SLIMbus®

Test Specification

Applicable to Version 1.01

Table of Content

REVISION HISTORY

Release date	Version	Comment
29 NOV 2012	0.5	Initial draft version. Framer tests are not all done yet. Pushed, Pulled, Locked and Asynchronous protocol tests are also missing. Contention tests are missing.
2 DEC 2012	0.6	Device enumeration tests made more robust. Guide Byte generation tests added.
3 DEC 2012	0.65	Added test DE10: Speed Enumeration
28 JAN 2013	0.70	Fix in RS6 (removed check on MS_LOST) Fix and improvements in MPC4

1. Purpose of the document

This document provides a list of tests or verifications to be executed in order to assess the conformance of a component to the SLIMbus specification (version 1.01).

The Method of Implementation (MoI) is not detailed as it depends on the test gears to be used to run the tests.

The test definition details unambiguously:

- The initial conditions of the system (when applicable)
- The test stimulus (when applicable)
- The success criterion

Additionally, the coverage of the test is provided under the form of a list of "shall" statements as they appear in the SLIMbus specification.

2. Test Methodology

The test specification is based on an analysis of the requirements of the SLIMbus specification. The requirement, identified by "shall" statements, cannot be all treated in the same manner.

The current document only relies on the phenomenon that are observable trough the SLIMbus wires (DATA and CLOCK lines).

We have to distinguish 4 levels of testability.

- **Testable.** It means that the Device Under Test (DUT) **can** be controlled to trigger an event that will allow to verify the requirement trough observable signals on the SLIMbus wires within a finite amount of time. The test will rely on well defined initial conditions, stimulus and success criterion. The test result is unambiguous and prove that the requirement is met.
- **Observable.** It means that the Device Under Test (DUT) **cannot** be controlled to trigger a specific event that could allow the verification of the requirement trough observable signals on the SLIMbus wires. The test will rely on continuous bus monitoring and success criterion. The test result is an good indication that the requirement is met. However, the level of confidence is lower than for a **Testable** requirement. In order to overcome this limitation, observable requirements will systematically be verified, irrespectively of the test on going. This will allow testing many requirements in a big variety of bus conditions.
- **Not Observable**. Some requirement are dealing with internal state machine behaviors that are not visible from the "outside", understand through the SLIMbus wires. They are impossible to verify directly. In some cases, some of these requirements are indirectly testable: a given event could not have happened without meeting these requirements. In this case, the requirements are flagged **Testable**. For all the others that are really not observable through SLIMbus wires, no test is proposed. However, in simulation, one can eventually access internal state machines and build a test for these requirements.

- **Not Testable**. Some requirements of the SLIMbus specification are not testable at all. They refer to requirements that are impossible to test in a finite amount of time, definitions, requirements applying to the specification itself or are just an inappropriate usage of the word "shall". Obviously, not test will ever be possible for those requirements.

The main objective is to propose a sequence of tests that will respect the natural flow of SLIMbus events in order to have the minimum number of tests and to start with the most critical requirements and finish with ones impacting the less the interoperability. This method of working will provide the most comprehensive test results, especially when a test is failing.

A lot of requirements linked to corner cases or error recovery will only be testable by producing a specific stimulus.

The test specification will therefore be split into 3 major test sections.

- The first section (Constant Monitoring) describes the verifications to be systematically performed on the SLIMbus bit stream. The tests are no linked to any specific stimulus. This covers the consistency of message format, framing information, bus processes and other events.
- The second section (Natural Flow of Events) will observe and verify what shall happen on the SLIMbus wires when a component runs in normal operation. This will require little specific stimulus. The tests described in this section will systematically run ALL the verifications described in the first section.
- The third section will have dedicated test stimuli for a given requirement. The tests described in this section will systematically run ALL the verifications described in the first section.

Note: Testing a SLIMbus Device means also testing the SLIMbus Component in which it resides. In some cases, it will be virtually impossible to detect trough the SLIMbus wires which device of the Component is accessing the bus, especially when the Device has a non compliant behavior.

3. Continuous Monitoring

This section describes all the verifications that shall be done on the SLIMbus transactions, irrespectively of the test undergoing. These verifications do not depend on any specific initial conditions nor any test stimulus.

3.1. Frame Structure and Information

The frame structure verification mainly tests the ability of the Active Framer to generate all the necessary information to allow the devices to acquire the required synchronization levels and to get enough information on the bus state to operate properly.

This section will cover specification requirements of the chapter 6 of the SLIMbus specification.

3.1.1. FS1 - Frame Synchronization Validity

Test Purpose: Verify the validity of the frame synchronization

The frame synchronization is the first level of SLIMbus synchronization. The frame sync symbol (coded in one slot) occurs every 192 slots (768 bits or clock periods). The interval between two frame sync symbols defines a frame.

Success Criterion:

Verify that the value 0b1011 repeats regularly every 192 slots.

Coverage:

Section	Statement
6.1.2	A Slot is defined as four contiguous Cells and is the unit of bandwidth allocation on SLIMbus. Cells within a Slot are labeled C0 through C3. Slots **shall** be transmitted C3 first followed by C2, C1 and C0, in that order.
6.1.4	A Frame is defined as 192 contiguous Slots. Slots within a Frame are labeled S0 through S191. Frames **shall** be transmitted S0 first followed by S1, S2,... S191, in that order.
6.1.5	Each Frame of a Superframe **shall** contain the Frame Sync symbol in Slot 0.
6.3.1	The Framing Channel **shall** occupy Slot 0 and Slot 96 of each Frame.
6.3.1	The Frame Sync symbol is a fixed bit-sequence, 0b1011. The active Framer **shall** transmit the Frame Sync symbol once per Frame in Slot 0.

State Machine requirements:

State Machine Name: **FrameSyncError**

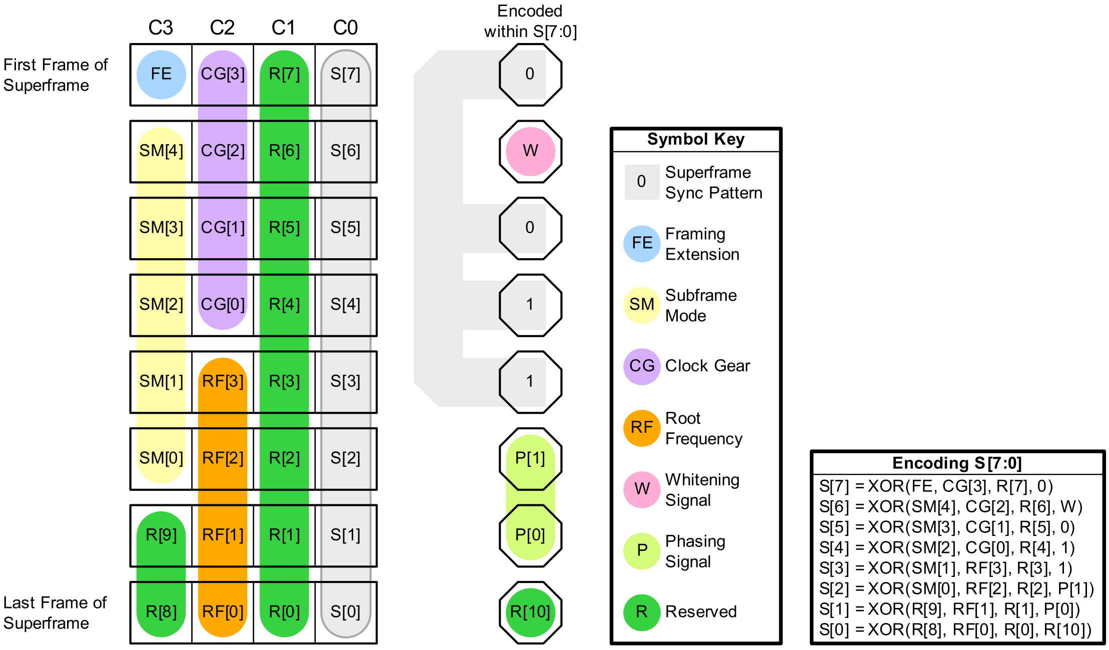
The state machine shall be resettable.

When interrogated, the state machine shall report the number of the last consecutive erroneous frame sync symbols. These values are not modified by a series of valid frame sync symbol. The previous value is reset when an erroneous sync symbol occurs and a new count starts.

3.1.2. FS2 - Superframe Synchronization Validity

Test Purpose: Verify the validity of the superframe synchronization

The superframe synchronization is the second level of SLIMbus synchronization. The superframe sync symbol is built in the 32 bits of the Framing Information (encoded within the S stripe). It repeats every 8 frames (6144 bits or clock periods). Four bit of Framing Information are carried per frames, in slot 96 of the frame. For this verification, the meaning of the bit fields encoded in the Framing Information is not relevant.



Encoding S[7:0]

S[7] = XOR(FE, CG[3], R[7], 0)
S[6] = XOR(SM[4], CG[2], R[6], W)
S[5] = XOR(SM[3], CG[1], R[5], 0)
S[4] = XOR(SM[2], CG[0], R[4], 1)
S[3] = XOR(SM[1], RF[3], R[3], 1)
S[2] = XOR(SM[0], RF[2], R[2], P[1])
S[1] = XOR(R[9], RF[1], R[1], P[0])
S[0] = XOR(R[8], RF[0], R[0], R[10])

Success Criterion:

Once the punctured pattern **0X011**XXX (where X is "don't care") has been decoded out of the S[7:0] stripe, verify that it repeats regularly every 8 frames.

Coverage:

Section	Statement
6.1.2	A Slot is defined as four contiguous Cells and is the unit of bandwidth allocation on SLIMbus. Cells within a Slot are labeled C0 through C3. Slots **shall** be transmitted C3 first followed by C2, C1 and C0, in that order.
6.1.5	A Superframe is defined as eight contiguous Frames (1536 Slots). Frames within a Superframe are labeled Frame 0 through Frame 7. Superframes **shall** be transmitted Frame 0 first followed by Frame 1, Frame 2,… Frame 7, in that order.
6.1.5	In addition, the first Frame (Frame 0) of a Superframe **shall** contain the first four bits of Framing Information in Slot 96.
6.1.5	Frames 1 through Frame 7 **shall** also contain four bits of Framing Information in Slot 96 with Frame 7 carrying the last four bits of Framing Information.
6.3.1	The active Framer **shall** transmit four bits of Framing Information once per Frame in Slot 96.
6.3.1	The complete Framing Information **shall** be sent in eight consecutive Frames.
6.3.1.1	It [Framing Information] **shall** also include a stripe S into which the active Framer encodes a Superframe Sync pattern, a Whitening Signal and a Phasing Signal.
6.3.1.1	The coding **shall** be performed via the parity of the Slot 96 nibbles. For example, in the sixth Frame of the Superframe, the active Framer shall send S[2] as the exclusive-OR of SM[0], RF[2], R[2], and P[1].

6.3.1.3	The Superframe Sync pattern is a punctured repeating pattern, 0X011XXX, that allows a Component to detect the beginning of a new Superframe. The active Framer **shall** encode the Superframe Sync pattern in stripe S of the Framing Information using the equations shown in Figure 24.

