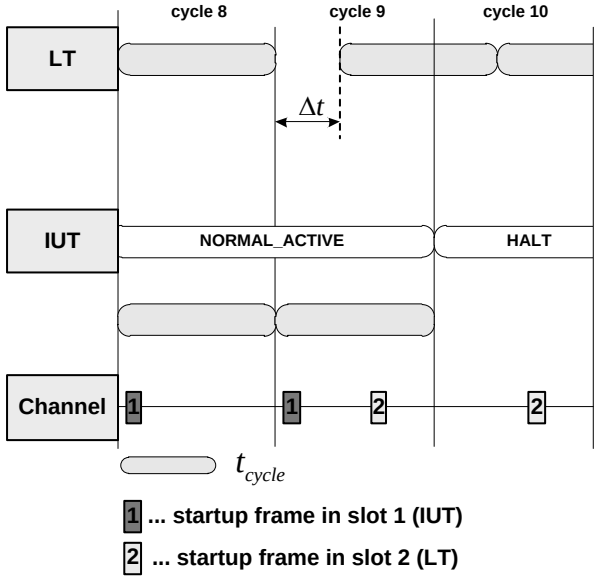


Figure 49 — Missing sync frame and deviation outside bounds (halt allowed); Δt greater 0

— **Postamble**

None, as the IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT recognizes and handles the offset correction violation correctly. The IUT stops frame transmission and macrotick generation and enters *POC:halt* state.

7.4.2.1.1.17 Maximum macrotick shortening

— **Test purpose**

Verify correct behaviour of offset correction when macroticks within the offset correction phase are shortened to *cMicroPerMacroMin*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 232.

Table 232 — Modification to basic configurations for POC:normal active – maximum macrotick shortening

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
pOffsetCorrectionStart [MT]	4 993	4 997	2 498	2 495	2 495
Δt [µT]	-138	-178	-118	-98	-98

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycles 9 to 13, the LT simulates startup frames in slot 2 and in slot 16.

2) In cycles 9 to 13, the IUT transmits its frame (no sync frame) in slot 1.

3) In cycle 10, the LT starts its cycle shifted by ∆t, i.e. all following cycles are shifted.

4) In the static segment of cycle 10 (in terms of LT the static segment is shifted by Δt), the UT requests manual (one-shot) MTS transmission.

5) In cycle 10, it is verified (LT) that
the interval between the IUT's frame in slot 1 and the LT's frame in slot 2 is *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + Δt + *ξ* + ξ_{IUT} µT and
the interval between the beginning of the IUT's frame in slot 1 and the beginning of the IUT's MTS is (*gMacroPerCycle* – *gdSymbolWindow* – *gdNIT* – *gdActionPointOffset* + *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + *ξ* + ξ_{IUT} µT.

6) In cycles 10 and 11, it is verified (LT) that the interval between the beginning of the IUT's MTS and the beginning of the IUT's frame in slot 1 is (*gdSymbolWindow* + *gdNIT* + *gdActionPointOffset* – *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + *ξ* + ξ_{IUT} µT.

7) In the static segment of cycle 11 (in terms of LT the static segment is shifted by ∆t), the UT requests manual (one-shot) MTS transmission.

8) In cycle 11, it is verified (LT) that
the interval between the IUT's frame in slot 1 and the LT's frame in slot 2 is *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + Δt + *ξ* + ξ_{IUT} µT and
the interval between the beginning of the IUT's frame in slot 1 and the beginning of the IUT's MTS is (*gMacroPerCycle* – *gdSymbolWindow* – *gdNIT* – *gdActionPointOffset* + *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + *ξ* + ξ_{IUT} µT.

9) In cycles 11 and 12, it is verified (LT) that the interval between the beginning of the IUT's MTS and the beginning of the IUT's frame in slot 1 is (*gdSymbolWindow* + *gdNIT* + *gdActionPointOffset* – *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + Δt + *ξ* + ξ_{IUT} µT.

10) In cycle 11, within 500 µT after cycle start and before NIT (in terms of LT this approximates to 500 µT + |∆t| after cycle start and until |Δt| after start of the NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

11) In the static segment of cycle 12, the UT requests manual (one-shot) MTS transmission.

12) In cycle 12, it is verified (LT) that
the interval between the IUT's frame in slot 1 and the LT's frame in slot 2 is *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT and
the interval between the beginning of the IUT's frame in slot 1 and the beginning of the IUT's MTS is (*gMacroPerCycle* – *gdSymbolWindow* – *gdNIT* – *gdActionPointOffset* + *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT.

13) In cycles 12 and 13, it is verified (LT) that the interval between the beginning of the IUT's MTS and the beginning of the IUT's frame in slot 1 is (*gdSymbolWindow* + *gdNIT* + *gdActionPointOffset* – *gdSymbolWindowActionPointOffset*) * *gdMacrotick* / *pdMicrotick* + ξ + ξ_{IUT} µT.

14) In cycle 13, within 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The intervals between the IUT's frames and the LT's frames are as expected and the IUT stays synchronized.

7.4.2.1.1.18 Additional verification for missing sync frames (sync node, halt allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling for odd cycle with the IUT as startup node when the LT stops startup frame transmission. The transition to *POC:halt* state is allowed.

— Applicability

DC.

— Configuration

All basic configurations using the modifications as listed in Table 233.

Table 233 — Modification to basic configurations for additional verification for POC:normal active – missing sync frames (sync node, halt allowed)

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1
gMaxWithoutClockCorrectionFatal	4

— Preamble (setup state)

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates in cycle 6 and 7 startup frames in slot 2 and in slot 3.

— Test execution

1) In cycle 8, the LT stops frame transmission.

2) In every cycle starting with cycle 10 and ending with cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is still in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is still *vPOC!ErrorMode* = *PASSIVE*.

Figure 50 depicts the additional verification for missing sync frames (sync node, halt allowed).

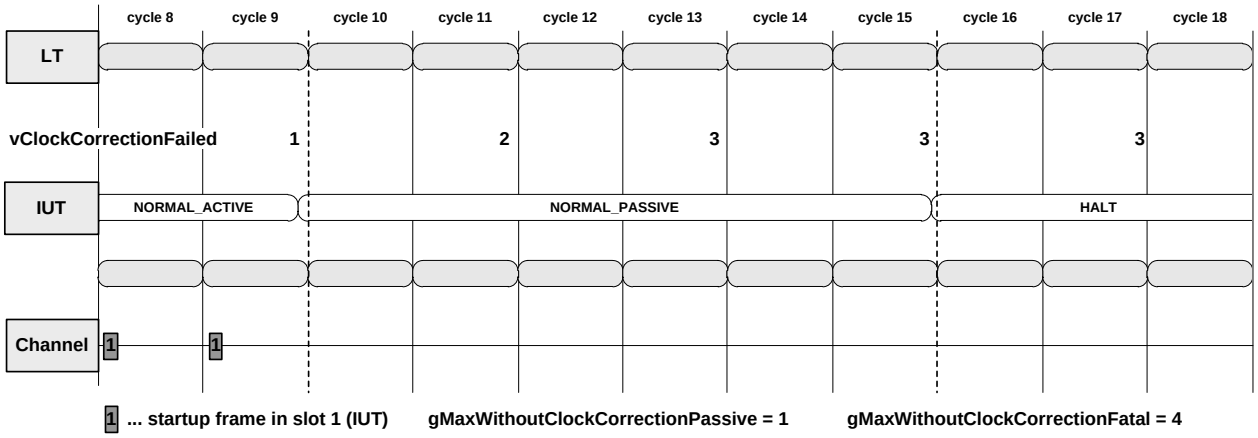


Figure 50 — Additional verification for missing sync frames (sync node, halt allowed)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.19 Missing sync frames (sync node, halt allowed, even cycle)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node when the LT stops startup frame transmission in an even cycle. The transition to *POC:halt* state is allowed. Additionally the correct handling of the synchronisation frame status (derived from the *vsSyncIDListA* and *vsSyncIDListB*) is verified.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 234.

Table 234 — Modification to basic configurations for POC:normal active – missing sync frames (sync node, halt allowed, even cycle)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates in cycle 6 and 7 startup frames in slot 2 and in slot 3.

— **Test execution**

Depending on the available channel(s) the respective received or transmitted sync frames lists shall be verified in the

test execution.

1) In cycle 8, the LT stops frame transmission.

2) In the symbol window of cycle 9, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!state* = *NORMAL_ACTIVE*), that *vClockCorrectionFailed* = 0 and that the following requirements are fulfilled in the synchronisation frame status:
The received or transmitted sync frames list for even cycle contains only the frame ID 1 (the IUT's startup frame ID).
The received or transmitted sync frames list for odd cycle contains the frame ID 1, 2 and 3 (2 and 3 are the LT's startup frame IDs).

3) In cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive,* that the received or transmitted sync frames lists for even and for odd cycle contain only the frame ID 1.
It is also verified (UT) that
(II,III, V) the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode = PASSIVE (here is gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*).
(I,IV, VI) the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode = COMM_HALT (here is gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*).

4) Beginning with the cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, it is verified (LT) that the IUT stops frame transmission and does not send anything till the test is finished at the end of cycle 9 + 2 * *gMaxWithoutClockCorrectionFatal*.

5) (II, III, V) In the symbol window of cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive* + 1, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!state* = *NORMAL_PASSIVE*) and that the following requirements are fulfilled in the synchronisation frame status:
The received or transmitted sync frames list for even cycle does not contain any frame ID.
The received or transmitted sync frames list for odd cycle contains only the frame ID 1.

6) (II, III, V) In cycle 8 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*), that the error mode is *vPOC!ErrorMode* = *COMM_HALT* and that the clock correction failed counter is

vClockCorrectionFailed = *gMaxWithoutClockCorrectionFatal* – 1. It is also verified (UT) that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status do not contain any frame ID.

Figure 51 depicts the missing sync frames (sync node, halt allowed, even cycle).

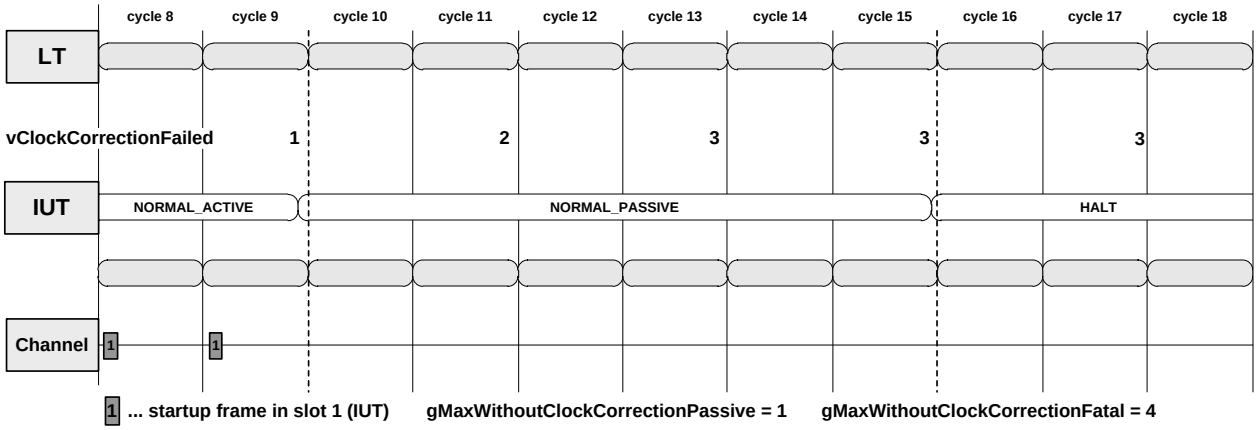


Figure 51 — Missing sync frames (sync node, halt allowed, even cycle)

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.20 Missing sync frames (sync node, halt allowed, odd cycle)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node when the LT stops startup frame transmission in an odd cycle. The transition to *POC:halt* state is allowed. Additionally the correct handling of the synchronisation frame status (derived from the *vsSyncIDListA* and *vsSyncIDListB*) is verified.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 235.

Table 235 — Modification to basic configurations for POC:normal active – missing sync frames (sync node, halt allowed, odd cycle)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15

— **Preamble (setup state)**

Preamble II.

— **Test execution**

Depending on the available channel(s) the respective received or transmitted sync frames lists shall be verified in the test execution.

1) In cycle 9, the LT stops frame transmission.

2) In the symbol window of cycle 9, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!state* = *NORMAL_ACTIVE*), that *vClockCorrectionFailed* = 0 and that the following requirements are fulfilled in the synchronisation frame status:
The received or transmitted sync frames lists for even and for odd cycle contain the frame ID 1 (the IUT's startup frame ID) and the frame ID 2 (the LT's startup frame ID).

3) In cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive*.

   It is also verified (UT) that

   (II, III, V) the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE (*here is *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*).

   (I, IV, VI) the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT* (here is *gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*).

   Further the UT verifies that the following requirements are fulfilled in the synchronisation frame status:

   (I, II, III) the received or transmitted sync frames list for even cycle contains the frame ID 1 and the frame ID 2, and the received or transmitted sync frames list for odd cycle contains only the frame ID 1 (here is *gMaxWithoutClockCorrectionPassive* = 1).

   (IV,V,VI) the received or transmitted sync frames lists for even and for odd cycle contain only the frame ID 1 (here is *gMaxWithoutClockCorrectionPassive* > 1).

4) Beginning with the cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive*, it is verified (LT) that the IUT stops frame transmission and does not send anything till the test is finished at the end of cycle 9 + 2 * *gMaxWithoutClockCorrectionFatal*.

5) (II, III, V) In the symbol window of cycle 8 + 2 * *gMaxWithoutClockCorrectionPassive* + 1, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!state* = *NORMAL_PASSIVE*) and that the following requirements are fulfilled in the synchronisation frame status:

The received or transmitted sync frames list for even cycle does not contain any frame ID.
The received or transmitted sync frames list for odd cycle contains only the frame ID 1.

6) (II, III, V) In cycle 8 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*), that the error mode is *vPOC!ErrorMode* = *COMM_HALT* and that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1. It is also verified (UT) that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status do not contain any frame ID.

Figure 52 depicts the missing sync frames (sync node, halt allowed, odd cycles).

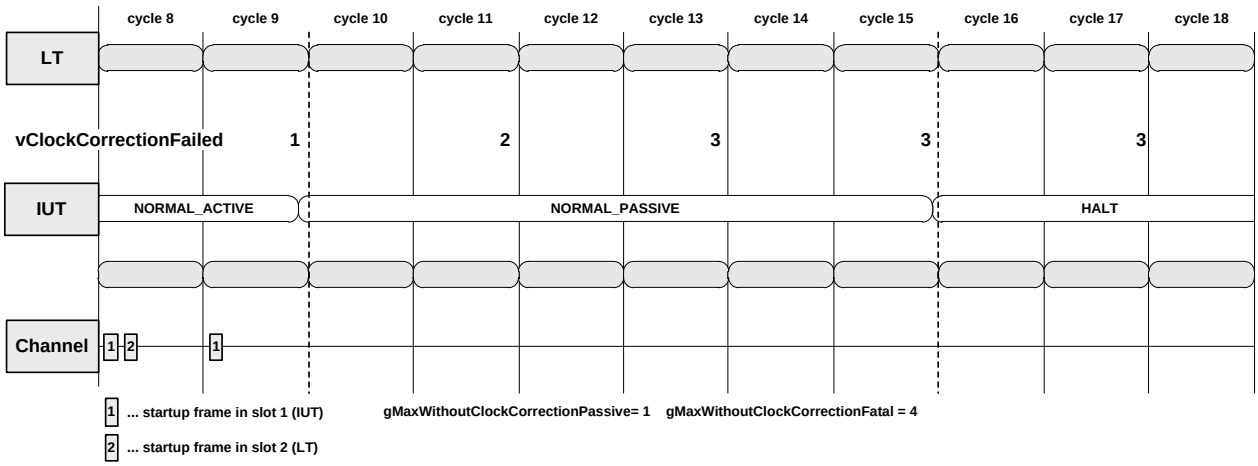


Figure 52 — Missing sync frames (sync node, halt allowed, odd cycles)

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.21 Missing sync frames (no sync node, halt allowed, even cycle, remain one sync frame)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as integrating node not sending a sync frame when the LT stops startup frame transmission in an even cycle. The transition to *POC:halt* state is allowed. Additionally the correct handling of the synchronisation frame status (derived from the *vsSyncIDListA* and *vsSyncIDListB*) is verified.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 236.

Table 236 — Modification to basic configurations for POC:normal active – missing sync frames (no sync node, halt allowed, even cycle, remain one sync frame)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15
pKeySlotUsedForSync	false					
pKeySlotUsedForStartup	false					

— **Preamble (setup state)**

Preamble III.

The LT simulates in cycle 9 startup frames in slot 2 and in slot 16.

— **Test execution**

Depending on the available channel(s) the respective received or transmitted sync frames lists shall be verified in the test execution.

1) In cycle 10, the LT stops startup frame transmission in slot 16.

2) In the symbol window of cycle 11, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!state* = *NORMAL_ACTIVE*), that *vClockCorrectionFailed* = 0 and that the following requirements are fulfilled in the synchronisation frame status:
The received or transmitted sync frames list for odd cycle contains the frame ID 2 and the ID 16 (the LT's startup frame Ids).
The received or transmitted sync frames list for even cycle contains only the frame ID 2.

3) In cycle 12, the LT also stops startup frame transmission in slot 2.

4) In the symbol window of cycle 12, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!state* = *NORMAL_ACTIVE*), that *vClockCorrectionFailed* = 0 and that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status contain only the frame ID 2.

5) In the symbol window of cycle 12 + *gMaxWithoutClockCorrectionPassive*, it is verified (UT) that IUT is in the *POC:normal active* state (*vPOC!state* = *NORMAL_ACTIVE*) and that the following requirements are fulfilled in the synchronisation frame status:

   (I, II, III) The received or transmitted sync frames list for odd cycle contains only the frame ID 2 and for even cycle does not contain any frame ID (here is *gMaxWithoutClockCorrectionPassive* = 1).

   (IV, V, VI) The received or transmitted sync frames lists for even and for odd cycle do not contain any frame ID (here is *gMaxWithoutClockCorrectionPassive* > 1).

6) In cycle 12 + 2 * *gMaxWithoutClockCorrectionPassive*, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive* and that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status do not contain any frame ID. It is also verified (UT) that

(II, III, V) the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE (*here is *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*).

(I, IV, VI) the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT* (here is *gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*).

7) Beginning with the cycle 12 + 2 * *gMaxWithoutClockCorrectionPassive*, it is verified (LT) that the IUT stops frame transmission and does not send anything till the test is finished at the end of cycle 13 + 2 * *gMaxWithoutClockCorrectionFatal*.

8) (II, III, V) In the symbol window of cycle 12 + 2 * *gMaxWithoutClockCorrectionPassive* + 1, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!state* = *NORMAL_PASSIVE*) and that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status do not contain any frame ID (here is *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*).

9) (II, III, V) In cycle 12 + 2 * *gMaxWithoutClockCorrectionFatal*, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*), that the error mode is *vPOC!ErrorMode* = *COMM_HALT* and that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1. It is also verified (UT) that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status do not contain any frame ID.

Figure 53 depicts the missing sync frames (no sync node, halt allowed, even cycle, remain one sync frame).

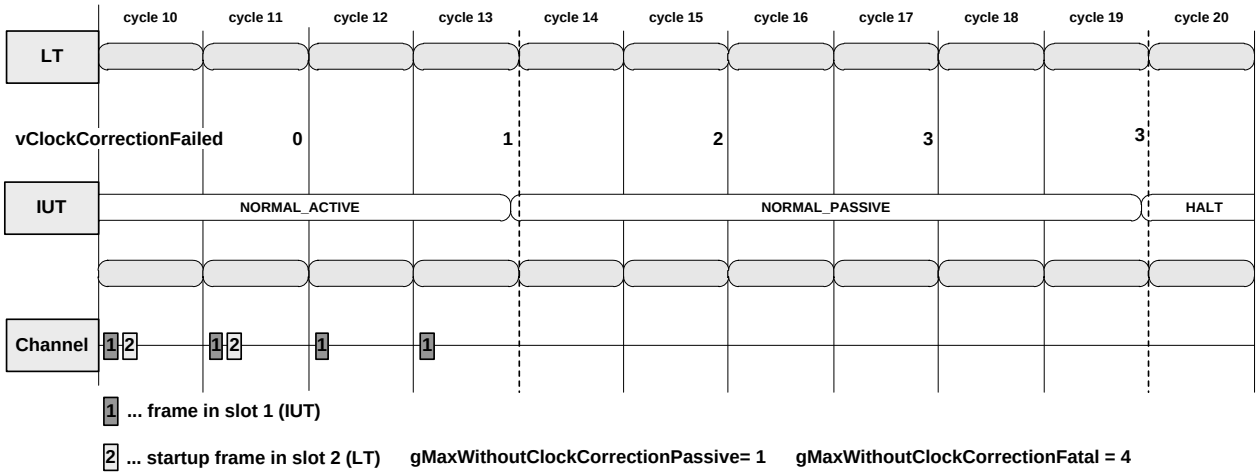


Figure 53 — Missing sync frames (no sync node, halt allowed, even cycle, remain one sync frame)

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.1.22 Missing sync frames (no sync node, halt allowed, odd cycle, remain one sync frame)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as integrating node not sending a sync frame when the LT stops startup frame transmission in an odd cycle. The transition to *POC:halt* state is allowed. Additionally the correct handling of the synchronisation frame status (derived from the *vsSyncIDListA* and *vsSyncIDListB*)(*vsSyncIDListA, vsSyncIDListB*) is verified.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 237.

Table 237 — Modification to basic configurations for POC:normal active – missing sync frames (no sync node, halt allowed, odd cycle, remain one sync frame)

Parameter	Modification					
	I	II	III	IV	V	VI
gMaxWithoutClockCorrectionPassive	1	1	1	4	4	15
gMaxWithoutClockCorrectionFatal	1	4	15	4	15	15
pKeySlotUsedForSync	false					
pKeySlotUsedForStartup	false					

— Preamble (setup state)

Preamble III.

The LT simulates in cycle 9 and cycle 10 startup frames in slot 2 and in slot 16.

— Test execution

Depending on the available channel(s) the respective received or transmitted sync frames lists shall be verified in the test execution.

1) In cycle 11, the LT stops startup frame transmission in slot 16.

2) In the symbol window of cycle 12, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!state* = *NORMAL_ACTIVE*), that *vClockCorrectionFailed* = 0 and that the following requirements are fulfilled in the synchronisation frame status:
The received or transmitted sync frames list for even cycle contains the frame ID 2 and the frame ID 16 (the LT's startup frame Ids).
The received or transmitted sync frames list for odd cycle contains only the frame ID 2.

3) In cycle 13, the LT also stops startup frame transmission in slot 2.

4) In the symbol window of cycle 13, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!state* = *NORMAL_ACTIVE*) and that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status contain only the frame ID 2.

5) In the symbol window of cycle 12 + 2 * *gMaxWithoutClockCorrectionPassive*, it is verified (UT) that clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive*.
It is also verified (UT) that

   (II, III, V) the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE* (here is *gMaxWithoutClockCorrectionPassive* < *gMaxWithoutClockCorrectionFatal*)

   (I, IV, VI) the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC! ErrorMode* = *COMM_HALT* (here is *gMaxWithoutClockCorrectionPassive* = *gMaxWithoutClockCorrectionFatal*).
   Further the UT verifies that the following requirements are fulfilled in the synchronisation frame status:

   (I, II, III) The received or transmitted sync frames list for even cycle contains the frame ID 2 and for odd cycle does not contain any frame ID (here is *gMaxWithoutClockCorrectionPassive* = 1).

   (IV, V, VI) The received or transmitted sync frames lists for even and for odd cycle do not contain any frame ID (here is *gMaxWithoutClockCorrectionPassive* > 1).

6) Beginning with the cycle 12 + 2 * *gMaxWithoutClockCorrectionPassive*, it is verified (LT) that the IUT stops frame transmission and does not send anything till the test is finished at the end of cycle 13 + 2 * *gMaxWithoutClockCorrectionFatal*.

7) (II, III, V) In the symbol window of cycle 12 + 2 * *gMaxWithoutClockCorrectionPassive* + 1, it is verified (UT) that the IUT is in the *POC:normal passive* state (*vPOC!state* = *NORMAL_PASSIVE*) and that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status do not contain any frame ID.

8) (II, III, V) In cycle 12 + 2 * *gMaxWithoutClockCorrectionFatal,* 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT* and that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionFatal* – 1.It is also verified (UT) that the received or transmitted sync frames lists for even and for odd cycle in the synchronisation frame status do not contain any frame ID.

Figure 54 depicts the missing sync frames (no sync node, halt allowed, odd cycle, remain one sync frame).

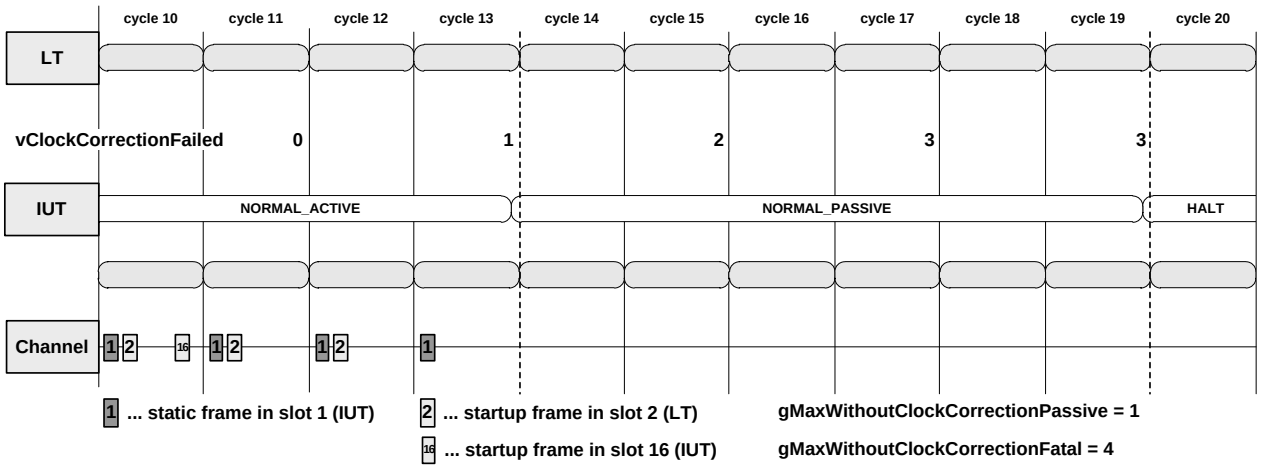


Figure 54 — Missing sync frames (no sync node, halt allowed, odd cycle, remain one sync frame)

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT recognizes and handles the missing traffic correctly.

7.4.2.1.2 POC:normal passive

7.4.2.1.2.1 Resynchronization (transition to POC:normal active allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as startup node. The transition from *POC:normal passive* state to *POC:normal active* state is allowed. The IUT shall return from *POC:normal passive* state to *POC:normal active* state after the offset correction stayed within bounds for *pAllowPassiveToActive* even / odd cycle pairs. Verify that interrupts are indicated after the state transitions from the *POC:normal passive* to the *POC:normal active* state and vice versa.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 238.

Table 238 — Modification to basic configurations for POC:normal passive – resynchronization (transition to POC:normal active allowed)

Parameter	I	II	III
pAllowPassiveToActive	1	16	31
pAllowHaltDueToClock	false		
pOffsetCorrectionOut	80		

Δt = 100 µT.

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates an additional startup frame in slot 3.

The preamble is followed by:

1) At the start of cycle 6 the UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

2) In cycles 6 to 11, the LT simulates its startup frames as in the preamble.

3) In cycle 9, 500 µT after cycle start and before NIT, and in cycle 10, 500 µT after cycle start and before NIT, it is checked (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

4) The LT delays the start of cycle 10 by Δt, i.e. all following cycles start delayed.

— **Test execution**

1) In cycle 10, 500 µT after cycle start and before cycle end (in terms of LT this appropriates to 500 µT + |Δt| after cycle start and |Δt| before cycle end), the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

2) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the IUT has entered the *POC:normal passive* state (*POC!State* = *NORMAL_PASSIVE*) and that the IUT has set the error mode *vPOC!ErrorMode* to *PASSIVE*.
t is also verified (UT) that the IUT has set the proper interrupt status indication within [100; 200] µT after the interrupt source event occurs and that the IUT has raised the proper interrupt request within [0; 100] µT + ξ_{IUT} µT after the state transition to the *POC:normal passive* state. The UT clears the interrupt status indication after its verification.

3) It is verified (LT) that the IUT stops frame transmission in cycle 11.

4) In cycles 12 to (13 + 2 * *pAllowPassiveToActive*)mod 64, the LT continues to simulate its frames.

5) In cycle 12, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

6) In cycle 13, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), and in cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt – *pOffsetCorrectionOut* + ξ_{IUT} µT.

7) In cycle 14 within [500; 600] µT after cycle start, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 0.

8) If *pAllowPassiveToActive* > 1: In every even cycle starting with cycle 14 and ending with cycle (10 + 2 * *pAllowPassiveToActive*)mod 64 within [500; 600] µT after cycle start, it is verified (UT) that the IUT has incremented the counter *vAllowPassiveToActive* (*vAllowPassiveToActive* = 1 in cycle 14, *vAllowPassiveToActive* = *pAllowPassiveToActive* – 1 in cycle (10 + 2 * *pAllowPassiveToActive*)mod 64).

9) In cycle (11 + 2 * *pAllowPassiveToActive*)mod 64, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT did not set any interrupt status indication and that the IUT did not raise any interrupt request (note that in terms of LT cycle 13 still has a deviation of Δt – *pOffsetCorrectionOut* compared to IUTs schedule).

10) In cycle (12 + 2 * *pAllowPassiveToActive*)mod 64 within [500; 800] µT after cycle start, it is verified (UT) that the IUT has set the counter *vAllowPassiveToActive* to *pAllowPassiveToActive* – 1, that the IUT has returned to the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the IUT has set the *vPOC!ErrorMode* to *ACTIVE*.
It is also verified (UT) that the IUT has set the proper interrupt status indication within [100; 200] µT after the interrupt source event occurs and that the IUT has raised the proper interrupt request within [0; 100] µT + ξ_{IUT} µT after the state transition to the *POC:normal active* state.

11) In cycle (12 + 2 * *pAllowPassiveToActive*)mod 64, it is verified (LT) that the IUT resumes frame transmission.

12) In cycle (13 + 2 * *pAllowPassiveToActive*)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

Figure 55 depicts the resynchronization (transition to *POC:normal active allowed*) – *pAllowPassiveToActive* = 1.

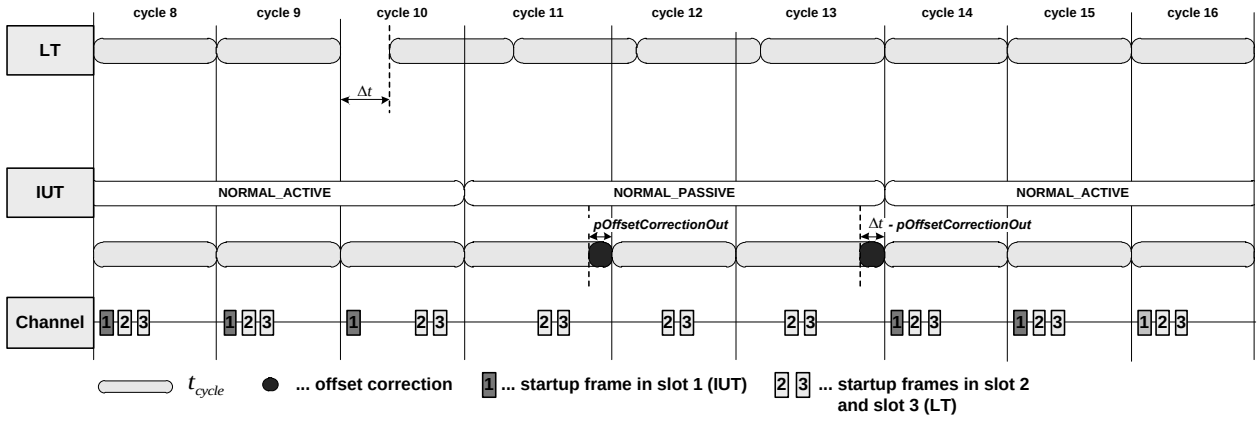


Figure 55 — Resynchronization (transition to POC:normal active allowed) – pAllowPassiveToActive = 1

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the offset correction violation accordingly. The IUT returns to *POC:normal active* state after the offset correction stayed within bounds for *pAllowPassiveToActive* even / odd cycle pairs. The clock correction failed counter *vClockCorrectionFailed* is reset to 0 after the first odd cycle with clock correction results within bounds. The counter *vAllowPassiveToActive* counts correctly. The IUT indicates an interrupt after it entered the *POC:normal active* and the *POC:normal passive* state.

7.4.2.1.2.2 Resynchronization (transition to POC:normal active not allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node. The transition from *POC:normal passive* state to *POC:normal active* state is not allowed. The IUT shall remain in *POC:normal passive* state even after the offset correction stayed within bounds for an even / odd cycle pair.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 239.

Table 239 — Modification to basic configurations for POC:normal passive – resynchronization (transition to POC:normal active not allowed)

Parameter	Modification
pAllowHaltDueToClock	false
pOffsetCorrectionOut	80

Δt = 100 µT.

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates an additional startup frame in slot 3, an continues sending this frame until the end of the Test.

The preamble is followed by:

1) In cycle 9, 500 µT after cycle start and before NIT, and in cycle 10, 500 µT after cycle start and before NIT (in terms of the LT for cycle 10 this approximates to 500 µT – Δt after cycle start and Δt before NIT), it is checked (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

2) The LT delays the start of cycle 10 by Δt, i.e. all following cycles start delayed.

3) In cycle 11, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), it is checked (UT) that the IUT has entered the *POC:normal passive* state (*POC!State* = *NORMAL_PASSIVE*) and that the IUT has set the error mode *vPOC!ErrorMode* to *PASSIVE*.

— **Test execution**

1) It is verified (LT) that the IUT stops frame transmission in cycle 11.

2) For cycle 12 and the following 2 + 2 * (*pAllowPassiveToActive* + 1) cycles, the LT simulates startup frames in slots 2 and 3 with the existing shift.

3) In cycle 12, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT.

4) In cycle 13, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), and in cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt – *pOffsetCorrectionOut* + ξ_{IUT} µT.

5) For cycle 14 and the following 2 * (*pAllowPassiveToActive* + 1) cycles, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is still in in the *POC:normal passive* state (*POC!State* = *NORMAL_PASSIVE*), that the clock correction failed counter is *vClockCorrectionFailed* = 0 and that the counter *vAllowPassiveToActive* is 0.

Figure 56 depicts the resynchronization (transition to *POC:normal active not allowed*) – *pAllowPassiveToActive* = 0.

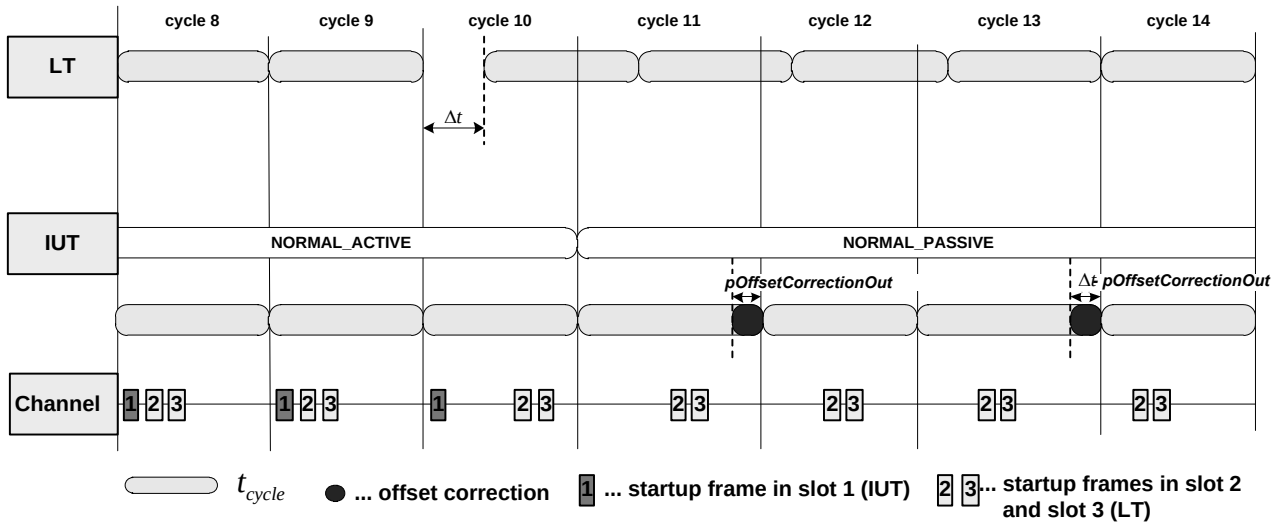


Figure 56 — Resynchronization (transition to POC:normal active not allowed) – pAllowPassiveToActive = 0

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the offset correction violation correctly. The IUT remains in *POC:normal passive* state even after the offset correction stayed within bounds for an even / odd cycle pair. The clock correction failed counter *vClockCorrectionFailed* is reset to 0 after the first odd cycle with clock correction results within bounds. The counter *vAllowPassiveToActive* stays 0.

7.4.2.1.2.3 Resynchronization after resumption of sync frame simulation (transition to POC:normal active allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node when the LT resumes frame simulation. The transition to *POC:normal active* state is allowed.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 240.

Table 240 — Modification to basic configurations for POC:normal passive – resynchronization after resumption of sync frame simulation (transition to POC:normal active allowed)

Parameter	Modification		
	I	II	III
pAllowPassiveToActive	1	16	31
pAllowHaltDueToClock	false		
gMaxWithoutClockCorrectionPassive	1		
gMaxWithoutClockCorrectionFatal	1		

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates an additional startup frame in slot 3. The IUT is configured to transmit an additional static frame in slot 4 on the channel(s) that the LT is simulating startup frame(s) on.

The preamble is supplemented by:

1) In cycle 4, the LT begins to simulate startup frames in slot 2 and in slot 3.

2) In cycle 8, the LT stops frame simulation.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive*. It is also verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE*.

2) In cycle 10, it is verified (LT) that the IUT stops frame transmission.

3) In cycle 12, the LT resumes frame transmission

   (a) of startup frames in slot 2 and slot 3.

   (b) of a startup frame in slot 2 and a sync frame in slot 3.

4) If *pAllowPassiveToActive* > 1: In every even cycle starting with cycle 14 and ending with cycle (10 + 2 * *pAllowPassiveToActive*)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has incremented the counter *vAllowPassiveToActive* (*vAllowPassiveToActive* = 1 in cycle 14, *vAllowPassiveToActive* = *pAllowPassiveToActive* – 1 in cycle (10 + 2 * *pAllowPassiveToActive*)mod 64).

5) In cycle (10 + 2 * *pAllowPassiveToActive*)mod 64, within the interval t_{AW} after the offset correction start + 10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames list(s) for the even cycle on the respective channel(s) contains frame IDs 2 and 3.

6) In cycle (11 + 2 * *pAllowPassiveToActive*)mod 64, within the interval t_{AW} after the offset correction start + 10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames list(s) for the odd cycle on the respective channel(s) contains frame IDs 2 and 3.

7) In cycle (12 + 2 * *pAllowPassiveToActive*)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has set the counter *vAllowPassiveToActive* to *pAllowPassiveToActive* – 1, that the IUT has returned to the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the IUT has set the *vPOC!ErrorMode* to *ACTIVE*.

8) In cycle (12 + 2 * *pAllowPassiveToActive*)mod 64, it is verified (LT) that the IUT resumes frame transmission in slot 1 and slot 4. It is also verified (UT) that the received or transmitted sync frames list(s) for the even cycle on the respective channel(s) contains frame IDs 1, 2 and 3.

Figure 57 depicts the resynchronization after resumption of sync frame simulation (transition to *POC:normal active allowed*) – *pAllowPassiveToActive* = 1.

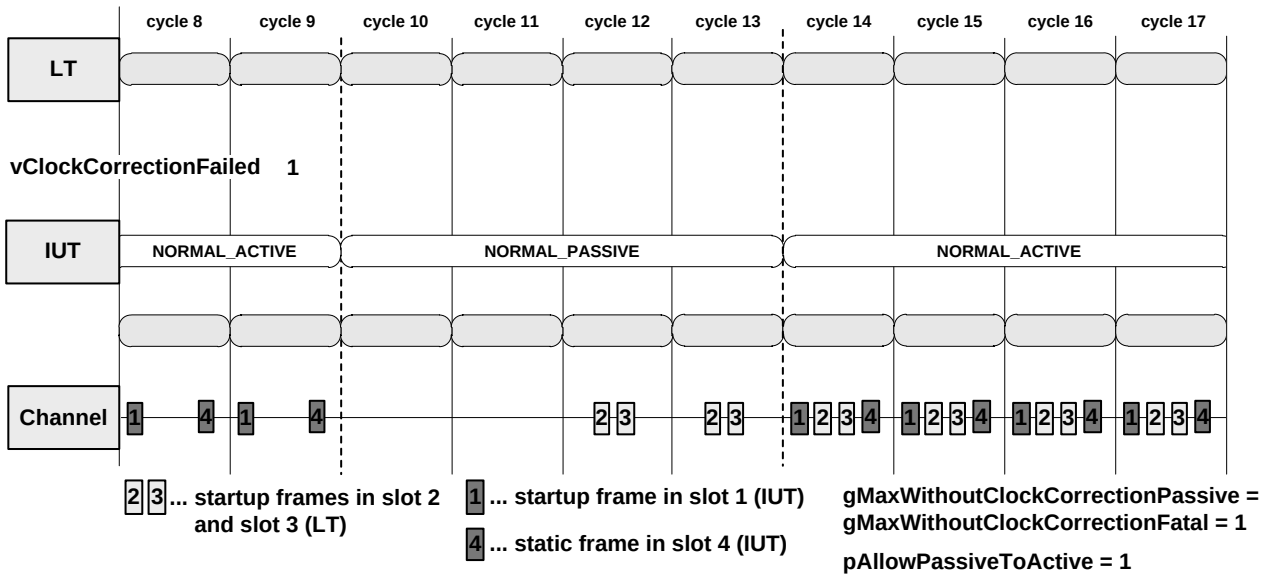


Figure 57 — Resynchronization after resumption of sync frame simulation (transition to POC:normal active allowed) – pAllowPassiveToActive = 1

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly. The IUT returns to the *POC:normal active* state *pAllowPassiveToActive* even / odd cycle pairs after the LT resumed frame simulation.

7.4.2.1.2.4 Deviation outside bounds (halt allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as integrating node when the LT resumes frame simulation exhibiting a deviation exceeding clock correction bounds while the IUT is in *POC:normal passive* state. The transition to *POC:halt* state is allowed. The IUT shall enter the *POC:halt* state after the clock correction result exceeds clock correction bounds. Verify that an interrupt is indicated after the state transition to the *POC:halt* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 241.

Table 241 — Modification to basic configurations for POC:normal passive – deviation outside bounds (halt allowed)

Parameter	Modification
gMaxWithoutClockCorrectionFatal	3
gMaxWithoutClockCorrectionPassive	1
pClusterDriftDamping [µT]	0
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pOffsetCorrectionOut [µT]	80

Δt = 100 µT.

— **Preamble (setup state)**

Preamble III.

The preamble is supplemented by:

1) At the start of cycle 4 the UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

2) In cycle 9, the LT simulates an additional frame in slot 3.

3) Beginning with cycle 10, the LT stops frame simulation.

4) In cycle 12, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE*.

— **Test execution**

1) In cycle 12, it is verified (LT) that the IUT stops frame transmission.

2) In cycle 14, the LT resumes frame simulation.

3) The LT delays the start of cycle 16 by Δt, i.e. the following cycle 17 starts delayed, as well.

4) In cycle 16, 500 µT after cycle start and before cycle end (in terms of LT this is 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before cycle end), the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

5) In cycle 17 of the LT, 500 µT + $|\Delta t|$ after LT cycle start and $|\Delta t|$ before LT NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $\Delta t + \xi_{IUT}$ µT.

6) In cycle 17 of the LT, 500 µT + |Δt| after LT cycle start and |Δt| before LT NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the error mode is *vPOC!ErrorMode* = *COMM_HALT*.
It is also verified (UT) that the IUT has set the proper interrupt status indication within [100; 200] µT after the interrupt source event occurs and that the IUT has raised the proper interrupt request within [0; 100] µT + ξ_{IUT} µT after the state transition to the *POC:halt* state.

Figure 58 depicts the deviation outside bounds (halt allowed) in *POC:normal passive*.

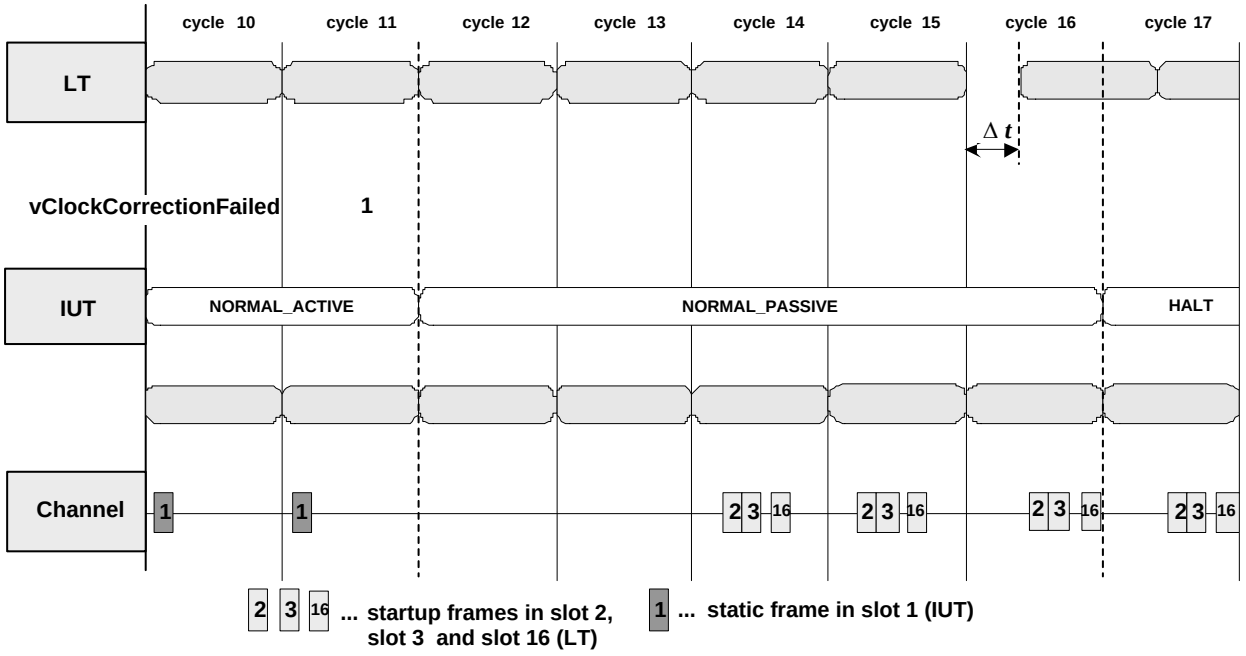


Figure 58 — Deviation outside bounds (halt allowed) in POC:normal passive

— Postamble

None. The IUT is already in *POC:halt* state.

— Pass criteria

The IUT enters *POC:halt* state because the clock correction result exceeds its configured bounds while the IUT is in *POC:normal passive* state. The IUT indicates an interrupt after it entered the *POC:halt* state.

7.4.2.1.2.5 Resynchronization after resumption of startup / sync frame simulation (integrating node, transition to POC:normal active allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as integrating node when the LT resumes frame simulation. The LT resumes simulation of either one startup frame and one sync frame or of two sync frames. The transition to *POC:normal active* state is allowed. The IUT shall not return to the *POC:normal active* state, because the LT does not resume simulation of at least two startup frames.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 242.

Table 242 — Modification to basic configurations for POC:normal passive – resynchronization after resumption of startup / sync frame simulation (integrating node, transition to POC:normal active allowed)

Parameter	Modification
pAllowHaltDueToClock	false
pAllowPassiveToActive	1
gMaxWithoutClockCorrectionPassive	1
gMaxWithoutClockCorrectionFatal	1
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pKeySlotOnlyEnabled	true

— **Preamble (setup state)**

Preamble III.

The IUT is configured to transmit static frames in slot 1 on the available channel(s).

The preamble is supplemented by the following statement: "In cycle 10, the LT stops frame simulation."

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE*. It is also verified (UT) that the counter *vAllowPassiveToActive* = 0.

2) In cycle 12, it is verified (LT) that the IUT stops frame transmission.

3) In cycle 14, the LT resumes frame transmission. The LT simulates

   (a) sync frames (no startup frames) in slot 2 and slot 16,

   (b) startup frames in slot 2 and sync frames in slot 16 and

   (c) startup frames in slot 2 and in slot 16.

4) In cycle 16, 500 µT after cycle start and before NIT, it is verified (UT) that the counter *vAllowPassiveToActive* = 0 (*pAllowPassiveToActive* – 1) and that

   (a, b) the IUT stays in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and

   (c) he IUT has returned to the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and has set the *vPOC!ErrorMode* to *ACTIVE*.

5) In cycle 16, it is verified (LT) that

   (a, b) he IUT does not resume frame transmission and

(c) he IUT resumes frame transmission in slot 1.

6) In cycle 16, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the IUT's sync frame ID is not contained in the received or transmitted sync frames list(s) for the even cycle on the respective channel(s).

Figure 59 depicts the resynchronization after resumption of startup / sync frame simulation (integrating node, transition to POC.

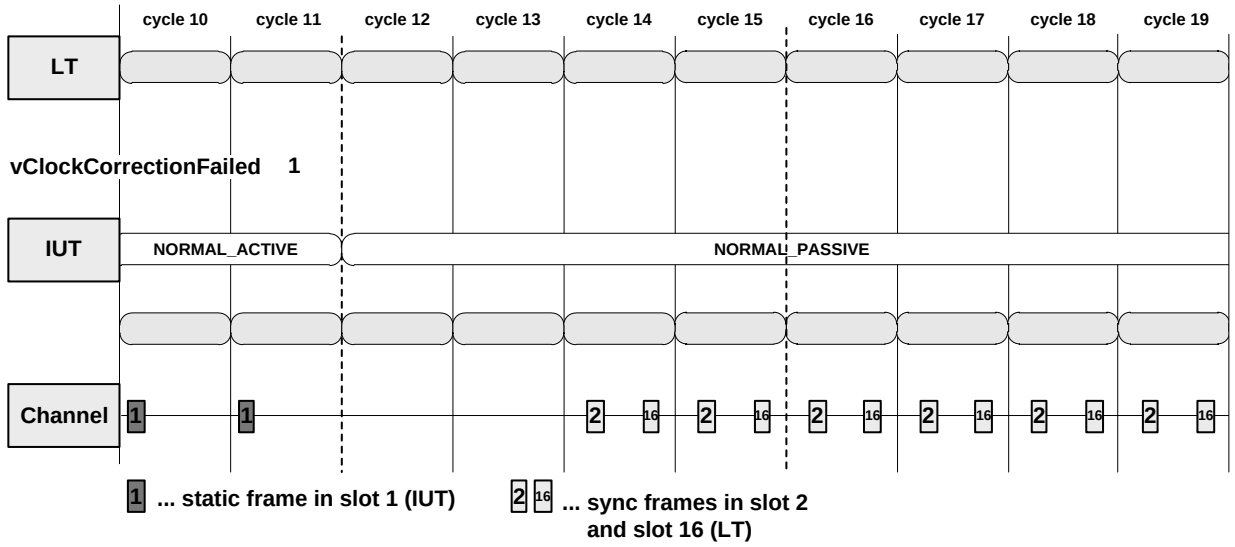


Figure 59 — Resynchronization after resumption of startup / sync frame simulation (integrating node, transition to POC:normal active allowed) – variant (a)

Figure 60 depicts the resynchronization after resumption of startup / sync frame simulation (integrating node, transition to POC.

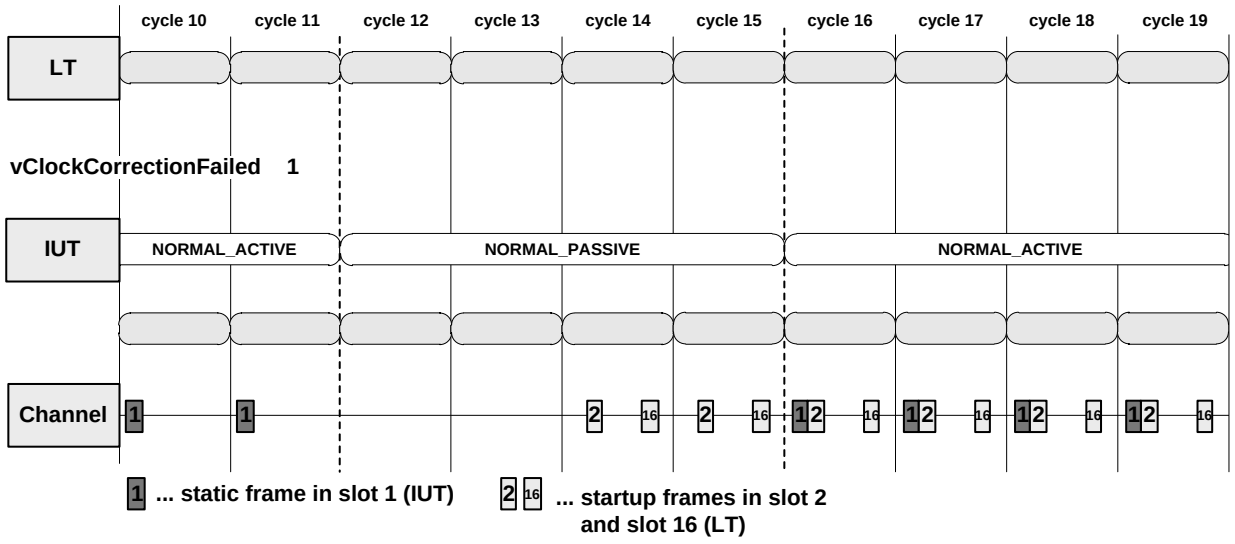


Figure 60 — Resynchronization after resumption of startup / sync frame simulation (integrating node, transition to POC:normal active allowed) – variant (c)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly. The IUT stays in the *POC:normal passive* state *pAllowPassiveToActive* even / odd cycle pairs after the LT resumed frame simulation of either one startup frame or no startup frame at all. The IUT returns to *POC:normal active* state *pAllowPassiveToActive* even / odd cycle pairs after the LT resumed frame simulation of two startup frames.

7.4.2.1.2.6 Additional verification for resynchronization after resumption of sync frame simulation (transition to POC:normal active allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling for each cycle with the IUT as startup node when the LT resumes frame simulation. The transition to *POC:normal active* state is allowed.

— Applicability

DC.

— Configuration

All basic configurations using the modifications as listed in Table 243.

Table 243 — Modification to basic configurations for POC:normal passive – aditional verification for resynchronization after resumption of sync frame simulation (transition to POC:normal active allowed)

Parameter	Modification
pAllowPassiveToActive	2
pAllowHaltDueToClock	false
gMaxWithoutClockCorrectionPassive	1
gMaxWithoutClockCorrectionFatal	1

— Preamble (setup state)

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates an additional startup frame in slot 3. The IUT is configured to transmit an additional static frame in slot 4 on both channels that the LT is simulating startup frame(s) on.

The preamble is supplemented by:

1) In cycle 4, the LT begins to simulate startup frames in slot 2 and in slot 3.

2) In cycle 8, the LT stops frame simulation.

3) In cycle 10, 500 µT after cycle start and before NIT, it is checked (UT) that the clock correction failed counter is *vClockCorrectionFailed* = *gMaxWithoutClockCorrectionPassive*.

4) In cycle 10, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State = NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode = PASSIVE*.

— **Test execution**

$t_{AW} \in [0, 500]$ µT

1) In cycle 10, it is verified (LT) that the IUT stops frame transmission.

2) In cycle 12, the LT resumes frame transmission

   (a) of startup frames in slot 2 and slot 3.

   (b) of a startup frame in slot 2 and a sync frame in slot 3.

3) In every cycle starting with cycle 12 and ending with cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is still in the *POC:normal passive* state (*vPOC!State = NORMAL_PASSIVE*) and that the error mode is still *vPOC!ErrorMode = PASSIVE*.

4) In cycle 14, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames lists for the even cycle on both channels contains frame IDs 2 and 3.

5) In cycle 15, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the received or transmitted sync frames lists for the odd cycle on both channels contains frame IDs 2 and 3.

6) In cycle 16, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has set the counter *vAllowPassiveToActive* to *pAllowPassiveToActive* – 1, that the IUT has returned to the *POC:normal active* state (*vPOC!State = NORMAL_ACTIVE*) and that the IUT has set the *vPOC!ErrorMode* to *ACTIVE*.

7) In cycle 16, it is verified (LT) that the IUT resumes frame transmission in slot 1 and slot 4.

Figure 61 depicts the additional verification for resynchronization after resumption of sync frame simulation.

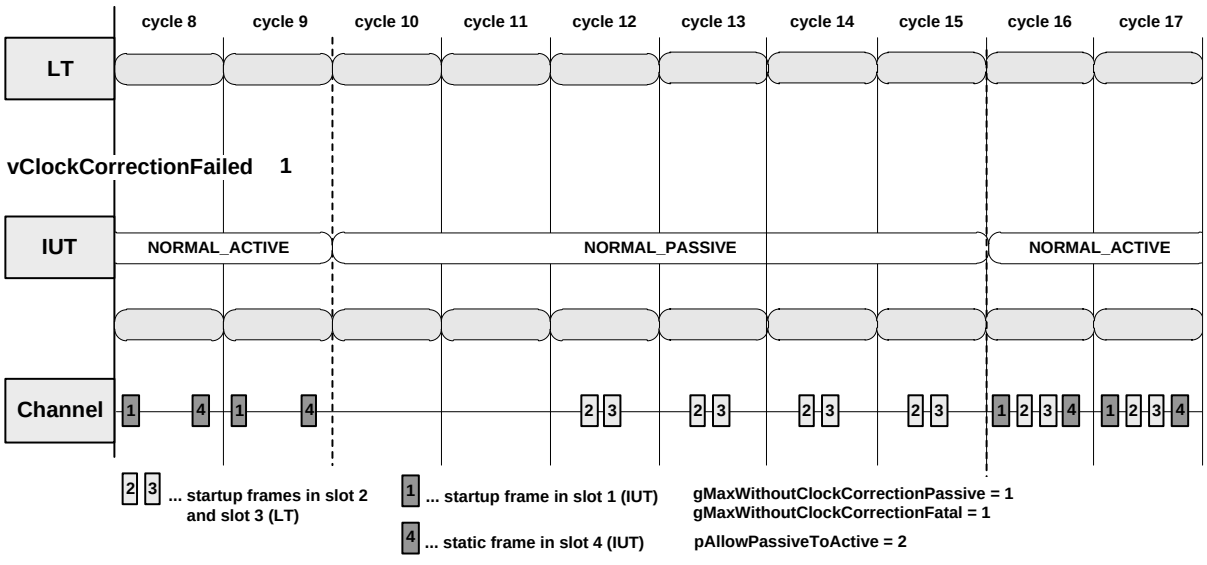


Figure 61 — Additional verification for resynchronization after resumption of sync frame simulation

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the missing traffic correctly. The IUT returns to the POC:normal active state *pAllowPassiveToActive* even / odd cycle pairs after the LT resumed frame simulation.

7.4.2.2 Rate correction

7.4.2.2.1 POC:normal active

7.4.2.2.1.1 No deviation (startup node)

— Test purpose

Verify correct behaviour of rate calculation with the IUT as startup node when there is no deviation. The IUT shall not perform any rate correction.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 244.

Table 244 — Modification to basic configurations for POC:normal active – no deviation (startup node)

Parameter	Modification
pClusterDriftDamping [µT]	0

— Preamble (setup state)

Preamble II.

The test has to be performed in 14 instances. For instance 1 the LT simulates no additional sync frame, for instance 2 one additional sync frame, …, and for instance 14 thirteen additional sync frames in successive slots starting with slot 3.

The preamble is followed by the following statement:"In cycle 6 and 7, the LT continues to simulate the startup frame and begins to simulate additional sync frames."

— Test execution

1) In cycle 8, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and that *vClockCorrectionFailed* = 0.

2) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 8 and slot 1 / cycle 9 is *pMicroPerCycle* + ξ + ξ_{IUT} µT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT does not perform any rate correction. The calculated rate correction is independent of the number of sync frames.

7.4.2.2.1.2 No deviation (integrating node)

— **Test purpose**

Verify correct behaviour of rate calculation with the IUT as integrating node when there is no deviation or with a deviation of $\Delta t + \xi_{IUT} < pClusterDriftDamping$. The IUT shall not perform any rate correction.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 245.

Table 245 — Modification I and II to basic configurations for POC:normal active – no deviation (integrating node)

Parameter	Modification I	Modification II
pKeySlotUsedForStartup	false	false
pKeySlotUsedForSync	false	false
pClusterDriftDamping [µT]	0	10

$\Delta t = 3$ µT.

— **Preamble (setup state)**

Preamble III.

The test has to be performed in 14 instances. For instance 1 the LT simulates no additional sync frame, for instance 2 one additional sync frame, …, and for instance 14 thirteen additional sync frames in successive slots starting with slot 3.

The preamble is supplemented and followed by:

1) In cycle 8, the LT begins to simulate additional sync frames.

2) In cycle 9, the LT continues to simulate startup frames in slots 2 and 16.

— **Test execution**

1) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is

   I) *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

   II) *vInterimRateCorrection* = 0 µT.

2) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is

   I) *pMicroPerCycle* + ξ + ξ_{IUT} µT

   II) *pMicroPerCycle* + ξ µT.

3) In cycle 13 the LT pauses for a period of Δt, i.e. the LT simulates its frames shifted by Δt.

4) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is

   I) *vInterimRateCorrection* = Δt + ξ_{IUT} µT and that *vClockCorrectionFailed* = 0.

   II) vInterimRateCorrection = 0 µT and that vClockCorrectionFailed = 0.

5) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 13 and slot 1 / cycle 14 is

   I) *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT

   II) *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT does not perform any rate correction when there is no deviation or when the deviation is smaller than *pClusterDriftDamping*. The calculated rate correction is independent of the number of sync frames.

7.4.2.2.1.3 Deviation inside bounds (startup node)

— Test purpose

Verify correct behaviour of rate correction with the IUT as startup node and with deviation inside clock correction bounds. The IUT shall perform appropriate rate correction.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 246.

Table 246 — Modification to basic configurations for POC:normal active – deviation inside bounds (startup node)

Parameter	Modification
pClusterDriftDamping [µT]	0

Δt = 50 µT.

— Preamble (setup state)

Preamble II.

(a) The LT does not simulate additional startup frames and sync frames.

(b) The test has to be performed in 13 instances with the LT simulating an additional startup frame in slot 3. For instance 1 the LT simulates no additional sync frame, for instance 2 one additional sync frames, …, and for instance 13 twelve additional sync frames in successive slots starting with slot 4.

The preamble is supplemented by:

1) (b) In cycle 4, the LT begins to simulate the startup frame in slot 3.

2) (b) In cycle 8, the LT begins to simulate additional sync frames.

— **Test execution**

1) In cycle 9, the LT

   (a) starts its cycle shifted by Δt and shortened by TruncateTowardsZero(Δt / 2), i.e. all following cycles start shifted. The LT still simulates its frames with the same offset as in cycle 8 from the (shifted) cycle start (see Figure 62).

   (b) starts its cycle shifted by Δt, i.e. all following cycles start shifted. (see Figure 63).

2) Starting with cycle 10, the LT stretches its cycles by

   (a) TruncateTowardsZero(Δt / 2) and

   (b) Δt.

   The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the point in times the LT starts frame simulation.

3) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is

   (a) *vInterimOffsetCorrection* = TruncateTowardsZero(∆t / 2) + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = TruncateTowardsZero(∆t / 2) + ξ_{IUT} µT and

   (b) *vInterimOffsetCorrection* = ∆t + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = ∆t + ξ_{IUT} µT.

4) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

5) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is

   (a) *pMicroPerCycle* + TruncateTowardsZero(Δt / 2) + ξ + ξ_{IUT} µT and

   (b) *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT.

6) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is

   (a) *vInterimRateCorrection* = TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT and

   (b) *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

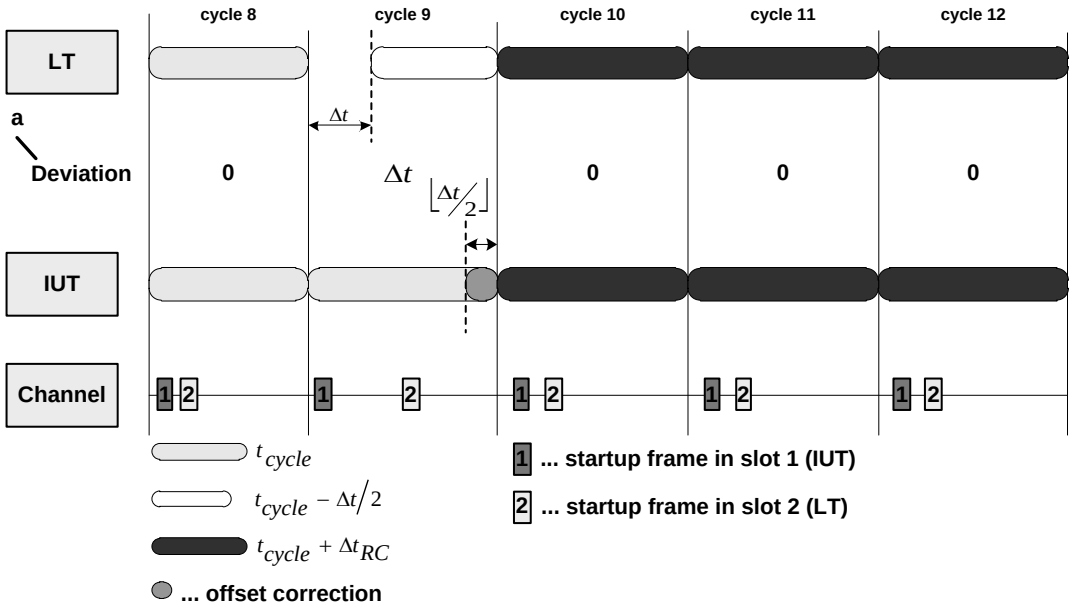
7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is

(a) *pMicroPerCycle* + TruncateTowardsZero(Δt / 2) + ξ + ξ_{IUT} µT *and*

(b) *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT.

8) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

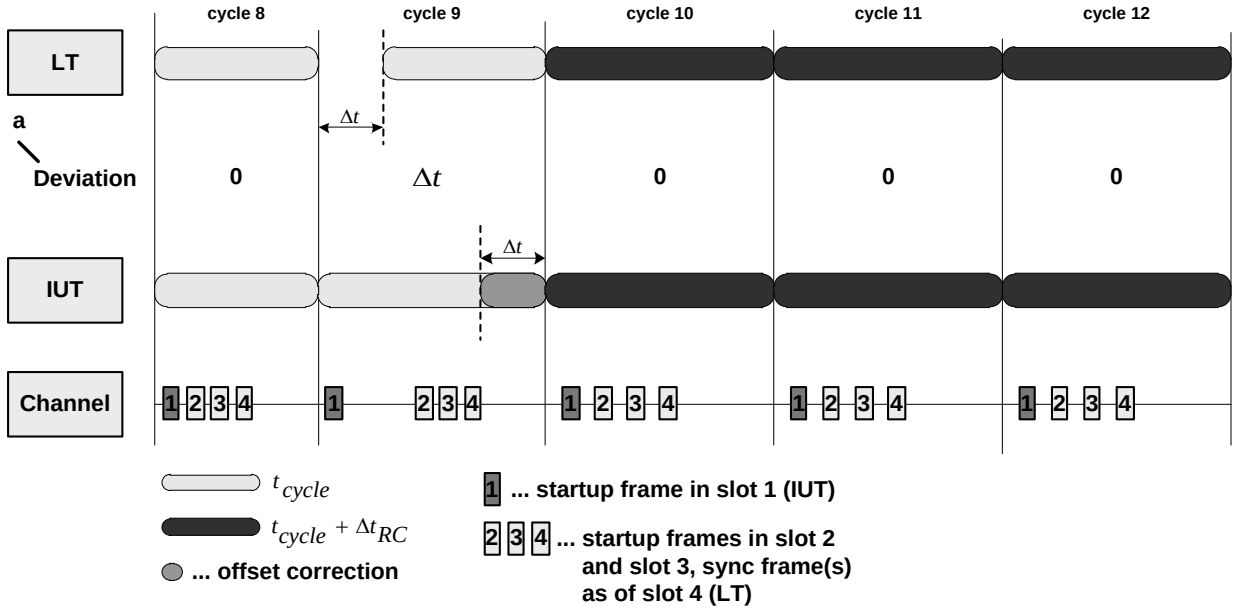
Figure 62 depicts the deviation inside bounds (startup node) – startup frame in slot 2.



a Expected deviation measured by the IUT.

Figure 62 — Deviation inside bounds (startup node) – startup frame in slot 2

Figure 63 depicts the deviation inside bounds (startup node) – startup frames in slot 2 and slot 3 and sync frame in slot 4.



a Expected deviation measured by the IUT.

Figure 63 — Deviation inside bounds (startup node) – startup frames in slot 2 and slot 3 and sync frame in slot 4

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate rate calculation and rate correction.

7.4.2.2.1.4 Deviation Inside bounds (Integrating node)

— **Test purpose**

Verify correct behaviour of rate correction with the IUT as integrating node and with deviation inside clock correction bounds. The IUT shall perform appropriate rate correction.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 247.

Table 247 — Modification to basic configurations for POC:normal active – deviation Inside bounds (Integrating node)

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pClusterDriftDamping [µT]	0

Δt = 50 µT.

— **Preamble (setup state)**

Preamble III.

The test has to be performed in 14 instances. For instance 1 the LT simulates no additional sync frame, for instance 2 one additional sync frame, …, and for instance 14 thirteen additional sync frames in successive slots starting with slot 3.

The preamble is supplemented by the following statement: "In cycle 8, the LT begins to simulate additional sync frames."

— **Test execution**

1) In cycle 9, the LT starts its cycle shifted by Δt, i.e. all following cycles start shifted (see Figure 64).

2) In cycle 9 within [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick*] µT (in terms of the LT this corresponds to [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 100 – Δt; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – Δt] µT), it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT.

3) Starting with cycle 10, the LT stretches its cycles by Δt. The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the point in times the LT starts frame simulation.

4) In cycle 10 within [500; 600] µT after cycle start and in cycle 10 within [(*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick* – 200; (*gMacroPerCycle* – *gdNIT*) * *gdMacrotick* / *pdMicrotick*] µT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

5) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

6) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT.

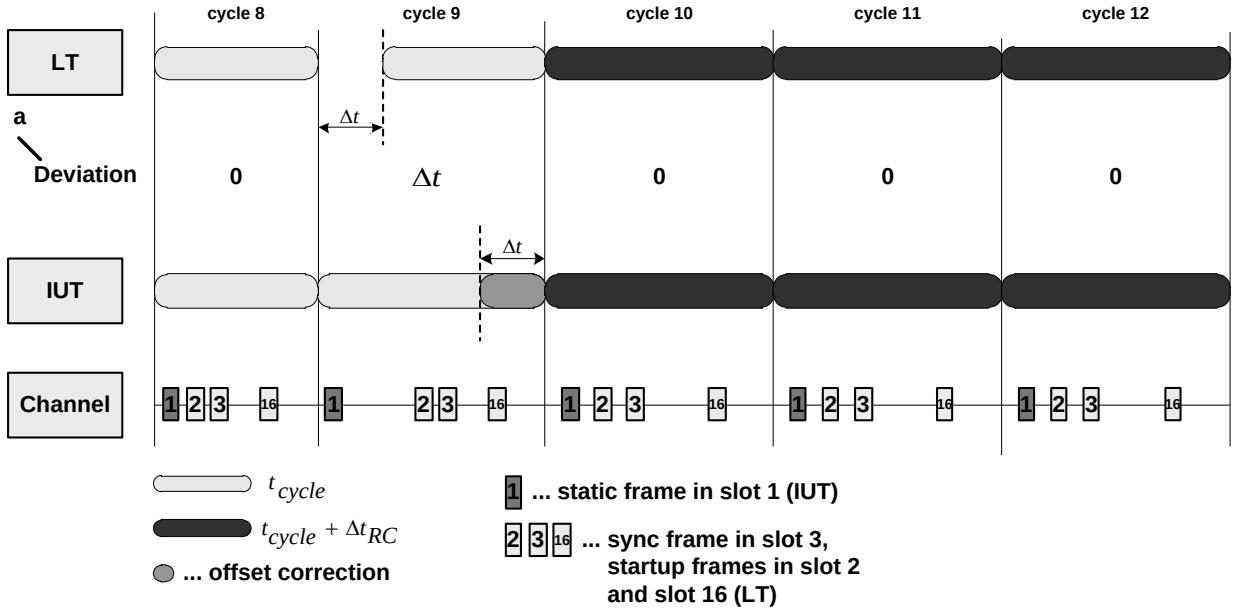
7) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

8) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT.

9) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

10) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State = NORMAL_ACTIVE*).

Figure 64 depicts the deviation inside bounds (integrating node).



a Expected deviation measured by the IUT.

Figure 64 — Deviation inside bounds (integrating node)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT stays synchronized and performs appropriate rate calculation and rate correction.

7.4.2.2.1.5 Cluster drift damping

— Test purpose

Verify correct behaviour of the cluster drift damping with the IUT as integrating node and with deviation inside clock correction bounds.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 248, Table 249 and Table 250.

Table 248 — Modification to basic configurations 1a and 1b for POC:normal active – cluster drift damping

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	4		5	4		
gdMinislot [MT]	9		12	9		
gdMinislotActionPointOffset [MT]	4		5	4		
gdStaticSlot [MT]	35		37	35		
gdSymbolWindowActionPointOffset [MT]	4		5	4		
gNumberOfMinislots	219			219		
gNumberOfStaticSlots	85		62	85		
gdNIT [MT]	14		38	14		
pdAcceptedStartupRange [µT]	281		408	403		538
pClusterDriftDamping [µT]	0	1	5	0	1	5
pMacroInitialOffset[A,B] [MT]	6		7	6		
pOffsetCorrectionOut [µT]	153		220	226		300
pOffsetCorrectionStart [MT]	4 990		4 988	4 990		

Table 249 — Modification to basic configurations 2a and 2b for POC:normal active – cluster drift damping

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdActionPointOffset [MT]	3			3		5
gdMinislot [MT]	7			7		11
gdMinislotActionPointOffset [MT]	3			3		5
gdStaticSlot [MT]	33			33		36
gdSymbolWindowActionPointOffset [MT]	3			3		5
gNumberOfMinislots	140			140		
gNumberOfStaticSlots	45			45		25
gdNIT [MT]	12			12		37
pdAcceptedStartupRange [µT]	283		418	221		348
pClusterDriftDamping [µT]	0	1	5	0	1	5
pMacroInitialOffset[A,B] [MT]	4			4		6
pOffsetCorrectionOut [µT]	154		230	117		190
pOffsetCorrectionStart [MT]	2 491			2 491		2 490

Table 250 — Modification to basic configuration 3 for POC:normal active – cluster drift damping

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdActionPointOffset [MT]	3		5
gdMinislot [MT]	7		11
gdMinislotActionPointOffset [MT]	3		5
gdStaticSlot [MT]	58		62
gdSymbolWindowActionPointOffset [MT]	3		5
gNumberOfStaticSlots	25		16
gdNumberOfMinislots	145		133
gdNIT [MT]	11		21
pdAcceptedStartupRange [µT]	224		359
pClusterDriftDamping [µT]	0	1	5
pMacroInitialOffset[A,B] [MT]	5		7
pOffsetCorrectionOut [µT]	118		190
pOffsetCorrectionStart [MT]	2 492		2 490

Δt = (a) 50 and (b) -50 µT

— **Preamble (setup state)**

Preamble III.

The IUT is an integrating node and transmits its frame (no sync frame) in slot 1. The LT simulates startup frames in slot 2 and 16 and an additional sync frame in slot 4.

The preamble is supplemented by the following statement: "In cycle 8, the LT begins to simulate the sync frame in slot 4."

— **Test execution**

1) In cycle 9, the LT

   (a) starts its cycle delayed by Δt, i.e. all following cycles start shifted (see Figure 65) and

   (b) starts its cycle Δt earlier, i.e. all following cycles start shifted (see Figure 66).

2) Starting with cycle 10, the LT

   (a) stretches its cycles by Δt and

   (b) shortens its cycles by Δt.

The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the point in times the LT starts frame simulation.

3) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $\Delta t + \xi_{IUT}$ µT and it is verified (UT) that the rate correction value is

   (a) *vInterimRateCorrection* = Δt – *pClusterDriftDamping* + ξ_{IUT} µT and

   (b) *vInterimRateCorrection* = Δt + *pClusterDriftDamping* + ξ_{IUT} µT.

4) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is

   (a) *pMicroPerCycle* + Δt – *pClusterDriftDamping* + ξ + ξ_{IUT} µT and

   (b) *pMicroPerCycle* + Δt + *pClusterDriftDamping* + ξ + ξ_{IUT} µT.

5) In cycle 12, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the rate correction value is

   (a) *vInterimRateCorrection* = Δt – *pClusterDriftDamping* + ξ_{IUT} µT and

   (b) *vInterimRateCorrection* = Δt + *pClusterDriftDamping* + ξ_{IUT} µT.

6) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 12 and slot 1 / cycle 13 is

   (a) *pMicroPerCycle* + Δt – *pClusterDriftDamping* + ξ + ξ_{IUT} µT and

   pMicroPerCycle + Δt + *pClusterDriftDamping* + ξ + ξ_{IUT} µT.

7) In cycle 14, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the rate correction value is

   (a) *vInterimRateCorrection* = Δt – *pClusterDriftDamping* + ξ_{IUT} µT and

   (b) *vInterimRateCorrection* = Δt + *pClusterDriftDamping* + ξ_{IUT} µT.

8) In cycle 14, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 65 depicts the cluster drift damping; Δt > 0.

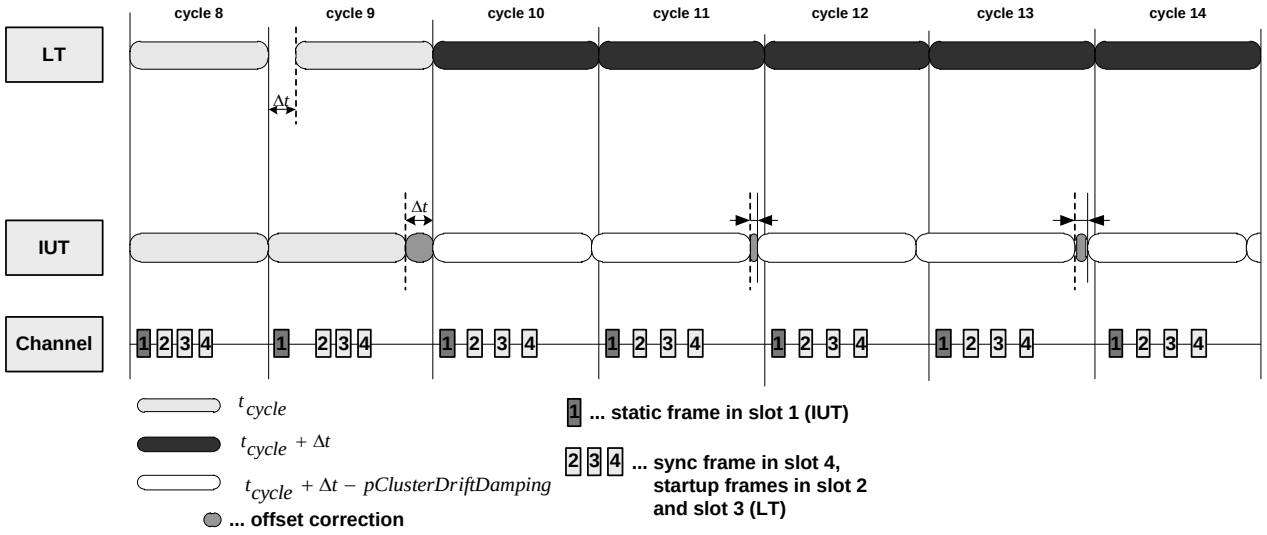


Figure 65 — Cluster drift damping; Δt > 0

Figure 66 depicts the cluster drift damping; Δt < 0.

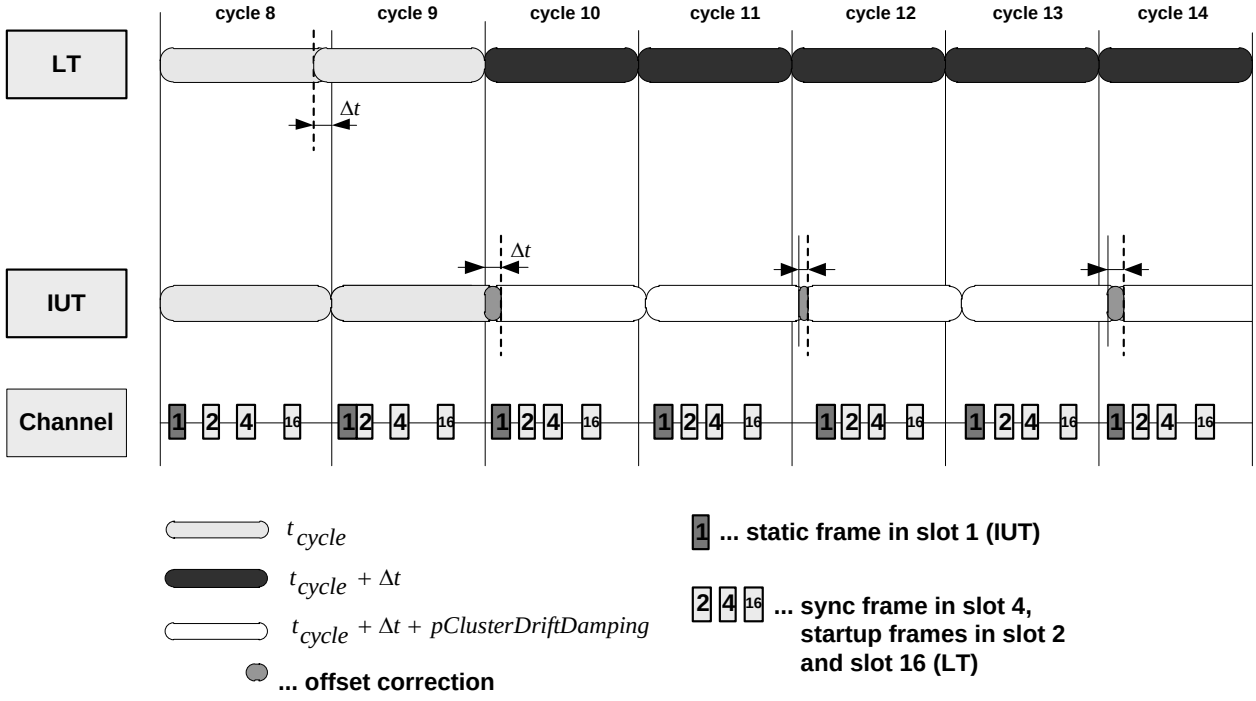


Figure 66 — Cluster drift damping; Δt < 0

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate rate calculation and rate correction.

7.4.2.2.1.6 Different deviation on channel "A&B"

— **Test purpose**

Verify correct behaviour of rate correction with the IUT as integrating node and with different deviation on channel A and channel B inside clock correction bounds.

— **Applicability**

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 251.

Table 251 — Modification to basic configurations for POC:normal active – different deviation on channel "A&B"

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pClusterDriftDamping [µT]	0

Δt = 50 µT for Basic configurations 1a, 1b and 2a.

Δt = 35 µT for Basic configurations 2b and 3.

— **Preamble (setup state)**

Preamble III.

The test has to be performed in 14 instances. For instance 1 the LT simulates no additional sync frame, for instance 2 one additional sync frame, …, and for instance 14 thirteen additional sync frames in successive slots starting with slot 3.

The preamble is supplemented by the following statement: "In cycle 8, the LT begins to simulate additional sync frames."

— **Test execution**

1) In cycle 9, the LT shifts its cycle start Δt on channel A and 2 * Δt on channel B, i.e. all following cycles are shifted. The LT shortens its cycle on channel B by ∆t (see Figure 67), i.e. the LT truncates its cycle on channel B at the cycle end.

2) Starting with cycle 10, the LT stretches its cycles by TruncateTowardsZero(3 * Δt / 2). The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the point in times the LT starts frame simulation.

3) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = TruncateTowardsZero(3 * Δt / 2) + ξ_{IUT} µT.

4) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

5) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is *pMicroPerCycle* + TruncateTowardsZero(3 * Δt / 2) + ξ + ξ_{IUT} µT.