State Machine requirements:

State Machine Name: **SuperFrameSyncError**

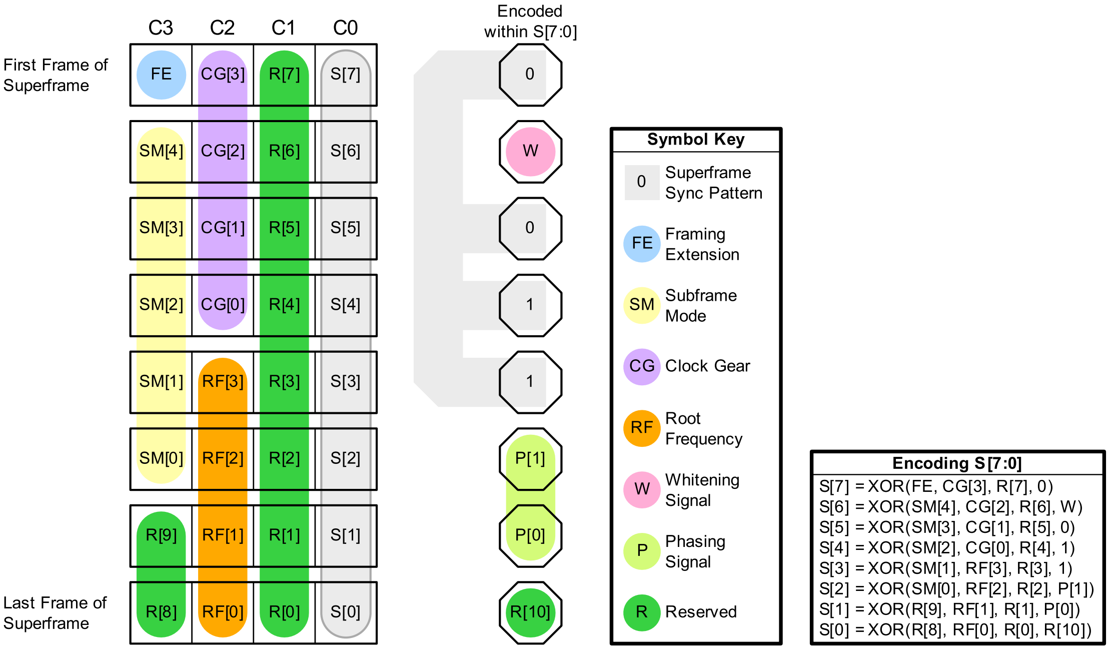
The state machine shall be resettable.

When interrogated, the state machine shall report the number of the last consecutive erroneous superframe sync symbols. These values are not modified by a series of valid superframe sync symbol. The previous value is reset when an erroneous sync symbol occurs and a new count starts.

3.1.3. FS3 - Framing Information Validity - Framing Extension bit

Test Purpose: Verify that the Framing Extension bit is always equal to 0

The Framing Extension bit is the first Framing Information bit transmitted at a beginning of a superframe (Frame 0, Slot 96, Cell C3). The Framing Extension bit must have the vale 0 in order to indicate a SLIMbus 1.0 compatible Framing Information block. SLIMbus 1.0 compliant Active Framer are not allowed to transmit anything else than the value 0.



Encoding S[7:0]

S[7] = XOR(FE, CG[3], R[7], 0)
S[6] = XOR(SM[4], CG[2], R[6], W)
S[5] = XOR(SM[3], CG[1], R[5], 0)
S[4] = XOR(SM[2], CG[0], R[4], 1)
S[3] = XOR(SM[1], RF[3], R[3], 1)
S[2] = XOR(SM[0], RF[2], R[2], P[1])
S[1] = XOR(R[9], RF[1], R[1], P[0])
S[0] = XOR(R[8], RF[0], R[0], R[10])

Success Criterion:

Verify that the Framing Extension bit always have value 0.

Coverage:

Section	Statement
6.3.1.1	Framing Information **shall** include a flag FE and fields SM, CG, RF and R as shown in Figure 24.
6.3.1.2	The active Framer **shall** set FE to 0 to indicate that the Subframe Mode (SM), Clock Gear (CG), Root Frequency (RF) and Reserved (R) fields are present in the current Superframe.

State Machine requirements:

State Machine Name: **FramingExtError**

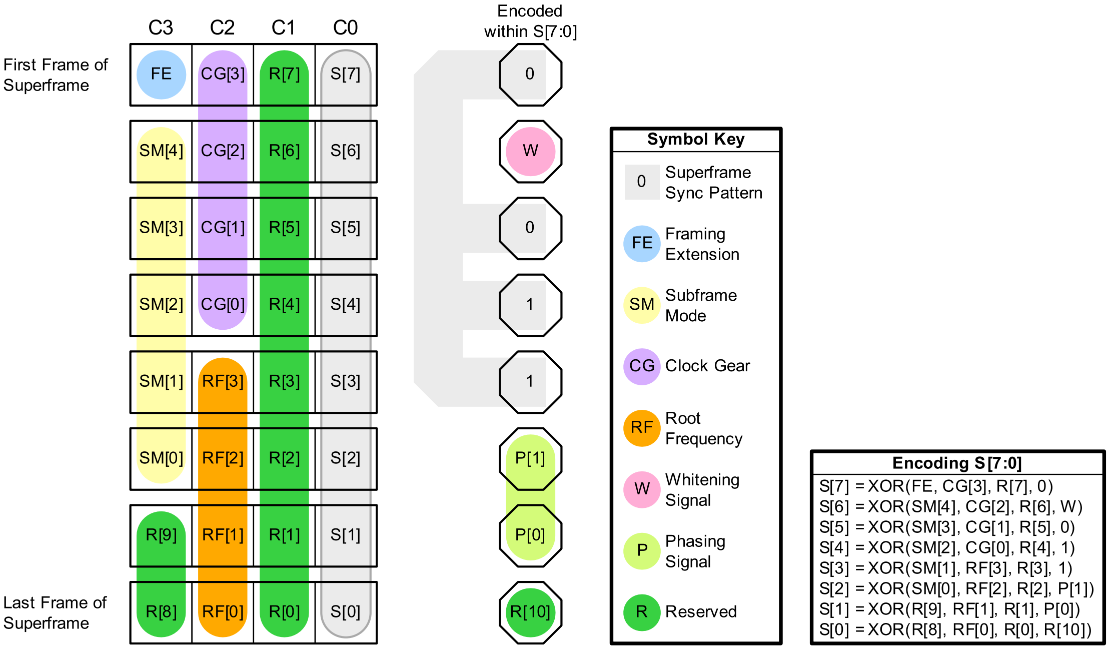
The state machine shall be resettable.

When interrogated, the state machine shall report the number of the last consecutive Framing Extension bit values equal to 1. These values are not modified by a series of valid Framing Extension bit value equal to 0. The previous value is reset when a Framing Extension bit values equal to 1 occurs and a new count starts.

3.1.4. FS4 - Framing Information Validity - Reserved bits

Test Purpose: Verify that the Reserved bits are always equal to 0

The eleven Reserved bits R[10:0] of the Framing Information must have a value equal to 0. Note that the actual value of the Reserved bits will not impact the decoding the the information contained in the stripe S[7:0].



Success Criterion:

Verify that the Reserved bit values are always equal to 0.

Coverage:

Section	Statement
6.3.1.1	Framing Information **shall** include a flag FE and fields SM, CG, RF and R as shown in Figure 24.
6.3.1.2	The active Framer **shall** set the reserved bits, R[10:0], to all zeros.

State Machine requirements:

State Machine Name: **FramingInfoResvBitError**

When interrogated, the state machine shall report the value of the field R[10:0].

3.1.5. FS5 - Framing Information Validity - Unexpected Changes

Test Purpose: Detect unexpected framing Information changes

The Subframe Mode SM[4:0], Clock Gear CG[3:0] and Root Frequency RF[3:0] bit field values cannot change autonomously. Any change in these fields must be preceded by a valid bus reconfiguration sequence issued by the Active Manager.

Success Criterion:

Verify that the value of the Subframe Mode SM[4:0], Clock Gear CG[3:0] and Root Frequency RF[3:0] bit fields does not change autonomously (in the absence of any valid reconfiguration sequence).

Coverage:

Section	Statement
6.3.1.1	A change in Framing Information without such an announcement **shall** indicate an error condition on the bus. See section 10.2.4 for the proper response to such an error condition.
6.3.1.4	The active Framer **shall** only change the Subframe Mode at the direction of the active Manager as specified in section 10.6.
6.3.1.5	The active Framer **shall** only change the Clock Gear at the direction of the active Manager as specified in section 10.5.
6.3.1.6	The active Framer **shall** only change the Root Frequency at the direction of the active Manager as specified in section 10.7.

State Machine requirements:

State Machine Name: **FramingInfoSpuriousChange**

The state machine shall be resettable.

The reported byte value has the following structure: [0,0,0,0,0,fRF,fCG,fSM]. **fRF** is set when a spurious change happens in the Root Frequency field. **fCG** is set when a spurious change happens in the Clock Gear field. **fSM** is set when a spurious change happens in the Subframe Mode field.

When interrogated, the state machine shall report the byte value.

The state machine value can only be cleared by a reset.

3.1.6. FS6 - Framing Information Validity - Clock Gear

Test Purpose: Verify the validity of the broadcasted Clock Gear

The bus frequency depends on 2 parameters: the Clock Gear and the Root Frequency. The actual bus clock is equal to the root frequency divided by 2 to the power 10-CG ($2^{\wedge(10\text{-}CG)}$).

Success Criterion:

If the broadcasted Clock Gear value is not equal to "Not Indicated", measure the duration of a clock period and compute the bus frequency. Verify that the bus frequency is compatible with the values given in table 16 of the SLIMbus specification.

Coverage:

Section	Statement
6.3.1.5	The Clock Gear field, CG[3:0], in the Framing Information **shall** be encoded as shown in Table 16.

State Machine requirements:

State Machine Name: **ClockGearMismatch**

When interrogated, the state machine shall compute the actual clock gear, based on the measured clock period and compare the actual clock gear with the broadcasted clock gear. When the 2 gears are equal, the state machine reports 0, else 1.

3.1.7. FS7 - Framing Information Validity - Root Frequency

Test Purpose: Verify the validity of the broadcasted Root Frequency

The bus frequency depends on 2 parameters: the Clock Gear and the Root Frequency. The actual bus clock is equal to the root frequency divided by 2 to the power 10-CG ($2^{\wedge(10\text{-}CG)}$).

Success Criterion:

If the broadcasted Clock Gear value is not equal to "Not Indicated", measure the duration of a clock period and compute the bus frequency. Use the broadcasted Clock Gear value to compute the Root Frequency. Verify that computed Root Frequency and the Root Frequency broadcasted in the Framing Information are similar (less than 5% difference). Use the values given in table 17 of the SLIMbus specification to derive the Root Frequency from the RF field value.

Coverage:

Section	Statement
6.2.3	The Root Frequency **shall** be $2^{\wedge(10\text{-}CG)}$ times the frequency of the CLK line, where CG is the current Clock Gear.
6.3.1.6	The Root Frequency field, RF[3:0], in the Framing Information **shall** indicate the Root Frequency of the bus as coded in Table 17.

State Machine requirements:

State Machine Name: **RootFrequencyMismatch**

When interrogated, the state machine shall compute the actual Root Frequency, based on the measured clock period and the broadcasted Clock Gear and compare the actual Root Frequency with the broadcasted Root Frequency. When the two frequencies are similar, the state machine reports 0, else 1.

3.1.8. FS8 - Framing Information Validity - Whitening Signal

Test Purpose: Verify the behavior of the Whitening signal

The Whitening signal purpose is to spread the energy contained in the SLIMbus data line signal. The Whitening bit is transmitted at the superframe rate. The absolute value of the whitening signal sequence's autocorrelation coefficient must be less than 0.1 for all lags from 1 up to 14.

Note: the autocorrelation is the cross-correlation of a signal with itself. Informally, it is the similarity between observations as a function of the time separation between them. It is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal which has been buried under noise. The discrete autocorrelation R_k at lag k for a discrete signal Y_t is:

$$R_k = \frac{\sum_{t=1}^{N-k} \left(Y_t - Y_{AVG}\right)\left(Y_{t+k} - Y_{AVG}\right)}{\sum_{t=1}^{N} \left(Y_t - Y_{AVG}\right)^2},$$

with Y_{AVG} the average value of the Y sequence.

Success Criterion:

Record **29** consecutive Whitening bits and perform the calculation of R_k with k ranging from 1 to 14. All R_k coefficient absolute values must be less than 0.1.

Coverage:

Section	Statement
6.3.1.7	The active Framer **shall** write a sequence having appropriate properties to the Whitening Signal.
6.3.1.7	Specifically, the absolute values of the sequence's autocorrelation coefficients **shall** be less than 0.1 for all lags from 1 up to at least 14.

State Machine requirements:

State Machine Name: **WhiteningSignalError**

The state machine shall be resettable (reset the **29** bits buffer).