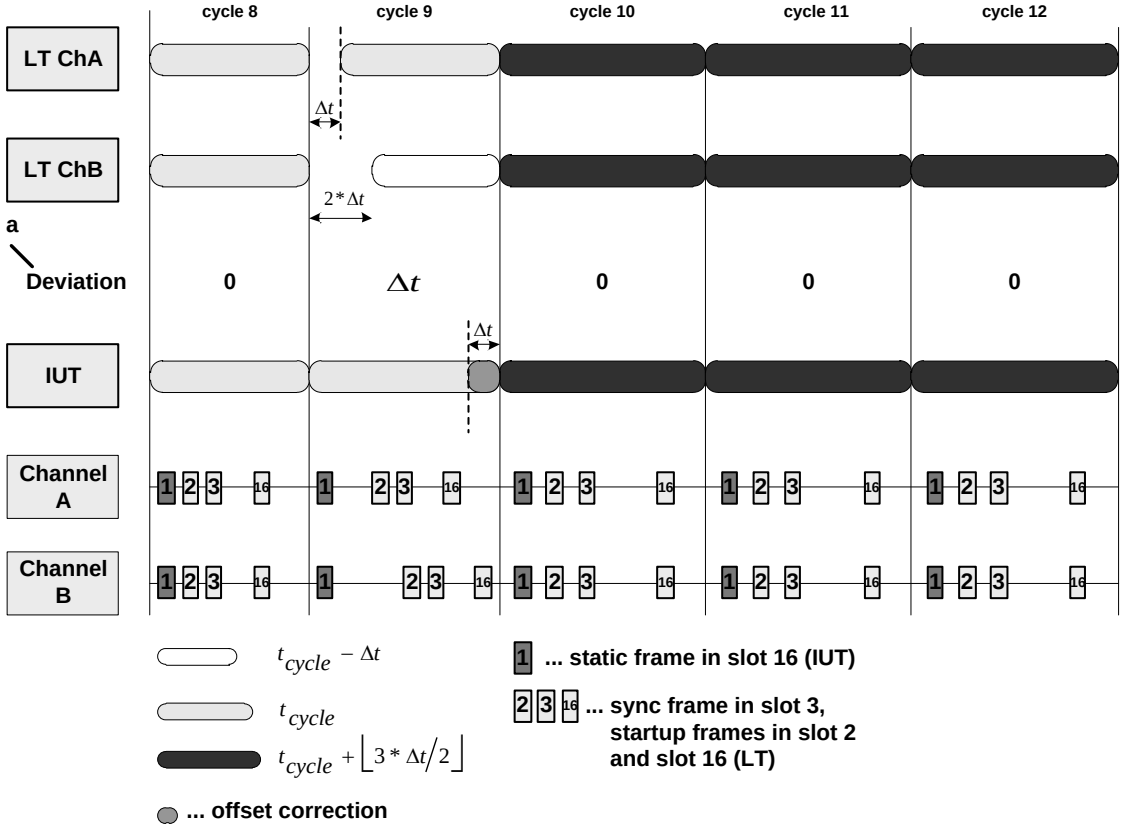
6) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = TruncateTowardsZero(3 * Δt / 2) + ξ_{IUT} µT.

7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + TruncateTowardsZero(3 * Δt / 2) + ξ + ξ_{IUT} µT.

8) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

9) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 67 depicts the rate correction - Different deviation on channel A&B.



a Expected deviation measured by the IUT.

Figure 67 — Rate correction - Different deviation on channel A&B

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate rate calculation and rate correction. The applied rate correction is averaged over the deviations measured on channel A and channel B.

7.4.2.2.1.7 Different deviation inside bounds

— **Test purpose**

Verify correct behaviour of rate correction with the IUT as integrating node and with different deviation.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 252.

Table 252 — Modification to basic configurations for POC:normal active – different deviation inside bounds

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
pKeySlotUsedForStartup	false				
pKeySlotUsedForSync	false				
pClusterDriftDamping [µT]	0				
pOffsetCorrectionOut [µT]	153	250	154	117	118

The test has to be performed repeatedly with
Δt = (a) *-pRateCorrectionOut* + 7, (b) -35, (c) -5, (d) 5, (e) 35 and (f) *pRateCorrectionOut* - 7 µT.

— **Preamble (setup state)**

Preamble III.

The IUT is an integrating node and transmits its frame (no sync frame) in slot 1. The LT simulates one additional sync frame in slot 4.

The preamble is supplemented by the following statement: "In cycle 8, the LT begins to simulate the sync frame in slot 4."

— **Test execution**

1) At the beginning of cycle 9, the LT starts its cycle earlier (Δt < 0) or later (Δt > 0), i.e. all following cycles are shifted.

2) Starting with cycle 10, the LT stretches its cycles by Δt. The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the point in times the LT starts frame simulation.

3) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

4) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

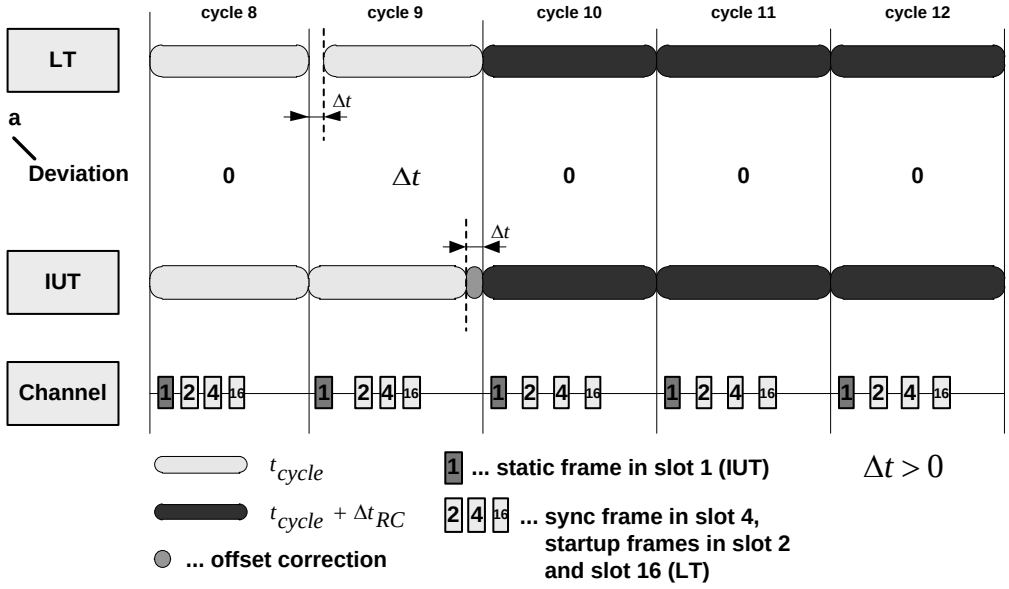
5) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT.

6) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

7) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + Δt + ξ + ξ_{IUT} µT.

8) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

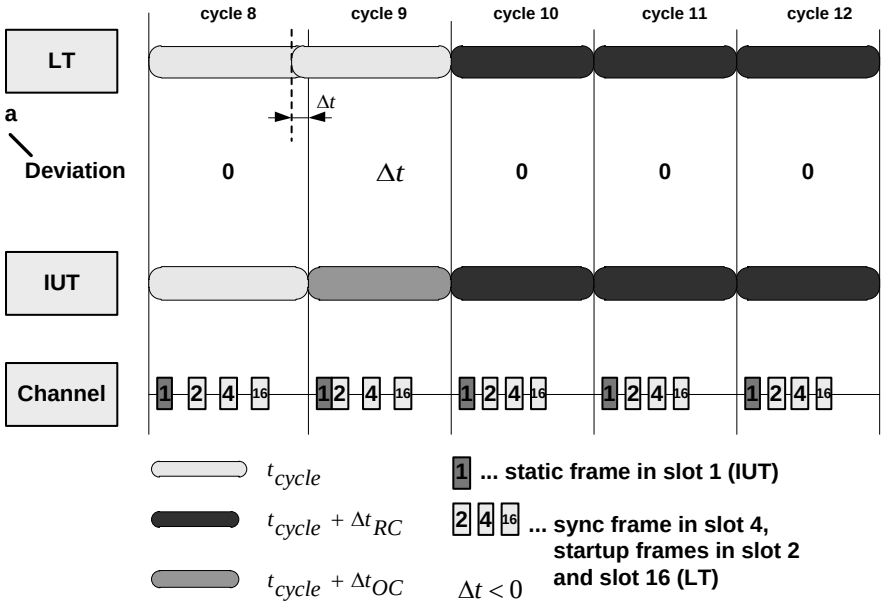
Figure 68 depicts the different deviation inside bounds; Δt > 0.



a Expected deviation measured by the IUT.

Figure 68 — Different deviation inside bounds; Δt > 0

Figure 69 depicts the different deviation inside bounds; Δt < 0.



a Expected deviation measured by the IUT.

Figure 69 — Different deviation inside bounds; Δt < 0

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT stays synchronized and performs appropriate rate calculation and rate correction.

7.4.2.2.1.8 Deviation outside bounds (transition to POC:halt allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node and with deviation outside rate correction bounds. The transition to *POC:halt* state is allowed. Verify that an interrupt is indicated after the state transition to the *POC:halt* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 253.

Table 253 — Modification to basic configurations for POC:normal active – deviation outside bounds (transition to POC:halt allowed)

Parameter	Modification
pRateCorrectionOut [µT]	48[a]
pClusterDriftDamping [µT]	0
[a] pRateCorrectionOut = 48 µT which is an intentional protocol constraints violation in order to ease test execution.	

The test shall be performed with a Δt of -60 and 60 µT.

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates an additional startup frame in slot 3.

The preamble is supplemented by:

1) The interrupt requests are not globally disabled. Solely, the interrupt request for the state transition is individually enabled.

2) In cycle 4, the LT begins to simulate the additional startup frame in slot 3. At the start of cycle 4 the UT enables the interrupt request for state transitions of the POC.

3) Starting with cycle 7, the LT continues to simulate its startup frame in slot 2.

— **Test execution**

1) At the beginning of cycle 9, the LT starts its cycle earlier (Δt < 0) or later (Δt > 0). All following cycles are shifted by Δt, too.

2) In cycle 9, 500 µT after cycle start and before NIT (in terms of LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), the UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

3) In cycle 10 of the LT, 500 µT + |Δt| after LT cycle start and |Δt| before LT NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

4) In cycle 10 of the LT, 500 µT + |Δt| after LT cycle start and |Δt| before LT NIT, it is verified (UT) that the IUT has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that the IUT has set the error mode *vPOC!ErrorMode* to *COMM_HALT*.
It is also verified (UT) that the IUT has set the proper interrupt status indication within [100; 200] µT after the interrupt source event occurs and that the IUT has raised the proper interrupt request within [0; 100] µT + ξ_{IUT} µT after the state transition to the *POC:halt* state.

5) In cycle 10 of the LT, it is verified (LT) that the IUT stops transmission.

Figure 70 depicts the deviation outside bounds (transition to *POC:halt allowed*).

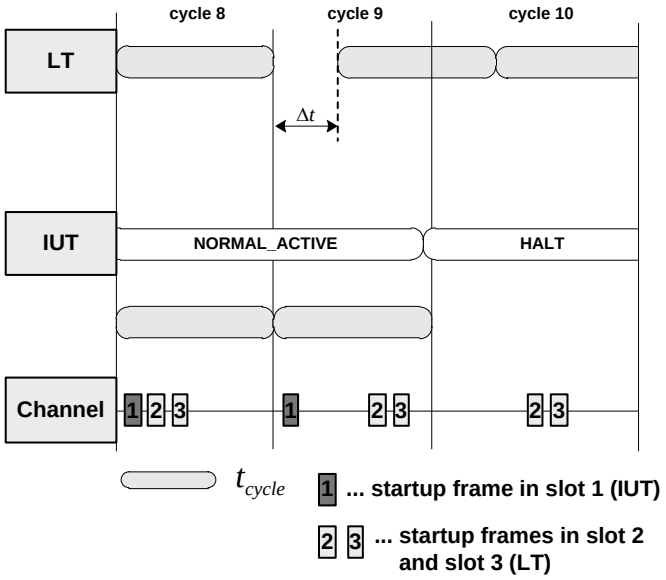


Figure 70 — Deviation outside bounds (transition to POC:halt allowed)

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT recognizes and handles the rate correction violation correctly. The IUT immediately enters the *POC:halt* state after the rate calculation result exceeded *pRateCorrectionOut*. The IUT indicates an interrupt after it entered the *POC:halt* state.

7.4.2.2.1.9 Deviation outside bounds (transition to POC:halt not allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as startup node and with deviation outside rate correction bounds. The transition to *POC:halt* state is not allowed.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 254.

Table 254 — Modification I and II to basic configurations for POC:normal active – deviation outside bounds (transition to POC:halt not allowed)

Parameter	Modification	Modification
	I	II
pRateCorrectionOut [µT]	48[a]	
pClusterDriftDamping [µT]	0	
pAllowHaltDueToClock	false	
Δt [µT]	-60	60
[a] pRateCorrectionOut = 48 µT which is an intentional protocol constraints violation in order to ease test execution.		

The test shall be performed with a Δt of -60 and 60 µT.

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstarters. The LT simulates an additional startup frame in slot 3.

The preamble is supplemented by the following statement: "In cycle 4, the LT begins to simulate the additional startup frame in slot 3."

— **Test execution**

1) At the beginning of cycle 9, the LT starts its cycle earlier (Δt < 0) or later (Δt > 0). All following cycles are shifted by Δt, too.

2) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT after cycle start and |Δt| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the IUT has set the error mode *vPOC!ErrorMode* to *PASSIVE*.

3) In cycle 10, it is verified (LT) that the IUT has stopped transmission.

Figure 71 depicts the deviation outside bounds (transition to *POC:halt not allowed*).

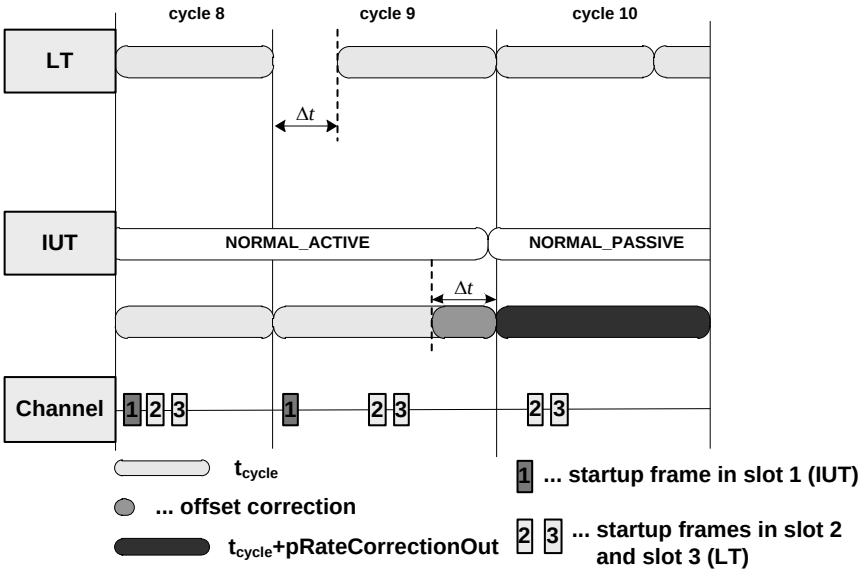


Figure 71 — Deviation outside bounds (transition to POC:halt not allowed)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT recognizes and handles the rate correction violation correctly. The IUT immediately enters the *POC:normal passive* state after the rate calculation result exceeded *pRateCorrectionOut*.

7.4.2.2.1.10 Deviation inside bounds (integrating node, single startup frame)

— **Test purpose**

Verify correct behaviour of rate correction with the IUT as integrating node and with deviation inside clock correction bounds. The IUT shall perform appropriate rate correction.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 255.

Table 255 — Modification to basic configurations for POC:normal active – deviation inside bounds (integrating node, single startup frame)

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pClusterDriftDamping [µT]	0

Δt = 50 µT.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT shifts its cycle start by Δt, i.e. all following cycles are shifted. The LT simulates a startup frame in slot 2, solely (see Figure 72). The LT does not simulate its startup frame in slot 16 / cycle 9.

2) Starting with cycle 10, the LT stretches its cycles by Δt. The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the point in time the LT starts frame simulation. The LT resumes to simulate startup frames in slots 2 and 16.

3) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $\Delta t + \xi_{IUT}$ µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = $\Delta t + \xi_{IUT}$ µT.

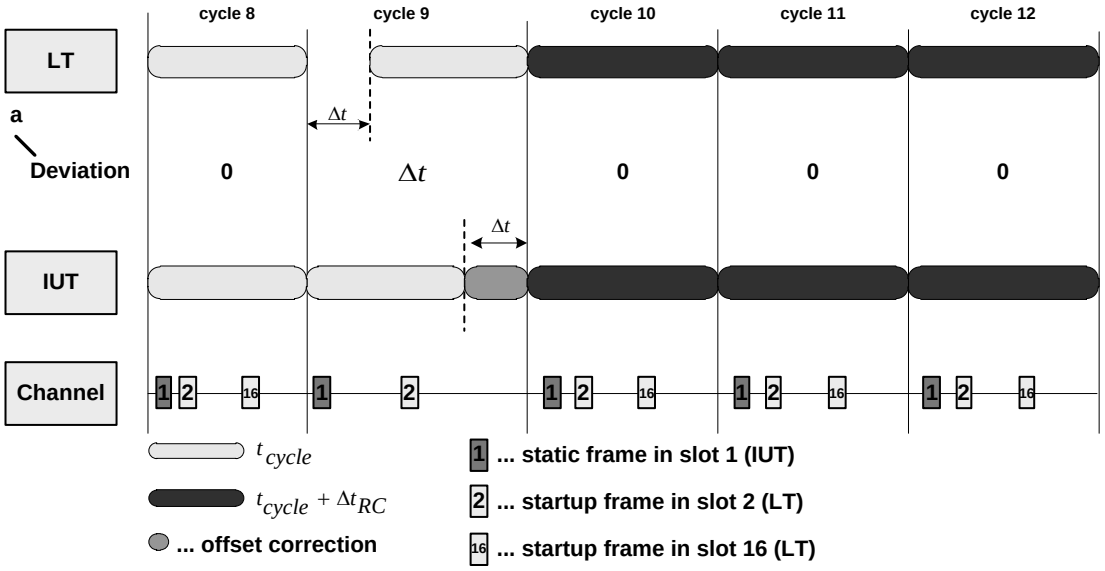
4) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 10 and slot 1 / cycle 11 is *pMicroPerCycle* + $\Delta t + \xi + \xi_{IUT}$ µT.

5) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = $0 + \xi_{IUT}$ µT and it is verified (UT) that the rate correction value is *vInterimRateCorrection* = $\Delta t + \xi_{IUT}$ µT.

6) It is verified (LT) that the interval between the IUT's frames in slot 1 / cycle 11 and slot 1 / cycle 12 is *pMicroPerCycle* + $\Delta t + \xi + \xi_{IUT}$ µT.

7) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 72 depicts the deviation inside bounds (integrating node, single startup Frame).



a Expected deviation measured by the IUT.

Figure 72 — Deviation inside bounds (integrating node, single startup Frame)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT stays synchronized and performs appropriate rate calculation and rate correction.

7.4.2.2.1.11 Missing sync frames (sync node, halt not allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as startup node when the LT's startup frame in an even cycle is missing. The transition to *POC:halt* state is not allowed. The IUT shall enter the *POC:normal passive* state due to the missing startup frame.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 256.

Table 256 — Modification to basic configurations for POC:normal active – missing sync frames (sync node, halt not allowed)

Parameter	Modification
pAllowHaltDueToClock	false
gMaxWithoutClockCorrectionPassive	1
gMaxWithoutClockCorrectionFatal	4

— **Preamble (setup state)**

Preamble II.

The LT simulates a following coldstart node. The LT simulates startup frames in slot 2.

— **Test execution**

1) In cycle 7, the LT simulates its startup frame in slot 2.

2) In cycle 8, the LT does not simulate its startup frame in slot 2.

3) In cycle 9, the LT resumes simulation of startup frames in slot 2.

4) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 1.

5) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the error mode is *vPOC!ErrorMode* = *PASSIVE*.

6) In cycle 10, it is verified (LT) that the IUT stops frame transmission and does not send anything in the following cycles.

7) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 1 and that the counter *vAllowPassiveToActive* is 0.

8) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 0 and that the counter *vAllowPassiveToActive* is 0.

9) In cycle 15, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 0 and that the counter *vAllowPassiveToActive* is 0.

10) In cycle 16, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT stays in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

Figure 73 depicts the missing sync frames (sync node, halt not allowed).

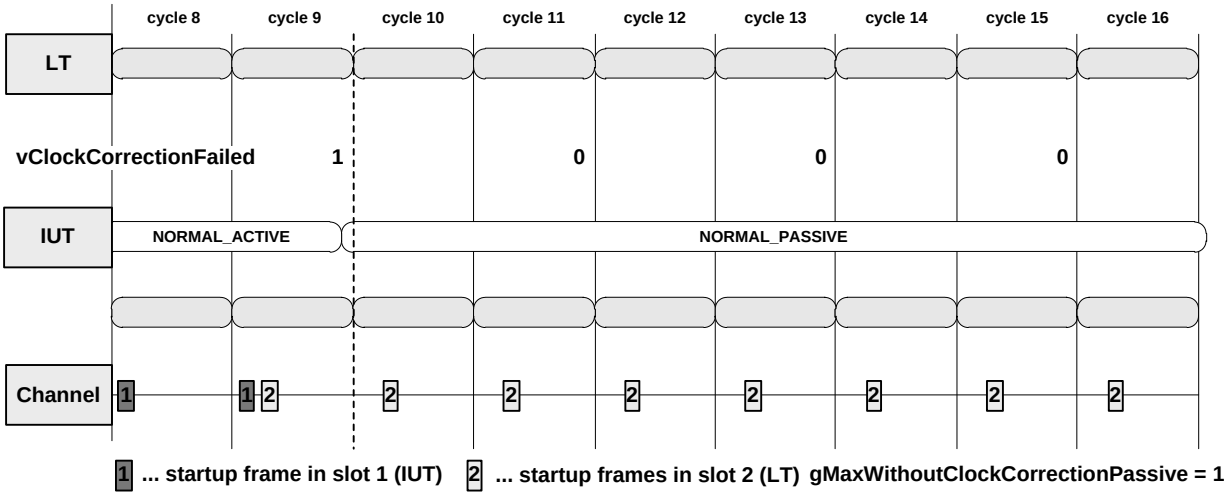


Figure 73 — Missing sync frames (sync node, halt not allowed)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT recognizes and handles the missing traffic in the even cycle correctly. The IUT enters the *POC:normal passive* state due to the missing startup frame and does not return to the *POC:normal active* state because *pAllowPassiveToActive* = 0.

7.4.2.2.1.12 gSyncFrameIDCountMax (1)

— **Test purpose**

Verify correct behaviour of rate correction with the IUT as integrating node and with different values for the parameter *gSyncFrameIDCountMax*. The IUT shall calculate its rate correction value over a number of *gSyncFrameIDCountMax* sync / startup frames.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 257.

Table 257 — Modification to basic configurations for POC:normal active – gSyncFrameIDCountMax (1)

Parameter	Modification to Basic Configurations														
	1a & 1b					2a & 2b					3				
	I	II	III	IV	V	I	II	III	IV	V	I	II	III	IV	V
gSyncFrameIDCountMax	2	3	7	8	15	2	3	7	8	15	2	3	7	8	15
pKeySlotUsedForSync	false														
pKeySlotUsedForStartup	false														
pClusterDriftDamping [µT]	0														
pOffsetCorrectionOut [µT]	160														
pRateCorrectionOut [µT]	160[a]														
pOffsetCorrectionStart [MT]	4 990					2 491									
[a] *pRateCorrectionOut* = 160 µT is an intentional protocol constraints violation.															

Δt shall be set different for each slot according to Table 258.

Table 258 — Shift of the frames by the LT in cycle 9 for gSyncFrameIDCountMax (1)

Slot #	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Δt [µT]	8	112	16	40	48	56	64	96	72	80	88	96	104	64	112

— **Preamble (setup state)**

Preamble III.

The LT simulates 13 sync frames in adjacent slots starting with slot 3.

In cycle 8, the LT begins to simulate additional frames.

— **Test execution**

1) In cycle 9, the LT deviates its frames according to Table 258.

2) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + max(Δt(slot i)) after cycle start and before min(Δt(slot i)) after begin of NIT), it is verified (UT) that the offset correction value and the rate correction value are

   (I) *vInterimOffsetCorrection* = 60 + ξ_{IUT} µT and *vInterimRateCorrection* = 60 + ξ_{IUT} µT,

   (II) *vInterimOffsetCorrection* = 16 + ξ_{IUT} µT and *vInterimRateCorrection* = 16 + ξ_{IUT} µT,

   (III) *vInterimOffsetCorrection* = 40 + ξ_{IUT} µT and *vInterimRateCorrection* = 40 + ξ_{IUT} µT,

   (IV) *vInterimOffsetCorrection* = 52 + ξ_{IUT} µT and *vInterimRateCorrection* = 52 + ξ_{IUT} µT and

   (V) *vInterimOffsetCorrection* = 72 + ξ_{IUT} µT and *vInterimRateCorrection* = 72 + ξ_{IUT} µT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT uses *gSyncFrameIDCountMax* sync / startup frames for the calculation of the offset and rate correction values.

7.4.2.2.1.13 gSyncFrameIDCountMax (2)

— **Test purpose**

Verify correct behaviour of rate correction with the IUT as integrating node and with different values for the parameter *gSyncFrameIDCountMax*. The IUT shall calculate its rate correction value over a number of *gSyncFrameIDCountMax* sync / startup frames.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 259.

Table 259 — Modification to basic configurations for POC:normal active – gSyncFrameIDCountMax (2)

Parameter	Modification to Basic Configurations														
	1a & 1b					2a & 2b					3				
	I	II	III	IV	V	I	II	III	IV	V	I	II	III	IV	V
gSyncFrameIDCountMax	2	3	7	8	15	2	3	7	8	15	2	3	7	8	15
pKeySlotUsedForSync	false														
pKeySlotUsedForStartup	false														
pClusterDriftDamping [µT]	0														
pOffsetCorrectionOut [µT]	160														
pRateCorrectionOut [µT]	160[a]														
pOffsetCorrectionStart [MT]	4 990					2 491									
[a] *pRateCorrectionOut* = 160 µT is an intentional protocol constraints violation.															

Δt shall be set differently for each slot according to Table 260.

Table 260 — Shift of the frames by the LT in cycle 9 for gSyncFrameIDCountMax (2)

Slot #	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Δt [µT]	-8	-112	-16	-40	-48	-56	-64	-96	-72	-80	-88	-96	-104	-64	-112

— **Preamble (setup state)**

Preamble III.

The LT simulates 13 sync frames in adjacent slots starting with slot 3.

In cycle 8, the LT begins to simulate additional frames.

— **Test execution**

1) In cycle 9, the LT deviates its frames according to Table 260.

2) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + *vInterimOffsetCorrection* after cycle start and |*vInterimOffsetCorrection* + *vInterimRateCorrection*| before NIT with *vInterimOffsetCorrection* and *vInterimRateCorrection* as defined below), it is verified (UT) that the offset correction value and the rate correction value are

   (I) *vInterimOffsetCorrection* = –60 + ξ_{IUT} µT and *vInterimRateCorrection* = –60 + ξ_{IUT} µT,

   (II) *vInterimOffsetCorrection* = –16 + ξ_{IUT} µT and *vInterimRateCorrection* = –16 + ξ_{IUT} µT,

   (III) *vInterimOffsetCorrection* = –40 + ξ_{IUT} µT and *vInterimRateCorrection* = –40 + ξ_{IUT} µT,

   (IV) *vInterimOffsetCorrection* = –52 + ξ_{IUT} µT and *vInterimRateCorrection* = –52 + ξ_{IUT} µT and

   (V) *vInterimOffsetCorrection* = –72 + ξ_{IUT} µT and *vInterimRateCorrection* = –72 + ξ_{IUT} µT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT uses *gSyncFrameIDCountMax* sync / startup frames for the calculation of the offset and rate correction values.

7.4.2.2.2 POC:normal passive

7.4.2.2.2.1 Resynchronization (transition to POC:normal active allowed)

— **Test purpose**

Verify correct behaviour of the clock correction error handling with the IUT as startup node and with deviation outside rate correction bounds for a single cycle. The transition to *POC:normal active* state is allowed.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 261.

Table 261 — Modification to basic configurations for POC:normal passive – resynchronization (transition to POC:normal active allowed)

Parameter	Modification		
	I	II	III
pAllowPassiveToActive	1	16	31
pAllowHaltDueToClock	false		
pRateCorrectionOut [µT]	48[a]		
pClusterDriftDamping [µT]	0		
[a] *pRateCorrectionOut* = 48 µT, which is an intentional protocol constraints violation in order to ease test execution.			

Δt = 60 µT.

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates an additional startup frame in slot 3.

The preamble is supplemented by:

1) In cycle 4, the LT begins to simulate the additional startup frame in slot 3.

2) The LT delays the start of cycle 9 by Δt, i.e. all following cycles start delayed.

3) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is checked (UT) that the offset correction value is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and that the rate correction value is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.

4) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is checked (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the IUT has set the error mode *vPOC!ErrorMode* to *PASSIVE*.

— **Test execution**

1) It is verified (LT) that the IUT stops frame transmissions in cycle 10 and that the IUT does not transmit anything in the succeeding cycles.

2) In cycle 12, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the rate correction value is *vInterimRateCorrection* < *pRateCorrectionOut*. Thus, cycle 11 is the first odd cycle during which the rate correction value does not exceed *pRateCorrectionOut*.

3) If *pAllowPassiveToActive* > 1: In every even cycle starting with cycle 12 and ending with cycle (8 + 2 * *pAllowPassiveToActive*)mod 64, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the IUT has incremented the counter *vAllowPassiveToActive* (*vAllowPassiveToActive* = 1 in cycle 12, *vAllowPassiveToActive* = *pAllowPassiveToActive* – 1 in cycle (8 + 2 * *pAllowPassiveToActive*)mod 64).

4) In cycle (10 + 2 * *pAllowPassiveToActive*)mod 64, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that

the IUT has set the counter *vAllowPassiveToActive* to *pAllowPassiveToActive* – 1 and that the IUT has returned to the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

5) In cycle (10 + 2 * *pAllowPassiveToActive*)mod 64, it is verified (LT) that the IUT resumes frame transmission.

Figure 74 depicts the resynchronization (transition to *POC:normal active allowed*).

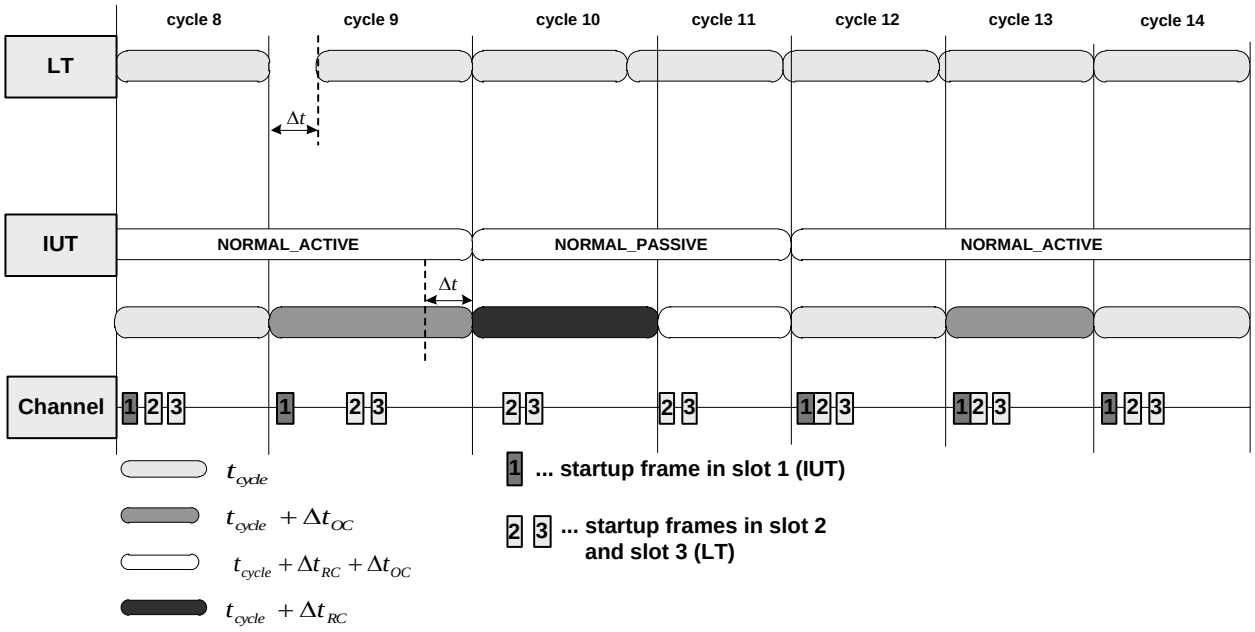


Figure 74 — Resynchronization (transition to POC:normal active allowed)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the rate correction violation correctly. The IUT returns to *POC:normal active* state after the rate correction stayed within bounds for *pAllowPassiveToActive* even / odd cycle pairs.

7.4.2.2.2.2 Resynchronization (transition to POC:normal active not allowed)

— Test purpose

Verify correct behaviour of the clock correction error handling with the IUT as startup node and with deviation outside rate correction bounds for a single cycle. The transition to *POC:normal active* state is not allowed.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 262.

Table 262 — Modification to basic configurations for POC:normal passive – resynchronization (transition to POC:normal active not allowed)

Parameter	Modification
pRateCorrectionOut [µT]	48[a]
pClusterDriftDamping [µT]	0
pAllowHaltDueToClock	false
[a] *pRateCorrectionOut* = 48 µT, which is an intentional protocol constraints violation in order to ease test execution.	

Δt = 60 µT.

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstart nodes. The LT simulates an additional startup frame in slot 3.

The preamble is supplemented by:

1) In cycle 4, the LT begins to simulate the additional startup frame in slot 3.

2) The LT delays the start of cycle 9 by Δt, i.e. all following cycles start delayed.

3) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), it is checked (UT) that the offset correction value is *vInterimOffsetCorrection* = $\Delta t + \xi_{IUT}$ µT and that the rate correction value is *vInterimRateCorrection* = $\Delta t + \xi_{IUT}$ µT.

4) In cycle 10, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), it is checked (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the IUT has set the error mode *vPOC!ErrorMode* to *PASSIVE*.

— **Test execution**

1) It is verified (LT) that the IUT stops frame transmissions in cycle 10 and that the IUT does not transmit anything in the succeeding cycles.

2) In cycle 12, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), it is verified (UT) that the rate correction value is *vInterimRateCorrection* < *pRateCorrectionOut* µT. Thus, cycle 11 is the first odd cycle during which the rate correction value does not exceed *pRateCorrectionOut*.

3) In cycle 12 and in cycle 14, 500 µT after cycle start and before NIT (in terms of the LT this approximates to 500 µT + $|\Delta t|$ after cycle start and $|\Delta t|$ before NIT), it is verified (UT) that the IUT stays in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and that the counter *vAllowPassiveToActive* = 0.

Figure 75 depicts the resynchronization (transition to *POC:normal active* not allowed).

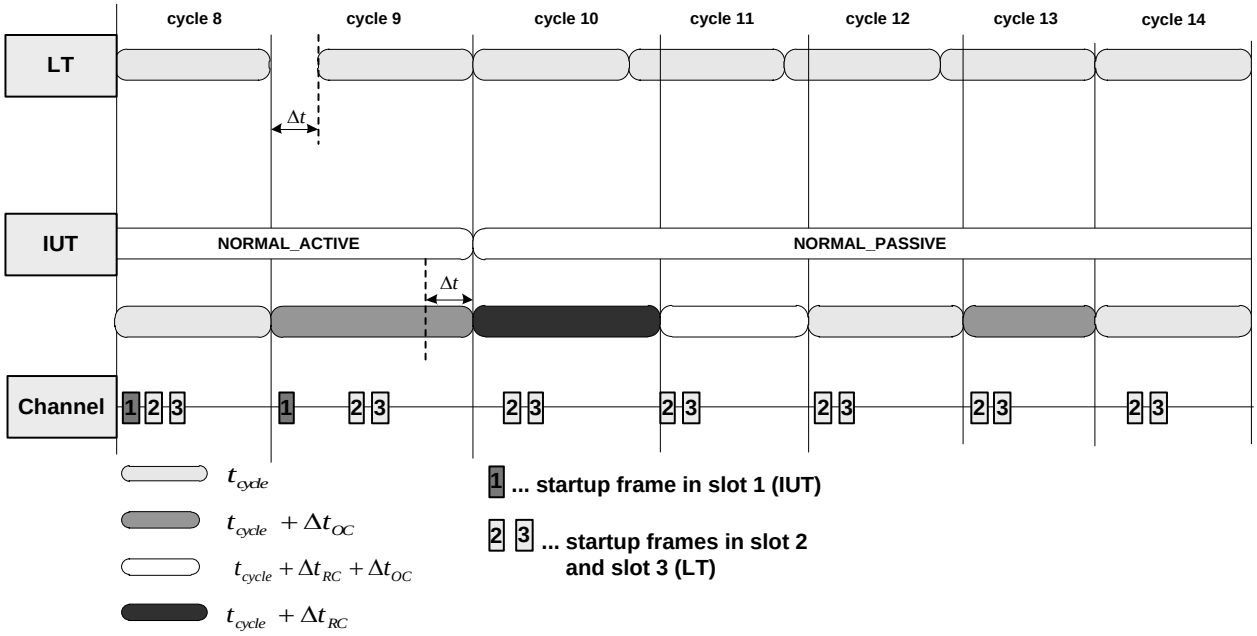


Figure 75 — Resynchronization (transition to POC:normal active not allowed)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes and handles the rate correction violation correctly. The IUT stays in *POC:normal passive* state even if the rate correction is within bounds again because the transition back to *POC:normal active* state is not allowed (*pAllowPassiveToActive* = 0).

7.4.3 Time representation

7.4.3.1 Cycle counter

— Test purpose

Verify the correct behaviour of the cycle counter *vCycleCounter*. The cycle counter is expected to increment by one with every cycle start and to wrap around to 0 after it reached its maximum value of *gCycleCountMax*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 263.

Table 263 — Modification to basic configurations for time representation – cycle counter

Parameter	Modification		
	I	II	III
gCycleCountMax	7	37	*cCycleCountMax*

— Preamble (setup state)

Preamble II.

— Test execution

1) The LT simulates its startup frames in slot 2 for 2 * (*gCycleCountMax* + 1) – 6 cycles starting with cycle 7.

2) In each of the next 2 * (*gCycleCountMax* + 1) – 6 cycles beginning with cycle 7, within [500;600] µT and within [*pMicroPerCycle* – 100; *pMicroPerCycle*] µT after cycle start, it is verified (UT) that the cycle counter is set correctly, i.e *vCycleCounter* = n mod (*gCycleCountMax* + 1) with n = 7, 8, …, 2 * (*gCycleCountMax* + 1). It is also verified (LT) that the IUT's frame holds the correct cycle count, i.e. in each cycle the cycle count equals to n mod (*gCycleCountMax* + 1) with n = 7, 8, …, 2 * (*gCycleCountMax* + 1).

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The cycle counter of the IUT counts correctly. The cycle counter increments by one with every cycle start and wraps around to 0 after it reached *gCycleCountMax*.

7.4.3.2 Macrotick counter

— Test purpose

Verify correct behaviour of the macrotick counter *vMacrotick*. The macrotick counter is expected to start with 0 at the cycle start (first MT) and to reach *gMacroPerCycle* – 1 (last MT) at the end of every cycle. The number of macroticks per cycle shall not be affected by the clock correction algorithm.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 264.

Table 264 — Modification to basic configurations for time representation – macrotick counter

Parameter	Modification
pClusterDriftDamping [µT]	0

Δt = 50 µT.

— **Preamble (setup state)**

Preamble II.

The IUT is the leading coldstart node and transmits its startup frame in slot 1. The LT simulates startup frames in slot 2.

— **Test execution**

1) In cycles 7 to 12, the LT simulates its startup frame in slot 2.

2) In cycles 8 and 9, it is verified (UT) that the macrotick counter is *vMacrotick* = 0 in the first macrotick of the cycle and *vMacrotick* = *gMacroPerCycle* – 1 in the last macrotick of the cycle.

3) In cycle 10, the LT starts its cycle shifted by Δt, i.e. all following cycles are shifted (see Figure 76).

4) In cycle 10, it is verified (UT) that the macrotick counter is *vMacrotick* = 0 in the first macrotick of the cycle (in terms of LT the first macrotick is shifted by Δt) and *vMacrotick* = *gMacroPerCycle* – 1 in the last macrotick of the cycle (in terms of LT the last macrotick is shifted by ∆t, as well).

5) In cycle 11, the LT shortens its cycle by TruncateTowardsZero(Δt / 2) µT. The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the length and the position of the macroticks. The LT still transmits its startup frame with the same offset as in cycle 10 from its cycle start.

6) In cycle 11, it is verified (UT) that the macrotick counter is *vMacrotick* = 0 in the first macrotick of the cycle (in terms of LT the first macrotick is shifted by ∆t).

7) In cycle 11, 500 µT after cycle start and before NIT (in terms of LT this approximates to 500 µT + |Δt| after cycle start and |∆t| before NIT), it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT.

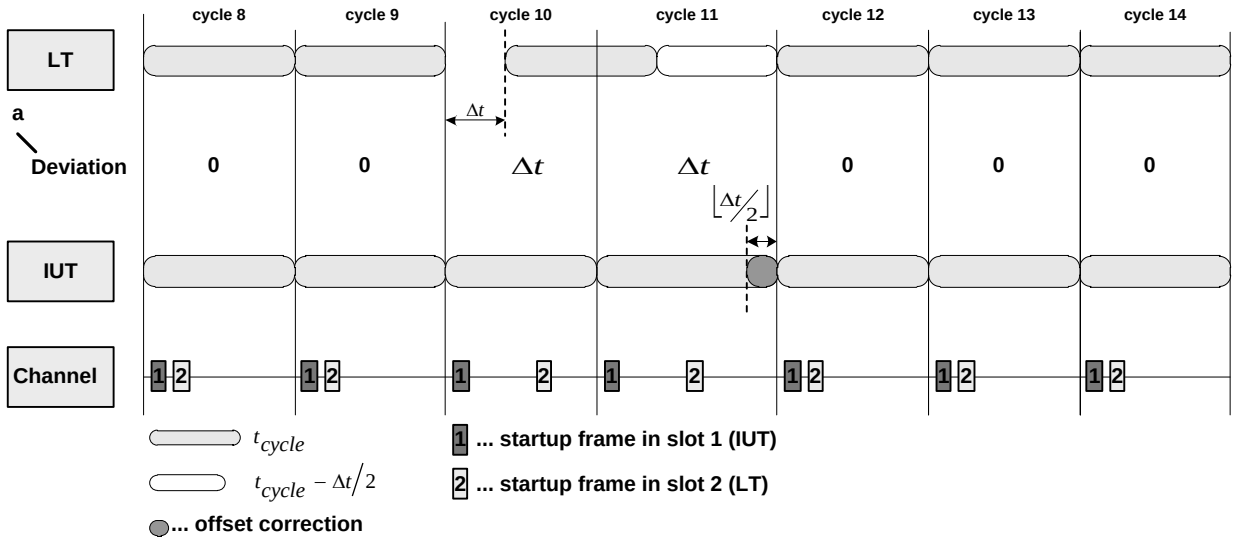
8) In cycle 11, it is verified (UT) that the macrotick counter is *vMacrotick* = *gMacroPerCycle* – 1 in the last macrotick of the cylce.

9) In cycle 12, it is verified (UT) that the macrotick counter is *vMacrotick* = 0 in the first macrotick of the cycle and *vMacrotick* = *gMacroPerCycle* – 1 in the last macrotick of the cycle.

10) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the rate correction value is *vInterimRateCorrection* = 0 + ξ_{IUT} µT and it is verified (UT) that the offset correction value is *vInterimOffsetCorrection* = 0 + ξ_{IUT} µT.

NOTE the time interval in which read access to *vMacrotick* shows the corresponding macrotick value of the protocol engine relies on IUT vendor specific information. See 6.6.4 for further information.

Figure 76 depicts the macrotick counter.



a Expected deviation measured by the IUT.

Figure 76 — Macrotick counter

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT's macrotick counter works correctly. The macrotick counter starts with 0 at the cycle start (first MT) and reaches gMacroPerCycle – 1 (last MT) at the end of every cycle. The number of macroticks per cycle is not affected by the clock correction algorithm.

7.4.3.3 Slot counter and cycle counter in POC:normal passive

— **Test purpose**

Verify the correct behaviour of the slot counter *vSlotCounter* and cycle counter *vCycleCounter* in the *POC:normal passive* state. The slot counter is expected to increment by one with every slot start and to wrap around to 0 after it reached its maximum value of *cSlotIDMax*. The cycle counter is expected to increment by one with every cycle start and to wrap around to 0 after it reached its maximum value of *gCycleCountMax*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 265.

Table 265 — Modification to basic configurations for time representation – slot counter and cycle counter in POC:normal passive

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
gdActionPointOffset [MT]	2	1	1	1	1
gdDynamicSlotIdlePhase [Minislot]	1	1	1	1	2
gdMinislot [MT]	5	4	2	2	2
gdMinislotActionPointOffset [MT]	2	1	1	1	1
gdStaticSlot [MT]	30	29	28	29	54
gMacroPerCycle [MT]	15 200	12 200	6 100	6 100	6 150
gMaxWithoutClockCorrectionPassive [even / odd cycle pair]	1				
gNumberOfMinislots	3 000	3 000	3 000[a]		
gNumberOfStaticSlots	2				
gdCycle [µs]	15 200	12 200	12 200	12 200	12 300
gdNIT [MT]	100	102	21	19	18
pAllowHaltDueToClock	false				
pClusterDriftDamping [µT]	0				
pdListenTimeout [µT]	1 216 730	1 953 172	976 586	488 294	492 296
pMicroPerCycle [µT]	608 000	976 000	488 000	244 000	246 000
pOffsetCorrectionOut [µT]	55	45	45	45	45
pOffsetCorrectionStart [MT]	15 196	12 190	6 095	6 095	6 145
pRateCorrectionOut [µT]	365	586	293	147	148

[a] *gNumberOfMinislots* = 3 000 which is an intentional protocol constraints violation for basic configuration 2a, 2b and 3.

— **Preamble (setup state)**

Preamble II.

The preamble is supplemented by:

1) In cycle 6, the LT stops frame transmission.

2) In cycle 8, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State = NORMAL_PASSIVE*).

— **Test execution**

1) In cycle (9 + o) mod *gCycleCountMax*, it is verified (UT) that *vSlotCounter* = k_n in slot k_n with $k_n = i * n + o$ and *gNumberOfStaticSlots* < k_n < min(*gNumberOfStaticSlots* + *gNumberOfMinislots; cSlotIDMax*) ($i \in [1; gMacroPerCycle]$, n = 0, 1, 2,... and o = 0 and is incremented by 1 in each cycle until o equals i–1).

It is also verified that the slot counters are 0 after the minislot during which the slot counter reached *cSlotIDMax*.

2) In every cycle (m)mod (*gCycleCountMax* + 1) (m = 9, 10, …, 2 * (*gCycleCountMax* + 1)) within [500;600] µT after cycle start and within [*pMicroPerCycle* – 100; *pMicroPerCycle*] µT, it is verified (UT) that the cycle counter is *vCycleCounter* = (m)mod (*gCycleCountMax* + 1).

NOTE the time interval in which read access to *vSlotCounter* shows the corresponding slot counter value of the protocol engine relies on IUT vendor specific information. See 6.6.4 for similar information regarding valid verification window.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The slot counter and cycle counter of the IUT counts correctly in the *POC:normal passive* state.

The slot counters shall be initialized with 1 at the start of each communication cycle and be incremented at the end boundary of each slot and shall be set to zero after they reaches the min(*gNumberOfStaticSlots* + *gNumberOfMinislots*; *cSlotIDMax*).

The cycle counter shall be incremented by one at the beginning of each communication cycle and ranges from 0 to *gCycleCountMax*. When *gCycleCountMax* is reached, the cycle counter *vCycleCounter* shall be reset to zero in the next communication cycle.

7.4.3.4 Macrotick counter in POC:normal passive

— **Test purpose**

Verify correct behaviour of the macrotick counter *vMacrotick* in the *POC:normal passive* state. The macrotick counter is expected to start with 0 at the cycle start (first MT) and to reach *gMacroPerCycle* – 1 (last MT) at the end of every cycle.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 266.

Table 266 — Modification to basic configurations for time representation – macrotick counter in POC:normal passive

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1
pAllowHaltDueToClock	false

— **Preamble (setup state)**

Preamble II.

The preamble is supplemented by:

1) In cycle 6, the LT stops frame transmission.

2) In cycle 8, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT has entered the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

— Test execution

1) In cycle 9, it is verified (UT) that *vMacrotick* = 0 in the first macrotick of the cycle and *vMacrotick* = *gMacroPerCycle* – 1 in the last macrotick of the cycle.

2) In cycle (10 + o) mod *gCycleCountMax*, it is verified (UT) that *vMacrotick* is set to k_n in macrotick k_n with $k_n = i * n + o$ and $k_n <$ *gMacroPerCycle* ($i \in [1; gMacroPerCycle]$, n = 0, 1, 2,... and o = 0 and is incremented by 1 in each cycle until o equals i–1).

NOTE 1 The step width i should be chosen as 1. In case of stringent timing requirements the step width i may be set to a value greater than 1.

NOTE 2 the time interval in which read access to *vMacrotick* shows the corresponding macrotick value of the protocol engine relies on IUT vendor specific information. See 6.6.4 for further information.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The macrotick counter of the IUT.counts correctly in the *POC:normal passive* state. The macrotick counter shall be incremented by one after every *gdMacrotick* µs beginning with zero in the first macrotick at cycle start and ending with *gMacroPerCycle* – 1 in the last macrotick at the end of the cycle.

7.5 Wakeup

7.5.1 General

For dual channel test execution, all test cases in 7.5 have to be performed repeatedly with *pWakeupChannel* = A and *pWakeupChannel* = B.

7.5.2 Correct wakeup decoding

— Test purpose

Verify the reception of a correct WUP. Verify that the bit synchronisation is performed correctly. The IUT shall accept the simulated WUP on channel *pWakeupChannel* comprising a low phase of length *gdWakeupTxActive*, an idle phase of length *gdWakeupTxIdle* – 2 / 8 *gdBit* and a low phase of length *gdWakeupRxLow* – 2 / 8 *gdBit* as correct WUP.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 267.

Table 267 — Modification to Basic configurations for correct wakeup decoding

Parameter	Modification to Basic Configurations					
	1a	1b	2a	2b	3	
	I	I	I	I	I	II
gdWakeupRxIdle [gdBit]	40	59	20	29	8	14
gdWakeupRxLow [gdBit]	40	59	20	29	8	14

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) The LT starts to generate a WUP on channel *pWakeupChannel* with a low phase of *gdWakeupTxActive*, an idle phase of length *gdWakeupTxIdle* and a low phase of length *gdWakeupTxActive* earliest 2 500 µT and latest 0,95 * *pdListenTimeout* µT after the CHI command WAKEUP.

4) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status *vPOC!WakeupStatus* = *RECEIVED_WUP*, that the wakeup pattern received indicator on channel *pWakeupChannel* in the wakeup and startup status is set and, for dual channel test execution, that the wakeup pattern received indicator on the non-wakeup channel in the wakeup and startup status is reset after a period of 0,95 * *pdListenTimeout* + (*gdWakeupTxActive* + *gdWakeupTxIdle* + *gdWakeupTxActive* + *gdWakeupRxIdle*) * *gdBit* / *pdMicrotick* + 500 µT past the CHI command WAKEUP.

5) The UT resets the wakeup pattern received indicator on channel *pWakeupChannel* in the wakeup and startup status and verifies that this indicator has been reset.

6) The UT triggers a second wakeup process with the CHI command WAKEUP.

7) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the second CHI command WAKEUP.

8) The LT starts to generate a WUP on channel *pWakeupChannel* with a low phase of *gdWakeupTxActive*, an idle phase of length *gdWakeupTxIdle* – 1 / 8 *gdBit* and a low phase of *gdWakeupTxActive* – 1 / 8 *gdBit* , earliest 2 500 µT and latest 0,95 * *pdListenTimeout* µT after the CHI command WAKEUP.

9) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status *vPOC!WakeupStatus* = *RECEIVED_WUP* after a period of 0,95 * *pdListenTimeout* + (*gdWakeupTxActive* + *gdWakeupRxIdle* + *gdWakeupRxLow* + *gdWakeupRxIdle*) * *gdBit* / *pdMicrotick* + 500 µT past the second CHI command WAKEUP.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT receives both WUP correctly.

7.5.3 Wakeup decoding with receive window violation

— Test purpose

Verify the correct interpretation of *gdWakeupRxWindow* in the WUS decoding process in case of a violation.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) The LT starts to generate wakeup symbols on channel *pWakeupChannel* earliest 2 500 µT and latest 0,95 * *pdListenTimeout* µT after the CHI command WAKEUP beginning with

   (a) a low phase of length *gdWakeupRxLow* + *gdWakeupRxWindow* + 1 *gdBit*.

   (b) a low phase of length *gdWakeupRxLow* + *gdWakeupTxActive* and a low phase of length *gdWakeupTxActive* separated by an idle phase of (*gdWakeupTxIdle* + *gdWakeupRxWindow* * 0,5 + 1 *gdBit*).

   (c) a low phase of length *gdWakeupRxLow* + *gdWakeupTxActive* and a low phase of length *gdWakeupTxActive* separated by an idle phase of (2 * *gdWakeupTxIdle* + 1 *gdBit*).

(d) a low phase of length *gdWakeupRxLow* + *gdWakeupTxActive* and a low phase of length *gdWakeupTxActive* separated by an idle phase of (2 * *gdWakeupTxIdle* – *gdWakeupRxIdle* + 1 *gdBit*).

(e) a low phase of length *gdWakeupRxLow* + *gdWakeupTxActive* and a low phase of length *gdWakeupTxActive* separated by an idle phase of (2 * *gdWakeupTxIdle* – *gdWakeupRxIdle* – 1 *gdBit*).

4) It is verified (UT)

(a, b, c, d) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*), that the wakeup status is *vPOC!WakeupStatus* = *UNDEFINED* and that the wakeup pattern received indicator in the wakeup and startup status is false on the available channel(s) *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the second low phase.

(e) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status is *vPOC!WakeupStatus* = *RECEIVED_WUP* and that the wakeup pattern received indicator in the wakeup and startup status is true on the wakeup channel *pWakeupChannel* and, for dual channel test execution, false on the non-wakeup channel *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the second low phase.

5) The LT continues to generate and transmit the remaining (*pWakeupPattern – 2)* wakeup symbols on channel *pWakeupChannel* with low phases of length *gdWakeupTxActive* and idle phases of length *gdWakeupTxIdle* starting *gdWakeupRxIdle* after the last low phase.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status is *vPOC!WakeupStatus* = *RECEIVED_WUP* and that the wakeup pattern received indicator in the wakeup and startup status is true on channel *pWakeupChannel* at *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase. For dual channel test execution, it is also verified (UT) that the wakeup pattern received indicator in the wakeup and startup status is false on the non-wakeup channel

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT interprets the WUP receive window correctly and does not recognize a valid WUS if the WUP receive window is violated.

7.5.4 Wakeup decoding with low phase violation

— Test purpose

Verify the correct interpretation of *gdWakeupRxLow* in the WUS decoding process. A WUS with a low phase shorter than *gdWakeupRxLow* should be ignored by the IUT.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) The LT starts to generate wakeup symbols on channel *pWakeupChannel,* earliest 2 500 µT and latest 0,95 * *pdListenTimeout* µT after the CHI command WAKEUP beginning with

   (a) two low phases of length *gdWakeupRxLow* – 1 *gdBit* and *gdWakeupTxActive* separated by an idle phase of length *gdWakeupTxIdle*.

   (b) two low phases of length *gdWakeupTxActive* and *gdWakeupRxLow* – 1 *gdBit* separated by an idle phase of length *gdWakeupTxIdle*.

   (c) a low phase of length (*gdWakeupTxActive* – *gdWakeupRxLow* – 1 *gdBit*), followed by an idle phase of length 1 *gdBit*, followed by a low phase of length *gdWakeupRxLow*, followed by an idle phase of length *gdWakeupTxIdle* and a low phase of length *gdWakeupTxActive*.

4) At *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase it is verified (UT) that the IUT

   (a, b) is in one of the wakeup states (*vPOC!State* = *WAKEUP*), that the wakeup status is *vPOC!WakeupStatus* = *UNDEFINED* and that the wakeup pattern received indicator in the wakeup and startup status is false on the available channel(s),

   (c) is in the state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status is *vPOC!WakeupStatus* = *RECEIVED_WUP* and that the wakeup pattern received indicator in the wakeup and startup status is true on channel *pWakeupChannel* and, for dual channel test execution, false on the non-wakeup channel, at *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase.

5) Starting at *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase of step 3 the LT continues to generate and transmit the remaining (*pWakeupPattern – 2)* wakeup symbols on channel *pWakeupChannel* with low phases of length *gdWakeupTxActive* and idle phases of length *gdWakeupTxIdle* starting *gdWakeupRxIdle* after the last low phase.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status is *vPOC!WakeupStatus* = *RECEIVED_WUP* and that the wakeup pattern received indicator in the wakeup and startup status is true on channel *pWakeupChannel* at *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase. For dual channel test execution, it is also verified (UT) that the wakeup pattern received indicator in the wakeup and startup status is false on the non-wakeup channel.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT interprets the low phase of WUP correctly and does not recognize a valid WUS if the low phase is too short.

7.5.5 Wakeup decoding idle phase violation

— **Test purpose**

Verify the correct interpretation of *gdWakeupRxIdle* in the WUS decoding process. A WUS with an idle phase shorter than *gdWakeupRxIdle* should be ignored by the IUT.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP.

2) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

3) The LT starts to generate wakeup symbols on channel *pWakeupChannel*, earliest 2 500 µT and latest 0,95 * *pdListenTimeout* µT after the CHI command WAKEUP beginning with

   (a) two low phases of length *gdWakeupTxActive* separated by an idle phase of *gdWakeupRxIdle* – 1 *gdBit*.

   (b) a low phase of length *gdWakeupTxActive* followed by an idle phase of length (*gdWakeupTxIdle* – *gdWakeupRxIdle* – 1 *gdBit*), followed by a low phase of length 1 *gdBit*, followed by an idle phase of length *gdWakeupRxIdle* and a low phase of length *gdWakeupTxActive*.

4) At *gdWakeupRxIdle* * *gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase it is verified (UT) that the IUT

   (a) is in one of the wakeup states (*vPOC!State* = *WAKEUP*), that the wakeup status is *vPOC!WakeupStatus* = *UNDEFINED* and that the wakeup received indicator in the wakeup and startup status is false on the available channel(s),

   (b) is in the state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status is *vPOC!WakeupStatus* = *RECEIVED_WUP* and that the wakeup pattern received indicator in the wakeup and startup status is true on channel *pWakeupChannel* and, for dual

channel test execution, false on the non-wakeup channel, at *gdWakeupRxIdle * gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase.

5) Starting at *gdWakeupRxIdle * gdBit / pdMicrotick* + 500 µT after the end of the last low phase of step 3 the LT continues to generate and transmit the remaining (*pWakeupPattern – 2)* wakeup symbols on channel *pWakeupChannel* with low phases of length *gdWakeupTxActive* and idle phases of length *gdWakeupTxIdle* starting *gdWakeupRxIdle* after the last low phase.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*), that the wakeup status is *vPOC!WakeupStatus* = *RECEIVED_WUP* and that the wakeup pattern received indicator in the wakeup and startup status is true on channel *pWakeupChannel* at *gdWakeupRxIdle * gdBit* / *pdMicrotick* + 500 µT after the end of the last low phase. For dual channel test execution, it is also verified (UT) that the wakeup pattern received indicator in the wakeup and startup status is false on the non-wakeup channel.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT interprets the idle phase of WUP correctly and does not recognize a valid WUS if the idle phase is too short.

7.5.6 Wakeup listen without noise

— Test purpose

Verify the interpretation of the listen time-out *pdListenTimeout* for a wakeup without bus interference.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) It is verified (UT) that the *vPOC!WakeupStatus* is initially *UNDEFINED* at t0.

2) The UT triggers the wakeup process with the CHI command WAKEUP at t1 2 500 µT after t0.

3) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) 2 500 µT after the CHI command WAKEUP.

4) It is verified (LT) that the IUT starts correct WUP generation at t2 = t1 + *pdListenTimeout* + 1 000 ± 1 000[4] + ξ µT.

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t3 (t3 = t1 + *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick*) + 2 500 µT.

6) For dual channel test execution it is verified (LT) that the non-wakeup channel (channel B if *pWakeupChannel* = A, channel A if *pWakeupChannel* = B) stays idle throughout the entire test.

Figure 77 depicts the wakeup listen without noise – *pWakeupChannel* = A.

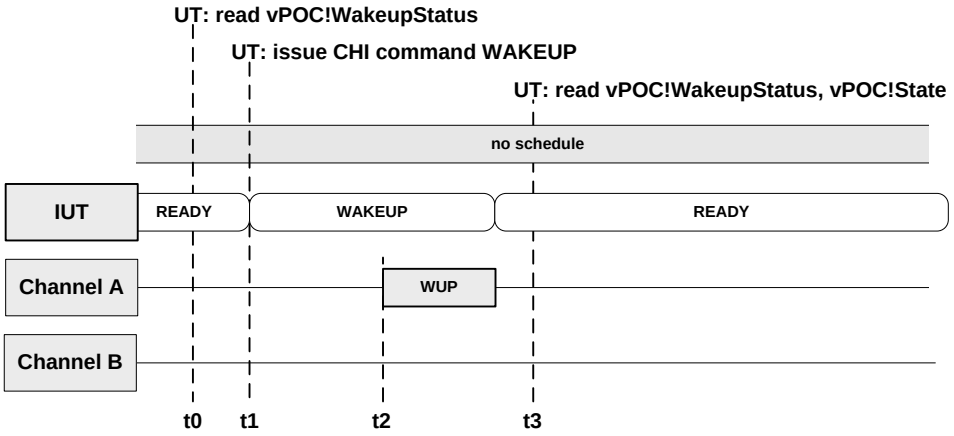


Figure 77 — Wakeup listen without noise – pWakeupChannel = A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits a correct WUP after the configured listen time-out has expired.

7.5.7 Wakeup listen with initial noise interference

— Test purpose

Verify the interpretation of the listen time-out *pdListenTimeout* for a wakeup with bus interference.

— Applicability

DC.

— Configuration

All basic configurations using the modifications as listed in Table 268.

---

4) The term (1 000 ± 1 000) is intended to compensate the 2 000 µT CHI delay which is allowed for command execution according to 6.6.2.

Table 268 — Modification to basic configurations for wakeup – wakeup listen with initial noise interference

Parameter	Modification
gListenNoise	9

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The LT sets

   (a) channel B,

   (b) channel A and

   (c) both channels

   to low at t0.

2) The UT triggers the wakeup process with the CHI command WAKEUP at t1 between t0 and t2 – 2 500 µT.

3) The LT sets

   (a) channel B,

   (b) channel A and

   (c) both channels

   to idle at t2 = t0 + ceil((*gListenNoise* – 2) * *pdListenTimeout* * *pSamplesPerMicrotick* / *cSamplesPerBit*) * *cSamplesPerBit* / *pSamplesPerMicrotick* µT.

4) It is verified (LT) that the IUT starts to transmit a correct WUP after the wakeup listen phase at t3 = t2 + (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* + ξ µT + (φ_{Rx} + φ_{Tx}) ns.

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t3 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

Figure 78 depicts the wakeup listen with Initial noise interference – *pWakeupChannel* = A – noise on channel B.

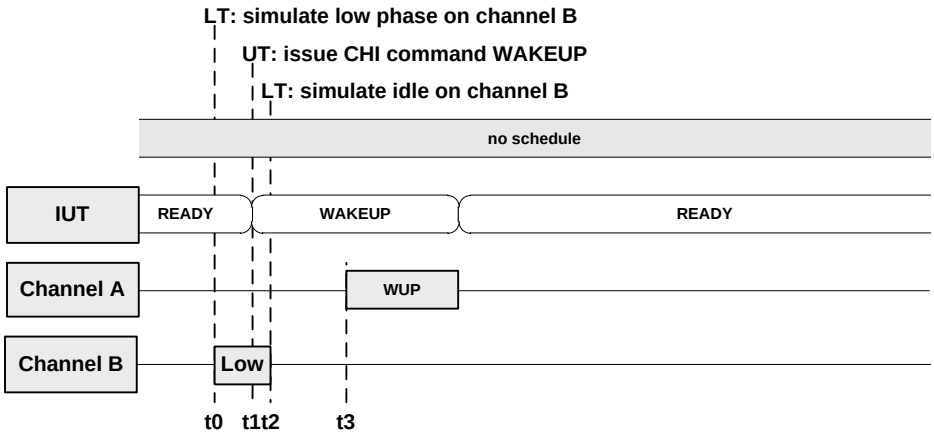


Figure 78 — Wakeup listen with Initial noise interference – pWakeupChannel = A – noise on channel B

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits a correct WUP with the appropriate delay after both channels became idle again.

7.5.8 Wakeup listen with noise interference

— Test purpose

Verify the interpretation of the listen time-out *pdListenTimeout* and noise multiplier *gListenNoise* for a wakeup with interrupting bus interference.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 269.

Table 269 — Modification to basic configurations for wakeup – wakeup listen with noise interference

Parameter	Modification	
	I	II
gListenNoise	2	10

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0.

2) The LT sets the wakeup channel to low at t1 = t0 + 0,4 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) For dual channel test execution the LT sets the wakeup channel to idle and the non-wakeup channel to low at t2 = t1 + ceil(0,4 * *pdListenTimeout) µT*.

4) The LT sets all channel(s) to idle at t3 = t1 + ceil(0,4 * *pdListenTimeout) µT* – 1 µT + ceil(*pdListenTimeout* * *pSamplesPerMicrotick* / *cSamplesPerBit*) * *cSamplesPerBit* / *pSamplesPerMicrotick* µT.

5) It is verified (LT) that the IUT starts to transmit a WUP after the wakeup listen phase at

   (I) t4 = t0 + 1 000 ± 1 000[5)] + *pdListenTimeout* * *gListenNoise* + *ξ* µT and

   (II) t4 = t3 + (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* + *ξ* µT + (φ_{Rx} + φ_{Tx}) ns.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t4 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

5) The term (1 000 ± 1 000) is intended to compensate the 2 000 µT CHI delay which is allowed for command execution according to 6.6.2.

Figure 79 depicts the wakeup listen with noise interference – DC, *pWakeupChannel* = A and *gListenNoise* = 2.

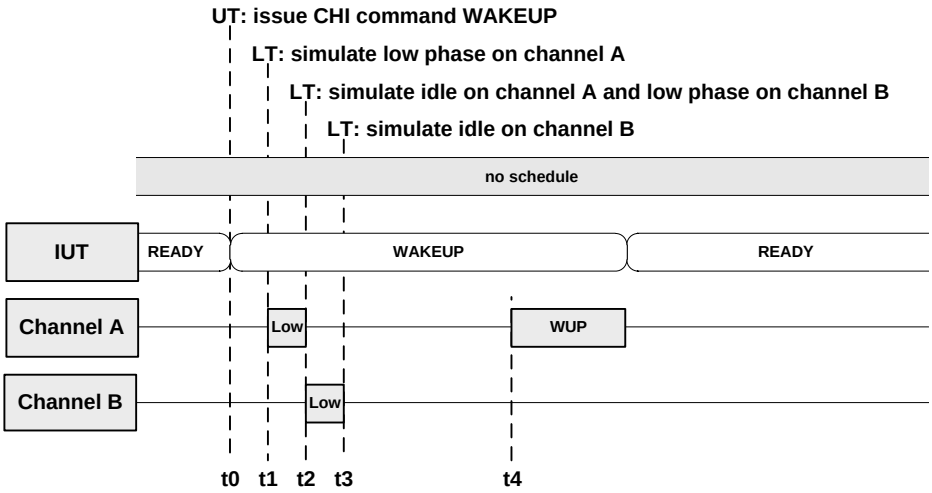


Figure 79 — Wakeup listen with noise interference – DC, pWakeupChannel = A and gListenNoise = 2

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

(I) The IUT transmits a correct WUP after 1 000 ± 1 000 + *gdListenTimeout* * *gListenNoise* + ξ µT past the CHI command WAKEUP.

(II) The IUT transmits a correct WUP with the appropriate delay after all channel(s) became idle again.

7.5.9 Wakeup listen with enduring noise interference

— Test purpose

Verify the interpretation of the listen time-out *pdListenTimeout* and noise multiplier *gListenNoise* for a wakeup with enduring interrupting bus interference. The IUT shall wait until *pdListenTimeout* * *gListenNoise* expires before it starts WUP transmission if the non-wakeup channel is permanently disturbed and the wakeup channel becomes idle.

— Applicability

DC.

— Configuration

All basic configurations using the modifications as listed in Table 270.

Table 270 — Modification to basic configurations for wakeup – wakeup listen with enduring noise interference

Parameter	Modification
gListenNoise	9

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The LT sets the non-wakeup channel (channel A if *pWakeupChannel* = B, channel B if *pWakeupChannel* = A) to low at t0.

2) The UT triggers the wakeup process with the CHI command WAKEUP at t1 = t0 + 0,2 * *pdListenTimeout* µT (*vPOC!State* = *WAKEUP*).

3) The LT sets channel *pWakeupChannel* to low at t2 = t0 + 0,95 * *pdListenTimeout* µT.

4) The LT sets channel *pWakeupChannel* to idle at t3 = t2 + 0,2 * *pdListenTimeout* µT.

5) It is verified (LT) that the IUT starts to transmit a correct WUP on channel *pWakeupChannel* after the wakeup listen phase at t4 = t1 + *gListenNoise* * *pdListenTimeout* + 1 000 ± 1 000 µT + ξ µT.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t4 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

Figure 80 depicts the wakeup listen with enduring noise interference – *pWakeupChannel* = A.

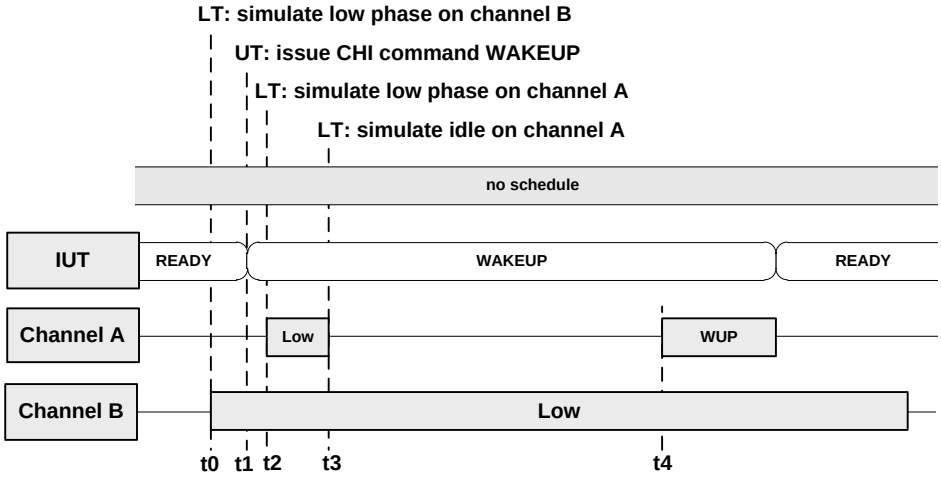


Figure 80 — Wakeup listen with enduring noise interference – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a correct WUP with the appropriate delay after the wakeup channel became idle again.

7.5.10 Wakeup listen with symbol interference (DC)

— **Test purpose**

Verify the interpretation of the listen time-out *pdListenTimeout* and noise multiplier *gListenNoise* for a wakeup with symbol interference. The IUT shall wait until *pdListenTimeout* * *gListenNoise* expires before it starts WUP transmission if the non-wakeup channel is permanently disturbed and a symbol is received on the wakeup channel.

— **Applicability**

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 271.

Table 271 — Modification to basic configurations for wakeup – wakeup listen with symbol interference (DC)

Parameter	Modification
gListenNoise	9

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The LT sets the non-wakeup channel (channel A if *pWakeupChannel* = B, channel B if *pWakeupChannel* = A) to low at t0.

2) The UT triggers the wakeup process with the CHI command WAKEUP at t1 = t0 + 0,2 * *pdListenTimeout* µT (*vPOC!State* = *WAKEUP*).

3) The LT simulates a valid symbol of length *gdTSSTransmitter* + *cdCAS* on channel *pWakeupChannel* starting at

   (a) t2 = t0 + 0,95 * *pdListenTimeout* µT and

(b) t2 = t1 + *gListenNoise* * *pdListenTimeout* – 0,5 * (*gdTSSTransmitter* + *cdCAS*) * *gdBit* / *pdMicrotick* µT.

4) It is verified (LT) that the IUT starts to transmit a WUP on *pWakeupChannel* at t4 = t3 + (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* + ξ µT + (φ_{Rx} + φ_{Tx}) ns (with t3 = t2 + *gdTSSTransmitter* + *cdCAS*).

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t4 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

Figure 81 depicts the wakeup listen with symbol interference (DC) – *pWakeupChannel* = A.

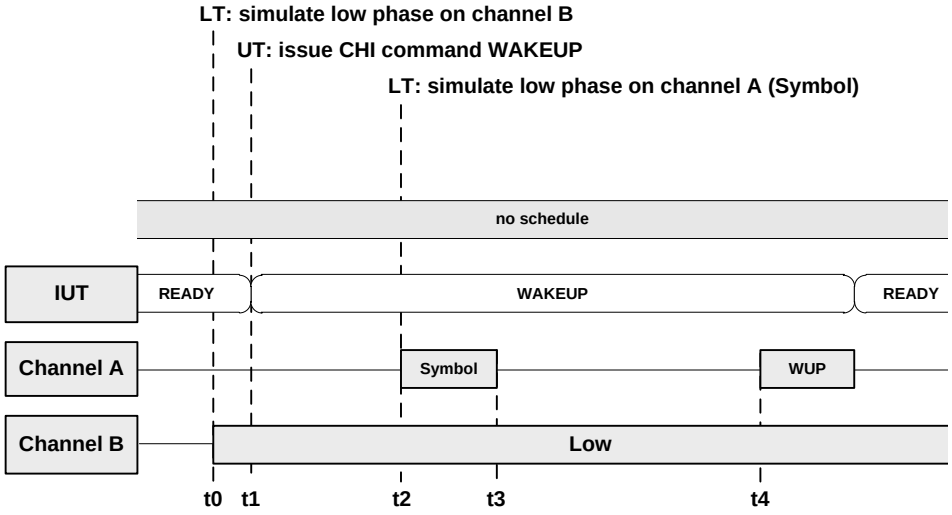


Figure 81 — Wakeup listen with symbol interference (DC) – pWakeupChannel = A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits a correct WUP with the appropriate delay, if the non-wakeup channel is permanently disturbed and a symbol is received on the wakeup channel.

7.5.11 Wakeup listen with indirect header interference

— Test purpose

Verify the recognition of a frame header on the non-wakeup channel (channel A if *pWakeupChannel* = B, channel B if *pWakeupChannel* = A) during wakeup listen phase.

— Applicability

DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT simulates a frame header of a sync frame in slot 1 (Null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the frame ID is 1, the payload length is 1, the cycle count 0. The LT has to simulate the header plus the BSS preceding the frame's payload section in order to generate a valid header.) on the non-wakeup channel (channel A if *pWakeupChannel* = B, channel B if *pWakeupChannel* = A) at t1 = t0 + 0,9 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) It is verified (UT) that the *vPOC!State* = *READY* and *vPOC!WakeupStatus* = *RECEIVED_HEADER* at t2 = t0 + 0,95 * *pdListenTimeout* + (*gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick* + 500 µT.

4) It is verified (LT) that the IUT does not transmit a WUP on channel *pWakeupChannel* until t2 and for at least (*gListenNoise* + 1) * *pdListenTimeout* after t2.

Figure 82 depicts the wakeup listen with indirect header interference – *pWakeupChannel* = A.

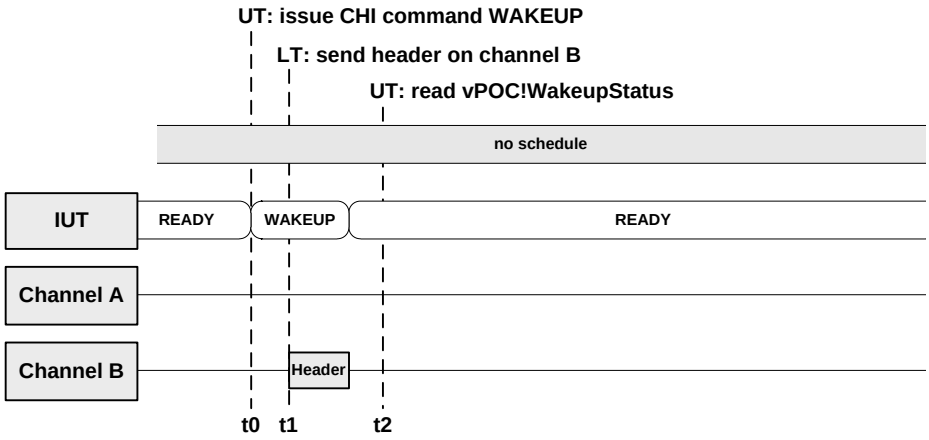


Figure 82 — Wakeup listen with indirect header interference – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the wakeup process after reception of a frame header.

7.5.12 Wakeup listen with direct header interference

— **Test purpose**

Verify the recognition of a frame header on the wakeup channel *pWakeupChannel* during wakeup listen phase.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT simulates a frame header of a sync frame in slot 1 (Null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the frame ID is 1, the payload length is 1, the cycle count 0. The LT has to simulate the header plus the BSS preceding the frame's payload section in order to generate a valid header.) on the wakeup channel *pWakeupChannel* at t1 = t0 + 0,9 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) It is verified (UT) that the *vPOC!State* = *READY* and *vPOC!WakeupStatus* = *RECEIVED_HEADER* at t2 = t0 + 0,95 * *pdListenTimeout* + (*gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick* + 500 µT.

4) It is verified (LT) that the IUT does not transmit a WUP on channel *pWakeupChannel* until t2 and for at least (*gListenNoise* + 1) * *pdListenTimeout* after t2.

Figure 83 depicts the wakeup listen with direct header interference – *pWakeupChannel* = A.

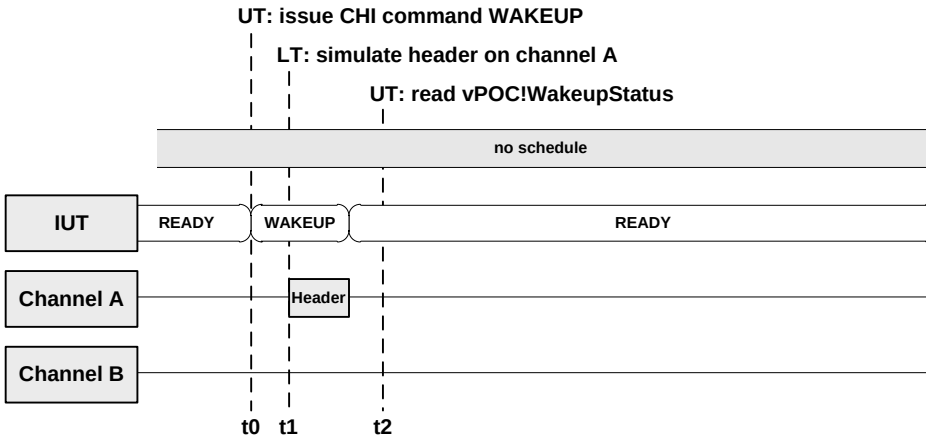


Figure 83 — Wakeup listen with direct header interference – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the wakeup process after reception of a frame header.

7.5.13 Wakeup listen with indirect WUP interference

— **Test purpose**

Verify the reaction to the reception of a WUP on the non-wakeup channel (channel A if *pWakeupChannel* = B, channel B if *pWakeupChannel* = A) during wakeup listen phase. The IUT shall delay its WUP transmission due to the WUP on the non-wakeup channel.

— **Applicability**

DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT simulates a valid WUP (with *pWakeupPattern* = 2, *gdWakeupTxActive* and *gdWakeupTxIdle* according to used basic configuration) on the non-wakeup channel at t1 = t0 + 0,2 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) It is verified (UT) that the IUT is in one of the wakeup states (*vPOC!State* = *WAKEUP*) after the reception of the LT's WUP at t2 = t0 + 0,25 * *pdListenTimeout* + (2 * *gdWakeupTxActive* + 2 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

4) It is verified (LT) that the IUT transmits a correct WUP on channel *pWakeupChannel* after t3 = t1 + (2 * *gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* + ξ µT + (φ_{Rx} + φ_{Tx}) ns.

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t2 + *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

Figure 84 depicts the wakeup listen with indirect WUP interference – *pWakeupChannel* = A.

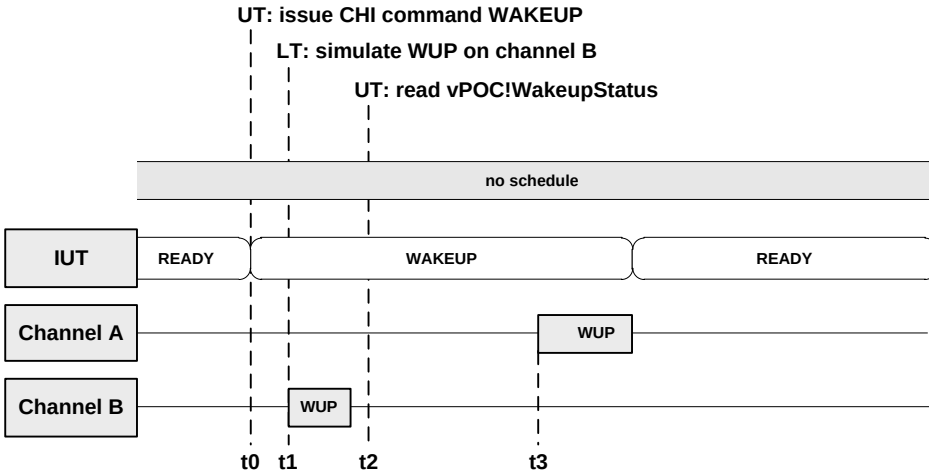


Figure 84 — Wakeup listen with indirect WUP interference – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT delays its WUP transmission due to the WUP received on the non-wakeup channel.

7.5.14 Wakeup listen with direct WUP interference

— Test purpose

Verify the reaction to the reception of a WUP on the wakeup channel *pWakeupChannel* during wakeup listen phase.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT simulates a valid WUP (with *pWakeupPattern* = 2, *gdWakeupTxActive* and *gdWakeupTxIdle* according to used basic configuration) on the wakeup channel *pWakeupChannel* at t1 = t0 + 0,2 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY* and *vPOC!WakeupStatus* = *RECEIVED_WUP*) at t2 = t0 + 0,25 * *pdListenTimeout* + (2 * *gdWakeupTxActive* + 2 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT after the reception of the LT's WUP.

4) It is verified (LT) that the IUT does not transmit a WUP on channel *pWakeupChannel* until t1 and for at least (*gListenNoise* + 1) * *pdListenTimeout* after t1.

Figure 85 depicts the wakeup listen with direct WUP interference – *pWakeupChannel* = A.

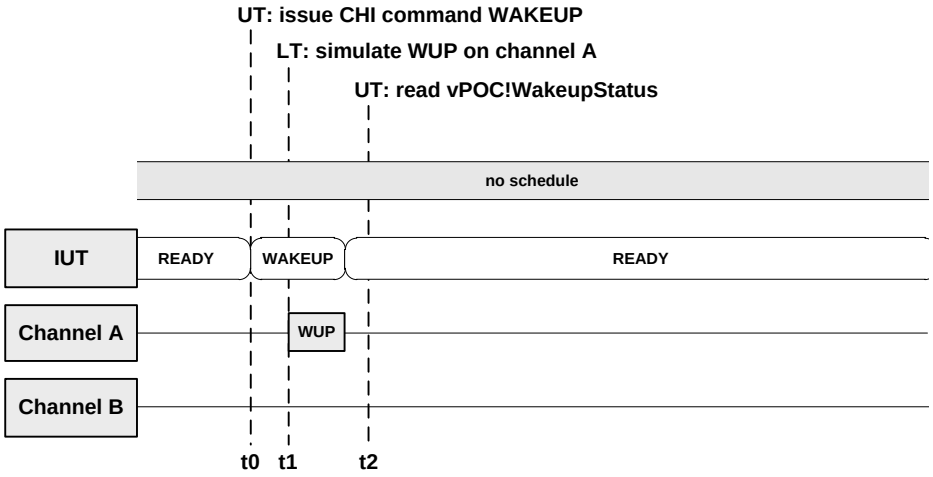


Figure 85 — Wakeup listen with direct WUP interference – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts its wakeup after reception of a correct WUP on the wakeup channel.

7.5.15 WUP with indirect collision

— **Test purpose**

Verify the correct handling of a collision on the non-wakeup channel. The IUT shall ignore a low phase or a WUP on the non-wakeup channel during WUP transmission.

— **Applicability**

DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT measures the start of the WUP sent by the IUT. The falling edge of the first WUS is at t1.

3) (a) At t2 = t1 + (*gdWakeupTxActive* + 0,5 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* µT during the first idle phase of the IUT's WUP, the LT simulates a low phase of length *cdWakeupMaxCollision* + 1 *gdBit* on the non-wakeup channel.

   (b) At t2 = t1 + (2 ± 2) * *gdBit / pdMicrotick* µT during the first low phase of the IUT's WUP, the LT simulates a WUP as configured (*pWakeupPattern* low phases of length *gdWakeupTxActive* interleaved by idle phases of length *gdWakeupTxIdle*) on the none wakeup channel.

4) It is verified (LT) that the IUT transmits a correct WUP according to the configuration.

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* at t3 = t1 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

Figure 86 depicts the WUP with indirect collision – *pWakeupChannel* = A, variant (a).