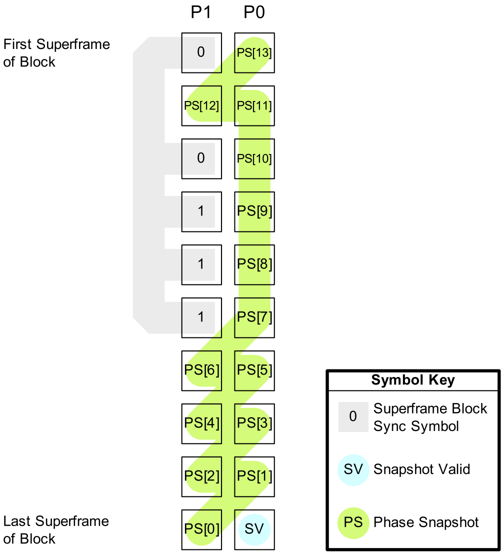
Once the input buffer is loaded with **29** consecutive whitening bits, perform the calculation of the sequence average (it should normally be close to 0.5). Then compute the 14 R_k coefficients. If one of the R_k absolute value is higher than 0.1, reports 1, else 0.

When the buffer is full and a new bit value is acquired, discard the oldest sample and shift down all values. Add in position **29** the new sample and proceed with the described calculations.

3.1.9. FS9 - Framing Information Validity - Phasing Signal Synchronization

Test Purpose: Detect valid block synchronization

The Phasing signal is built on 20 bits. 2 bits (P0 and P1) are transmitted in the Framing Information (encoded in the S stripe) every superframe. A superframe block is built on 10 superframes. The block boundary is indicated by a punctured pattern: **0**X**0111**XXXX (where X is "don't care") coded in the bit P1.



The other fields of the Phasing Signal are not of interest for the detection of the sync pattern.

Success Criterion:

Once the punctured pattern **0**X**0111**XXXX has been decoded out of the P1 sequence, verify that it repeats regularly every 10 superframes.

Coverage:

Section	Statement
6.3.1.8	The active Framer **shall** generate the Phasing Signal.
6.3.1.8	The active Framer **shall** group the Superframes into consecutive blocks of ten.
6.3.1.8	Within each Superframe block, the active Framer **shall** organize the Phasing Signal as shown in Figure 25.

State Machine requirements:

State Machine Name: **BlockSyncError**

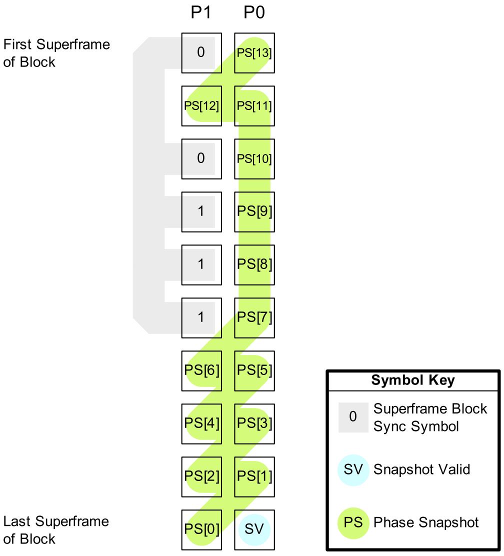
The state machine shall be resettable.

When interrogated, the state machine shall report the number of the last consecutive erroneous block sync symbols. These values are not modified by a series of valid block sync symbol. The previous value is reset when an erroneous sync symbol occurs and a new count starts.

3.1.10. FS10 - Framing Information Validity - Phase SnapShot

Test Purpose: Verify the validity of the Phase Snapshot

The Phase Snapshot is coded on 14 bits: PS[13:0], partly on P1 and P0. The value must be ready in the Active Framer at the beginning of a superframe block. It contains the number of the first "virtual" root superframe in the following block. A valid Phase Snapshot is flagged by the Snapshot Valid bit (SV) equal to 1.



The following formula proposes a method to compute the Phase Snapshot:

PS=(PhaseBlock + 10*2^(10-CG)) Modulo PM

where PhaseBlock is the number of the first Root Superframe in the block, CG is the Clock Gear during that Root Superframe, and PM is the Phase Modulus during that Root Superframe.

Success Criterion:

In the absence of a change in Root Frequency or Clock Gear in a block, the difference between two consecutive valid Phase Snapshot values must be equal to **10*2^(10-CG)** and the SV bit value must be equal to **1.** The modulo PM must be taken into account during the verification. When PM-PhaseBlock< **10*2^(10-CG)**, one must expect the modulo to happen in the next PS value.

Coverage:

Section	Statement
6.3.1.8	Table 17 specifies the PM values that **shall** be used at particular Root Frequencies.
6.3.1.8	If the active Framer knows PS field contains valid information, it **shall** set the SV field to 1
6.3.1.8	A valid PS field **shall** contain the number of the first Root Superframe in the following block, calculated with the assumption that there will be neither a Root Frequency change nor a Clock Pause at the start of that block.
6.3.1.8	The active Framer may transmit SV=0 in blocks that span or immediately follow a Clock Gear change, a Root Frequency change or a Clock Pause. In all other blocks it **shall** transmit a valid Phase Snapshot with SV=1, except when the current Clock Gear is "Not Indicated" in which case it **shall** transmit PS[13:0]=0x0000 and SV=0.

State Machine requirements:

State Machine Name: **PhaseSnapshotError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of the last consecutive erroneous Phase Snapshot values. This value are not modified by a series of valid Phase Snapshot values. The previous value is reset when an erroneous Phase Snapshot value occurs and a new count starts.

3.1.11. FS11 - Framing Information Validity - SnapShot Valid

Test Purpose: Verify the validity of the Snapshot Valid bit

The Snapshot Valid bit must be set to 0 certain events happen in the current block:

- A bus reconfiguration that modifies the Root Frequency
- A bus reconfiguration that modifies the Clock Gear
- A Clock Pause
- A Clock Gear of value 0 ("Not Indicated")

Success Criterion:

When one or more of the events described above happen in the current block, verify that the PV bit value is equal to 0.

Coverage:

Section	Statement
6.3.1.8	If the active Framer knows PS field does not contain valid information, it **shall** set the SV field to 0.
6.3.1.8	The active Framer may transmit SV=0 in blocks that span or immediately follow a Clock Gear change, a Root Frequency change or a Clock Pause. In all other blocks it **shall** transmit a valid Phase Snapshot with SV=1, except when the current Clock Gear is "Not Indicated" in which case it **shall** transmit PS[13:0]=0x0000 and SV=0.

State Machine requirements:

State Machine Name: **SnapshotValidError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of the last consecutive erroneous Snapshot Valid values. This value is not modified by a series of good Snapshot Valid values. The previous value is reset when an erroneous Snapshot Valid value occurs and a new count starts.

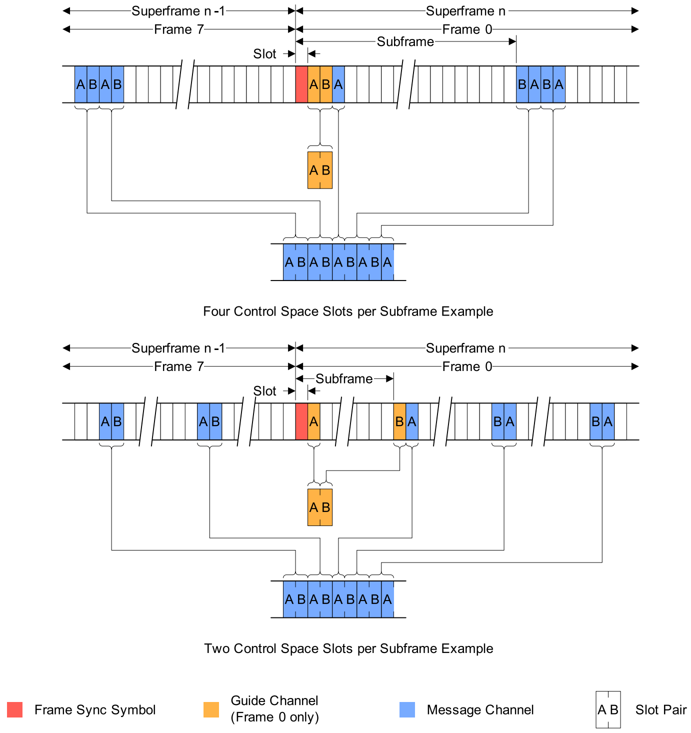
3.2. Guide Channel

The guide channel is built on 2 slots of the control space. It is the message synchronization method.

The Guide Byte is spread over the 2 first slots of the control space following the frame sync symbol of the first frame of a superframe. The 2 slots are not necessarily contiguous in the frame, depending on the subframe mode.

The Guide Byte slots are not tagged in a special way. The only way the verify the Guide byte validity is to monitor the value of the slots where the guide byte is supposed to be located. For instance, when the Message Channel is idle, the slot containing the least significant bit of the guide byte must exhibit the value **0x3** and the slot containing the most significant bit must exhibit the value **0x0**.

The Guide Channel slot containing the **most significant** bits is the slot (A) of the control space that immediately follows the frame sync slot. The Guide Channel slot containing the **least significant** bit is the slot (B) of the control space that immediately follows the MSb slot of the Guide Byte.



Four Control Space Slots per Subframe Example

Two Control Space Slots per Subframe Example

The Guide Byte contains 3 bit fields:

- The ENT bit (GB[7]) that indicates where the superframe boundary happens in the message.
- The GV[4:0] bits (GB[6:2]), called Guide Value, indicates the remaining number of bytes (the messages are byte based) until the end of the message or the end of the RL (Remaining Length) field, depending on the ENT bit value.
- The GI[1:0] bits (GB[1:0]), called Guide Integrity, indicates a possible error in the Guide Byte validity. It is a dual parity check.

Bit	7	6	5	4	3	2	1	0
	ENT	GV[4]	GV[3]	GV[2]	GV[1]	GV[0]	GI[1]	GI[0]

This section will mostly cover Guide Channel specification requirements of the chapter 8 of the SLIMbus specification.

3.2.1. GC1 - Guide Byte - Integrity Validity

Test Purpose: Verify the validity of the Guide Byte integrity

The verification performed in this test does not verify if the value of the ENT bit or the GV field is valid. It just computes the GI[0] and GI[1] values based on the ENT bit and GV bit field and then compare the results with the GI[1:0] bit values broadcasted by the Guide Byte.

- GI[0] = NOT(GV[4] AND GV[2] AND GV[0])
- GI[1] = NOT(ENT AND GV[3] AND GV[1])

A difference between the computed values and the broadcasted values will indicate a corruption of the Guide Value or of the ENT bit and will make the Guide Byte unusable for message synchronization.

Success Criterion:

Verify that the computed GI[1:0] bit values with the ENT and GV values broadcasted by the Guide Byte and the GI[1:0] bit values broadcasted by the Guide Byte have equal values.

Coverage:

Section	Statement
6.3.2	The Guide Byte is the only information sent in the Guide Channel and **shall** be carried in the Superframe Slots as shown in Table 18.
8.1	The Guide Channel is composed of two Slots per Superframe, called the Guide Byte. The Guide Byte **shall** be written by the active Framer, and is illustrated in Table 23.

8.1.1	The active Framer **shall** generate the Guide Integrity field as follows:
	GI[0] = ~(GV[4] ^ GV[2] ^ GV[0])
	GI[1] = ~(ENT ^ GV[3] ^ GV[1])

State Machine requirements:

State Machine Name: **GuideByteIntegrityError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the Guide Byte integrity check. The state machine is not reset by a valid integrity check. Only a reset can clear the state machine.

3.2.2. GC2 - Guide Byte - ENT bit Validity

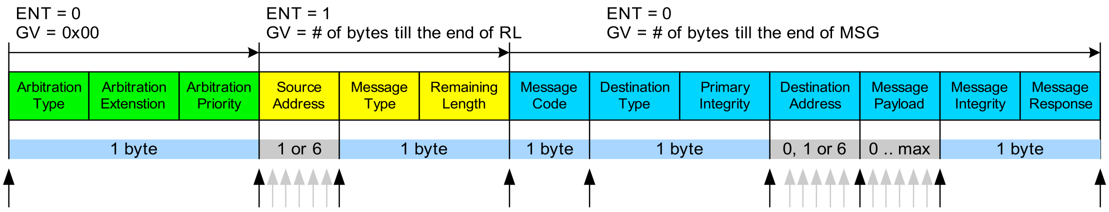
Test Purpose: Verify the validity of the Guide Byte ENT bit

The verification performed in this test only verify if the value of the ENT bit broadcasted in the Guide Byte correspond with the expected value.

The ENT bit indicates if the Guide Byte occurred between the Remaining Length (RL) field and the start of the next Message.

If the Guide Byte occurs after the RL field, but before the start of the next Message, the ENT bit must be set to zero. If the Guide Byte occurs elsewhere, the ENT bit must be set to one.

As the messages are byte based, the superframe boundary (indicated by black / grey vertical arrows) cannot happen anywhere in the messages but only at a byte boundary.



Success Criterion:

Verify that the computed ENT bit and the ENT bit broadcasted by the Guide Byte are equal.

Coverage:

Section	Statement
6.3.2	The Guide Byte is the only information sent in the Guide Channel and **shall** be carried in the Superframe Slots as shown in Table 18.
8.1	The Guide Channel is composed of two Slots per Superframe, called the Guide Byte. The Guide Byte **shall** be written by the active Framer, and is illustrated in Table 23.
8.1.2	If the Guide Byte occurs after the RL field, but before the start of the next Message, the ENT bit **shall** be set to zero.
8.1.2	If the Guide Byte occurs elsewhere [not after the RL field, but before the start of the next Message], the ENT bit **shall** be set to one.
8.1.3	If the Guide Byte occurs immediately after the end of a Message, the ENT bit and the Guide Value **shall** be zero.

State Machine requirements:

State Machine Name: **GuideByteENTbitError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the ENT bit validity check. The state machine is not reset by a valid ENT bit check. Only a reset can clear the state machine.

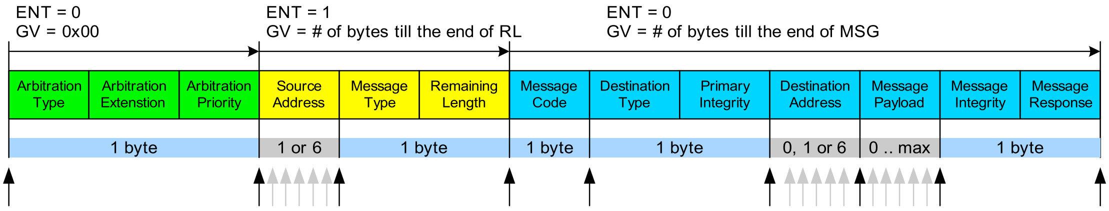
3.2.3. GC3 - Guide Byte - Guide Value Validity

Test Purpose: Verify the validity of the Guide Value

The verification performed in this test does only verify if the value of the GV field broadcasted in the Guide Byte correspond with the expected value. It supposes that the ENT bit and the Guide Byte integrity are valid.

Depending where the Guide Byte is inserted, the GV value has different meanings:

- When ENT=0, the Guide Value shall be the number of bytes in the Message Channel, 0 to 31, that remains after the Guide Byte until the end of the current Message.
- When ENT=1, the Guide Value shall be the number of bytes in the Message Channel, 1 to 7, that remains after the Guide Byte until the end of the RL field.



Success Criterion:

Verify that the computed GV[4:0] bit values and the GV[4:0] bit values broadcasted by the Guide Byte are equal.

Coverage:

Section	Statement
6.3.2	The Guide Byte is the only information sent in the Guide Channel and **shall** be carried in the Superframe Slots as shown in Table 18.
8.1	The Guide Channel is composed of two Slots per Superframe, called the Guide Byte. The Guide Byte **shall** be written by the active Framer, and is illustrated in Table 23.
8.1.3	When ENT=0, the Guide Value **shall** be the number of bytes in the Message Channel, 0 to 31, that remains after the Guide Byte until the end of the current Message.
8.1.3	When ENT=1, the Guide Value **shall** be the number of bytes in the Message Channel, 1 to 7, that remains after the Guide Byte until the end of the RL field.
8.1.3	If the Guide Byte occurs immediately after the end of a Message, the ENT bit and the Guide Value **shall** be zero.

State Machine requirements:

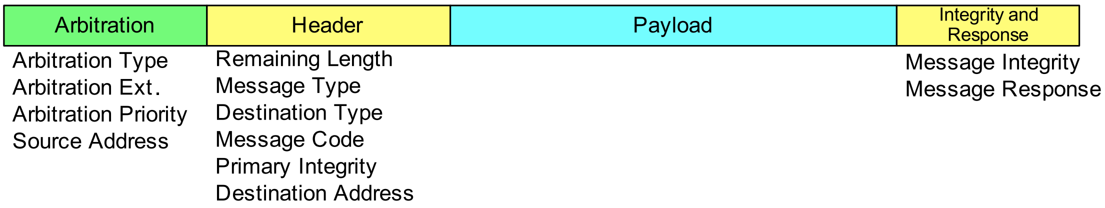
State Machine Name: **GuideValueError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the Guide Value validity check. The state machine is not reset by a valid Guide Value check. Only a reset can clear the state machine.

3.3. Message Syntax

The message rules checking verifies the consistency of the various message fields. The verification does not depends on the kind of message nor on the source or destination of the message.



Most of the fields have reserved or illegal values that cannot be used in nominal operations.

Some others have interdependencies like the Source Address and the Arbitration Type or the Destination Type and Destination Address. If the wrong address structure is used, the only way to detect it will be through a failed Primary Integrity CRC and likely numerous errors in the Arbitration and Header fields.

The verification performed in this section assumes that the bit fields are sequenced as defined in the SLIMbus specification. It is not possible to detect a misplaced bit field. In such case, the CRC checks will fail (and likely some other verifications), without telling exactly what was wrong.

This section will mostly cover specification requirements of the chapter 8 of the SLIMbus specification.

3.3.1. MS1 - Arbitration - Arbitration Type Validity

Test Purpose: Verify that the value of the Arbitration Type bit field is not reserved or illegal

The Arbitration Type is coded on 4 bits (AT[3:0]). Not because 16 types are possible but to handle properly the logical OR signaling used during arbitration. The Short arbitration has priority over the long arbitration and therefore gets the highest possible value. Only 3 values can happen in the Arbitration Type: 0b0000, 0b0101 and 0b1111. All others are illegal.

Arbitration Type	Description
0b0000	No Arbitration
0b0001 to 0b0100	**Illegal**
0b0101	Long Arbitration
0b0110 to 0b1110	**Illegal**
0b1111	Short Arbitration

Success Criterion:

Verify that none of the illegal values occur in the Arbitration Type bit field.

Coverage:

Section	Statement
8.2	A Device **shall** not send a Message with a field set to any of its reserved or illegal values.

State Machine requirements:

State Machine Name: **ArbitrationTypeError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Arbitration Type value. The state machine is not reset by a valid message Arbitration Type value. Only a reset can clear the state machine.

3.3.2. MS2 - Arbitration - Arbitration Extension Validity

Test Purpose: Verify that the value of the Arbitration Extension bit is not reserved

The Arbitration Extension bit is set by default to value 0. It is meant for future development of the message structure (if any). In SLIMbus 1.0, this bit can't have another value than 0.

Arbitration Extension	Description
0b0	Normal
0b1	**Reserved**

Success Criterion:

Verify that the Arbitration Extension bit is always equal to 0.

Coverage:

Section	Statement
8.2	A Device **shall** not send a Message with a field set to any of its reserved or illegal values.

State Machine requirements:

State Machine Name: **ArbitrationExtError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Arbitration Extension value. The state machine is not reset by a valid message Arbitration Extension value. Only a reset can clear the state machine.

3.3.3. MS3 - Arbitration - Arbitration Priority Validity

Test Purpose: Verify that the value of the Arbitration Priority bit field is not set to “reserved”

The Arbitration Priority bit is coded on 3 bits (AP[2;:0]). All values are allowed but 0b000.

Arbitration Priority	Description
0b000	**Reserved**
0b001	Low Priority Messages
0b010	Default Messages
0b011	High Priority Messages
0b100	Manager assigned only
0b101	Manager assigned only
0b110	Manager assigned only
0b111	Maximum Priority, for test and debug only

Success Criterion:

Verify that the Arbitration Priority bit field value is never equal to 0.

Coverage:

Section	Statement
8.2	A Device **shall** not send a Message with a field set to any of its reserved or illegal values.

State Machine requirements:

State Machine Name: **ArbitrationPriorityError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Arbitration Priority value. The state machine is not reset by a valid message Arbitration Priority value. Only a reset can clear the state machine.

3.3.4. MS4 - Arbitration - Source Address Validity

Test Purpose: Verify that the value of the Source Address bit field is not reserved

When the Source Address is a Logical address (corresponding to Arbitration Type value 0b1111), its value must not be within 0xF0 to 0xFE, as it is a reserved address range.

Logical Address	Description
0xFF	Active Manager. Shall not be assigned by active Manager
0xF0 to 0xFE	**Reserved. Shall not be assigned by active Manager.**
0x00 to 0xEF	Available for general use

Success Criterion:

Verify that the Source Address value, when a Logical Address, is never within 0xF0 to 0xFE.

Coverage:

Section	Statement
8.2.5.2	The logical addresses 0xF0 to 0xFE **shall** not be assigned by the Active Manager (reserved).

State Machine requirements:

State Machine Name: **SourceAddressError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Source Address value. The state machine is not reset by a valid Source Address value. Only a reset can clear the state machine.

3.3.5. MS5 - Header - Reserved Cells Value

Test Purpose: Verify that the value of the Reserved cells is always 0

In the message header, 3 Cells are Reserved. They are used as padding bits to keep the byte alignment of the message header.

Byte	7	6	5	4	3	2	1	0
0	MT[2]	MT[1]	MT[0]	RL[4]	RL[3]	RL[2]	RL[1]	RL[0]
1	Rsvd=0	MC[6]	MC[5]	MC[4]	MC[3]	MC[2]	MC[1]	MC[0]
2	Rsvd=0	Rsvd=0	DT[1]=1	DT[0]=1	PI[3]	PI[2]	PI[1]	PI[0]

Success Criterion:

Verify that the bit 7 of the second byte of the message header is always equal to 0. Verify that the bits 6 and 7 of the third byte of the message header are always equal to 0.

Coverage:

Section	Statement
8.2	A Device **shall** set all reserved Cells in a field to zero when sending a Message.
8.2.2	There are three reserved Cells in the Header field. A Device **shall** set these Cells to zero.

State Machine requirements:

State Machine Name: **HeaderResvBitError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in message header reserved bits. The state machine is not reset by a valid message header reserved bit value. Only a reset can clear the state machine.

3.3.6. MS6 - Header - Message Type Validity

Test Purpose: Verify that the value of the Message Type bit field is not reserved or illegal

The Message Type bit field is coded on 3 bits (MT[2:0]). Some of its values are reserved or illegal and must not be used.

Message Type	Description
0b000	Core Message
0b001	Destination-referred Class-specific Message
0b010	Destination-referred User Message
0b011	**Illegal**
0b100	**Reserved**
0b101	Source-referred Class-specific Message
0b110	Source-referred User Message
0b111	**Illegal**

Success Criterion:

Verify that the Message Type value is never equal to any of the Reserved or Illegal values (0b011, 0b100 and 0b111).

Coverage:

Section	Statement
8.2	A Device **shall** not send a Message with a field set to any of its reserved or illegal values.

State Machine requirements:

State Machine Name: **MessageTypeError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Message Type value. The state machine is not reset by a valid Message Type value. Only a reset can clear the state machine.

3.3.7. MS7 - Header - Message Code Validity

Test Purpose: Verify that the value of the Message Code bit field is not reserved

The Message Code bit field is coded on 7 bits (MC[6:0]). Some of its values are reserved and must not be used. Table 65 of the SLIMbus specification list all the authorized Core Message Codes. Any other codes are reserved.

Success Criterion:

Verify that the Message Code value is never equal to any of the Reserved value when the Message Type indicates a Core Message (MT=0b000).

Coverage:

Section	Statement
11	A source Device **shall** not send a Message with a reserved Core Message Code.