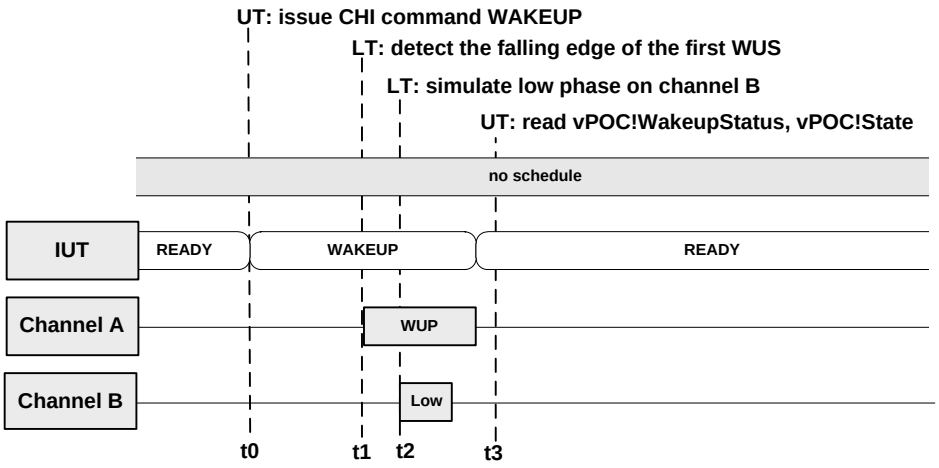


Figure 86 — WUP with indirect collision – pWakeupChannel = A, variant (a)

Figure 87 depicts the WUP with indirect collision – *pWakeupChannel* = A, variant (b).

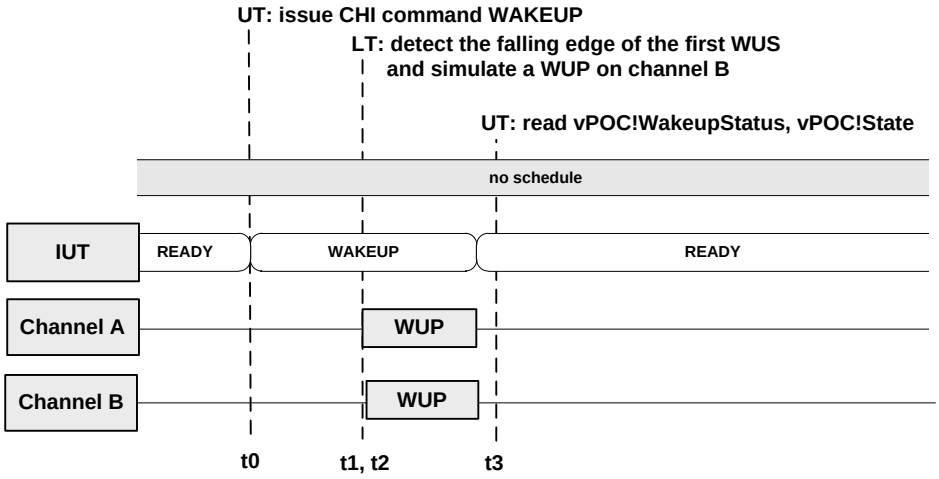


Figure 87 — WUP with indirect collision – pWakeupChannel = A, variant (b)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT ignores the low phase or a WUP on the non-wakeup channel during WUP transmission and transmits its WUP correctly.

7.5.16 WUP with WUP collision

— **Test purpose**

Verify the correct handling of a collision on the wakeup channel. The IUT shall abort its WUP transmission if the collision occurs during a WUS idle phase and shall continue its WUP transmission if the collision is at the same timing.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT measures the start of the WUP sent by the IUT. The falling edge of the first WUS is at t1.

3) (a) At t2 = t1 + (*gdWakeupTxActive* + 0,5 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* µT during the first idle phase of the IUT's WUP, the LT starts to generate a WUP (three low phases of length *gdWakeupTxActive* and two idle phases of length *gdWakeupTxIdle*) on the wakeup channel *pWakeupChannel*.

   (b) At t2 = t1 + (2 ± 2) * *gdBit* / *pdMicrotick* µT during the first low phase of the IUT's WUP, the LT starts to generate a WUP (*pWakeupPattern* low phases of length *gdWakeupTxActive* interleaved by idle phases of length *gdWakeupTxIdle*) on the wakeup channel *pWakeupChannel*.

4) It is verified (LT)

   (a) that the IUT stops WUP generation, i.e. it is verified (LT) that the IUT does not generate any low phase for at least (*gListenNoise* + 1) * *pdListenTimeout* after t2.

   (b) that the IUT continues the WUP transmission.

5) It is verified (UT) that the IUT

   (a) is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *COLLISION_WUP* at t3 = t2 + (3 * *gdWakeupTxActive* + 2 * *gdWakeupTxIdle* + *gdWakeupRxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

   (b) is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t3 = t2 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT. It is also verfified (UT) that the wakeup pattern received indicator on channel *pWakeupChannel* in the wakeup and startup status is reset.

Figure 88 depicts the WUP with WUP collision – *pWakeupChannel* = A, variant (a).

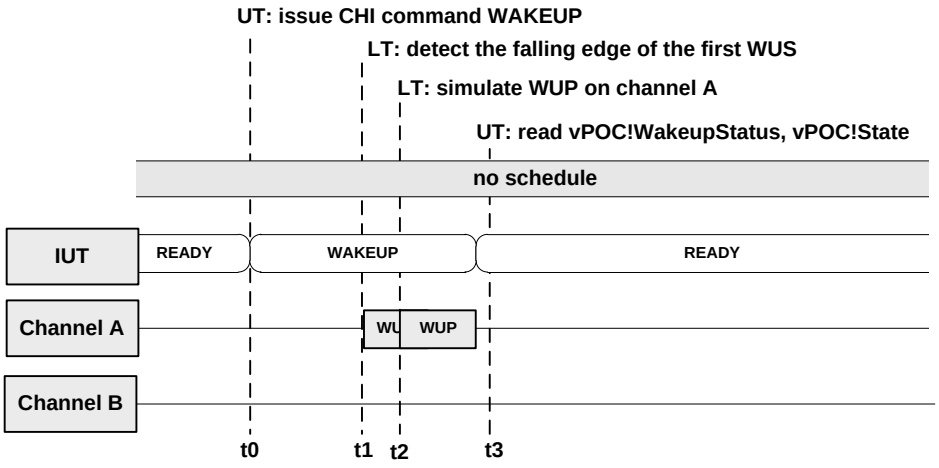


Figure 88 — WUP with WUP collision – pWakeupChannel = A, variant (a)

Figure 89 depicts the WUP with WUP collision – WUP collision at the same timing, pWakeupChannel = A, variant (b).

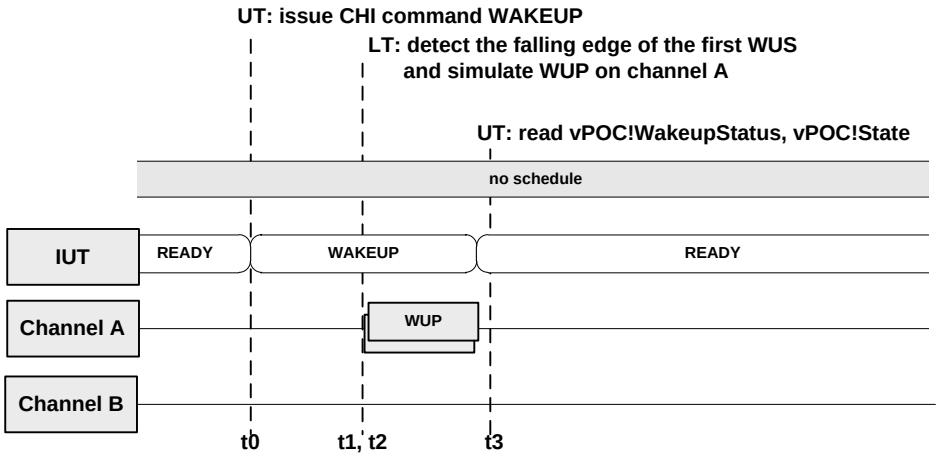


Figure 89 — WUP with WUP collision – WUP collision at the same timing, pWakeupChannel = A, variant (b)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT shall recognize a collision on the wakeup channel during the idle phase of WUP transmission, shall abort its WUP transmission and indicate the cause for the wakeup abortion accordingly. The IUT shall continue WUP transmission when a WUP collision occurs at the same timing.

7.5.17 WUP with header collision

— Test purpose

Verify the correct handling of a header collision on the wakeup channel. The IUT shall abort its WUP transmission.

— Applicability

SC, DC.

NOTE DC only test variant (a).

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT measures the start of the WUP sent by the IUT. The falling edge of the first WUS is at t1.

3) At t2 = t1 + (*gdWakeupTxActive* + 0,5 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* µT during the first idle phase of the IUT's WUP, the LT starts to generate two sync frame headers for slot 1 (Null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the payload length is 1, the cycle count 0. The LT has to simulate the header plus the BSS preceding the frame's payload section in order to generate a valid header.)

   (a) on the wakeup channel *pWakeupChannel* and

   (b) on the non-wakeup channel.

   The interval between the beginning of the two frame headers is *gdCycle*.

4) It is verified (LT) that the IUT

   (a) stops WUP generation, i.e. it is verified (LT) that the IUT does not generate any low phase for at least (*gListenNoise* + 1) * *pdListenTimeout* after t2.

   (b) continues WUP transmission

5) It is verified (UT) that the IUT is

   (a) in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *COLLISION_HEADER* at t3 = t1 + (*gdWakeupTxActive* + 0,75 * *gdWakeupTxIdle* + *gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick* + *gdMacroPerCycle* * *gdMacrotick* / *pdMicrotick* + 500 µT.

   (b) in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* at t3 = t1 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

Figure 90 depicts the WUP with header collision – *pWakeupChannel* = A.

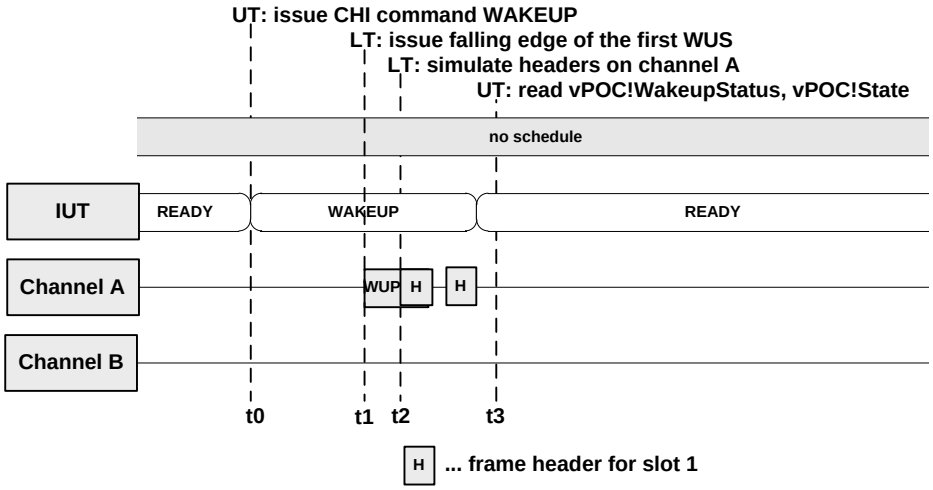


Figure 90 — WUP with header collision – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

(a) The IUT recognizes the collision on the wakeup channel during WUP transmission, aborts its WUP transmission and indicates the cause for the wakeup abortion accordingly.

(b) The IUT's wakeup transmission is not disturbed by headers received on the non-wakeup channel.

7.5.18 WUP with unknown collision

— **Test purpose**

Verify the correct handling of an unspecified collision on the wakeup channel. The IUT shall abort its WUP transmission.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT measures the start of the WUP sent by the IUT. The falling edge of the first WUS is at t1.

3) At t2 = t1 + (*gdWakeupTxActive* + 0,5 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* µT during the first idle phase of the IUT's WUP, the LT generates two low phases of length *cdWakeupMaxCollision* + 1 *gdBit* on the wakeup channel *pWakeupChannel*. The interval between the two low phases is 2 * *cChannelIdleDelimiter*.

4) It is verified (LT) that the IUT stops WUP generation, i.e. it is verified (LT) that the IUT does not generate any low phase for at least (*gListenNoise* + 1) * *pdListenTimeout* after t2.

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *COLLISION_UNKNOWN* at t3 = t2 + (*cdWakeupMaxCollision* + 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* µT + 500 µT.

Figure 91 depicts the WUP with unknown collision – *pWakeupChannel* = A.

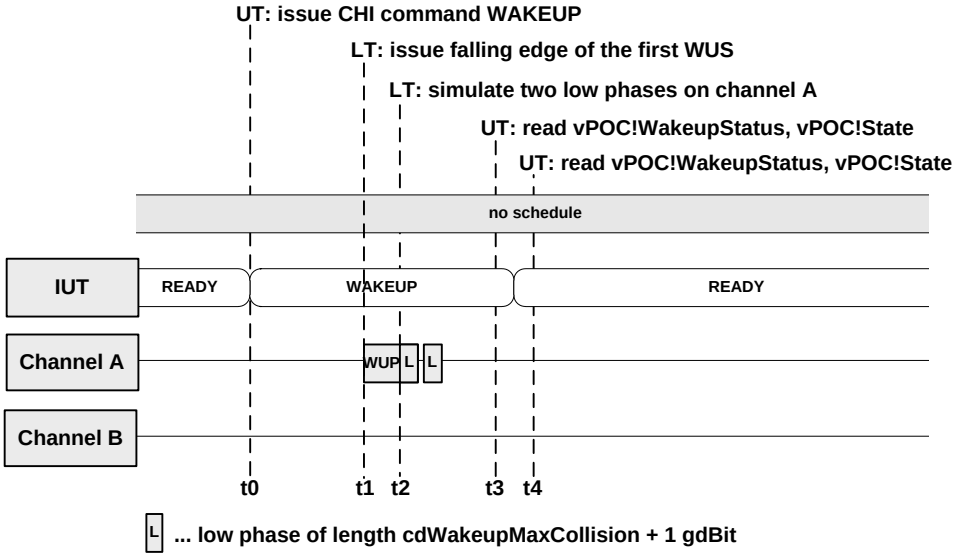


Figure 91 — WUP with unknown collision – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT recognizes the collision on the wakeup channel during WUP transmission, aborts its WUP transmission and indicates the cause for the wakeup abortion accordingly.

7.5.19 WUP with insignificant collision

— Test purpose

Verify the correct handling of an insignificant collision during the encoding process. The IUT shall ignore the collision and shall transmit its WUP correctly. Verify that the bit synchronisation is performed correctly. The IUT shall not abort its wakeup transmission after a reception of a low phase of length *cdWakeupMaxCollision* – 5 / 8 *gdBit*.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT measures the start of the WUP sent by the IUT. The falling edge of the first WUS is at t1.

3) At t2 = t1 + (*gdWakeupTxActive* + 0,5 * *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* ± 0,25 * *gdWakeupTxIdle* * *gdBit* / *pdMicrotick* µT during the first idle phase of the IUT's WUP, the LT generates two low phases of length *cdWakeupMaxCollision* – 5 / 8 *gdBit* interleaved by an idle phase of length 13 / 8 *gdBit* on the wakeup channel *pWakeupChannel*.

4) It is verified (LT) that the IUT continues WUP generation and that the IUT transmits a correct WUP. The length of the idle phases of the IUT's WUP has to be within the range [*gdWakeupTxIdle* * *gdBit* – 4 * *gdSampleClockPeriod*; *gdWakeupTxIdle* * *gdBit* + 6 * *gdSampleClockPeriod*].

5) At t3 = t1 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT, it is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *TRANSMITTED*.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT ignores the insignificant collision on the wakeup channel and transmits its WUP correctly.

7.5.20 Wakeup listen and permanent noise on available channel(s) longer than tWakeupNoise

— Test purpose

Verify the wakeup behaviour when channel *pWakeupChannel* becomes idle after a low phase on all available channel(s), which lasts until the timer *tWakeupNoise* expires. The IUT shall start WUP transmission immediately after channel *pWakeupChannel* becomes idle again.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The LT sets all available channel(s) to low at t0.

2) The UT triggers the wakeup process with the CHI command WAKEUP at t1 between t0 and t0 + 0,2 * *pdListenTimeout* µT (*vPOC!State* = *WAKEUP*).

3) The LT sets channel *pWakeupChannel* to idle at t2 = t0 + ceil((*gListenNoise* + 1) * *pdListenTimeout* * *pSamplesPerMicrotick* / *cSamplesPerBit*) * *cSamplesPerBit* / *pSamplesPerMicrotick* µT.

4) It is verified (LT) that the IUT transmits a correct WUP on channel *pWakeupChannel* at t3 = t2 + (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * (*gdBit* / *pdMicrotick*) + ξ µT + (φ_{Rx} + φ_{Tx}) ns.

5) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* at t3 + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick* + 500 µT after the UT issued the CHI command WAKEUP.

Figure 92 depicts the wakeup listen and permanent noise on available channel(s) longer than *tWakeupNoise* – DC, *pWakeupChannel* = A.

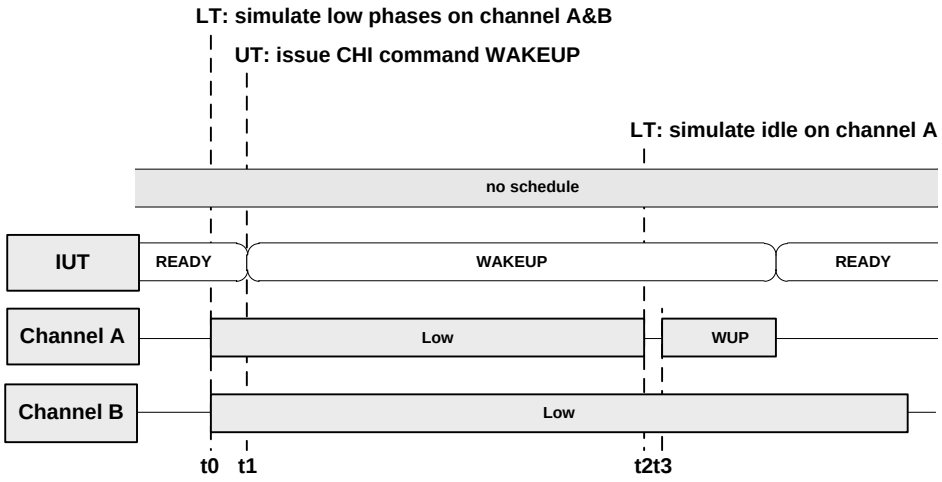


Figure 92 — Wakeup listen and permanent noise on available channel(s) longer than tWakeupNoise – DC, pWakeupChannel = A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT transmits a correct WUP with the appropriate delay after the wakeup channel became idle again. The IUT transmits its WUP after a period of *cChannelIdleDelimiter* past the wakeup channel became idle.

7.5.21 Wakeup listen with Header Interference on both channels

— Test purpose

Verify the recognition of a frame header on the wakeup channel and on the non-wakeup channel during wakeup listen phase.

— Applicability

DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT simulates a frame header of a sync frame in slot 1 (Null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the frame ID is 1, the payload length is 1, the cycle count 0. The LT has to simulate the header plus the BSS preceding the frame's payload section in order to generate a valid header.) on both channels at t1 = t0 + 0,9 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *RECEIVED_HEADER* at t2 = t0 + 0,95 * *pdListenTimeout* + (*gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick* + 500 µT.

4) It is verified (LT) that the IUT does not transmit a WUP on channel *pWakeupChannel* for at least (*gListenNoise* + 1) * *pdListenTimeout* after t1.

Figure 93 depicts the wakeup listen with header interference on both channels – *pWakeupChannel* = A.

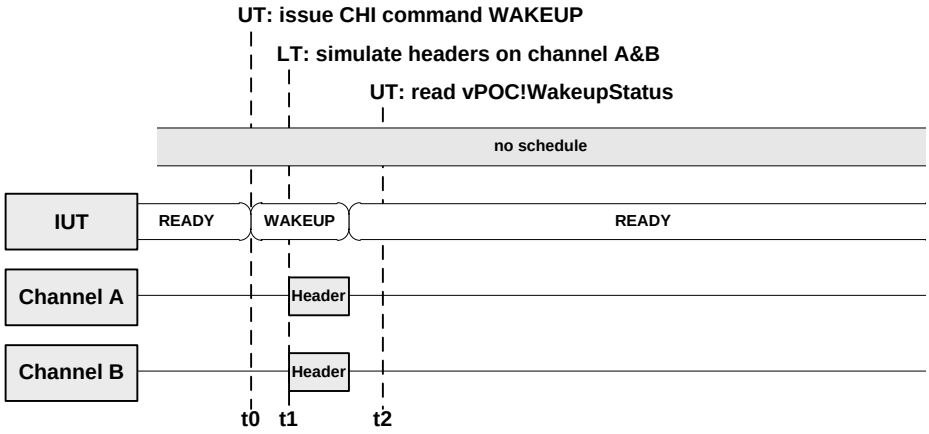


Figure 93 — Wakeup listen with header interference on both channels – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the wakeup process after reception of a frame header.

7.5.22 Wakeup listen with symbol interference on both channels

— Test purpose

Verify the correct wakeup behaviour after reception of a symbol on the wakeup channel and on the non-wakeup channel during wakeup listen phase.

— Applicability

DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0.

2) The LT simulates a valid symbol of length *gdTSSTransmitter* + *cdCAS* on both channels at t1 = t0 + 0,9 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) It is verified (LT) that the IUT starts to transmit a WUP on channel *pWakeupChannel* at t3 = t2 + (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* + ξ µT + (φ_{Rx} + φ_{Tx}) ns (with t2 = t1 + *gdTSSTransmitter* + *cdCAS*).

4) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* at t0 + 2 * *pdListenTimeout* + *pWakeupPattern* * (*gdWakeupTxIdle* + *gdWakeupTxActive*) * *gdBit* / *pdMicrotick* + 500 µT.

5) It is verified (LT) that the IUT does not transmit any communication elements on the non-wakeup channel during the test.

Figure 94 depicts the wakeup listen with symbol interference on both channels – *pWakeupChannel* = A.

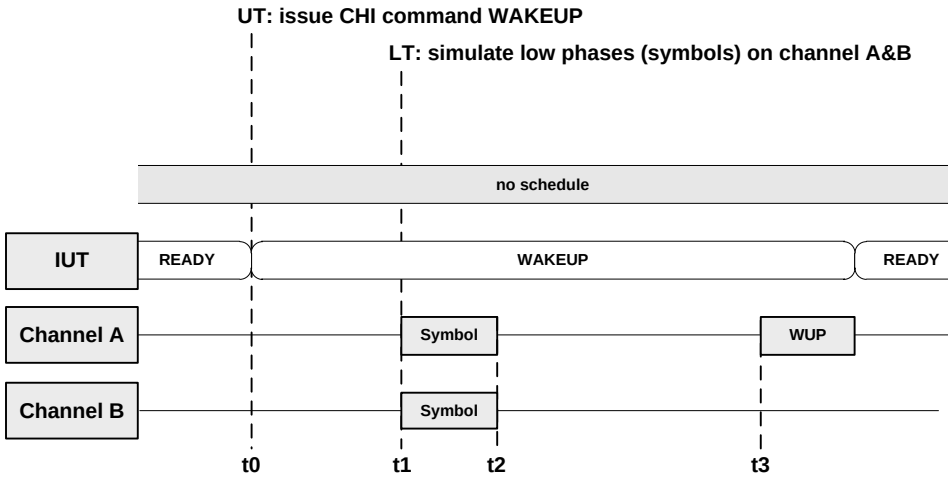


Figure 94 — Wakeup listen with symbol interference on both channels – pWakeupChannel = A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a correct WUP with the appropriate delay after the wakeup channel became idle again.

7.5.23 Wakeup listen with symbol interference (SC)

— **Test purpose**

Verify the interpretation of the listen time-out *pdListenTimeout* and noise multiplier *gListenNoise* for a wakeup with symbol interference. The IUT shall wait until *pdListenTimeout* * *gListenNoise* expires before it starts WUP transmission, if the channel is permanently disturbed after reception of a symbol.

— **Applicability**

SC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT simulates a valid symbol of length *gdTSSTransmitter* + *cdCAS* starting at t1 = t0 + 0,4 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT.

3) The LT sets the channel to low at t2 = t1 +0,4 * *pdListenTimeout* µT.

4) The LT sets the channel to idle at t2 + (*gListenNoise* – 0,5) * *pdListenTimeout* µT.

5) It is verified (LT) that the IUT starts to transmit a WUP at t3 = t1 + (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*gdTSSTransmitter* + *cdCAS*) * *gdBit* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* + ξ µT + (φ_{Rx} + φ_{Tx}) ns.

6) It is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and that *vPOC!WakeupStatus* = *TRANSMITTED* after the WUP has been transmitted at t3 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT.

Figure 95 depicts the wakeup listen with symbol interference (SC).

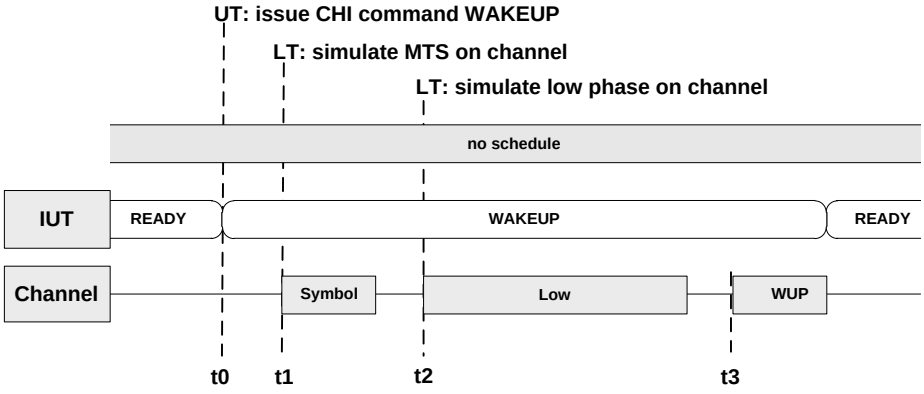


Figure 95 — Wakeup listen with symbol interference (SC)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits a correct WUP with the appropriate delay, if a channel has been permanently disturbed after reception of a valid symbol.

7.5.24 WUP with unknown collision during blind

— Test purpose

Verify the correct handling of an unspecified collision on the wakeup channel. The IUT shall detect collision depending on the configuration of *gdIgnoreAfterTx*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 272.

Table 272 — Modification to basic configurations for wakeup – WUP with unknown collision during blind

Parameter	Modification	
	I	II
gdIgnoreAfterTx [gdBit]	0	7

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT measures the start of the WUP sent by the IUT. The falling edge of the first WUS is at t1.

3) At t2 = t1 (at the beginning of the first low phase) of the IUT's WUP, the LT generates a low phase of length (*gdWakeupTxActive* + *cdWakeupMaxCollision* + 1 *gdBit*) on the wakeup channel *pWakeupChannel*.

4) It is verified (LT) that

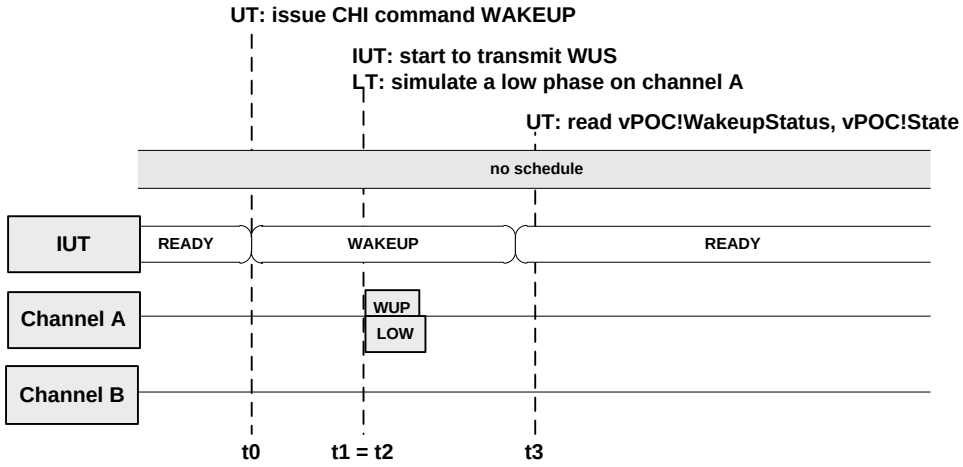
   (I) the IUT stops WUP generation, i.e. it is verified (LT) that the IUT does not generate any low phase for at least (*gListenNoise* + 1) * *pdListenTimeout* after the end of the 1st transmitted WUS (rising edge of TxD).

   (II) the IUT continues WUP generation and that the IUT transmits a correct WUP.

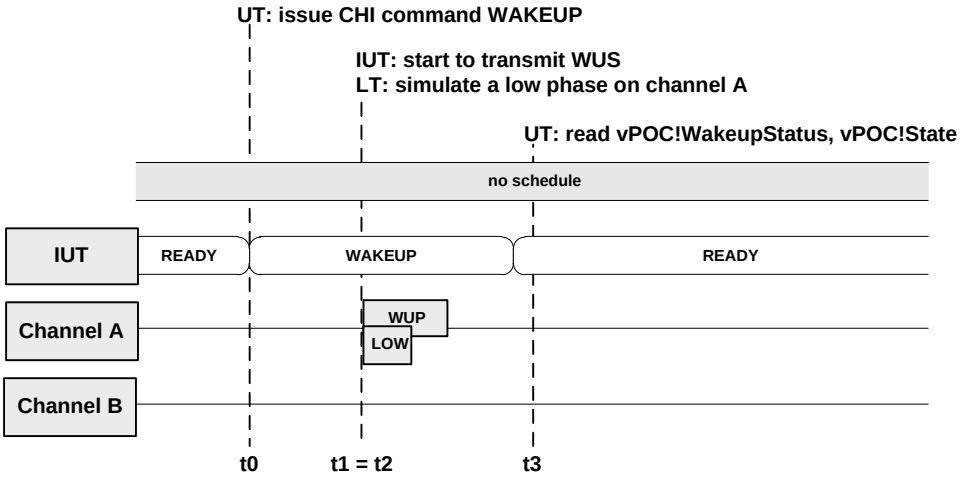
5) (I) At t3 = t2 + (*gdWakeupTxActive* + *cdWakeupMaxCollision*) * *gdBit* / *pdMicrotick* + *pdListenTimeout* µT + 500 µT, it is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *COLLISION_UNKNOWN*.

(II) At t3 = t1 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT, it is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *TRANSMITTED*.

Figure 96 depicts the WUP with unknown collision during blind.



a) Modification I



b) Modification II

Figure 96 — WUP with unknown collision during blind

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT recognizes or ignores the collision on the wakeup channel during WUP transmission, depending on the configuration parameter *gdIgnoreAfterTx*.

7.5.25 WUP with insignificant collision during blind

— Test purpose

Verify the correct handling of an insignificant collision during the encoding process. The IUT shall ignore the collision and shall transmit its WUP correctly. Verify that the bit synchronisation is performed correctly. The IUT shall not abort its wakeup transmission after reception of a low phase of length *cdWakeupMaxCollision* – 5 / 8 *gdBit*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 273.

Table 273 — Modification to basic configurations for wakeup – WUP with insignificant collision during blind

Parameter	Modification	
	I	II
gdIgnoreAfterTx [gdBit]	0	7

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT triggers the wakeup process with the CHI command WAKEUP at t0 (*vPOC!State* = *WAKEUP*).

2) The LT measures the start of the WUP sent by the IUT. The falling edge of the first WUS is at t1.

3) At t2 = t1 + *gdWakeupTxActive* * *gdBit* / *pdMicrotick* µT (at the beginning of the first idle phase) of the IUT's WUP, the LT generates a low phase of length *cdWakeupMaxCollision* – 5 / 8 *gdBit* on the wakeup channel *pWakeupChannel*.

4) It is verified (LT) that the IUT continues WUP generation and that the IUT transmits a correct WUP.

5) At t3 = t1 + *pWakeupPattern* * (*gdWakeupTxActive* + *gdWakeupTxIdle*) * *gdBit* / *pdMicrotick* + 500 µT, it is verified (UT) that the IUT is in state *POC:ready* (*vPOC!State* = *READY*) and *vPOC!WakeupStatus* = *TRANSMITTED*.

Figure 97 depicts the WUP with insignificant collision during blind.

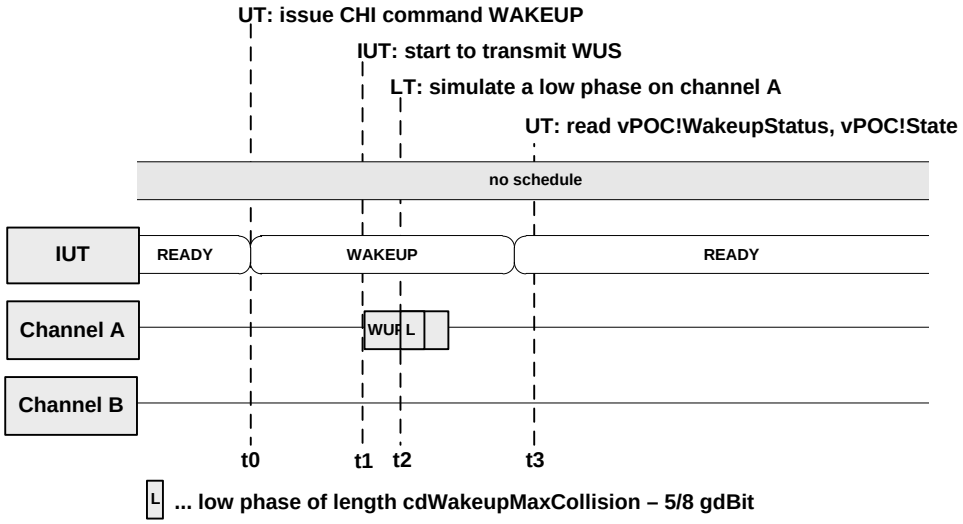


Figure 97 — WUP with insignificant collision during blind

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT ignores the insignificant collision on the wakeup channel and transmits its WUP correctly.

7.6 Startup

7.6.1 Coldstart listen

7.6.1.1 Channel idle

— **Test purpose**

Verify correct behaviour of the IUT when idle is received during *POC:coldstart listen* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) It is verified (LT) that the IUT sends a CAS (low phase of length *gdTSSTransmitter* + *cdCAS*) starting *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN. It is also verified (LT) that the TxEN output and the TxD output become low simultaneously, that TxEN becomes high after a low period of *gdTSSTransmitter* + *cdCAS* and that TxD becomes high after a low period of *gdTSSTransmitter* + *cdCAS* + *cdStaggerDelay*.

4) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

Figure 98 depicts the channel idle in *POC:coldstart listen*.

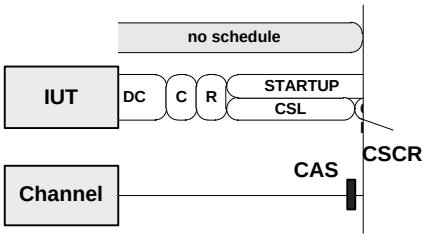


Figure 98 — Channel idle in POC:coldstart listen

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends CAS and passes through the *POC:coldstart listen* state and reaches *POC:coldstart collision resolution* state.

7.6.1.2 Permanent noise on single channel

— Test purpose

Verify correct behaviour of the IUT when permanent noise is received during *POC:coldstart listen* state on a single communication channel. Verify the correct behaviour of the coldstart noise flag.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 274.

Table 274 — Modification to basic configurations for coldstart listen – permanent noise on single channel

Parameter	Modification
gListenNoise	8

The IUT is configured as coldstarter sending startup frames in slot 1.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— Test execution

1) The LT starts to generate noise (low phase) (a) on channel A and (b) on channel B latest 2 000 – 2 * *cChannelIdleDelimiter* * *gdBit* / *pdMicrotick* µT after the CHI command ALLOW_COLDSTART.

2) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart noise flag is reset (*vPOC!ColdstartNoise* = false) 2 500 µT after the CHI command RUN.

4) The LT ends noise generation (a) on channel A and (b) on channel B (*gListenNoise* – 0,5) * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT after at the CHI command RUN.

5) It is verified (LT) that the IUT sends a CAS *gListenNoise* * *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

6) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION* after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

7) The LT simulates startup frames in slot 2 of cycles 4 to 6.

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

9) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

Figure 99 depicts the permanent noise on single channel in *POC:coldstart listen* – permanent noise on channel A.

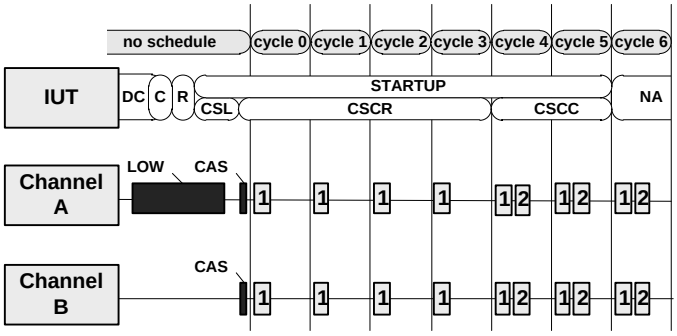


Figure 99 — Permanent noise on single channel in POC:coldstart listen – permanent noise on channel A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.3 Permanent and intermittent noise

— **Test purpose**

Verify correct behaviour of the IUT when noise is received during *POC:coldstart listen* state on both channels.

— **Applicability**

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 275.

Table 275 — Modification to basic configurations for coldstart listen – permanent and intermittent noise

Parameter	Modification
gListenNoise	8

The IUT is configured as coldstarter sending startup frames in slot 1 on both channels.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— Test execution

1) The LT starts to generate noise (low phase) (a) on channel A and (b) on channel B latest 2 000 – 2 * *cChannelIdleDelimiter* * *gdBit* / *pdMicrotick* µT after the CHI command ALLOW_COLDSTART.

2) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupStat*e = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

4) The LT generates noise (low phase of length 10 * *gdBit* starting 0,5 * *pdListenTimeout* after the CHI command RUN) (a) on channel B and (b) on channel A.

5) The LT ends noise generation (a) on channel A and (b) on channel B (*gListenNoise* – 0,5) * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT after at the CHI command RUN.

6) It is verified (LT) that the IUT sends a CAS *pdListenTimeout* * *gListenNoise* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

7) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

8) The LT simulates startup frames in slot 2 of cycles 4 to 6.

9) In cyle 5, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is not set (*vPOC!ColdstartNoise* = false).

10) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

11) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

Figure 100 depicts the permanent and intermittent noise in *POC:coldstart listen* – permanent noise on channel B.

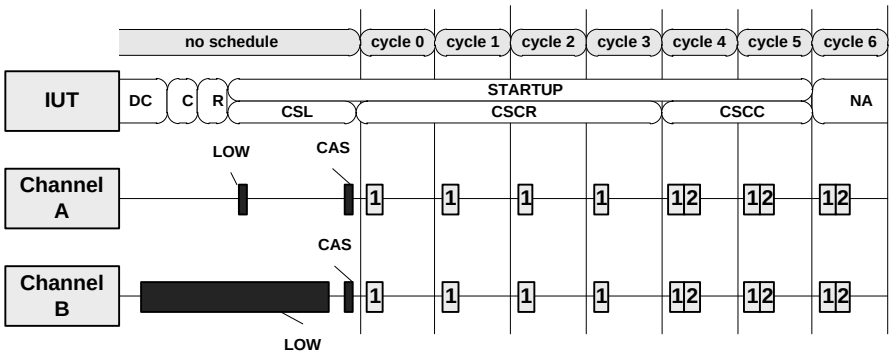


Figure 100 — Permanent and intermittent noise in POC:coldstart listen – permanent noise on channel B

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.4 Permanent noise on available channel(s) longer than tStartupNoise

— Test purpose

Verify correct behaviour of the IUT when noise is received during *POC:coldstart listen* state on all available channel(s) ending after the timer tStartupNoise expired.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 276.

Table 276 — Modification to basic configurations for coldstart listen – permanent noise on available channel(s) longer than tStartupNoise

Parameter	Modification
gListenNoise	8

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT generates noise (low phase of length ceil(*pdListenTimeout* * (*gListenNoise* + 1) * *pSamplesPerMicrotick* / *cSamplesPerBit*) * *cSamplesPerBit* / *pSamplesPerMicrotick* µT during the *POC:coldstart listen* state) on all available channel(s) starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN.

4) It is verified (LT) that the IUT sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter*-1) * *gdBit* / *pdMicrotick* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the LT has stopped noise generation.

5) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

6) The LT simulates startup frames in slot 2 of cycles 4 to 6.

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

9) In slot 3 of cycle 7, the UT issues the CHI command IMMEDIATE_READY.

10) It is verified (UT) that the IUT is in the *POC:ready* state (*vPOC!State* = *READY*) and that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true) 2 500 µT after the CHI command IMMEDIATE_READY.

11) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

12) The UT initiates the startup procedure with the CHI command RUN.

13) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart noise flag is reset (*vPOC!ColdstartNoise* = false) 2 500 µT after the CHI command RUN.

14) It is verified (LT) that the IUT sends a CAS *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

15) The LT simulates startup frames in slot 2 of cycles 4 to 6 during the second startup.

16) In cycle 6 of the second startup, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

17) In cycle 6 of the second startup , 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is reset (*vPOC!ColdstartNoise* = false).

Figure 101 depicts the permanent noise on available channel(s) longer than *tStartupNoise*.

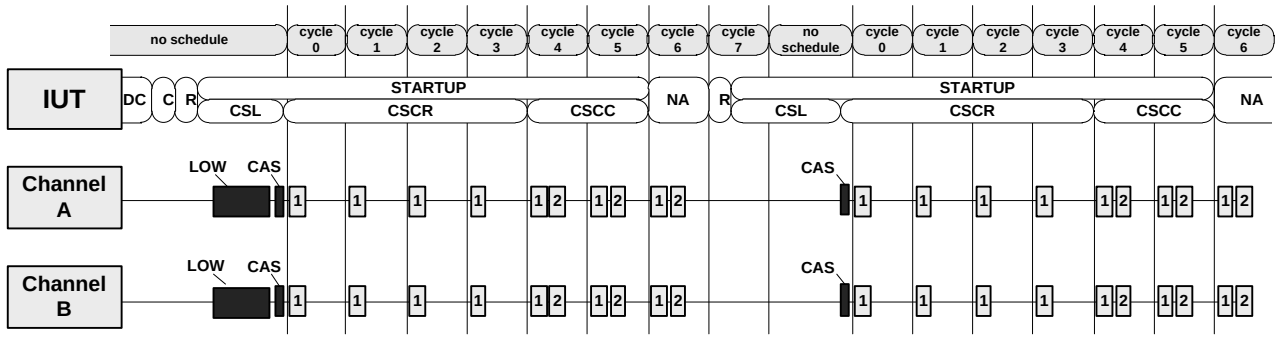


Figure 101 — Permanent noise on available channel(s) longer than tStartupNoise

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.5 Permanent noise on available channel(s) shorter than tStartupNoise

— Test purpose

Verify correct behaviour of the IUT when noise is received during *POC:coldstart listen* state on all available channel(s) ending before the timer tStartupNoise expires.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 277.

Table 277 — Modification to basic configurations for coldstart listen – permanent noise on available channel(s) shorter than tStartupNoise

Parameter	Modification		
	I	II	III
gListenNoise	2	8	16

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— Test execution

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) The LT generates noise (low phase of length *pdListenTimeout* * (*gListenNoise* – 1) during the *POC:coldstart listen* state) on all available channel(s) 2 000 µT after the CHI command RUN.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

4) It is verified (LT) that the IUT sends a CAS *pdListenTimeout* * *gListenNoise* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

5) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

6) The LT simulates startup frames in slot 2 of cycles 4 to 6.

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

Figure 102 depicts the permanent noise on available channel(s) shorter than *tStartupNoise*.

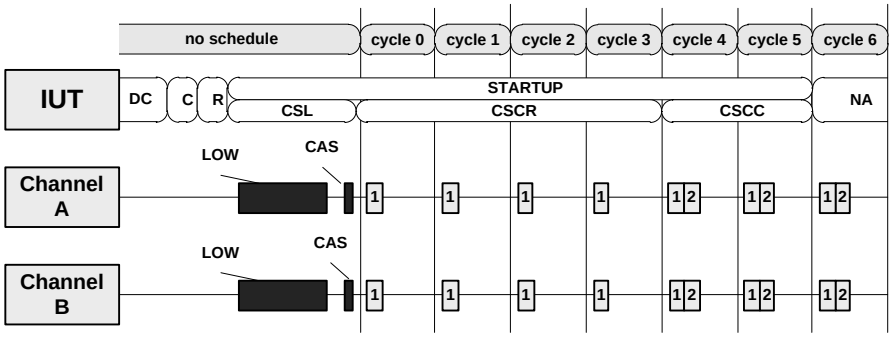


Figure 102 — Permanent noise on available channel(s) shorter than tStartupNoise

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.6 CE start

— **Test purpose**

Verify correct behaviour of the IUT when CE start is recognized during *POC:coldstart listen* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 278.

Table 278 — Modification to basic configurations coldstart listen – for CE start

Parameter	Modification
gListenNoise	8

The IUT is configured as coldstarter sending startup frames in slot 1 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates a CE start for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT generates a CE start (low phase of length 10 * *gdBit* during the *POC:coldstart listen* state) on the selected channel before *pdListenTimeout* / 2 expires.

4) It is verified (LT) that the IUT sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *pdListenTimeout* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the end of the noise.

5) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

6) The LT simulates startup frames in slot 2 of cycles 4 to 6.

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is reset (*vPOC!ColdstartNoise* = false).

Figure 103 depicts the CE start in *POC:coldstart listen* – low phase on channel A.

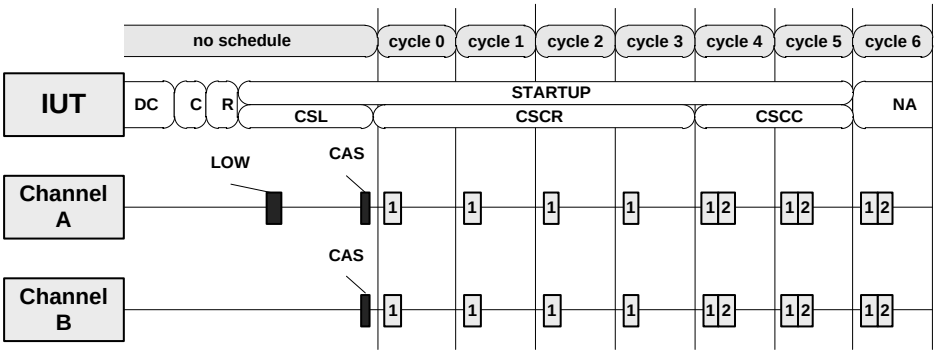


Figure 103 — CE start in POC:coldstart listen – low phase on channel A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.7 Header reception (SC)

— **Test purpose**

Verify correct behaviour of the IUT when a header is received during *POC:coldstart listen* state.

— **Applicability**

SC.

— **Configuration**

All basic configurations using the modifications as listed in Table 279.

Table 279 — Modification to basic configurations for coldstart listen – header reception (SC)

Parameter	Modification
gListenNoise	8

The IUT is configured as coldstarter sending startup frames in slot 1.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

1) The UT initiates the startup procedure at t0 with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT generates a header of a sync frame in slot 2 (null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the payload length is 1, the cycle count 0) starting at t1 = t0 + (0,5 ± 0,05) * *pdListenTimeout*, followed by

(a) a low phase of length (*gListenNoise* – 1) * *pdListenTimeout* starting at t2 = t0 + (*0,8* ± 0,05) * *pdListenTimeout*.

(b) a low phase of length *gListenNoise* * *pdListenTimeout* starting at t2 = t0 + (*0,8* ± 0,05) * *pdListenTimeout*.

The LT has to simulate the header plus the BSS preceding the frame's payload section in order to generate a valid header. The length of the frame header is (*gdTSSTransmitter* + *cdFSS* +*cdBSS* + 5 * 10) * *gdBit* / *pdMicrotick* µT.

4) It is verified (LT) that the IUT

(a) sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the falling edge of the LT's final BSS.

(b) sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the rising edge of the low phase.

5) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

6) The LT simulates startup frames in slot 2 of cycles 4 to 6.

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

Figure 104 depicts the header reception (SC) in *POC:coldstart listen*.

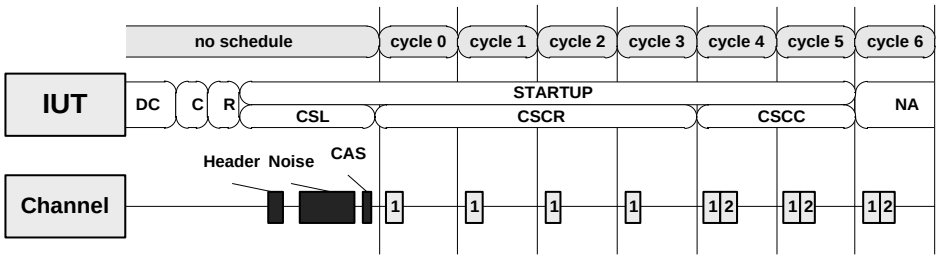


Figure 104 — Header reception (SC) in POC:coldstart listen

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends CAS and passes through the POC:coldstart collision resolution state. The coldstart noise flag is set accordingly.

7.6.1.8 CAS reception (SC)

— **Test purpose**

Verify correct behaviour of the IUT when CAS is received during *POC:coldstart listen* state.

— **Applicability**

SC.

— **Configuration**

All basic configurations using the modifications as listed in Table 280.

Table 280 — Modification to basic configurations for coldstart listen – CAS reception (SC)

Parameter	Modification
gListenNoise	8

The IUT is configured as coldstarter sending startup frames in slot 1.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

1) The UT initiates the startup procedure at t0 with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT generates a CAS starting at t1 = t0 + (0,5 ± 0,05) * *pdListenTimeout* followed by

   (a) a low phase of length (*gListenNoise* – 1) * *pdListenTimeout* starting t2 = t0 + (0,8 ± 0,05) * *pdListenTimeout*.

   (b) a low phase of length *gListenNoise* * *pdListenTimeout* starting at t2 = t0 + (0,8 ± 0,05) * *pdListenTimeout*.

4) It is verified (LT) that the IUT

   (a) sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the end of LT's CAS on the selected channel.

   (b) sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the rising edge of the low phase.

5) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

6) The LT simulates startup frames in slot 2 of cycles 4 to 6.

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

Figure 105 depicts the CAS reception (SC) in *POC:coldstart listen*.

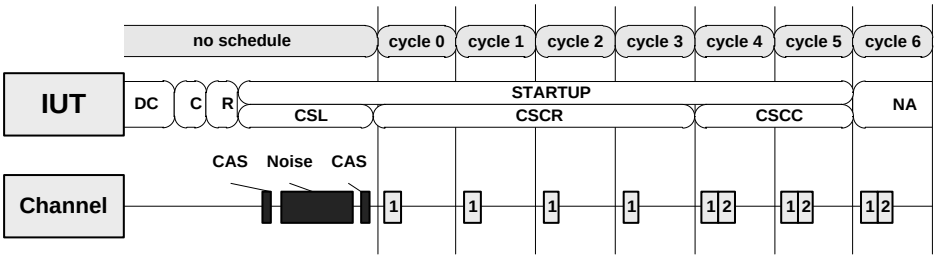


Figure 105 — CAS reception (SC) in POC:coldstart listen

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.9 Startup frame reception

— Test purpose

Verify correct behaviour of the IUT when startup frame is received during *POC:coldstart listen* state. Noise and CAS before the startup frame shall not disturb the reception of the startup frame.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 281.

Table 281 — Modification to basic configurations for coldstart listen – startup frame reception

Parameter	Modification
pKeySlotID	2

The IUT is configured to send startup frames in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— Test execution

The LT simulates the leading coldstarter for this test.

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT simulates

   (a) idle.

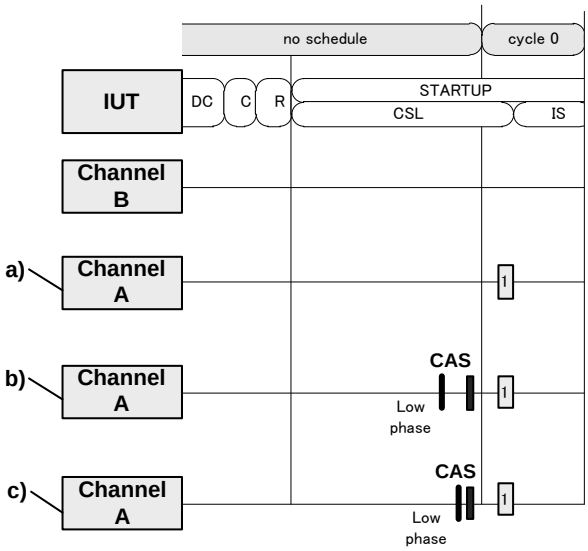
   (b) low phase of length 1 * *gdBit*, followed by a high phase of length 4 * *gdBit*, followed by a CAS, starting 0,4 * *pdListenTimeout*±0,05 * *pdListenTimeout* after the CHI command RUN.

(c) low phase of length 1 * *gdBit*, followed by a high phase of length 1 * *gdBit*, followed by a CAS, starting 0,4 * *pdListenTimeout*±0,05 * *pdListenTimeout* after the CHI command RUN.

4) The LT simulates a startup frame with frame ID set to 1 and cycle count set to 0 during the *POC:coldstart listen* state starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* µT after the LT has applied the frame in previous test step.

Figure 106 depicts the startup frame reception in *POC:coldstart listen* – startup frame on channel A.



a) see test execution 3) (a)
b) see test execution 3) (b)
c) see test execution 3) (c)

Figure 106 — Startup frame reception in POC:coldstart listen – startup frame on channel A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters *POC:initialize schedule* state after reception of a valid even startup frame while in *POC:coldstart listen* state.

7.6.1.10 Header reception (DC)

— Test purpose

Verify correct behaviour of the IUT when a header is received during *POC:coldstart listen* state.

— Applicability

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 282.

Table 282 — Modification to basic configurations for coldstart listen – header reception (DC)

Parameter	Modification
gListenNoise	8

The IUT is configured as coldstarter sending startup frames in slot 1.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

Differing from the general dual channel execution, channel A and channel B have to be stimulated unequally. The test has to be performed in two instances. For instance 1 the selected channel is A and for instance 2 the selected channel is B.

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

   (a) The LT generates a header of a sync frame with the frame ID 2 (null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the payload length is 1, the cycle count 0) on the selected channel and a low phase of length *gListenNoise* * *pdListenTimeout* on the other channel starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN.

   (b) The LT generates noise (low phase of length *cdCAS*) starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN followed by a header of a sync frame in slot 2 (null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the payload length is 1, the cycle count 0) starting *gListenNoise* * *pdListenTimeout* – 0,5 * (*gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick* ± 0,45 * (*gdTSSTransmitter* + 53) * *gdBit* / *pdMicrotick* after the noise on the selected channel and a low phase of length 2 * *gListenNoise* * *pdListenTimeout* starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN on the other channel.

   The LT has to simulate the header plus the BSS preceding the frame's payload section in order to generate a valid header. The length of the frame header is (*gdTSSTransmitter* + 1 (FSS) + 50 (5 header bytes including BSS) + 2 (BSS preceding the frame's payload section)) * *gdBit* / *pdMicrotick* µT.

3) It is verified (LT) that the IUT sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the falling edge of the LT's final BSS.

4) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

5) The LT simulates startup frames in slot 2 of cycles 4 to 6.

6) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

Figure 107 depicts the header reception (DC) in *POC:coldstart listen* – frame header on channel A.

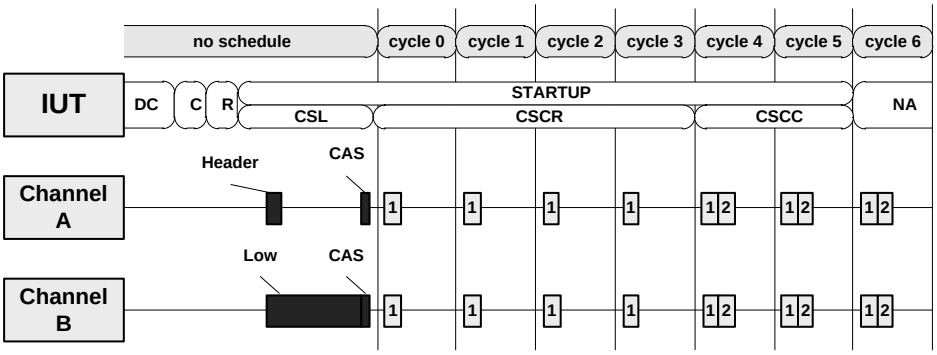


Figure 107 — Header reception (DC) in POC:coldstart listen – frame header on channel A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.11 CAS reception (DC)

— Test purpose

Verify correct behaviour of the IUT when CAS is received during *POC:coldstart listen* state.

— Applicability

DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 283.

Table 283 — Modification to basic configurations for coldstart listen – CAS reception (DC)

Parameter	Modification
gListenNoise	8

The IUT is configured as coldstarter sending startup frames in slot 1.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

Differing from the general dual channel execution, channel A and channel B have to be stimulated unequally. The test has to be performed in two instances. For instance 1 the selected channel is A and for instance 2 the selected channel is B.

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT generates:

   (a) a CAS on the selected channel and a low phase of length *gListenNoise* * *pdListenTimeout* on the other channel starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN and

   (b) noise (low phase of length *cdCAS*) starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN followed by a CAS starting *gListenNoise* * *pdListenTimeout* – 0,5 * (*gdTSSTransmitter* + *cdCAS*) * *gdBit* / *pdMicrotick* ± 0,45 * (*gdTSSTransmitter* + *cdCAS*) * *gdBit* / *pdMicrotick* after the noise on the selected channel and a low phase of length 2 * *gListenNoise* * *pdListenTimeout* starting 0,5 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN on the other channel.

4) It is verified (LT) that the IUT sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *gListenNoise* * *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the end of LT's CAS on the selected channel.

5) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

6) The LT simulates startup frames in slot 2 of cycles 4 to 6.

7) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the coldstart noise flag is set (*vPOC!ColdstartNoise* = true).

Figure 108 depicts the CAS reception (DC) in *POC:coldstart listen* – LT's CAS on channel A.

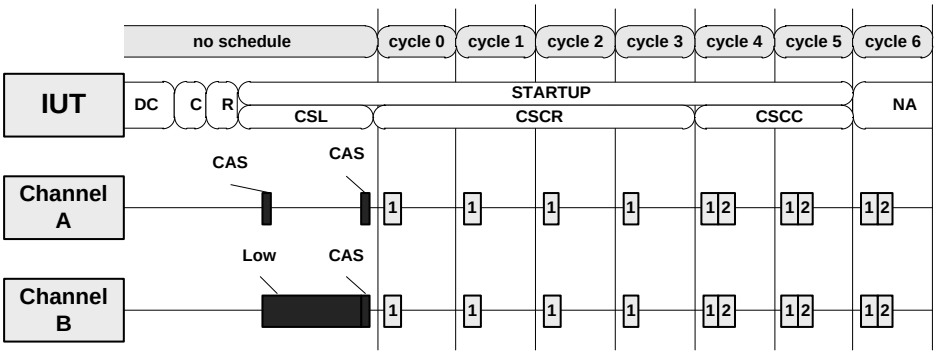


Figure 108 — CAS reception (DC) in POC:coldstart listen – LT's CAS on channel A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and passes through the *POC:coldstart collision resolution* state. The coldstart noise flag is set accordingly.

7.6.1.12 Late integration into running cluster: following coldstarter

— Test purpose

Verify correct behaviour of the IUT as following coldstarter when startup frames are received during *POC:coldstart listen* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 284.

Table 284 — Modification to basic configurations for coldstart listen – late integration into running cluster: following coldstarter

Parameter	Modification
pKeySlotID	2

The IUT is configured to send startup frames in slot 2 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000µT after the CHI command CONFIG_COMPLETE.

— **Test execution**

The LT simulates a running cluster.

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) 2 000µT after CHI command ALLOW_COLDSTART, the LT simulates startup frames in slots 1 and 3 of the cycle 0 to 8.

2) The UT initiates the startup procedure with the CHI command RUN:

   (a) In cycle 7 after the startup frame in slot 3.

   (b) In cycle 6 after the startup frame in slot 3

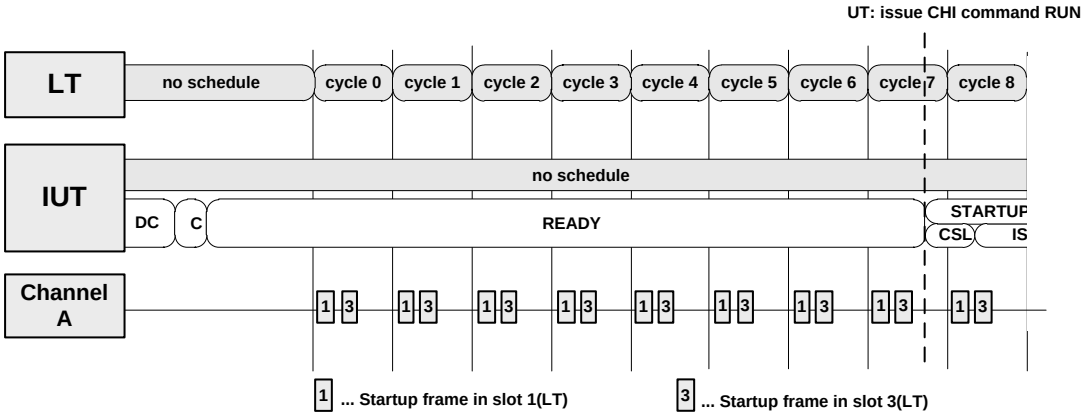
3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*):

   (a) 2 500 µT after the CHI command RUN.

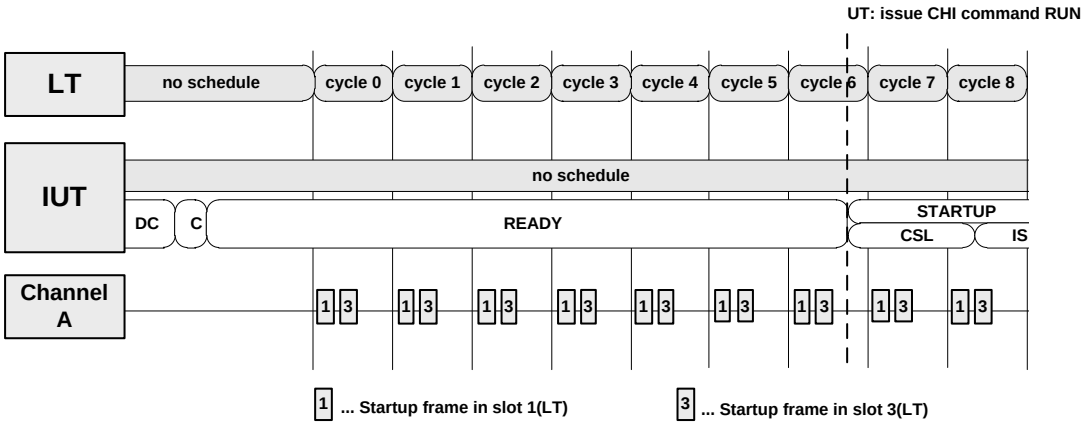
   (b) 2 500 µT after the CHI command RUN and *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* µT after the startup frame in slot 3 in cycle 7.

4) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 500 µT after the start of slot 2 and before slot 3 in cycle 8.

Figure 109 depicts the late integration into running cluster following coldstarter – startup frames on channel A.



a) Following coldstarter in cycle 7



b) Following coldstarter in cycle 6

Figure 109 — Late integration into running cluster following coldstarter – startup frames on channel A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters *POC:initialize schedule* state after reception of a valid even startup frame while in *POC:coldstart listen* state.

7.6.1.13 Late integration into running cluster: integrating node

— Test purpose

Verify correct behaviour of the IUT as integrating node when startup frames are received during *POC:integration listen* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 285.

Table 285 — Modification to basic configurations for coldstart listen – late integration into running cluster: integrating node

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false
pKeySlotID	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

The LT simulates a running cluster.

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) 2 000µT after CHI command CONFIG_COMPLETE, the LT simulates startup frames in slots 1 and 3 of the cycle 0 to 8.

2) The UT initiates the startup procedure with the CHI command RUN:

   (a) In cycle 7 after the startup frame in slot 3.

   (b) In cycle 6 after the startup frame in slot 3

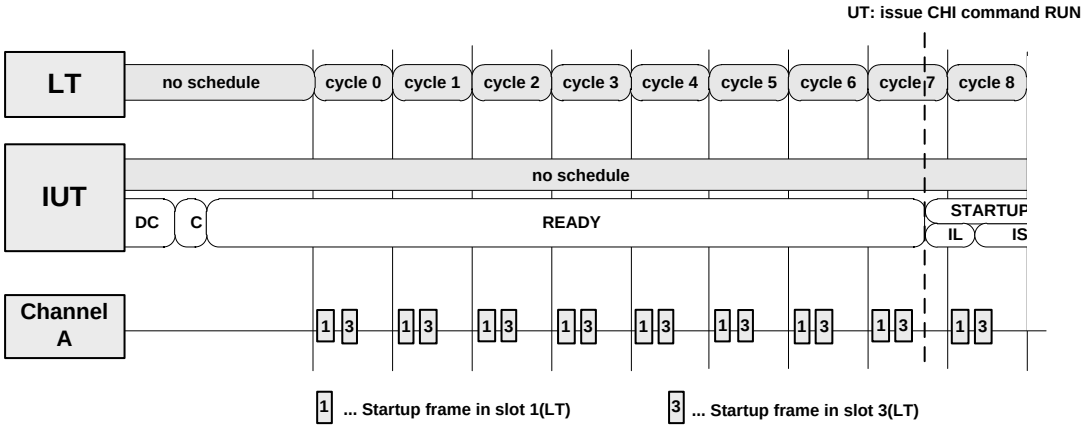
3) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*):

   (a) 2 500 µT after the CHI command RUN

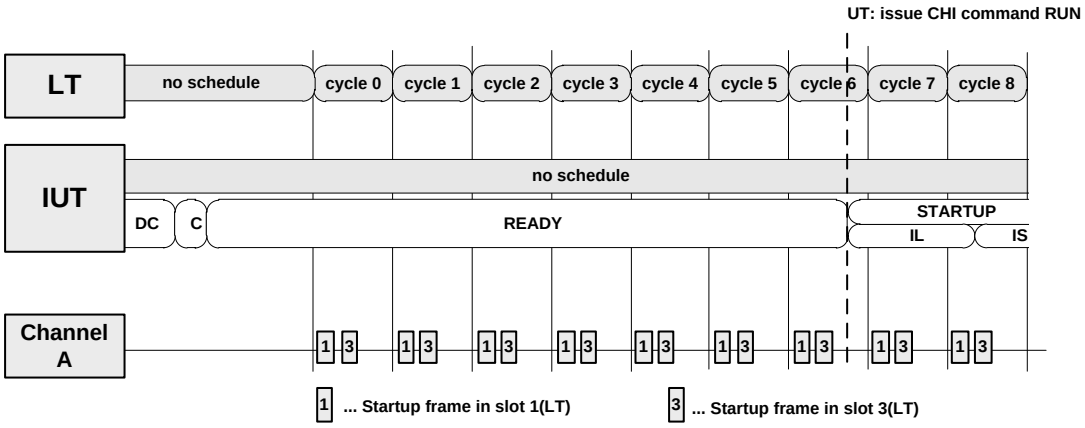
(b) 2 500 µT after the CHI command RUN and *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* µT after the startup frame in slot 3 in cycle 7.

4) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 500 µT after the start of slot 2 and before slot 3 in cycle 8.

Figure 110 depicts the late integration into running cluster integrating node – startup frames on channel A.



a) Integrating node in cycle 7



b) Integrating node in cycle 6

Figure 110 — Late integration into running cluster integrating node – startup frames on channel A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters *POC:initialize schedule* state after reception of a valid even startup frame while in *POC:integration listen* state.

7.6.1.14 Late integration into running cluster: WUDOP reception

— Test purpose

Verify correct behaviour of the IUT when startup frames are received during *POC:coldstart listen* state and a WUDOP is received in the symbol window of the first cycle when the IUT is in *POC:initialize schedule state*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 286.

Table 286 — Modification to basic configurations for coldstart listen – late integration into running cluster: WUDOP reception

Parameter	Modification
pKeySlotID	2

The IUT is configured to send startup frames in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000µT after the CHI command CONFIG_COMPLETE.

— Test execution

The LT simulates a leading coldstarter for this test.

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS 2 000 µT after CHI command ALLOW_COLDSTART.

2) The LT simulates startup frames in slot 1 of the cycle 0 to 8, and in slot 3 of the cycle 4 to 8.

3) The UT initiates the startup procedure with the CHI command RUN in cycle 6 after the LT has applied the startup frame in slot 3.

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN and *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* µT after the LT has applied the startup frame in slot 3 in cycle 7.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 500 µT after the start of slot 2 and before slot 3 in cycle 8.

6) The LT simulates a WUDOP in the symbol window of cycle 8, starting at the symbol window action point.

7) It is verified (UT) that the wakeup pattern received indicator in the wakeup and startup status of the respective channel(s) is set 500 µT after the LT has finished transmitting the WUDOP.

Figure 111 depicts the late integration into running cluster reception – WUDOP during *POC:initialize schedule*.

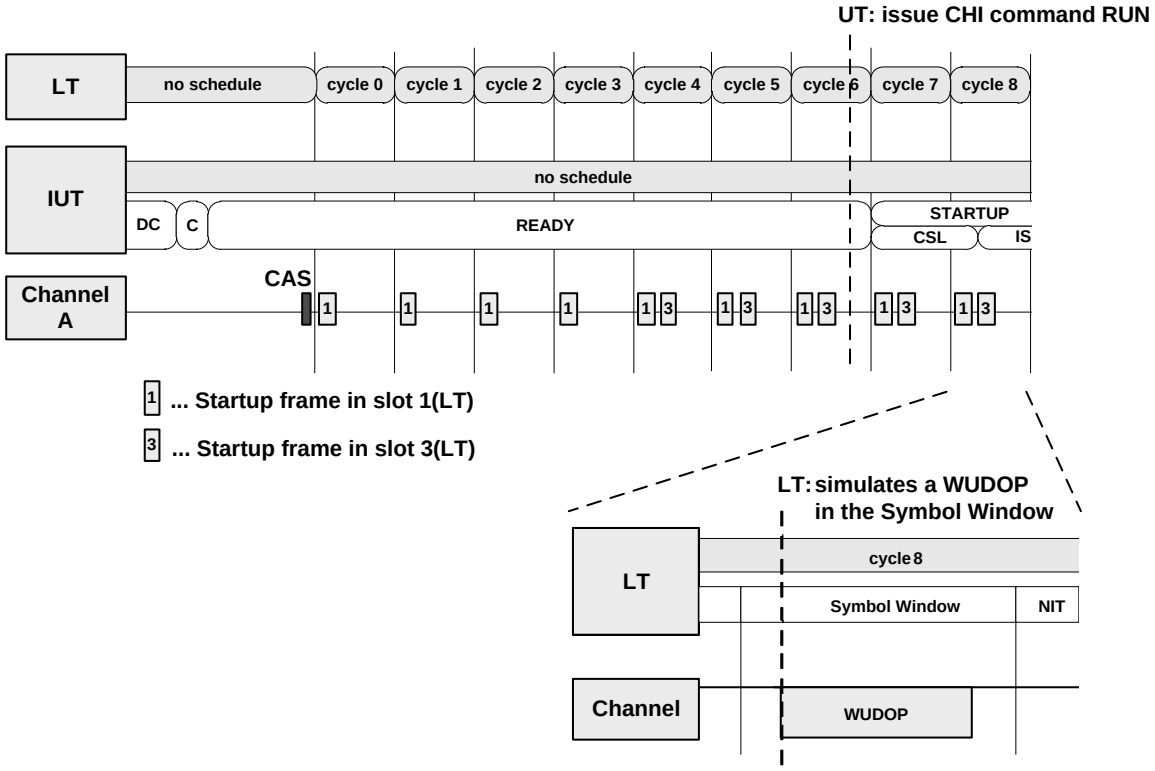


Figure 111 — Late integration into running cluster reception – WUDOP during POC:initialize schedule

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters *POC:initialize schedule* state after reception of a valid even startup frame while in *POC:coldstart listen* state, and sets the wakeup pattern received indicator in the wakeup and startup status after reception of a WUDOP.

7.6.1.15 Late integration into running cluster: WUDOP reception and command IMMEDIATE_READY

— Test purpose

Verify correct behaviour of the IUT when startup frames are received during *POC:coldstart listen* state, a WUDOP is received in the symbol window of the first cycle when the IUT is in *POC:initialize schedule* and shortly before the WUDOP, the CHI command IMMEDIATE_READY is initiated.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 287.

Table 287 — Modification to basic configurations for coldstart listen – late integration into running cluster: WUDOP reception and command IMMEDIATE_READY

Parameter	Modification
pKeySlotID	2

The IUT is configured to send startup frames in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000µT after the CHI command CONFIG_COMPLETE.

— Test execution

The LT simulates a leading coldstarter for this test.

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS 2 000 µT after CHI command ALLOW_COLDSTART.

2) The LT simulates startup frames in slot 1 of the cycle 0 to 8, and in slot 3 of the cycle 4 to 8.

3) The UT initiates the startup procedure with the CHI command RUN in cycle 6 after the LT has applied the startup frame in slot 3.

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN and *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* µT after the LT has applied the startup frame in slot 3 in cycle 7.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 500 µT after the start of slot 2 and before slot 3 in cycle 8.

6) The UT issues the CHI command IMMEDIATE_READY, 1500 µT before the symbol window action point of cycle 8.

7) The LT simulates a WUDOP in the symbol window of cycle 8, starting at the symbol window action point.

8) 2 500 µT after the CHI command IMMEDIATE_READY, it is verified (UT) that the IUT has entered the *POC:ready state* (*vPOC!State* = *READY* and *vPOC!StartupState* = *UNDEFINED*), that *vPOC!CHIHaltRequest* = false, *vPOC!CHIReadyRequest* = false and that the slot mode is *vPOC!SlotMode* = *ALL*.

9) It is verified (UT) that the wakeup pattern received indicator in the wakeup and startup status of the respective channel(s) is set 500 µT after the LT has finished transmitting the WUDOP.

Figure 112 depicts the late integration into running cluster WUDOP reception and command IMMEDIATE_READY.

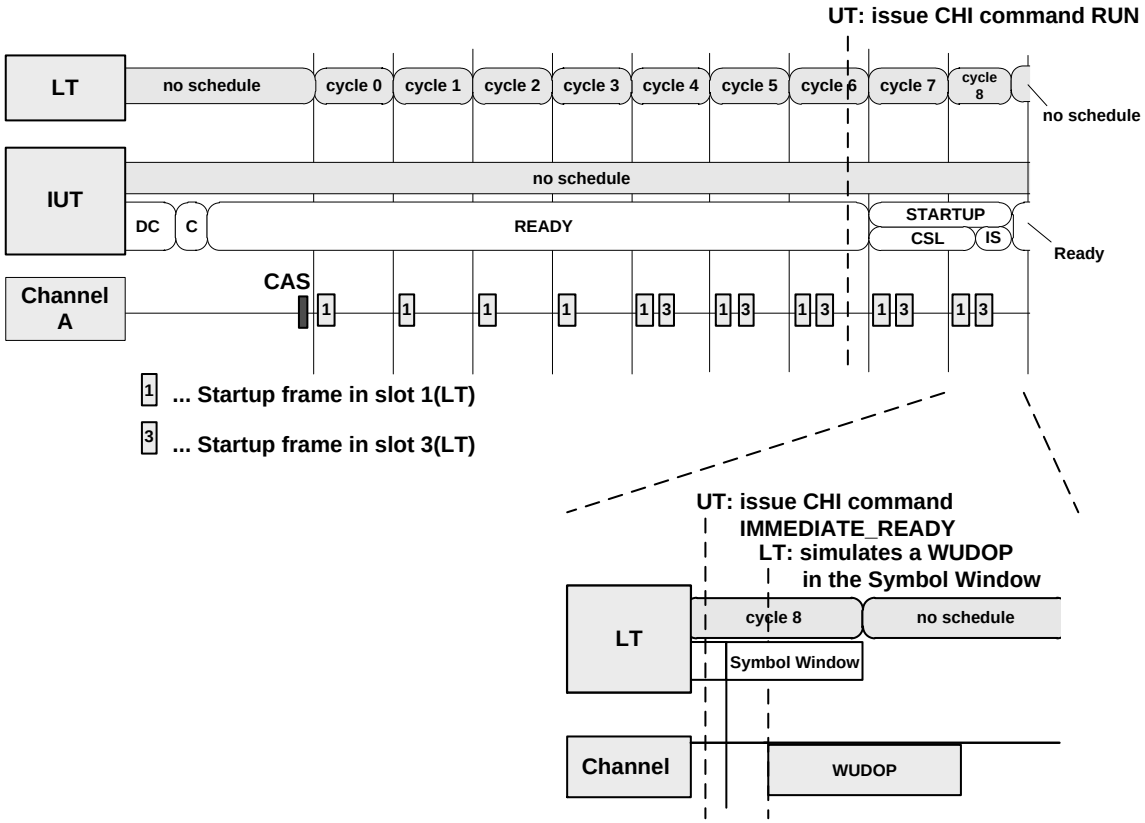


Figure 112 — Late integration into running cluster WUDOP reception and command IMMEDIATE_READY

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters *POC:initialize schedule* state after reception of a valid even startup frame while in *POC:coldstart listen* state, and sets the wakeup pattern received indicator in the wakeup and startup status after reception of a WUDOP.

7.6.2 Coldstart collision resolution

7.6.2.1 Channel idle

— **Test purpose**

Verify correct behaviour of the IUT when idle is received during *POC:coldstart collision resolution* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step.

2) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 6.

3) In cycle 4, 500 µT after cycle start and before cycle end, and in cycle 5, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

4) The LT simulates startup frames in slot 2 of cycles 4 to 6.

5) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 113 depicts the channel idle in *POC:coldstart collision resolution*.

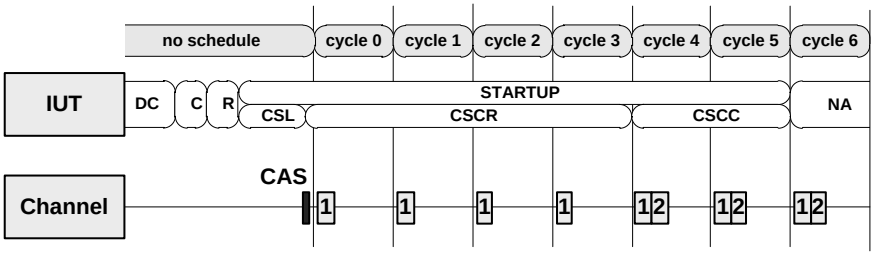


Figure 113 — Channel idle in POC:coldstart collision resolution

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and startup frames accordingly and enters the correct states.

7.6.2.2 Header reception

— Test purpose

Verify correct behaviour of the IUT when a header is received during *POC:coldstart collision resolution* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 288.

Table 288 — Modification to basic configurations for coldstart collision resolution – header reception

Parameter	Modification
pKeySlotID	2
gColdstartAttempts	3

The IUT is configured as coldstarter sending startup frames in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

— **Test execution**

The LT sends a header before the end of cycle n = 0, 1, 2 and 3. For dual channel test execution the test has to be performed in three instances. For dual channel test execution, the LT generates the header for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the IUT has started CAS transmission and that the *vRemainingColdstartAttempts* = 2.

2) It is verified (LT) that the IUT sends a startup frame in slot 2 of cycles 0 to n.

3) The LT simulates a header of a sync frame with frame ID set to 1 (null frame indicator and sync frame indicator are set to '1', reserved bit, startup frame indicator and payload preamble indicator are set to '0', the payload length is 1, the cycle count 0) starting 2 * *gdStaticSlot* after the IUT's frame in cycle n on the selected channel. The LT has to simulate the header plus the BSS preceding the frame's payload section in order to generate a valid header.

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart abort indicator is true after (*gdTSSTransmitter* + *cdFSS* + *cdBSS* + 5 * 10) * *gdBit* / *pdMicrotick* + 500 µT after the LT has started the header transmission in the previous step.

5) It is verified (LT) that the IUT sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the LT's final BSS.

6) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step and that the *vRemainingColdstartAttempts* = 1.

7) It is verified (LT) that the IUT sends startup frames in slot 2 of cycles 0 to 6.

8) In cycle 4, 500 µT after cycle start and before cycle end, and in cycle 5, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

9) The LT simulates startup frames in slot 1 of cycles 4 to 6.

10) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 114 depicts the header reception in *POC:coldstart collision resolution* – header on channel A and n = 0.

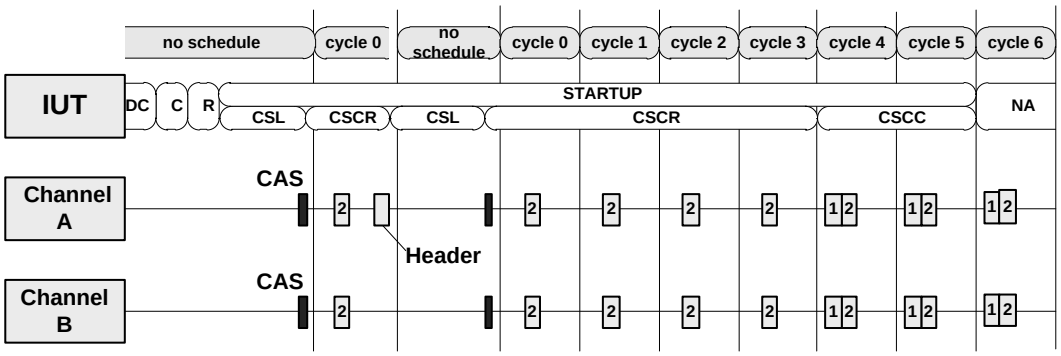


Figure 114 — Header reception in POC:coldstart collision resolution – header on channel A and n = 0

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and startup frames accordingly and enters the correct states. The IUT aborts the first startup attempt when it receives a frame header while in *POC:coldstart collision resolution* state.

7.6.2.3 CAS reception

— Test purpose

Verify correct behaviour of the IUT when a CAS is received during *POC:coldstart collision resolution* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 289.

Table 289 — Modification to basic configurations for coldstart collision resolution – CAS reception

Parameter	Modification
pKeySlotID	2
gColdstartAttempts	3

The IUT is configured as coldstarter sending startup frames in slot 2 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

— **Test execution**

The LT sends a CAS before the end of cycle n = 0, 1, 2 and 3. For dual channel test execution the test has to be performed in three instances. For dual channel test execution the LT generates the CAS for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISSION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the IUT has started CAS transmission and that the *vRemainingColdstartAttempts* = 2.

2) It is verified (LT) that the IUT sends a startup frame in slot 2 of cycles 0 to n.

3) The LT simulates a CAS starting 2 * *gdStaticSlot* after the IUT's frame in cycle n on the selected channel(s).

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart abort indicator is true 500 µT after the LT has finished the CAS transmission in the previous step.

5) It is verified (LT) that the IUT sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after end of the LT's CAS.

6) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step and that the *vRemainingColdstartAttempts* = 1.

7) It is verified (LT) that the IUT sends startup frames in slot 2 of cycles 0 to 6.

8) In cycle 4, 500 µT after cycle start and before cycle end, and in cycle 5, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

9) The LT simulates startup frames in slot 1 of cycles 4 to 6.

10) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 115 depicts the CAS reception in *POC:coldstart collision resolution* – CAS on channel A and n = 0.

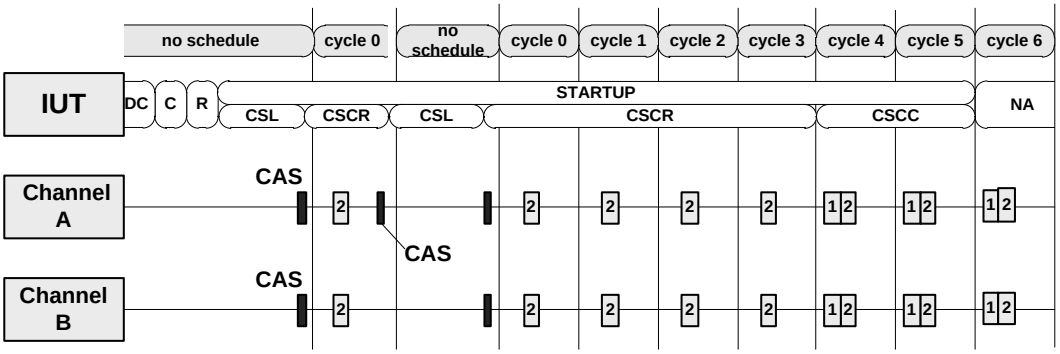


Figure 115 — CAS reception in POC:coldstart collision resolution – CAS on channel A and n = 0

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and startup frames accordingly and enters the correct states. The IUT aborts the first startup attempt when it receives a CAS while in *POC:coldstart collision resolution* state.

7.6.2.4 Parallel startup (CAS collision): IUT as leading coldstarter

— Test purpose

Verify correct behaviour of the IUT as leading coldstarter when a CAS collision occurs at the beginning.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 290.

Table 290 — Modification to basic configurations for coldstart collision resolution – parallel startup (CAS collision): IUT as leading coldstarter

Parameter	Modification
pKeySlotID	2
gColdstartAttempts	3

The IUT is configured as coldstarter sending startup frames in slot 2 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS. (low phase of length *gdTSSTransmitter* + *cdCAS*) starting *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

— **Test execution**

1) The LT simulates a CAS, starting at (2 ± 2) * *gdBit* / *pdMicrotick* µT after the falling edge of that CAS sent by the IUT.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 – *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step and that the *vRemainingColdstartAttempts* = 2.

3) It is verified (LT) that the IUT sends a startup frame in slot 2 of cycles 0 to 6.

4) The LT simulates startup frames in slot 4 of cycles 4 to 6.

5) In cycle 4, 500 µT after cycle start and before cycle end, and in cycle 5, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

6) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 116 depicts the parallel startup (CAS collision) IUT as leading coldstarter – Δt = (2 ± 2) * gdBit / pdMicrotick µT.

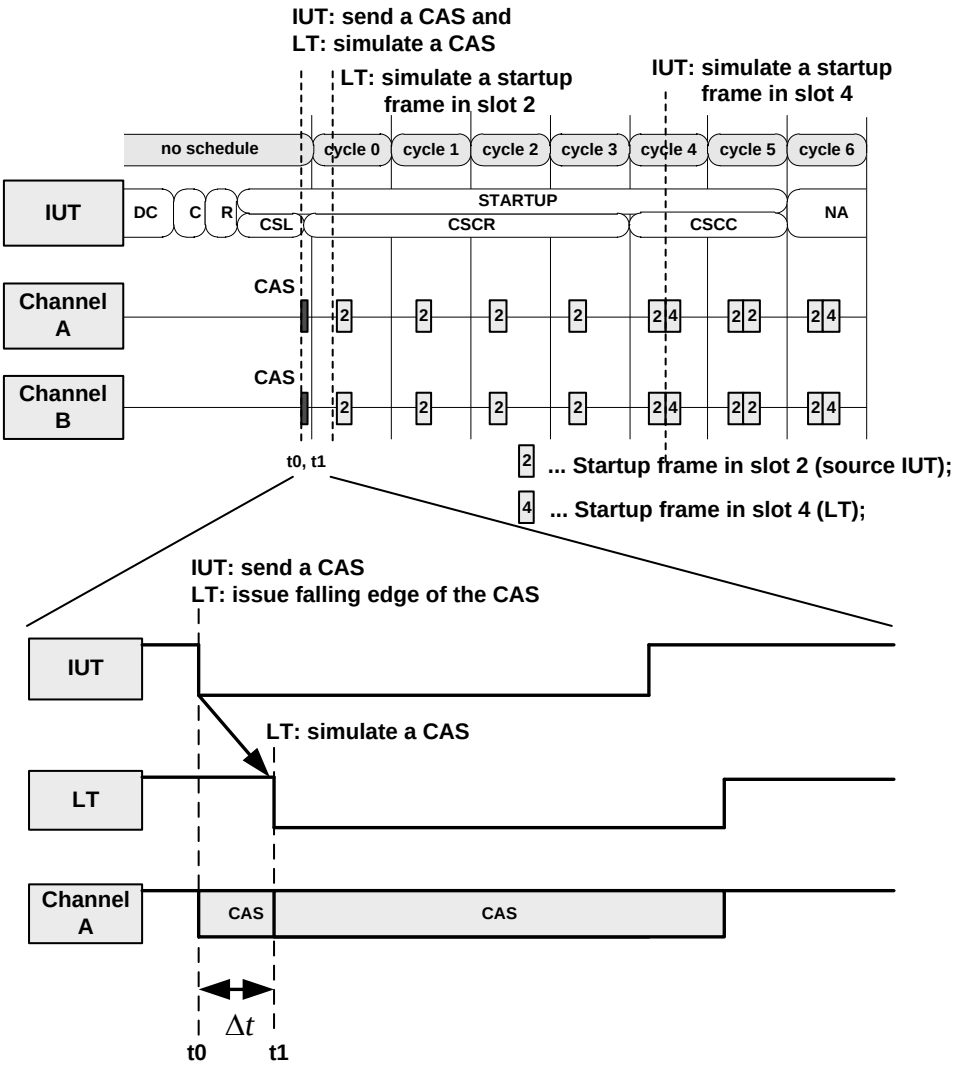


Figure 116 — Parallel startup (CAS collision) IUT as leading coldstarter – Δt = (2 ± 2) * gdBit / pdMicrotick µT

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends CAS and startup frames accordingly and enters the correct states when a CAS collision occurs at the beginning.

7.6.2.5 Parallel startup (CAS collision): IUT as following coldstarter

— Test purpose

Verify correct behaviour of the IUT when a CAS collision occurs at the beginning and a sync frame is received during *POC:coldstart collision resolution* state, which turns the IUT from a leading coldstarter into a following coldstarter.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 291.

Table 291 — Modification to basic configurations for coldstart collision resolution – parallel startup (CAS collision): IUT as following coldstarter

Parameter	Modification
pKeySlotID	2
gColdstartAttempts	3

The IUT is configured as coldstarter sending startup frames in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS. (low phase of length *gdTSSTransmitter* + *cdCAS*) starting *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

— Test execution

1) The LT simulates a CAS, starting at (2 ± 2) * *gdBit* / *pdMicrotick* µT after the falling edge of that CAS sent by the IUT.

2) It is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) after t = (500 –

*cdCASActionPointOffset * gdMacrotick* / *pdMicrotick*) µT (t>0) after the CAS has been started in the previous test step and that the *vRemainingColdstartAttempts* = 2.

3) The LT simulates startup frames in slot 1 of cycles 0 to 9.

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart abort indicator is true after (*gdTSSTransmitter* + *cdFSS* + 5 * 10 + *cdBSS*) * *gdBit* / *pdMicrotick* + 500 µT after the LT has started the transmission of the header in slot 1 of cycle 0 in the previous step.

5) In the NIT of cycle 1, it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

6) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* µT after the LT has applied the frame in slot1 of cycle 2.

7) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 500 µT after the start of slot 2 of cycle 3.

8) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

9) It is verified (LT) that the IUT sends a startup frame in slot 2 of cycle 6 to 9.

10) In cycle 9, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 117 depicts the parallel startup (CAS collision) IUT as following coldstarter – Δt = (2 ± 2) * gdBit / pdMicrotick µT.

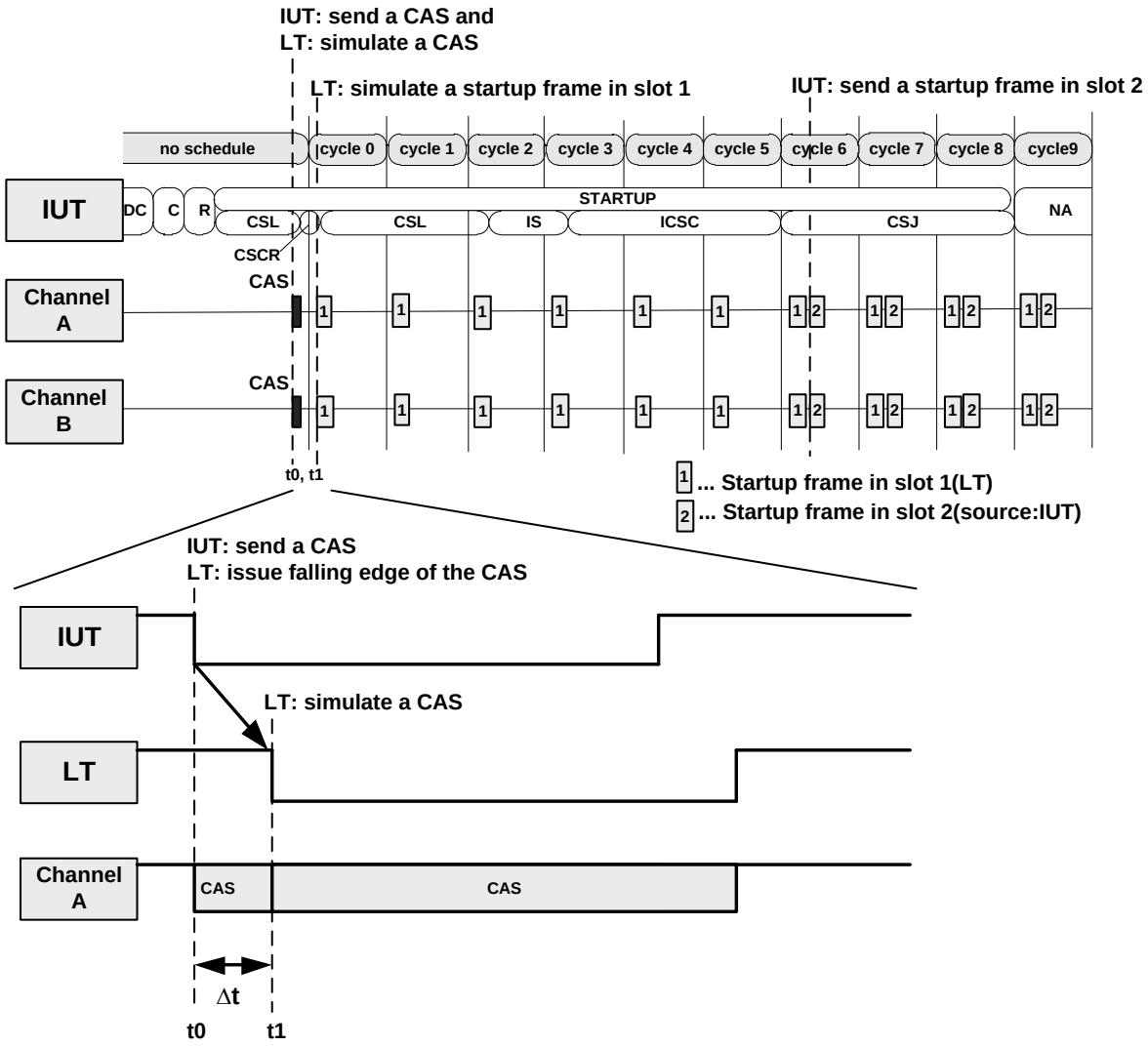


Figure 117 — Parallel startup (CAS collision) IUT as following coldstarter – Δt = (2 ± 2) * gdBit / pdMicrotick µT

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends CAS and startup frames accordingly and enters the correct states when a CAS collision occurs at the beginning.

7.6.3 Coldstart consistency check

7.6.3.1 Channel idle until last coldstart attempt

— Test purpose

Verify correct startup with the IUT as leading coldstarter and with interaction during the last coldstart attempt. The IUT should enter *POC:coldstart gap* state when no other startup node is present and should be able to reach POC:normal active state with the last coldstart attempt.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 292.

Table 292 — Modification to basic configurations for coldstart consistency check – channel idle until last coldstart attempt

Parameter	Modification		
	I	II	III
gColdstartAttempts	2	15	31
pKeySlotID	2		

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

— Test execution

The test has to be performed repeatedly with different values for *gColdstartAttempts*

1) For n = 0 to *gColdstartAttempts* – 2:
In cycle (6n)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart collision resolution* (*vPOC!State* = *STARTUP* and *vPOC!StartupState* =

COLDSTART_COLLISION_RESOLUTION) state, it is also verified (UT) that *vRemainingColdstartAttempts* = *gColdstartAttempts* – (n+1).

In cycles 6n to (6n + 4)mod 64, it is verified (LT) that the IUT sends startup frames in slot 2. In cycle (6n + 4)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

In cycle (6n + 5)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!StartupState* = *COLDSTART_GAP*). In cycle (6n + 5)mod 64, it is verified (LT) that the IUT does not send anything.

2) In cycle (6 * (*gColdstartAttempts* – 1))mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) and it is verified (UT) that *vRemainingColdstartAttempts* = 0.

3) It is verified (LT) that the IUT sends startup frames in slot 2 of cycles (6 * (*gColdstartAttempts* – 1))mod 64 to (6 * (*gColdstartAttempts* – 1) + 6)mod 64.

4) The LT simulates startup frames in slot 1 of cycles (6 * (*gColdstartAttempts* – 1) + 4)mod 64 to (6 * (*gColdstartAttempts* – 1) + 6)mod 64.

5) In cycle (6 * (*gColdstartAttempts* – 1) + 4)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

6) In cycle (6 * (*gColdstartAttempts* – 1) + 5)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

7) In cycle (6 * (*gColdstartAttempts* – 1) + 6)mod 64, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 118 depicts the channel idle until last coldstart attempt in *POC:coldstart consistency check*.

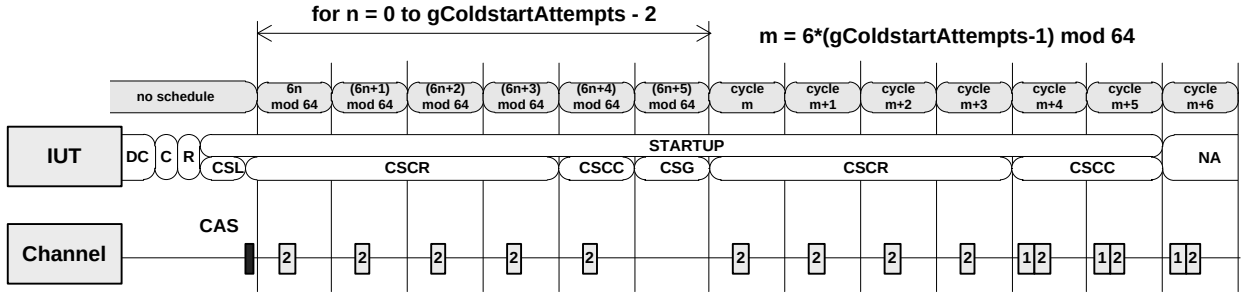


Figure 118 — Channel idle until last coldstart attempt in POC:coldstart consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT successfully performs a startup after interaction during the last coldstart attempt. The IUT enters *POC:coldstart gap* state if no startup frame is received while in *POC:coldstart consistency check* state.

7.6.3.2 Channel idle

— Test purpose

Verify correct startup behaviour with the IUT as leading coldstarter when no startup frame is received. The IUT should enter *POC:integration listen* state after the configured number of coldstart attempts. Verify de-activated MTS transmission during startup.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 293.

Table 293 — Modification to basic configurations for coldstart consistency check – channel idle

Parameter	Modification
pKeySlotID	2
gColdstartAttempts	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) state 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*)*.*

10) It is checked (LT) that the IUT sends a startup frame in slot 2 of cycles 0 to 3.

— Test execution

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

2) According to IUT documentation, the UT requests the IUT to transmit an MTS in cycle 4.

3) It is verified (LT) that the IUT sends a startup frame in slot 2 of cycle 4.

4) In the symbol window of cycle 4, it is verified (LT) that the IUT does not transmit an MTS.

5) In cycle 5, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

6) It is verified (LT) that the IUT does not send anything in cycle 5.

7) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

8) According to IUT documentation, the UT requests the IUT to transmit an MTS in cycle 6.

9) In the symbol window of cycle 6, it is verified (LT) that the IUT does not transmit an MTS.

10) It is verified (LT) that the IUT sends startup frames in slot 2 of cycles 6 to 10.

11) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

12) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) and that the coldstart abort indicator is true.

13) It is verified (LT) that the IUT does not send anything after cycle 10 for at least 4 * *gListenNoise* * *pdListenTimeout*.

14) It is verified (UT) that the UT can reset the coldstart abort indicator after the IUT has entered *POC:integration listen* state.

Figure 119 depicts the channel idle in *POC:coldstart consistency check*.

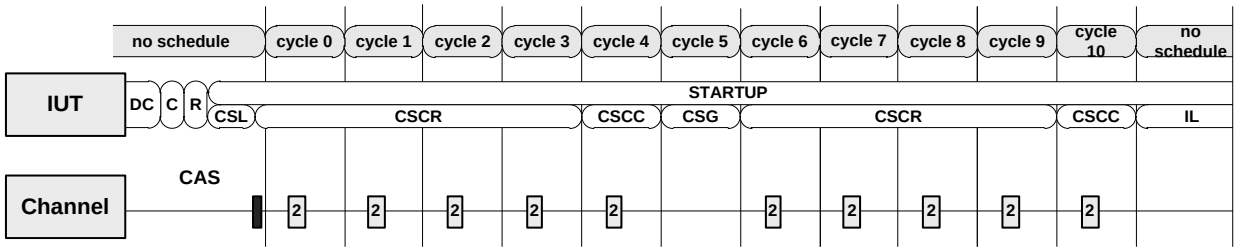


Figure 119 — Channel idle in POC:coldstart consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the coldstart due to missing interaction after the configured number of coldstart attempts were performed. The IUT does not transmit an MTS during startup.

7.6.3.3 No startup frames in second cycle

— Test purpose

Verify correct behaviour with the IUT as leading coldstarter when no startup frames are received in the second cycle of *POC:coldstart consistency check* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 294.

Table 294 — Modification to basic configurations for coldstart consistency check – no startup frames in second cycle

Parameter	Modification
gColdstartAttempts	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

— Test execution

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

2) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 4 and 5.

3) The LT simulates startup frames in slots 2 and 3 of cycle 4.

4) The LT simulates no startup frames in cycle 5.

5) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) and that the coldstart abort indicator is true.

Figure 120 depicts the no startup frames in second cycle in *POC:coldstart consistency check*.

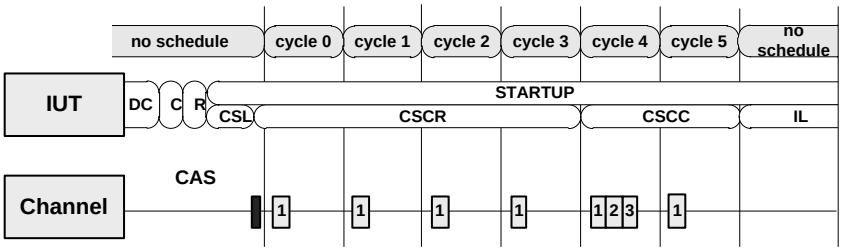


Figure 120 — No startup frames in second cycle in POC:coldstart consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the coldstart due to missing interaction in the odd cycle while the IUT is in the *POC:coldstart consistency check* state.

7.6.3.4 Deviation outside bounds

— **Test purpose**

Verify correct behaviour with the IUT as leading coldstarter when the measured deviation exceeds clock correction bounds during *POC:coldstart consistency check* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 295.

Table 295 — Modification to basic configurations for coldstart consistency check – deviation outside bounds

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
gColdstartAttempts	2		
pClusterDriftDamping [µT]	0		
pOffsetCorrectionOut [µT]	32		

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

— Test execution

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

2) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 4 and 5.

3) The LT simulates startup frames in slots 2 and 3 of cycle 4.

4) The LT simulates startup frames in slots 2 and 3 of cycle 5 with a deviation Δt = 3 * *pOffsetCorrectionOut*.

5) In cycle 6 of the LT, 500 µT after LT cycle start and before LT NIT, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) and that the coldstart abort indicator is true.

Figure 121 depicts the deviation outside bounds in *POC:coldstart consistency check*.

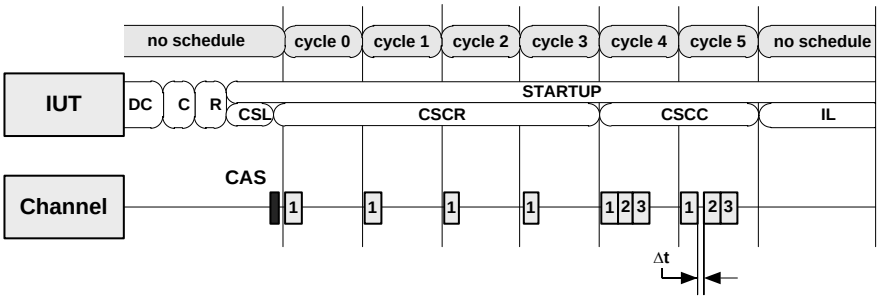


Figure 121 — Deviation outside bounds in POC:coldstart consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the coldstart due to a synchronisation result not within bounds while the IUT is in the *POC:coldstart consistency check* state.

7.6.3.5 Less startup frames in second cycle

— **Test purpose**

Verify correct behaviour with the IUT as leading coldstarter when less startup frames are received in the second cycle of *POC:coldstart consistency check* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

— **Test execution**

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

2) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 4.

3) The LT simulates startup frames in slots 2 and 3 of cycle 4.

4) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 5.

5) The LT simulates a startup frame in slot 2 of cycle 5.

6) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

7) The LT simulates startup frames in slots 2 and 3 of cycle 6.

8) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 6.

Figure 122 depicts the less startup frames in second cycle in *POC:coldstart consistency check*.

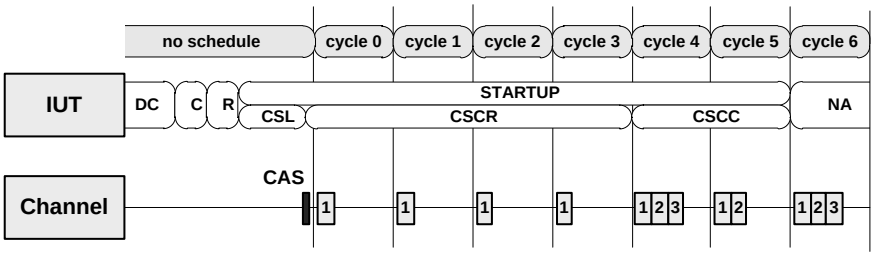


Figure 122 — Less startup frames in second cycle in POC:coldstart consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT reaches *POC:normal active* state if at least one startup frame is received in even / odd cycle while the IUT is in the *POC:coldstart consistency check* state.

7.6.3.6 Number of startup frames

— Test purpose

Verify correct behaviour with the IUT as leading coldstarter when up to 14 startup frames are received while the IUT is in the *POC:coldstart consistency check* state.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

— Test execution

This test has to be performed repeatedly with the LT simulating (a) 1, (b) 7 and (c) 14 startup frames in adjacent slots starting with slot 2 while the IUT is in the *POC:coldstart consistency check* state.

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

2) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 4.

3) In cycles 4 to 6, the LT simulates startup frames

   (a) in slot 2,

   (b) in 7 adjacent slots starting with slot 2 and

(c) in 14 adjacent slots starting with slot 2.

4) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 5.

5) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State = NORMAL_ACTIVE*).

6) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 6.

Figure 123 depicts the number of startup frames in *POC:coldstart consistency check* – one startup frame.

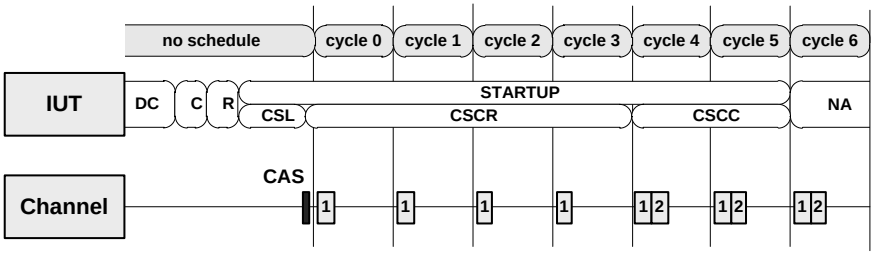


Figure 123 — Number of startup frames in POC:coldstart consistency check – one startup frame

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT reaches *POC:normal active* state if at least one startup frame pair is received in an even / odd double-cycle while the IUT is in the *POC:coldstart consistency check* state.

7.6.3.7 Short noise during coldstart consistency check

— Test purpose

Verify correct behaviour of the IUT as leading coldstarter when short noise is recognized during *POC:coldstart consistency check* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 296.

Table 296 — Modification to basic configurations for coldstart consistency check – short noise during coldstart consistency check

Parameter	Modification
gColdstartAttempts	2

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 3.

— **Test execution**

1) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

2) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 4.

3) The LT simulates a short noise (low phase of length 1 * *gdBit*) in slot 2 of cycle 4.

4) In cycle 5, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

5) It is verified (LT) that the IUT does not send anything in cycle 5.

6) In cycle 6, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

7) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 6 to 10.

8) The LT simulates a startup frame in slot 2 of cycle 10.

9) It is verified (LT) that the IUT sends a startup frame in slot 1 of cycle 11.

10) The LT simulates a short noise (low phase of length 1 * *gdBit*) in slot 2 of cycle 11.

11) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

12) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) and that the coldstart abort indicator is true.

13) It is verified (LT) that the IUT does not send anything after cycle 11 for at least 4 * *gListenNoise* * *pListenTimeout*.

14) It is verified (UT) that the UT can reset the coldstart abort indicator after the IUT has entered *POC:integration listen* state.

Figure 124 depicts the short noise during coldstart consistency check.

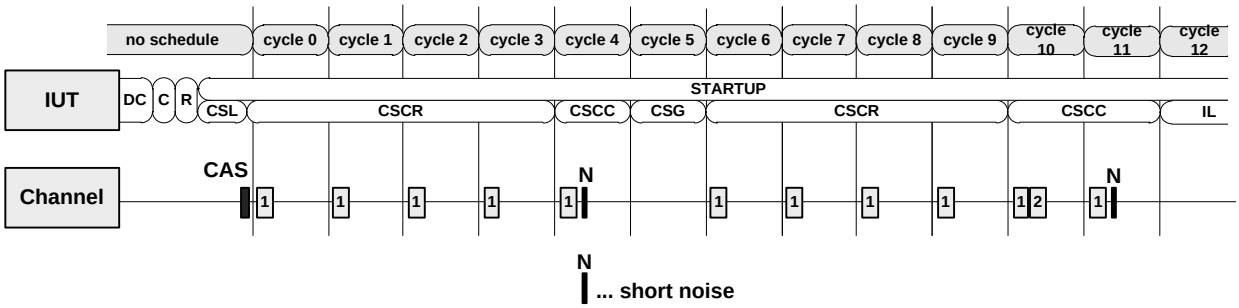


Figure 124 — Short noise during coldstart consistency check

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts the coldstart due to missing interaction in the odd cycle while the IUT is in the *POC:coldstart consistency check* state.

7.6.4 Coldstart gap

7.6.4.1 Header reception

— Test purpose

Verify correct behaviour with the IUT as leading coldstarter when a frame header is received during *POC:coldstart gap* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 297.

Table 297 — Modification to basic configurations for coldstart gap – header reception

Parameter	Modification
pKeySlotID	2

The IUT is configured as coldstarter sending startup frames in slot 2 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 2 of cycles 0 to 3.

11) In cycle 4, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

12) It is checked (LT) that the IUT sends a startup frame in slot 2 of cycle 4.

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) In slot 1 of cycle 5, 500 µT after slot start and before slot end, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

2) The LT simulates a startup frame header in slot 3 of cycle 5 (null frame indicator, sync frame indicator and startup frame indicator are set to '1', reserved bit and payload preamble indicator are set to '0', the frame ID is 3, the payload length 1, the cycle count 5). The LT has to simulate the complete header including the following BSS after the last header byte.

3) In cycle 5, 500 µT after slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart abort indicator is true.

4) It is verified (LT) that the IUT sends a CAS (*cVotingDelay* + *cStrobeOffset* + *cdInternalRxDelayMax*) * *gdSampleClockPeriod* / *pdMicrotick* + *pdListenTimeout* + (*cChannelIdleDelimiter* – 1) * *gdBit* / *pdMicrotick* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* + ξ µT + (φ_{Rx} + φ_{Tx}) ns after the end of the LT's final BSS in test step 2.

Figure 125 depicts the header reception in *POC:coldstart gap*.

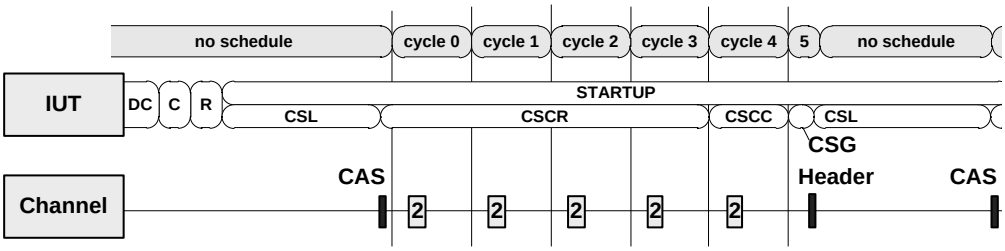


Figure 125 — Header reception in POC:coldstart gap

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts a coldstart if a header is received during the *POC:coldstart gap* state.

7.6.4.2 CAS reception

— Test purpose

Verify correct behaviour with the IUT as leading coldstarter when a symbol is received during *POC:coldstart gap* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 298.

Table 298 — Modification to basic configurations for coldstart gap – CAS reception

Parameter	Modification
pKeySlotID	2

The IUT is configured as coldstarter sending startup frames in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

6) The UT initiates the startup procedure with the CHI command RUN.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) It is checked (LT) that the IUT sends a CAS.

9) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

10) It is checked (LT) that the IUT sends startup frames in slot 2 of cycles 0 to 3.

11) In cycle 4, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

12) It is checked (LT) that the IUT sends a startup frame in slot 2 of cycle 4.

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) In cycle 5, 500 µT after cycle start and before slot 4, it is verified (UT) that the IUT is in the *POC:coldstart gap* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_GAP*).

2) It is verified (LT) that the IUT does not send a startup frame in slot 2 of cycle 5.

3) The LT simulates a CAS in slot 4 of cycle 5.

4) In cycle 5, 500 µT after slot 4 and before cycle end, it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart abort indicator is true.

Figure 126 depicts the CAS reception in *POC:coldstart gap*.

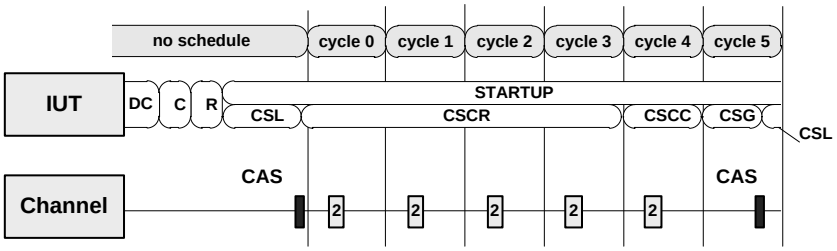


Figure 126 — CAS reception in POC:coldstart gap

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts a coldstart if a CAS is received during the *POC:coldstart gap* state.

7.6.5 Integration listen

7.6.5.1 ALLOW_COLDSTART command (coldstarter)

— Test purpose

Verify startup with the IUT as leading coldstarter and with coldstart inhibit reset in the *POC:integration listen* state.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 * *pdListenTimeout* after the CHI command RUN.

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command ALLOW_COLDSTART.

5) It is verified (LT) that the IUT sends a CAS.

Figure 127 depicts the ALLOW_COLDSTART command (coldstarter) in *POC:integration listen*.

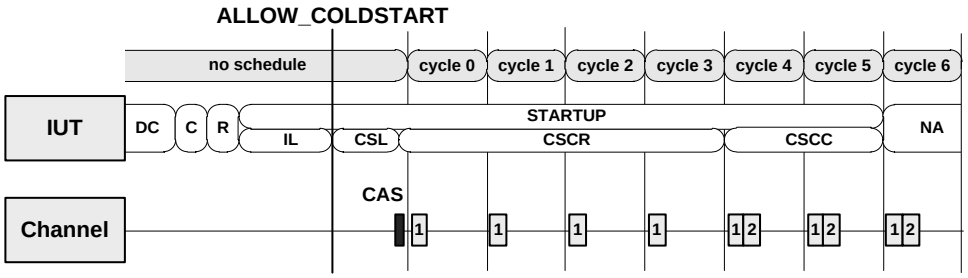


Figure 127 — ALLOW_COLDSTART command (coldstarter) in POC:integration listen

— Postamble

1) In cycle 0, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*).

2) It is checked (LT) that the IUT sends startup frames in slot 1 of cycles 0 to 6.

3) In cycle 4, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:coldstart consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_CONSISTENCY_CHECK*).

4) The LT simulates startup frames in slot 2 of cycles 4 to 6.

5) In cycle 6, 500 µT after cycle start and before NIT, it is checked (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

6) The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters the *POC:coldstart listen* state and initiates a coldstart after the UT issued the CHI command ALLOW_COLDSTART while the IUT is in the *POC:integration listen* state.

7.6.5.2 ALLOW_COLDSTART command (integrating node)

— Test purpose

Verify startup with the IUT as integrating node and with coldstart inhibit reset in the *POC:integration listen* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 299.

Table 299 — Modification to basic configurations for coldstart gap – ALLOW_COLDSTART command (integrating node)

Parameter	Modification
pKeySlotUsedForStartup	false

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— Test execution

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 * *pdListenTimeout* after the CHI command RUN.

4) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command ALLOW_COLDSTART.

5) It is verified (LT) that the IUT does not send anything during the test. The LT shall observe the IUT for a period of 2 * *gListenNoise* * *pdListenTimeout*.

Figure 128 depicts the ALLOW_COLDSTART command (integrating node) in *POC:integration listen*.

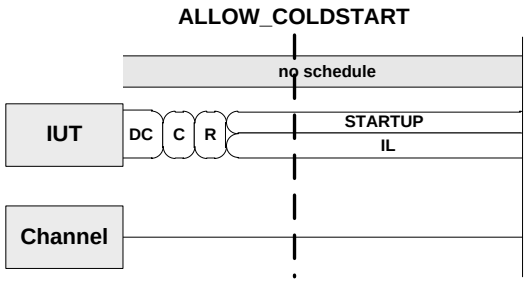


Figure 128 — ALLOW_COLDSTART command (integrating node) in POC:integration listen

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT re-enters the *POC:integration listen* state after the UT issued the CHI command ALLOW_COLDSTART while the IUT is in the *POC:integration listen* state.

7.6.5.3 Wait for startup

— **Test purpose**

Verify startup with the IUT as leading coldstarter but with coldstart inhibited.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

The IUT is configured to send startup frames in slot 1 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) It is verified (LT) that the bus stays idle for a time interval of 2 * *gListenNoise* * *pdListenTimeout*.

4) It is verified (UT) that the IUT is still in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) after 2 * *gListenNoise* * *pdListenTimeout*.

5) The LT simulates a CAS 3 * *gListenNoise* * *pdListenTimeout* ± 0,05 * *pdListenTimeout* µT after the CHI command RUN.

6) The LT simulates startup frames in slot 2 of cycles 0 to 8.

7) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 3 * *gListenNoise* * *pdListenTimeout* ± 0,05 * *pdListenTimeout* + 0,5 * *gdCycle* after the CHI command RUN.

8) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 3 * *gListenNoise* * *pdListenTimeout* ± 0,05 * *pdListenTimeout* + 1,5 * *gdCycle* after the CHI command RUN.

9) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

10) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 4 to 7.

11) In cycle 7, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 129 depicts the wait for startup in *POC:integration listen*.

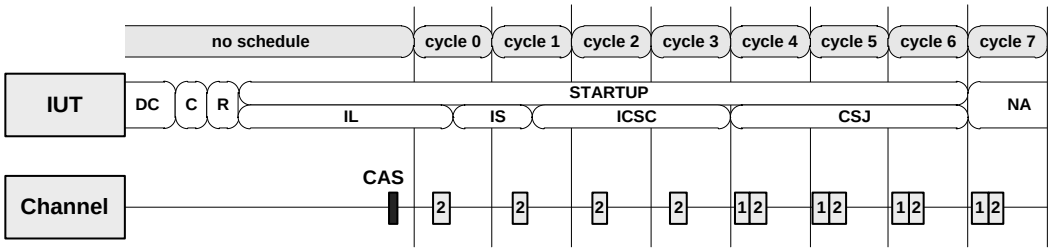


Figure 129 — Wait for startup in POC:integration listen

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters the *POC:initialize schedule* state after the LT initialized startup while the IUT is in the *POC:integration listen* state.

7.6.5.4 Invalid startup frame reception

— **Test purpose**

Verify correct behaviour of the IUT when invalid startup frames are received during *POC:integration listen* state. The IUT shall ignore the invalid startup frames and stay in the *POC:integration listen* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 300.

Table 300 — Modification to basic configurations for coldstart gap – invalid startup frame reception

Parameter	Modification
pKeySlotID	2
gCycleCountMax	31

The IUT is configured to send startup frames in slot 2 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

2) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT begins to simulate a startup frame with the frame ID set to 0 (frame ID = 0, payload length = *gPayloadLengthStatic*, sync frame indicator and startup frame indicator set to '1', null frame indicator and payload preamble indicator set to '0', cycle count = 0) *pdListenTimeout* / 2 ± 0,05 * *pdListenTimeout* µT after the CHI command RUN.

4) It is verified (UT) that the IUT stays in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

5) The LT begins to simulate a startup frame with the frame ID set to *gNumberOfStaticSlots* + 1 (first dynamic slot ID) (frame ID = *gNumberOfStaticSlots* + 1, payload length = *gPayloadLengthStatic*, sync frame indicator and startup frame indicator set to '1', null frame indicator and payload preamble indicator set to '0', cycle count = 0) *gdCycle* after the beginning of the LT's previous frame.

6) It is verified (UT) that the IUT stays in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

7) The LT begins to simulate a startup frame with the payload length set to *gPayloadLengthStatic* + 1 (frame ID = 1, payload length = *gPayloadLengthStatic* + 1, sync frame indicator and startup frame indicator set to '1', null frame indicator and payload preamble indicator set to '0', cycle count = 0) *gdCycle* after the beginning of the LT's previous frame.

8) It is verified (UT) that the IUT stays in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

9) The LT begins to simulate a startup frame with the sync frame indicator set to '0' (frame ID = 1, payload length = *gPayloadLengthStatic*, sync frame indicator set to '0', startup frame indicator set to '1', null frame indicator and payload preamble indicator set to '0', cycle count = 0) *gdCycle* after the beginning of the LT's previous frame.

10) It is verified (UT) that the IUT stays in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

11) The LT begins to simulate a startup frame with the startup frame indicator set to '0' (correct sync frame) (frame ID = 1, payload length = *gPayloadLengthStatic*, sync frame indicator set to '1', startup frame indicator set to '0', null frame indicator and payload preamble indicator set to '0', cycle count = 0) *gdCycle* after the beginning of the LT's previous frame.

12) It is verified (UT) that the IUT stays in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

13) The LT begins to simulate a valid startup frame with an odd cycle count (frame ID = 1, payload length = *gPayloadLengthStatic*, sync frame indicator and startup frame indicator set to '1', null frame indicator and payload preamble indicator set to '0', cycle count = 5) *gdCycle* after the LT's previous frame.

14) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

15) The LT begins to simulate a startup frame with the null frame indicator set to '0' and the payload preamble indicator set to '1' (frame ID = 1, payload length = *gPayloadLengthStatic*, sync frame indicator and startup frame indicator set to '1', null frame indicator set to '0', payload preamble indicator set to '1', cycle count = 0) *gdCycle* after the LT's previous frame.

16) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

17) The LT begins to simulate a startup frame with an even cycle count of 32 (frame ID = 1, payload length = *gPayloadLengthStatic*, sync frame indicator and startup frame indicator set to '1', null frame indicator and payload preamble indicator set to '0', cycle count = 32) *gdCycle* after the LT's previous frame.

18) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the startup frame reception.

19) The LT begins to simulate short noise (low phase of length 1 * *gdBit*) *gdCycle* after the LT's previous frame.

20) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 500 µT after the noise reception.

Figure 130 depicts the invalid startup frame reception in *POC:integration listen* – invalid startup frames on channel A.

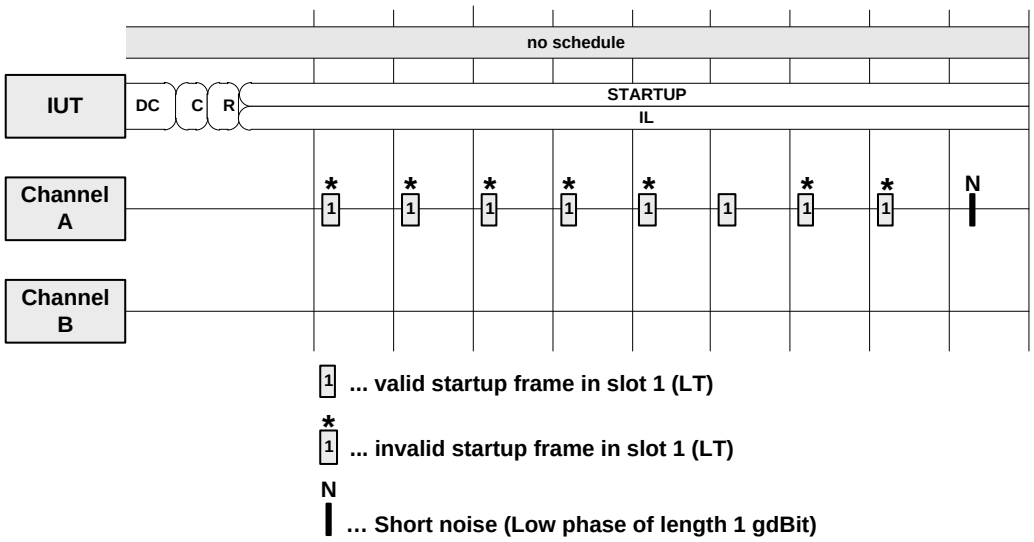


Figure 130 — Invalid startup frame reception in POC:integration listen – invalid startup frames on channel A

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT ignores invalid startup frames and stays in the *POC:integration listen* state.

7.6.6 Initialize schedule

7.6.6.1 Startup frame reception (coldstarter)

— **Test purpose**

Verify startup with the IUT as coldstarter but with coldstart inhibited. The IUT should leave the *POC:initialize schedule* state and enter the *POC:integration coldstart check* state after reception of a startup frame from the leading coldstarter.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

The IUT is configured to send startup frames in slot 1 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT simulates startup frames in slot 2 of cycles 0 to 8.

3) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

5) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

6) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 4 to 7 starting at the static action point + ξ + ξ_{IUT} µT and holding the appropriate cycle count and the frame ID 1. The sync frame and the startup frame indicator are set.

7) In cycle 7, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 131 depicts the startup frame reception (coldstarter) in *POC:initialize schedule*.

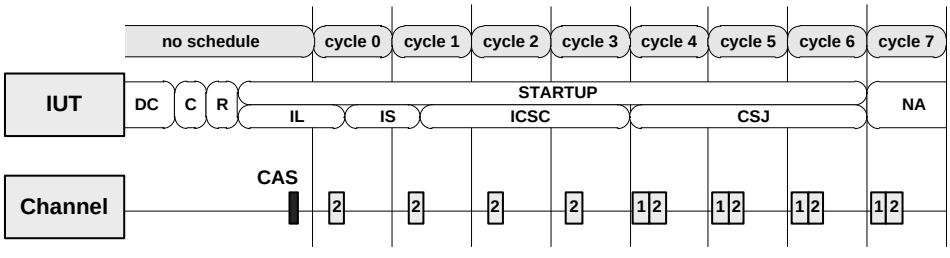


Figure 131 — Startup frame reception (coldstarter) in POC:initialize schedule

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT joins the simulated leading coldstarter and derives its schedule correctly. The IUT sends startup frames with appropriate cycle count, frame ID, sync frame indicator and startup frame indicator starting at the static slot action point.

7.6.6.2 Startup frame reception (integrating node)

— **Test purpose**

Verify startup with the IUT as integrating node. The IUT shall join the simulated coldstarters and derive the cycle schedule correctly.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 301.

Table 301 — Modification to basic configurations for coldstart gap – startup frame reception (integrating node)

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

The IUT is configured to send sync frames in slot 4 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

— **Test execution**

The LT simulates three coldstarters (one leading, two following) for this test. For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT simulates startup frames in slot 1 of cycles 0 to 8.

3) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

5) The LT simulates startup frames in slots 2 and 3 of the cycles 4 to 8.

6) In cycle 8, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

7) It is verified (LT) that the IUT sends a frame in slot 4 of cycle 8 starting at the static action point + ξ + ξ_{IUT} µT and holding cycle count 8 and frame ID 4. The sync frame indicator is set to '1', the startup frame indicator is set to '0'.

Figure 132 depicts the startup frame reception (integrating node) in *POC:initialize schedule*.

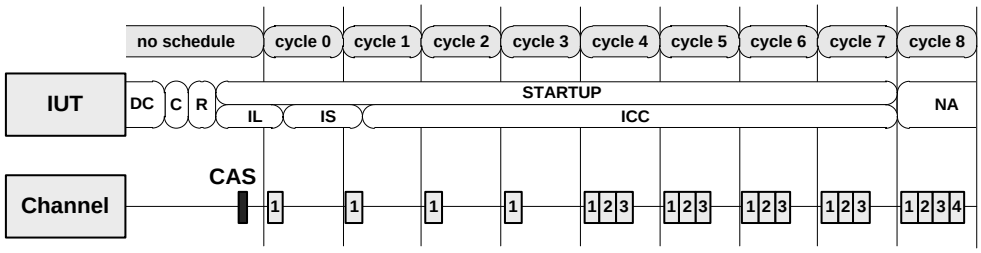


Figure 132 — Startup frame reception (integrating node) in POC:initialize schedule

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT joins the simulated coldstarters and derives its schedule correctly. The IUT's sends a sync frame with appropriate cycle count, frame ID, sync frame indicator and startup frame indicator starting at the static slot action point.

7.6.6.3 Missing, incomplete, or too early startup frame (integrating node)

— Test purpose

Verify startup with the IUT as integrating node when a startup frame is missing, incomplete, or too early in state *POC:initialize schedule*. The IUT shall abort the integration attempt as a result of the missing or incomplete, or too early startup frame.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 302.

Table 302 — Modification to basic configurations for coldstart gap – missing, incomplete, or too early startup frame (integrating node)

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

The IUT is configured to send sync frames in slot 4 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 1 of cycle 0.

3) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) The LT simulates

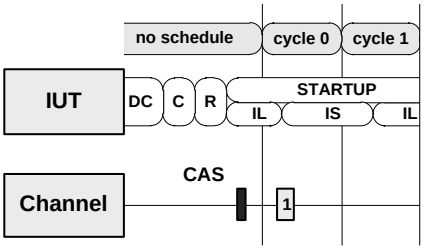
   (a) idle in cycle 1.

   (b) a short noise (low phase of length 1 * *gdBit*) in slot 1 of cycle 1.

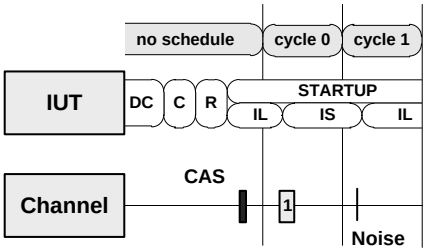
   (c) a startup frame in slot 1 of cycle 1. The LT's cycle length (from cycle 0 to cycle 1) is (*pMicroPerCycle* – (*pRateCorrectionOut* + 1)) * *pSamplesPerMicrotick* – 1 σT.

5) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

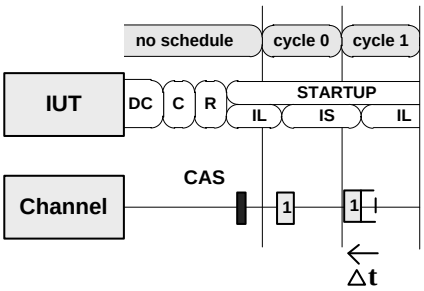
Figure 133 depicts the missing, incomplete, or too early startup frame (integrating node) in *POC:initialize schedule*.



a) Second startup frame missing



b) Second startup frame incomplete



c) Second startup frame too early
Δt = (pRateCorrectionOut + 1) * pSamplesPerMicrotick + 1 μT

Figure 133 — Missing, incomplete, or too early startup frame (integrating node) in POC:initialize schedule

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT re-enters the *POC:integration listen* state because the integration attempt is aborted due to missing, incorrect, or too early startup frame(s) on the respective channel(s).

7.6.6.4 Missing startup frame on one channel

— Test purpose

Verify startup with the IUT as integrating node when a startup frame is missing on either channel A or channel B in state *POC:initialize schedule*. The IUT shall not abort the integration attempt as a result of the missing startup frame.

— Applicability

DC.

— Configuration

All basic configurations using the modifications as listed in Table 303.

Table 303 — Modification to basic configurations for coldstart gap – missing startup frame on one channel

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

— **Test execution**

The test has to be performed repeatedly (a), (b), (c) and (d) with the LT simulating startup frames according to Table 304.

Table 304 — Startup frames for variants (a), (b), (c), (d)

Variant	Cycle 0, Channel A	Cycle 0, Channel B	Cycle 1, Channel A	Cycle 1, Channel B
(a)	1 MT after Action Point	At Action Point	At Action Point	none
(b)	1 MT after Action Point	At Action Point	none	At Action Point
(c)	At Action Point	1 MT after Action Point	At Action Point	none
(d)	At Action Point	1 MT after Action Point	none	At Action Point

1) The LT simulates a CAS on both channels *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 1 of cycle 0 on both channels. The frame (a, b) on channel A and (c, d) on channel B starts 1 MT after the start of the frame on the other channel. The earlier frame starts at the static action point.

3) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) The LT simulates a startup frame in slot 1 of cycle 1 (a, c) on channel A and (b, d) on channel B, only.

5) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 134 depicts the missing startup frame on one channel in *POC:initialize schedule* – missing startup frame on channel B.

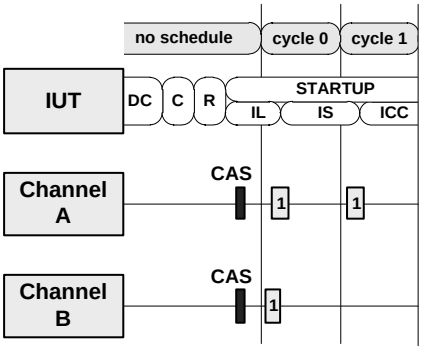


Figure 134 — Missing startup frame on one channel in POC:initialize schedule – missing startup frame on channel B

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT does not abort the integration attempt due to a missing startup frame on one channel during the *POC:initialize schedule* state. The IUT enters the *POC:integration consistency check* state.

7.6.6.5 Missing, incomplete, or too early startup frame (following coldstarter)

— Test purpose

Verify startup with the IUT as following coldstarter when a startup frame is missing, incomplete, or too early in state *POC:initialize schedule*. The IUT shall abort the integration attempt as a result of the missing, incomplete, or too early startup frame.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 305.

Table 305 — Modification to basic configurations for coldstart gap – missing, incomplete, or too early startup frame (following coldstarter)

Parameter	Modification
pKeySlotID	4

The IUT is configured to send startup frames in slot 4 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000µT after the CHI command CONFIG_COMLETE.

— **Test execution**

The LT simulates the leading coldstarter for this test.

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

2) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

3) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

4) The LT simulates a startup frame in slot 1 of cycle 0.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

6) The LT

   (a) does not simulate any startup frame in cycle 1.

   (b) a short noise (low phase of length 1 * gdBit) in slot 1 of cycle 1.

   (c) simulates a startup frame in slot 1 of cycle 1. The LT's cycle length (from cycle 0 to cycle 1) is (*pMicroPerCycle* – (*pRateCorrectionOut* + 1)) * *pSamplesPerMicrotick* – 1 σT.

7) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 135 depicts the missing, incomplete, or too early startup frame (following coldstarter) in *POC:initialize schedule*.

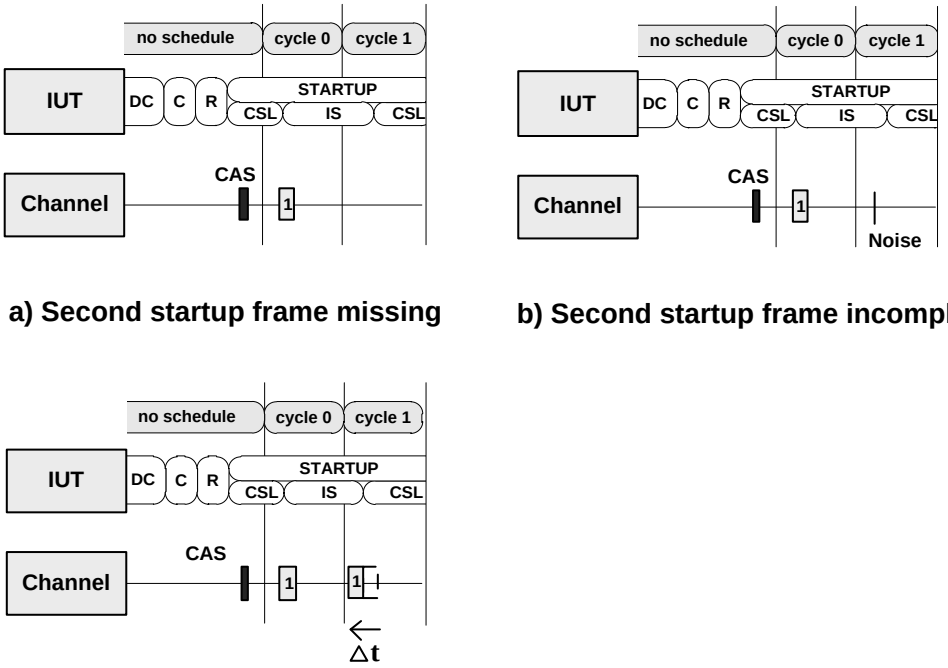


Figure 135 — Missing, incomplete, or too early startup frame (following coldstarter) in POC:initialize schedule

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT re-enters the *POC:coldstart listen* state because the integration attempt is aborted due to missing, incomplete, or too early startup frame(s) on the respective channel(s).

7.6.7 Integration coldstart check

7.6.7.1 One startup frame per cycle

— Test purpose

Verify startup with the IUT as coldstarter but with coldstart inhibited. The IUT should leave the *POC:integration coldstart check* state and enter the *POC:coldstart join* state after reception of two startup frames from the reference node (leading coldstarter).

— Applicability

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 7.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

3) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 4 to 7.

4) In cycle 7, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 136 depicts the one startup frame per cycle in POC:integration coldstart check.

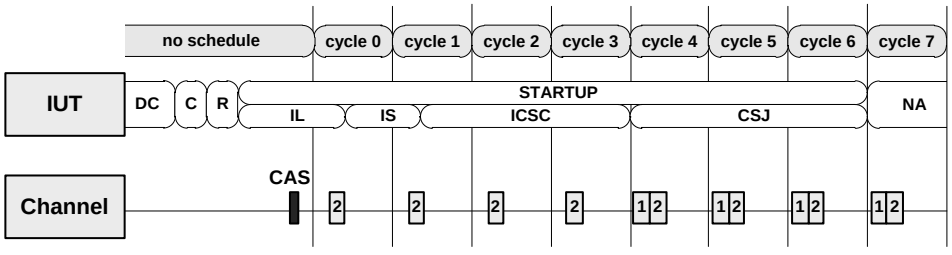


Figure 136 — One startup frame per cycle in POC:integration coldstart check

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters the *POC:coldstart join* state after the IUT received a startup frame from the reference node (leading coldstarter) while the IUT is in the *POC:integration coldstart check* state.

7.6.7.2 Receive sync frames but no startup frames

— Test purpose

Verify startup with the IUT as coldstarter with missing startup frame during *POC:integration coldstart check* state. The IUT should leave the *POC:integration coldstart check* state and enter the *POC:coldstart listen* state after reception of a sync frame instead of a startup frame.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 306.

Table 306 — Modification to basic configurations for integration coldstart check – receive sync frames but no startup frames

Parameter	Modification
pKeySlotID	4

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates startup frames in slot 1 of cycles 0 to 2.

10) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates a sync frame in slot 1 of cycle 3.

3) The LT simulates a startup frame in slot 1 of cycle 4.

4) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) and that the coldstart abort indicator is false 5,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 137 depicts the receive sync frames but no startup frames in *POC:integration coldstart check*.

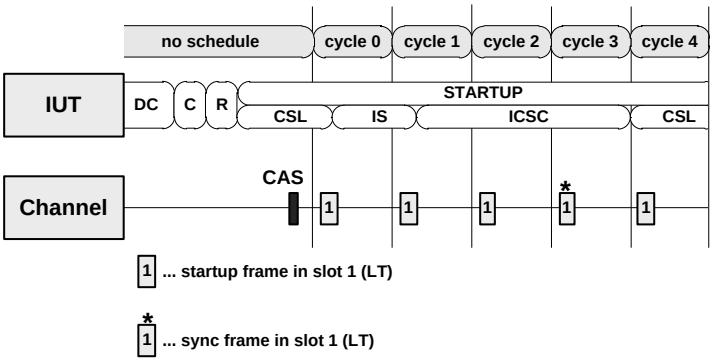


Figure 137 — Receive sync frames but no startup frames in POC:integration coldstart check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters the *POC:coldstart listen* state after the IUT received a sync frame instead of a startup frame while the IUT is in the *POC:integration coldstart check* state.

7.6.7.3 Startup frames

— **Test purpose**

Verify startup with the IUT as coldstarter but with coldstart inhibited. The IUT should leave the *POC:integration coldstart check* state and enter the *POC:coldstart join* state after reception of startup frames within two succeeding cycles.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 and 1.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

This test has to be performed repeatedly with the LT simulating (a) 2 and (b) 14 startup frames in adjacent static slots starting with slot 2.

1) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) In cycles 2 to 7, the LT simulates

   (a) two startup frames in slots 2 and 3 and

   (b) 14 startup frames in adjacent slots starting with slot 2.

3) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

4) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 4 to 7.

5) In cycle 7, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 138 depicts the startup frames in *POC:integration coldstart check* – two startup frames.

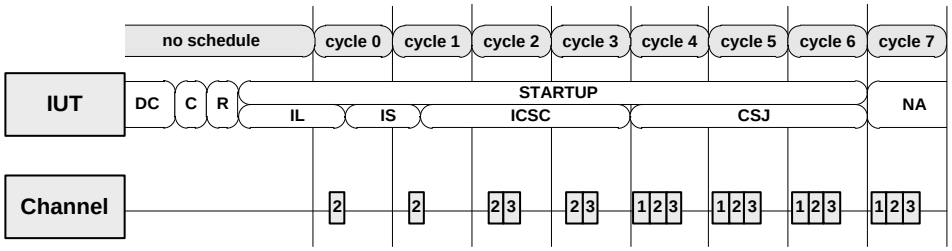


Figure 138 — Startup frames in POC:integration coldstart check – two startup frames

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT enters the *POC:coldstart join* state after the IUT received more than one (2 or 14) startup frames while the IUT is in the *POC:integration coldstart check* state.

7.6.7.4 Startup frame missing in reference slot

— Test purpose

Verify startup with the IUT as coldstarter when the startup frame in the reference slot is replaced by a startup frame in another slot during *POC:integration coldstart check* state. The IUT should leave the *POC:integration coldstart check* state and enter the *POC:coldstart listen* state when the startup frame in the reference slot is missing even though a startup frame in another slot is received.

— Applicability

SC, DC.

— Configuration

Table 307 defines the modification to basic configurations for *integration coldstart check* – startup frame missing in reference slot.

Table 307 — Modification to basic configurations for integration coldstart check – startup frame missing in reference slot

Parameter	Modification
pKeySlotID	4

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates startup frames in slot 1 of cycles 0 to 2.

10) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 2 of cycle 3.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 5,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 139 depicts the startup frame missing in reference slot in *POC:integration coldstart check*.

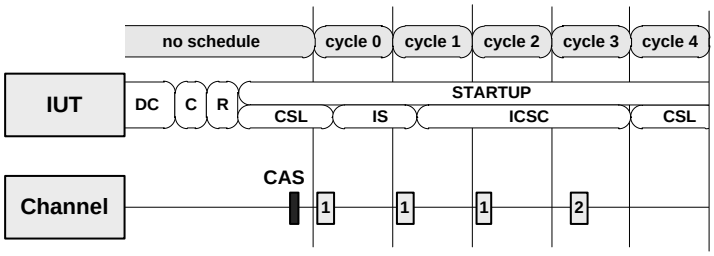


Figure 139 — Startup frame missing in reference slot in POC:integration coldstart check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT enters the *POC:coldstart listen* state at the end of the cycle during which the startup frame in the reference node is missing but another startup frame in another slot is received.

7.6.7.5 Missing or incomplete startup frame

— Test purpose

Verify correct behaviour with IUT as following coldstarter when no or incomplete startup frame is received during *POC:integration coldstart check* state. The IUT should abort the startup and enter the *POC:coldstart listen* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 308.

Table 308 — Modification to basic configurations for integration coldstart check – missing or incomplete startup frame

Parameter	Modification
pKeySlotID	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates startup frames in slot 1 of cycles 0 and 1.

10) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— Test execution

1) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates

   (a) no startup frame in slot 1 of cycle 2

   (b) a short noise (low phase of length 1 * *gdBit*) in slot 1 of cycle 2

   (c) a startup frame in slot 1 of cycle 2 but none in cycle 3

   (d) a startup frame in slot 1 of cycle 2 and a short noise (low phase of length 1 * *gdBit*) in slot 1 of cycle 3.

3) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*)

   (a, b) 4,5 * *gdCycle* ± 0,1 * *gdCycle* or

   (c, d) 5,5 * *gdCycle* ± 0,1 * *gdCycle*

   after the CHI command RUN.

Figure 140 depicts the missing or incomplete startup frame in *POC:integration coldstart check* – missing startup frame in cycle.

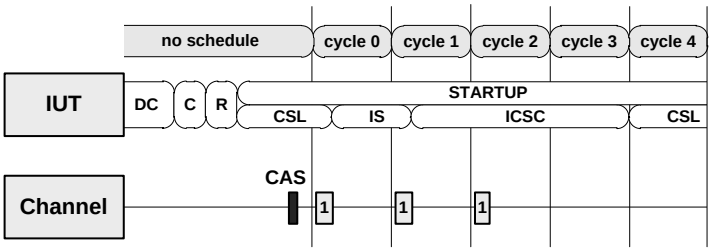


Figure 140 — Missing or incomplete startup frame in POC:integration coldstart check – missing startup frame in cycle 3

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts the startup and enters the *POC:coldstart listen* state when no or incomplete startup frame is received within one cycle while the IUT is in the *POC:integration coldstart check* state.

7.6.7.6 Deviation outside bounds

— Test purpose

Verify correct behaviour with the IUT as coldstarter when the measured deviation exceeds clock correction bounds during *POC:integration coldstart check* state.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 309.

Table 309 — Modification to basic configurations for integration coldstart check – deviation outside bounds

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
pClusterDriftDamping [µT]	0		
pKeySlotID	2		
pOffsetCorrectionOut [µT]	43		

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) state 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates a startup frame in slot 1 of cycles 0 and 1.

10) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 1 of cycle 2.

3) The LT simulates a startup frame in slot 1 of cycle 3 with a deviation $\Delta t = 3 * pOffsetCorrectionOut$.

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 5,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 141 depicts the deviation outside bounds in *POC:integration coldstart check*.

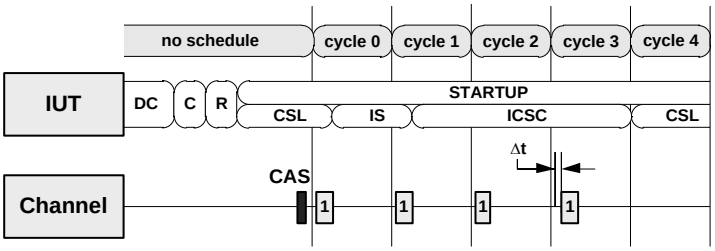


Figure 141 — Deviation outside bounds in POC:integration coldstart check

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts the startup and enters the *POC:coldstart listen* state when the deviation of the startup frame exceeds clock synchronisation bounds while the IUT is in the *POC:integration coldstart check* state.

7.6.7.7 ALLOW_COLDSTART command

— Test purpose

Verify whether the *vColdstartInhibit* flag can be cleared while the IUT is in *POC:integration coldstart check* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 310.

Table 310 — Modification to basic configurations for integration coldstart check – ALLOW_COLDSTART command

Parameter	Modification
pKeySlotID	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 1 of cycles 0 and 1.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

3) The LT simulates a startup frame in slot 1 of cycle 2 but none in cycle 3.

4) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 5,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 142 depicts the ALLOW_COLDSTART command – missing startup frame in *POC:integration coldstart check*.

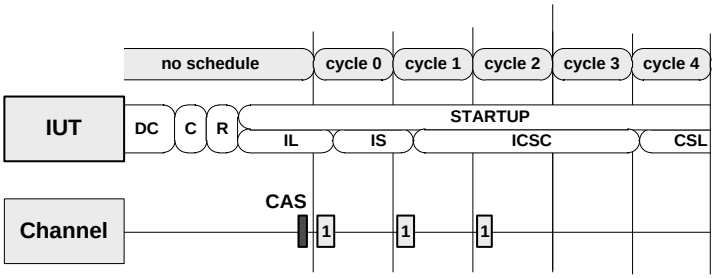


Figure 142 — ALLOW_COLDSTART command – missing startup frame in POC:integration coldstart check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the startup and enters the *POC:coldstart listen* state on the second startup attempt because the *vColdstartInhibit* flag was cleared in *POC:integration coldstart check* state during the first startup attempt.

7.6.8 Coldstart join

7.6.8.1 Missing or incomplete startup frame

— Test purpose

Verify correct behaviour with IUT as following coldstarter when no or incomplete startup frame is received during *POC:coldstart join* state. The IUT should abort the startup and enter the *POC:coldstart listen* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 311.

Table 311 — Modification to basic configurations for coldstart join – missing or incomplete startup frame

Parameter	Modification
pKeySlotID	2

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates a startup frame in slot 1 of cycle 0.

10) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

11) The LT simulates a startup frame in slot 1 of cycle 1.

12) It is checked (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

The test has to be performed repeatedly for missing startup frame in slot 1 of cycle (a) 4, (b) 5 and (c) 6 and for sync frame instead of startup frame in slot 1 of cycle (d) 4, (e) 5 and (f) 6 for short noise (low phase of length 1 * gdBit) in slot 1 of cycle (g) 4, (h) 5 and (i) 6.

1) The LT simulates startup frames in slot 1 of cycles 2 and 3.

2) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

3) The LT simulates frames according to Table 312:

Table 312 — LT stimuli in cycles 4, 5 and 6 for variants (a) to (i)

Variant	Cycle 4	Cycle 5	Cycle 6
(a)	missing	startup frame	missing
(b)	startup frame	missing	missing
(c)	startup frame	startup frame	missing
(d)	sync frame	startup frame	missing
(e)	startup frame	sync frame	missing
(f)	startup frame	startup frame	sync frame
(g)	short noise	startup frame	missing
(h)	startup frame	short noise	missing
(i)	startup frame	startup frame	short noise

4) It is verified (LT) that the IUT sends startup frames in slot 2 of

   (a, d, g) cycle 4,

   (b, e, h) cycles 4 and 5 and

   (c, f, i) cycles 4, 5 and 6.

5) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) and that the coldstart abort indicator is false

   (a, d, g) in cycle 5, 500 µT after cycle start and before NIT,

   (b, e, h) in cycle 6, 500 µT after cycle start and before NIT or

   (c, f, i) in cycle 7, 500 µT after cycle start and before NIT.

Figure 143 depicts the missing or incomplete startup frame in *POC:coldstart join* – missing startup frame in cycle 5.

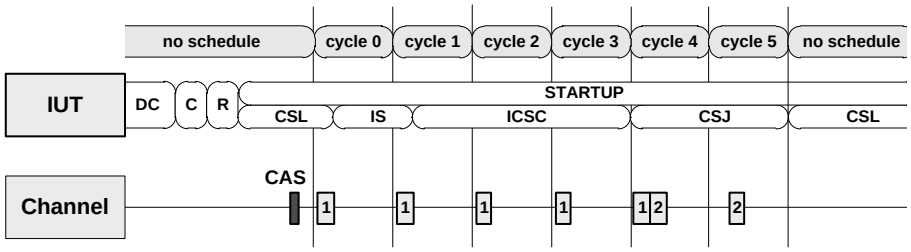


Figure 143 — Missing or incomplete startup frame in POC:coldstart join – missing startup frame in cycle 5

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts the startup and enters the *POC:coldstart listen* state when no or incomplete startup frame is received within one cycle while the IUT is in the *POC:coldstart join* state.

7.6.8.2 Startup frames

— Test purpose

Verify startup with the IUT as coldstarter but with coldstart inhibited. The IUT should enter the *POC:normal active* state after reception of at least one startup frame in 3 consecutive cycles while the IUT is in the *POC:coldstart join* state.

— Applicability

SC, DC.

— Configuration

All basic configurations.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 2 of cycles 0 to 3.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

10) It is checked (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

This test has to be performed repeatedly with the LT simulating (a) 1 and (b) 14 startup frames in adjacent slots starting with slot 2 in cycles 4 to 7.

1) In cycles 4 to 7, the LT simulates startup frames

   (a) in slot 2 and

   (b) in 14 adjacent slots starting with slot 2.

2) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

3) It is verified (LT) that the IUT sends startup frames in slot 1 of cycles 4 to 7.

4) In cycle 7, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 144 depicts the startup frames in *POC:coldstart join*, startup frames in slot 2.

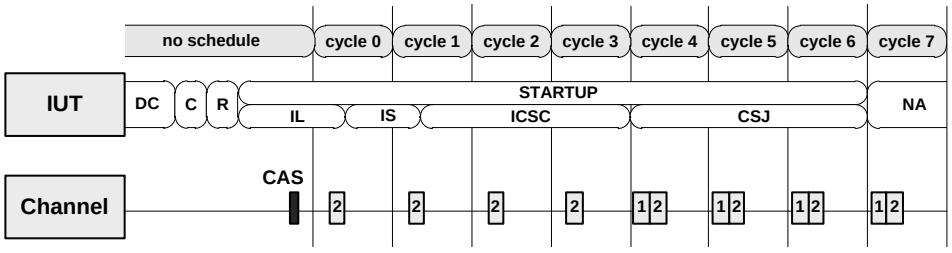


Figure 144 — Startup frames in POC:coldstart join, startup frames in slot 2

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT passes the *POC:coldstart join* state and enters the *POC:normal active* state if at least one startup frame is received in 3 consecutive cycles.

7.6.8.3 Deviation outside bounds

— Test purpose

Verify correct behaviour with the IUT as coldstarter when the measured deviation exceeds clock correction bounds during *POC:coldstart join* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 313.

Table 313 — Modification to basic configurations for coldstart join – deviation outside bounds

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
pClusterDriftDamping [µT]	0		
pKeySlotID	2		
pOffsetCorrectionOut [µT]	32		

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART 2 000 µT after the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command ALLOW_COLDSTART.

7) It is checked (UT) that the IUT is in the *POC:coldstart listen* (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) state 2 500 µT after the CHI command RUN.

8) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

9) The LT simulates a startup frame in slot 1 of cycle 0.

10) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

11) The LT simulates a startup frame in slot 1 of cycle 1.

12) It is checked (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) The LT simulates startup frames in slot 1 of cycles 2 and 3.

2) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

3) The LT simulates a startup frame in slot 1 of cycle 4.

4) It is verified (LT) that the IUT sends a startup frame in slot 2 of cycle 4.

5) The LT simulates a startup frame in slot 1 of cycle 5 with a deviation Δt = 3 * *pOffsetCorrectionOut*.

6) It is verified (LT) that the IUT sends a startup frame in slot 2 of cycle 5.

7) In cycle 6 of the LT, 500 µT after LT cycle start and before LT NIT, it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

Figure 145 depicts the deviation outside bounds in *POC:coldstart join*.

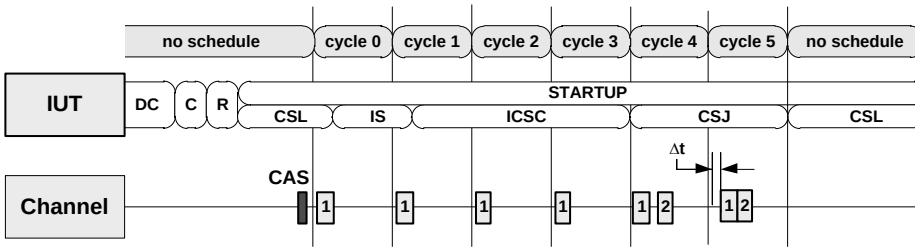


Figure 145 — Deviation outside bounds in POC:coldstart join

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the startup and enters the *POC:coldstart listen* state when the deviation of the startup frame exceeds clock synchronisation bounds while the IUT is in the *POC:coldstart join* state.

7.6.9 Integration consistency check

7.6.9.1 Deviation exceeds clock correction bounds

— Test purpose

Verify correct behaviour of the IUT as integrating node when the measured deviation exceeds clock correction limits during *POC:integration consistency check* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 314.

Table 314 — Modification to basic configurations for integration consistency check – deviation exceeds clock correction bounds

Parameter	Modification to Basic Configurations		
	1a & 1b	2a & 2b	3
pClusterDriftDamping [µT]	0		
pKeySlotID	4		
pKeySlotUsedForStartup	false		
pOffsetCorrectionOut [µT]	43		

— Preamble (setup state)

The LT simulates three coldstarters (one leading, two following) for this test.

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 1 of cycles 0 and 1.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates startup frames in slot 1 of cycles 2 to 4.

3) The LT simulates startup frames in slots 2 and 3 of cycle 4.

4) It is verified (LT) that the IUT does not send anything in cycles 0 to 6.

5) In cycle 5, the LT shifts its simulated frames by $\Delta t = 3 * pOffsetCorrectionOut$.

6) The LT simulates startup frames in slots 1, 2 and 3 of cycle 5.

7) In cycle 6 of the LT, 500 µT after LT cycle start and before LT NIT, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) and that the coldstart abort indicator is false.

Figure 146 depicts the deviation exceeds clock correction bounds in *POC:integration consistency check*.

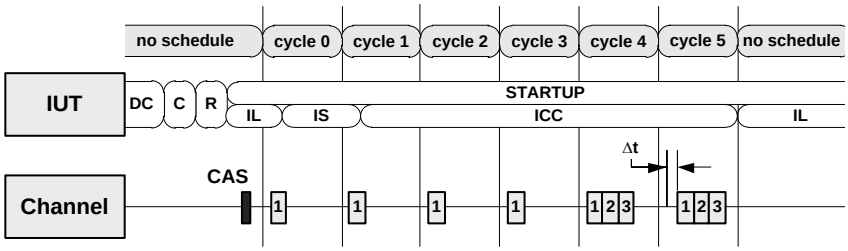


Figure 146 — Deviation exceeds clock correction bounds in POC:integration consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the integration attempt and enters the *POC:integration listen* state when the deviation of the startup frames exceeds clock synchronisation bounds while the IUT is in the *POC:integration consistency check* state.

7.6.9.2 Missing or incomplete startup frames

— **Test purpose**

Verify the correct startup with the IUT as integrating node and with no or incomplete startup frame during *POC:integration consistency check* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 315.

Table 315 — Modification to basic configurations for integration consistency check – missing or incomplete startup frames

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 1 of cycles 0 and 1.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates

   (a) idle in cycle 2.

   (b) a short noise (low phase of length 1 * gdBit) in slot 1 of cycle 2.

   (c) a startup frame in slot 1 of cycle 2 and a short noise (low phase of length 1 * gdBit) in slot 1 of cycle 3.

3) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*)

   (a, b) 4,5 * *gdCycle* ± 0,1 * *gdCycle or*

(c) 5,5 * *gdCycle* ± 0,1 * *gdCycle*

after the CHI command RUN.

Figure 147 depicts the missing or incomplete startup frames in *POC:integration consistency check* – missing startup frame in.

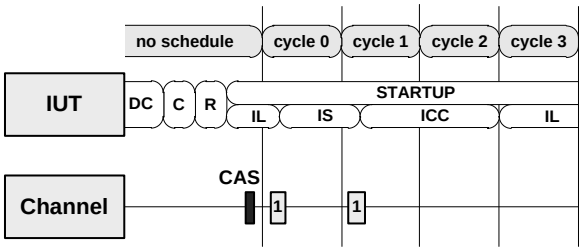


Figure 147 — Missing or incomplete startup frames in POC:integration consistency check – missing startup frame in cycle 2

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts the integration attempt and enters the *POC:integration listen* state when no or incomplete startup frame is received within one cycle while the IUT is in the *POC:integration consistency check* state.

7.6.9.3 Sync frames

— Test purpose

Verify correct startup with the IUT as integrating node when only sync frames are received in the *POC:integration consistency check* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 316.

Table 316 — Modification to basic configurations for integration consistency check – sync frames

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT begins to simulate startup frames in slots 1, 2 and 3 of cycles 7 through 9 0,2 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates sync frames in slots 1, 2 and 3 of cycle 10.

3) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 4,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 148 depicts the sync frames in *POC:integration consistency check*.

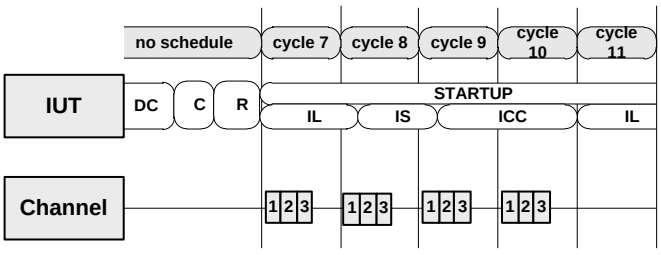


Figure 148 — Sync frames in POC:integration consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts the integration attempt and enters the *POC:integration listen* state when only sync frames and no startup frames are received within one cycle while the IUT is in the *POC:integration consistency check* state.

7.6.9.4 Single startup frame or incomplete second startup frame

— Test purpose

Verify correct startup with the IUT as integrating node when startup frames from a single coldstarter or incomplete startup frames from a second coldstarter are received until cycle 5.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 317.

Table 317 — Modification to basic configurations for integration consistency check – single startup frame or incomplete second startup frame

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 1 of cycles 0 and 1.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— Test execution

1) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates startup frames in slot 1 of cycles 2, 3.

3) The LT simulates

   (a) a startup frame in slot 1 of cycle 4.

   (b) a startup frame in slot 1 of cycle 4 and a short noise in slot 2 of cycle 4.

   (c) startup frames in slot 1 and 2 of cycle 4, a startup frame in slot 1 of cycle 5 and a short noise in slot 2 of cycle 5.

   (d) startup frames in slot 1 and 2 of cycles 4 and 5, a startup frame in slot 1 of cycle 6 and a short noise in slot 2 of cycle 6.

   (e) startup frames in slot 1 and 2 of cycles 4 to 6, a startup frame in slot 1 of cycle 7 and a short noise in slot 2 of cycle 7.

4) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*)

   (a, b) in cycle 5, 500 µT after cycle start and before NIT.

   (c) in cycle 6, 500 µT after cycle start and before NIT.

   (d) in cycle 7, 500 µT after cycle start and before NIT.

   (e) in cycle 8, 500 µT after cycle start and before NIT.

5) It is verified (LT) that the IUT does not send anything

   (a, b) in cycles 0 to 5.

   (c) in cycles 0 to 6.

   (d) in cycles 0 to 7.

   (e) in cycles 0 to 8.

Figure 149 depicts the single startup frame or incomplete second startup frame in *POC:integration consistency check* – single startup frame.

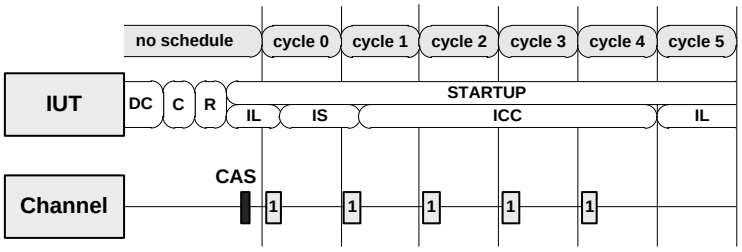


Figure 149 — Single startup frame or incomplete second startup frame in POC:integration consistency check – single startup frame

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the integration attempt and enters the *POC:integration listen* state when the IUT is in the *POC:integration consistency check* state and two startup frames are necessary, but only a single startup frame or a single startup frame combined with an incomplete second startup frame is received.

7.6.9.5 Integration into starting cluster

— **Test purpose**

Verify integration of the IUT into a coldstarting cluster. Verify that the IUT does not provide received frames to the UT in the startup states. Verify that the IUT does not transmit frames during startup.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 318.

Table 318 — Modification to basic configurations for integration consistency check – integration into starting cluster

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

A receive buffer is assigned to slot 1 in the IUT for the available channel(s).

— **Preamble (setup state)**

The LT simulates three coldstarters (one leading, two following) for this test.

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 1 of cycles 0 and 1.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates startup frames in slot 1 of cycles 2 to 8.

3) The LT simulates startup frames in slots 2 and 3 of the cycles 4 to 8.

4) In the NIT of cycle 7, it is verified (UT) that the receive buffer assigned to slot 1 is not updated, i.e. the receive buffer still holds its initial contents, and that the valid frame indicator signals no reception of a valid frame in the aggregated channel status of the IUT. The IUT does not process the LT's startup frame in slot 1 during startup.

5) In cycle 8, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

6) It is verified (LT) that the IUT only sends a frame in slot 4 of cycle 8 holding cycle count 8 and frame ID 4. The sync frame indicator is set to '1', the startup frame indicator is set to '0' and that the IUT does not send any frame before cycle 8.

Figure 150 depicts the integration into starting cluster in *POC:integration consistency check*.

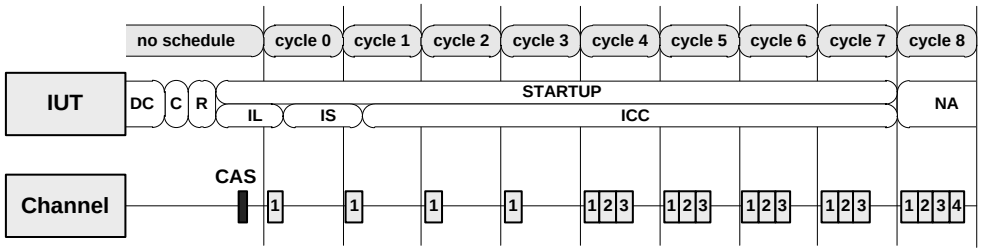


Figure 150 — Integration into starting cluster in POC:integration consistency check

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT passes the *POC:integration consistency check* state and enters the *POC:normal active* state if at least two startup frames are received within two consecutive even / odd cycle pairs. The IUT does not provide received frames to the UT during startup. The IUT does not transmit frames during startup.

7.6.9.6 Integration into running cluster

— Test purpose

Verify integration of the IUT into a running cluster.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 319.

Table 319 — Modification to basic configurations for integration consistency check – integration into running cluster

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT begins to simulate startup frames in slots 2 and 3 of cycles 1 and 2 0,2 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— Test execution

This test has to be performed repeatedly with the LT simulating (a) 2 and (b) 15 startup frames in adjacent slots starting at slot 2.

1) In cycles 3 to 8, the LT simulates startup frames

   (a) in slots 2 and 3 and

   (b) in 15 adjacent slots starting in slot 2.

2) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

3) In cycle 8, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

4) It is verified (LT) that the IUT sends a frame in slot 1 of cycle 8 holding cycle count 8 and frame ID 1. The sync frame indicator and the startup frame indicator are set to '0'.

Figure 151 depicts the integration into running cluster in *POC:integration consistency check* – two startup frames.

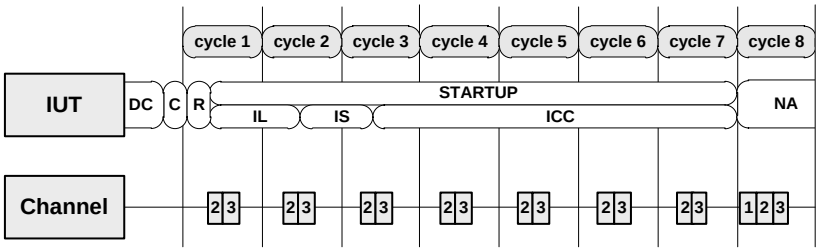


Figure 151 — Integration into running cluster in POC:integration consistency check – two startup frames

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT passes the *POC:integration consistency check* state and enters the *POC:normal active* state if at least two startup frames are received within two consecutive even / odd cycle pairs.

7.6.9.7 Startup frame missing in reference slot

— Test purpose

Verify correct startup with the IUT as integrating node when the startup frame in the reference slot is replaced by a startup frame in another slot. Only one startup frame is received per cycle.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 320.

Table 320 — Modification to basic configurations for integration consistency check – startup frame missing in reference slot

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

8) The LT simulates startup frames in slot 1 of cycles 0 and 1.

9) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 1 of cycle 2.

3) The LT simulates a startup frame in slot 2 of cycle 3.

4) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 5,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

5) It is verified (LT) that the IUT does not send anything during test execution.

Figure 152 depicts the startup frame missing in reference slot in *POC:integration consistency check*.

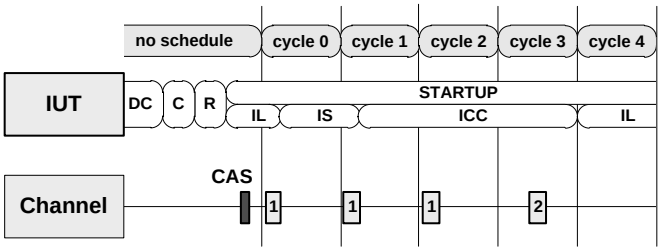


Figure 152 — Startup frame missing in reference slot in POC:integration consistency check

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT aborts the integration attempt and enters the *POC:integration listen* state when the startup frame in the reference slot is replaced by a startup frame in another slot while the IUT is in the *POC:integration consistency check* state.

7.6.9.8 Integration into running cluster with missing startup frame

— Test purpose

Verify integration of the IUT into a running cluster with one missing startup frame while the IUT is in the *POC:integration consistency check* state.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 321.

Table 321 — Modification to basic configurations for integration consistency check – integration into running cluster with missing startup frame

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN 2 000 µT after the CHI command CONFIG_COMPLETE.

6) It is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2 500 µT after the CHI command RUN.

7) The LT begins to simulate startup frames for slots 2 and 3 of cycles 1 and 2 0,2 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is checked (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

— **Test execution**

This test has to be performed repeatedly with the LT simulating (a) 2 and (b) 15 startup frames in adjacent slots starting at slot 2.

1) The LT simulates startup frames

   (a) in slots 2 and 3 and

   (b) in 15 adjacent slots starting with slot 2

   of cycle 3.

2) It is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

3) The LT simulates a startup frame in slot 2 of cycle 4.

4) The LT simulates startup frames

   (a) in slots 2 and 3 and

   (b) in 15 adjacent slots starting with slot 2

   of cycles 5 to 10.

5) In cycle 9, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

6) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

7) It is verified (LT) that the IUT sends a frame in slot 1 of cycle 10 holding cycle count 10 and frame ID 1. The sync frame indicator and the startup frame indicator are set to '0'.

Figure 153 depicts the integration into running cluster with missing startup frame in *POC:integration consistency check*.

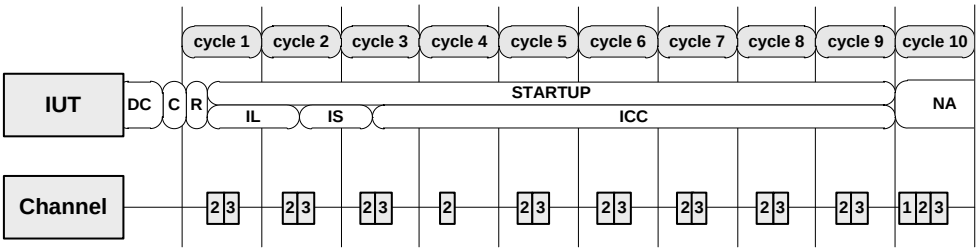


Figure 153 — Integration into running cluster with missing startup frame in POC:integration consistency check

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT passes the *POC:integration consistency check* state and enters the *POC:normal active* state at the end of an odd cycle if at least two startup frames are received within two consecutive even / odd cycle pairs.