State Machine requirements:

State Machine Name: **MessageCodeError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Message Code value. The state machine is not reset by a valid Message Code value. Only a reset can clear the state machine.

3.3.8. MS8 - Header - Destination Type Validity

Test Purpose: Verify that the value of the Destination Type bit field is not reserved

The Destination Type bit field is coded on 2 bits (DT[1:0]). One of its values (0b10) is reserved and must not be used.

Destination Type	Description
0b00	Destination is a Logical Address
0b01	Destination is an Enumeration Address
0b10	**Reserved**
0b11	All Devices are Destinations, no Destination Address included in Header

Success Criterion:

Verify that the Destination Type value is never equal to its Reserved values (0b10).

Coverage:

Section	Statement
8.2	A Device **shall** not send a Message with a field set to any of its reserved or illegal values.

State Machine requirements:

State Machine Name: **DestinationTypeError**

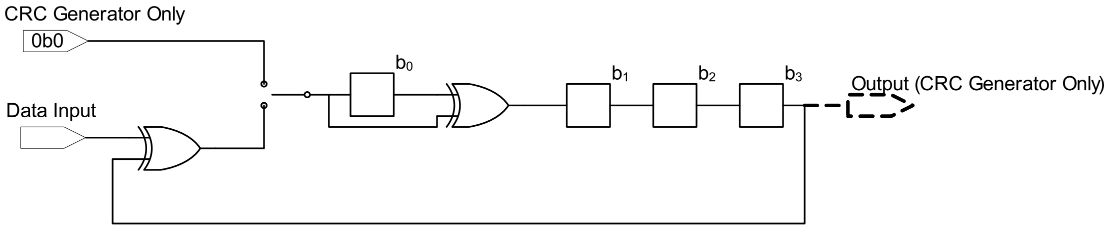
The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Destination Type value. The state machine is not reset by a valid Destination Type value. Only a reset can clear the state machine.

3.3.9. MS9 - Header - Primary Integrity Validity

Test Purpose: Verify that the value of Primary Integrity CRC is valid

Arbitration and Header fields (excluding the Destination Address) are protected by a 4 bit CRC called Primary Integrity. Though specific isolated requirements of the CRC generation are not observable as such through SLIMbus wires, the end result (the CRC value) is.



The formula to compute the CRC value is X4+X+1.

Success Criterion:

Verify that the broadcasted CRC value matches the computed CRC value.

Coverage:

Section	Statement
8.2.4.1	Reserved Cells **shall** be included when performing Integrity CRC calculations.
8.2.4.1	The CRC **shall** be calculated in the following manner [include the paragraph description - redundant shall].
8.2.4.1	The CRC generator **shall** be initialized to 0b0000.
8.2.4.1	During each following Cell the CRC **shall** be updated according to the formula CRC = X4 + X + 1, as implemented in Figure 38 with the switch set to the XOR gate, or equivalent.

State Machine requirements:

State Machine Name: **PrimaryIntegrityError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the Primary Integrity CRC. The state machine is not reset by a valid Primary Integrity CRC. Only a reset can clear the state machine.

3.3.10. MS10 - Header - Destination Address Validity

Test Purpose: Verify that the value of the Destination Address bit field is not reserved

When the Destination Address is a Logical address (corresponding to DestinationType value 0b00), its value must not be within 0xF0 to 0xFE, as it is a reserved address range.

Logical Address	Description
0xFF	Active Manager. Shall not be assigned by active Manager
0xF0 to 0xFE	**Reserved. Shall not be assigned by active Manager.**
0x00 to 0xEF	Available for general use

Success Criterion:

Verify that the Destination Address value, when a Logical Address, is never within 0xF0 to 0xFE.

Coverage:

Section	Statement
8.2.5.2	The logical addresses 0xF0 to 0xFE **shall** not be assigned by the Active Manager (reserved).

State Machine requirements:

State Machine Name: **DestinationAddressError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in Destination Address value. The state machine is not reset by a valid Destination Address value. Only a reset can clear the state machine.

3.3.11. MS11 - Payload - Reserved Cells Value

Test Purpose: Verify that the value of the Reserved cells in the Core Message Payload is always 0

In the Core Message Payload, some Cells are Reserved. They are used as padding bits to keep the byte alignment of the message payload. The Reserved Cells must always have a value equal to 0.

CHANGE_ARBITRATION_PRIORITY (AP)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	-	-	-	-	-	AP[2]	AP[1]	AP[0]

CONNECT_SOURCE (PN, CN)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	-	-	PN[5]	PN[4]	PN[3]	PN[2]	PN[1]	PN[0]
Byte 1	CN[7]	CN[6]	CN[5]	CN[4]	CN[3]	CN[2]	CN[1]	CN[0]

CONNECT_SINK (PN, CN)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	-	-	PN[5]	PN[4]	PN[3]	PN[2]	PN[1]	PN[0]
Byte 1	CN[7]	CN[6]	CN[5]	CN[4]	CN[3]	CN[2]	CN[1]	CN[0]

DISCONNECT_PORT (PN)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	-	-	PN[5]	PN[4]	PN[3]	PN[2]	PN[1]	PN[0]

CHANGE_CONTENT (CN, FL, PR, AF, DT, CL, DL)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	CN[7]	CN[6]	CN[5]	CN[4]	CN[3]	CN[2]	CN[1]	CN[0]
Byte 1	FL	PR[6]	PR[5]	PR[4]	PR[3]	PR[2]	PR[1]	PR[0]
Byte 2	AF[3]	AF[2]	AF[1]	AF[0]	DT[3]	DT[2]	DT[1]	DT[0]
Byte 3	-	-	CL	DL[4]	DL[3]	DL[2]	DL[1]	DL[0]

NEXT_SUBFRAME_MODE (SM)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	-	-	-	SM[4]	SM[3]	SM[2]	SM[1]	SM[0]

NEXT_CLOCK_GEAR (CG)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	-	-	-	-	CG[3]	CG[2]	CG[1]	CG[0]

NEXT_ROOT_FREQUENCY (RF)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	-	-	-	-	RF[3]	RF[2]	RF[1]	RF[0]

NEXT_DEFINE_CHANNEL (CN, TP, SD, SL)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	CN[7]	CN[6]	CN[5]	CN[4]	CN[3]	CN[2]	CN[1]	CN[0]
Byte 1	SD[7]	SD[6]	SD[5]	SD[4]	SD[3]	SD[2]	SD[1]	SD[0]
Byte 2	TP[3]	TP[2]	TP[1]	TP[0]	SD[11]	SD[10]	SD[9]	SD[8]
Byte 3	-	-	-	SL[4]	SL[3]	SL[2]	SL[1]	SL[0]

NEXT_DEFINE_CONTENT (CN, FL, PR, AF, DT, CL, DL)

	D7	D6	D5	D4	D3	D2	D1	D0
Byte 0	CN[7]	CN[6]	CN[5]	CN[4]	CN[3]	CN[2]	CN[1]	CN[0]
Byte 1	FL	PR[6]	PR[5]	PR[4]	PR[3]	PR[2]	PR[1]	PR[0]
Byte 2	AF[3]	AF[2]	AF[1]	AF[0]	DT[3]	DT[2]	DT[1]	DT[0]
Byte 3	-	-	CL	DL[4]	DL[3]	DL[2]	DL[1]	DL[0]

Success Criterion:

Verify that value of the Core Message Payload Reserved bits is always equal to 0.

Coverage:

Section	Statement
11	[message] Sources **shall** send zeros in all reserved Cells.

State Machine requirements:

State Machine Name: **PayloadResvBitError**

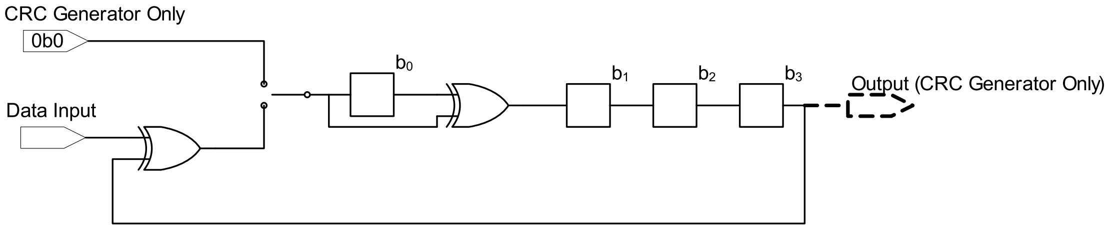
The state machine shall be resettable.

When interrogated, the state machine shall report the Core Message Code where errors occurred in message Payload reserved bits. The state machine is not reset by a valid message Payload reserved bit value. Only a reset can clear the state machine.

3.3.12. MS12 - Message Integrity Validity

Test Purpose: Verify that the value of Message Integrity CRC is valid

The Message Integrity CRC covers all the message bit fields excluding the Message Response. Though specific isolated requirements of the CRC generation are not observable as such through SLIMbus wires, the end result (the CRC value) is.



The formula to compute the CRC value is X4+X+1.

Success Criterion:

Verify that the broadcasted CRC value matches the computed CRC value.

Coverage:

Section	Statement
8.2.2	The [three reserved Cells] contents **shall** be used only for CRC generation.
8.2.4.1	Reserved Cells **shall** be included when performing Integrity CRC calculations.
8.2.4.1	The CRC **shall** be calculated in the following manner [include the paragraph description - redundant shall].
8.2.4.1	The CRC generator **shall** be initialized to 0b0000.
8.2.4.1	During each following Cell the CRC **shall** be updated according to the formula CRC = X4 + X + 1, as implemented in Figure 38 with the switch set to the XOR gate, or equivalent.

State Machine requirements:

State Machine Name: **MessageIntegrityError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the Message Integrity CRC. The state machine is not reset by a valid Message Integrity CRC. Only a reset can clear the state machine.

3.3.13. MS13 - Message Response Validity

Test Purpose: Verify that the value of Message Response is valid

There are 3 allowed values in the Message Response field:

- PACK (0b1010) = **P**ositive **ACK**nowledge
- NACK (0b1111) = **N**egative **ACK**nowledge
- NORE (0b0000) = **NO RE**sponse

All other values are Undefined (UDEF) and a signify an error condition. Note that it is formally impossible to detect which Device of the Component Under Test is actually issuing UDEF.

Success Criterion:

Verify that the broadcasted Message Response value is not an UDEF value.

Coverage:

Section	Statement
8.4.2.7	A destination Device **shall** not issue UDEF.

State Machine requirements:

State Machine Name: **MessageResponseError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the Message Response. The state machine is not reset by a valid Message Response. Only a reset can clear the state machine.

3.3.14. MS14 - Message Start Location

Test Purpose: Verify that the Message starts in an appropriate location

A message cannot start in any location of the Message Channel. The Message Channel must be idle (not currently having an on-going transmission) and a Message can only start in the first slot of a Slot Pair and in the first cell (C3) of that Slot.

Failure to start in the right cell will immediately lead to CRC check errors as the message will not be byte aligned, seen from other devices.

Starting on the second Slot of the Slot Pair will look like a valid Message structure. However, seen from the other devices, the first Arbitration Slot will be equal to 0, indicating an idle Message Channel so they will not try to decode the first byte. The second byte will be treated again as a start of arbitration, leading in most of the case to an invalid Arbitration Type. And all the device will loose their Message Synchronization.

Success Criterion:

Verify that the Message starts in the first Slot of a Slot Pair and in the first Cell (C3) of that Slot when the Message Channel is available (idle).

Coverage:

Section	Statement
8.4.1	A Device may initiate arbitration in the first Cell of any Message Channel Slot-pair when the channel is available. A Device **shall** not start arbitration in any other Cell, or while its Component has determined that the channel is unavailable.

State Machine requirements:

State Machine Name: **MessageStartError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the Message Start Location. The state machine is not reset by a valid Message Start Location. Only a reset can clear the state machine.

3.4. Message Protocol

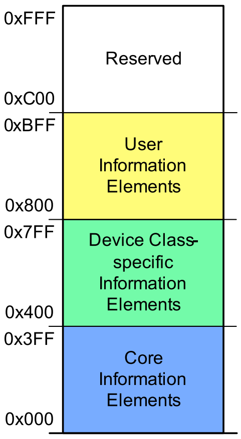
The Message protocol checking mainly verifies Message parameters validity and the Message sequence validity.