7.7 Miscellaneous

7.7.1 Transmission conflict in static segment

— Test purpose

Verify correct handling of a transmission conflict in the static segment. The IUT shall recognize a transmission conflict, if decoding or channel idle detection is in progress when the IUT starts frame transmission. The IUT shall not abort its transmission upon detection of a transmission conflict. The IUT shall not process frames received while the IUT's transmission is ongoing.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 322.

Table 322 — Modification to basic configurations for transmission conflict in static segment

Parameter	Modification
gMaxWithoutClockCorrectionPassive	7
gMaxWithoutClockCorrectionFatal	7

For dual channel test execution a receive buffer and a transmit buffer are assigned to slot 1 / channel A&B in the IUT.

For single channel test execution a receive buffer and a transmit buffer are assigned to slot 1 on the available channel in the IUT.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

The transmit buffer of the IUT is committed for transmission in cycles 7 to 12 and for dual channel test execution also in cycles 13 and 14. The flags and the indicators shall be verified for all available channels. The LT does not simulate startup frames in slot 2 of cycles 7 to 14.

1) In the NIT of cycle 7, it is verified (UT) that the syntax error flag / indicator, the content error flag / indicator, the slot boundary violation flag / indicator, the valid frame flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.

2) In cycle 8, the LT simulates a low phase of length *gdActionPointOffset* in slot 1 starting 1 MT after the slot boundary.

3) In cycle 8, it is verified (LT) that the IUT transmits its frame in slot 1 disregarding the transmission conflict.

4) In the NIT of cycle 8, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator, the slot boundary violation flag / indicator and the valid frame flag / indicator are false and the transmission conflict flag / indicator is true in the transmit buffer status of slot 1 and in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.

5) In cycle 9, the LT simulates a low phase of length 1 *gdBit* in slot 1 starting 0,5 MT after the slot boundary.

6) In cycle 9, it is verified (LT) that the IUT transmits its frame in slot 1 disregarding the transmission before the static slot action point.

7) In the NIT of cycle 9, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator, the slot boundary violation flag / indicator, the valid frame flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.

8) In cycle 10, the LT simulates a startup frame (frame ID is 1, cycle count 10, null frame indicator is '0') with payload length 0 starting 1 MT after the static action point of slot 1.

9) In cycle 10, it is verified (LT) that the IUT transmits its frame in slot 1 disregarding the transmission conflict.

10) In the NIT of cycle 10, it is verified (UT) that the syntax error flag / indicator, the content error flag / indicator, the slot boundary violation flag / indicator, the valid frame flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot

status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification. Additionally, it is verified (UT) that the receive buffer assigned to slot 1 is not updated, i.e. the receive buffer still holds its initial contents (The IUT does not process the LT's startup frame in slot 1).

11) In cycle 11, it is verified (LT) that the IUT transmits its frame in slot 1.

12) In the NIT of cycle 11, it is verified (UT) that the syntax error flag / indicator, the content error flag / indicator, the slot boundary violation flag / indicator, the valid frame flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.

13) In cycle 12, the LT simulates a low phase of length *gdActionPointOffset* – 1,5 MT in slot 1 starting 1 MT after the slot boundary.

14) In cycle 12, it is verified (LT) that the IUT transmits its frame in slot 1 disregarding the transmission conflict during channel idle recognition.

15) In the dynamic segment of cycle 12, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator, the slot boundary violation flag / indicator and the valid frame flag / indicator are false and the transmission conflict flag / indicator is true in the transmit buffer status of slot 1 and in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.

16) In the NIT of cycle 12, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator are false in the transmit buffer status of slot 1.

For dual channel test execution add:

17) In cycle 13, the LT simulates a low phase of length *gdActionPointOffset* – 1,5 MT on channel A in slot 1 starting 1 MT after the slot boundary.

18) In cycle 13, it is verified (LT) that the IUT transmits its frame in slot 1 on both channels disregarding the transmission conflict during channel idle recognition on channel A.

19) In the NIT of cycle 13, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator, the slot boundary violation flag / indicator and the valid frame flag / indicator are false and the transmission conflict flag / indicator is true for channel A in the transmit buffer status of slot 1 and for channel A in the aggregated channel status. It is also verified (UT) that the syntax error flag / indicator, the content error flag / indicator, the slot boundary violation flag / indicator, the valid frame flag / indicator and the transmission conflict flag / indicator are false for channel B in the transmit buffer status of slot 1 and for channel B in the aggregated channel status.

20) In cycle 14, the LT simulates a low phase of length *gdActionPointOffset* – 1,5 MT on channel B in slot 1 starting 1 MT after the slot boundary.

21) In cycle 14, it is verified (LT) that the IUT transmits its frame in slot 1 on both channels disregarding the transmission conflict during channel idle recognition on channel B.

22) In the NIT of cycle 14, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator, the slot boundary violation flag / indicator and the valid frame flag / indicator are false and the transmission conflict flag / indicator is true for channel B in the transmit buffer status of slot 1 and for channel B in the aggregated channel status. It is also verified (UT) that the syntax error flag /

indicator, the content error flag / indicator, the slot boundary violation flag / indicator, the valid frame flag / indicator and the transmission conflict flag / indicator are false for channel A in the transmit buffer status of slot 1 and for channel A in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT does not stop frame transmission and sets the proper indication upon detection of a transmission conflict in the static segment. The IUT recognizes a transmission conflict, if decoding or channel idle detection is in progress when the IUT starts frame transmission. The IUT does not process a received frame causing a transmission conflict.

7.7.2 Transmission conflict in symbol window

— **Test purpose**

Verify correct handling of a transmission conflict in the symbol window. The IUT shall recognize a transmission conflict, if decoding or channel idle detection is in progress when the IUT starts symbol transmission. The IUT shall not abort its transmission upon detection of a transmission conflict.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations.

— **Preamble (setup state)**

Preamble II.

— **Test execution**

The flags and the indicators shall be verified for all available channels.

1) In the NIT of cycle 7, it is verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are false in the symbol window status and in the aggregated channel status.

2) The UT requests the IUT to transmit an MTS in the symbol window of cycle 8.

3) In cycle 8, the LT simulates a low phase of length *gdSymbolWindowActionPointOffset* in the symbol window starting 1 MT after the symbol window start.

4) It is verified (LT) that the IUT transmits its MTS in the symbol window disregarding the transmission conflict.

5) In the NIT of cycle 8, it is verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are true in the symbol window status and in the aggregated channel status.

6) The UT requests the IUT to transmit an MTS in the symbol window of cycle 9.

7) In cycle 9, the LT simulates a low phase of length 1 *gdBit* in the symbol window starting 0,5 MT after the symbol window start.

8) It is verified (LT) that the IUT transmits its MTS in the symbol window disregarding the transmission before the action point.

9) In the NIT of cycle 9, it is verified (UT) that the syntax error flag / indicator is false and the transmission conflict flag / indicator is false in the symbol window status and in the aggregated channel status.

10) The UT requests the IUT to transmit an MTS in the symbol window of cycle 10.

11) In cycle 10, the LT simulates a low phase of length *gdSymbolWindowActionPointOffset* – 1,5 MT in the symbol window starting 1 MT after the symbol window start.

12) It is verified (LT) that the IUT transmits its MTS in the symbol window disregarding the transmission conflict during channel idle recognition.

13) In the NIT of cycle 10, it is verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are true in the symbol window status and in the aggregated channel status.

14) The UT requests the IUT to transmit an MTS in the symbol window of cycle 11.

15) In cycle 11, it is verified (LT) that the IUT transmits its MTS in the symbol window.

16) In the NIT of cycle 11, it is verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are false in the symbol window status and in the aggregated channel status.

For dual channel test execution add:

17) The UT requests the IUT to transmit an MTS in the symbol window of cycle 12.

18) In cycle 12, the LT simulates a low phase of length *gdSymbolWindowActionPointOffset* – 1,5 MT on channel A in the symbol window starting 1 MT after the symbol window start.

19) It is verified (LT) that the IUT transmits its MTS on both channels in the symbol window disregarding the transmission conflict during channel idle recognition.

20) In the NIT of cycle 12, it is verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are true for channel A in the symbol window status and for channel A in the aggregated channel status. It is also verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are false for channel B in the symbol window status and for channel B in the aggregated channel status.

21) The UT requests the IUT to transmit an MTS in the symbol window of cycle 13.

22) In cycle 13, the LT simulates a low phase of length *gdSymbolWindowActionPointOffset* – 1,5 MT on channel B in the symbol window starting 1 MT after the symbol window start.

23) It is verified (LT) that the IUT transmits its MTS on both channels in the symbol window disregarding the transmission conflict during channel idle recognition.

24) In the NIT of cycle 13, it is verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are true for channel B in the symbol window status and for channel B in the aggregated channel status. It is also verified (UT) that the syntax error flag / indicator and the transmission conflict flag / indicator are false for channel A in the symbol window status and for channel A in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT does not stop MTS transmission and sets the proper indication upon detection of a transmission conflict in the symbol window. The IUT recognizes a transmission conflict, if decoding or channel idle detection is in progress when the IUT starts symbol transmission.

7.7.3 Traffic in transmission slot

— **Test purpose**

Verify correct handling of traffic in the IUT's transmission slot. The IUT shall decode and indicate additional traffic within its transmission slot.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 323.

Table 323 — Modification to basic configurations for traffic in transmission slot

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdStaticSlot [MT]	42	40	68
gMaxWithoutClockCorrectionPassive	7		
gMaxWithoutClockCorrectionFatal	7		
gNumberOfStaticSlots	70	37	21
gPayloadLengthStatic [two-byte word]	1		
gdNIT [MT]	49	17	33

— **Preamble (setup state)**

Preamble II.

— **Test execution**

1) The LT does not simulate startup frames in slot 2 of cycles 7 to 12.

2) In the NIT of cycle 7, it is verified (UT) that the valid frame flag / indicator, the syntax error flag / indicator, the content error flag / indicator, the boundary violation flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status.

3) In cycle 8, the LT simulates an additional correct frame in slot 1 (payload preamble indicator, sync frame indicator and startup frame indicator are set to '0', null frame indicator is set to '1', frame ID = 1, cycle count = 8, all payload bytes set to 0x00) starting 3 MT after the end of the IUT's frame. In the NIT of this cycle, it is verified (UT) that the valid frame flag / indicator is false, the syntax error flag / indicator is true, the content error flag / indicator, the boundary violation flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status.

4) In cycle 9, the LT simulates an additional frame with the frame ID set to 0 (payload preamble indicator, sync frame indicator and startup frame indicator are set to '0', null frame indicator is set to '1', frame ID = 0, cycle count = 9, all payload bytes set to 0x00) in slot 1 starting 3 MT after the end of the IUT's frame. In the NIT of this cycle, it is verified (UT) that the valid frame flag / indicator is false, the syntax error flag / indicator is true, the content error flag / indicator is false, the boundary violation flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status.

5) In cycle 10, the LT simulates an additional frame (payload preamble indicator, sync frame indicator and startup frame indicator are set to '0', null frame indicator is set to '1', frame ID = 1, cycle count = 10, all payload bytes set to 0x00) starting in slot 1 and ending in slot 2 (the last byte of the frame CRC starts at the boundary to slot 2). In the NIT of this cycle, it is verified (UT) that the valid frame flag / indicator, the syntax error flag / indicator and the content error flag / indicator are false, the boundary violation flag / indicator is true and the transmission conflict flag / indicator is false in the transmit buffer status of slot 1 and in the aggregated channel status.

6) In cycle 11, the LT simulates an additional frame (payload preamble indicator, sync frame indicator and startup frame indicator are set to '0', null frame indicator is set to '1', frame ID = 1, cycle count = 11, all payload bytes set to 0x00) with a bit flip in the last bit of the frame CRC in slot 1 starting 3 MT after the end of the IUT's frame. In the NIT of this cycle, it is verified (UT) that the valid frame flag / indicator is false, the syntax error flag / indicator is true, the content error flag / indicator, the boundary violation flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status.

7) In the NIT of cycle 12, it is verified (UT) that the valid frame flag / indicator, the syntax error flag / indicator, the content error flag / indicator, the boundary violation flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator, the syntax error flag / indicator, the content error flag / indicator, the boundary violation flag / indicator and the transmission conflict flag / indicator in the transmit buffer status and in the aggregated channel status are set accordingly.

7.7.4 Clock sync startup

7.7.4.1 Second startup frame missing (1)

— **Test purpose**

Verify correct startup with the IUT as integrating node when the second startup frame is missing. The IUT shall base the second startup attempt – after startup abortion due to the missing second startup frame in slot 1 / cycle 1 – on the startup frame in slot 1.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 324.

Table 324 — Modification to basic configurations for clock sync startup – second startup frame missing (1)

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

The IUT is configured to transmit a sync frame in slot 4 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 1 of cycle 0.

3) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

5) The LT simulates startup frames in slot 1 of cycles 2 and 3.

6) The LT simulates startup frames in slot 1 and slot 2 of cycles 4 to 10.

7) In cycle 4, 500 µT after start of slot 2 and before cycle end, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

8) In cycle 5, 500 µT after start of slot 2 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

9) It is verified (LT) that the IUT does not send anything in cycles 0 to 9.

10) In cycle 10, 500 µT after cycle start end before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

11) It is verified (LT) that the IUT sends a frame in slot 4 of cycle 10.

Figure 154 depicts the second startup frame missing (1).

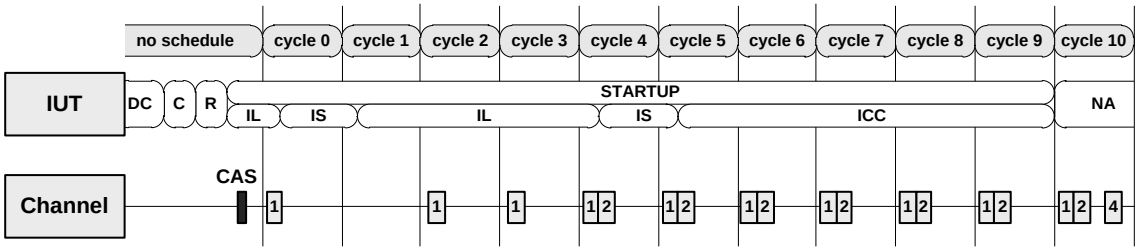


Figure 154 — Second startup frame missing (1)

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sends frames accordingly and enters the correct states. After abortion of the first startup attempt due to the missing startup frame in cycle 1, the IUT ignores the following startup frame with frame ID 1 in cycle 2. Thereafter, the IUT accepts the startup frame with frame ID 1 again in the next even cycle and finishes the startup successfully.

7.7.4.2 Second startup frame with wrong cycle count

— Test purpose

Verify correct startup with the IUT as integrating node when the second startup frame holds a wrong cycle count. The IUT shall abort the startup because the second startup frame is invalid due to a wrong cycle count.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 325.

Table 325 — Modification to basic configurations for clock sync startup – second startup frame with wrong cycle count

Parameter	Modification
pKeySlotID	4
pKeySlotUsedForStartup	false

The IUT is configured to transmit a sync frame in slot 4 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 1 of cycle 0.

3) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) The LT simulates a startup frame in slot 1 of cycle 1 with the cycle count set to 3.

5) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

Figure 155 depicts the second startup frame with wrong cycle count.

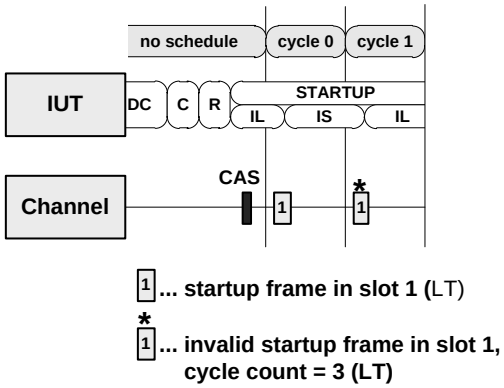


Figure 155 — Second startup frame with wrong cycle count

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the startup because the second startup frame is invalid due to the wrong cycle count.

7.7.4.3 Second startup frame in earlier slot

Verify correct startup with the IUT as integrating node when an additional valid odd startup frame appears in an earlier slot. The IUT shall proceed the startup and ignore the earlier valid startup frame.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 326.

Table 326 — Modification to basic configurations for clock sync startup – second startup frame in earlier slot

Parameter	Modification
pKeySlotID	3
pKeySlotUsedForStartup	false

The IUT is configured to transmit a sync frame in slot 3 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.
The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

4) The UT initiates the startup procedure with the CHI command RUN.

5) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a startup frame in slot 2 of cycle 0 *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

3) The LT simulates startup frames in slots 1 and 2 of cycles 1 to 6.

4) In cycle 1, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

5) It is verified (LT) that the IUT does not send anything in cycles 0 to 5.

6) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

7) It is verified (LT) that the IUT sends a frame in slot 3 of cycle 6.

Figure 156 depicts the second startup frame in earlier slot.

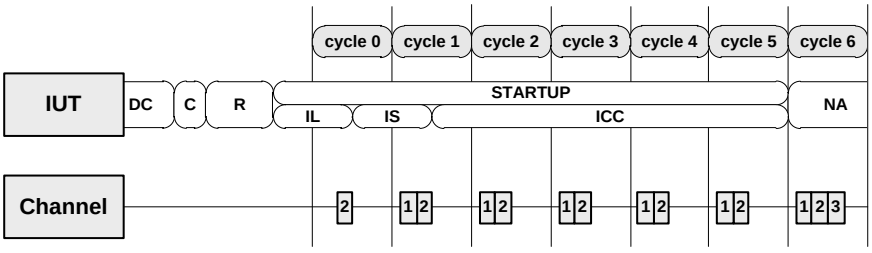


Figure 156 — Second startup frame in earlier slot

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs a successful integration on the startup frame in slot 2 and ignores the startup frame in slot 1 of cycle 1.

7.7.4.4 Sync frame counting

— Test purpose

Verify correct startup with the IUT as integrating node and correct sync frame counting for even / odd cycles on the available channel(s).

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 327.

Table 327 — Modification to basic configurations for clock sync startup – sync frame counting

Parameter	Modification
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false

The IUT is configured to transmit a normal frame in slot 1 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— Test execution

$t_{AW} \in [0, 500]$ µT

For dual channel test execution the test has to be performed in two instances. The LT generates its stimuli for instance 1 on channel A (selected channel is channel A) and for instance 2 on channel B (selected channel is channel B). For single channel tests the active channel is denoted as selected channel in this test case.

1) *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN, the LT starts to simulate startup frames

   (a) in slots 2 and 3 on the selected channel and

   (b) in 15 adjacent slots starting with slot 2 on the selected channel

   in cycles 0 to 6.

2) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

3) In cycle 1, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

4) In cycle 2, 500 µT after cycle start and before NIT, it is verified (UT) that the number of valid sync frames received or transmitted on the available channel(s) in the even communication cycle is 0.

5) In cycle 2, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the sync frame overflow indicator is not set and that

   (a) the number of valid sync frames received or transmitted on the selected channel in the even communication cycle is 2 and, for dual channel test execution, the number of valid sync frames received or transmitted on the other channel in the even communication cycle is 0 and

   (b) the number of valid sync frames received or transmitted on the selected channel in the even communication cycle is 15 and, for dual channel test execution, the number of valid sync frames received or transmitted on the other channel in the even communication cycle is 0.

6) In cycle 3, 500 µT after cycle start and before NIT, it is verified (UT) that the number of valid sync frames received or transmitted on the available channel(s) in the odd communication cycle is 0.

7) In cycle 3, within the interval t_{AW} after the offset correction start +10 * *gdMacrotick* / *pdMicrotick* µT, it is verified (UT) that the sync frame overflow indicator is not set and that

   (a) the number of valid sync frames received or transmitted on the selected channel in the odd communication cycle is 2 and, for dual channel test execution, the number of valid sync frames received or transmitted on the other channel in the odd communication cycle is 0 and

   (b) the number of valid sync frames received or transmitted on the selected channel in the odd communication cycle is 15 and, for dual channel test execution, the number of valid sync frames received or transmitted on the other channel in the odd communication cycle is 0.

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 157 depicts the sync frame counting – startup frames on channel A.

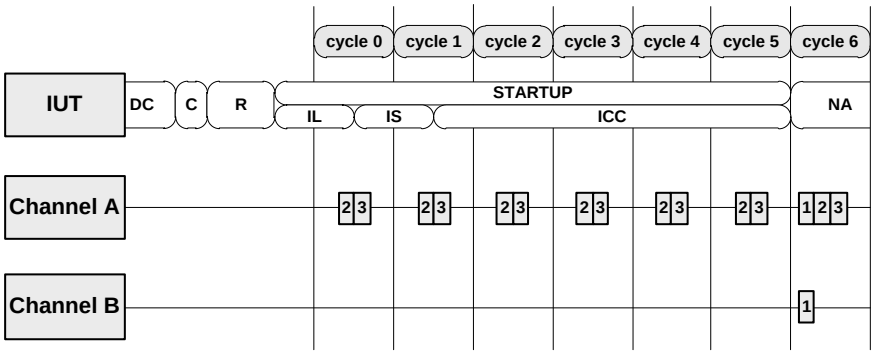


Figure 157 — Sync frame counting – startup frames on channel A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs a successful startup and counts the sync frames correctly.

7.7.4.5 Sync frame overflow

— Test purpose

Verify correct sync frame overflow indicator after reception of more than *gSyncFrameIDCountMax* sync / startup frames.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 328.

Table 328 — Modification to basic configurations for clock sync startup – sync frame overflow

Parameter	Modification			
	I	II	III	IV
gSyncFrameIDCountMax	2	8	15	15
pKeySlotUsedForStartup	false			
pKeySlotUsedForSync	false			

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— Test execution

1) *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN, the LT starts to simulate startup frames

   (I) in 3 adjacent slots,

(II) in 9 adjacent slots,

(III) in 16 adjacent slots and

(IV) in 15 adjacent slots

starting with slot 2 on the available channel(s) in cycles 0 to 6.

2) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

3) In cycle 1, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

4) In cycle 4, 500 µT after the end of the last slot holding a startup frame and before NIT, it is verified (UT) that

(I, II, III) the sync frame overflow indicator is set and

(IV) the sync frame overflow indicator is not set.

5) (I, II, III) In cycle 4, right after the verification of the sync frame overflow indicator, it is verified (UT) that the UT is able to reset the sync frame overflow indicator.

6) In cycle 5, 500 µT after end of the last slot holding a startup frame and before symbol window, it is verified (UT) that

(I, II, III) the sync frame overflow indicator is set and

(IV) the sync frame overflow indicator is not set.

The UT resets the sync frame overflow indicator after its verification.

7) In NIT of cycle 5, it is verified (UT) that the sync frame overflow indicator is not set.

8) In cycle 6, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

9) In cycle 6, 500 µT after end of the last slot holding a startup frame and before symbol window the UT resets the sync frame overflow indicator.

10) In cycle 7, the LT simulates startup frames in slots 2 and 3 on the available channel(s).

11) In cycle 8, the LT simulates startup frames

(I) in 2 adjacent slots,

(II) in 8 adjacent slots,

(III) in 15 adjacent slots and

(IV) in 14 adjacent slots

starting with slot 2 on the available channel(s).

12) In cycle 8, 500 µT after end of the last slot holding a startup frame and before NIT, it is verified (UT) that the sync frame overflow indicator is not set.

13) In cycle 9, the LT simulates startup frames

    (I) in 2 adjacent slots,

    (II) in 8 adjacent slots,

    (III) in 15 adjacent slots and

    (IV) in 14 adjacent slots

    starting with slot 3 on the available channel(s).

14) In cycle 9, 500 µT after end of the last slot holding a startup frame and before NIT, it is verified (UT) that

    (I, II, III) the sync frame overflow indicator is set and

    (IV) the sync frame overflow indicator is not set.

Figure 158 depicts the sync frame overflow – *gSyncFrameIDCountMax* = 2.

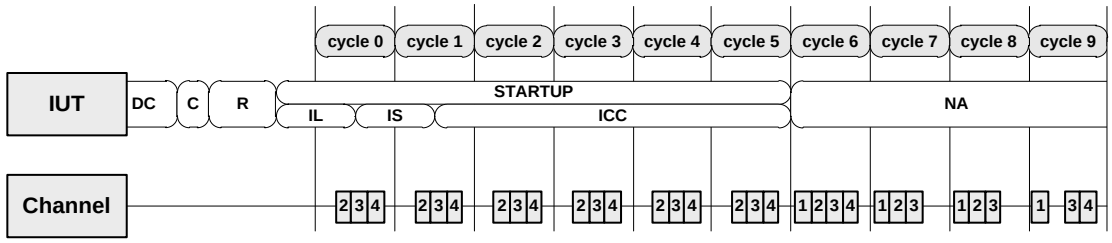


Figure 158 — Sync frame overflow – gSyncFrameIDCountMax = 2

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs a successful startup and indicates sync frame overflows correctly. The sync frame overflow indicator is set when more than *gSyncFrameIDCountMax* sync frames are received within a single cycle or an even / odd cycle pair. The UT is able to reset the sync frame overflow indicator.

7.7.4.6 Second startup frame too late

— Test purpose

Verify clock synchronisation startup when the second startup frame on either channel A or channel B arrives too late.

— Applicability

DC.

— Configuration

All basic configurations using the modifications as listed in Table 329.

Table 329 — Modification to basic configurations for clock sync startup – second startup frame too late

Parameter	Modification
pKeySlotID	2

The IUT is configured to transmit a startup frame in slot 2.

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— **Test execution**

1) The UT initiates the startup procedure with the CHI command RUN.

2) After 2 500 µT it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

3) The LT simulates a CAS on both channels 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

4) The LT simulates startup frames on both channels in slot 1 of cycle 0. The LT's cycle length is *pMicroPerCycle* + *pRateCorrectionOut* – *50 µT*.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

6) In cycle 1, the LT simulates its startup frame

   (a) on channel A delayed by Δt = 1,5 * *pMicroInitialOffset[A]* + *50 µT*

   (b) on channel B delayed by Δt = 1,5 * *pMicroInitialOffset[B]* + *50 µT*.

7) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is verified (LT) that the IUT does not send any frames during the test execution.

Figure 159 depicts the second startup frame too late – delayed on channel A.

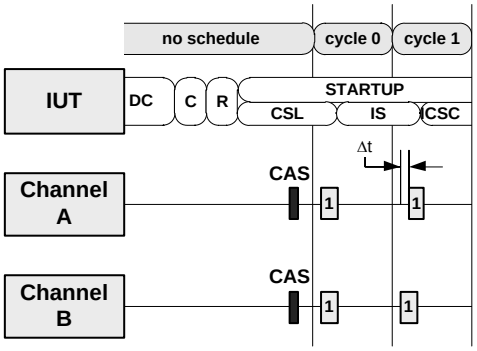


Figure 159 — Second startup frame too late – delayed on channel A

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT does not abort the clock synchronisation startup process due to the delayed second startup frame.

7.7.4.7 Clock synchronisation startup abortion (1)

— Test purpose

The IUT shall ignore valid even startup frames if their corresponding potential frame start is not detected for any reason.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 330.

Table 330 — Modification to basic configurations for clock sync startup – clock synchronisation startup abortion (1)

Parameter	Modification
pKeySlotID	2

The IUT is configured to transmit a startup frame in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— **Test execution**

For dual channel test execution the test has to be performed in two instances. The LT generates its stimuli for instance 1 on channel A and for instance 2 on channel B.

1) The UT initiates the startup procedure with the CHI command RUN.

2) After 2 500 µT it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

3) The LT simulates a CAS 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

4) In slot 1 of cycle 0, the LT simulates a startup frame with frame ID 1 and cycle count 0.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

6) In slot 1 of cycle 1, the LT simulates a startup frame with frame ID 2 and cycle count 0. The time interval between the LT's startup frames is *pMicroPerCycle* + *pRateCorrectionOut* + 4 µT.

7) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is verified (LT) that the IUT does not send any frames.

Figure 160 depicts the clock synchronisation startup abortion (1).

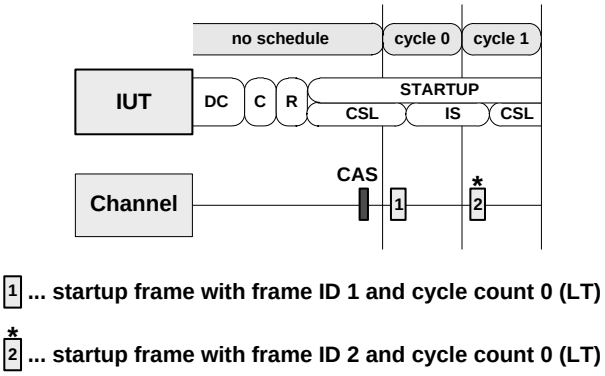


Figure 160 — Clock synchronisation startup abortion (1)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT ignores valid even startup frames if their corresponding potential frame start is not detected for any reason.

7.7.4.8 Clock synchronisation startup abortion (2)

— Test purpose

Verify clock synchronisation startup abortion due to a frame start on either channel A or channel B arriving before the valid second startup frame on the other channel. The IUT selects the channel on which the frame start is received for startup and ignores the valid second startup frame on the other channel. Eventually, the IUT aborts the clock synchronisation startup because no valid frame is received on the selected channel and the valid second startup frame is ignored.

— Applicability

DC.

— Configuration

All basic configurations using the modifications as listed in Table 331.

Table 331 — Modification to basic configurations for clock sync startup – clock synchronisation startup abortion (2)

Parameter	Modification
pKeySlotID	2

The IUT is configured to transmit a startup frame in slot 2.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— Test execution

1) The UT initiates the startup procedure with the CHI command RUN.

2) After 2 500 µT it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

3) The LT simulates a CAS on both channels 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

4) In slot 1 of cycle 0, the LT simulates a startup frame on both channels.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

6) In slot 1 of cycle 1, the LT simulates

   (a) the beginning of a frame (TSS, FSS and BSS) on channel B and, after a delay of Δt = 1 MT, a startup frame on channel A and

   (b) the beginning of a frame (TSS, FSS and BSS) on channel A and, after a delay of Δt = 1 MT, a startup frame on channel B.

7) It is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is verified (LT) that the IUT does not send any frames.

Figure 161 depicts the clock synchronisation startup abortion (2) – test variant (a).

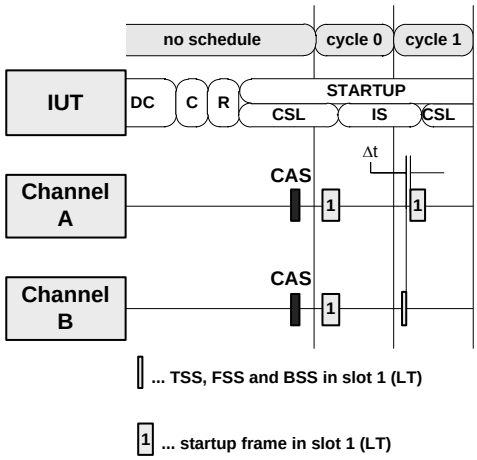


Figure 161 — Clock synchronisation startup abortion (2) – test variant (a)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the clock synchronisation startup process.

7.7.4.9 Clock synchronisation startup retrigger

— Test purpose

Verify clock synchronisation startup when the integration is requested twice while the IUT waits for the second startup frame. The IUT receives a frame start and a valid startup frame within the acceptance window of the second startup frame.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 332.

Table 332 — Modification to basic configurations for clock sync startup – clock synchronisation startup retrigger

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
pdListenTimeout [µT]	400 242	800 482	400 402	200 282	200 242
pKeySlotID	2				
pRateCorrectionOut [µT]	121	241	201	141	121
pClusterDriftDamping [µT]	0				
gdActionPointOffset [MT]	4	4	3	3	6
gdStaticSlot [MT]	35	35	33	33	61
gNumberOfStaticSlots	85	85	45	45	23
gNIT [MT]	14	14	12	12	55
pMacroInitialOffset [A,B][MT]	6	6	4	4	8

Δt = 30 gdBit for BC1a and BC1b and 25 gdBit for BC2a, BC2b and BC3.

The IUT is configured to transmit a startup frame in slot 2 on the available channel(s).

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

— Test execution

For dual channel test execution the test has to be performed in two instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B.

1) The UT initiates the startup procedure with the CHI command RUN.

2) After 2 500 µT it is verified (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

3) The LT simulates a CAS 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

4) The LT simulates startup frames in slot 1 of cycles 0 to 7. The LT's cycle length is *pMicroPerCycle*.

5) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

6) Δt before the static action point of slot 1 in cycle 1, the LT simulates the beginning of a frame (TSS, FSS and BSS) followed by an idle phase with a length of 16 gdBit and a valid startup frame.

7) It is verified (UT) that the IUT is in the *POC:integration coldstart check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_COLDSTART_CHECK*) 2 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

8) It is verified (UT) that the rate correction value is *vInterimRateCorrection* = (*gdTSSTransmitter* + *cdFSS* + *cdBSS* + 16) * *gdBit* / *pdMicrotick* – 2 * *gdMacrotick* / *pdMicrotick* + ξ_{IUT} µT and *vInterimOffsetCorrection* = 0
3 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

9) In cycle 4, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:coldstart join* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_JOIN*).

10) In cycle 7, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

Figure 162 depicts the clock synchronisation startup retrigger.

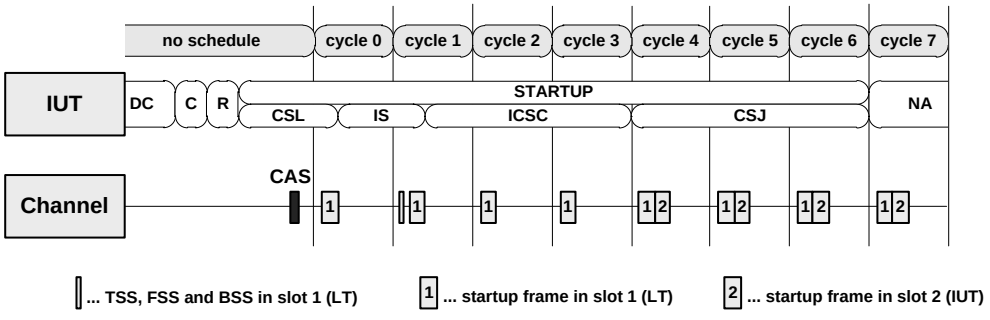


Figure 162 — Clock synchronisation startup retrigger

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT performs the startup accordingly. The IUT sets up the clock correctly and reaches *POC:normal active* state.

7.7.4.10 Startup frames violate accepted startup range in odd cycle

— Test purpose

Verify the correct handling of startup frames appearing outside of the accepted startup range during cycle 3.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 333.

Table 333 — Modification to basic configurations for clock sync startup – startup frames violate accepted startup range in odd cycle

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gPayloadLengthStatic [two-byte word]	0	0	0
pdAcceptedStartupRange [µT]	80	80	80
pClusterDriftDamping [µT]	0	0	0
pKeySlotID	3	3	3
pKeySlotUsedForStartup	false	false	false
pKeySlotUsedForSync	false	false	false
gdNIT [MT]	14	11	11
gdActionPointOffset [MT]	4	4	3
gdActionPointOffset [MT]	6	5	5
pMacroInitialOffset[B] [MT]	6	5	5

The IUT is an integrating node and is configured to send a static frame in slot 3.

The test has to be performed with a Δt of -(*pdAcceptedStartupRange* + 2) and +(*pdAcceptedStartupRange* + 2) µT.

— Preamble (setup state)

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Test execution**

1) The LT begins to simulate startup frames in slots 1 and 2 of cycles 0 to 4 0,5 * *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) In cycle 0, 500 µT after the start of slot 2 and before NIT, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

3) In cycle 1, 500 µT after the start of slot 2 and before NIT, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

4) The LT shifts its startup frames by ∆t in cycle 3.

5) In cycle 4 of the LT, 500 µT after the LT's cycle start, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) and, in the NIT of the LT's cycle 4, that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

Figure 163 depicts the startup frames violate accepted startup range in odd cycle; Δt = - (pdAcceptedStartupRange + 2) µT.

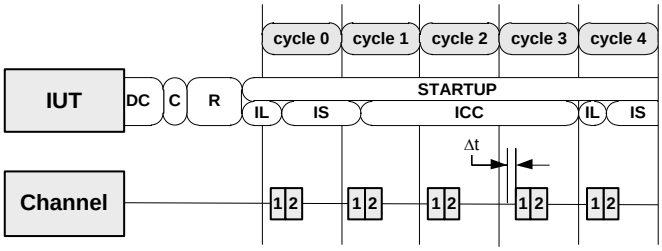


Figure 163 — Startup frames violate accepted startup range in odd cycle; Δt = - (pdAcceptedStartupRange + 2) µT

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT aborts the startup when startup frames outside of the accepted startup range are received within cycle 3.

7.7.4.11 Second startup frame missing (2)

— **Test purpose**

Verify correct startup with the IUT as integrating node when the second startup frame is missing. The IUT shall base the second startup attempt – after startup abortion due to the missing second startup frame in slot 1 / cycle 1 – on the startup frame in slot 4.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 334.

Table 334 — Modification to basic configurations for clock sync startup – second startup frame missing (2)

Parameter	Modification
pKeySlotID	3
pKeySlotUsedForStartup	false

The IUT is configured to transmit a sync frame in slot 3 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Test execution**

For dual channel test execution this test has to be performed in three instances. The LT generates startup frames in slots 1 and 4 for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a CAS *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) The LT simulates a startup frame in slot 1 of cycle 0.

3) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

4) It is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*) 2,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

5) The LT simulates startup frames in slot 1 and slot 4 of cycles 2 to 8.

6) In slot 3 of cycle 2, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

7) In slot 6 of cycle 2, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

8) In slot 3 of cycle 3, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

9) In slot 6 of cycle 3, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

10) It is verified (LT) that the IUT does not send anything in cycles 0 to 7.

11) In cycle 8, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

12) It is verified (LT) that the IUT sends a frame in slot 3 of cycle 8.

Figure 164 depicts the second startup frame missing (2).

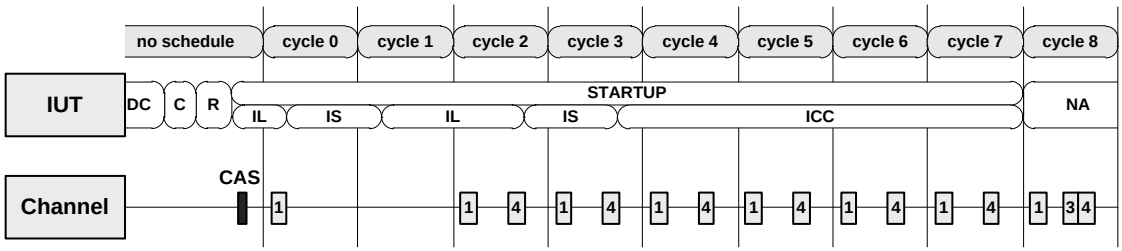


Figure 164 — Second startup frame missing (2)

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sends frames accordingly and enters the correct states. After abortion of the first startup attempt due to the missing startup frame in cycle 1, the IUT ignores the following startup frame with frame ID 1 in cycle 2 and accepts the startup frame with frame ID 4 in cycle 2. The IUT finishes the startup successfully.

7.7.4.12 Second valid startup frame too early

— **Test purpose**

Verify correct startup with the IUT as integrating node when the odd startup frame is received too early.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 335.

Table 335 — Modification to basic configurations for clock sync startup – second valid startup frame too early

Parameter	Modification
pKeySlotID	3
pKeySlotUsedForStartup	false

The IUT is configured to transmit a sync frame in slot 3 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Test execution**

For dual channel test execution this test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a startup frame in slot 2 of cycle 0 *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

3) In cycle 1, the LT starts the simulation of the startup frame in slot 2 *pRateCorrectionOut* + 4 µT earlier. I.e., the time interval between the LT's startup frames in cycles 0 and 1 is *pMicroPerCycle* – *pRateCorrectionOut* – 4 µT.

4) In cycle 2, within 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

5) The LT simulates startup frames in slot 2 of cycles 2 to 12.

6) In cycle 3, within 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

7) In cycle 4, within 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

8) In cycle 5, within 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

9) The LT simulates startup frames in slots 1 of cycles 8 to 12.

10) It is verified (LT) that the IUT does not send anything in cycles 0 to 11.

11) In cycle 12, within 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

12) It is verified (LT) that the IUT sends a frame in slot 3 of cycle 12.

Figure 165 depicts the second valid startup frame too early.

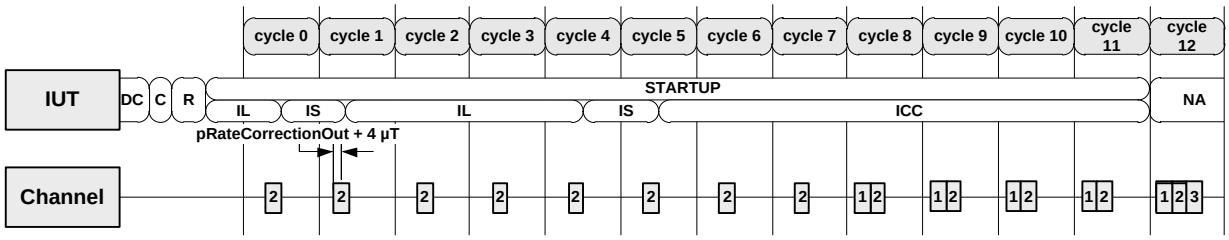


Figure 165 — Second valid startup frame too early

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT restarts integration after the odd startup frame in cycle 1 was received too early.

7.7.4.13 Second odd startup frame too late

— Test purpose

Verify correct startup with the IUT as integrating node when a valid odd startup frame appears too late. The IUT shall proceed the startup and ignore the frame ID that was too late.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 336.

Table 336 — Modification to basic configurations for clock sync startup – second odd startup frame too late

Parameter	Modification
pKeySlotID	3
pKeySlotUsedForStartup	false

The IUT is configured to transmit a sync frame in slot 3 on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*).

5) The UT initiates the startup procedure with the CHI command RUN.

6) After 2 500 µT it is checked (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

— **Test execution**

For dual channel test execution the test has to be performed in three instances. The LT generates its stimuli for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) The LT simulates a valid startup frame with ID 1 in slot 2 of cycle 0 *gdCycle* ± 0,25 * *gdCycle* after the CHI command RUN.

2) It is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*) 1,5 * *gdCycle* ± 0,1 * *gdCycle* after the CHI command RUN.

3) The LT simulates a startup frame in slot 2 of cycle 1.

4) In cycle 1, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

5) The LT simulates a startup frame in slot 1 of cycle 2.

6) In cycle 2, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

7) The LT simulates startup frames in slot 1 and 2 of cycle 3 to 10.

8) In cycle 3, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_LISTEN*).

9) In cycle 4, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:initialize schedule* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INITIALIZE_SCHEDULE*).

10) In cycle 5, 500 µT after start of slot 3 and before cycle end, it is verified (UT) that the IUT is in the *POC:integration consistency check* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *INTEGRATION_CONSISTENCY_CHECK*).

11) It is verified (LT) that the IUT does not send anything in cycles 0 to 9.

12) In cycle 10, 500 µT after cycle start and before cycle end, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

13) It is verified (LT) that the IUT sends a frame in slot 3 of cycle 10.

Figure 166 depicts the second odd startup frame too late.

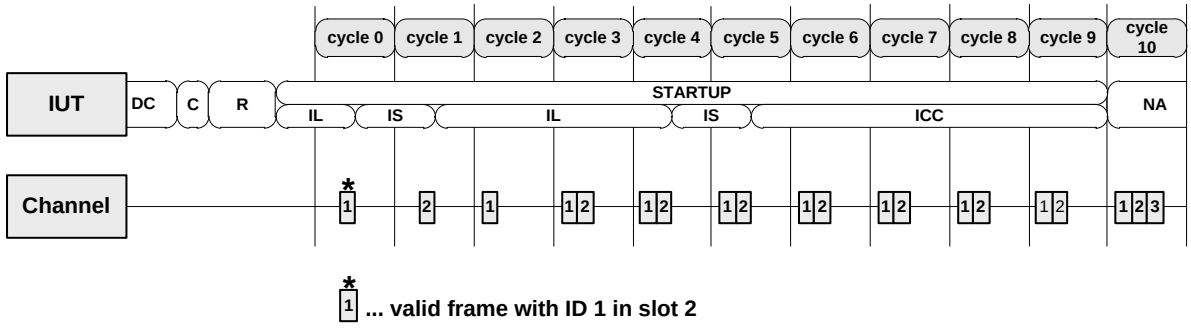


Figure 166 — Second odd startup frame too late

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT performs a successful integration beginning with cycle 4.

7.7.5 Frame format

— Test purpose

Verify correct frame format of static and dynamic frames sent by the IUT. Verify that the IUT uses the configured header CRCs and does not generate them by itself. Verify correct behaviour of TxEN signal. Verify correct generation of DTS with minimal length of 1 bit.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 337, Table 338 and Table 339.

Table 337 — Modification to basic configurations 1a and 1b for clock sync startup – frame format

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	97	99	101	97	99	101
gdMinislot [MT]	5			5		
gdMinislotActionPointOffset [MT]	2			2		
gdTSSTransmitter [gdBit]	11	12	13	11	12	13
gNumberOfMinislots	393			393		
gPayloadLengthStatic [two-byte word]	1			1		
gdNIT [MT]	18			18		
pClusterDriftDamping [µT]	0			0		
pDecodingCorrection [µT]	52	56	60	104	112	120
pKeySlotID	2			2		
pMacroInitialOffset[A,B] [MT]	6			6		
pMicroInitialOffset[A,B] [µT]	23	19	15	46	38	30

Table 338 — Modification to basic configurations 2a and 2b for clock sync startup – frame format

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdCASRxLowMax [gdBit]	80	82	84	80	82	84
gdMinislot [MT]	5			5		
gdMinislotActionPointOffset [MT]	2			2		
gdTSSTransmitter [gdBit]	6	7	8	6	7	8
gNumberOfMinislots	195			195		
gPayloadLengthStatic [two-byte word]	1			1		
gdNIT [MT]	16			16		
pClusterDriftDamping [µT]	2			0		
pDecodingCorrection [µT]	64	72	80	32	36	40
pKeySlotID	2			2		
pMacroInitialOffset[A,B] [MT]	4	5		4	5	
pMicroInitialOffset[A,B] [µT]	6	78	70	3	39	35

Table 339 — Modification to basic configuration 3 for clock sync startup – frame format

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdCASRxLowMax [gdBit]	72	74	74
gdMinislot [MT]	5		
gdMinislotActionPointOffset [MT]	2		
gdTSSTransmitter [gdBit]	4	5	
gNumberOfMinislots	202		
gPayloadLengthStatic [two-byte word]	1		
gdNIT [MT]	15		
pClusterDriftDamping [µT]	0		
pDecodingCorrection [µT]	48	56	
pKeySlotID	2		
pMacroInitialOffset[A,B] [MT]	5		
pMicroInitialOffset[A,B] [µT]	22	14	

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{FlStat}[µs] = (*gdTSSTransmitter* + *cdFSS* + 50 + *gPayloadLengthStatic* * 20 + 30 + *cdFES*) * *gdBit*.
n_{FlDyn}[µs] = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + 3 * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** * (*gdMinislot* * *gdMacrotick*).

The IUT is configured to transmit a startup frame in slot 2, a static frame in slot 10 and a dynamic frame in slot n_{dyn}. The dynamic payload length is 3 two-byte words. The static payload in slot 10 is 0x01 in byte 0 and 0x02 in byte 1, the dynamic payload in slot n_{dyn} 0x10 in byte 0, 0x20 in byte 1, ..., and 0x60 in byte 5. The IUT's startup frame is always transmitted as null frame.

— **Preamble (setup state)**

Preamble I with the modification that in cycle 6 the IUT transmits a startup frame with the null frame indicator set to '0'.

NOTE The measurements in this test execution shall be done by measuring a continuous bit stream length and not a difference between two time stamps. Therefore a tolerance of ±1 σT instead of ±2 σT is applied.

— **Test execution**

1) In cycle 8, the LT simulates its frame in slot 1 correctly.

2) In slot 1 of cycle 8, the UT configures the header CRC for the static frame in slot 10 to 0x123 and for the dynamic frame in slot n_{dyn} to 0x321. Those invalid header CRCs are used to verify that the IUT does not generate header CRCs by itself.

3) In cycle 8, it is verified (LT) that the IUT transmits its frames according to the data link layer specification, i.e. the TSS comprises a low phase of length *gdTSSTransmitter* * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT,

the FSS a high phase of length 1 * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT,
every BSS a high phase of length 1 * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT followed by a low phase of length 1 * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT,
the FES a low phase of length 1 * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT followed by a high phase of length 1 * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT and
every other bit within the frame a corresponding high or low phase of length 1 * *gdBit* * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT.

The static frames in slots 2 and 10 exhibit a length of n_{FIStat} * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT and the dynamic frame in slot n_{dyn} exhibits a length of n_{FIDyn} * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT.

It is also verified (LT) that the TxEN output and the TxD output become low simultaneously, at the beginning of the TSS and that the TxEN output stays low for a period of n_{FIStat} * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT for the static frames in slot 2 and 10 and (n_{FIDyn} + 1 * *gdBit*) * *pSamplesPerMicrotick* / *pdMicrotick* ± 1 σT for the dynamic frame in slot n_{dyn}.

The TxD outputs reflect the transmitted frames holding the respective values as shown in Table 340.

Table 340 — Expected transmission of the IUT at the TxD outputs

Frame element	Value				
	Slot 2	Slot 10	basic configuration		
			1a/1b	2a/2b	3
			Slot n_{dyn}		
Reserved Bit	0				
Payload Preamble Indicator	0	1	0		
Null Frame Indicator	0	1	1		
Sync Frame Indicator	1	0	0		
Startup Frame Indicator	1	0	0		
Frame ID	2	10	86	46	26
Payload Length	1	1	3		
Header CRC	0x026	0x123	0x321		
Cycle Count	8				
Data 0	0x00	0x01	0x10		
Data 1	0x00	0x02	0x20		
Data 2	–	–	0x30		
Data 3	–	–	0x40		
Data 4	–	–	0x50		
Data 5	–	–	0x60		
Frame CRC 0 on channel A	0xF3	0x64	0x47	0x96	0xB0
Frame CRC 1 on channel A	0x39	0x6D	0x43	0xC9	0x72
Frame CRC 2 on channel A	0xC1	0x70	0x80	0xD1	0xEB
Frame CRC 0 on channel B	0xD5	0x42	0xE6	0x37	0x11
Frame CRC 1 on channel B	0xB9	0xED	0xD9	0x53	0xE8
Frame CRC 2 on channel B	0x10	0xA1	0xBE	0xEF	0xD5

Each transition of the TxD and TxEN states of the frame shall be placed at the respective nominal position with an allowed deviation of ±1 σT.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT transmits static and dynamic frames exhibiting proper timing and logical values. The IUT transmits the configured header CRCs.

7.7.6 Transmission across slot boundary violation

— **Test purpose**

Verify correct behaviour of the IUT when a transmission across slot boundary violation occurs. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition to *POC:halt*, initiated by a fatal protocol error).

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 341, Table 342 and Table 343.

Table 341 — Modification to basic configurations 1a and 1b for transmission across slot boundary violation

Parameter	Modification to Basic Configurations							
	1a				1b			
	I	II	III	IV	I	II	III	IV
gdActionPointOffset [MT]	4				4			
gdCASRxLowMax [gdBit]	105				105			
gdMinislotActionPointOffset [MT]	4				4			
gdStaticSlot [MT]	24[a]				24[a]			
gdSymbolWindow [MT]	40			0	40			0
gdTSSTransmitter [gdBit]	15				15			
gNumberOfMinislots	323		0		323		0	
gNumberOfStaticSlots	85		206		85		206	
gdNIT [MT]	13		16	56	13		16	56
pDecodingCorrection [µT]	68				136			
pKeySlotID	1	85	206		1	85	206	
pLatestTx [Minislot]	188		0		188		0	
pMacroInitialOffset[A,B] [MT]	6				6			
pMicroInitialOffset[A,B] [µT]	7				14			

[a] intentional protocol constraints violation.

Table 342 — Modification to basic configurations 2a and 2b for transmission across slot boundary violation

Parameter	Modification to Basic Configurations							
	2a				2b			
	I	II	III	IV	I	II	III	IV
gdActionPointOffset [MT]	4				3			
gdCASRxLowMax [gdBit]	80				80			
gdMinislotActionPointOffset [MT]	4				3			
gdStaticSlot [MT]	23[a]				22[a]			
gdSymbolWindow [MT]	23			0	23			0
gdTSSTransmitter [gdBit]	6				6			
gNumberOfMinislots	204		0		204		0	
gNumberOfStaticSlots	45		106		45		106	
gdNIT [MT]	14		39	62	59		145	168
pDecodingCorrection [µT]	64				32			
pKeySlotID	1	45	106		1	45	106	
pLatestTx [Minislot]	101		0		101		0	
pMacroInitialOffset[A,B] [MT]	5				4			
pMicroInitialOffset[A,B] [µT]	6				3			

[a] intentional protocol constraints violation.

Table 343 — Modification to basic configuration 3 for transmission across slot boundary violation

Parameter	Modification to Basic Configuration 3			
	I	II	III	IV
gdActionPointOffset [MT]	3			
gdCASRxLowMax [gdBit]	74			
gdMinislotActionPointOffset [MT]	3			
gdStaticSlot [MT]	41[a]			
gdSymbolWindow [MT]	24			0
gdTSSTransmitter [gdBit]	5			
gNumberOfMinislots	198		0	
gNumberOfStaticSlots	25		57	
gdNIT [MT]	65		139	163
pDecodingCorrection [µT]	56			
pKeySlotID	1	25	57	
pLatestTx [Minislot]	68		0	
pMacroInitialOffset[A,B] [MT]	5			
pMicroInitialOffset[A,B] [µT]	14			

[a] intentional protocol constraints violation.

The IUT is configured to transmit a startup frame in the slot *pKeySlotID* on the available channel(s).

— **Preamble (setup state)**

1) The UT resets the IUT (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG (*vPOC!State* = *CONFIG*).

3) The UT configures the IUT with the given configuration.

4) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE (*vPOC!State* = *READY*). The UT waits 2 000 µT to assure completion of the CHI command.

5) The UT enables interrupt requests of the IUT for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests. The UT clears (resets) the interrupt status indication flags of the IUT for all interrupt sources.

6) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART. The UT waits 2 000 µT to assure completion of the CHI command.

7) The UT initiates the startup procedure with the CHI command RUN 2 * *cChannelIdleDelimiter* after the CHI command ALLOW_COLDSTART.

8) After 2 500 µT it is checked (UT) that the IUT is in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*).

9) It is checked (LT) that the IUT sends a CAS (low phase of length *gdTSSTransmitter* + *cdCAS*) starting *pdListenTimeout* + *cdCASActionPointOffset* * *gdMacrotick* / *pdMicrotick* ± 0,05 * *pdListenTimeout* µT after the UT has initiated the startup procedure via CHI command RUN.

10) It is checked (UT) that the IUT is in the *POC:coldstart collision resolution* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_COLLISION_RESOLUTION*) *pdListenTimeout* + (*gdTSSTransmitter* + *cdCAS*) * *gdBit* / *pdMicrotick* + 2 500 µT after the CHI command RUN.

— **Test execution**

1) It is verified (UT) that the IUT is in the *POC:halt* state (*vPOC!State* = *STARTUP* and *vPOC!FREEZE* = true) and that the transmission across slot boundary violation indicator(s) for the available channel(s) is(are) set *pdListenTimeout* + (*pKeySlotID* + 2) * *gdStaticSlot* * *gdMacrotick* / *pdMicrotick* + 2 500 µT after the CHI command RUN.

2) It is verified (LT) that the IUT transmits the frame header and the first payload bytes of its startup frame in static slot *pKeySlotID* on the available channel(s). All payload bits are expected to be 0. It is also verified (LT) that the IUT stops the ongoing transmission and that TxEN and TxD become high simultaneously at the end of slot *pKeySlotID* on the available channel(s).

3) It is verified (UT) that the IUT has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The IUT is already in *POC:halt* state.

— **Pass criteria**

The IUT detects the fatal protocol error, sets the transmission across slot boundary violation indicators and enters *POC:halt* state. The IUT indicates an interrupt request on the POC state transition to *POC:halt* state.

7.7.7 Clock correction failed counter reset in POC:normal active

— **Test purpose**

Verify that the clock correction failed counter is incremented and reset in odd cycles only while the IUT is in *POC:normal active* state.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 344.

Table 344 — Modification to basic configurations for clock correction failed counter reset in POC:normal active

Parameter	Modification
gMaxWithoutClockCorrectionPassive	3
gMaxWithoutClockCorrectionFatal	3

— **Preamble (setup state)**

Preamble II.

The LT simulates two following coldstarters. The LT simulates startup frames in slot 2 and in slot 3. The IUT is configured to transmit a startup frame in slot 1 and a static frame in slot 4.

The preamble is supplemented by the following statement:"In cycle 4, the LT begins to simulate startup frames in slot 2 and in slot 3.

— **Test execution**

1) In cycle 8, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 0.

2) In cycles 8 to 14, 500 µT after cycle start and before NIT, it is verified (UT) that the IUT is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that *vPOC!ErrorMode* is *ACTIVE*.

3) In cycle 9, the LT stops frame simulation.

4) In cycle 10, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 1.

5) In cycle 11, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 1.

6) In cycle 12, the LT resumes frame transmission.

7) In cycle 12, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 2.

8) In cycle 13, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 2.

9) In cycle 14, 500 µT after cycle start and before NIT, it is verified (UT) that the clock correction failed counter is *vClockCorrectionFailed* = 0.

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT increments and resets the clock correction failed counter in odd cycles only while in *POC:normal active* state.

7.7.8 WUDOP transmission across symbol window boundary

— Test purpose

Verify the correct handling when the IUT transmits a WUDOP crossing the end of the symbol window.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 345.

Table 345 — Modification to basic configurations for WUDOP transmission across symbol window boundary

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdSymbolWindowActionPointOffset [MT]	30	15	12

— Preamble (setup state)

Preamble II.

— Test execution

For dual channel test execution has to be performed in three instances. The UT configures the IUT to transmit a WUDOP for instance 1 on channel A, for instance 2 on channel B and for instance 3 on both channels.

1) In cycle 7, within 500 µT after the cycle start and before the dynamic segment, it is verified (UT) that *vPOC!Freeze* = false and that the transmission across slot boundary violation indicator(s) is(are) reset for the available channel(s).

2) In cycle 7, within 500 µT after the cycle start and before the dynamic segment, the UT requests the IUT to transmit a WUDOP in the symbol window of cycle 7.

3) Within 500 µT and 2 500 µT after symbol window end of cycle 7, it is verified (UT) that the IUT is in the *POC:halt* state (*vPOC!State* = *NORMAL_ACTIVE*), that *vPOC!Freeze* = true and that the transmission across slot boundary violation indicator(s) is(are) set for the channel(s) on which the IUT started WUDOP transmission.

— Postamble

None. The IUT is already in the *POC:halt* state.

— Pass criteria

The IUT enters the *POC:halt* state and sets *vPOC!Freeze* = true and the transmission across slot boundary violation indicator(s) accordingly.

7.7.9 Dynamic segment robustness

7.7.9.1 Resynchronization attempt after single short noise

— Test purpose

Verify slot counting in the dynamic segment without noise. Verify correct slot counting and blocking of frame transmission in the dynamic segment after the occurrence of noise with a length less than *cFrameThreshold*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 346, Table 347 and Table 348.

Table 346 — Modification to basic configurations 1a and 1b for dynamic segment robustness – resynchronization attempt after single short noise

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gdMinislot [MT]	6			6		
gdMinislotActionPointOffset [MT]	2			2		
gNumberOfMinislots	328			328		
gdNIT [MT]	15			15		
pLatestTx [Minislot]	284	283	282	284	283	282

Table 347 — Modification to basic configurations 2a and 2b for dynamic segment robustness – resynchronization attempt after single short noise

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gdMinislot [MT]	7			7		
gdMinislotActionPointOffset [MT]	3			3		
gNumberOfMinislots	140			140		
gdNIT [MT]	12			12		
pLatestTx [Minislot]	102	101	100	102	101	100

Table 348 — Modification to basic configuration 3 for dynamic segment robustness – resynchronization attempt after single short noise

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2
gdMinislot [MT]	7		
gdMinislotActionPointOffset [MT]	2	3	
gNumberOfMinislots	145		
gdNIT [MT]	10	11	
pLatestTx [Minislot]	69	68	67

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit frames in the dynamic slots n_{dyn} + 1 and n_{dyn} + 2 with the payload 0xAA in byte 1 and 0x55 in byte 2 in continuous transmission mode on the active channel(s).

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, it is verified (LT) that the IUT transmits its frames in the dynamic slots n_{dyn} + 1 and n_{dyn} + 2 correctly.

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified(UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1 and n_{dyn} + 2 after their verification.

3) In cycle 10, the LT simulates noise (a low phase) on the active channel(s)

   (a) starting at the action point of minislot 1 and ending 5 * *gdBit* after the action point of minislot 1.

   (b) starting at the action point of minislot 1 and ending 5 * *gdBit* before the end of minislot 1.

   (c) starting at the action point of minislot 1 and ending 5 * *gdBit* after the end of minislot 1.

   (d) starting 5 * *gdBit* before the end of minislot 1 and ending 5 * *gdBit* before the end of minislot 2.

4) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, 2, 3 and 4 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values

   (a) n_{dyn}, n_{dyn} + 1, n_{dyn} + 2 and n_{dyn} + 2,

(b, c) n_{dyn}, n_{dyn}, n_{dyn} + 2 and n_{dyn} + 3 and

(d) n_{dyn}, n_{dyn}, n_{dyn} and n_{dyn} + 1.

5) In the dynamic segment of cycle 10, it is verified (LT) that the IUT

(a) transmits only the frame in the dynamic slot n_{dyn} + 2,

(b, c) does not transmit any frame and

(d) transmits its frames in the dynamic slots n_{dyn} + 1 and n_{dyn} + 2.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is

(a, b, c) true and

(d) false

in the dynamic segment status.

The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1 and n_{dyn} + 2 after their verification.

7) In cycle 11, it is verified (LT) that the IUT transmits its frames in the dynamic slots n_{dyn} + 1 and n_{dyn} + 2 correctly.

8) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 349 defines the transmit buffer status and slot status data for *dynamic segment robustness* – resynchronization attempt after single short noise.

Table 349 — Transmit buffer status and slot status data for dynamic segment robustness – resynchronization attempt after single short noise

Cycle	9 and 11			10		
Slot	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**
Frame transmitted indicator	-	true	true	-	(d) true (a, b, c) false	(a, d) true (b, c) false
Transmission conflict flag	-	false	false	-	(a, d) false (b, c) -	false
Syntax error flag	false	false	false	true	(a, d) false (b, c) -	false
Slot boundary violation flag	false	false	false	false	(a, d) false (b, c) -	false
Slot status updated indicator	true	true	true	true	(a, d) true (b, c) false	true

Table 350 defines the aggregated channel status for *dynamic segment robustness* – resynchronization attempt after single short noise.

Table 350 — Aggregated channel status for dynamic segment robustness – resynchronization attempt after single short noise

Cycle	9 and 11	10
Syntax error indicator	false	true
Slot boundary violation indicator	false	false
Transmission conflict indicator	false	false

Figure 167 depicts the resynchronization attempt after single short noise – cycle 10.

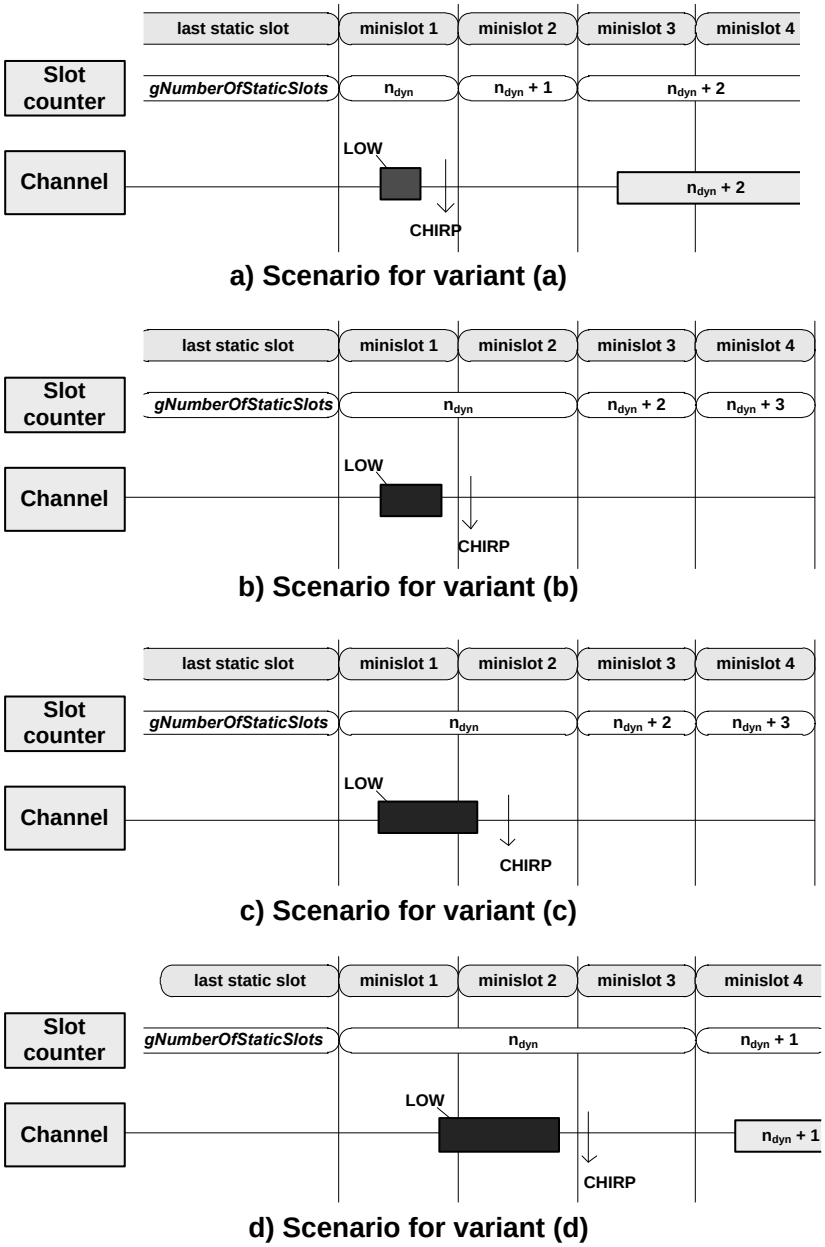


Figure 167 — Resynchronization attempt after single short noise – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the transmission conflict flag / indicator, the syntax error flag / indicator, the frame transmitted indicator, the slot status updated indicator, slot boundary violation flag / indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame transmission as expected.

7.7.9.2 Resynchronization attempt after multiple short noise

— Test purpose

Verify correct slot counting and blocking of frame transmission in the dynamic segment after the occurrence of multiple noise with a length less than *cFrameThreshold*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 351.

Table 351 — Modification to basic configurations for dynamic segment robustness – resynchronization attempt after multiple short noise

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2 and n_{dyn} + 3 with the payload 0xAA in byte 1 and 0x55 in byte 2 in continuous transmission mode on the active channel(s).

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, it is verified (LT) that the IUT transmits its frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2 and n_{dyn} + 3 correctly.

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified(UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.
The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1, n_{dyn} + 2 and n_{dyn} + 3 after their verification.

3) In minislot 1 of cycle 10, the LT simulates a low phase on the active channel(s) starting at the minislot action point and ending 5 * *gdBit* before the end of the minislot.

4) In minislot 2 of cycle 10, the LT simulates a low phase on the active channel(s) starting at the minislot action point and ending 12 * *gdBit* before the end of the minislot.

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, 2, 3 and 4 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn}, n_{dyn} + 2 and n_{dyn} + 3.

6) In the dynamic segment of cycle 10, it is verified (LT) that the IUT transmits only the frame in the dynamic slot n_{dyn} + 3.

7) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is true in the dynamic segment status.
The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1, n_{dyn} + 2 and n_{dyn} + 3 after their verification.

8) In cycle 11, it is verified (LT) that the IUT transmits its frames in the dynamic n_{dyn} + 1, slots n_{dyn} + 2 and n_{dyn} + 3 correctly.

9) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 352 defines the transmit buffer status and slot status data for *dynamic segment robustness* – resynchronization attempt after multiple short noise.

Table 352 — Transmit buffer status and slot status data for dynamic segment robustness – resynchronization attempt after multiple short noise

Cycle	**9 and 11**				**10**			
Slot	**n_{dyn}**	**n_{dyn} + 1**	**n_{dyn} + 2**	**n_{dyn} + 3**	**n_{dyn}**	**n_{dyn} + 1**	**n_{dyn} + 2**	**n_{dyn} + 3**
Frame transmitted indicator	-	true	true	true	-	false	false	true
Transmission conflict flag	-	false	false	false	-	-	false	false
Syntax error flag	false	false	false	false	true	-	false	false
Slot boundary violation flag	false	false	false	false	false	-	false	false
Slot status updated indicator	true	true	true	true	true	false	true	true

Table 353 defines the aggregated channel status for *dynamic segment robustness* – resynchronization attempt after multiple short noise.

Table 353 — Aggregated channel status for dynamic segment robustness – resynchronization attempt after multiple short noise

Cycle	**9 and 11**	**10**
Syntax error indicator	false	true
Slot boundary violation indicator	false	false
Transmission conflict indicator	false	false

Figure 168 depicts the resynchronization attempt after multiple short noise – cycle 10.

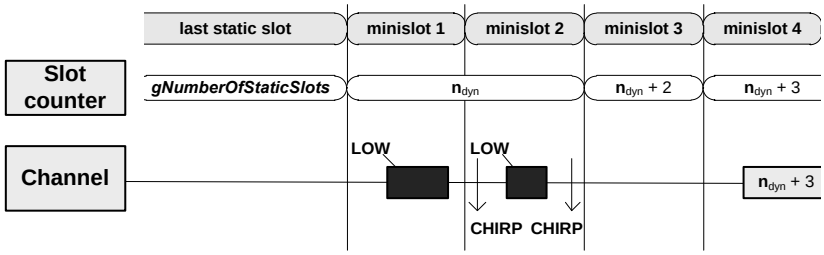


Figure 168 — Resynchronization attempt after multiple short noise – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the transmission conflict flag / indicator, the syntax error flag / indicator, the frame transmitted indicator, the slot status updated indicator, slot boundary violation flag / indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame transmission as expected.

7.7.9.3 Short noise in consecutive minislots (1)

— Test purpose

Verify correct slot counting and blocking of frame transmission in the dynamic segment after the occurrence of multiple noise with a length less than *cFrameThreshold*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 354.

Table 354 — Modification to basic configurations for dynamic segment robustness – short noise in consecutive minislots (1)

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3 and n_{dyn} + 4 with the payload 0xAA in byte 1 and 0x55 in byte 2 in continuous transmission mode on the active channel(s).

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, it is verified (LT) that the IUT transmits its frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3 and n_{dyn} + 4 correctly.

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3 and n_{dyn} + 4 after their verification.

3) In minislot 1 of cycle 10, the LT simulates a low phase on the active channel(s) starting at the minislot action point and ending 5 * *gdBit* before the end of the minislot.

4) In minislot 2 of cycle 10, the LT simulates a low phase on the active channel(s) starting 5 * *gdBit* before the end of the minislot and ending 5 * *gdBit* after the end of the minislot.

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, 2, 3, 4 and 5 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn}, n_{dyn} + 1, n_{dyn} + 3 and n_{dyn} + 4.

6) In the dynamic segment of cycle 10, it is verified (LT) that the IUT transmits only the frame in the dynamic slot n_{dyn} + 4.

7) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is true in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3 and n_{dyn} + 4 after their verification.

8) In cycle 11, it is verified (LT) that the IUT transmits its frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3 and n_{dyn} + 4 correctly.

9) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 355 defines the transmit buffer status and slot status data for *dynamic segment robustness* – short noise in consecutive minislots (1).

Table 355 — Transmit buffer status and slot status data for dynamic segment robustness – short noise in consecutive minislots (1)

Cycle	9 and 11					10				
Slot	**n_{dyn}**	**n_{dyn} + 1**	**n_{dyn} + 2**	**n_{dyn} + 3**	**n_{dyn} + 4**	**n_{dyn}**	**n_{dyn} + 1**	**n_{dyn} + 2**	**n_{dyn} + 3**	**n_{dyn} + 4**
Frame transmitted indicator	-	true	true	true	true	-	false	false	false	true
Transmission conflict flag	-	false	false	false	false	-	false	-	false	false
Syntax error flag	false	false	false	false	false	true	true	-	false	false
Boundary violation flag	false	false	false	false	false	true	true	-	false	false
Slot status updated indicator	true	true	true	true	true	true	true	false	true	true

Table 356 defines the aggregated channel status for *dynamic segment robustness* – short noise in consecutive minislots (1).

Table 356 — Aggregated channel status for dynamic segment robustness – short noise in consecutive minislots (1)

Cycle	9 and 11	10
Syntax error indicator	false	true
Boundary violation indicator	false	true
Transmission conflict indicator	false	false

Figure 169 depicts the short noise in consecutive minislots (1) – cycle 10.

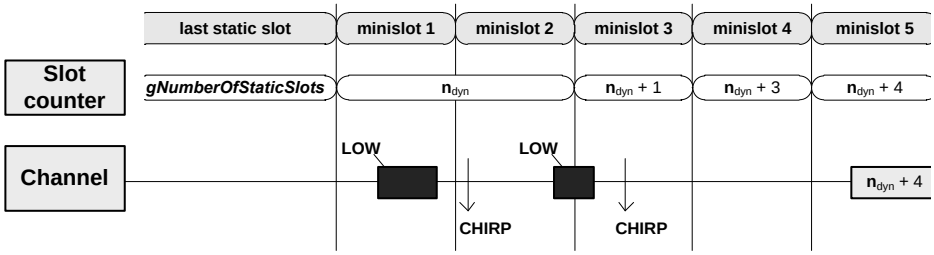


Figure 169 — Short noise in consecutive minislots (1) – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the transmission conflict flag / indicator, the syntax error flag / indicator, the frame transmitted indicator, the slot status updated indicator, slot boundary violation flag / indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame transmission as expected.

7.7.9.4 Short noise in consecutive minislots (2)

— Test purpose

Verify correct slot counting and blocking of frame transmission in the dynamic segment after the occurrence of multiple noise with a length less than *cFrameThreshold*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 357.

Table 357 — Modification to basic configurations for dynamic segment robustness – short noise in consecutive minislots (2)

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3, n_{dyn} + 4 and n_{dyn} + 5 with the payload 0xAA in byte 1 and 0x55 in byte 2 in continuous transmission mode on the active channel(s).

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, it is verified (LT) that the IUT transmits its frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3, n_{dyn} + 4 and n_{dyn} + 5 correctly.

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3, n_{dyn} + 4 and n_{dyn} + 5 after their verification.

3) In minislot 1 of cycle 10, the LT simulates a low phase on the active channel(s) starting at the minislot action point and ending 5 * *gdBit* before the end of the minislot.

4) In minislot 2 of cycle 10, the LT simulates a low phase on the active channel(s) starting at the minislot action point and ending 5 * *gdBit* before end of the minislot.

5) In minislot 3 of cycle 10, the LT simulates a low phase on the active channel(s) starting at the minislot action point and ending 5 * *gdBit* after the end of the minislot.

6) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, 2, 3, 4, 5 and 6 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn}, n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 4 and n_{dyn} + 5.

7) In the dynamic segment of cycle 10, it is verified (LT) that the IUT transmits only the frame in the dynamic slot n_{dyn} + 5.

8) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is true in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slot n_{dyn} and resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3, n_{dyn} + 4 and n_{dyn} + 5 after their verification.

9) In cycle 11, it is verified (LT) that the IUT transmits its frames in the dynamic slots n_{dyn} + 1, n_{dyn} + 2, n_{dyn} + 3, n_{dyn} + 4 and n_{dyn} + 5 correctly.

10) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 358 defines the transmit buffer status and slot status data for *dynamic segment robustness* – short noise in consecutive minislots (2).

Table 358 — Transmit buffer status and slot status data for dynamic segment robustness – short noise in consecutive minislots (2)

Cycle	**9 and 11**						**10**					
Slot	**n_{dyn}**	**n_{dyn}+ 1**	**n_{dyn}+ 2**	**n_{dyn}+ 3**	**n_{dyn}+ 4**	**n_{dyn}+ 5**	**n_{dyn}**	**n_{dyn}+ 1**	**n_{dyn}+ 2**	**n_{dyn}+ 3**	**n_{dyn}+ 4**	**n_{dyn}+ 5**
Frame transmitted indicator	-	true	true	true	true	true	-	false	false	false	false	true
Transmission conflict flag	-	false	false	false	false	false	-	false	false	-	false	false
Syntax error flag	false	false	false	false	false	false	true	false	true	-	false	false
Slot boundary violation flag	false	false	false	false	false	false	true	true	true	-	false	false
Slot status updated indicator	true	true	true	true	true	true	true	true	true	false	true	true

Table 359 defines the aggregated channel status for *dynamic segment robustness* – short noise in consecutive minislots (2).

Table 359 — Aggregated channel status for dynamic segment robustness – short noise in consecutive minislots (2)

Cycle	9 and 11	10
Syntax error indicator	false	true
Slot boundary violation indicator	false	true
Transmission conflict indicator	false	false

Figure 170 depicts the short noise in consecutive minislots (2) – cycle 10.

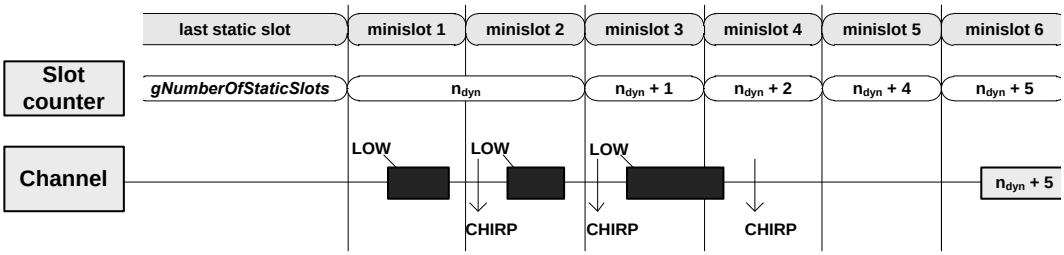


Figure 170 — Short noise in consecutive minislots (2) – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the transmission conflict flag / indicator, the syntax error flag / indicator, the frame transmitted indicator, the slot status updated indicator, slot boundary violation flag / indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame transmission as expected.

7.7.9.5 Resynchronization attempt before frame reception (1)

— Test purpose

Verify correct slot counting and frame reception in the dynamic segment after the occurrence of a noise with a length less than *cFrameThreshold*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 360.

Table 360 — Modification to basic configurations for dynamic segment robustness – resynchronization attempt before frame reception (1)

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 2.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} + 1 with the frame ID set to n_{dyn} + 1 on the active channel(s).

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, n_{dyn} + 1 and n_{dyn} + 2 after their verification.

3) In cycle 10, the LT simulates noise (a low phase) on the active channel(s) starting at the action point of minislot 1 and ending 5 * *gdBit* after the action point of minislot 1.

4) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} + 1 with the frame ID set to n_{dyn} + 1 on the active channel(s).

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, 2, n_{end} and n_{end} + 1 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn} + 1, n_{dyn} + 1 and n_{dyn} + 2.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is true in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, n_{dyn} + 1 and n_{dyn} + 2 after their verification.

7) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} + 1 with the frame ID set to n_{dyn} + 1 on the active channel(s).

8) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 361 defines the frame contents data and slot status data for *dynamic segment robustness* – resynchronization attempt before frame reception (1).

Table 361 — Frame contents data and slot status data for dynamic segment robustness – resynchronization attempt before frame reception (1)

Cycle	9 and 11			10		
Slot	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**
Frame ID	-	n_{dyn}+1	-	-	n_{dyn}+1	-
Valid frame flag	false	true	false	false	true	false
Syntax error flag	false	false	false	true	false	false
Content error flag	false	false	false	false	false	false
Slot boundary violation flag	false	false	false	false	false	false
Slot status updated indicator	true	true	true	true	true	true

Table 362 defines the aggregated channel status for *dynamic segment robustness* – resynchronization attempt before frame reception (1).

Table 362 — Aggregated channel status for dynamic segment robustness – resynchronization attempt before frame reception (1)

Cycle	9 and 11	10
Syntax error indicator	false	true
Content error indicator	false	false
Slot boundary violation indicator	false	false

Figure 171 depicts the resynchronization attempt before frame reception (1) – cycle 10.

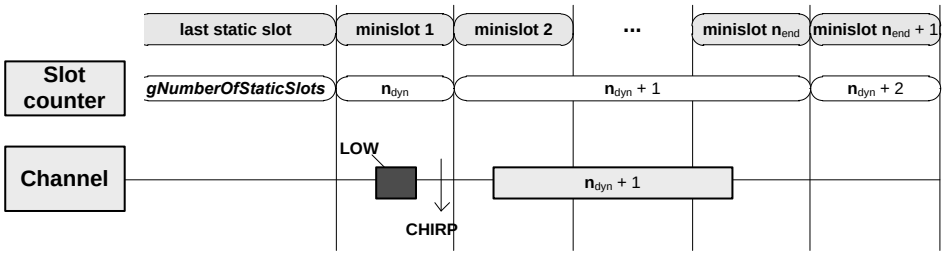


Figure 171 — Resynchronization attempt before frame reception (1) – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sets the valid frame flag, the syntax error flag / indicator, the content error flag / indicator, the slot status updated indicator, slot boundary violation flag / indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame reception as expected.

7.7.9.6 Resynchronization attempt before frame reception (2)

— **Test purpose**

Verify correct slot counting and frame reception in the dynamic segment after the occurrence of a noise with a length less than *cFrameThreshold*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 363.

Table 363 — Modification to basic configurations for dynamic segment robustness – resynchronization attempt before frame reception (2)

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 3.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} + 2 with the frame ID set to n_{dyn} + 2 on the active channel(s).

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, n_{dyn} + 1, n_{dyn} + 2 and n_{dyn} + 3 after their verification.

3) In cycle 10, the LT simulates noise (a low phase) on the active channel(s)

   (a) starting at the action point of minislot 1 and ending 5 * *gdBit* before end of minislot 1,

   (b) starting at the action point of minislot 1 and ending 5 * *gdBit* after end of minislot 1.

4) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} + 2 with the frame ID set to n_{dyn} + 2 on the active channel(s).

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, 2, n_{end} and n_{end} + 1 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn}, n_{dyn} + 2 and n_{dyn} + 3.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is true in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, n_{dyn} + 1, n_{dyn} + 2 and n_{dyn} + 3 after their verification.

7) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} + 2 with the frame ID set to n_{dyn} + 2 on the active channel(s).

8) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 364 defines the frame contents data and slot status data for *dynamic segment robustness* – resynchronization attempt before frame reception (2).

Table 364 — Frame contents data and slot status data for dynamic segment robustness – resynchronization attempt before frame reception (2)

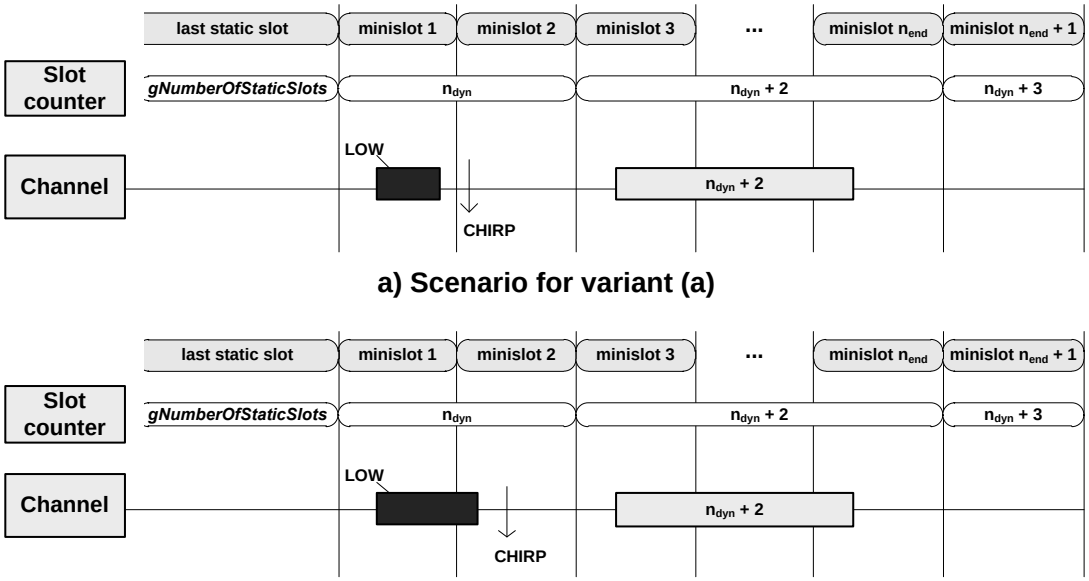
Cycle	**9 and 11**				**10**			
Slot	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**	**n_{dyn}+3**	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**	**n_{dyn}+3**
Frame ID	-	-	n_{dyn}+2	-	-	-	n_{dyn}+2	-
Valid frame flag	false	false	true	false	false	-	true	false
Syntax error flag	false	false	false	false	true	-	false	false
Content error flag	false	false	false	false	false	-	false	false
Slot boundary violation flag	false	false	false	false	false	-	false	false
Slot status updated indicator	true	true	true	true	true	false	true	true

Table 365 defines the aggregated channel status for *dynamic segment robustness* – resynchronization attempt before frame reception (2).