This section will cover some specification requirements of the chapters 8 and 10 of the SLIMbus specification.

3.4.1. MP1 - Information Element Address validity

Test Purpose: Verify the validity of the Element Code (EC) parameter

When accessing or writing to an Information Element, some addresses are Reserved. The range from 0xC00 to 0xFFF must not be used, irrespectively from the access method (byte-based or elemental).


0xFFF
Reserved
0xC00
0xBFF
User Information Elements
0x800
0x7FF
Device Class-specific Information Elements
0x400
0x3FF
Core Information Elements
0x000

Success Criterion:

Verify that the value of the Byte Address in the Element Code of the messages in the following list is not in the range 0xC00 to 0XFFF.

- REQUEST_INFORMATION
- REQUEST_CLEAR_INFORMATION
- CLEAR_INFORMATION
- REPORT_INFORMATION

Coverage:

Section	Statement
7.1	Byte Addresses in the range 0xC00 to 0xFFF are reserved. A Device **shall** not transmit a "Reserved" Byte Address.

State Machine requirements:

State Machine Name: **IEByteAddressError**

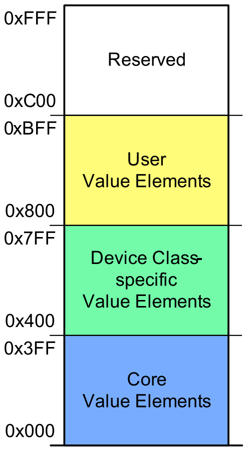
The state machine shall be resettable.

When interrogated, the state machine shall report the number of Bye Address range errors that occurred in the Message payload. The state machine is not reset by a valid address range. Only a reset can clear the state machine.

3.4.2. MP2 - Value Element Address validity

Test Purpose: Verify the validity of the Element Code (EC) parameter

When accessing or writing to a Value Element, some addresses are Reserved. The range from 0xC00 to 0xFFF must not be used.



Success Criterion:

Verify that the value of the Byte Address in the Element Code of the messages in the following list is not in the range 0xC00 to 0XFFF.

- REQUEST_VALUE
- REQUEST_CHANGE_INFORMATION
- CHANGE_INFORMATION

Coverage:

Section	Statement
7.2	Byte Addresses in the range 0xC00 to 0xFFF are reserved. A Device **shall** not transmit a "Reserved" Byte Address.

State Machine requirements:

State Machine Name: **VEByteAddressError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Bye Address range errors that occurred in the Message payload. The state machine is not reset by a valid address range. Only a reset can clear the state machine.

3.4.3. MP3 - Subframe Mode (SM) parameter Validity

Test Purpose: Verify the validity of the Subframe Mode (SM) parameter

The Active Manager must not send any Subframe Mode Reserved values in the NEXT_SUBFRAME_MODE(SM) message. The Reserved values are 0b00001 (1), 0b00010 (2), 0b00011 (3), 0b10100 (20) and 0b11110 (30).

Success Criterion:

Verify that the value of the parameter of the message "NEXT_SUBFRAME_MODE" is not equal to any of the reserved values listed in table 15 of the SLIMbus specification.

Coverage:

Section	Statement
6.3.1.4	The active Manager **shall** not configure the bus to use any "Reserved" Subframe Modes.
11.4.3	The Subframe Mode **shall** be encoded using the values in Table 15.
11.4.3	[SUBFRAME MODE] Reserved values **shall** not be used.

State Machine requirements:

State Machine Name: **SMParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Subframe Mode parameter errors that occurred in the NEXT_SUBFRAME_MODE Message payload. The state machine is not reset by a valid parameter. Only a reset can clear the state machine.

3.4.4. MP4 - Clock Gear (CG) parameter Validity

Test Purpose: Verify the validity of the Clock Gear (CG) parameter

The Active Manager must not send any Clock Gear Reserved values in the NEXT_CLOCK_GEAR(CG) message. The Reserved values are 0b1011 (11), 0b1100 (12), 0b1101 (13), 0b1110 (14) and 0b1111 (15).

Success Criterion:

Verify that the value of the parameter of the message "NEXT_CLOCK_GEAR" is not equal to any of the reserved values listed in table 16 of the SLIMbus specification.

Coverage:

Section	Statement
6.3.1.5	The active Manager **shall** not configure the bus to use any "Reserved" Clock Gear values.
11.4.4	The Clock Gear **shall** be encoded using the values in Table 16.
11.4.4	[CLOCK GEAR] Reserved values **shall** not be used.

State Machine requirements:

State Machine Name: **CGParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Clock Gear parameter errors that occurred in the NEXT_CLOCK_GEAR Message payload. The state machine is not reset by a valid parameter. Only a reset can clear the state machine.

3.4.5. MP5 - Root Frequency (RF) parameter Validity

Test Purpose: Verify the validity of the Root Frequency (RF) parameter

The Active Manager must not send any Root Frequency Reserved values in the NEXT_ROOT_FREQUENCY(RF) message. The Reserved values are 0b1010 to 0b1111.

Success Criterion:

Verify that the value of the parameter of the message "NEXT_ROOT_FREQUENCY" is not equal to any of the reserved values listed in table 17 of the SLIMbus specification.

Coverage:

Section	Statement
6.3.1.6	The active Manager **shall** not configure the bus to use any "Reserved" Root Frequency values.
11.4.4	The Root Frequency **shall** be encoded using the values in Table 17.
11.4.4	[ROOT FREQUENCY] Reserved values **shall** not be used.

State Machine requirements:

State Machine Name: **RFParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Root Frequency parameter errors that occurred in the NEXT_ROOT_FREQUENCY Message payload. The state machine is not reset by a valid parameter. Only a reset can clear the state machine.

3.4.6. MP6 - Segment Distribution (SD) parameter Validity

Test Purpose: Verify the validity of the Segment Distribution (SD) parameter

The Active Manager must not send any Segment Distribution Reserved values in the NEXT_DEFINE_CHANNEL(CN, TP, **SD**; SL) message. The Reserved values are 0b110000000001, 0b110000000000, 0b100000000000, 0b0b010000000000. Additionally to the Reserved value, the Segment Offset (S[n:0]) must not be equal to 0 (otherwise the data channel will collide with the Framing Channel) nor larger than or equal to the Segment Interval.

Success Criterion:

Verify that the value of the SD parameter of the message "NEXT_DEFINE_CHANEL" is not equal to any of the reserved values listed in table 20 of the SLIMbus specification. Verify that the Segment Offset is not equal to 0 nor larger than or equal to the Segment Interval.

Coverage:

Section	Statement
6.4.2	The Segment Offset **shall** be less than the Segment Interval.
6.4.2	A Device **shall** not send a reserved Segment Distribution code.

State Machine requirements:

State Machine Name: **SDParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Segment Distribution parameter errors that occurred in the NEXT_DEFINE_CHANNEL Message payload. The state machine is not reset by a valid SD parameter. Only a reset can clear the state machine.

3.4.7. MP7 - Segment Length (SL) parameter Validity

Test Purpose: Verify the validity of the Segment Length (SL) parameter

The Active Manager must not send any Segment Length Reserved values in the NEXT_DEFINE_CHANNEL(CN, TP, SD; **SL**) message. The Reserved value is equal to 0b00000 (null length).

Success Criterion:

Verify that the value of the Segment Length parameter of the message "NEXT_DEFINE_CHANEL" is not equal to any of the reserved values listed in table 19 of the SLIMbus specification.

Coverage:

Section	Statement
6.4.2	A Device **shall** not send a reserved Segment Length code.

State Machine requirements:

State Machine Name: **SLParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Segment Length parameter errors that occurred in the NEXT_DEFINE_CHANNEL Message payload. The state machine is not reset by a valid SL parameter. Only a reset can clear the state machine.

3.4.8. MP8 - Transport Protocol (TP) parameter Validity

Test Purpose: Verify the validity of the Transport Protocol (TP) parameter

The Active Manager must not send any Transport Protocol Reserved values in the NEXT_DEFINE_CHANNEL(CN, **TP**, SD; SL) message. The Reserved values are ranging from 8 to 13.

TP	Protocol Name	Type	# of TAG field Slots
0	Isochronous Protocol	Multicast	0
1	Pushed Protocol	Multicast	1
2	Pulled Protocol	Unicast	1
3	Locked Protocol	Multicast	0
4	Asynchronous Protocol – Simplex	Unicast	1
5	Asynchronous Protocol – Half-duplex	Unicast	1
6	Extended Asynchronous Protocol – Simplex	Unicast	2
7	Extended Asynchronous Protocol – Half-duplex	Unicast	2
8 to 13	**Reserved**	**-**	**-**
14	User Defined Protocol 1	-	1
15	User Defined Protocol 2	-	2

Success Criterion:

Verify that the value of the Transport Protocol parameter of the message "NEXT_DEFINE_CHANEL" is not equal to any of the reserved values listed in table 47 of the SLIMbus specification.

Coverage:

Section	Statement
9.3	"Reserved" values **shall** not be used when defining a channel.

State Machine requirements:

State Machine Name: **TPParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Transport Protocol parameter errors that occurred in the NEXT_DEFINE_CHANNEL Message payload. The state machine is not reset by a valid TP parameter. Only a reset can clear the state machine.

3.4.9. MP9 - Logical Address assignment Validity

Test Purpose: Verify the validity of the Logical Address parameter

The Active Manager must not send any Logical Address reserved values in the ASSIGN_LOGICAL_ADDRESS message or in the CHANGE_LOGICAL_ADDRESS message. Addresses in the range 0xF0 to 0xFE are Reserved. Logical Address 0xFF cannot be assigned as it is the default address for the Active Manager.

Success Criterion:

Verify that the value of the Logical Address (LA) parameter of the "ASSIGN_LOGICAL_ADDRESS" or "CHANGE_LOGICAL_ADDRESS" message is not equal to any of the following reserved values : 0xF0 to 0xFF.

Coverage:

Section	Statement
8.2.5.2	The logical address 0xFF **shall** not be assigned by the active Manager.
8.2.5.2	The logical addresses 0xF0 to 0xFE **shall** not be assigned by the Active Manager (reserved).

State Machine requirements:

State Machine Name: **LAParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Logical Address parameter errors that occurred in the ASSIGN_LOGICAL_ADDRESS or CHANGE_LOGICAL_ADDRESS Message payload. The state machine is not reset by a valid LA parameter. Only a reset can clear the state machine.

3.4.10. MP10 - AUX bit Format (AF) parameter Validity

Test Purpose: Verify the validity of the AUX bit Format (AF) parameter

The Active Manager must not send any AUX bit Format Reserved values in the NEXT_DEFINE_CONTENT(CN, FL, PR, **AF**, DT, CL, DL) message. Any Device must not send any AUX bit Format Reserved values in the CHANGE _CONTENT(CN, FL, PR, **AF**, DT, CL, DL) message. The Reserved values are 0b0010 to 0b1010 and 0b1100 to 0b1110.

Success Criterion:

Verify that the value of the AUX bit Format (AF) parameter of the "NEXT_DEFINE_CONTENT" or "CHANGE _CONTENT" messages is not equal to any of the reserved values listed in table 59 of the SLIMbus specification.

Coverage:

Section	Statement
9.4.3	Table 59 lists the AUX Field Format IDs that **shall** be used when defining a channel.
9.4.3	A Device **shall** not send a reserved AUX Field Format code when defining or updating the content of a Data Channel.

State Machine requirements:

State Machine Name: **AFParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of AUX bit Format parameter errors that occurred in the NEXT_DEFINE_CONTENT message payload or in the CHANGE_CONTENT message payload. The state machine is not reset by a valid AF parameter. Only a reset can clear the state machine.

3.4.11. MP11 - Data Type (DT) parameter Validity

Test Purpose: Verify the validity of the Data Type (DT) parameter

The Active Manager must not send any Data Type Reserved values in the NEXT_DEFINE_CONTENT(CN, FL, PR, AF, **DT**, CL, DL) message. Any Device must not send any Data Type Reserved values in the CHANGE _CONTENT(CN, FL, PR, AF, **DT**, CL, DL) message. The Reserved values are 0b0100 to 0b1111.

Success Criterion:

Verify that the value of the Data Type (DT) parameter of the "NEXT_DEFINE_CONTENT" or "CHANGE _CONTENT" messages is not equal to any of the reserved values listed in table 60 of the SLIMbus specification.

Coverage:

Section	Statement
9.5	The Data Type field identifies the DATA field content of a Segment and **shall** be coded as shown in Table 60.
9.5	A Device **shall** neither define nor update the content of a Data Channel using a "Reserved" Data Type.

State Machine requirements:

State Machine Name: **DTParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Data Type (DT) parameter errors that occurred in the NEXT_DEFINE_CONTENT message payload or in the CHANGE_CONTENT message payload. The state machine is not reset by a valid DT parameter. Only a reset can clear the state machine.

3.4.12. MP12 - Presence Rate (PR) parameter Validity

Test Purpose: Verify the validity of the Presence Rate (PR) parameter

The Active Manager must not send any Presence Rate Reserved values in the NEXT_DEFINE_CONTENT(CN, FL, **PR**, AF, DT, CL, DL) message. Any Device must not send any Presence Rate Reserved values in the CHANGE _CONTENT(CN, FL, **PR**, AF, DT, CL, DL) message. The Reserved values are 0b0011XXXX to 0b1111XXXX.

Success Criterion:

Verify that the value of the Presence Rate (PR) parameter of the "NEXT_DEFINE_CONTENT" or "CHANGE _CONTENT" messages is not equal to any of the reserved values listed in table 62 of the SLIMbus specification.

Coverage:

Section	Statement
9.6	A Device **shall** neither define nor update the content of a Data Channel using a "Reserved" Presence Rate.

State Machine requirements:

State Machine Name: **PRParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Presence Rate (PR) parameter errors that occurred in the NEXT_DEFINE_CONTENT message payload or in the CHANGE_CONTENT message payload. The state machine is not reset by a valid PR parameter. Only a reset can clear the state machine.

3.4.13. MP13 - Restart Time (RT) parameter Validity

Test Purpose: Verify the validity of the Restart Time (RT) parameter

The Active Manager must not send any Restart Time Reserved values in the NEXT_PAUSE_CLOCK (RT) message. The Reserved values are 0x03 to 0xFF.

Success Criterion:

Verify that the value of the Restart Time (RT) parameter of the "NEXT_PAUSE_CLOCK" message is not equal to any of the reserved values listed in table 66 of the SLIMbus specification.

Coverage:

Section	Statement
11.4.6	The Restart Time held in the Payload field **shall** be encoded using the values in Table 66.
11.4.6	[RESTART TIME] Reserved values **shall** not be used.

State Machine requirements:

State Machine Name: **RTParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of Restart Time (RT) parameter errors that occurred in the NEXT_PAUSE_CLOCK message payload. The state machine is not reset by a valid RT parameter. Only a reset can clear the state machine.

3.4.14. MP14 - Framer Handover Parameter Validity

Test Purpose: Verify that the parameters *NCo*, *NCi* are valid

Before proceeding to a Framer Handover, the active Manager must compute the appropriate *NCo* and *NCi* parameter values used in the Reconfiguration message NEXT_FRAMER_HANDOVER. These values must ensure that the handover operation will be glitch free and contention free on the clock line.

The recommended values for *NCo* and *NCi* are given by the following formulas:

NCi >= MAX(4, INT(4*Fi/Fo))

NCo >= INT(4*Fo/Fi)+1

With ***Fi*** the clock frequency of the incoming Framer and ***Fo*** the clock frequency of the outgoing Framer.

Success Criterion:

Verify that the value of the *NCo* and *NCi* parameters are fulfilling the conditions listed above.

Coverage:

Section	Statement
10.8	The active Manager **shall** determine the values for the Message parameters NCo and NCi such that there is no contention on the CLK line during the handover.

State Machine requirements:

State Machine Name: **FramerHandoverParameterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the parameter that failed: a value equal to 1 for *NCo* and a value equal to 2 for *NCi*. The state machine is not reset by a valid Source Address value. Only a reset can clear the state machine.

3.4.15. MP15 - Device Class Code Parameter validity

Test Purpose: Verify the validity of the Device Class code

When a Device is entering the Operational State, it must identify itself on the bus by sending to the active Manager a REPORT_PRESENT message. The parameter of the REPORT_PRESENT message are the Device Class code (DC) and the Device Class Version code (DCV).

The Device Class code can only be one of the code defined by the MIPI Alliance.

As per SLIMbus specification 1.01.01, the following class codes have been assigned:

- 0xFF: Manager
- 0xFE: Framer
- 0xFD: Interface
- 0x00: Generic

Other Class Codes will be added in the future.

Success Criterion:

Verify that the Device Class code (DC) provided in the REPROT_PRESENT payload is one of those listed above.

Coverage:

Section	Statement
12.1	Only a Device Class code assigned by the MIPI Alliance **shall** be used by a Device.

State Machine requirements:

State Machine Name: **DeviceClassCodeError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the REPORT_PRESENT Device Class code. The state machine is not reset by a valid Device Class code value. Only a reset can clear the state machine.

3.4.16. MP16 - Device Class Version Code Parameter validity

Test Purpose: Verify the validity of the Device Class Version code

When a Device is entering the Operational State, it must identify itself on the bus by sending to the active Manager a REPORT_PRESENT message. The parameter of the REPORT_PRESENT message are the Device Class code (DC) and the Device Class Version code (DCV).

The Device Class Version code value must always be greater than zero.

As per SLIMbus specification 1.01.01, the following class version codes have been assigned:

- 0x01: Manager
- 0x01: Framer
- 0x01: Interface
- 0x01: Generic

Class Version codes will probably be revised in the future.

Success Criterion:

Verify that the Device Class Version code (DCV) provided in the REPORT_PRESENT payload is greater than 0.

Coverage:

Section	Statement
12.1	The first version of a Device Class definition **shall** specify DCV equal to 0x01.

State Machine requirements:

State Machine Name: **DeviceClassVersionCodeError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the REPORT_PRESENT Device Class Version code. The state machine is not reset by a valid Device Class Version code value. Only a reset can clear the state machine.

3.5. Transport Protocols

The Transport protocol checking mainly verifies the flow control and flow information in the data channels.

This section will cover some specification requirements of the chapter 9 of the SLIMbus specification.

3.5.1. TP1 - Tag Bits Reserved value

Test Purpose: Verify that the value of the Reserved TAG bits is always 0

When using the Pushed or the Pulled protocol, the source device must set the value of the TAG slot reserved bits (C1 and C3) to 0.

Success Criterion:

Verify that the value of Reserved bits (C1 and C3) of the Pulled or Pushed transport protocol TAG slot is always equal to 0.

Coverage:

Section	Statement
9.3	TAG bits that are identified as "Reserved" **shall** be written as zeros.

State Machine requirements:

State Machine Name: **TAGResvBitError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors detected in the Reserved bit values of the TAG slot used in the Pushed or Pulled transport protocol. The state machine is not reset by a valid Reserved bit value. Only a reset can clear the state machine.

3.5.2. TP2 - Strobe Bit Usage

Test Purpose: Verify the validity of the Strobe bit value

With the Pushed transport protocol, when a value is present in the segment data field, the STR bit (and the P bit as well) must be equal to 1.

Note: The above assertion is only true if the empty segments are not used by other devices, for other purposes. Such a use is not described in the SLIMbus specification but is not forbidden.

Success Criterion:

When using the Pushed transport protocol, verify that a segment data field content not equal to zero corresponds to a Strobe (STR) bit value equal to 1.

Coverage:

Section	Statement
9.3.2	When the source does use the Segment, it **shall** set the STROBE bit.

State Machine requirements:

State Machine Name: **StrobeBitError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors detected in the Strobe bit values of the TAG slot used in the Pushed transport protocol. The state machine is not reset by a valid Strobe bit value. Only a reset can clear the state machine.

3.5.3. TP3 - Presence Bit Usage

Test Purpose: Verify the validity of the Presence bit value

With the Pushed or the Pulled transport protocol, when a value is present in the segment data field, the Presence (P) bit must be equal to 1.

With the Pushed Protocol, the P bit cannot be equal to 1 if the STR bit is equal to 0. With the Pulled Protocol, the P bit cannot be equal to 1 if the SRQ bit is equal to 0.

Note: The first assertion is only true if the empty segments are not used by other devices, for other purposes. Such a use is not described in the SLIMbus specification but is not forbidden.

Success Criterion:

When using the Pushed or the Pulled transport protocol, verify that a segment data field content that is not equal to zero corresponds to a Presence (P) bit value equal to 1.

When using the Pushed Protocol, verify that the Presence bit is not equal to 1 if the Strobe (STR) bit is equal to 0.

When using the Pulled Protocol, verify that the Presence bit is not equal to 1 if the Sample Request (SRQ) bit is equal to 0.

Coverage:

Section	Statement
9.3.2	When the source decides to send data in the DATA field, it **shall** set the P bit.
9.3.2	When the STROBE bit is equal to zero, the PRESENCE bit **shall** not be driven by the source and should not be read by a sink.
9.3.3	When SRQ=0, the source **shall** not drive the remaining Cells of the Segment following C1.

State Machine requirements:

State Machine Name: **PresenceBitError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors detected in the Presence bit values of the TAG slot used in the Pushed or Pulled transport protocol. The state machine is not reset by a valid Presence bit value. Only a reset can clear the state machine.

3.5.4. TP4 - Pushed Data Validity

Test Purpose: Verify the validity of the DATA and AUX field values when using the Pushed protocol

With the Pushed transport protocol, when the Strobe (STR) bit is equal to 1 and the Presence (P) bit is equal to 0, the content of the DATA field and the AUX field must have the value 0.

Success Criterion:

When using the Pushed transport protocol, verify that a segment DATA field and AUX field content is equal to zero when the Strobe (STR) bit is equal to 1 and the Presence (P) bit is equal to 0.

Coverage:

Section	Statement
9.3.2	If the STROBE bit is one, but the P bit is zero, the source **shall** drive the AUX field Slots and the DATA field Slots with zeros.

State Machine requirements:

State Machine Name: **PushedDataError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors detected in the DATA and AUX field values of the TAG slot used in the Pushed protocol. The state machine is not reset by a valid DATA or AUX field value. Only a reset can clear the state machine.

3.5.5. TP5 - Presence Bit Wide Band Jitter

Test Purpose: Verify the quality of the Presence (P) bit pattern

With the Pushed or the Pulled transport protocol, the Presence (P) bit pattern must have a wide band peak-to-peak jitter that is constrained. In other words, the unoccupied segments (with P=0) must be as equally spread as possible in order to minimize the FIFO size on both end of the link.

The jitter is a measure of the event timing variation with respect to an expected timing. In our case, the expected timing is the Presence period:

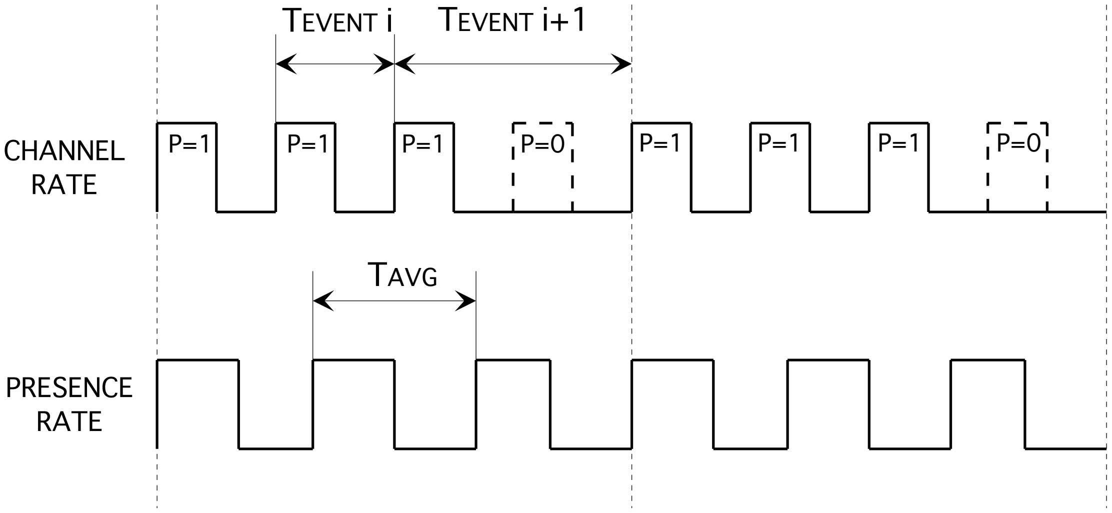
T_{AVG}=1/Presence Rate

The Event timing is the time elapsed between two occupied segments (P=1). Segments (and P bits) are transmitted with a rate equal to the channel rate. If we define ***n*** as being the number of consecutive unoccupied segments between two occupied segments, the event period is equal to:

T_{EVENT}= (1+***n***)/Channel Rate

Most of the time, as the Channel Rate is larger than the Presence Rate, T_{EVENT} is smaller than T_{AVG}, except when a sample (or more) is skipped (P=0).