Table 365 — Aggregated channel status for dynamic segment robustness – resynchronization attempt before frame reception (2)

Cycle	**9 and 11**	**10**
Syntax error indicator	false	true
Content error indicator	false	false
Slot boundary violation indicator	false	false

Figure 172 depicts the resynchronization attempt before frame reception (2) – cycle 10.



a) Scenario for variant (a)

b) Scenario for variant (b)

Figure 172 — Resynchronization attempt before frame reception (2) – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the valid frame flag, the syntax error flag / indicator, the content error flag / indicator, slot boundary violation flag / indicator, the slot status updated indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame reception as expected.

7.7.9.7 Resynchronization attempt before frame reception (3)

— Test purpose

Verify correct slot counting and frame reception in the dynamic segment after the occurrence of a noise with a length less than *cFrameThreshold*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 366.

Table 366 — Modification to basic configurations for dynamic segment robustness – resynchronization attempt before frame reception (3)

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 2.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} + 1 with the frame ID set to n_{dyn} + 1 on the active channel(s).

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, n_{dyn} + 1 and n_{dyn} + 2 after their verification.

3) In cycle 10, the LT simulates noise (a low phase) on the active channel(s) starting 20 * *gdBit* before the end of minislot 1 and ending

   (a) 5 * *gdBit* before the end of minislot 1 and

   (b) 2 * *gdBit* after the end of minislot 1.

4) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} + 1 with the frame ID set to n_{dyn} + 1 on the active channel(s).

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, 2, 3, n_{end} and n_{end} + 1 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn}, n_{dyn} + 1, n_{dyn} + 1 and n_{dyn} + 2.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is true in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, n_{dyn} + 1 and n_{dyn} + 2 after their verification.

7) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} + 1 with the frame ID set to n_{dyn} + 1 on the active channel(s).

8) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 367 defines the frame contents data and slot status data for *dynamic segment robustness* – resynchronization attempt before frame reception (3).

Table 367 — Frame contents data and slot status data for dynamic segment robustness – resynchronization attempt before frame reception (3)

Cycle	**9 and 11**			**10**		
Slot	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**
Frame ID	-	n_{dyn}+1	-	-	-	-
Valid frame flag	false	true	false	false	false	false
Syntax error flag	false	false	false	true	false	false
Content error flag	false	false	false	false	false	false
Boundary violation flag	false	false	false	true	true	false
Slot status updated indicator	true	true	true	true	true	true

Table 368 defines the aggregated channel status for *dynamic segment robustness* – resynchronization attempt before frame reception (3).

Table 368 — Aggregated channel status for dynamic segment robustness – resynchronization attempt before frame reception (3)

Cycle	**9 and 11**	**10**
Syntax error indicator	false	true
Content error indicator	false	false
Boundary violation indicator	false	true

Figure 173 depicts the resynchronization attempt before frame reception (3) – cycle 10.

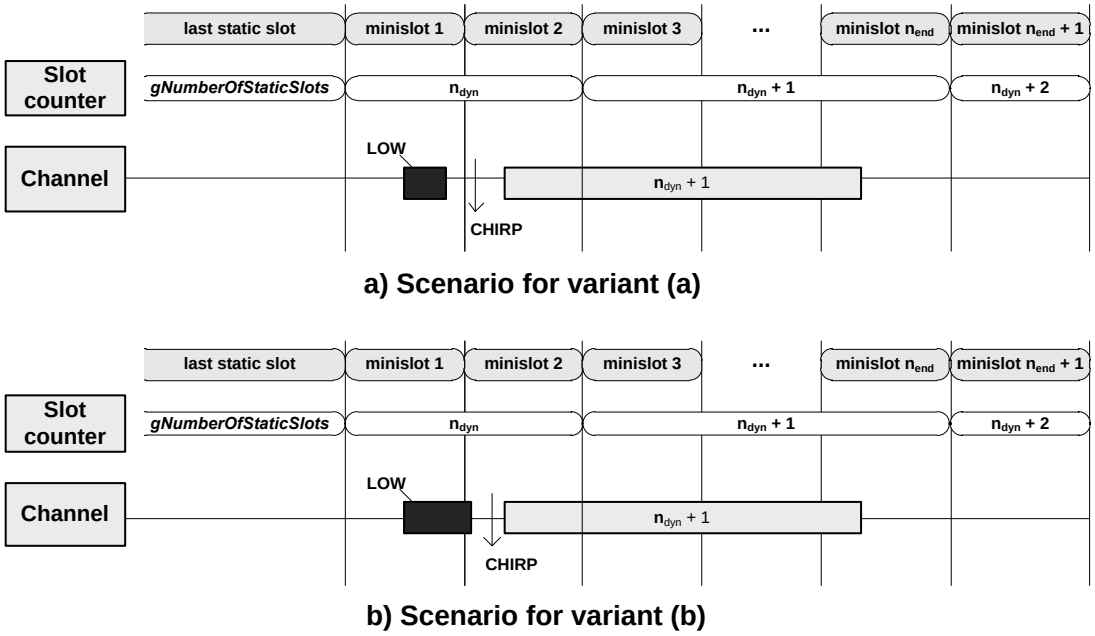


Figure 173 — Resynchronization attempt before frame reception (3) – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the valid frame flag, the syntax error flag / indicator, the content error flag / indicator, slot boundary violation flag / indicator, the slot status updated indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame reception as expected.

7.7.9.8 Valid frame reception after noise

— Test purpose

Verify correct handling of noise and a subsequent frame separated by an idle phase that is longer than *cChannelIdleDelimiter*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 369.

Table 369 — Modification to basic configurations for dynamic segment robustness – valid frame reception after noise

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 1.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn} and n_{dyn} + 1 after their verification.

3) In cycle 10, the LT simulates noise (a low phase) on the active channel(s) starting 5 * *gdBit* after the begin of minislot 1 and ending 10 * *gdBit* after the begin of minislot 1.

4) In cycle 10, the LT simulates a valid dynamic frame with the frame ID set to n_{dyn} starting 23 * *gdBit* after the begin of minislot 1 on the active channel(s).

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, n_{end} and n_{end} + 1 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn} and n_{dyn} + 1.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn} and n_{dyn} + 1 after their verification.

7) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

8) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 370 defines the frame contents data and slot status data for *dynamic segment robustness* – valid frame reception after noise.

Table 370 — Frame contents data and slot status data for dynamic segment robustness – valid frame reception after noise

Cycle	9 and 11		10	
Slot	**n_{dyn}**	**$n_{dyn}+1$**	**n_{dyn}**	**$n_{dyn}+1$**
Frame ID	n_{dyn}	-	n_{dyn}	-
Valid frame flag	true	false	true	false
Syntax error flag	false	false	true	false
Content error flag	false	false	false	false
Slot boundary violation flag	false	false	false	false
Slot status updated indicator	true	true	true	true

Table 371 defines the aggregated channel status for *dynamic segment robustness* – valid frame reception after noise.

Table 371 — Aggregated channel status for dynamic segment robustness – valid frame reception after noise

Cycle	9 and 11	10
Syntax error indicator	false	true
Content error indicator	false	false
Slot boundary violation indicator	false	false

Figure 174 depicts the valid frame reception after noise – cycle 10.

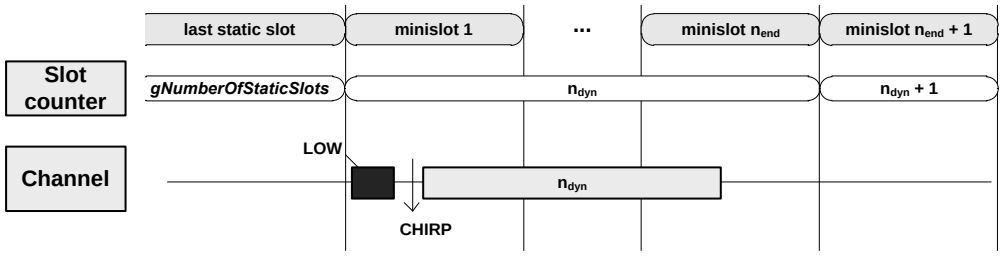


Figure 174 — Valid frame reception after noise – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sets the valid frame flag, the syntax error flag / indicator, the content error flag / indicator, slot boundary violation flag / indicator, the slot status updated indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame reception as expected.

7.7.9.9 Frame reception during noise (1)

— **Test purpose**

Verify correct handling of noise and a subsequent frame separated by an idle phase that is shorter than *cChannelIdleDelimiter*.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 372.

Table 372 — Modification to basic configurations for dynamic segment robustness – frame reception during noise (1)

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 2.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn} and n_{dyn} + 1 after their verification.

3) In cycle 10, the LT simulates noise (a low phase) on the active channel(s) starting 20 * *gdBit* before the end of minislot 1 and

   (a) ending 3 * *gdBit* before the end of minislot 1 and

   (b) ending 3 * *gdBit* after the end of minislot 1

4) In cycle 10, the LT simulates a frame on the active channel(s) starting 6 * *gdBit* after the begin of minislot 2 with the frame ID set to n_{dyn} and the DTS ending at the action point of minislot (n_{end} – *gdDynamicSlotIdlePhase*).

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, n_{end} and n_{end} + 1 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn} and n_{dyn} + 1.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn} and n_{dyn} + 1 after their verification.

7) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

8) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 373 defines the frame contents data and slot status data for *dynamic segment robustness* – frame reception during noise (1).

Table 373 — Frame contents data and slot status data for dynamic segment robustness – frame reception during noise (1)

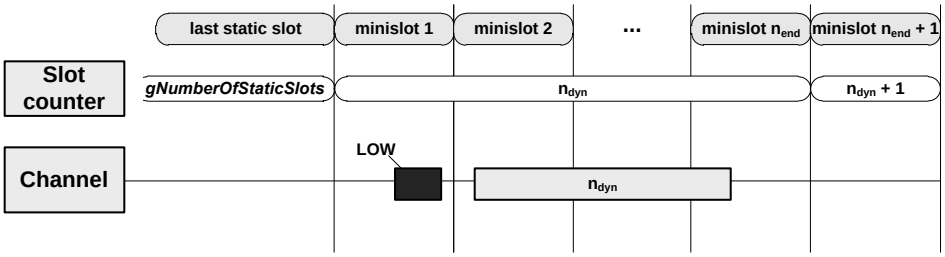
Cycle	**9 and 11**		**10**	
Slot	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}**	**n_{dyn}+1**
Frame ID	n_{dyn}	-	-	-
Valid frame flag	true	false	false	false
Syntax error flag	false	false	true	false
Content error flag	false	false	false	false
Slot boundary violation flag	false	false	false	false
Slot status updated indicator	true	true	true	true

Table 374 defines the aggregated channel status for *dynamic segment robustness* – frame reception during noise (1).

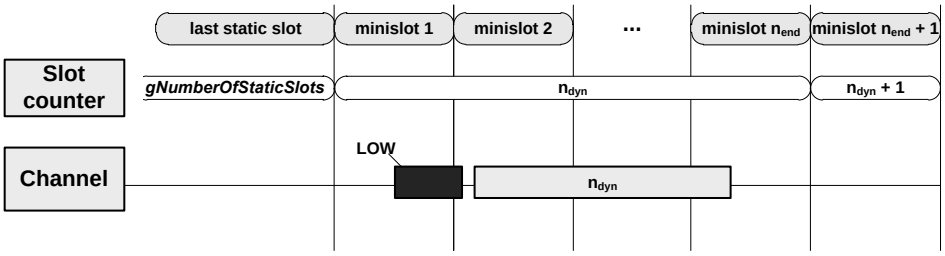
Table 374 — Aggregated channel status for dynamic segment robustness – frame reception during noise (1)

Cycle	9 and 11	10
Syntax error indicator	false	true
Content error inidicaotr	false	false
Slot boundary violation indicator	false	false
Transmission conflict indicator	false	false

Figure 175 depicts the frame reception during noise (1) – cycle 10.



a) Scenario for variant (a)



b) Scenario for variant (b)

Figure 175 — Frame reception during noise (1) – cycle 10

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The IUT sets the valid frame flag, the syntax error flag / indicator, the content error flag / indicator, slot boundary violation flag / indicator, the slot status updated indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame reception as expected.

7.7.9.10 Frame reception during noise (2)

— Test purpose

Verify correct handling of noise and a subsequent frame separated by an idle phase that is shorter than *cChannelIdleDelimiter*.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 375.

Table 375 — Modification to basic configurations for dynamic segment robustness – frame reception during noise (2)

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 1.

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

2) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status. The UT resets the slot status updated indicator in the slot status data of slots n_{dyn} and n_{dyn} + 1 after their verification.

3) In cycle 10, the LT simulates noise (a low phase) on the active channel(s) starting 5 * *gdBit* after the begin of minislot 1 and ending 5 * *gdBit* before the action point of minislot 1.

4) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

5) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, n_{end} and n_{end} + 1 and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values n_{dyn}, n_{dyn} and n_{dyn} + 1.

6) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

The UT resets the slot status updated indicator in the slot status data of slots n_{dyn} and n_{dyn} + 1 after their verification.

7) In cycle 11, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

8) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 376 defines the frame contents data and slot status data for *dynamic segment robustness* – frame reception during noise (2).

Table 376 — Frame contents data and slot status data for dynamic segment robustness – frame reception during noise (2)

Cycle	**9 and 11**		**10**	
Slot	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}**	**n_{dyn}+1**
Frame ID	n_{dyn}	-	-	-
Valid frame flag	true	false	false	false
Syntax error flag	false	false	true	false
Content error flag	false	false	false	false
Boundary violation flag	false	false	false	false
Slot status updated indicator	true	true	true	true

Table 377 defines the aggregated channel status for *dynamic segment robustness* – frame reception during noise (2).

Table 377 — Aggregated channel status for dynamic segment robustness – frame reception during noise (2)

Cycle	**9 and 11**	**10**
Syntax error indicator	false	true
Content error indicator	false	false
Boundary violation indicator	false	false

Figure 176 depicts the frame reception during noise (2) – cycle 10.

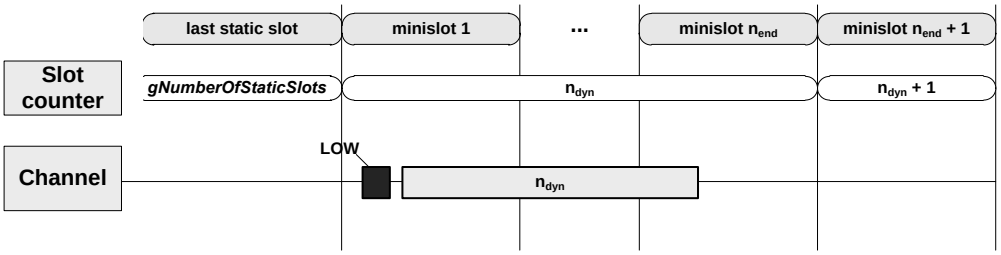


Figure 176 — Frame reception during noise (2) – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the valid frame flag, the syntax error flag / indicator, the content error flag / indicator, slot boundary violation flag / indicator, the slot status updated indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame reception as expected.

7.7.9.11 Noise after frame reception

— Test purpose

Verify correct slot counting after frame reception with following noise before channel idle recognition.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 378.

Table 378 — Modification to basic configurations for dynamic segment robustness – noise after frame reception

Parameter	Modification		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + 1.

The IUT is configured to transmit frames in the dynamic slot n_{dyn} + 2 with the payload 0xAA in byte 1 and 0x55 in byte 2 in continuous transmission mode on the active channel(s).

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

2) In the dynamic segment of cycle 9, it is verified (LT) that the IUT transmits the frame in the dynamic slot $n_{dyn} + 2$.

3) In cycle 9, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the receive buffer content, the transmit buffer status and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.
The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, $n_{dyn} + 1$ and $n_{dyn} + 2$ after their verification.

4) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} with the frame ID set to n_{dyn} on the active channel(s).

5) In cycle 10, the LT simulates noise (a low phase) on the active channel(s) starting 5 * *gdBit* after the action point of minislot n_{end} and ending at the action point of minislot $n_{end} + 1$.

6) In the dynamic segment of cycle 10, it is verified (LT) that

   (I) the IUT does not transmit any frame,

   (II) the IUT starts the transmission of the dynamic frame $n_{dyn} + 2$ in the minislot $n_{end} + 3$ and transmits the frame correctly

   (III) the IUT starts the transmission of the dynamic frame $n_{dyn} + 2$ in the minislot $n_{end} + 4$ and transmits the frame correctly

7) In the dynamic segment of cycle 10, 50 µT after the start of minislots 1, $n_{end,}$ $n_{end} + 1$, $n_{end} + 2$, $n_{end} + 3$ and $n_{end} + 4$ and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values

   (I) n_{dyn}, n_{dyn}, $n_{dyn} + 1$, $n_{dyn} + 2$, $n_{dyn} + 3$ and $n_{dyn} + 4$,

   (II) n_{dyn}, n_{dyn}, n_{dyn}, $n_{dyn} + 1$, $n_{dyn} + 2$ and $n_{dyn} + 2$

   (III) n_{dyn}, n_{dyn}, n_{dyn}, n_{dyn}, $n_{dyn} + 1$ and $n_{dyn} + 2$

8) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the receive buffer content, the transmit buffer status and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is (I) true and (II,III) false in the dynamic segment status.
The UT resets the slot status updated indicator in the slot status data of slots n_{dyn}, $n_{dyn} + 1$ and $n_{dyn} + 2$ after their verification.

9) In cycle 11, the LT simulates a frame in dynamic slot $n_{dyn} + 1$ with the frame ID set to $n_{dyn} + 1$ on the active channel(s).

10) In the dynamic segment of cycle 11, it is verified (LT) that the IUT transmits the frame in the dynamic slot $n_{dyn} + 2$.

11) In cycle 11, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the receive buffer content, the transmit buffer status and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false in the dynamic segment status.

Table 379 defines the receive buffer contents and transmit buffer status for *dynamic segment robustness* – noise after frame reception.

Table 379 — Receive buffer contents and transmit buffer status for dynamic segment robustness – noise after frame reception

Cycle	**9**			**10**			**11**		
Slot	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**	**n_{dyn}**	**n_{dyn}+1**	**n_{dyn}+2**
Frame transmitted indicator	-	-	true	-	-	(I) false (II,III) true	-	-	true
Transmission conflict flag	-	-	false	-	-	(I) - (II,III) false			false
Frame ID	n_{dyn}	-	-	n_{dyn}	-	-	-	n_{dyn}+1	-
Valid frame flag	true	false	false	true	false	false	false	true	false
Syntax error flag	false	false	false	false	false	false	false	false	false
Content error flag	false	false	false	false	false	false	false	false	false
Slot boundary violation flag	false	false	false	(I) true (II,II) false	(I) true (II,III) false	false	false	false	false
Slot status updated indicator	true	true	true	true	true	true	true	true	true

Table 380 defines the aggregated channel status for *dynamic segment robustness* – noise after frame reception.

Table 380 — Aggregated channel status for dynamic segment robustness – noise after frame reception

Cycle	**9 and 11**	**10**
Syntax error indicator	false	false
Content error indicator	false	false
Slot boundary violation indicator	false	(I) true (II,III) false
Transmission conflict indicator	false	false

Figure 177 depicts the noise after frame reception – cycle 10.

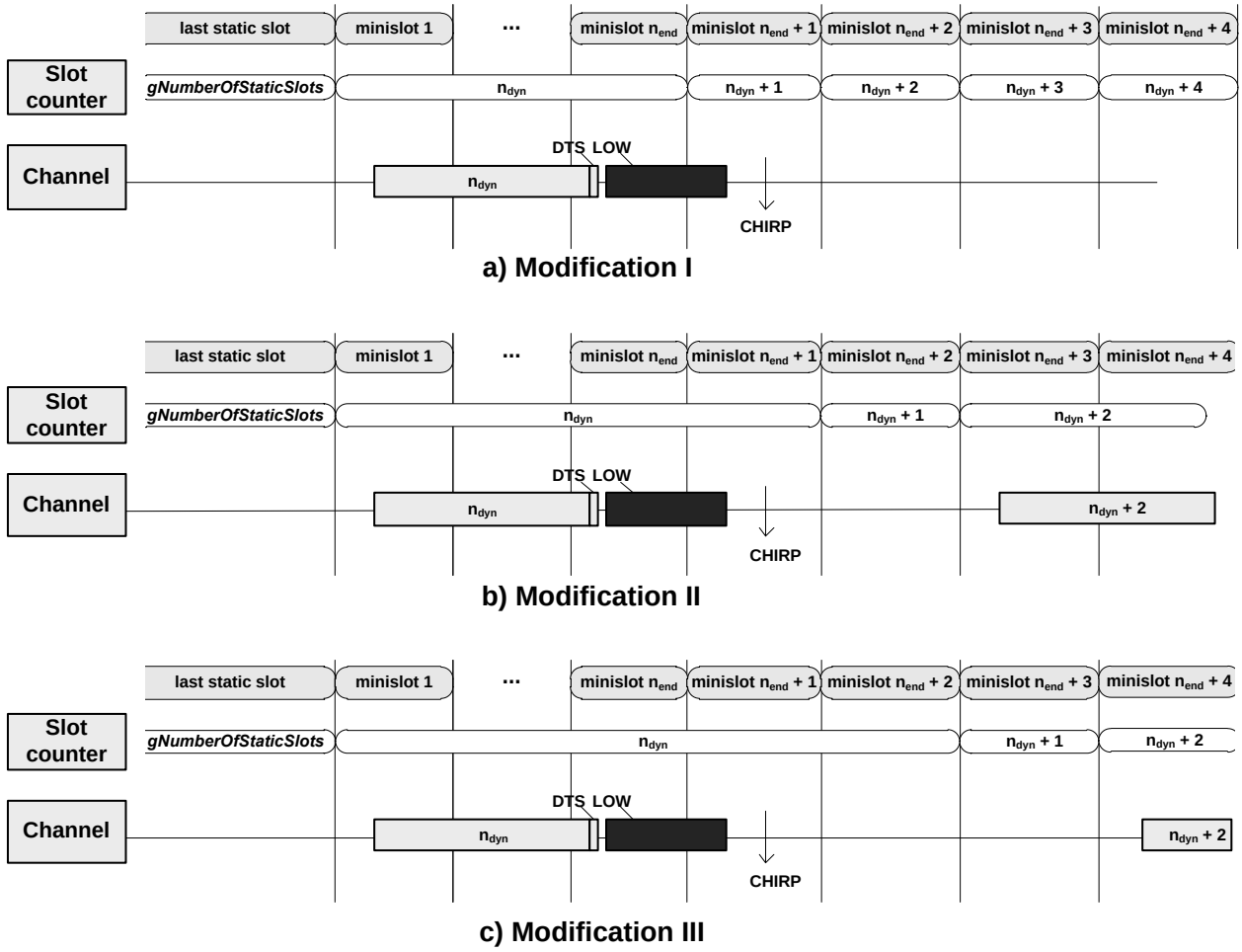


Figure 177 — Noise after frame reception – cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the transmission conflict flag / indicator, the syntax error flag / indicator, the content error flag / indicator, the frame transmitted indicator, the slot status updated indicator, slot boundary violation flag / indicator and the value of *vDynResyncAttempt[Ch]* accordingly and performs slot counting and dynamic frame transmission and reception as expected.

7.7.9.12 Frame transmission after noise crossing segment boundary

— Test purpose

Verify the frame transmission, the associated slot status and aggregated channel status indications after noise crossing the boundary between static segment and dynamic segment.

Generate noise under the following conditions:

— Starting before the segment boundary and crossing the segment boundary.

— Ending before the dynamic frame in the first dynamic slot.

— After the noise, there is enough time space for CHIRP detection before the start of the next frame.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to transmit a frame in the dynamic slot n_{dyn} with the payload 0xAA in byte 1 and 0x55 in byte 2.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, it is verified (LT) that the IUT transmit its frame in the dynamic slot n_{dyn} correctly.

2) In cycle 8, 500 uT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.
It is also verified (UT) that *vDynResyncAttempt[Ch]* is false.
The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn} after their verification.

3) In cycle 9, the LT simulates noise (a low phase)

   (a) beginning 6 * *gdBit* before the segment boundary and ending 5 * *gdBit* before the segment boundary

   (b) beginning (5 + (*gdTSSTransmitter* + 3)) * *gdBit* before the segment boundary and ending 5 * *gdBit* before the segment boundary

   (c) beginning (5 + *cdCASRxLowMin*) * *gdBit* before the segment boundary and ending 5 * *gdBit* before the segment boundary

   (d) beginning (5 + (*gdCASRxLowMax* + 1)) * *gdBit* before the segment boundary and ending 5 * *gdBit before the segment boundary* between static and dynamic segment.

4) In the dynamic segment of cycle 9, it is verified (LT) that the IUT transmits its frame in the dynamic slot n_{dyn}.

5) In cycle 9, 500 uT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below.

It is also verified (UT) that *vDynResyncAttempt[Ch]* is false.
The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn} after their verification.

6) In cycle 10, it is verified (LT) that the IUT transmit its frame in the dynamic slot n_{dyn} correctly.

7) In cycle 10, 500 uT after the end of the dynamic segment and before cycle end, it is verified (UT) that the transmit buffer status, the slot status data and the aggregated channel status holds the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* is false.

8) The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn} after their verification.

Table 381 defines the transmit buffer status and slot status data for *dynamic segment robustness* – frame transmission after noise crossing segment.

Table 381 — Transmit buffer status and slot status data for dynamic segment robustness – frame transmission after noise crossing segment boundary

Cycle	**8 and 10**	**9**
Slot	**n_{dyn}**	**n_{dyn}**
Frame transmitted indicator	true	true
Transmission conflict flag	false	false
Syntax error flag	false	false
Slot boundary violation flag	false	true
Slot status updated indicator	true	true

Table 382 defines the aggregated channel status for *dynamic segment robustness* – frame transmission after noise crossing segment boundary.

Table 382 — Aggregated channel status for dynamic segment robustness – frame transmission after noise crossing segment boundary

Cycle	**8 and 10**	**9**
Syntax error indicator	false	false
Slot boundary violation indicator	false	true
Transmission conflict indicator	false	false

Figure 178 depicts the frame transmission after noise crossing segment boundary.

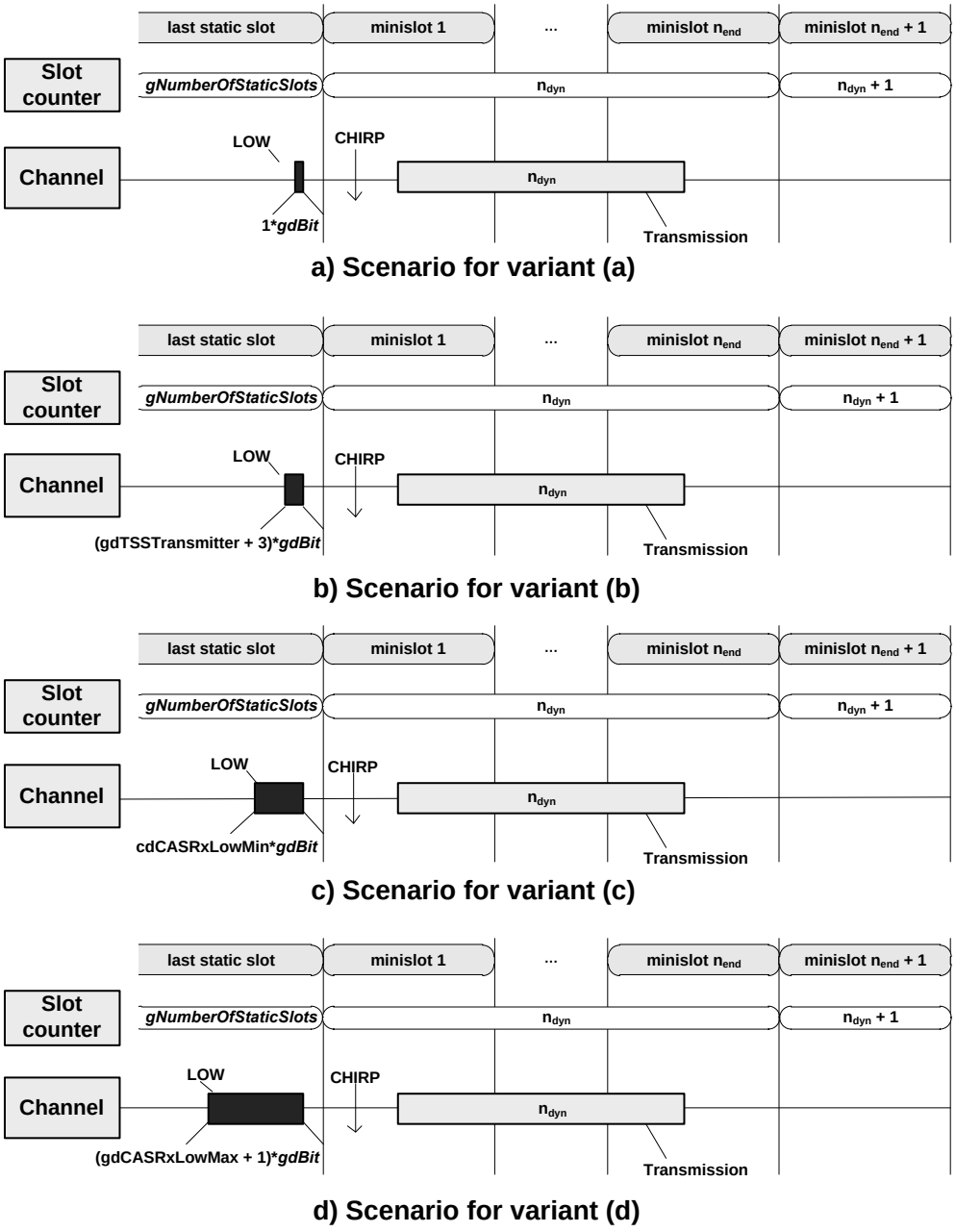


Figure 178 — Frame transmission after noise crossing segment boundary

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT sets the frame transmitted indicator, the slot boundary violation flag / indicator, the syntax error flag / indicator, the transmission conflict flag / indicator, the slot status updated indicator, and the value of *vDynResyncAttempt[Ch]* accordingly.

7.7.9.13 Frame reception after noise crossing segment boundary

— Test purpose

Verify the frame reception, the associated slot status and aggregated channel status indications after noise crossing the boundary between static segment and dynamic segment.

Generate noise under the following condition.

— Starting before the segment boundary and crossing the segment boundary.

— Ending before the dynamic frame in the first dynamic slot.

— After the noise, there is enough time space for CHIRP detection before the start of the next frame.

— Applicability

SC, DC.

— Configuration

All basic configurations.

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil**(**(*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)**)** + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to receive frame in the dynamic slot $n_{dyn.}$.

— Preamble (setup state)

Preamble I.

— Test execution

1) In cycle 8, the LT simulates the frame in dynamic slot n_{dyn} It is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below in the NIT of the cycle.

2) In cycle 9, the LT simulates noise (a low phase)

   (a) beginning 6 * *gdBit* before the segment boundary and ending 5 * *gdBit* before the segment boundary

   (b) beginning (5 + (*gdTSSTransmitter* + 3)) * *gdBit* before the segment boundary and ending 5 * *gdBit* before the segment boundary

   (c) beginning (5 + *cdCASRxLowMin*) * *gdBit* before the segment boundary and ending 5 * *gdBit* before the segment boundary

   (d) beginning(5 + (*gdCASRxLowMax* + 1)) * *gdBit* before the segment boundary and ending 5 * *gdBit* before the segment boundary

   between static and dynamic segment.

3) In cycle 9, the LT simulates an valid frame in dynamic slot n_{dyn} beginning at the action point of minislot 1.

4) It is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below in the NIT of cycle 9.

5) In cycle 10, the LT simulates the frame in dynamic slot n_{dyn}. It is verified (UT) that the slot status data and the aggregated channel status holds the values as listed in the tables below in the NIT of cycle 10.

Table 383 defines the frame contents data and slot status data for *dynamic segment robustness* – frame reception after noise crossing segment boundary.

Table 383 — Frame contents data and slot status data for dynamic segment robustness – frame reception after noise crossing segment boundary

Cycle	**8 and 10**	**9**
Slot	**n_{dyn}**	**n_{dyn}**
Valid frame flag	true	true
Slot boundary violation flag	false	true

Table 384 defines the aggregated channel status for *dynamic segment robustness* – frame reception after noise crossing segment boundary.

Table 384 — Aggregated channel status for dynamic segment robustness – frame reception after noise crossing segment boundary

Cycle	**8 and 10**	**9**
Slot boundary violation indicator	false	true

Figure 179 depicts the frame reception after noise crossing segment boundary.

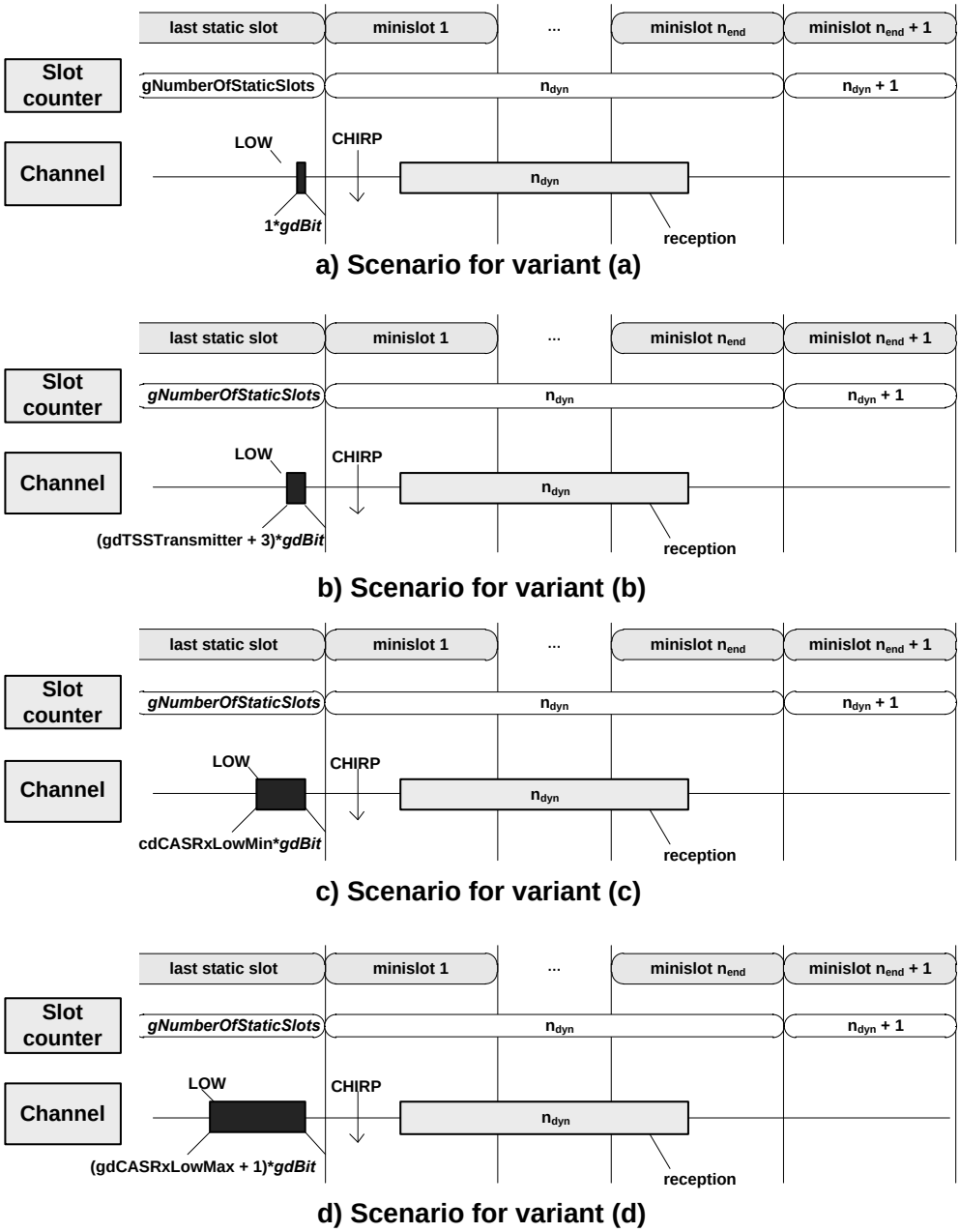


Figure 179 — Frame reception after noise crossing segment boundary

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator and the slot boundary violation flag / indicator are set accordingly in the slot status and in the aggregated channel status.

7.7.9.14 Resynchronization attempt after single short noise detected after valid frame reception (1)

— Test purpose

Verify correct slot counting in the dynamic segment after the occurrence of single noise of various length after valid frame reception without CHIRP before the noise start.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 385.

Table 385 — Modification to basic configurations for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2	0	1	2
gdMinislot [MT]	11			11			22		
gNumberOfMinislots	179			89			46		
gdNIT [MT]	16			13			14		
pLatestTx [Minislot]	150			60			20		

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} on the active channel(s). It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase on the active channel(s) starting 5 * *gdBit* after end of DTS and

   (a) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase.*

   (b) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*.

   (c) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase+1*.

   (d) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase+1*.

(e) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase+2*.

In the dynamic segment of cycle 10, 50 µT after the start of each minislot and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values defined in the tables below.

3) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the valid frame flag / indicator, the slot boundary violation flag / indicator and the syntax error flag / indicator in the slot status of slot n_{dyn} and slot n_{dyn}+1 and in the aggregated channel status hold the values as listed in the tables below. It is also verified (UT) that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 386 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 386 — Value of vSlotCounter[A] and / or vSlotCounter[B] (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(a)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(b)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2

Table 387 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 387 — Value of vSlotCounter[A] and / or vSlotCounter[B] (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(a)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(b)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2

Table 388 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 388 — Value of vSlotCounter[A] and / or vSlotCounter[B] (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(a)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(b)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3

Table 389 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 389 — Slot status data and *vDynResyncAttempt[Ch]* (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Slot n_{dyn}			**Slot n_{dyn}+1**			***vDynResyncAttempt [Ch]***
	Valid frame flag	**Syntax error flag**	**Slot boundary violation flag**	**Valid frame flag**	**Syntax error flag**	**Slot boundary violation flag**	
(a)	true	false	false	false	false	false	false
(b)	true	false	true	false	false	true	true
(c)	true	false	true	false	false	true	false
(d)	true	false	true	false	false	true	false
(e)	true	false	true	false	false	true	false

Table 390 defines the aggregated channel status (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 390 — Aggregated channel status (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Valid frame indicator	Syntax error indicator	Slot boundary violation indicator
(a)	true	false	false
(b)	true	false	true
(c)	true	false	true
(d)	true	false	true
(e)	true	false	true

Table 391 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 391 — Slot status data and *vDynResyncAttempt[Ch]* (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Slot n_{dyn}			Slot n_{dyn}+1			*vDynResyncAttempt [Ch]*
	Valid frame flag	Syntax error flag	Slot boundary violation flag	Valid frame flag	Syntax error flag	Slot boundary violation flag	
(a)	true	false	false	false	false	false	false
(b)	true	false	false	false	false	false	false
(c)	true	false	false	false	false	false	false
(d)	true	false	true	false	false	true	true
(e)	true	false	true	false	false	true	false

Table 392 defines the aggregated channel status (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 392 — Aggregated channel status (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Valid frame indicator	Syntax error indicator	Slot boundary violation indicator
(a)	true	false	false
(b)	true	false	false
(c)	true	false	false
(d)	true	false	true
(e)	true	false	true

Table 393 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 393 — Slot status data and vDynResyncAttempt[Ch] (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Slot n_{dyn}			Slot n_{dyn}+1			*vDynResyncAttempt [Ch]*
	Valid frame flag	Syntax error flag	Slot boundary violation flag	Valid frame flag	Syntax error flag	Slot boundary violation flag	
(a)	true	false	false	false	false	false	false
(b)	true	false	false	false	false	false	false
(c)	true	false	false	false	false	false	false
(d)	true	false	false	false	false	false	false
(e)	true	false	false	false	false	false	false

Table 394 defines the aggregated channel status (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (1).

Table 394 — Aggregated channel status (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (1)

	Valid frame indicator	Syntax error indicator	Slot boundary violation indicator
(a)	true	false	false
(b)	true	false	false
(c)	true	false	false
(d)	true	false	false
(e)	true	false	false

Figure 180 depicts the resynchronization (1) – *gdDynamicSlotIdlePhase* = 0, cycle 10.

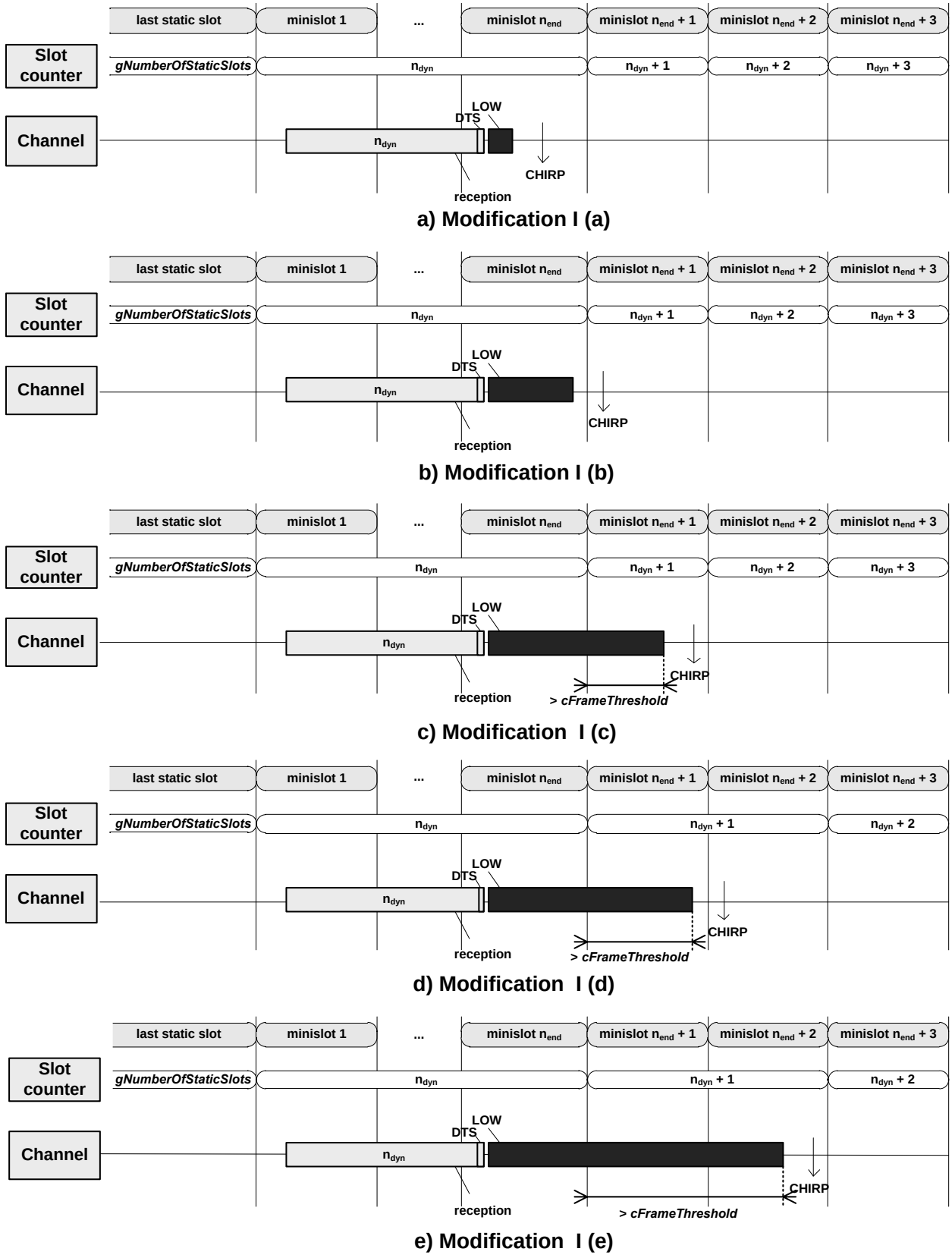


Figure 180 — Resynchronization (1) – gdDynamicSlotIdlePhase = 0, cycle 10

Figure 181 depicts the resynchronization (1) – *gdDynamicSlotIdlePhase* = 1, cycle 10.

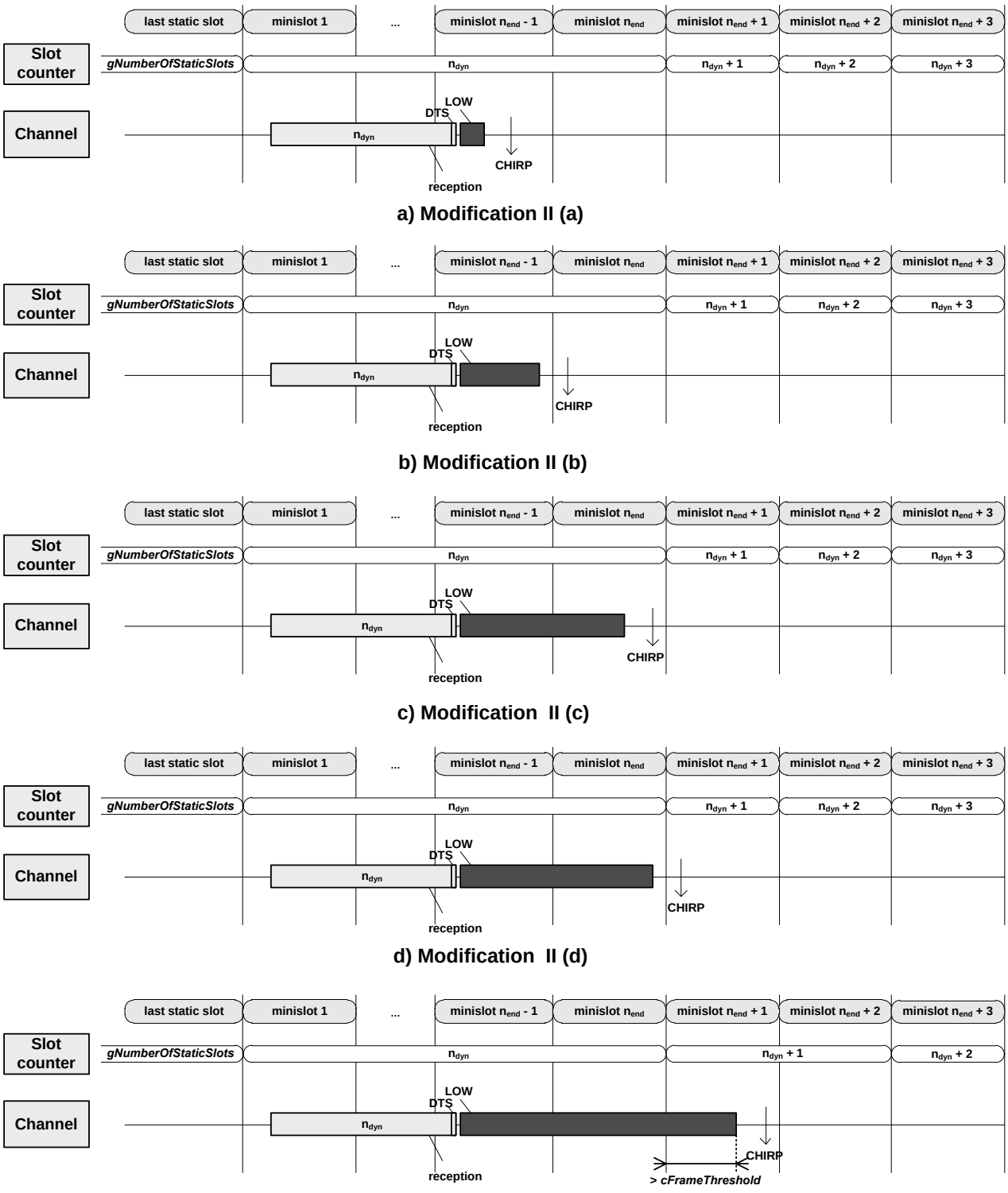


Figure 181 — Resynchronization (1) – gdDynamicSlotIdlePhase = 1, cycle 10

Figure 182 depicts the resynchronization (1) – *gdDynamicSlotIdlePhase* = 2, cycle 10.

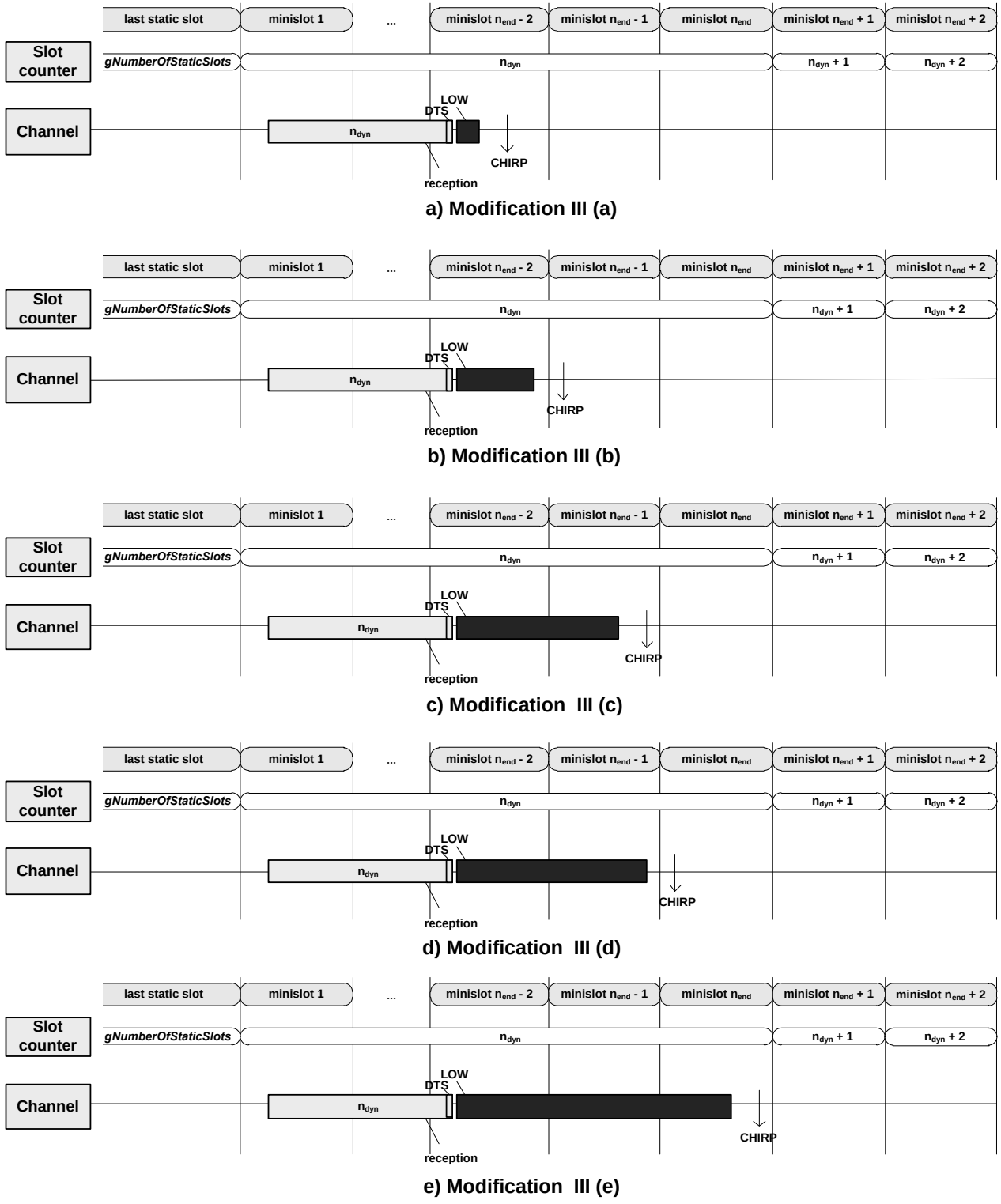


Figure 182 — Resynchronization (1) – gdDynamicSlotIdlePhase = 2, cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator, the slot boundary violation flag / indicator and the syntax error flag / indicator are set accordingly in the slot status data and in the aggregated channel status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status.

7.7.9.15 Resynchronization attempt after single short noise detected after valid frame reception (2)

— Test purpose

Verify correct slot counting in the dynamic segment after the occurrence of single noise of various length starting in the minislot after valid frame reception.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 395.

Table 395 — Modification to basic configurations for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2	0	1	2
gdMinislot [MT]	11			11			22		
gNumberOfMinislots	179			89			46		
gdNIT [MT]	16			13			14		
pLatestTx [Minislot]	150			60			20		

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} from minislot 1. It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase starting 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and

   (a) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1

   (b) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1

   (c) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2

   (d) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2

   (e) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+3

   In the dynamic segment of cycle 10, 50 µT after the start of each minislots and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values defined in the following table.

3) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the valid frame flag / indicator, the boundary violation flag / indicator and the syntax error flag / indicator in the slot status of slot n_{dyn} and slot n_{dyn}+1 and in the aggregated channel status hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 396 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 396 — Value of vSlotCounter[A] and / or vSlotCounter[B] (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(a)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(b)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+1
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+1

Table 397 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 397 — Value of vSlotCounter[A] and / or vSlotCounter[B] (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(a)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(b)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+1

Table 398 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 398 — Value of vSlotCounter[A] and / or vSlotCounter[B] (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(a)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(b)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+1

Table 399 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 399 — Slot status data and vDynResyncAttempt[Ch] (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Slot n_{dyn}			Slot $n_{dyn}+1$			*vDynResync Attempt[Ch]*
	Valid frame flag	**Syntax error flag**	**Slot boundary violation flag**	**Valid frame flag**	**Syntax error flag**	**Slot boundary violation flag**	
(a)	true	false	false	false	true	false	false
(b)	true	false	false	false	true	false	false
(c)	true	false	false	false	true	false	false
(d)	true	false	false	false	true	false	false
(e)	true	false	false	false	true	false	false

Table 400 defines the aggregated channel status (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 400 — Aggregated channel status (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Valid frame indicator	**Syntax error indicator**	**Slot boundary violation indicator**
(a)	true	true	false
(b)	true	true	false
(c)	true	true	false
(d)	true	true	false
(e)	true	true	false

Table 401 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 401 — Slot status data and vDynResyncAttempt[Ch] (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Slot n_{dyn}			Slot n_{dyn}+1			*vDynResync Attempt[Ch]*
	Valid frame flag	**Syntax error flag**	**Slot boundary violation flag**	**Valid frame flag**	**Syntax error flag**	**Slot boundary violation flag**	
(a)	true	true	false	false	false	false	false
(b)	true	true	true	false	false	true	true
(c)	true	true	true	false	false	true	false
(d)	true	true	true	false	false	true	false
(e)	true	true	true	false	false	true	false

Table 402 defines the aggregated channel status (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 402 — Aggregated channel status (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Valid frame indicator	**Syntax error indicator**	**Slot boundary violation indicator**
(a)	true	true	false
(b)	true	true	true
(c)	true	true	true
(d)	true	true	true
(e)	true	true	true

Table 403 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification III) for *dynamic segment robustness* – resynchronization attempt after single.

Table 403 — Slot status data and vDynResyncAttempt[Ch] (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Slot n_{dyn}			Slot $n_{dyn}+1$			*vDynResync Attempt[Ch]*
	Valid frame flag	**Syntax error flag**	**Slot boundary violation flag**	**Valid frame flag**	**Syntax error flag**	**Slot boundary violation flag**	
(a)	true	true	false	false	false	false	false
(b)	true	true	false	false	false	false	false
(c)	true	true	false	false	false	false	false
(d)	true	true	true	false	false	true	true
(e)	true	true	true	false	false	true	false

Table 404 defines the aggregated channel status (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (2).

Table 404 — Aggregated channel status (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (2)

	Valid frame indicator	**Syntax error indicator**	**Slot boundary violation indicator**
(a)	true	true	false
(b)	true	true	false
(c)	true	true	false
(d)	true	true	true
(e)	true	true	true

Figure 183 depicts the resynchronization (2) – *gdDynamicSlotIdlePhase* = 0, cycle 10.

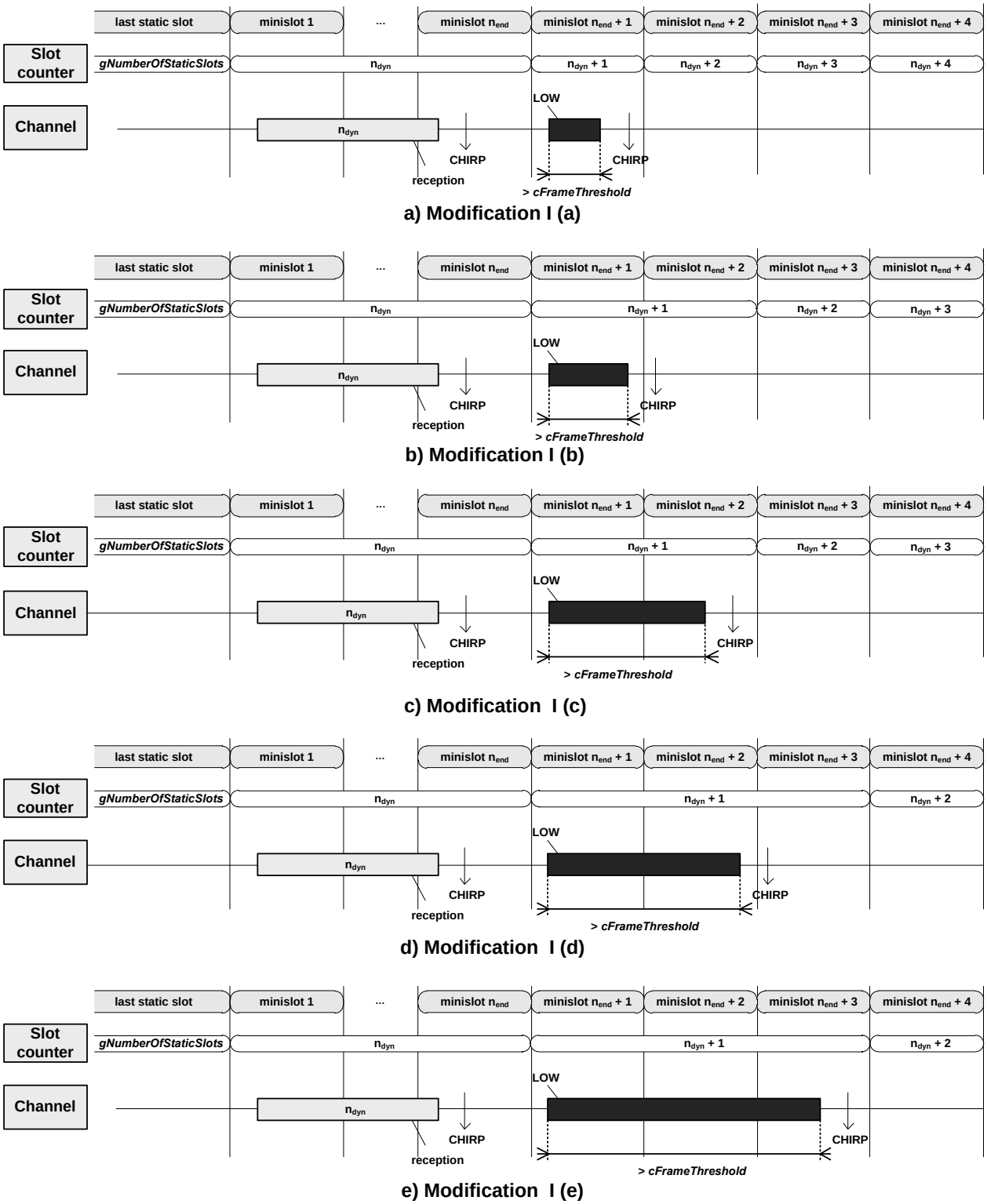


Figure 183 — Resynchronization (2) – gdDynamicSlotIdlePhase = 0, cycle 10

Figure 184 depicts the resynchronization (2) – *gdDynamicSlotIdlePhase* = 1, cycle 10.

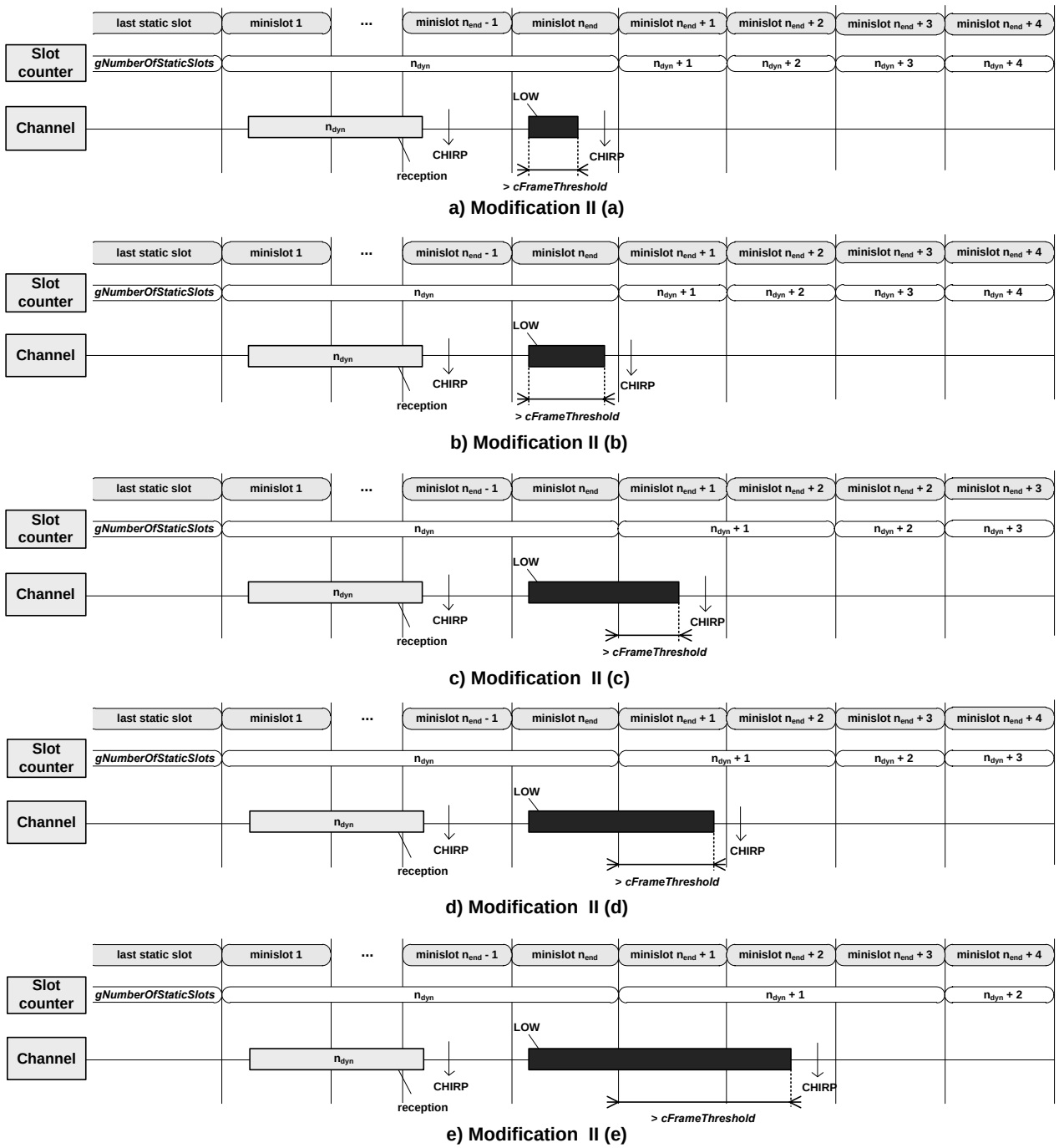


Figure 184 — Resynchronization (2) – gdDynamicSlotIdlePhase = 1, cycle 10

Figure 185 depicts the resynchronization (2) – *gdDynamicSlotIdlePhase* = 2, cycle 10.

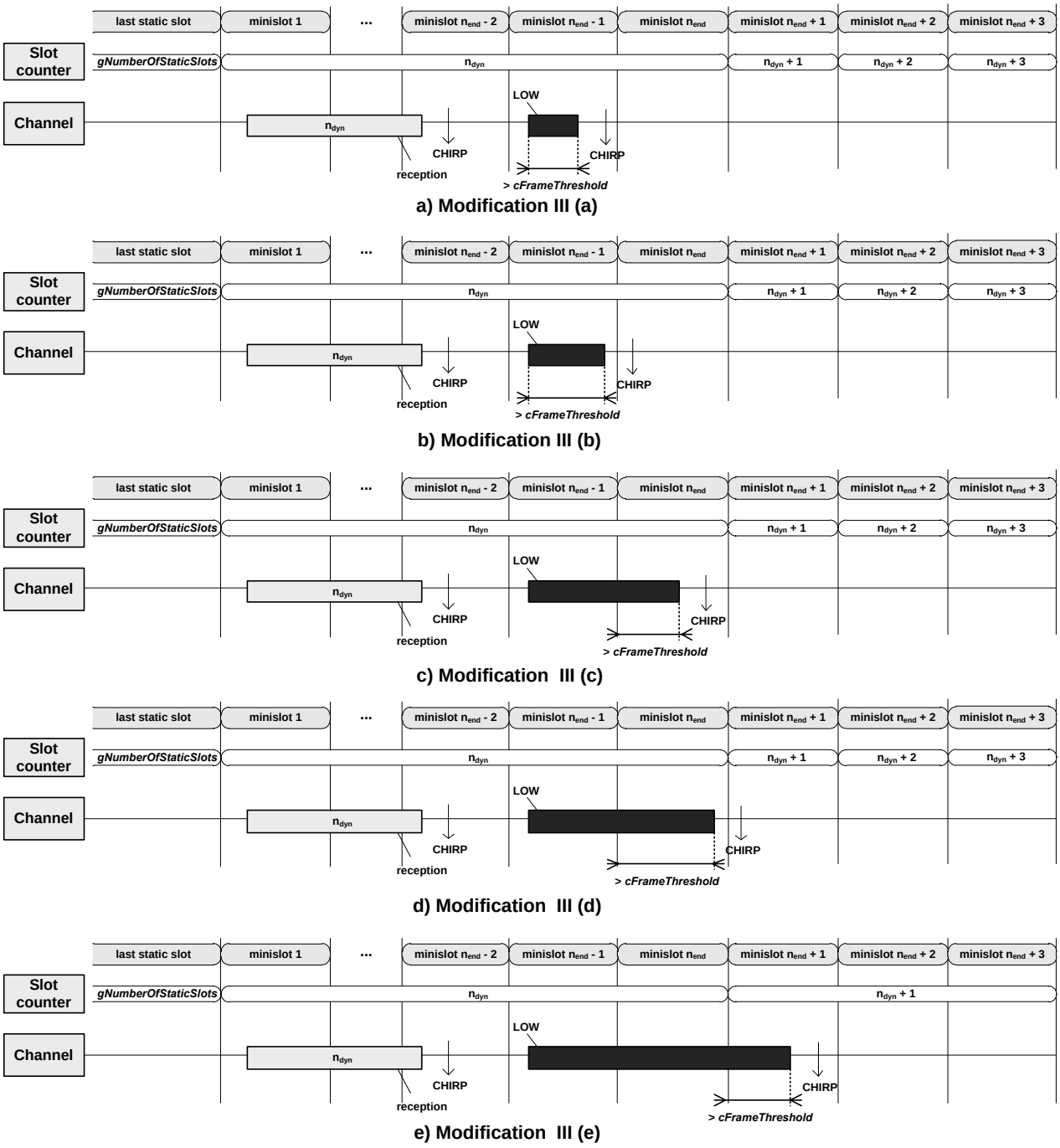


Figure 185 — Resynchronization (2) – gdDynamicSlotIdlePhase = 2, cycle 10

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The valid frame flag / indicator, the boundary violation flag / indicator and the syntax error flag / indicator are set accordingly in the slot status and in the aggregated channel status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status.

7.7.9.16 Resynchronization attempt after single short noise detected after valid frame reception (3)

— Test purpose

Verify correct slot counting in the dynamic segment after the occurrence of single noise of various length starting in the minislot after valid frame reception.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 405, Table 406 and Table 407.

Table 405 — Modification to basic configurations 1a and 1b for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gdMinislot [MT]	6			6		
gdMinislotActionPointOffset [MT]	2			2		
gNumberOfMinislots	328			328		
gdNIT [MT]	15			15		
pLatestTx [Minislot]	188			188		

Table 406 — Modification to basic configurations 2a and 2b for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gdMinislot [MT]	6			6		
gdMinislotActionPointOffset [MT]	2			2		
gNumberOfMinislots	163			163		
gdNIT [MT]	13			13		
pLatestTx [Minislot]	101			101		

Table 407 — Modification to basic configuration 3 for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2
gdMinislot [MT]	12		
gdMinislotActionPointOffset [MT]	3		
gNumberOfMinislots	84		
gdNIT [MT]	18		
pLatestTx [Minislot]	37		

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} on the active channel(s) from minislot 1. It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase on the active channel(s)

(a) starting 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 15 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1

(b) starting 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1

(c) starting 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2

(d) starting 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2

(e) starting 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+3

Modification I: (a), (b), (c), (d) and (e)

Modification II: (c), (d) and (e)

Modification III: (e)

In the dynamic segment of cycle 10, 50 µT after the start of each minislots and before the respective minislot end, it is verified (UT) that the slot counter(s) (*vSlotCounter[A]* and / or *vSlotCounter[B]*) exhibit(s) the values defined in the following table.

3) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) the valid frame flag / indicator, the boundary violation flag / indicator and the syntax error flag / indicator in the slot status of slot n_{dyn} and slot n_{dyn}+1 and in the aggregated channel status hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 408 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 408 — Value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Minislot					
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**	**n_{end}+4**
(a)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3	n_{dyn}+4
(b)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+3	n_{dyn}+4
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+3	n_{dyn}+4
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+1	n_{dyn}+2

Table 409 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 409 — Value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(c)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3
(d)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+3
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+1	n_{dyn}+3

Table 410 defines the value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 410 — Value of *vSlotCounter[A]* and / or *vSlotCounter[B]* (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Minislot				
	1	**n_{end}**	**n_{end}+1**	**n_{end}+2**	**n_{end}+3**
(e)	n_{dyn}	n_{dyn}	n_{dyn}+1	n_{dyn}+2	n_{dyn}+3

Table 411 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 411 — Slot status data and *vDynResyncAttempt[Ch]* (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Slot n_{dyn}			Slot n_{dyn}+1			***vDynResyncAttempt [Ch]***
	Valid frame flag	**Syntax error flag**	**Slot boundary violation flag**	**Valid frame flag**	**Syntax error flag**	**Slot boundary violation flag**	
(a)	true	false	false	false	true	false	true
(b)	true	false	false	false	true	false	true
(c)	true	false	false	false	true	false	true
(d)	true	false	false	false	true	false	false
(e)	true	false	false	false	true	false	false

Table 412 defines the aggregated channel status (Modification I) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 412 — Aggregated channel status (Modification I) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Valid frame indicator	Syntax error indicator	Slot boundary violation indicator
(a)	true	true	false
(b)	true	true	false
(c)	true	true	false
(d)	true	true	false
(e)	true	true	false

Table 413 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification II) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 413 — Slot status data and vDynResyncAttempt[Ch] (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Slot n_{dyn}			Slot $n_{dyn}+1$			*vDynResync Attempt[Ch]*
	Valid frame flag / indicator	Syntax error flag / indicator	Slot boundary violation flag / indicator	Valid frame flag / indicator	Syntax error flag / indicator	Slot boundary violation flag / indicator	
(c)	true	true	true	false	false	true	true
(d)	true	true	true	false	false	true	true
(e)	true	true	true	false	false	true	true

Table 414 defines the aggregated channel status (Modification II) for *dynamic segment robustness* – resynchronization attempt.

Table 414 — Aggregated channel status (Modification II) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Valid frame indicator	Syntax error indicator	Slot boundary violation indicator
(c)	true	true	true
(d)	true	true	true
(e)	true	true	true

Table 415 defines the slot status data and *vDynResyncAttempt[Ch]* (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 415 — Slot status data and vDynResyncAttempt[Ch] (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Slot n_{dyn}			Slot n_{dyn}+1			
	Valid frame flag / indicator	Syntax error flag / indicator	Slot boundary violation flag / indicator	Valid frame flag / indicator	Syntax error flag / indicator	Slot boundary violation flag / indicator	*vDynResync Attempt[Ch]*
(e)	true	true	true	false	false	true	true

Table 416 defines the aggregated channel status (Modification III) for *dynamic segment robustness* – resynchronization attempt after single short noise detected after valid frame reception (3).

Table 416 — Aggregated channel status (Modification III) for dynamic segment robustness – resynchronization attempt after single short noise detected after valid frame reception (3)

	Valid frame indicator	Syntax error indicator	Slot boundary violation indicator
(e)	true	true	true

Figure 186 depicts the resynchronization (3) – *gdDynamicSlotIdlePhase* = 0, cycle 10.

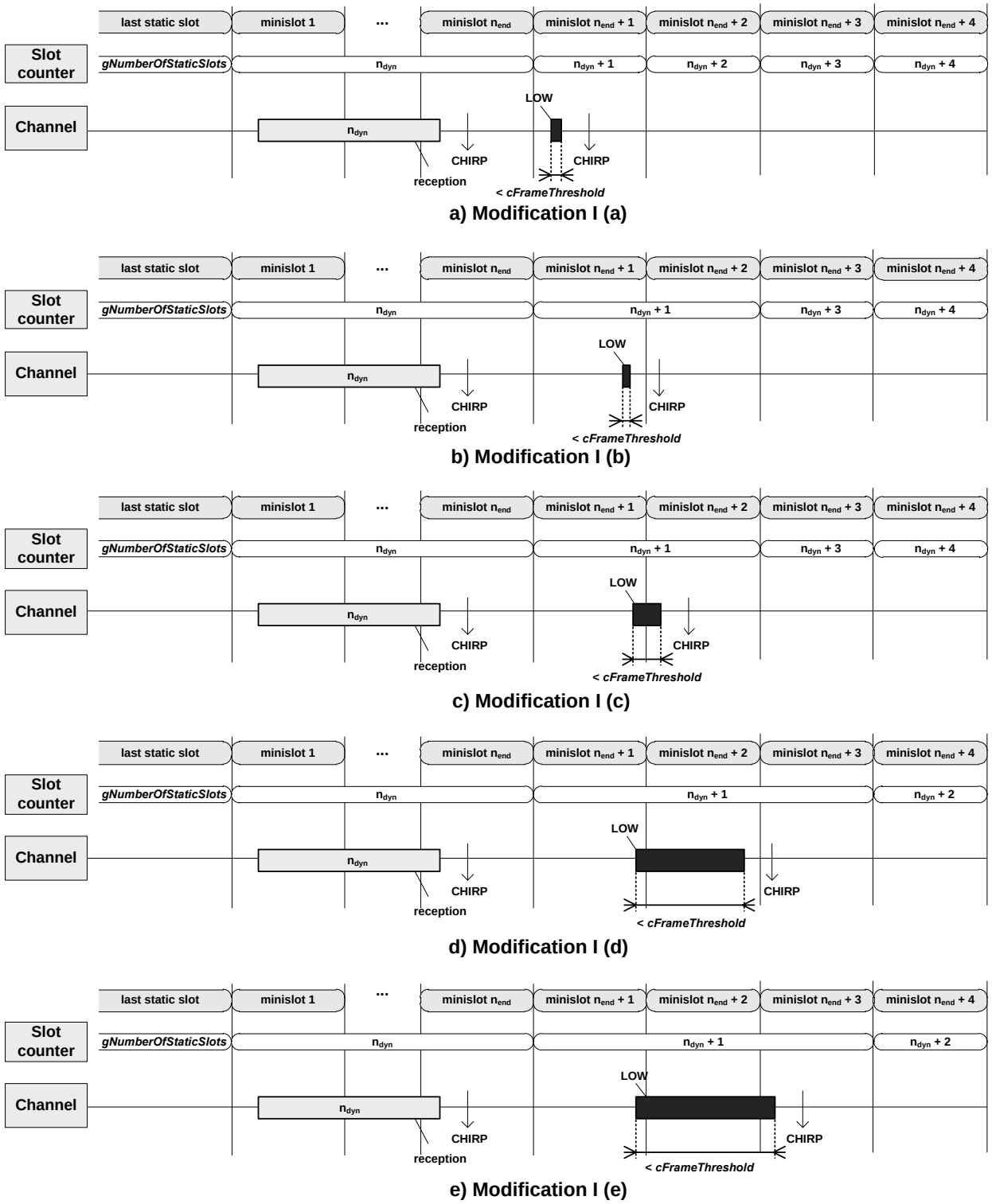


Figure 186 — Resynchronization (3) – gdDynamicSlotIdlePhase = 0, cycle 10

Figure 187 depicts the resynchronization (3) – *gdDynamicSlotIdlePhase* = 1, cycle 10.

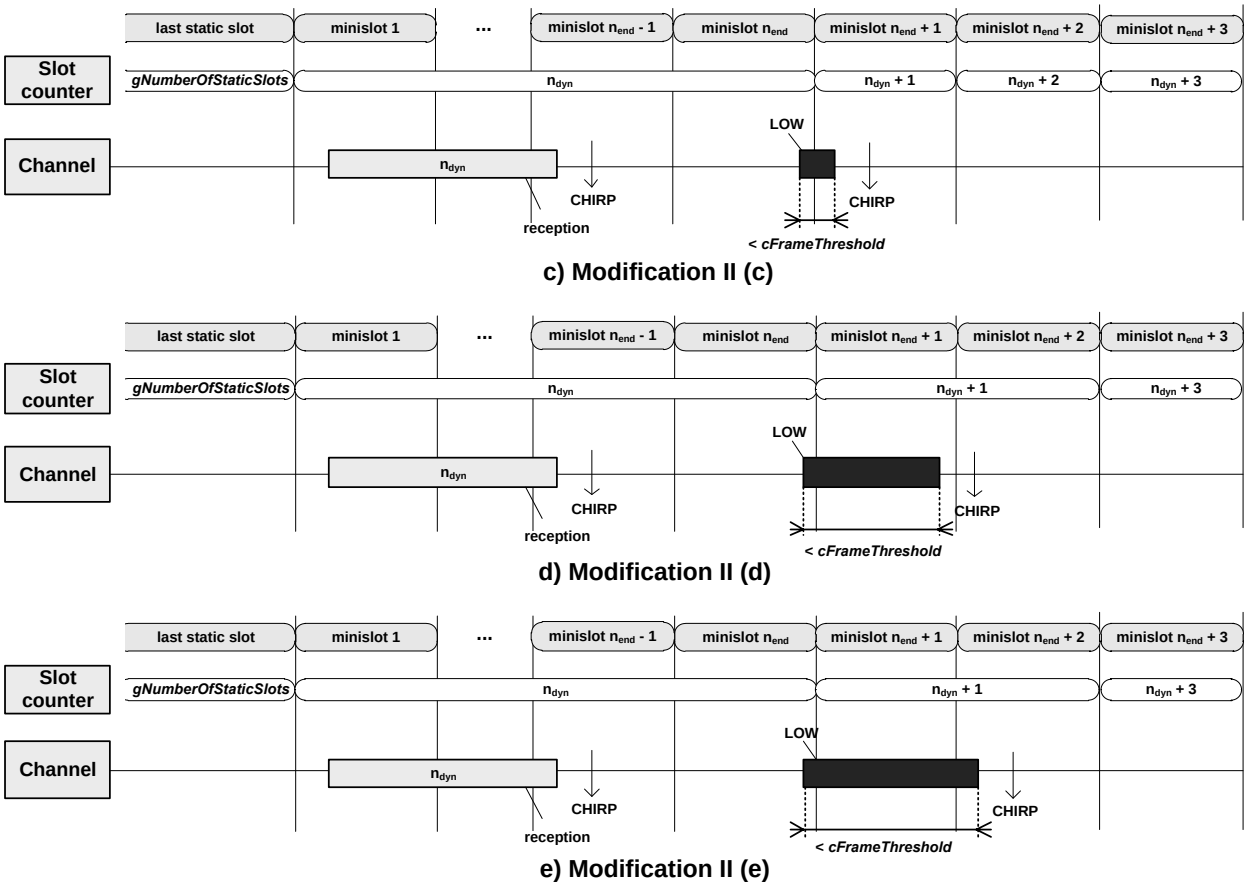


Figure 187 — Resynchronization (3) – gdDynamicSlotIdlePhase = 1, cycle 10

Figure 188 depicts the resynchronization (3) – *gdDynamicSlotIdlePhase* = 2, cycle 10.

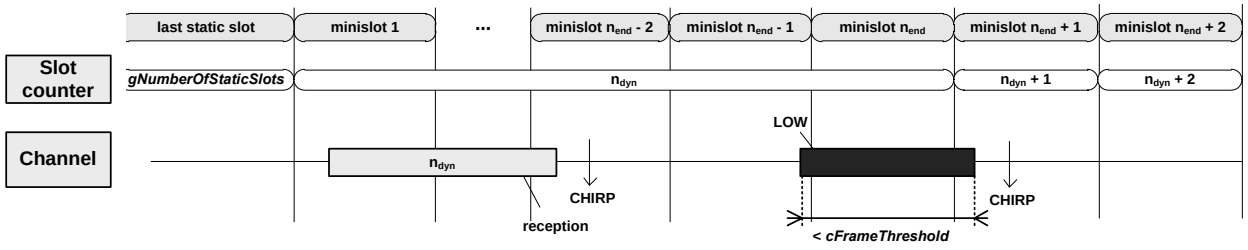


Figure 188 — Resynchronization (3) – gdDynamicSlotIdlePhase = 2, cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The valid frame flag / indicator, the boundary violation flag / indicator and the syntax error flag / indicator are set accordingly in the slot status and in the aggregated channel status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status.

7.7.9.17 Frame transmission after single short noise detected after valid frame reception (1)

— **Test purpose**

Verify correct frame transmission in the dynamic segment after the occurrence of single noise of various length after valid frame reception without CHIRP.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 417.

Table 417 — Modification to basic configurations for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (1)

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2	0	1	2
gdMinislot [MT]	11			11			22		
gNumberOfMinislots	179			89			46		
gdNIT [MT]	16			13			14		
pLatestTx [Minislot]	150			60			20		

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to receive frames in the dynamic slot n_{dyn} and transmit a frame in the dynamic slot n_{dyn}+1 in continuous mode on the active channel(s).

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn}. It is verified (UT) that the valid frame flag / indicator is true in the slot status data of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmits its frame in the dynamic slot n_{dyn}+1 correctly.

The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit-buffer status of slot n_{dyn}+1 after their verification.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase starting 5 * *gdBit* after end of DTS and

   (a) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase.*

   (b) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase.*

   (c) ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (d) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (e) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (f) ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (g) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   Modification I: (a), (b), (c) and (d)

   Modification II: (a), (b), (d), (e), (f) and (g)

   Modification III: (a), (b), (d), (e) and (g)

3) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator in the transmit buffer status and the slot status updated indicator in the slot status of slot n_{dyn}+1 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 418 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (1).

Table 418 — Transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (1)

	Cycle 10		
	Frame transmitted indicator	**Slot status updated indicator**	**vDynResyncAttempt[Ch]**
(a)	true	true	false
(b)	true	true	false
(c)	true	true	false
(d)	true	true	false

Table 419 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (1).

Table 419 — Transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (1)

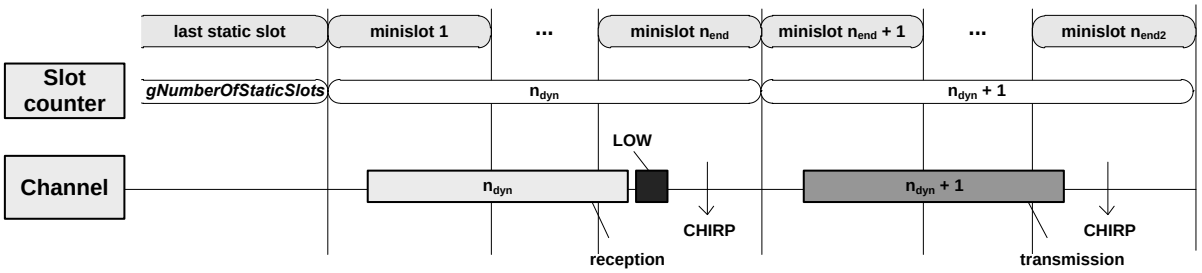
	Cycle 10		
	Frame transmitted indicator	**Slot status updated indicator**	**vDynResyncAttempt[Ch]**
(a)	true	true	false
(b)	true	true	false
(d)	true	true	false
(e)	true	true	false
(f)	true	true	false
(g)	true	true	false

Table 420 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification III) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (1).

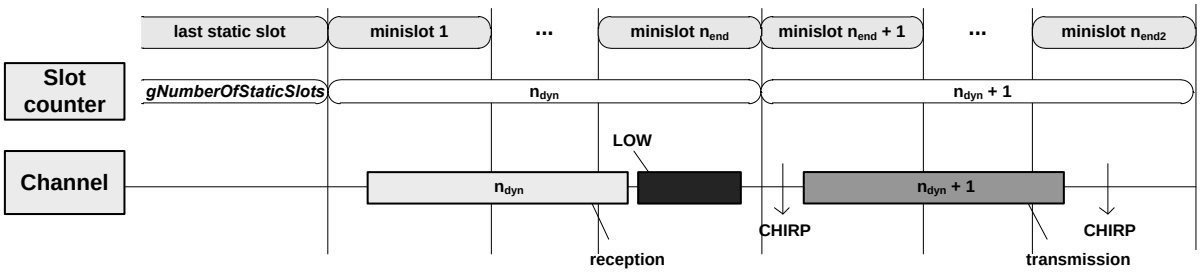
Table 420 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification III) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (1)

	Cycle 10		
	Frame transmitted indicator	**Slot status updated indicator**	**vDynResyncAttempt[Ch]**
(a)	true	true	false
(b)	true	true	false
(d)	true	true	false
(e)	true	true	false
(g)	true	true	false

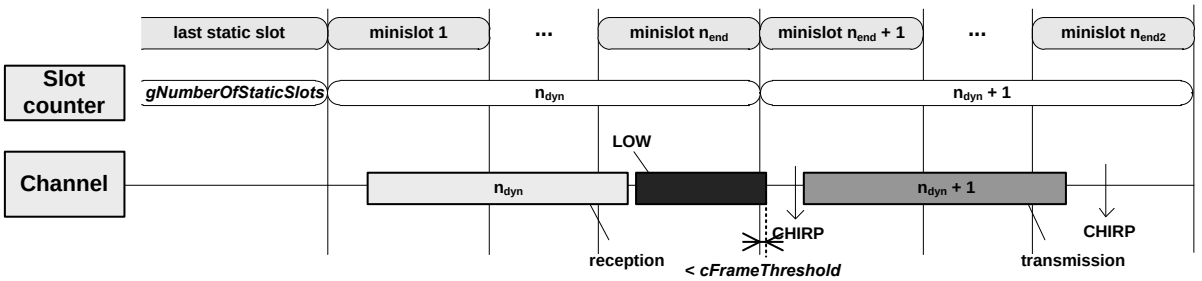
Figure 189 depicts the frame transmission (1) – *gdDynamicSlotIdlePhase* = 0, cycle 10.



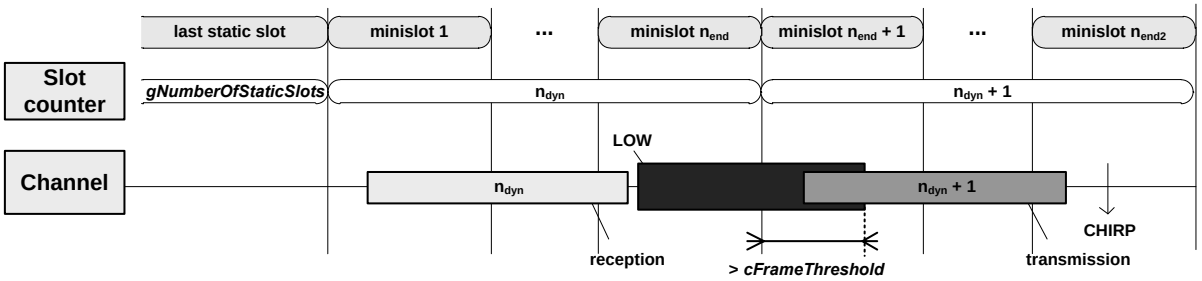
a) Modification I (a)



b) Modification I (b)



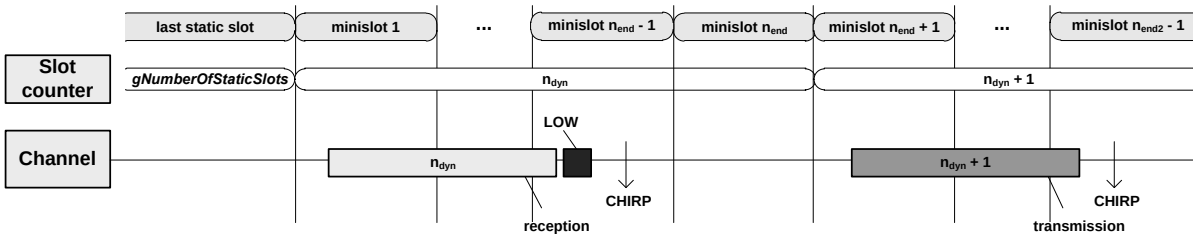
c) Modification I (c)



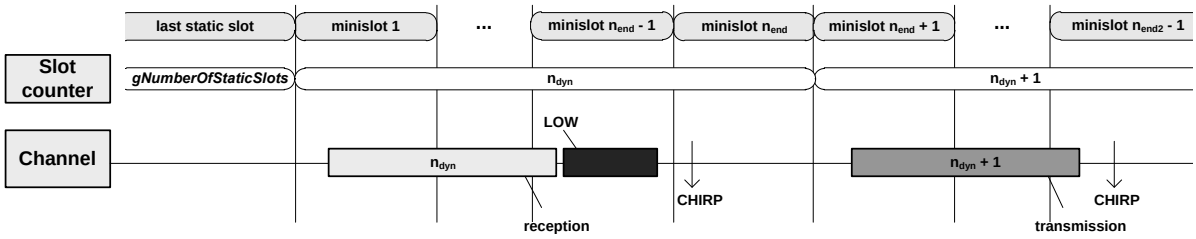
d) Modification I (d)

Figure 189 — Frame transmission (1) – gdDynamicSlotIdlePhase = 0, cycle 10

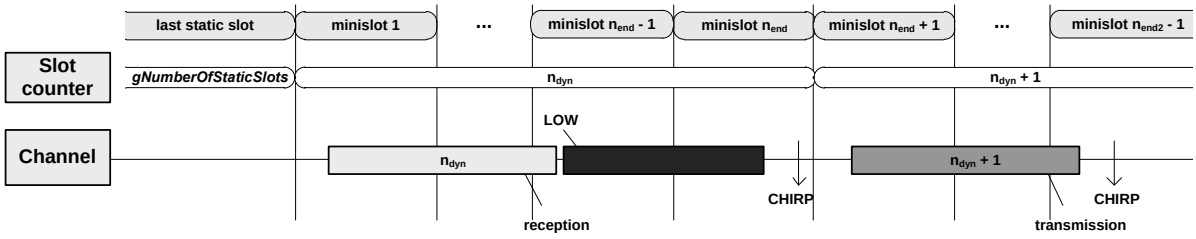
Figure 190 depicts the frame transmission (1) – *gdDynamicSlotIdlePhase* = 1, cycle 10.



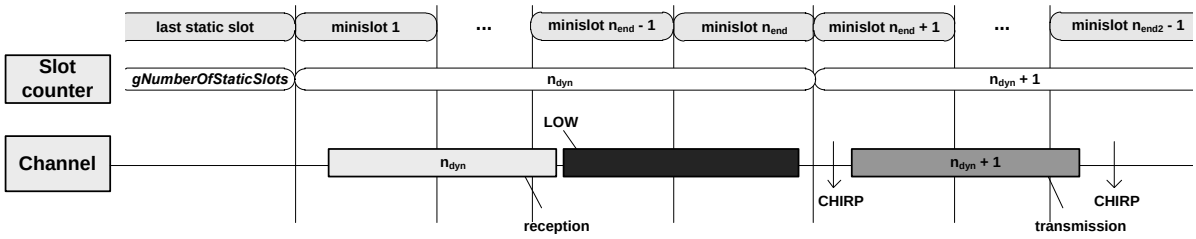
a) Modification II (a)



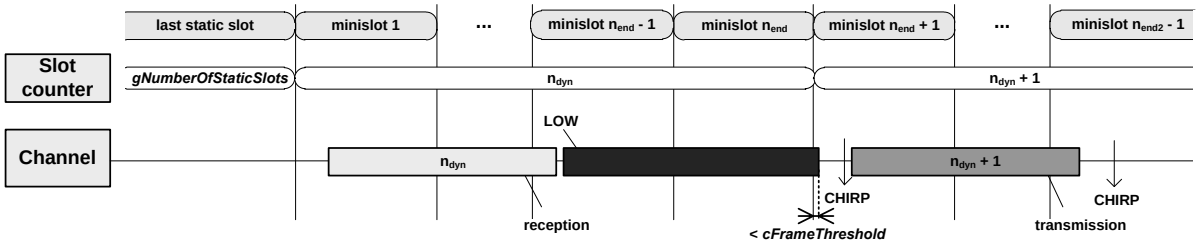
b) Modification II (b)



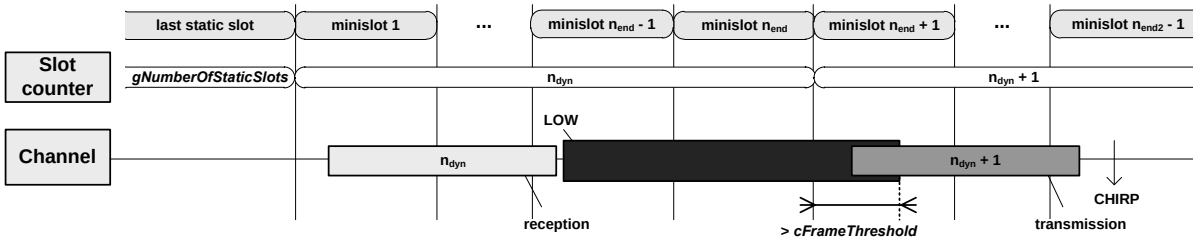
c) Modification II (d)



d) Modification II (e)



e) Modification II (f)



e) Modification II (g)

Figure 190 — Frame transmission (1) – gdDynamicSlotIdlePhase = 1, cycle 10

Figure 191 depicts the frame transmission (1) – *gdDynamicSlotIdlePhase* = 2, cycle 10.

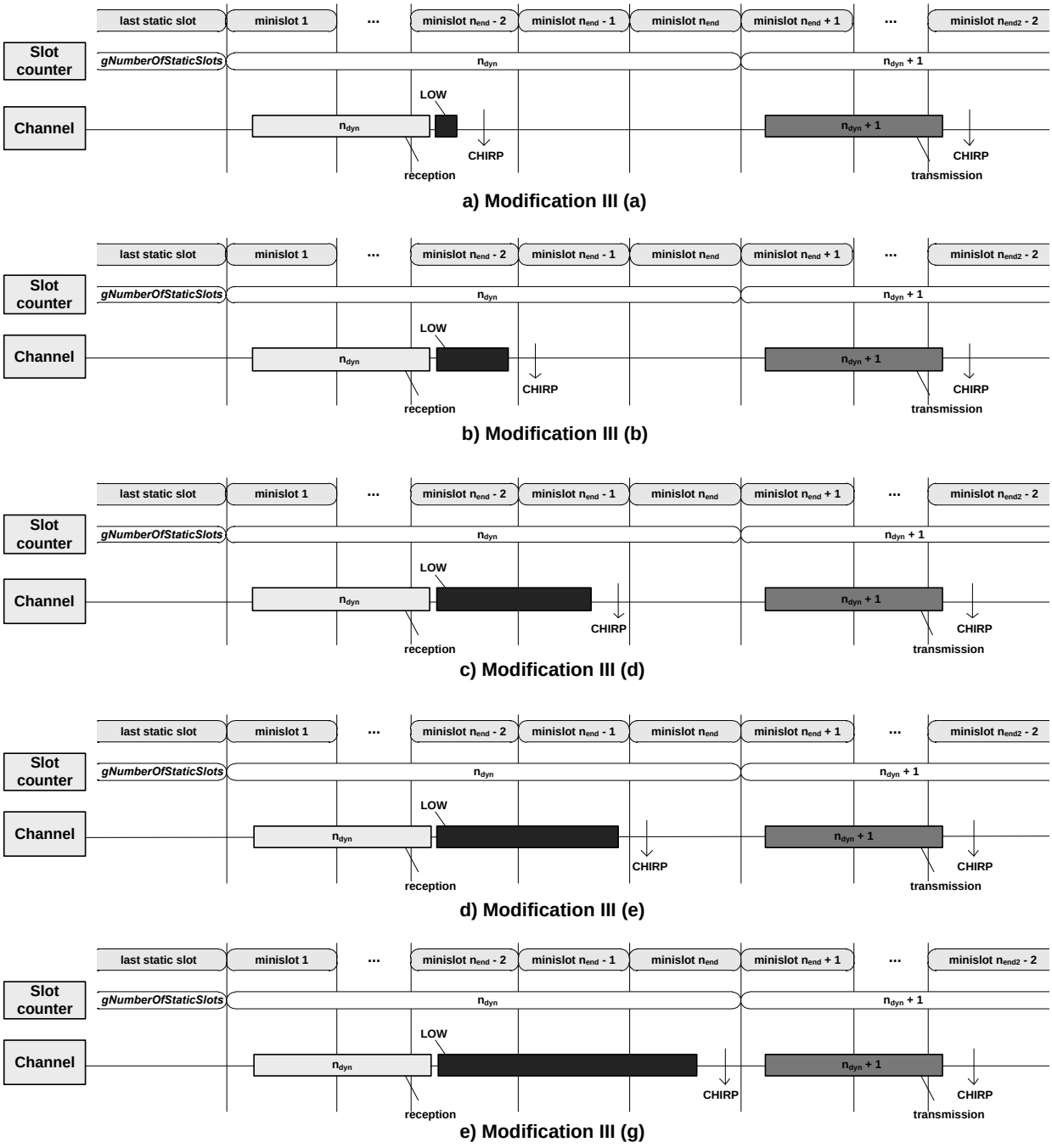


Figure 191 — Frame transmission (1) – gdDynamicSlotIdlePhase = 2, cycle 10

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The frame transmitted indicator and the slot updated indicator are set accordingly in the transmit buffer-status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status.

7.7.9.18 Frame transmission after single short noise detected after valid frame reception (2)

— **Test purpose**

Verify correct blocking of frame transmission in the dynamic segment after the occurrence of single noise of various length after valid frame reception without CHIRP.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 421.

Table 421 — Modification to basic configurations for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (2)

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2	0	1	2
gdMinislot [MT]	11			11			22		
gNumberOfMinislots	179			89			46		
gdNIT [MT]	16			13			14		
pLatestTx [Minislot]	150			60			20		

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to receive frames in the dynamic slot n_{dyn} on the active channel(s) and to transmit a frame in the dynamic slot n_{dyn}+2 and slot n_{dyn}+3 in continuous mode on the active channel(s).

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} on the active channel(s). It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmit its frames in the dynamic slots n_{dyn}+2 and n_{dyn}+3 correctly.

The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot $n_{dyn,}$ n_{dyn}+2 and n_{dyn}+3 after their verification.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase on the active channel(s) starting 5 * *gdBit* after end of DTS and

   (a) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*

   (b) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*

   (c) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (d) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (e) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2

3) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

   The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot $n_{dyn,}$ n_{dyn}+2 and n_{dyn}+3 after their verification.

4) In cycle 11, the LT does not simulate its frame in dynamic slot n_{dyn}. 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 422 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (2).

Table 422 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification I, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (2)

	Slot n_{dyn}+2		Slot n_{dyn}+3		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	true	true	true	true	false
(b)	false	true	true	true	true
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 423 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (2).

Table 423 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification I, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (2)

	Slot n_{dyn}+2		Slot n_{dyn}+3		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 424 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (2).

Table 424 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification II, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (2)

	Slot n_{dyn}+2		Slot n_{dyn}+3		vDynResyncAttempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	false	true	true	true	true
(e)	true	true	true	true	false

Table 425 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (2).

Table 425 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification II, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (2)

	Slot n_{dyn}+2		Slot n_{dyn}+3		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 426 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification III, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (2).

Table 426 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification III, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (2)

	Slot n_{dyn}+2		Slot n_{dyn}+3		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 427 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification III, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (2).

Table 427 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification III, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (2)

	Slot n_{dyn}+2		Slot n_{dyn}+3		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Figure 192 depicts the frame transmission (2) – *gdDynamicSlotIdlePhase* = 0, cycle 10.

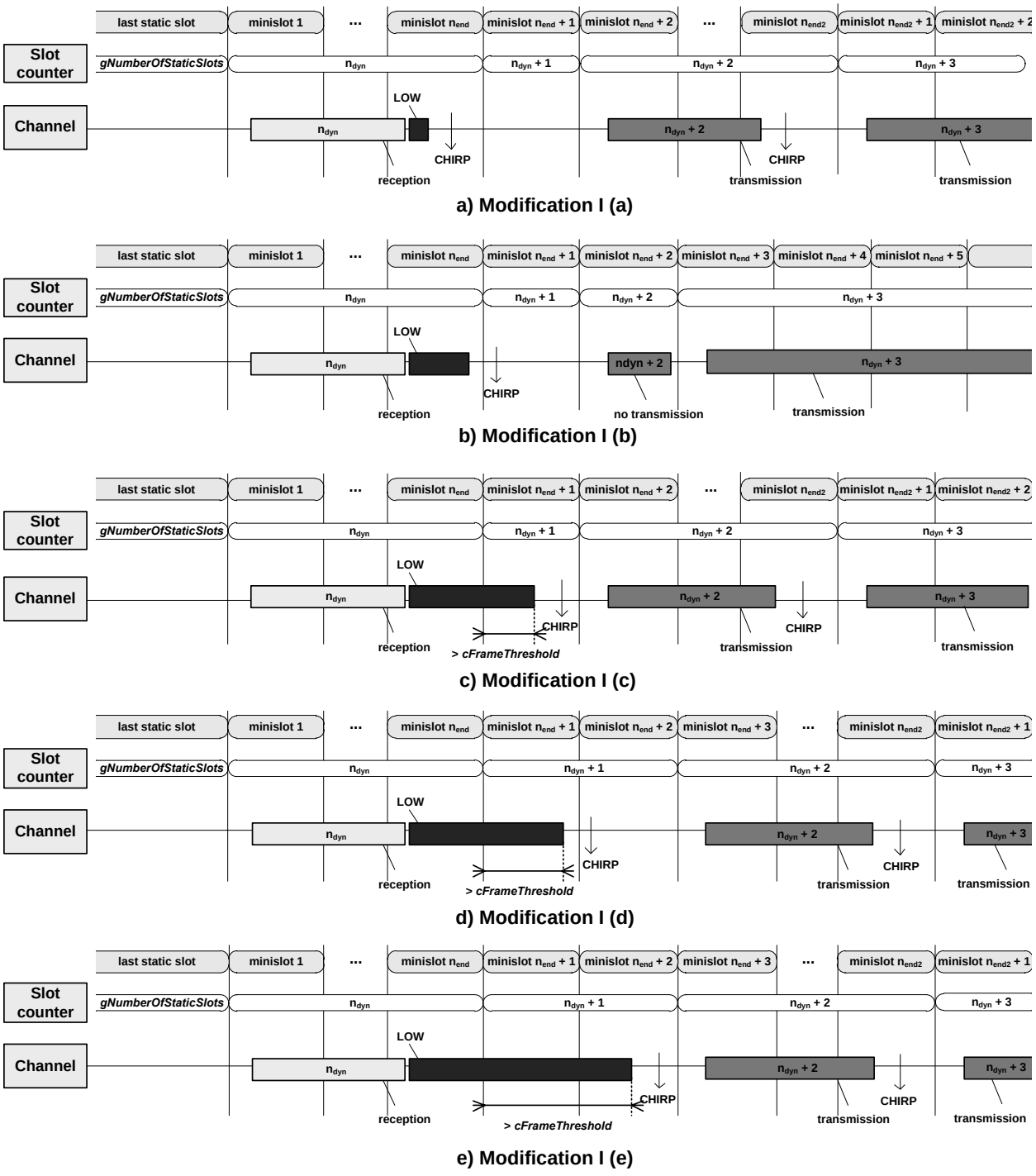


Figure 192 — Frame transmission (2) – gdDynamicSlotIdlePhase = 0, cycle 10

Figure 193 depicts the frame transmission (2) – *gdDynamicSlotIdlePhase* = 1, cycle 10.

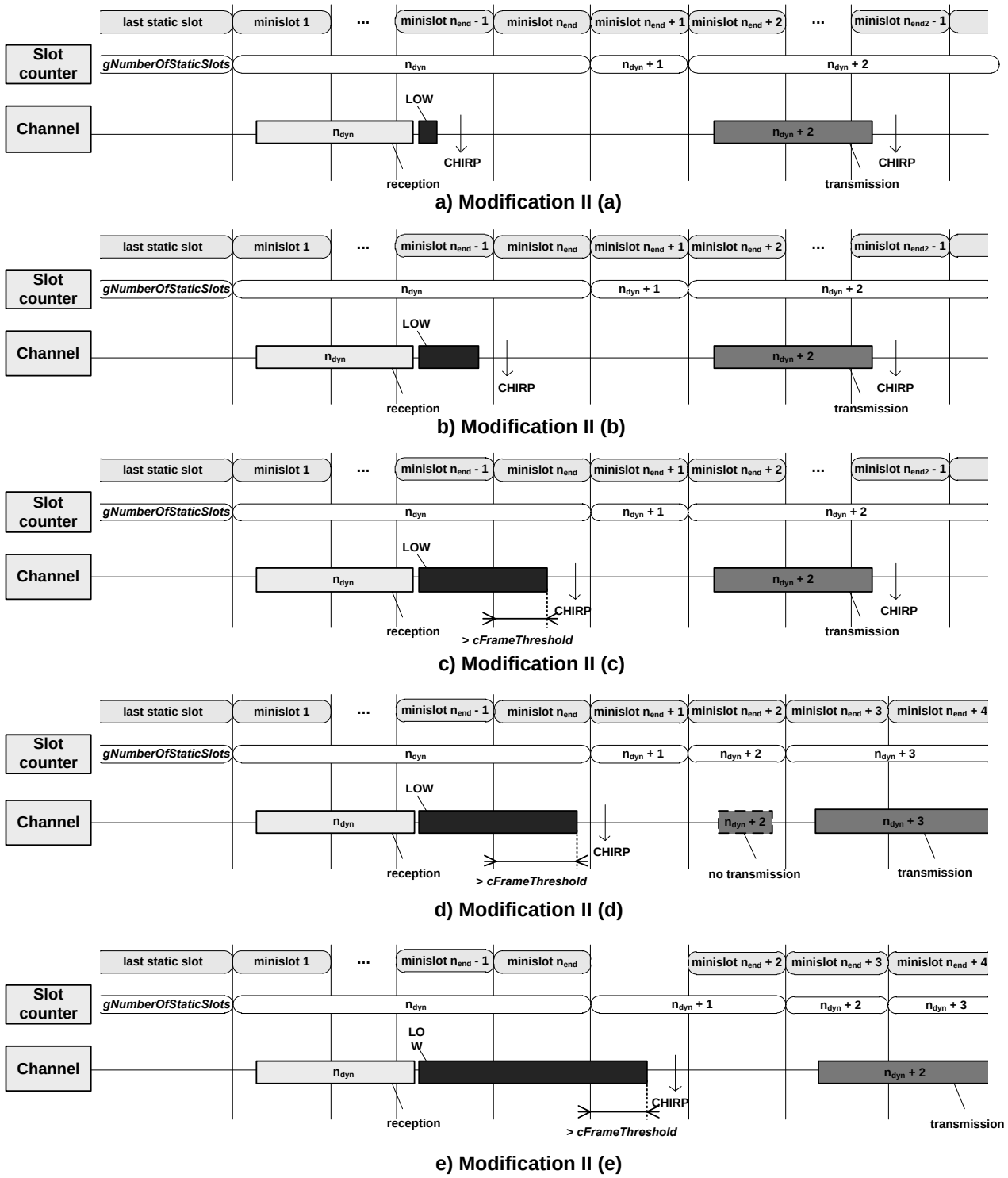


Figure 193 — Frame transmission (2) – gdDynamicSlotIdlePhase = 1, cycle 10

Figure 194 depicts the frame transmission (2) – *gdDynamicSlotIdlePhase* = 2, cycle 10.

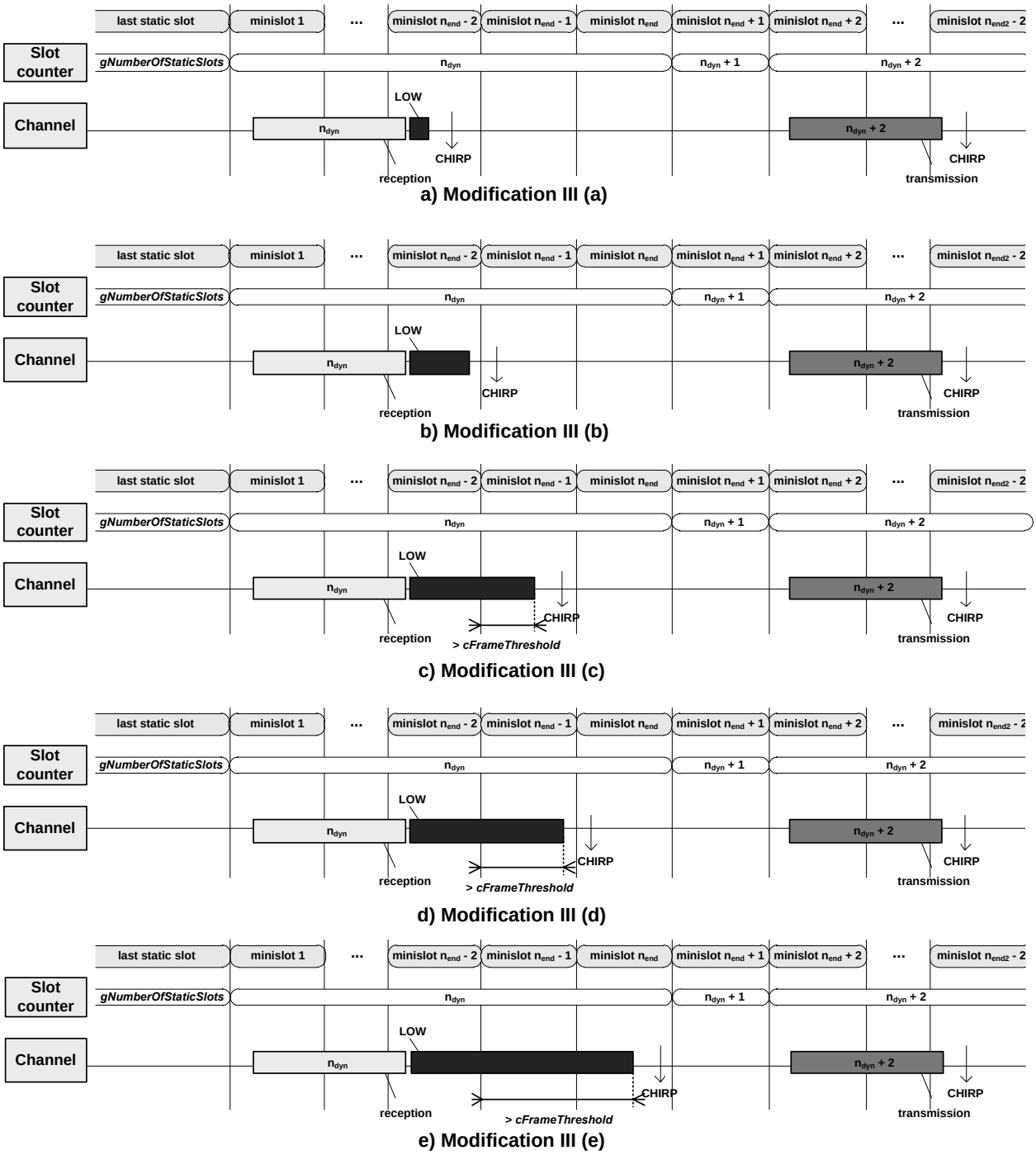


Figure 194 — Frame transmission (2) – gdDynamicSlotIdlePhase = 2, cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The frame transmitted indicator and the slot updated indicator are set accordingly in the transmit buffer status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status. Frame transmission is blocked as expected.

7.7.9.19 Frame transmission after single short noise detected after valid frame reception (3)

— **Test purpose**

Verify correct blocking of frame transmission and correct frame transmission in the dynamic segment after the occurrence of single noise of various length after valid frame reception without CHIRP.

— **Applicability**

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 428, Table 429 and Table 430.

Table 428 — Modification to basic configurations 1a and 1b for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (3)

Parameter	Modification to Basic Configurations			
	1a		1b	
	I	II	I	II
gdDynamicSlotIdlePhase [Minislot]	0	1	0	1
gdMinislot [MT]	6		6	
gdMinislotActionPointOffset [MT]	2		2	
gNumberOfMinislots	328		328	
gdNIT [MT]	15		15	
pLatestTx [Minislot]	188		188	

Table 429 — Modification to basic configurations 2a and 2b for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (3)

Parameter	Modification to Basic Configurations			
	2a		2b	
	I	II	I	II
gdDynamicSlotIdlePhase [Minislot]	0	1	0	1
gdMinislot [MT]	6		6	
gdMinislotActionPointOffset [MT]	2		2	
gNumberOfMinislots	163		163	
gdNIT [MT]	13		13	
pLatestTx [Minislot]	101		101	

Table 430 — Modification to basic configuration 3 for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (3)

Parameter	Modification to Basic Configuration 3	
	I	II
gdDynamicSlotIdlePhase [Minislot]	0	1
gdMinislot [MT]	12	
gdMinislotActionPointOffset [MT]	3	
gNumberOfMinislots	84	
gdNIT [MT]	18	
pLatestTx [Minislot]	37	

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to receive frames in the dynamic slot n_{dyn} on the active channel(s) and transmit a frame in the dynamic slot n_{dyn}+2, n_{dyn}+3 and slot n_{dyn}+4 in continuous mode on the active channel(s).

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frames in dynamic slot n_{dyn} on the configured channel(s). It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmit its frames in the dynamic slots n_{dyn}+2, n_{dyn}+3 and n_{dyn}+4 correctly.

2) The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot $n_{dyn,}$ n_{dyn}+2, n_{dyn}+3 and n_{dyn}+4 after their verification.

3) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase on the active channel(s) starting 5 * *gdBit* after end of DTS and

   (a) ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (b) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (c) ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   Modification I: (a), (b) and (c)

   Modification II: (c)

4) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2, n_{dyn}+3 and n_{dyn}+4 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

5) The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2, n_{dyn}+3 and n_{dyn}+4 after their verification.

6) In cycle 11, the LT does not simulate its frame in dynamic slot n_{dyn}. 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2, n_{dyn}+3 and n_{dyn}+4 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 431 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (3).

Table 431 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification I, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (3)

	Slot n_{dyn}+2		Slot n_{dyn}+3		Slot n_{dyn}+4		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	false	true	true	true	true	true	true
(b)	false	false	false	true	true	true	true
(c)	false	false	false	true	true	true	true

Table 432 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (3).

Table 432 — Transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (3)

	Slot n_{dyn}+2		Slot n_{dyn}+3		Slot n_{dyn}+4		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(a)	true	true	true	true	true	true	false
(b)	true	true	true	true	true	true	false
(c)	true	true	true	true	true	true	false

Table 433 defines the transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification II, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (3).

Table 433 — Transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (3)

	Slot n_{dyn}+2		Slot n_{dyn}+3		Slot n_{dyn}+4		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(c)	false	true	true	true	true	true	true

Table 434 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (3).

Table 434 — Transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (3)

	Slot n_{dyn}+2		Slot n_{dyn}+3		Slot n_{dyn}+4		vDynResync Attempt[Ch]
	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	Frame transmitted indicator	Slot status updated indicator	
(c)	true	true	true	true	true	true	false

Figure 195 depicts the frame transmission (3) – *gdDynamicSlotIdlePhase* = 0, cycle 10.

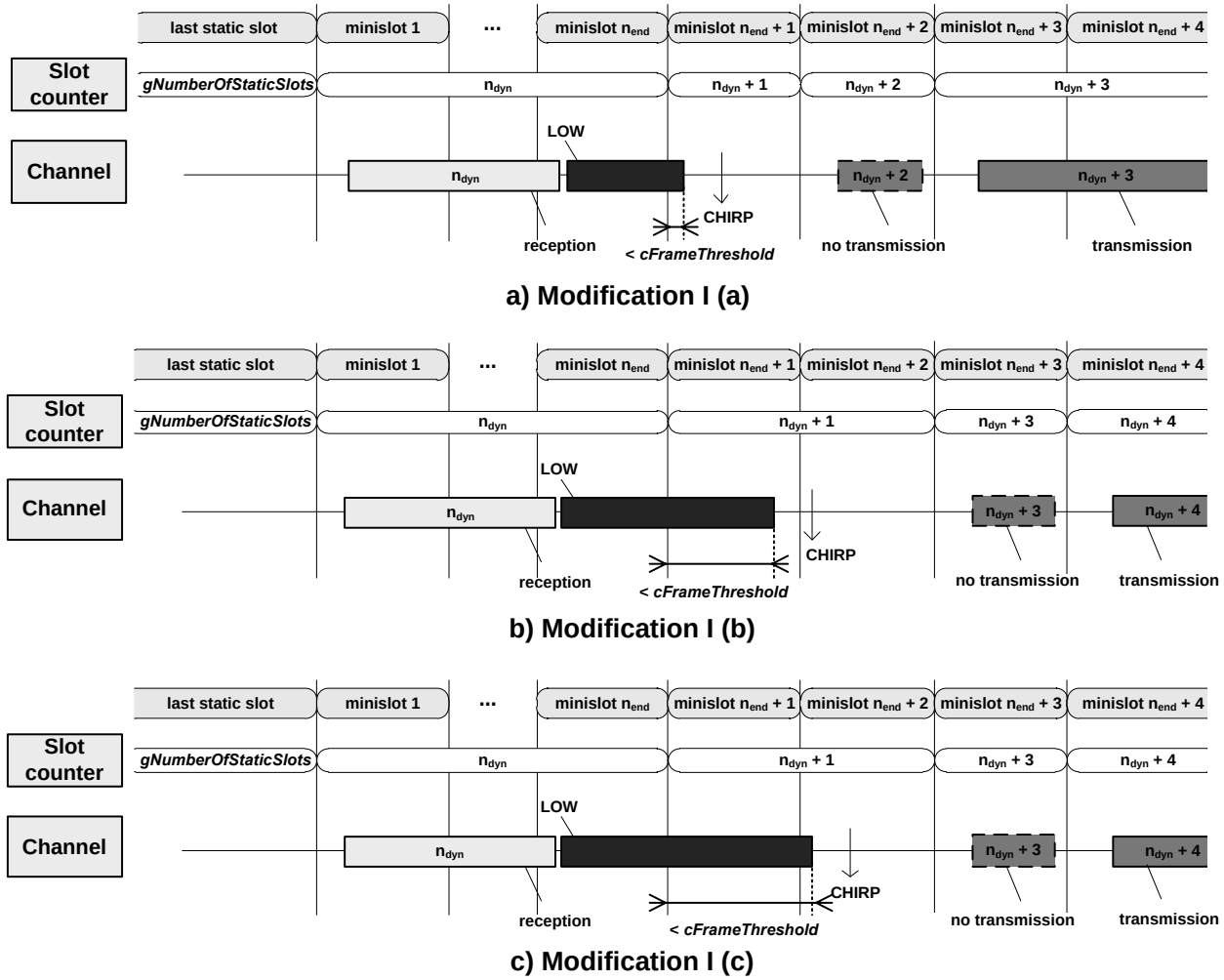


Figure 195 — Frame transmission (3) – gdDynamicSlotIdlePhase = 0, cycle 10

Figure 196 depicts the frame transmission (3) – *gdDynamicSlotIdlePhase* = 1, cycle 10.

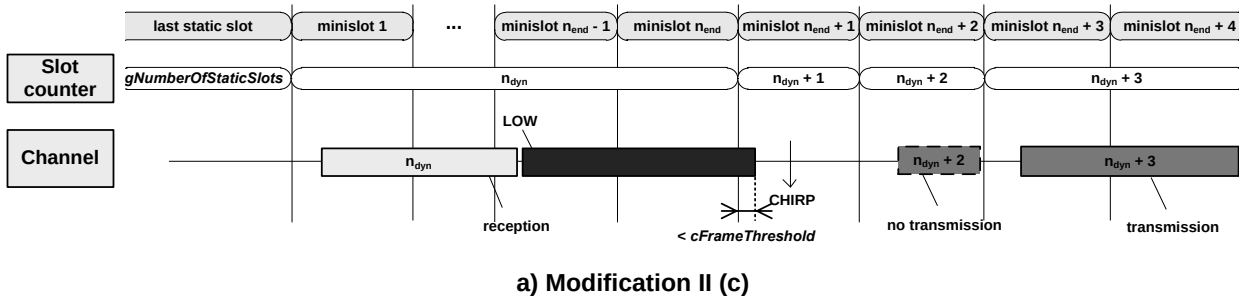


Figure 196 — Frame transmission (3) – gdDynamicSlotIdlePhase = 1, cycle 10

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The frame transmitted indicator and the slot updated indicator are set accordingly in the transmit buffer status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status. Frame transmission is blocked as expected.

7.7.9.20 Frame transmission after single short noise detected after valid frame reception (4)

— Test purpose

Verify correct frame transmission in the dynamic segment after the occurrence of single noise of various length starting in the first minislot of the dynamic slot idle phase following a valid frame reception.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 435.

Table 435 — Modification to basic configurations for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (4)

Parameter	Modification to Basic Configurations					
	1a & 1b		2a & 2b		3	
	I	II	I	II	I	II
gdDynamicSlotIdlePhase [Minislot]	1	2	1	2	1	2
gdMinislot [MT]	11		11		22	
gNumberOfMinislots	179		89		46	
gdNIT [MT]	16		13		14	
pLatestTx [Minislot]	150		60		20	

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to receive frames in the dynamic slot n_{dyn} on the active channel(s) and transmit a frame in the dynamic slot n_{dyn}+1 in continuous mode on the active channel(s).

— Preamble (setup state)

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} on the active channel(s). It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmit its frame in the dynamic slot n_{dyn}+1 correctly.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase on the active channel(s) starting 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and

   (a) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (b) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (c) ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (d) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (e) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (f) ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+3.

   Modification I: (a), (b) and (c)

   Modification II: (a), (b), (d), (e) and (f)

3) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+1 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 436 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (4).

Table 436 — Transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (4)

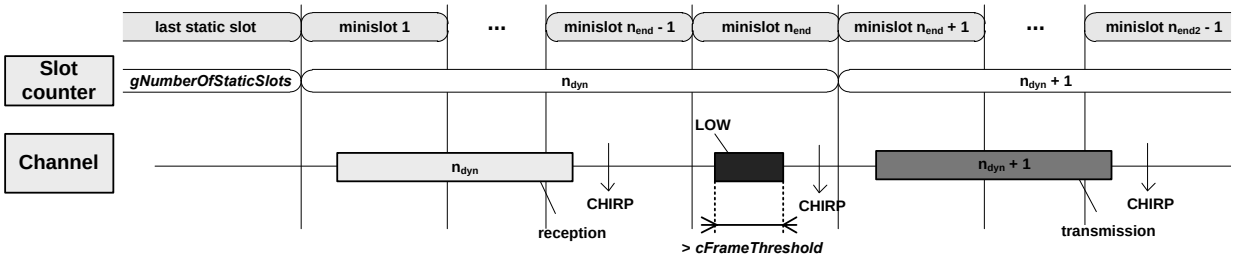
	Cycle 10		
	Frame transmitted indicator	**Slot status updated indicator**	**vDynResyncAttempt[Ch]**
(a)	true	true	false
(b)	true	true	false
(c)	true	true	false

Table 437 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (4).

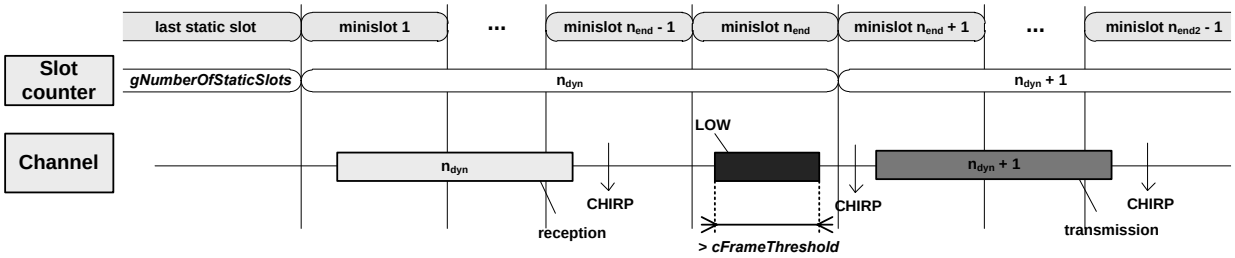
Table 437 — Transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (4)

	Cycle 10		
	Frame transmitted indicator	**Slot status updated indicator**	**vDynResyncAttempt[Ch]**
(a)	true	true	false
(b)	true	true	false
(d)	true	true	false
(e)	true	true	false
(f)	true	true	false

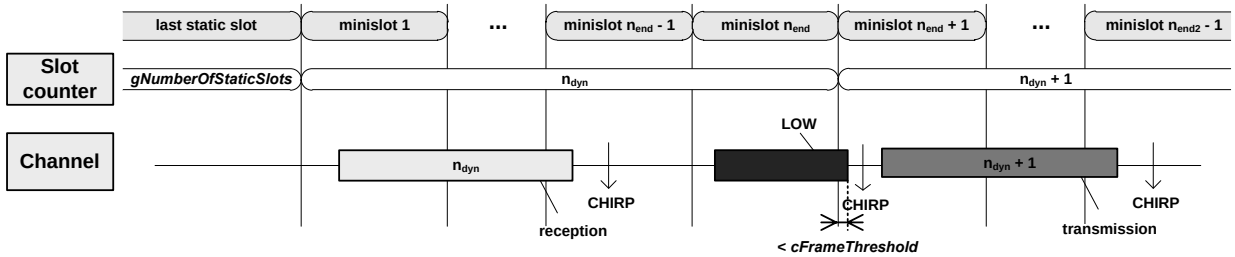
Figure 197 depicts the frame transmission (4) – *gdDynamicSlotIdlePhase* = 1, cycle 10.



a) Modification I (a)



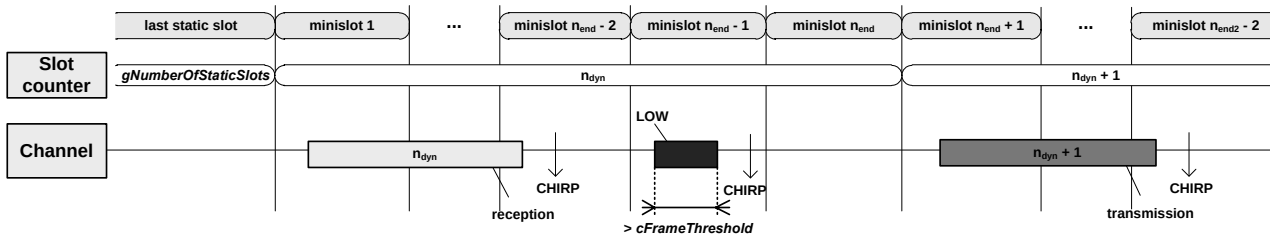
b) Modification I (b)



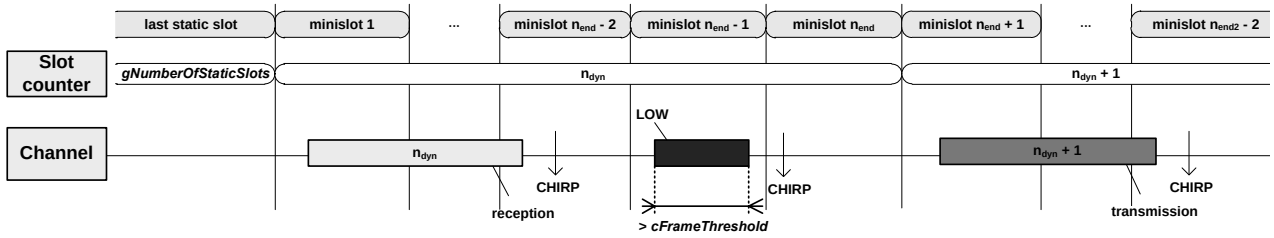
c) Modification I (c)

Figure 197 — Frame transmission (4) – gdDynamicSlotIdlePhase = 1, cycle 10

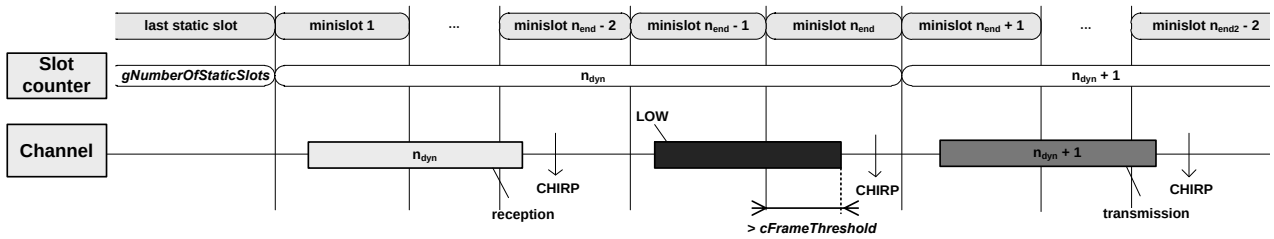
Figure 198 depicts the frame transmission (4) – *gdDynamicSlotIdlePhase* = 2, cycle 10.



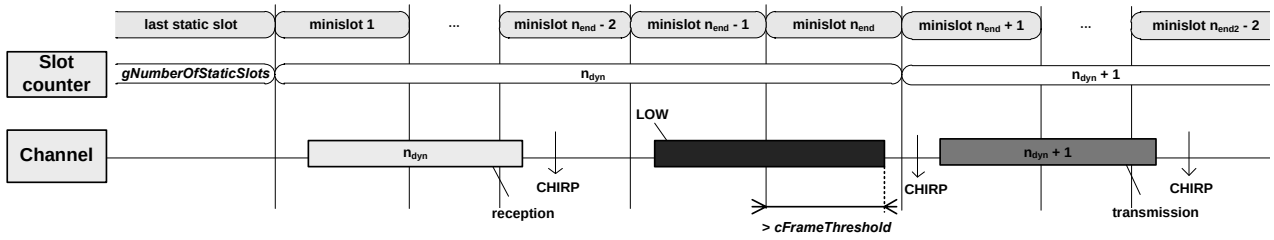
a) Modification II (a)



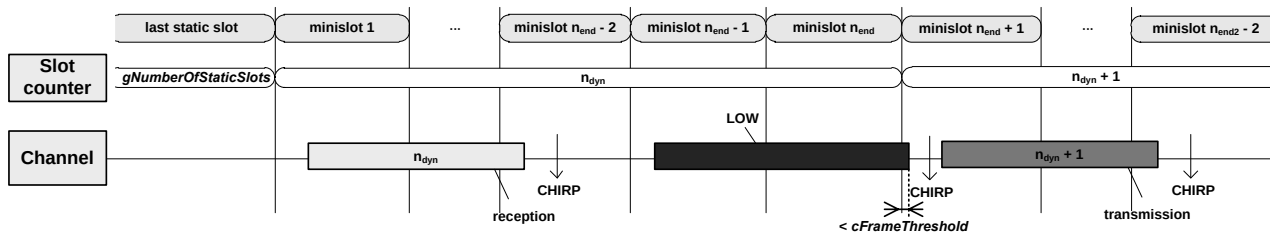
b) Modification II (b)



c) Modification II (d)



d) Modification II (e)



e) Modification II (f)

Figure 198 — Frame transmission (4) – gdDynamicSlotIdlePhase = 2, cycle 10

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The frame transmitted indicator and the slot updated indicator are set accordingly in the transmit buffer status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status.

7.7.9.21 Frame transmission after single short noise detected after valid frame reception (5)

— Test purpose

Verify correct blocking of frame transmission and correct frame transmission in the dynamic segment after the occurrence of single noise of various length starting in the minislot following a valid frame reception.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 438.

Table 438 — Modification to basic configurations for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (5)

Parameter	Modification to Basic Configurations								
	1a & 1b			2a & 2b			3		
	I	II	III	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2	0	1	2
gdMinislot [MT]	11			11			22		
gNumberOfMinislots	179			89			46		
gdNIT [MT]	16			13			14		
pLatestTx [Minislot]	150			60			20		

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to receive frames in the dynamic slot n_{dyn} on the active channel(s) and transmit frame in the dynamic slot n_{dyn}+2 and n_{dyn}+3 in continuous mode on the active channel(s).

— Preamble (setup state)

Preamble III.

— Test execution

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} on the active channel(s). It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmit its frames in the dynamic slots n_{dyn}+2 and n_{dyn}+3 correctly.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase on the active channel(s) starting 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and

   (a) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (b) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (c) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (d) ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (e) ending 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+3.

3) In cycle 10, 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

   The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 after their verification.

4) In cycle 11, the LT does not simulate its frame in dynamic slot n_{dyn}. 500 µT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot ndyn+2 and ndyn+3 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 439 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (5).

Table 439 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification I, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (5)

	Slot n_{dyn}+2		**Slot n_{dyn}+3**		***vDynResync Attempt[Ch]***
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 440 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (5).

Table 440 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification I, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (5)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 441 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (5).

Table 441 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification II, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (5)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	true	true	true	true	false
(b)	false	true	true	true	true
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 442 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (5).

Table 442 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification II, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (5)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 443 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch] (*Modification III, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (5).

Table 443 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification III, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (5)

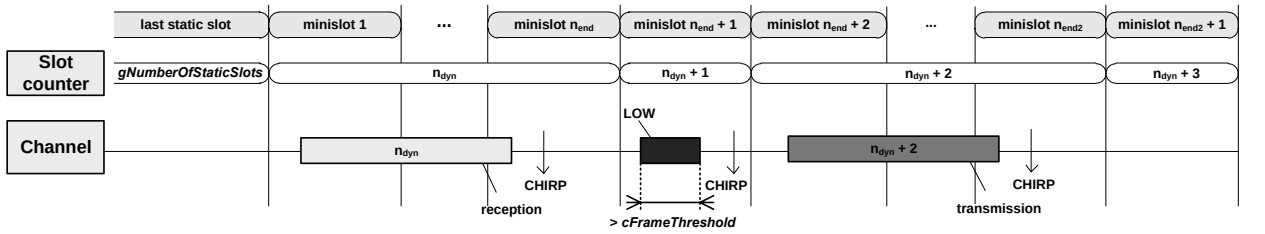
	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	false	true	true	true	true
(e)	true	true	true	true	false

Table 444 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification III, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (5).

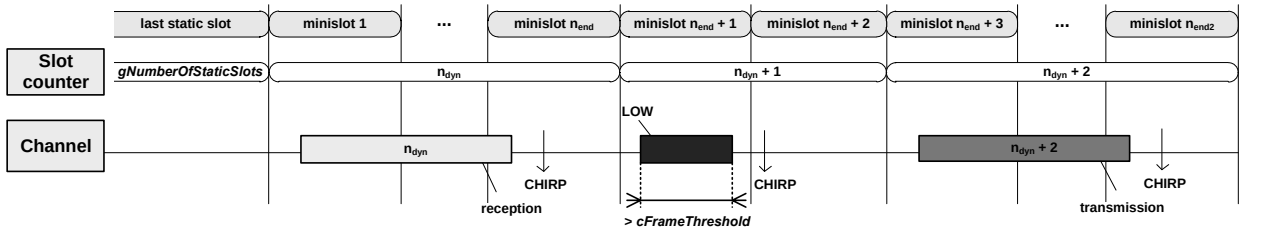
Table 444 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification III, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (5)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

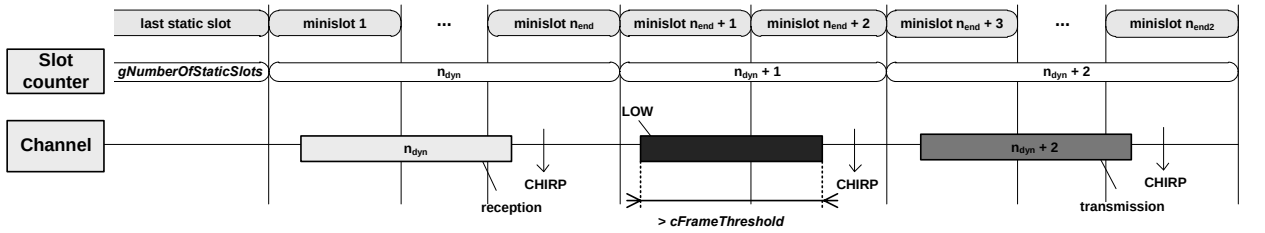
Figure 199 depicts the frame transmission (5) – *gdDynamicSlotIdlePhase* = 0, cycle 10.



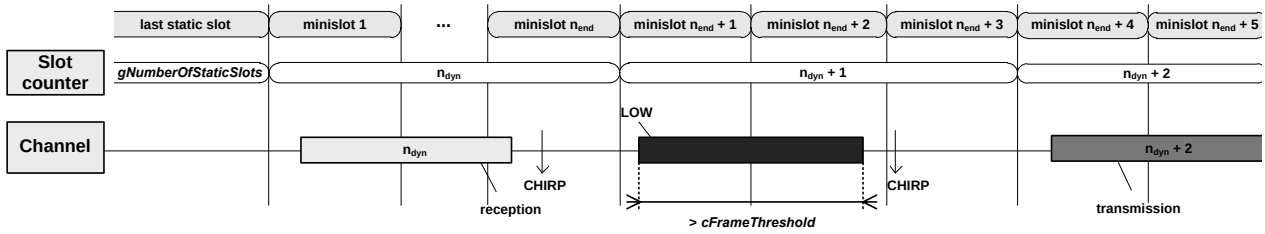
a) Modification I (a)



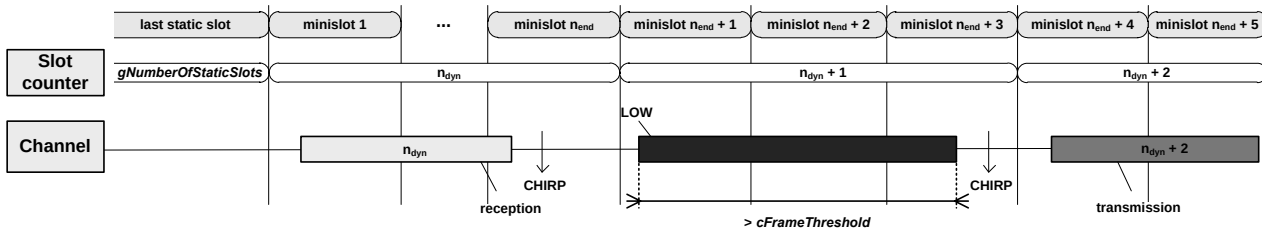
b) Modification I (b)



c) Modification I (c)



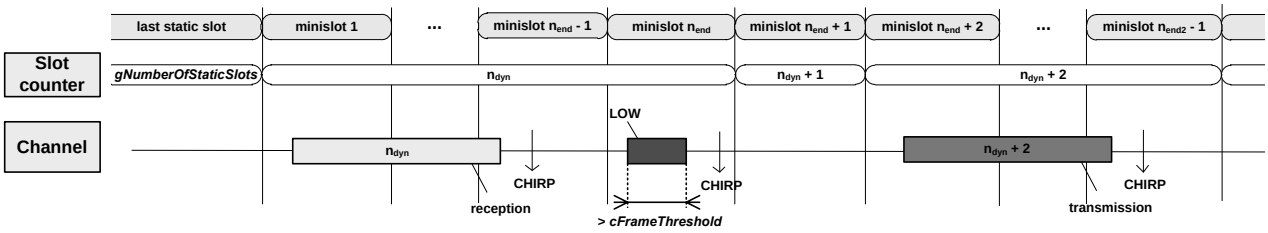
d) Modification I (d)



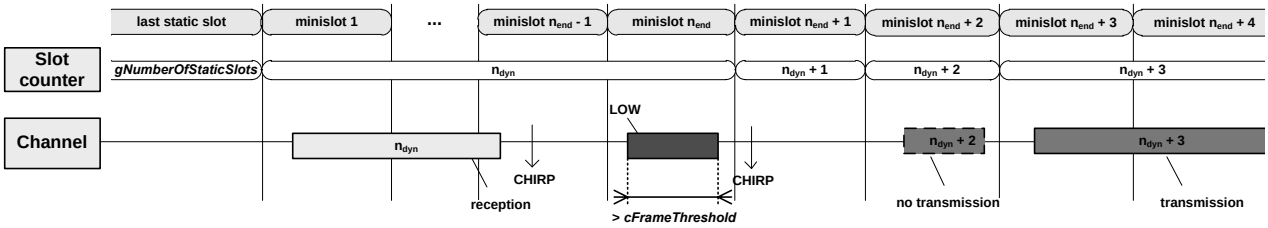
e) Modification I (e)

Figure 199 — Frame transmission (5) – gdDynamicSlotIdlePhase = 0, cycle 10

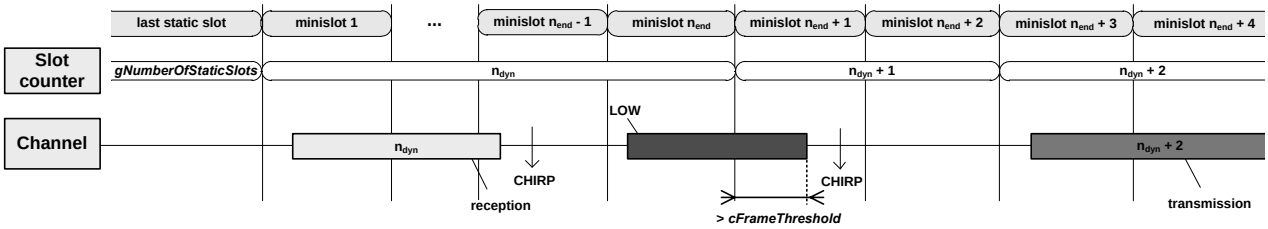
Figure 200 depicts the frame transmission (5) – *gdDynamicSlotIdlePhase* = 1, cycle 10.



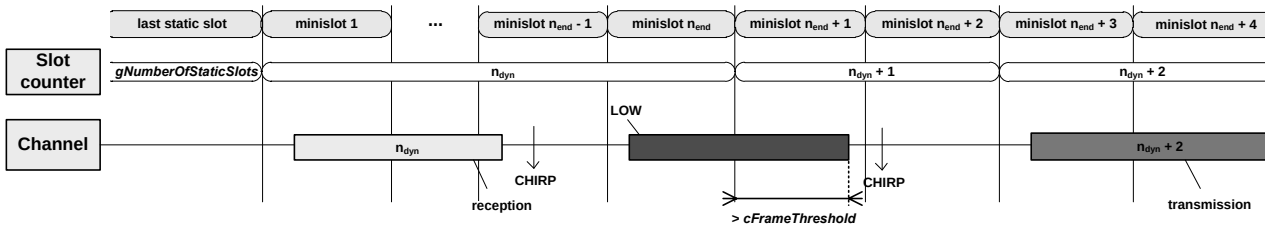
a) Modification II (a)



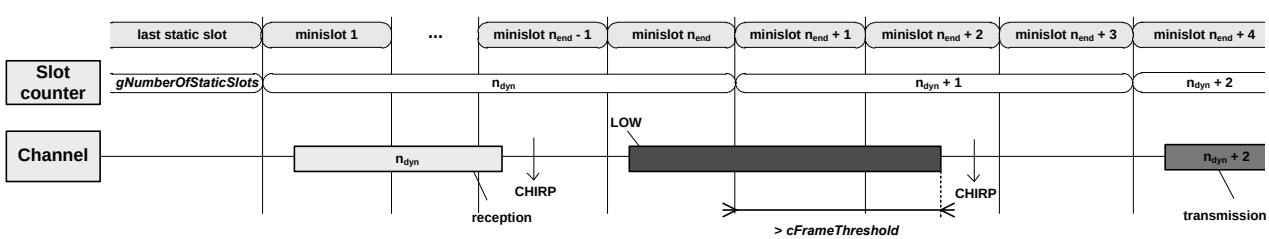
b) Modification II (b)



c) Modification II (c)



d) Modification II (d)



e) Modification II (e)

Figure 200 — Frame transmission (5) – gdDynamicSlotIdlePhase = 1, cycle 10

Figure 201 depicts the frame transmission (5) – *gdDynamicSlotIdlePhase* = 2, cycle 10.

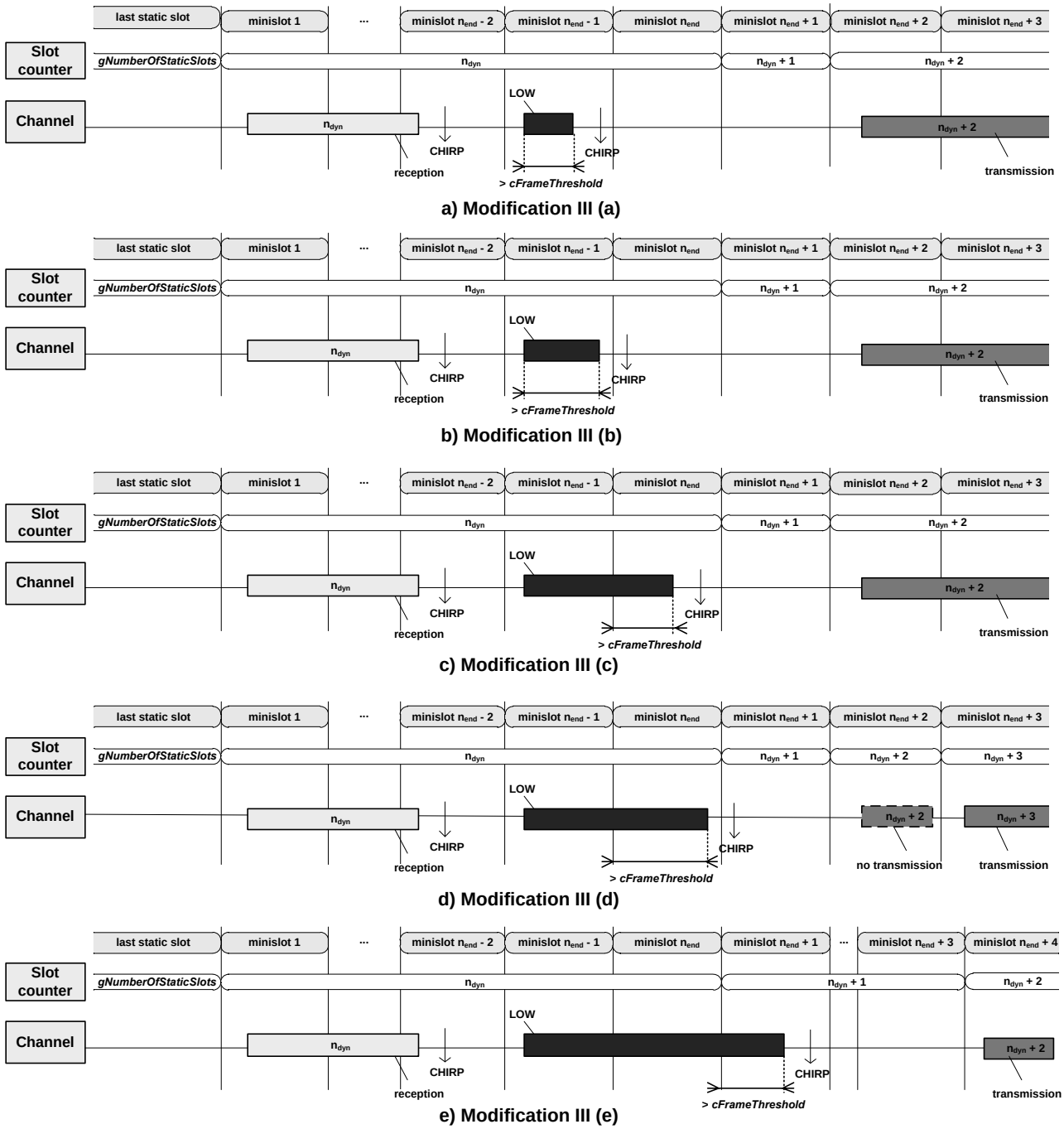


Figure 201 — Frame transmission (5) – gdDynamicSlotIdlePhase = 2, cycle 10

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The frame transmitted indicator and the slot updated indicator are set accordingly in the transmit buffer status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status. Frame transmission is blocked as expected.

7.7.9.22 Frame transmission after single short noise detected after valid frame reception (6)

— Test purpose

Verify correct blocking of frame transmission and correct frame transmission in the dynamic segment after the occurrence of single noise of various length starting in the minislot following a valid frame reception.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 445, Table 446 and Table 447.

Table 445 — Modification to basic configurations 1a and 1b for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

Parameter	Modification to Basic Configurations					
	1a			1b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gdMinislot [MT]	6			6		
gdMinislotActionPointOffset [MT]	2			2		
gNumberOfMinislots	328			328		
gdNIT [MT]	15			15		
pLatestTx [Minislot]	188			188		

Table 446 — Modification to basic configurations 2a and 2b for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

Parameter	Modification to Basic Configurations					
	2a			2b		
	I	II	III	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2	0	1	2
gdMinislot [MT]	6			6		
gdMinislotActionPointOffset [MT]	2			2		
gNumberOfMinislots	163			163		
gdNIT [MT]	13			13		
pLatestTx [Minislot]	101			101		

Table 447 — Modification to basic configuration 3 for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

Parameter	Modification to Basic Configuration 3		
	I	II	III
gdDynamicSlotIdlePhase [Minislot]	0	1	2
gdMinislot [MT]	12		
gdMinislotActionPointOffset [MT]	3		
gNumberOfMinislots	84		
gdNIT [MT]	18		
pLatestTx [Minislot]	37		

n_{dyn} = *gNumberOfStaticSlots* + 1.

n_{end} = ceil((*gdTSSTransmitter* + *cdFSS* + 50 + PayloadLength * 20 + 30 + *cdFES* + 1) * *gdBit* / (*gdMinislot* * *gdMacrotick*)) + *gdDynamicSlotIdlePhase* + 1.

The IUT is configured to receive frames in the dynamic slot n_{dyn} on the active channel(s) and transmit frame in the dynamic slot n_{dyn}+2 and slot n_{dyn}+3 in continuous mode on the active channel(s).

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In cycle 9, the LT simulates a frame in dynamic slot n_{dyn} on the active channel(s). It is verified (UT) that the valid frame flag / indicator is true in the slot status of slot n_{dyn} and in the aggregated channel status in the NIT of the cycle. It is verified (LT) that the IUT transmit its frames in the dynamic slot n_{dyn}+2 and n_{dyn}+3 correctly.

The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 after their verification.

2) In cycle 10, the LT simulates a frame in dynamic slot n_{dyn} and a low phase on the active channel(s)

   (a) starting 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 15 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (b) starting 15 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1.

   (c) starting 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (d) starting 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+2.

   (e) starting 5 * *gdBit* before end of the minislot n_{end}-*gdDynamicSlotIdlePhase*+1 and ending 5 * *gdBit* after begin of the minislot n_{end}-*gdDynamicSlotIdlePhase*+3.

   Modification I: (a), (b), (c), (d) and (e)

   Modification II: (c), (d) and (e)

   Modification III: (e)

3) In cycle 10, 500uT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

   The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 after their verification.

4) In cycle 11, the LT does not simulate its frame in dynamic slot n_{dyn}. 500 uT after the end of the dynamic segment and before cycle end, it is verified (UT) that the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn}+2 and n_{dyn}+3 hold the values as listed in the tables below. It is also verified (UT) in the dynamic segment status that *vDynResyncAttempt[Ch]* holds the values as listed in the tables below.

Table 448 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (6).

Table 448 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification I, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

	Slot $n_{dyn}+2$		Slot $n_{dyn}+3$		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	false	true	true	true	true
(b)	false	false	false	true	true
(c)	false	false	false	true	true
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 449 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification I, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (6).

Table 449 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification I, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

	Slot $n_{dyn}+2$		Slot $n_{dyn}+3$		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(a)	true	true	true	true	false
(b)	true	true	true	true	false
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 450 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (6).

Table 450 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification II, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(c)	false	true	true	true	true
(d)	false	false	false	true	true
(e)	false	false	false	true	true

Table 451 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification II, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (6).

Table 451 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification II, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(c)	true	true	true	true	false
(d)	true	true	true	true	false
(e)	true	true	true	true	false

Table 452 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification III, Cycle 10) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (6).

Table 452 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification III, Cycle 10) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(e)	false	true	true	true	true

Table 453 defines the transmit buffer status, slot status data and *vDynResyncAttempt[Ch]* (Modification III, Cycle 11) for *dynamic segment robustness* – frame transmission after single short noise detected after valid frame reception (6).

Table 453 — Transmit buffer status, slot status data and vDynResyncAttempt[Ch] (Modification III, Cycle 11) for dynamic segment robustness – frame transmission after single short noise detected after valid frame reception (6)

	Slot n_{dyn}+2		Slot n_{dyn}+3		*vDynResync Attempt[Ch]*
	Frame transmitted indicator	**Slot status updated indicator**	**Frame transmitted indicator**	**Slot status updated indicator**	
(e)	true	true	true	true	false

Figure 202 depicts the frame transmission (6) – *gdDynamicSlotIdlePhase* = 0, cycle 10.

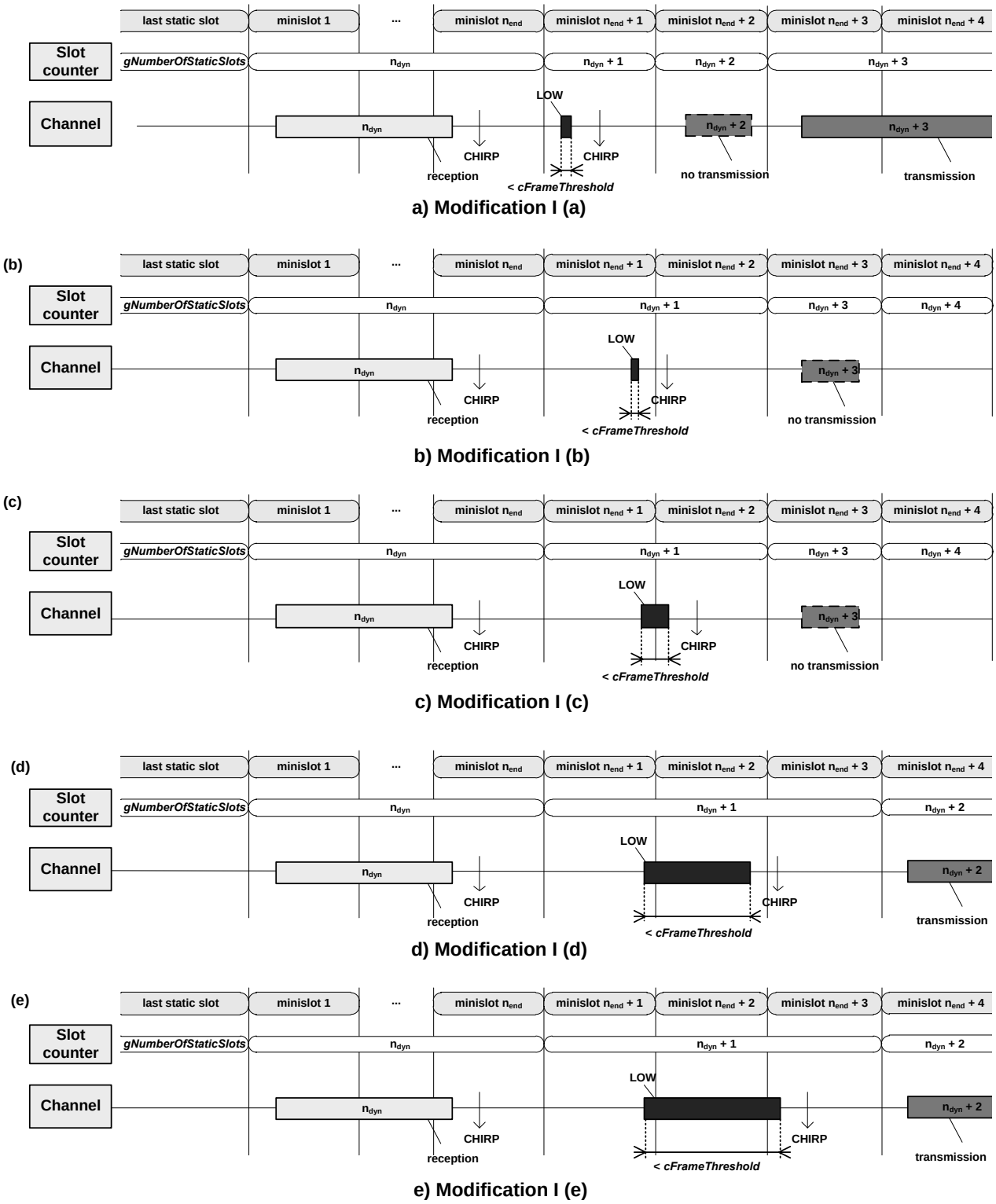


Figure 202 — Frame transmission (6) – gdDynamicSlotIdlePhase = 0, cycle 10

Figure 203 depicts the frame transmission (6) – *gdDynamicSlotIdlePhase* = 1, cycle 10.

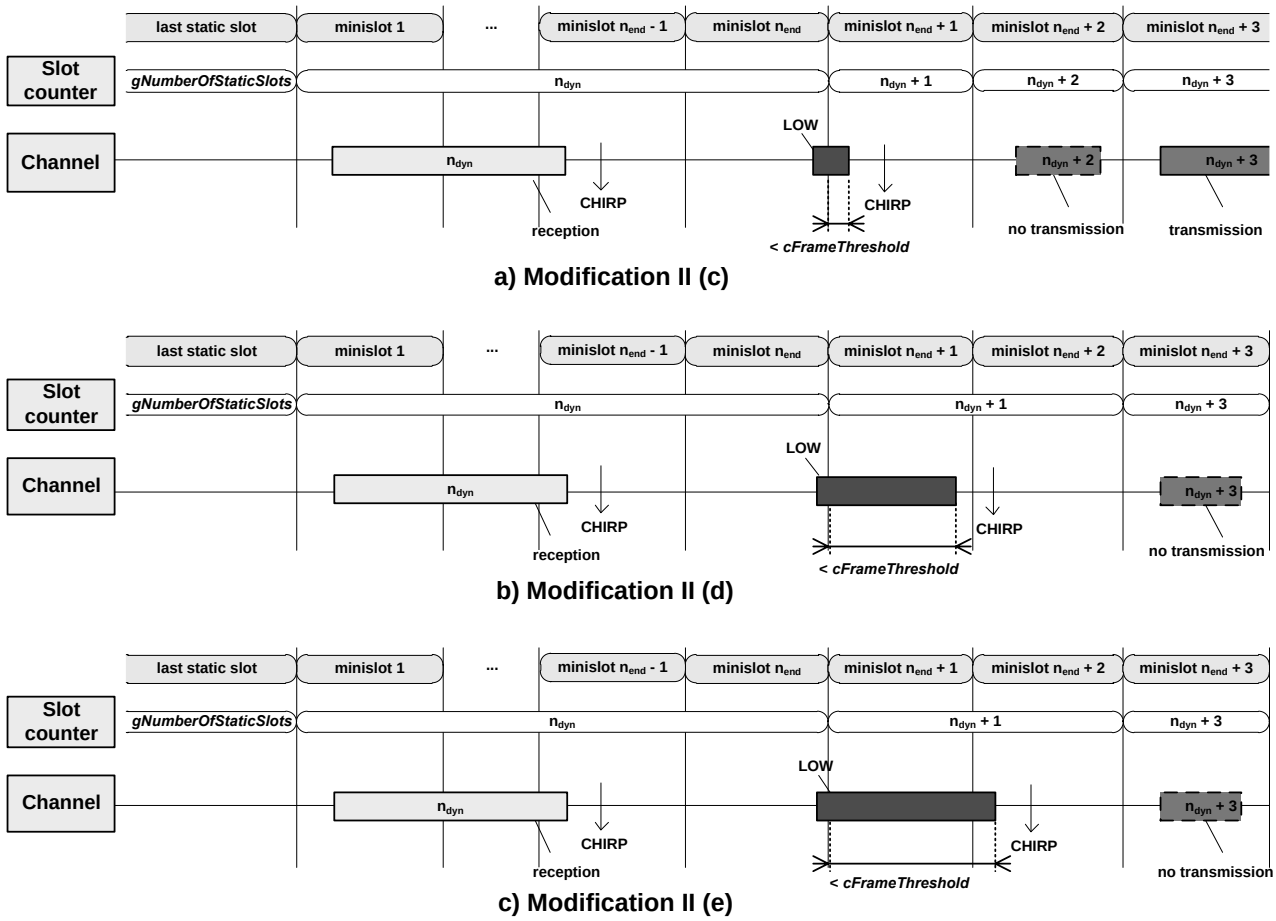


Figure 203 — Frame transmission (6) – gdDynamicSlotIdlePhase = 1, cycle 10

Figure 204 depicts the frame transmission (6) – *gdDynamicSlotIdlePhase* = 2, cycle 10.

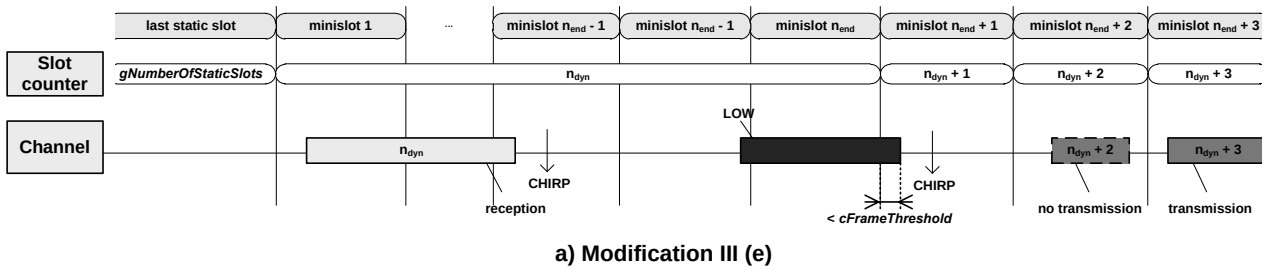


Figure 204 — Frame transmission (6) – gdDynamicSlotIdlePhase = 2, cycle 10

— **Postamble**

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The frame transmitted indicator and the slot updated indicator are set accordingly in the transmit buffer status. The IUT performs slot counting as expected and sets the value of *vDynResyncAttempt[Ch]* accordingly in the dynamic segment status. Frame transmission is blocked as expected.

7.7.10 Transmission conflict in the static and dynamic segment

— Test purpose

Verify correct handling of a transmission conflict in the static and dynamic segment. The IUT shall recognize a transmission conflict, if decoding is in progress when the IUT starts frame transmission. The IUT shall not abort its transmission upon detection of a transmission conflict. The IUT shall not process frames received while the IUT's transmission is ongoing.

— Applicability

SC, DC.

— Configuration

All basic configurations using the modifications as listed in Table 454.

Table 454 — Modification to basic configurations for transmission conflict in the static and dynamic segment

Parameter	Modification to Basic Configurations			
	1a & 1b	**2a**	**2b**	**3**
gdActionPointOffset [MT]	26	26	26	26
gdMinislot [MT]	63	63	63	63
gdMinislotActionPointOffset [MT]	31	31	31	31
gdStaticSlot [MT]	57	57	57	57
gdSymbolWindow [MT]	54	53	54	60
gdSymbolWindowActionPointOffset [MT]	31	31	31	31
gNumberOfMinislots	63	24	24	24
gNumberOfStaticSlots	16	16	16	16
gPayloadLengthStatic [two-byte word]	1	1	1	1
gdNIT [MT]	65	23	22	16
pLatestTx [Minislot]	57	18	18	14
pMacroInitialOffset[A] [MT]	28	27	27	28
pMacroInitialOffset[B] [MT]	28	27	27	28

For dual channel test execution a receive buffer and a transmit buffer are assigned to slot 1 / channel A&B in the IUT.

For single channel test execution a receive buffer and a transmit buffer are assigned to slot 1 on the available channel in the IUT.

n_{dyn} = *gNumberOfStaticSlots* + 1.

The IUT is configured to transmit a frame in dynamic slot n_{dyn}.

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In the NIT of cycle 9, it is verified (UT) that the syntax error flag / indicator, the content error flag / indicator, the slot boundary violation flag / indicator and the transmission conflict flag / indicator are false in the transmit buffer status of slot 1 and in the aggregated channel status. It is further verified that the valid frame flag is false in the transmit buffer status of slot 1 and the valid frame indicator is true in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.
   It is also verified that the slot status updated indicator is false in the slot status data of the receive buffer of slot 1.

2) In cycle 10, the LT simulates a frame in slot 1 starting:

   (a) ceil(*gdTSSTransmitter* / 2) [*gdBit*] before the action point of slot 1.

   (b) *gdTSSTransmitter* + 2 [*gdBit*] before the action point of slot 1.

   (c) *gdTSSTransmitter* + 50 (header) + 2 [*gdBit*] before the action point of slot 1.

   (d) *gdTSSTransmitter* + 50 (header) + 1 + 2 [*gdBit*] before the action point of slot 1.

   (e) *gdTSSTransmitter* + 50 (header) + *cdBSS* + 2 [*gdBit*] before the action point of slot 1.

   (f) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + 2 [*gdBit*] before the action point of slot 1.

   (g) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + 1 + 2 [*gdBit*] before the action point of slot 1.

3) In cycle 10, it is verified (LT) that the IUT transmits its frame in slot 1 disregarding the transmission conflict.

4) In the NIT of cycle 10, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator and the slot boundary violation flag / indicator are false and the transmission conflict flag / indicator is true in the transmit buffer status of slot 1 and in the aggregated channel status. It is further verified that the valid frame flag is false in the transmit buffer status of slot 1 and the valid frame indicator is true in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.
   It is also verified that the slot status updated indicator is false in the slot status data of the receive buffer of slot 1.

5) In cycle 11, the LT simulates a frame in slot n_{dyn} starting before the action point of slot n_{dyn}.

   (a,b,c,d) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + *cdFES* + 2 [*gdBit*] before the action point of slot n_{dyn}.

(e,f,g) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + *cdFES* + 1 + 2 [*gdBit*] before the action point of slot n_{dyn}.

6) In cycle 11, it is verified (LT) that the IUT transmits its frame in slot n_{dyn} disregarding the transmission conflict.

7) In the NIT of cycle 11, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator and the slot boundary violation flag / indicator are false and the transmission conflict flag / indicator is true in the transmit buffer status of slot n_{dyn} and in the aggregated channel status. It is further verified that the valid frame flag is false in the transmit buffer status of slot 1 and the valid frame indicator is true in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot n_{dyn}. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot n_{dyn} after their verification.
It is also verified that the slot status updated indicator is false in the slot status data of the receive buffer of slot 1.

8) In cycle 12, the LT simulates a valid MTS in slot 1 starting *gdTSSTransmitter* + ceil(*cdCAS* / 2) [*gdBit*] before the action point of slot 1.

9) In cycle 12, it is verified (LT) that the IUT transmits its frame in slot 1 disregarding the transmission conflict.

10) In the NIT of cycle 12, it is verified (UT) that the syntax error flag / indicator is true, the content error flag / indicator and the slot boundary violation flag / indicator are false and the transmission conflict flag / indicator is true in the transmit buffer status of slot 1 and in the aggregated channel status. It is further verified that the valid frame flag is false in the transmit buffer status of slot 1 and the valid frame indicator is true in the aggregated channel status. It is also verified (UT) that the frame transmitted indicator and the slot status updated indicator are true in the transmit buffer status of slot 1. The UT resets the frame transmitted indicator and the slot status updated indicator in the transmit buffer status of slot 1 after their verification.
It is also verified that the slot status updated indicator is false in the slot status data of the receive buffer of slot 1.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT does not stop frame transmission and sets the proper indication upon detection of a transmission conflict in the static segment. The IUT recognizes a transmission conflict, if decoding or channel idle detection is in progress when the IUT starts frame transmission. The IUT does not process a received frame causing a transmission conflict.

7.7.11 Transmission conflict in the symbol window

— Test purpose

Verify correct handling of a transmission conflict in the symbol window. The IUT shall recognize a transmission conflict, if decoding is in progress when the IUT starts symbol transmission. The IUT shall not abort its transmission upon detection of a transmission conflict. The IUT shall not process frames received while the IUT's transmission is ongoing.

— Applicability

SC, DC.

— **Configuration**

All basic configurations using the modifications as listed in Table 455.

Table 455 — Modification to basic configurations for transmission conflict in the symbol window

Parameter	Modification to Basic Configurations		
	1a & 1b	**2a & 2b**	**3**
gdSymbolWindow [MT]	150	150	145
gdSymbolWindowActionPointOffset [MT]	63	63	63
gNumberOfMinislots	206	121	127
gPayloadLengthStatic [two-byte word]	1	1	1
gdNIT [MT]	21	18	16
pLatestTx [Minislot]	175	82	50

— **Preamble (setup state)**

Preamble III.

— **Test execution**

1) In the NIT of cycle 9, it is verified (UT) that the syntax error flag / indicator, the slot boundary violation flag / indicator and the transmission conflict flag / indicator are false in the symbol window status of slot 1 and in the aggregated channel status.

2) The UT requests the IUT to transmit an MTS in the symbol window of cycle 10.

3) In the symbol window of cycle 10, the LT simulates a frame with a DTS of 5 [*gdBit*] starting:

   (a) ceil(*gdTSSTransmitter* / 2) [*gdBit*] before the action point of the symbol window.

   (b) *gdTSSTransmitter* + 2 [*gdBit*] before the action point of the symbol window.

   (c) *gdTSSTransmitter* + 50 (header) + 2 [*gdBit*] before the action point of the symbol window.

   (d) *gdTSSTransmitter* + 50 (header) + 1 + 2 [*gdBit*] before the action point of the symbol window.

   (e) *gdTSSTransmitter* + 50 (header) + *cdBSS* + 2 [*gdBit*] before the action point of the symbol window.

   (f) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + 2 [*gdBit*] before the action point of the symbol window.

   (g) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + 1 + 2 [*gdBit*] before the action point of the symbol window.

   (h) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + *cdFES* + 2 [*gdBit*] before the action point of the symbol window.

   (i) *gdTSSTransmitter* + 50 (header) + 20 (payload) + 30 (trailer) + *cdFES* + 1 + 2 [*gdBit*] before the action point of the symbol window.

4) In cycle 10, it is verified (LT) that the IUT transmits its MTS in the symbol window disregarding the transmission conflict.

5) 500 µT after start of the NIT and before end of the NIT of cycle 10, it is verified (UT) that the syntax error flag / indicator is true, the slot boundary violation flag / indicator is false and the transmission conflict flag / indicator is true in the symbol window status and in the aggregated channel status.

6) The UT requests the IUT to transmit an MTS in the symbol window of cycle 11.

7) In cycle 11, the LT simulates a valid MTS in the symbol window starting *gdTSSTransmitter* + ceil(*cdCAS* / 2) [*gdBit*] before the action point of the symbol window.

8) In cycle 11, it is verified (LT) that the IUT transmits its MTS in the symbol window disregarding the transmission conflict.

9) In the NIT of cycle 11, it is verified (UT) that the syntax error flag / indicator is true, the slot boundary violation flag / indicator is false and the transmission conflict flag / indicator is true in the symbol window status and in the aggregated channel status.

— Postamble

The UT sets the IUT into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The IUT does not stop symbol transmission and sets the proper indication upon detection of a transmission conflict in the symbol window. The IUT recognizes a transmission conflict, if decoding is in progress when the IUT starts symbol transmission. The IUT does not process a received frame causing a transmission conflict.

7.8 Optional TT-E feature

7.8.1 Independent behaviour of gateway source and gateway sink

— Test purpose

Verify correct independent startup behaviour of the Source:CC and Sink:CC if *pExternalSync* = false.

— Applicability

TT-E.

— Configuration

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications as listed in Table 456.

Table 456 — Modification to basic configurations for independent behaviour of gateway source and gateway sink, Sink: CC

Parameter	Modification
pExternalSync	false
pSecondKeySlotID	3
pTwoKeySlotMode	true

— **Preamble (setup state)**

1) The UT resets the Sink:CC.

2) The UT resets the Source:CC.

3) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

4) The UT configures the Sink:CC with the given configuration.

5) The UT sets the Source:CC into *POC:config* with the CHI command CONFIG.

6) The UT configures the Source:CC with the given configuration.

7) The UT sets the Sink:CC into *POC:ready* state (*vPOC!State* = *READY*) with the CHI command CONFIG_COMPLETE.

8) The UT sets the Source:CC into *POC:ready* state (*vPOC!State* = *READY*) with the CHI command CONFIG_COMPLETE.

9) The UT clears the *vColdstartInhibit* flag of the Sink:CC with the CHI command ALLOW_COLDSTART.

— **Test execution**

1) The UT initiates the startup procedure of the Source:CC and the Sink:CC with the CHI command RUN.

2) It is verified (UT) that the Source:CC and the Sink:CC are in the *POC:coldstart listen* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *COLDSTART_LISTEN*) 2 500 µT after the UT has issued the CHI command RUN for both Source:CC and Sink:CC.

3) The LT represents the leading coldstart node in the source cluster and simulates a CAS 0,9 * *pdListenTimeout* ± 0,05 * *pdListenTimeout* after the CHI command RUN of Sink:CC.

4) In slot 3 of cycles 0 to 6 of the source cluster, the LT simulates a startup frame in the source cluster with the null frame indicator, the reserved bit and the payload preamble indicator set to '0', holding a payload of 0x00 in all payload bytes.

5) In cycle 6 of the sink cluster, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster, it is verified (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

6) In slot 3 of cycle 7 of the source cluster, the LT simulates a startup frame in the source cluster with the null frame indicator set to '1', the reserved bit and the payload preamble indicator set to 0, holding the respective payload.

7) In the static segment of cycle 7 of the source cluster, the UT issues the command DEFERRED_HALT to the Source:CC.

8) In cycle 7 of the sink cluster, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster, it is verified (UT) that the Source:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and locally synchronized (*vExternalSync* = false).

9) In cycles 7 and 8 of the sink cluster, it is verified (LT) that the Sink:CC transmits startup frames in slot 1 and slot 3.

10) In cycle 8 of the sink cluster, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster, it is verified (UT) that the Source:CC is in the *POC:halt* state (*vPOC!State* = *HALT*) and that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

— Postamble

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

The Source:CC is already in the *POC:halt* state.

— Pass criteria

The Sink:CC successfully performs its startup independent of the Source:CC and all POC states of the Source:CC and the Sink:CC are set accordingly.

7.8.2 Clock synchronisation

7.8.2.1 Cycle start supervision

— Test purpose

Verify correct behaviour of the cycle start supervision, that is always a constant delay between the cycle starts of the Sink:CC and Source:CC.

— Applicability

TT-E.

— Configuration

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications as listed in Table 457, Table 458 and Table 459.

Table 457 — Modification to basic configurations 1a and 1b for cycle start supervision

Parameter	Modification to Basic Configurations	
	1a	**1b**
gExternOffsetCorrection [µs]	0,175	0,0875
pClusterDriftDamping [µT]	0	0
pExternOffsetCorrection [µT]	7[a]	7[a]
pFallBackInternal	true	true
pKeySlotOnlyEnabled	true	true
[a] *pExternOffsetCorrection* = 7 µT which is an intentional protocol constraints violation. For testability reasons *pExternOffsetCorrection* should be configurable in the Sink:CC, even a constraint in the data link layer specification requires *pExternOffsetCorrection* = 0 µT.		

Table 458 — Modification to basic configurations 2a and 2b for cycle start supervision

Parameter	Modification to Basic Configurations	
	2a	**2b**
gExternOffsetCorrection [µs]	0,175	0,35
pClusterDriftDamping [µT]	0	0
pExternOffsetCorrection [µT]	7[a]	7[a]
pFallBackInternal	true	true
pKeySlotOnlyEnabled	true	true
[a] *pExternOffsetCorrection* = 7 µT which is an intentional protocol constraints violation. For testability reasons *pExternOffsetCorrection* should be configurable in the Sink:CC, even a constraint in the data link layer specification requires *pExternOffsetCorrection* = 0 µT.		

Table 459 — Modification to basic configuration 3 for cycle start supervision

Parameter	Modification to Basic Configuration **3**
gExternOffsetCorrection [µs]	0,35
pClusterDriftDamping [µT]	0
pExternOffsetCorrection [µT]	7[a]
pFallBackInternal	true
pKeySlotOnlyEnabled	true
[a] *pExternOffsetCorrection* = 7 µT which is an intentional protocol constraints violation. For testability reasons *pExternOffsetCorrection* should be configurable in the Sink:CC, even a constraint in the data link layer specification requires *pExternOffsetCorrection* = 0 µT.	

— **Preamble (setup state)**

Preamble IV.

The preamble is followed by the UT, which enables interrupt requests of the IUT for the cycle start interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

The test has to be executed repeatedly with *vExternOffsetControl* set to (a) 0, (b) –1 and (c) +1.

1) In the static segment of cycle 9, the UT sets *vExternOffsetControl* to the respective value.

2) In cycle 9, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster, it is verified (UT) that the Sink:CC is externally synchronized with the Source:CC (*vExternalSync* = true).

3) In cycle 9, it is verified (UT) that the delay between the cycle start interrupts of the Source:CC and the Sink:CC is *cdTSrcCycleOffset* ± *cdCycleStartTimeout* µT.

4) In cycle 10, it is verified (UT) that the delay between the cycle start interrupts of the Source:CC and the Sink:CC is

(a) *cdTSrcCycleOffset* ± *cdCycleStartTimeout* µT,

(b) *cdTSrcCycleOffset* ± *cdCycleStartTimeout* – *pExternOffsetCorrection* µT and

(c) *cdTSrcCycleOffset* ± *cdCycleStartTimeout* + *pExternOffsetCorrection* µT.

5) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster, it is verified (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC is

(a) externally synchronized with the Source:CC (*vExternalSync* = true) or

(b, c) locally synchronized (*vExternalSync* = false).

6) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster, it is verified (LT) that the Sink:CC transmits startup frames with the null frame indicator set to '1' holding the respective payload in static slot 2 and static slot 3.

Figure 205 depicts the cycle start supervision – *vExternOffsetControl* = 1.

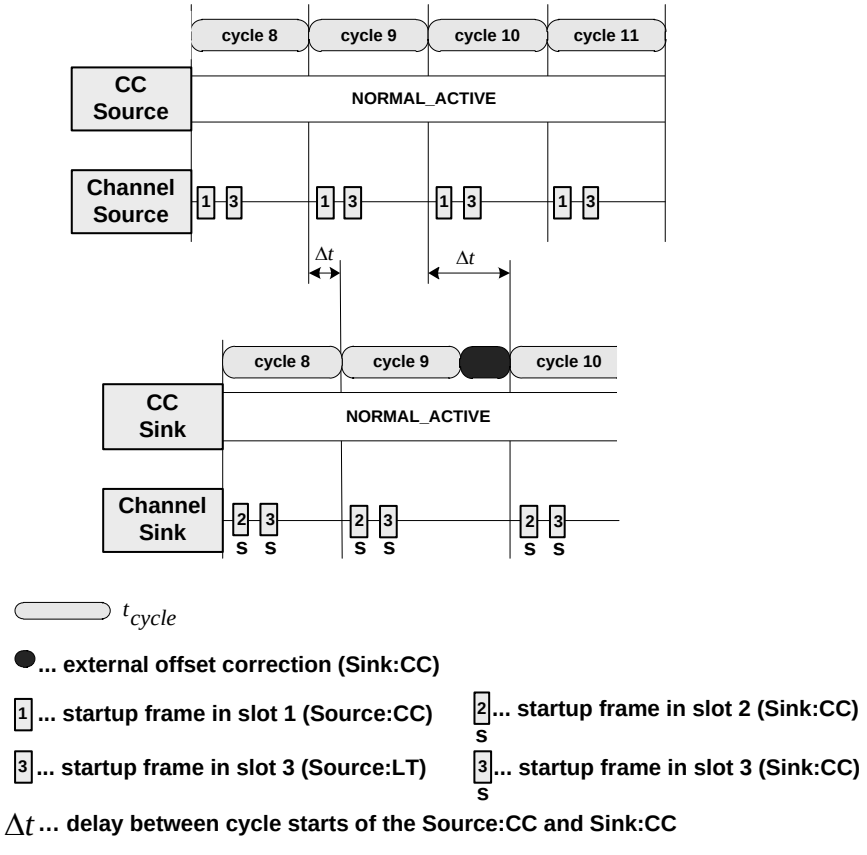


Figure 205 — Cycle start supervision – vExternOffsetControl = 1

— Postamble

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

The UT sets the Source:CC into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The Sink:CC looses synchronisation to the Source:CC, if the delay between the cycle starts of Sink:CC and Source:CC is below *cdTSrcCycleOffset* – *cdCycleStartTimeout* or above *cdTSrcCycleOffset* + *cdCycleStartTimeout*.

7.8.2.2 Offset correction inside bounds

— **Test purpose**

Verify correct behaviour of offset values at the Sink:CC when there is a deviation inside clock correction bounds at the Source:CC.

— **Applicability**

TT-E.

— **Configuration**

Source:CC: All basic configurations.

Sink:CC: All basic configurations.

The test has to be performed repeatedly with:

- Δt = (a) *–pOffsetCorrectionOut* + 1 µT,
- Δt = (b) 0 µT and
- Δt = (c) *pOffsetCorrectionOut* – 1 µT.

— **Preamble (setup state)**

Preamble IV.

— **Test execution**

1) LT starts its cycle 8 of the source cluster shifted by Δt, i.e. all following cycles are shifted.

2) In cycle 8, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC in externally synchronized (*vExternalSync* = true).

3) In cycle 9, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that the offset correction value *vInterimOffsetCorrection* of the Sink:CC is TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT.

4) In cycle 9 the LT shortens / stretches its cycle at the end by -TruncateTowardsZero(Δt / 2) µT (The LT still transmits its startup frame with the same offset from its cycle start).

5) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + *cdTSrcCycleOffset* after cycle start and *cdTSrcCycleOffset* before NIT), it is verified (UT) that the offset correction value *vInterimOffsetCorrection* of the Sink:CC is TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT.

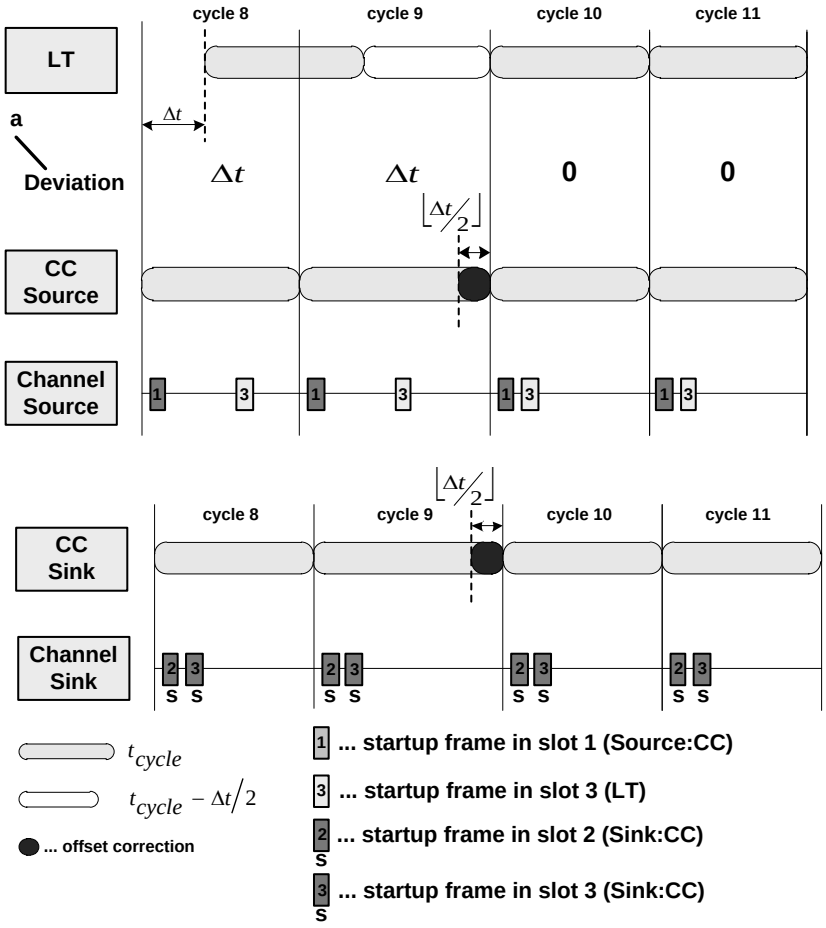
6) Starting with cycle 10 of the source cluster, the LT proceeds to generate cycles with the nominal cycle length.

7) It is verified (LT) that the interval between the Sink:CC's frames in slot2 / cycle 9 and slot 2 / cycle 10 is *pMicroPerCycle* + TruncateTowardsZero(Δt / 2) + ξ_{IUT} + ξ µT.

8) In cycle 11, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + *cdTSrcCycleOffset* after cycle start and *cdTSrcCycleOffset* before NIT), it is verified (UT) that the offset correction value *vInterimOffsetCorrection* of the Sink:CC is 0 + ξ_{IUT} µT.

9) In cycle 11, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + *cdTSrcCycleOffset* after cycle start and *cdTSrcCycleOffset* before NIT), it is verified (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC in externally synchronized (*vExternalSync* = true).

Figure 206 depicts the offset correction inside bounds; Δt greater 0.



a Expected deviation measured by the Source:CC

Figure 206 — Offset correction inside bounds; Δt greater 0

— Postamble

The UT sets the Source:CC into *POC:halt* state with the CHI command FREEZE.

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The Sink:CC stays synchronized and performs appropriate offset correction if the offset correction value is timely available.

7.8.2.3 Offset correction outside bounds

— **Test purpose**

Verify correct behaviour of the clock correction error handling. The Source:CC has a deviation outside offset correction bounds. The behaviour of the Sink:CC is defined by the *pFallBackInternal* parameter.

— **Applicability**

TT-E.

— **Configuration**

Source:CC: All basic configurations using the modifications of Table 460.

Table 460 — Modification to basic configurations for offset correction outside bounds, Source:CC

Parameter	Modification		
	I	II	III
pAllowHaltDueToClock	false		true
pOffsetCorrectionOut	80		80

Sink:CC: All basic configurations using the modifications of Table 461.

Table 461 — Modification to basic configurations for offset correction outside bounds, Sink:CC

Parameter	Modification		
	I	II	III
pAllowHaltDueToClock	true	false	true
pFallBackInternal	true	false	true

The test has to be performed repeatedly with:

— Δt = (a) –*pOffsetCorrectionOut* – 20 µT and

— Δt = (b) *pOffsetCorrectionOut* + 20 µT.

— **Preamble (setup state)**

Preamble IV.

The preamble is supplemented by starting with cycle 7. The LT additionally simulates a startup frame in slot 2 in the source cluster.

— **Test execution**

1) The LT starts its cycle 8 of the source cluster shifted by Δt. All following cycles are shifted by Δt, too.

2) In cycle 9, within 500 µT after cycle start of the source cluster and before NIT of the source cluster (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the deviation results in an equivalent offset correction value *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT in the Source:CC.

3) In cycle 9, within 500 µT after cycle start of the source cluster and before NIT of the source cluster (in terms of the LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the Source:CC is

   (I) in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*),

   (II) in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and

   (III) in the *POC:halt* state (*vPOC!State* = *HALT*).

4) In cycle 9, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster (in terms of the LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that

   (I) the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and the Sink:CC is externally synchronized (*vExternalSync* = true),

   (II) the Sink:CC is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and the Sink:CC is externally synchronized (*vExternalSync* = true) and

   (III) the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and the Sink:CC is locally synchronized (*vExternalSync* = false).

5) In cycle 9, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster (in terms of the LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that (I, II) the offset correction value *vInterimOffsetCorrection* of the Sink:CC is

   (a) –Source:*pOffsetCorrectionOut* and

   (b) +Source:*pOffsetCorrectionOut*.

6) It is verified (LT) that

   (I) the interval between the Sink:CC's frames in slot 2 / cycle 9 and slot 2 / cycle 10 is

      (a) *pMicroPerCycle* – Source:*pOffsetCorrectionOut* + ξ_{IUT} + ξ µT.

      (b) *pMicroPerCycle* + Source:*pOffsetCorrectionOut* + ξ_{IUT} + ξ µT.

   (II) in cycle 9 the Sink:CC stops frame transmission and

   (III) the interval between the Sink:CC's frames in slot 2 / cycle 9 and slot 2 / cycle 10 is *pMicroPerCycle* + ξ µT.

7) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster (in terms of the LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that

(I) the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and the Sink:CC is externally synchronized (*vExternalSync* = true),

(II) the Sink:CC is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and the Sink:CC is externally synchronized (*vExternalSync* = true) and

(III) the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and the Sink:CC is locally synchronized (*vExternalSync* = false).

8) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster (in terms of the LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that

(I, II) the offset correction value *vInterimOffsetCorrection* of the Sink:CC is

(a) –Source:*pOffsetCorrectionOut*,

(b) +Source:*pOffsetCorrectionOut*,

(III) the offset correction value *vInterimOffsetCorrection* of the Sink:CC is 0 µT.

Figure 207 depicts the offset correction outside bounds – modification I.

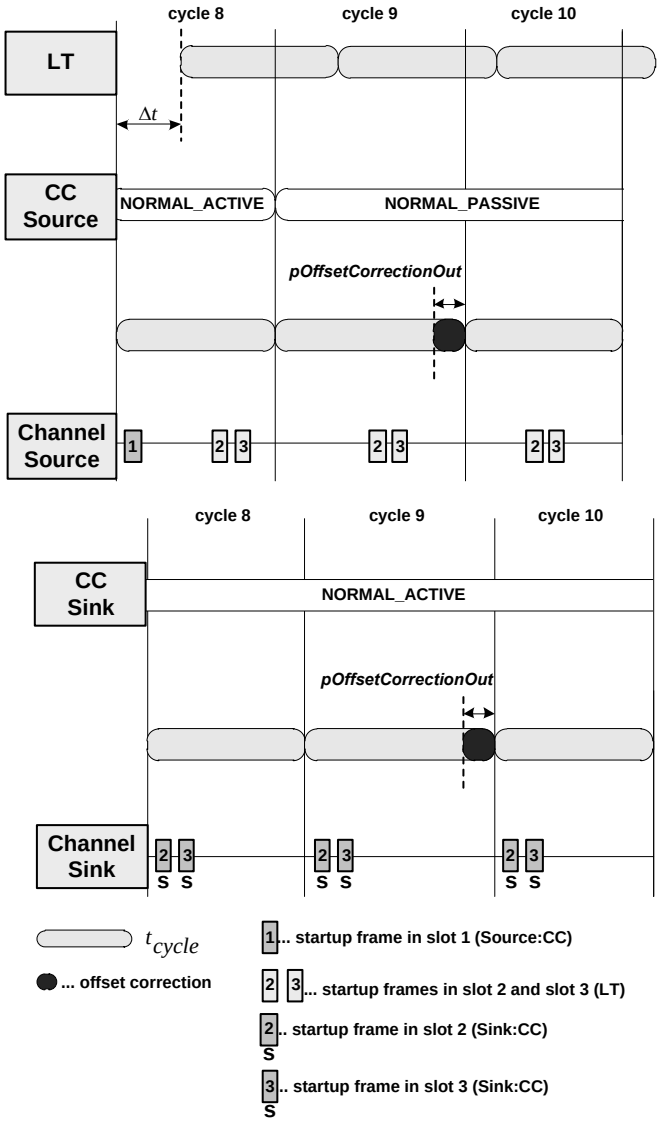


Figure 207 — Offset correction outside bounds – modification I

— **Postamble**

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

(I)(II) The UT sets the Source:CC into *POC:halt* state with the CHI command FREEZE.

(III) None. The Source:CC is already in the *POC:halt* state.

— **Pass criteria**

The Sink:CC recognizes and handles the offset correction violation in the source cluster correctly.

7.8.2.4 Rate correction inside bounds

— Test purpose

Verify correct behaviour of rate values at the Sink:CC when there is a deviation inside clock correction bounds at the Source:CC.

— Applicability

TT-E.

— Configuration

Source:CC: All basic configurations using the modifications of Table 462.

Table 462 — Modification to basic configurations for rate correction inside bounds, Source:CC

Parameter	Modification
pClusterDriftDamping [µT]	0

Sink:CC: All basic configurations using the modifications of Table 463.

Table 463 — Modification to basic configurations for rate correction inside bounds, Sink:CC

Parameter	Modification
pClusterDriftDamping [µT]	0

The test has to be performed repeatedly with:

— Δt = (a) –50 µT,

— Δt = (b) 0 µT and

— Δt = (c) +50 µT.

— Preamble (setup state)

Preamble IV.

— Test execution

1) At the beginning of cycle 9, the LT shifts its cycle by Δt and generates a cycle with a cycle length altered by –TruncateTowardsZero(Δt / 2), i.e. all following cycles are shifted. The LT simulates its frames with the regular offset to the LT's cycle start (see Figure 208).

2) Starting with cycle 10, the LT stretches / shortens (according to Δt) its cycles of the source cluster by TruncateTowardsZero(Δt / 2). The modification to the cycle length is distributed over the cycle according to the macrotick generation process as rate correction as specified in the protocol, which impacts the point in times the LT starts frame simulation.

3) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + *cdTSrcCycleOffset* after cycle start and *cdTSrcCycleOffset* before NIT), it is verified (UT) that the offset correction value *vInterimOffsetCorrection* of the Sink:CC

is TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT and the rate correction value *vInterimRateCorrection* of the Sink:CC is TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT.

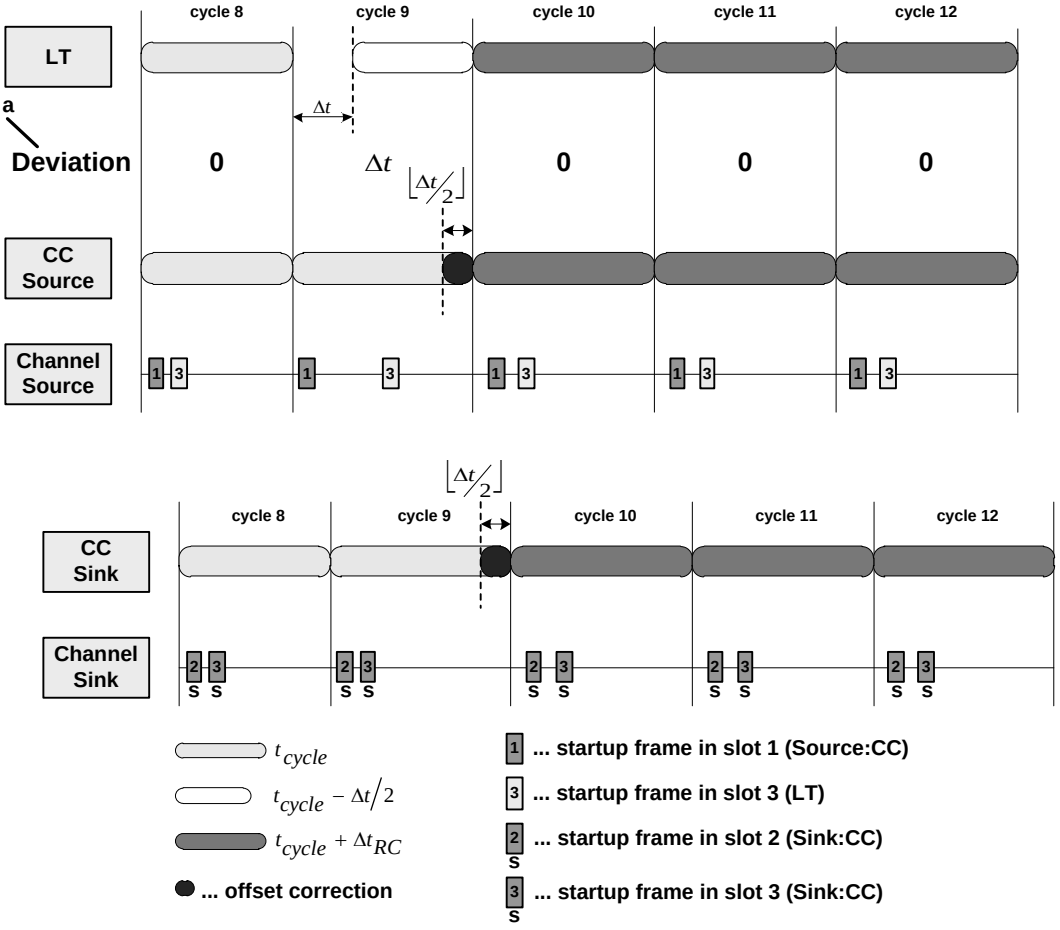
4) In cycle 11, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + *cdTSrcCycleOffset* after cycle start and *cdTSrcCycleOffset* before NIT), it is verified that the offset correction value *vInterimOffsetCorrection* of the Sink:CC is 0 + ξ_{IUT} µT.

5) It is verified (LT) that the interval between the Sink:CC's frames in slot 2 / cycle 10 and slot 2 / cycle 11 is *pMicroPerCycle* + TruncateTowardsZero(Δt / 2) + ξ_{IUT} + ξ µT.

6) In cycle 12, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + *cdTSrcCycleOffset* after cycle start and *cdTSrcCycleOffset* before NIT), it is verified (UT) that the offset correction value *vInterimOffsetCorrection* of the Sink:CC is 0 + ξ_{IUT} µT and the rate correction value *vInterimRateCorrection* of the Sink:CC is TruncateTowardsZero(Δt / 2) + ξ_{IUT} µT.

7) In cycle 12, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + *cdTSrcCycleOffset* after cycle start and *cdTSrcCycleOffset* before NIT), it is verified (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC in externally synchronized (*vExternalSync* = true).

Figure 208 depicts the rate correction inside bounds; Δt greater 0.



a Expected deviation measured by the Source:CC

Figure 208 — Rate correction inside bounds; Δt greater 0

— **Postamble**

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

The UT sets the Source:CC into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The Sink:CC stays synchronized and performs appropriate rate correction.

7.8.2.5 Rate correction outside bounds

— **Test purpose**

Verify correct behaviour of the clock correction error handling. Source:CC has a deviation outside rate correction bounds. The behaviour of the Sink:CC is defined by the *pFallBackInternal* parameter.

— **Applicability**

TT-E.

— **Configuration**

Source:CC: All basic configurations using the modifications of Table 464.

Table 464 — Modification to basic configurations for rate correction outside bounds, Source:CC

Parameter	Modification		
	I	II	III
pAllowHaltDueToClock	false		true
pAllowPassiveToActive	1		
pClusterDriftDamping [µT]	0		
pRateCorrectionOut [µT]	48[a]		
[a] *pRateCorrectionOut* = 48 [µT] which is an intentional protocol constraints violation in order to ease test execution.			

Sink:CC: All basic configurations using the modifications of Table 465.

Table 465 — Modification to basic configurations for rate correction outside bounds, Sink:CC

Parameter	Modification		
	I	II	III
pAllowHaltDueToClock	true	false	true
pAllowPassiveToActive	1		
pClusterDriftDamping [µT]	0		
pFallBackInternal	true	false	true
pKeySlotOnlyEnabled	true		

The test has to be performed repeatedly with:

— Δt = (a) –60 µT and

— Δt = (b) +60 µT.

— Preamble (setup state)

Preamble IV.

The preamble is supplemented by starting with cycle 7. The LT additionally simulates a startup frame in slot 2 in the source cluster.

— Test execution

1) At the beginning of cycle 9, the LT shifts its cycle in the source cluster by Δt. All following cycles are shifted by Δt, too.

2) In cycle 10, within 500 µT after cycle start of the source cluster and before NIT of the source cluster (in terms of LT this approximates to 500 µT + |Δt| after cycle start and |Δt| before NIT), it is verified (UT) that the offset correction value of the Source:CC is *vInterimOffsetCorrection* = Δt + ξ_{IUT} µT and the rate correction value of the Source:CC is *vInterimRateCorrection* = Δt + ξ_{IUT} µT.
It is also verified (UT) that the Source:CC is

   (I) in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*),

   (II) in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and

   (III) in the *POC:halt* state (*vPOC!State* = *HALT*).

3) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster (in terms of LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT)

   (I) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC is externally synchronized (*vExternalSync* = true),

   (II) that the Sink:CC is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*) and the Sink:CC is externally synchronized (*vExternalSync* = true) and

   (III) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and the Sink:CC is locally synchronized (*vExternalSync* = false).

4) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the sink cluster (in terms of LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that (I, II) the rate correction value *vInterimRateCorrection* of the Sink:CC is

   (a) –Source:*pRateCorrectionOut and*

   (b) +Source:*pRateCorrectionOut*.

5) It is verified (LT) that

   (I) the interval between the Sink:CC's frames in slot 2 / cycle 10 and slot 2 / cycle 11 is

      (a) *pMicroPerCycle* – Source:*pRateCorrectionOut* + ξ_{IUT} + ξ µT,

      (b) *pMicroPerCycle* + Source:*pRateCorrectionOut* + ξ_{IUT} + ξ µT,

(II) in cycle 10 of the sink cluster the Sink:CC stops frame transmission.

6) In cycle 12 of the sink cluster, within 500 µT after cycle start and before NIT (in terms of LT this approximates to 500 µT + |Δt| + *cdTSrcCycleOffset* after cycle start and |Δt| + *cdTSrcCycleOffset* before NIT), it is verified (UT) that

   (I, II) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC is externally synchronized (*vExternalSync* = true) and

   (III) the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*) and that the Sink:CC is locally synchronized (*vExternalSync* = false).

7) In cycle 12 of the sink cluster, it is verified (LT) that the Sink:CC transmits its frames in slot 2 and slot 3.

Figure 209 depicts the rate correction outside bounds – modification II.

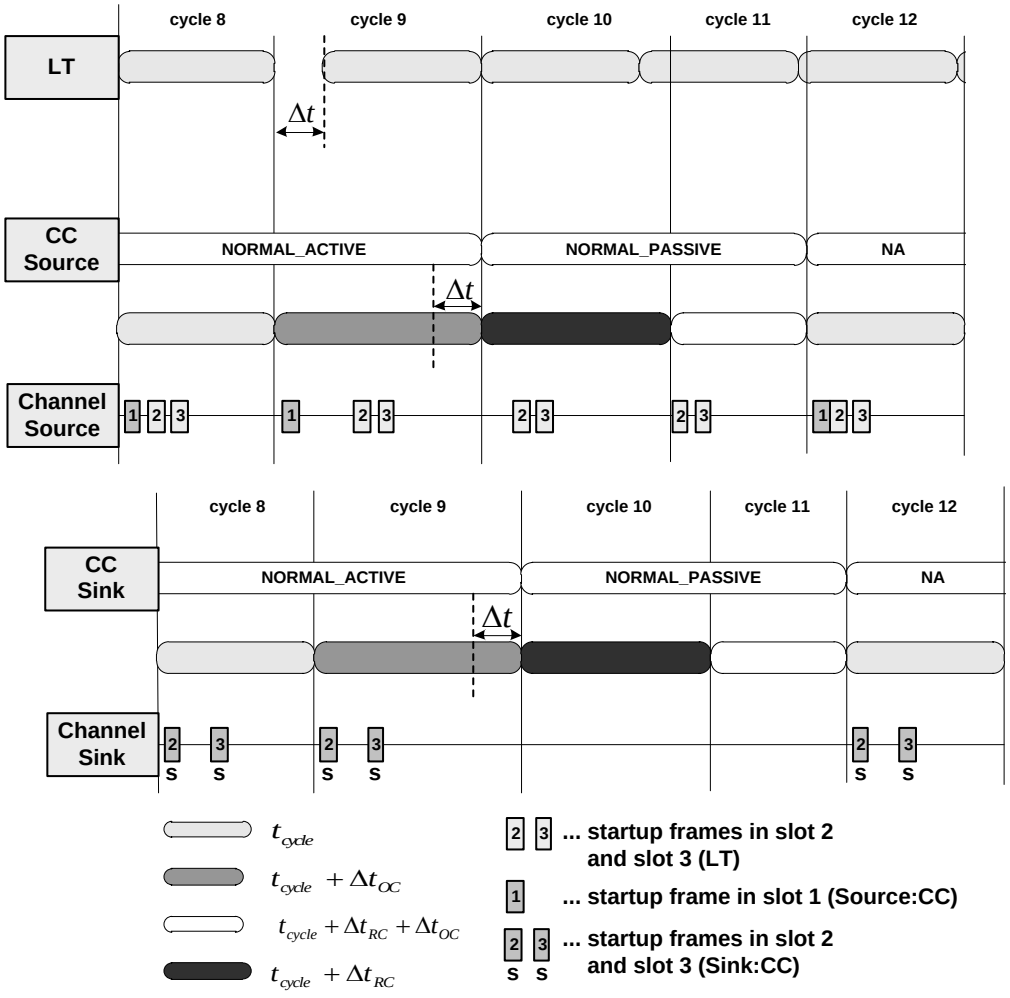


Figure 209 — Rate correction outside bounds – modification II

— **Postamble**

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

(I)(II) The UT sets the Source:CC into *POC:halt* state with the CHI command FREEZE.

(III) None. The Source:CC is already in the *POC:halt* state.

— Pass criteria

The Sink:CC recognizes and handles the rate correction violation in the source cluster correctly.

7.8.3 Startup

7.8.3.1 External startup (1)

— Test purpose

Verify correct startup behaviour of the Sink:CC. The Sink:CC enters the *POC:normal active* state in the same cycle as Source:CC.

— Applicability

TT-E.

— Configuration

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications of Table 466.

Table 466 — Modification to basic configurations for external startup (1), Sink:CC

Parameter	Modification	
	I	II
pExternalSync	true	
pFallBackInternal	false	
pKeySlotOnlyEnabled	true	false
pKeySlotID	2	
pSecondKeySlotID	3	
pTwoKeySlotMode	true	

— Preamble (setup state)

The LT emulates the leading coldstart node in the TT-D cluster (source cluster) and sends its startup frames in slot 3. The Source:CC is configured as following TT-D coldstart node. The Sink:CC is configured as TT-E coldstart node in the TT-E cluster (sink cluster).

1) The UT resets the Sink:CC.

2) The UT resets the Source:CC.

3) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

4) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

5) The UT configures the Sink:CC with the given configuration. The Sink:CC is configured to send startup frames in slot 2 and slot 3 (*pTwoKeySlotMode* = true) and one frame in slot 6.

6) The UT sets the Source:CC into *POC:config* with the CHI command CONFIG.

7) The UT configures the Source:CC with the given configuration.

8) The UT sets the Sink:CC into *POC:ready* state (*vPOC!State* = *READY*) with the CHI command CONFIG_COMPLETE.

9) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY*).

— **Test execution**

1) The UT (a) issues, (b) does not issue the command ALLOW_COLDSTART to the Sink:CC.

2) After 2 500 µT, the UT initiates the startup procedure of the Sink:CC with the CHI command RUN.

3) It is verified (UT) that the Sink:CC is in the *POC:external startup* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *EXTERNAL_STARTUP*) 2 500 µT after the CHI command RUN.

4) The UT sets the Source:CC into *POC:ready* state (*vPOC!State* = *READY*) with the CHI command CONFIG_COMPLETE.

5) The UT initiates the startup procedure of the Source:CC with the CHI command RUN.

6) The LT represents the leading coldstart node in the source cluster and simulates a CAS 0,5 * *pdListenTimeout* ± 0,1 * *pdListenTimeout* after the CHI command RUN.

7) In slot 3 of cycles 0 to 6, the LT simulates a startup frame in the source cluster with the null frame indicator, the reserved bit and the payload preamble indicator set to '0', holding a payload of 0x00 in all payload bytes.

8) In slot 3 of cycle 7, the LT simulates a startup frame in the source cluster with the null frame indicator set to '1', the reserved bit and the payload preamble indicator set to '0', holding the respective payload.

9) In cycle 7, within 500 µT after cycle start and before NIT of the source cluster, it is verified (UT) that the Source:CC is synchronized *(vPOC!State* = *NORMAL_ACTIVE*).

10) In cycle 7, within 500 µT after cycle start and before NIT of the sink cluster, it is verified (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

11) In cycle 7, it is verified (LT) that the Sink:CC transmits

    (I) startup frames with the null frame indicator set to '1' holding the respective payload in static slot 2 and static slot 3 and

    (II) startup frames with the null frame indicator set to '1' holding the respective payload in static slot 2 and static slot 3 and a frame in static slot 6 with the null frame indicator set to '1' holding the respective payload.

12) In cycle 7, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster, it is verified (UT) that the cycle counters of the Sink:CC and the Source:CC are set to 7 (Sink:*vCycleCounter* = Source:*vCycleCounter* = 7).

13) In the static segment of cycle 7, the UT issues the command DEFERRED_HALT to the Source:CC.

14) Within 500 µT and 2 500 µT after the end of cycle 7 of the sink cluster, it is verified (UT) that the Source:CC is in the *POC:halt* state *(vPOC!State = HALT*) and that the Sink:CC is in the *POC:halt* state (*vPOC!State = HALT*).

Figure 210 depicts the external startup (1) – *pKeySlotOnlyEnabled* = false.

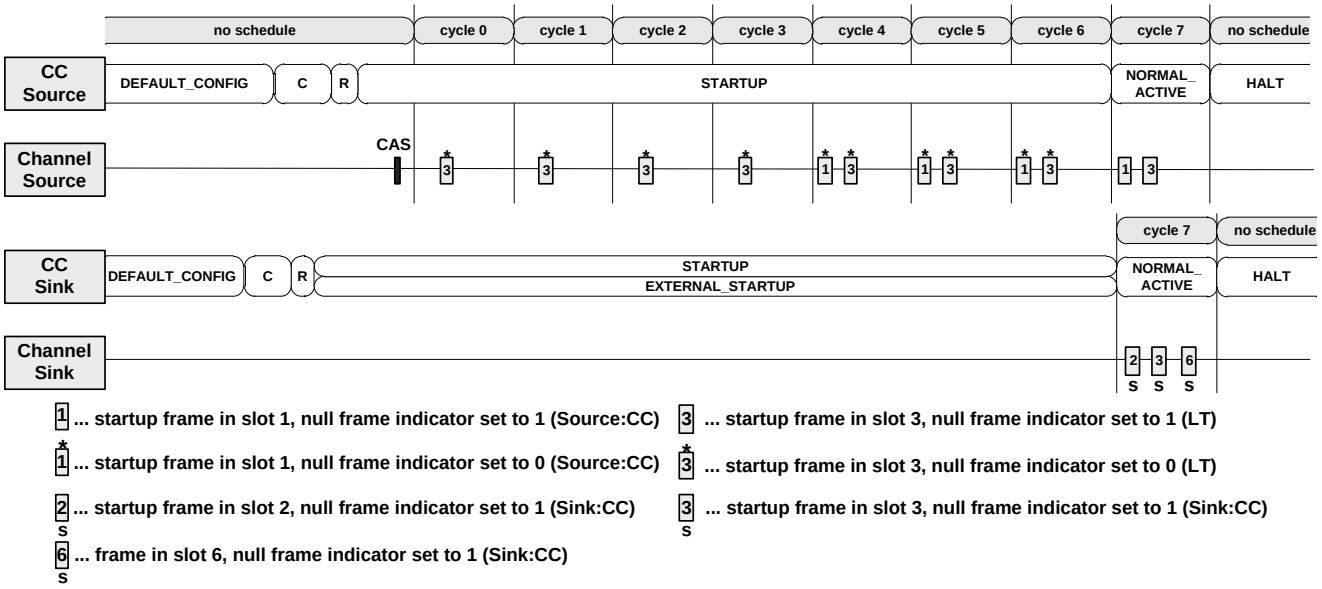


Figure 210 — External startup (1) – pKeySlotOnlyEnabled = false

— Postamble

None. Both the Source:CC and the Sink:CC are already in the *POC:halt* state.

— Pass criteria

The Sink:CC enters the *POC:normal active* state in the same cycle as the Source:CC enters the *POC:normal active* state and transmits the configured frames with the null frame indicator set to '1'.

7.8.3.2 External startup (2)

— Test purpose

Verify the correct startup behaviour of the Sink:IUT when the startup of the Sink:IUT is triggered while the Source:IUT is already in the *POC:normal active* state.

— Applicability

TT-E.

— Configuration

Source:CC: All basic configurations using the modifications of Table 467.

Table 467 — Modification to basic configurations for external startup (2), Source:CC

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1

Sink:CC: All basic configurations using the modifications of Table 468.

Table 468 — Modification to basic configurations for external startup (2), Sink:CC

Parameter	Modification
gMaxWithoutClockCorrectionPassive	1
pExternalSync	true
pKeySlotID	2
pSecondKeySlotID	3
pTwoKeySlotMode	true

— **Preamble (setup state)**

Source:CC:

Preamble I.

Sink:CC:

1) The UT resets the Sink:CC (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

3) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:config* state (*vPOC!State* = *CONFIG*).

4) The UT configures the Sink:CC with the given configuration. The Sink:CC is configured to send startup frames in slot 2 and slot 3 (*pTwoKeySlotMode* = true).

5) The UT issues the CHI command CONFIG_COMPLETE to the Sink:CC.

6) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY*).

— **Test execution**

1) In cycle (a) 8 and (b) 9 of the source cluster, within 500 µT and 1 500 µT after cycle start, the UT initiates the startup procedure of the Sink:CC with the CHI command RUN.

2) It is verified (UT) that the Sink:CC is in the *POC:external startup* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *EXTERNAL_STARTUP)* 2 500 µT after the CHI command RUN.

3) (a) In cycle 9 of the source cluster, within 500 µT after cycle start and before NIT, it is verified (UT) that the Sink:CC is in the *POC:external startup* state *(vPOC!State* = *STARTUP and vPOC!StartupState* = *EXTERNAL_STARTUP).*

4) In cycle 10 of the sink cluster, within 500 µT after cycle start and before NIT, it is verified (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State* = *NORMAL_ACTIVE*).

5) In cycle 10 of the sink cluster, it is verified (LT) that the Sink:CC transmits startup frames with the null frame indicator set to '1' holding the respective payload in static slot 2 and static slot 3.

6) In cycle 10, within 500 µT after cycle start of the sink cluster and before NIT of the source cluster, it is verified (UT) that the cycle counters of the Sink:CC and the Source:CC are set to 10 (Sink:*vCycleCounter* = Source:*vCycleCounter* = 10).

7) In cycle 11 of the source cluster, the LT stops frame simulation in the source cluster.

8) In cycle 12 of the sink cluster, within 500 µT after cycle start and before NIT, it is verified (UT) that the Sink:CC is in the *POC:normal passive* state (*vPOC!State* = *NORMAL_PASSIVE*).

9) In the static segment of cycle 12, the UT issues the command DEFERRED_HALT to the Source:CC.

10) Within 500 µT and 2 500 µT after the end of cycle 12 of the sink cluster, it is verified (UT) that the Source:CC is in the *POC:halt* state (*vPOC!State* = *HALT*) and that the Sink:CC is in the *POC:halt* state (*vPOC!State* = *HALT*).

Figure 211 depicts the external startup (2) – startup of Sink CC initiated in cycle 8.

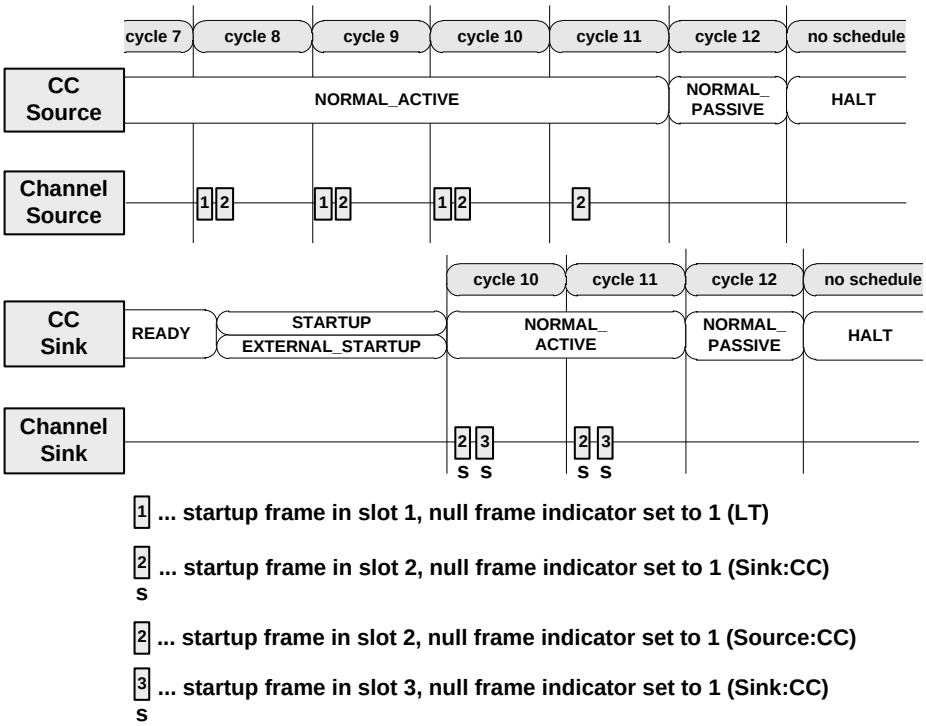


Figure 211 — External startup (2) – startup of Sink CC initiated in cycle 8

— **Postamble**

None. Both the Source:CC and the Sink:CC are already in the *POC:halt* state.

— **Pass criteria**

The Sink:CC performs proper startup.

7.8.4 Control of protocol operation control

7.8.4.1 Command IMMEDIATE_READY

— Test purpose

Verify the correct behaviour of the CHI command IMMEDIATE_READY in the *POC:external startup* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:external startup* to *POC:ready*, initiated by the CHI command IMMEDIATE_READY).

— Applicability

TT-E.

— Configuration

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications of Table 469.

Table 469 — Modification to basic configurations for control of protocol operation control – command IMMEDIATE_READY, Sink:CC

Parameter	Modification
pExternalSync	true
pSecondKeySlotID	2
pTwoKeySlotMode	true

— Preamble (setup state)

1) The UT resets the Sink:CC.

2) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

3) The UT configures the Sink:CC with the given configuration.

4) The UT sets the Sink:CC into *POC:ready* state with the CHI command CONFIG_COMPLETE.

5) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY*).

6) The UT enables interrupt requests of the Sink:CC for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— Test execution

1) The UT initiates the startup procedure of the Sink:CC with the CHI command RUN.

2) It is verified (UT) that the Sink:CC is in the *POC:external startup* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *EXTERNAL_STARTUP*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the Sink:CC for all interrupt sources.

4) 2 500 µT later the UT issues the CHI command IMMEDIATE_READY to the Sink:CC.

5) 2 500 µT after the CHI command IMMEDIATE_READY it is verified (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY)* and that the startup state of the Sink:CC is undefined *(vPOC!StartupState* = *UNDEFINED*).

6) It is verified (UT) that the Sink:CC did not generate any interrupt request.

— Postamble

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

— Pass criteria

The Sink:CC performs the expected state transitions accordingly. The Sink:CC does not indicate an interrupt request on the POC state transition to *POC:ready* state.

7.8.4.2 Command FREEZE

— Test purpose

Verify the correct behaviour of the CHI command FREEZE in the *POC:external startup* state. Verify that no interrupt request is indicated for the POC state transition interrupt source (transition from *POC:external startup* to *POC:halt*, initiated by the CHI command FREEZE).

— Applicability

TT-E.

— Configuration

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications of Table 470.

Table 470 — Modification to basic configurations for control of protocol operation control – command FREEZE, Sink:CC

Parameter	Modification
pExternalSync	true
pSecondKeySlotID	2
pTwoKeySlotMode	true

— Preamble (setup state)

1) The UT resets the Sink:CC.

2) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

3) The UT configures the Sink:CC with the given configuration.

4) The UT sets the Sink:CC into *POC:ready* state with the CHI command CONFIG_COMPLETE.

5) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY*).

6) The UT enables interrupt requests of the Sink:CC for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT initiates the startup procedure of the Sink:CC with the CHI command RUN.

2) It is verified (UT) that the Sink:CC is in the *POC:external startup* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *EXTERNAL_STARTUP*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the Sink:CC for all interrupt sources.

4) 2 500 µT later the UT issues the CHI command FREEZE.

5) 2 500 µT after the CHI command FREEZE it is verified (UT) that the Sink:CC has entered the *POC:halt* state (*vPOC!State* = *STARTUP*, *vPOC!StartupState* = *EXTERNAL_STARTUP)* and *vPOC!Freeze* = true.

6) It is verified (UT) that the Sink:CC did not generate any interrupt request.

— **Postamble**

None. The Sink:CC is already in *POC:halt* state.

— **Pass criteria**

The Sink:CC performs the expected state transitions and *vPOC!Freeze* is set accordingly. The Sink:CC does not indicate an interrupt request on the POC state transition to *POC:halt* state.

7.8.4.3 Command DEFERRED_READY

— **Test purpose**

Verify the correct behaviour of the CHI command DEFERRED_READY in the *POC:external startup* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:external startup* to *POC:ready*, initiated by the CHI command DEFERRED_READY).

— **Applicability**

TT-E.

— **Configuration**

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications of Table 471.

Table 471 — Modification to basic configurations for control of protocol operation control – command DEFERRED_READY, Sink:CC

Parameter	Modification
pExternalSync	true
pSecondKeySlotID	2
pTwoKeySlotMode	true

— **Preamble (setup state)**

1) The UT resets the Sink:CC.

2) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

3) The UT configures the Sink:CC with the given configuration.

4) The UT sets the Sink:CC into *POC:ready* state with the CHI command CONFIG_COMPLETE.

5) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY*).

6) The UT enables interrupt requests of the Sink:CC for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT initiates the startup procedure of the Sink:CC with the CHI command RUN.

2) It is verified (UT) that the Sink:CC is in the *POC:external startup* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *EXTERNAL_STARTUP*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the Sink:CC for all interrupt sources.

4) 2 500 µT later the UT issues the CHI command DEFERRED_READY.

5) 2 500 µT after the CHI command DEFERRED_READY it is verified (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY*), that the startup state is undefined (*vPOC!StartupState* = *UNDEFINED*) and that *vPOC!CHIReadyRequest* = false.

6) It is verified (UT) that the Sink:CC has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

The UT sets the Sink:CC into *POC:halt* state with the CHI command FREEZE.

— **Pass criteria**

The Sink:CC performs the state transition to *POC:ready* accordingly and the *vPOC!CHIReadyRequest* is set accordingly. The Sink:CC indicates an interrupt request on the POC state transition to *POC:ready* state.

7.8.4.4 Command DEFERRED_HALT

— **Test purpose**

Verify the correct behaviour of the CHI command DEFERRED_HALT in the *POC:external startup* state. Verify that an interrupt request is indicated for the POC state transition interrupt source (transition from *POC:external startup* to *POC:halt*, initiated by the CHI command DEFERRED_HALT).

— **Applicability**

TT-E.

— **Configuration**

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications of Table 472.

Table 472 — Modification to basic configurations for control of protocol operation control – command DEFERRED_HALT, Sink:CC

Parameter	Modification
pExternalSync	true
pSecondKeySlotID	2
pTwoKeySlotMode	true

— **Preamble (setup state)**

1) The UT resets the Sink:CC.

2) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

3) The UT configures the Sink:CC with the given configuration.

4) The UT sets the Sink:CC into *POC:ready* state with the CHI command CONFIG_COMPLETE.

5) After 2 500 µT it is checked (UT) that the Sink:CC has entered the *POC:ready* state (*vPOC!State* = *READY*).

6) The UT enables interrupt requests of the Sink:CC for the POC state transition interrupt source. All other interrupt sources are disabled for generating interrupt requests.

— **Test execution**

1) The UT initiates the startup procedure of the Sink:CC with the CHI command RUN.

2) It is verified (UT) that the Sink:CC is in the *POC:external startup* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *EXTERNAL_STARTUP*) 2 500 µT after the CHI command RUN.

3) The UT clears (resets) the interrupt status indication flags of the Sink:CC for all interrupt sources.

4) 2 500 µT later the UT issues the CHI command DEFERRED_HALT.

5) 2 500 µT after the CHI command DEFERRED_HALT it is verified (UT) that the Sink:CC has entered the *POC:halt* state (*vPOC!State* = *HALT*) and that *vPOC!CHIHaltRequest* = false.

6) It is verified (UT) that the Sink:CC has generated an interrupt request only for the interrupt source POC state transition.

— **Postamble**

None. The Sink:CC is already in *POC:halt* state.

— **Pass criteria**

The Sink:CC performs the state transition to *POC:halt* accordingly and the *vPOC!CHIHaltRequest* is set accordingly. The Sink:CC indicates an interrupt request on the POC state transition to *POC:halt* state.

7.9 Preambles

7.9.1 Preamble I

— Description

The IUT is configured as coldstart node sending startup frames in slot 2. The IUT assumes the role of the following coldstart node, the LT the role of the leading coldstart node simulating startup frames in slot 1.

— Configuration

All basic configurations using the modifications of Table 473.

Table 473 — Modification to basic configurations for preambles – preamble I

Parameter	Modification
pKeySlotID	2
pKeySlotUsedForStartup	true
pKeySlotUsedForSync	true

Table 474 defines the preamble I applied default frame contents.

Table 474 — Preamble I applied default frame contents

Default frame contents		
Frame ID	1	2
Reserved bit	0	
Payload preamble indicator	0	
Null frame indicator	1	
Sync frame indicator	1	
Startup frame indicator	1	
Payload length	*gPayloadLengthStatic*	
Payload data on channel A	byte 0: 0xAA, byte 1: 0x55, all other bytes set to 0x00	
Payload data on channel B	byte 0: 0xCC, byte 1: 0x33, all other bytes set to 0x00	

— Execution

1) The UT resets the IUT.

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG.

3) The UT configures the IUT with the given configuration.

4) The UT configures the IUT to transmit a startup frame in slot 2 according to the respective default frame contents.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN.

7) The LT represents the leading coldstart node and simulates a CAS 0,5 * *pdListenTimeout* ± 0,1 * *pdListenTimeout* after the CHI command RUN.

8) In slot 1 of cycles 0 to 6, the LT simulates a startup frame with the null frame indicator, the reserved bit and the payload preamble indicator set to 0, holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes.

9) In cycles 4 to 6, it is checked (LT) that the IUT transmits a startup frame with the null frame indicator set to 0 holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes in static slot 2.

10) In slot 1 of cycle 7, the LT simulates a startup frame with the respective default frame contents.

11) In cycle 7, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is synchronized (*vPOC!State = NORMAL_ACTIVE*).

12) In cycle 7, it is checked (LT) that the IUT transmits a startup frame with the null frame indicator set to 1 holding the respective payload in static slot 2.

Figure 212 depicts the preamble I.

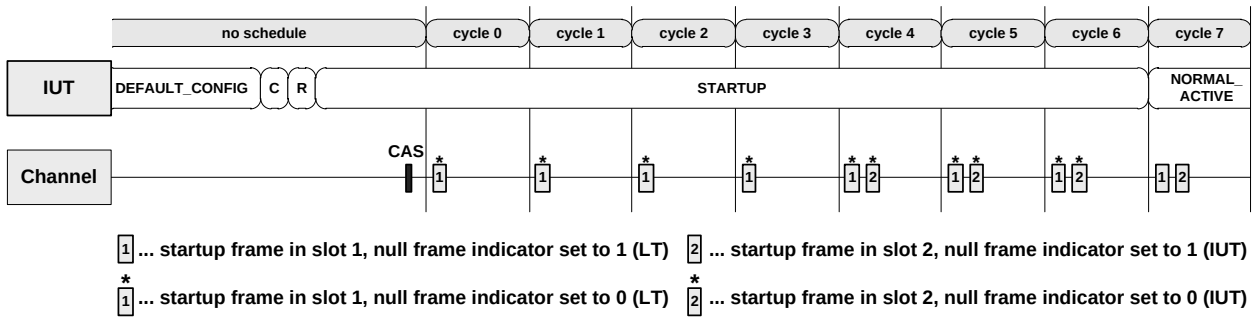


Figure 212 — Preamble I

7.9.2 Preamble II

— Description

The IUT is configured as coldstart node sending startup frames in slot 1. The IUT assumes the role of the leading coldstart node, the LT the role of the following coldstart node simulating startup frames in slot 2.

— Configuration

All basic configurations using the modifications of Table 475.

Table 475 — Modification to basic configurations for preambles – preamble II

Parameter	Modification
pKeySlotID	1
pKeySlotUsedForStartup	true
pKeySlotUsedForSync	true

Table 476 defines the preamble II applied default frame contents.

Table 476 — Preamble II applied default frame contents

Default frame contents		
Frame ID	1	2
Reserved bit	0	
Payload preamble indicator	0	
Null frame indicator	1	
Sync frame indicator	1	
Startup frame indicator	1	
Payload length	*gPayloadLengthStatic*	
Payload data on channel A	byte 0: 0xAA, byte 1: 0x55, all other bytes set to 0x00	
Payload data on channel B	byte 0: 0xCC, byte 1: 0x33, all other bytes set to 0x00	

— **Execution**

1) The UT resets the IUT.

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG.

3) The UT configures the IUT with the given configuration.

4) The UT configures the IUT to transmit a startup frame in slot 1 holding the respective default payload. The payload preamble indicator is set to 0.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE.

6) The UT clears the *vColdstartInhibit* flag with the CHI command ALLOW_COLDSTART.

7) The UT initiates the startup procedure with the CHI command RUN.

8) The IUT is the leading coldstart node. It is checked (LT) that the IUT transmits a CAS.

9) In cycles 0 to 5, it is checked (LT) that the IUT transmits a startup frame with the null frame indicator set to 0 holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes in slot 1.

10) The LT represents the following coldstart node. In slot 2 of cycles 4 to 6, the LT simulates a startup frame with the null frame indicator, the reserved bit and the payload preamble indicator set to 0, holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes.

11) In cycle 6, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

12) In cycle 6, it is checked (LT) that the IUT transmits a startup frame with the null frame indicator set to 1 holding the respective payload in slot 1.

Figure 213 depicts the preamble II.

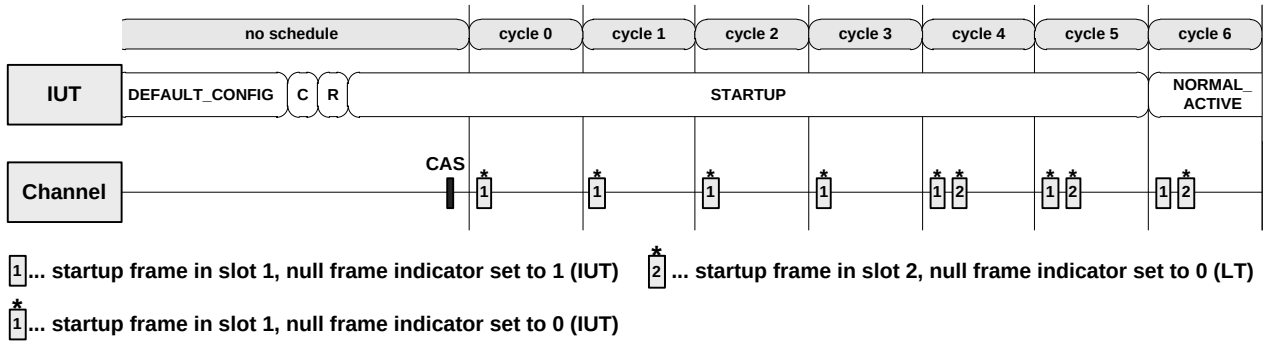


Figure 213 — Preamble II

7.9.3 Preamble III

— Description

The IUT is configured as integrating node sending static frames in slot 1. The LT assumes the role of the leading coldstart node simulating startup frames in slot 2 and the following coldstart node simulating startup frames in slot 16.

— Configuration

All basic configurations using the modifications of Table 477.

Table 477 — Modification to basic configurations for preambles – preamble III

Parameter	Modification
pKeySlotID	0
pKeySlotUsedForStartup	false
pKeySlotUsedForSync	false

Table 478 defines the preamble III applied default frame contents.

Table 478 — Preamble III applied default frame contents

Default frame contents			
Frame ID	2	16	1, 3..15
Reserved bit	0		
Payload preamble indicator	0		
Null frame indicator	1		
Sync frame indicator	1		0
Startup frame indicator	1		0
Payload length	*gPayloadLengthStatic*		
Payload data on channel A	byte 0: 0xAA, byte 1: 0x55, all other bytes set to 0x00		
Payload data on channel B	byte 0: 0xCC, byte 1: 0x33, all other bytes set to 0x00		

— **Execution**

1) The UT resets the IUT.

2) The UT sets the IUT into *POC:config* state with the CHI command CONFIG.

3) The UT configures the IUT with the given configuration.

4) The UT configures the IUT to transmit a static frame (no sync frame) in slot 1.

5) The UT sets the IUT into *POC:ready* state with the CHI command CONFIG_COMPLETE.

6) The UT initiates the startup procedure with the CHI command RUN.

7) The LT represents two coldstart nodes and simulates a CAS 0,5 * *pdListenTimeout* ± 0,1 * *pdListenTimeout* after the CHI command RUN.

8) In slot 2 of cycles 0 to 7, the LT simulates a startup frame with the null frame indicator, the reserved bit and the payload preamble indicator set to 0, holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes.

9) In slot 16 of cycles 4 to 7, the LT simulates a startup frame with the null frame indicator, the reserved bit and the payload preamble indicator set to 0, holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes.

10) In slots 2 and 16 of cycle 8, the LT simulates startup frames with the null frame indicator set to 1, the reserved bit and the payload preamble indicator set to 0, holding the respective default payload.

11) In cycle 8, 500 µT after cycle start and before cycle end, it is checked (UT) that the IUT is synchronized (*vPOC!State* = *NORMAL_ACTIVE*).

12) In cycle 8, it is checked (LT) that the IUT transmits a static frame with the null frame indicator set to 1 holding the respective payload in slot 1.

Figure 214 depicts the preamble III.

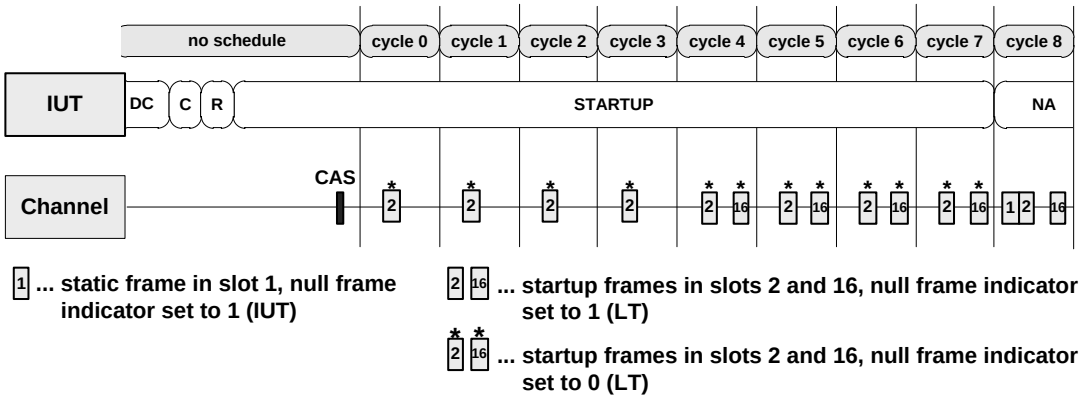


Figure 214 — Preamble III

7.9.4 Preamble IV

— Description

The TT-E time gateway source (Source:CC) is configured as TT-D coldstart node sending startup frames in slot 1. The Source:CC takes the role of the following coldstart node, the LT takes the role of the leading coldstart node in the TT-D cluster (source cluster) simulating startup frames in slot 3. The TT-E time gateway sink (Sink:CC) is configured as TT-E coldstart node in the TT-E cluster (sink cluster) and sending startup frames in slot 2 and 3.

— Configuration

Source:CC: All basic configurations.

Sink:CC: All basic configurations using the modifications of Table 479.

Table 479 — Modification to basic configurations for preambles – preamble IV, Sink:CC

Parameter	Modification
pExternalSync	true
pKeySlotID	2
pKeySlotUsedForStartup	true
pKeySlotUsedForSync	true
pSecondKeySlotID	3
pTwoKeySlotMode	true

Both clusters, the source cluster and the sink cluster use the same basic configuration (unless otherwise specified in the test case).

Table 480 defines the default frame contents applied for Preamble IV for both Source:CC and Sink:CC.

Table 480 — Default frame contents applied for Preamble IV for both Source:CC and Sink:CC

Default frame contents		
Frame ID (Source:CC)	1	3
Frame ID (Sink:CC)	2	3
Reserved bit	0	
Payload preamble indicator	0	
Null frame indicator	1	
Sync frame indicator	1	
Startup frame indicator	1	
Payload length	*gPayloadLengthStatic*	
Payload data on channel A	byte 0: 0xAA, byte 1: 0x55, all other bytes set to 0x00	
Payload data on channel B	byte 0: 0xCC, byte 1: 0x33, all other bytes set to 0x00	

— **Execution**

1) The UT resets the Source:CC (*vPOC!State* = *DEFAULT_CONFIG*).

2) The UT resets the Sink:CC (*vPOC!State* = *DEFAULT_CONFIG*).

3) The UT sets the Sink:CC into *POC:config* with the CHI command CONFIG.

4) The UT configures the Sink:CC with the given configuration. The Sink:CC is configured to send startup frames in slot 2 and slot 3 (*pTwoKeySlotMode* = true).

5) The UT sets the Source:CC into *POC:config* with the CHI command CONFIG.

6) The UT configures the Source:CC with the given configuration.

7) The UT sets the Sink:CC into *POC:ready* state with the CHI command CONFIG_COMPLETE.

8) The UT sets the Source:CC into *POC:ready* state with the CHI command CONFIG_COMPLETE.

9) The UT initiates the startup procedure of the Sink:CC with the CHI command RUN.

10) It is checked (UT) that the Sink:CC is in the *POC:external startup* state (*vPOC!State* = *STARTUP* and *vPOC!StartupState* = *EXTERNAL_STARTUP*) 2 500 µT after the CHI command RUN.

11) The UT initiates the startup procedure of the Source:CC with the CHI command RUN.

12) The LT represents the leading coldstart node in the source cluster and simulates a CAS 0,5 * *pdListenTimeout* ± 0,1 * *pdListenTimeout* after the CHI command RUN.

13) In slot 3 of cycles 0 to 6, the LT simulates a startup frame in the source cluster with the null frame indicator, the reserved bit and the payload preamble indicator set to 0, holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes.

14) In cycles 4 to 6 of the source cluster, it is checked (LT) that the Source:CC transmits a startup frame with the null frame indicator set to 0 holding no payload if the payload length is zero or a payload of 0x00 in all payload bytes in static slot 1.

15) In slot 3 of cycle 7, the LT simulates a startup frame in source cluster with the null frame indicator set to 1, the reserved bit and the payload preamble indicator set to 0, holding the respective payload.

16) In cycle 7 of the source cluster, within 500 µT after cycle start and before NIT, it is checked (UT) that the Source:CC is synchronized *(vPOC!State = NORMAL_ACTIVE*).

17) In cycle 7, it is checked (LT) that the Source:CC transmits a startup frame with the null frame indicator set to 1 holding the respective payload in static slot 1.

18) In cycle 7, within 500 µT after cycle start of sink cluster and before NIT, it is checked (UT) that the Sink:CC is in the *POC:normal active* state (*vPOC!State = NORMAL_ACTIVE*).

19) In cycle 7, it is checked (LT) that the Sink:CC transmits a startup frame with the null frame indicator set to 1 holding the respective payload in static slot 2 and static slot 3.

Figure 215 depicts the preamble IV.

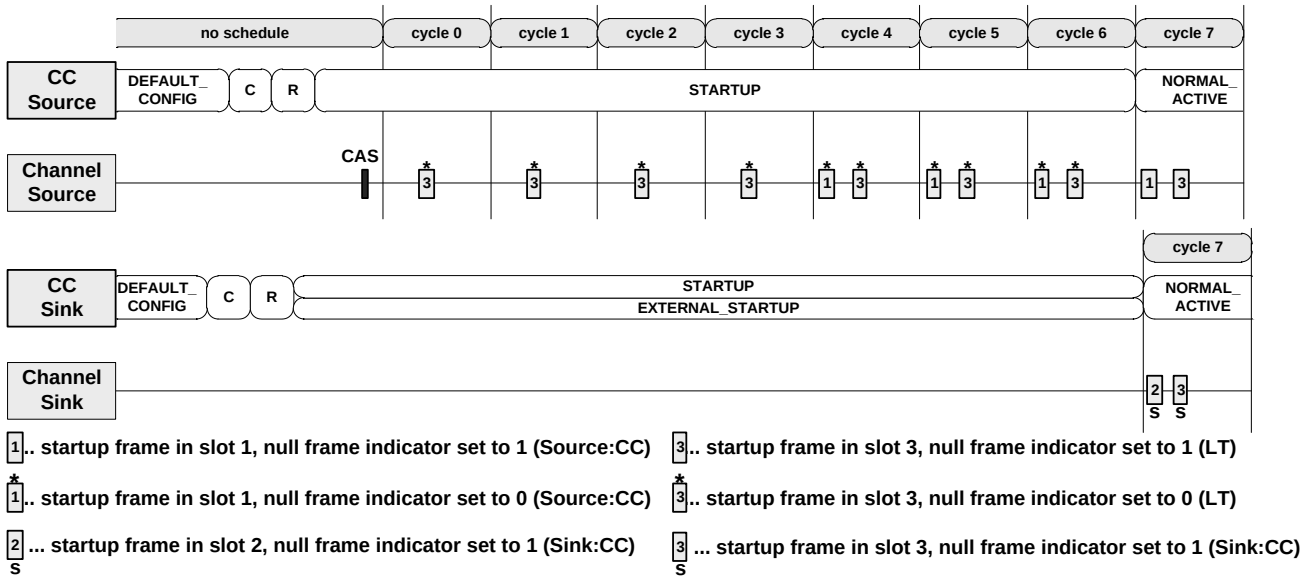


Figure 215 — Preamble IV

8 Configuration

8.1 Basic configuration

The basic configuration is the default configuration. Deviating parameter values are explicitly stated in the test cases and preambles.

Table 481 defines the protocol relevant global cluster parameters.

Table 481 — Protocol relevant global cluster parameters

Parameter	Basic Configuration				
	1a	**1b**	**2a**	**2b**	**3**
gColdstartAttempts	8				
gCycleCountMax	63				
gdActionPointOffset [MT]	4		3		
gdCASRxLowMax [gdBit]	97		80		72
gdDynamicSlotIdlePhase [Minislot]	1				
gdIgnoreAfterTx [gdBit]	5		3		2
gdMinislot [MT]	9		7		
gdMinislotActionPointOffset [MT]	4		3		
gdStaticSlot [MT]	35		33		58
gdSymbolWindow [MT]	40		23		24
gdSymbolWindowActionPointOffset [MT]	4		3		
gdTSSTransmitter [gdBit]	11		6		4
gdWakeupRxIdle [gdBit]	40		20		10
gdWakeupRxLow [gdBit]	40		20		10
gdWakeupRxWindow [gdBit]	480		240		120
gdWakeupTxIdle [gdBit]	180		90		45
gdWakeupTxActive [gdBit]	60		30		15
gListenNoise	2				
gMacroPerCycle [MT]	5 000		2 500		
gMaxWithoutClockCorrectionFatal	2				
gMaxWithoutClockCorrectionPassive	2				
gNumberOfMinislots	219		140		145
gNumberOfStaticSlots	85		45		25
gPayloadLengthStatic [two-byte word]	8				
gSyncFrameIDCountMax	15				

Table 482 defines the protocol related global cluster parameters.

Table 482 — Protocol related global cluster parameters

Parameter	Basic Configuration				
	1a	**1b**	**2a**	**2b**	**3**
gChannels	value of *pChannels*				
gClockDeviationMax	0,0003				
gClusterDriftDamping [µT]	2				
gdBit [µs]	0,1		0,2		0,4
gdCycle [µs]	5 000				
gdMacrotick [µs]	1		2		
gdNIT [MT]	14		12		11
gdSampleClockPeriod [µs]	0,0125		0,025		0,050
gExternOffsetCorrection [µs]	0				
gExternRateCorrection [µs]	0				
gNetworkManagementVectorLength [byte]	2				

Table 483 defines the protocol relevant Node parameters.

Table 483 — Protocol relevant Node parameters

Parameter	Basic Configuration				
	1a	**1b**	**2a**	**2b**	**3**
pAllowHaltDueToClock	true				
pAllowPassiveToActive	0				
pChannels	See 7.1.1 for single channel test execution: A or B for dual channel test execution: A&B				
pdAcceptedStartupRange [µT]	281	403	283	221	224
pClusterDriftDamping [µT]	2				
pDecodingCorrection [µT]	52	104	64	32	48
pDelayCompensation[A] [µT]	5	10	10	5	10
pDelayCompensation[B] [µT]	5	10	10	5	10
pdListenTimeout [µT]	400 242	800 482	400 242	200 122	200 122
pExternOffsetCorrection [µT]	0				
pExternRateCorrection [µT]	0				
pExternalSync	false				
pFallBackInternal	false				
pKeySlotID	1				
pKeySlotUsedForStartup	true				
pKeySlotUsedForSync	true				
pLatestTx [Minislot]	188		101		68
pMacroInitialOffset[A] [MT]	6		4		5
pMacroInitialOffset[B] [MT]	6		4		5
pMicroInitialOffset[A] [µT]	23	46	6	3	22
pMicroInitialOffset[B] [µT]	23	46	6	3	22
pMicroPerCycle [µT]	200 000	400 000	200 000	100 000	100 000
pOffsetCorrectionOut [µT]	153	226	154	117	118
pOffsetCorrectionStart [MT]	4 990		2 491		2 492
pRateCorrectionOut [µT]	121	241	121	61	61
pKeySlotOnlyEnabled	false				
pSecondKeySlotID	0				
pTwoKeySlotMode	false				
pWakeupChannel	For single channel test execution: value of *pChannels* For dual channel test execution and unless otherwise specified: A				
pWakeupPattern	14				

Table 484 defines the protocol related Node parameters.

Table 484 — Protocol related Node parameters

Parameter	Basic Configuration				
	1a	**1b**	**2a**	**2b**	**3**
pdMicrotick [µs]	0,025	0,0125	0,025	0,05	0,05
pNMVectorEarlyUpdate	true				
pPayloadLengthDynMax [two-byte word]	127				
pSamplesPerMicrotick	2	1	1	2	1

Table 485 defines the Physical Layer relevant parameters.

Table 485 — Physical Layer relevant parameters

Parameter	Basic Configuration				
	1a	**1b**	**2a**	**2b**	**3**
dFrameTSSLengthChangeM,N [ns]	-850				
dFrameTSSEMIInfluenceM,N [ns]	-50				
dSymbolLengthChangeM,N [ns]	675				
dSymbolEMIInfluenceM,N [ns]	400				
dStarDelay01k [ns]	150				
dStarDelay10k [ns]	150				
nStarPathM,N	1				
dBDRxia [ns]	350				
dBDRxai [ns]	275				
dBDTxActiveMax [µs]	650				
dBDTxDM [ns]	50				
dBDTxia [ns]	25				
dBDTxai [ns]	75				
dBDRx01 [ns]	0				
dBDRx10 [ns]	75				
dBDTx01 [ns]	0				
dBDTx10 [ns]	75				
T0 [ns / m]	10				
line_length [m]	24				

Table 486 defines the auxiliary parameters.

Table 486 — Auxiliary parameters

Parameter	Basic Configuration				
	1a	**1b**	**2a**	**2b**	**3**
adPropagationDelayMax [µs]	0,735		0,873		1,148
aWUDOP	false				

The values for parameters *line_length* and *T0* have been introduced only as a calculation basis for depending parameters. A *line_length* of 24 m is not a requirement for the test system. Nevertheless, not to invalidate the above parameter values and test results, a maximum propagation delay of 0,01 µs shall not be exceeded.

All basic configurations are calculated with *adInternalRxDelay* = 4 [*samples*].

8.2 Standard modifications

8.2.1 Standard modification set 1

— Description

Modifies the number of static slots (*gNumberOfStaticSlots*).

— Modifications

Table 487 defines the modification to basic configurations for standard modification set 1.

Table 487 — Modification to basic configurations for standard modification set 1

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
gdActionPointOffset [MT]	1				
gdStaticSlot [MT]	15				26
gdSymbolWindow [MT]	0				
gMacroPerCycle [MT]	15 500		8 000		
gNumberOfMinislots	0				
gNumberOfStaticSlots	1 023		530		305
gPayloadLengthStatic [two-byte word]	1				
gdCycle [µs]	15 500		16 000		
gdNIT [MT]	155		50		70
pClusterDriftDamping [µT]	0				
pdListenTimeout [µT]	1 240 746	2 481 490	1 280 770	640 386	
pKeySlotID	3				
pLatestTx [Minislot]	0				
pMacroInitialOffset[A,B] [MT]	3		2		3
pMicroPerCycle [µT]	620 000	1 240 000	640 000	320 000	
pOffsetCorrectionOut [µT]	45				
pOffsetCorrectionStart [MT]	15 490		7 995		
pRateCorrectionOut [µT]	373	745	385	193	

8.2.2 Standard modification set 2

— Description

Modifies the number of minislots (*gNumberOfMinislots*).

— Modifications

Table 488 defines the modification to basic configurations for standard modification set 2.

Table 488 — Modification to basic configurations for standard modification set 2

Parameter	Modification to Basic Configurations				
	1a	**1b**	**2a**	**2b**	**3**
gdActionPointOffset [MT]	1				
gdDynamicSlotIdlePhase [Minislot]	1				2
gdMinislot [MT]	2[a]				
gdMinislotActionPointOffset [MT]	1				
gdStaticSlot [MT]	13				22
gdSymbolWindow [MT]	0				
gMacroPerCycle [MT]	10 050		8 000		
gNumberOfMinislots	5 000		3 985		3 976
gNumberOfStaticSlots	2				
gPayloadLengthStatic [two-byte word]	0				
gdCycle [µs]	10 050		16 000		
gdNIT [MT]	24		4		
pClusterDriftDamping [µT]	0				
pdListenTimeout [µT]	804 484	1 608 966	1 280 770	640 386	
pLatestTx [Minislot]	4 867	4 867	3 852	3 852	3 711
pMacroInitialOffset[A,B] [MT]	3		2		3
pMicroPerCycle [µT]	402 000	804 000	640 000	320 000	
pOffsetCorrectionOut [µT]	45				
pOffsetCorrectionStart [MT]	10 040		7 997		
pRateCorrectionOut [µT]	242	483	385	193	

[a] intentional protocol constraints violation.

9 Static test cases

9.1 Electrical interface

— Test purpose

ISO 17458-2 describes the electrical interface of a FlexRay Communication Controller by a set of parameters and allowed ranges for each parameter. The purpose of the static test case for the electrical interface is to show that the electrical interface of the IUT is according to ISO 17458-4.

— Description

To be able to claim the static test cases as passed in the sense of this part of ISO 17458, one of the following two alternatives shall be fulfilled for the IUT:

- Alternative 1:
  All gray areas of the form *"Technical data of the electrical interface of a FlexRay Communication Controller V3.0"* shall be filled in, the filled template shall be handed out, and the handed out template shall have the same binding level as the datasheet.

- Alternative 2:
  All relevant parts of the form *"Technical data of the electrical iInterface of a FlexRay Communication Controller V3.0"* (i.e., Table A.2 footnote [a)], and Statements 1 .. 3) shall be included in the datasheet, and all gray areas shall be filled in. Additional references or explanations in the Table A.2 are possible.

See Annex A for the template *"Technical data for the electrical interface of a FlexRay Communication Controller V3.0"*.

9.2 Protocol parameter

— Test purpose

ISO 17458-2 describes in its Annex B a device specific parameter. The purpose of the static test case for the protocol parameter is to show that the device specific value of this parameter is according to ISO 17458-2.

— Description

To be able to claim the static test cases as passed in the sense of this part of ISO 17458, one of the following two alternatives shall be fulfilled for the IUT:

- Alternative 1:
  All gray areas of the template *"Technical data of the protocol parameter of a FlexRay Communication Controller V3.0"* shall be filled in, the filled template shall be handed out, and the handed out template shall have the same binding level as the datasheet.

- Alternative 2:
  All relevant parts of the form *"Technical Data of the Protocol Parameter of a FlexRay Communication Controller V3.0"* (i.e., Table B.2 and Statement 4) shall be included in the datasheet and all gray areas shall be filled in. Additional content (e.g., in the Table B.2) is allowed.

See Annex B for the template *"Technical data for the protocol parameter of a FlexRay Communication Controller V3.0"*.

Annex A
(normative)

Technical data for the electrical interface of a FlexRay Communication Controller V3.0

Annex A contains a template for the technical data for the electrical interface of a FlexRay Communication Controller V3.0. The template consists of Table A.1, Table A.2, and Statements 1 to 3. Table A.1 defines details for the identification of the FlexRay Communication Controller.

Table A.1 — Identification of the FlexRay Communication Controller (electrical interface parameters)

Device Name:	
Company:	
Date:	

Table A.2 defines the electrical interface parameters of the FlexRay Communication Controller according to ISO 17458-4. A minimum or maximum value of '-' means 'not applicable'.

Table A.2 — Electrical interface parameters of ISO 17458-4

Parameter Name	Device specific range		Range as defined in ISO 17458-4		Unit
	Minimum	Maximum	Minimum	Maximum	
dCCTxEN01			-	25	ns
dCCTxEN10			-	25	ns
dCCTxD01			-	25	ns
dCCTxD10			-	25	ns
dCCRxD01			-	10	ns
dCCRxD10			-	10	ns
C_CCRxD			-	10	pF
uCCLogic_1			35	70	%
uCCLogic_0			30	65	%
dCCTxENRise25			-	9	ns
dCCTxENFall25			-	9	ns
dCCTxAsym			-2,45	2,45	ns
dCCTxDRise25 + dCCTxDFall25			-	9	ns
dCCTxDRise25			-	9	ns
dCCTxDFall25			-	9	ns
TxD signal sum of rise and fall time at TP1_BD [a]			-	9	ns
dCCRxAsymAccept15			-31,5	44,0	ns
dCCRxAsymAccept25			-30,5	43,0	ns

[a] The maximum of this parameter is verified by simulation during the design process of the Communication Controller. This is performed according to ISO 17458-4 and the entire temperature range of the device has been taken into account.

Statement 1 The ranges are fulfilled for the temperature range from °C to °C

Statement 2 Device qualification was made according to AEC-Q 100 [17].

Statement 3 The ranges fulfill the allowed ranges as defined in ISO 17458-4.

Annex B
(normative)

Technical data for the protocol parameter of a FlexRay Communication Controller V3.0

Annex B contains a template for the technical data for the protocol parameter of a FlexRay Communication Controller V3.0. The template consists of Table B.1 , Table B.2, and Statement 4.

Table B.1 defines the technical data for the protocol parameter of a FlexRay Communication Controller.

Table B.1 — Identification of the FlexRay Communication Controller (protocol parameter)

Device or IP Name:	
Company:	
Date:	

Table B.2 defines the protocol related parameter of the FlexRay Communication Controller according to ISO 17458-2.

Table B.2 — Protocol parameter

Parameter Name	Bit rate	Value	Unit
adInternalRxDelay	10 Mbit/s		samples
	5 Mbit/s		samples
	2,5 Mbit/s		samples

Statement 4 The value fulfills the allowed range as defined in ISO 17458-2.

Bibliography

[1] ISO/IEC 7498-1, *Information processing systems — Open Systems Interconnection — Basic Reference Model: The Basic Model*

[2] ISO 7498-2, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture*

[3] ISO/IEC 7498-3, *Information technology — Open Systems Interconnection — Basic Reference Model: Naming and addressing*

[4] ISO/IEC 7498-4, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 4: Management framework*

[5] ISO/IEC 9646-1:1994, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 1: General concepts*

[6] ISO/IEC 9646-2:1994, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 2: Abstract Test Suite specification*

[7] ISO/IEC 9646-3:1998, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 3: The Tree and Tabular Combined Notation (TTCN)*

[8] ISO/IEC 9646-6:1994, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 6: Protocol profile test specification*

[9] ISO/IEC 9646-5:1994, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 5: Requirements on test laboratories and clients for the conformance assessment process*

[10] ISO/IEC 9646-7:1995, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 7: Implementation Conformance Statements*

[11] ISO/IEC 9646-7:1995/Cor.1:1997, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 7: Implementation Conformance Statements — Technical Corrigendum 1*

[12] ISO 10681 (all parts), *Road vehicles — Communication on FlexRay*

[13] ISO/IEC 10731, *Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services*

[14] ISO 14229-1:2012, *Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements*

[15] ISO 14229-2:2012, *Road vehicles — Unified diagnostic services (UDS) — Part 2: Session layer services*

[16] ISO 14229-4:2012, *Road vehicles — Unified diagnostic services (UDS) — Part 4: Unified diagnostic services on FlexRay implementation (UDSonFR)*

[17] AEC-Q 100, Stress Qualification For Integrated Circuits, available at: http://www.aecouncil.com/AECDocuments.html

[18] ISO 17458-5, *Road vehicles — FlexRay communications system — Part 5: Electrical physical layer conformance test specification*

ICS 43.040.15

Price based on 865 pages

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

bsi.

...making excellence a habit.™