The following figure shows an example where 3 samples are passed in 4 segments. Therefore, one segment is always empty every 4 segments.



As all the delays are cumulative, the Jitter function ***J*** is equal to:

$$J(N)=\sum_{i=1}^{N}\left(\frac{1+n_i}{CR}-\frac{1}{PR}\right)$$ with N the index of the Nth occupied segment.

The sum only applies to the occupied segments. CR is the Channel Rate, PR is the Presence Rate and $\boldsymbol{n_i}$ is the number of contiguous zeros between the $Pi|_{P=1}$ and the $Pi\text{-}1|_{P=1}$ events. The ***J*** function only provides values for the occupied segments having a Presence bit equal to one. Segments with a presence bit equal to 0 do not exist from a streaming point of view.

When the value J is multiplied by PR, it provides an image of the delay in terms of sample unit. When this value is constrained between 0 and 1, the optimal spread is achieved.

The figure of merit is actually the peak to peak measure, which is defined as the maximum value minus the minimum value.

Success Criterion:

The peak to peak ***J(N)*** value must always be less than 2/PR or, when multiplied by PR, less than 2 samples for any value of N.

Coverage:

Section	Statement
9.3.2	When the Pushed protocol is used to carry constant bit rate streams such as LPCM audio, the source **shall** constrain the pattern of occupied Segments, i.e. Segments with P=1, to avoid buffer overflow/underflow at the sink(s).
9.3.2	In steady-state operation, the source **shall** constrain the peak-to-peak wide band jitter of the occupied output Segments to less than two times the reciprocal of the stream's Presence Rate (section 9.6).
9.3.3	When the Pulled Protocol is used to carry constant bit rate streams such as LPCM audio, the sink **shall** constrain the pattern of Segment requests, i.e. Segments with SRQ=1, to avoid buffer overflow/underflow at the source.
9.3.3	In steady-state operation, the sink **shall** constrain the peak-to-peak wide band jitter of the Segment requests to less than two times the reciprocal of the stream's Presence Rate.

State Machine requirements:

State Machine Name: **PresenceBitJitterError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors detected in Presence Bit wide band peak-to-peak jitter value. The state machine is not reset by a valid jitter value. Only a reset can clear the state machine.

3.5.6. TP6 - P bit w.r.t. SRQ bit

Test Purpose: Verify that the P bit is set when the SRQ bit is set

With the Pulled transport protocol, when the Sample Request (SRQ) bit is equal to 1, source must provide a valid sample and therefore the Presence (P) bit must be equal to 1.

When the SRQ bit is 0, the source must not drive the rest of the segment and therefore the P bit value must be equal to 0.

Success Criterion:

When using the Pulled transport protocol, verify that the SRQ bit and the P bit have the same value.

Coverage:

Section	Statement
9.3.3	In steady-state operation, when SRQ=1 the source **shall** provide a valid sample in the current Segment.
9.3.3	Whenever it [the Data Source] provides a valid sample it **shall** set the P bit to indicate that the sample is present.
9.3.3	When SRQ=0, the source **shall** not drive the remaining Cells of the Segment following C1.
9.3.3	The Sink and Source **shall** be designed such that, in steady-state operation, every Segment with a sample request is serviced within the Segment.

State Machine requirements:

State Machine Name: **SRQAndPbitError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of detected mismatch between P and SRQ bits of the TAG slot used in the Pulled protocol. The state machine is not reset by matching values. Only a reset can clear the state machine.

3.5.7. TP7 - Asynchronous Protocol Flow control

Test Purpose: Verify that the data flow is interrupted when CTS=0

When using the Asynchronous transport protocol, the data flow is regulated by the CTS tag bit. When the CTS value becomes a valid 0 or becomes invalid, the data flow must be interrupted in the next segment. When CTS value is steadily a valid 0, the data flow shall stay idle.

A contrario, a CTS value being a valid 1 does not mandate the presence of any data in the segment. Therefore, only the flow interruption can be tested.

Success Criterion:

Verify that the data flow is interrupted (P=0 and DATA+AUX=0) when CTS transitions from valid 1 to invalid.

Verify that the flow is off when CTS is steadily equal to valid 0 or CTS transitions from valid 0 to invalid.

Coverage:

Section	Statement
9.3.5	CTS=0 indicates that the source Device **shall** stop sending data
9.3.6	Clear To Send: CTS=0 indicates that the source **shall** stop transmitting data

State Machine requirements:

State Machine Name: **FlowControlError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of detected violation in the flow control. The state machine is not reset by a correct flow control behavior. Only a reset can clear the state machine.

3.6. Bus Processes

The Bus Process checking mainly verifies the sequence of messages.

This section will cover some specification requirements of the chapter 10 of the SLIMbus specification.

3.6.1. BP1 - Reconfiguration Message Source Address

Test Purpose: Verify that the Reconfiguration Messages are sent by the Active Manager only

All the messages involved in a reconfiguration sequence must be sent by the Active Manager only (source address is equal to 0xFF). The Reconfiguration messages are: BEGIN_RECONFIGURATION, RECONFIGURE_NOW and all the NEXT_ messages.

Success Criterion:

Verify that the value of the source address of a Reconfiguration Message is equal to 0xFF.

Coverage:

Section	Statement
11.4.1	[BEGIN_RECONFIGURATION] This Message **shall** be sent only by the active Manager.
11.4.2	[NEXT_ACTIVE_FRAMER] This Message **shall** only be sent by the active Manager.
11.4.3	[NEXT_SUBFRAME_MODE] This Message **shall** be sent only by the active Manager.
11.4.4	[NEXT_CLOCK_GEAR] This Message **shall** be sent only by the active Manager.
11.4.5	[NEXT_ROOT_FREQUENCY] This Message **shall** be sent only by the active Manager.
11.4.6	[NEXT_PAUSE_CLOCK] This Message **shall** be sent only by the active Manager.
11.4.7	[NEXT_RESET_BUS] This Message **shall** be sent only by the active Manager.
11.4.8	[NEXT_SHUTDOWN_BUS] This Message **shall** be sent only by the active Manager.
11.4.9	[NEXT_DEFINE_CHANNEL] This Message **shall** be sent only by the active Manager.
11.4.10	[NEXT_DEFINE_CONTENT] This Message **shall** be sent only by the active Manager.
11.4.11	[NEXT_ACTIVATE_CHANNEL] This Message **shall** be sent only by the active Manager.
11.4.12	[NEXT_DEACTIVATE_CHANNEL] This Message **shall** be sent only by the active Manager.
11.4.13	[NEXT_REMOVE_CHANNEL] This Message **shall** be sent only by the active Manager.
11.4.14	[RECONFIGURE_NOW] This Message **shall** be sent only by the active Manager.

State Machine requirements:

State Machine Name: **ReconfigSourceAddressError**

The state machine shall be resettable.

When interrogated, the state machine shall report the last Message Code where an error has been detected in Source Address. The state machine is not reset by a valid Source Address value. Only a reset can clear the state machine.

3.6.2. BP2 - Reconfiguration Sequence Validity

Test Purpose: Verify that the Reconfiguration Messages are sent in a valid sequence

A valid Reconfiguration sequence has the following general form:

- BEGIN_RECONFIGURATION (mandatory)
- NEXT_ (optional)
- BEGIN_RECONFIGURATION (option - abort previous NEXT_ messages)
- NEXT_ (optional)
- RECONFIGURE_NOW (mandatory)

The following sequences are erroneous:

- NEXT_ message without a valid BEGIN_RECONFIGURATION message
- RECONFIGURE_NOW without a valid BEGIN_RECONFIGURATION message

Success Criterion:

Verify that there has been a valid BEGIN_RECONFIGURATION message (with a PACK message response) before any other Reconfiguration message is sent.

Coverage:

Section	Statement
10.3.1	The active Manager **shall** indicate the start of a Reconfiguration Sequence by broadcasting the BEGIN_RECONFIGURATION Message.
10.3.1	No Reconfiguration Messages **shall** be transmitted without first ending the BEGIN_RECONFIGURATION Message.

State Machine requirements:

State Machine Name: **ReconfigSequenceError**

The state machine shall be resettable.

When interrogated, the state machine shall report the last Reconfiguration Message Code where an error has been detected in the Reconfiguration Sequence Validity. The state machine is not reset by a valid Reconfiguration Sequence. Only a reset can clear the state machine.

3.6.3. BP3 - Reconfiguration Sequence Start

Test Purpose: Verify that a Reconfiguration Sequence always start after a Reconfiguration Boundary

When a RECONFIGURE_NOW message finishes a valid Reconfiguration Sequence, no BEGIN_RECONFIGURATION message can be sent before the Reconfiguration Boundary has happened.

Success Criterion:

Verify that the Active Manager does not send any BEGIN_RECONFIGURATION message between the end of a valid RECONFIGURE_NOW message and the corresponding Reconfiguration Boundary.

Coverage:

Section	Statement
10.3.1.4	After the RECONFIGURE_NOW Message is sent, the active Manager **shall** not broadcast any Reconfiguration Messages until after the Reconfiguration Boundary

State Machine requirements:

State Machine Name: **ReconfigSequenceStartError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors in the Reconfiguration Sequence start. The state machine is not reset by a valid Reconfiguration Sequence start. Only a reset can clear the state machine.

3.6.4. BP4 - Reconfiguration Message Destination Type Validity

Test Purpose: Verify that all the Reconfiguration Messages have a valid Destination Type

All the Reconfiguration Messages must be broadcasted (having all the devices as destination). The Destination Type value must be equal to 0b11.

Success Criterion:

Verify that the Active Manager sets the Destination Type of all the Reconfiguration messages to value 0b11 (3).

Coverage:

Section	Statement
10.3.1	The active Manager **shall** indicate the start of a Reconfiguration Sequence by **broadcasting** the BEGIN_RECONFIGURATION Message.

All the Reconfiguration Messages have the following sentence in Section 11: “This Message uses a Short Arbitration Field and a Broadcast Header field”

State Machine requirements:

State Machine Name: **ReconfigMsgBroadcastError**

The state machine shall be resettable.

When interrogated, the state machine shall report the last Reconfiguration Message Code where an error has been detected in the Destination Type Validity. The state machine is not reset by a valid Destination Type. Only a reset can clear the state machine.

3.6.5. BP5 - Logical Address assignment by Active Manager

Test Purpose: Verify that the Logical Address assignment is done only by Manager

The Active Manager is the only Device allowed to manage the Logical addresses used on the bus.

Success Criterion:

Verify that the value of the source address of a ASSIGN_LOGICAL_ADDRESS message or of a CHANGE_LOGICAL_ADDRESS message is equal to 0xFF.

Coverage:

Section	Statement
10.10	As already stated in section 10.9, the active Manager **shall** be responsible for the assignment of Logical Addresses.
10.10	The ASSIGN_LOGICAL_ADDRESS Message **shall** be used by the active Manager to assign a Logical Address to a Device.
10.10	The CHANGE_LOGICAL_ADDRESS Message **shall** be used by the active Manager to inform a Device of a change in the Logical Address [that it shall use during the Short arbitration sequence of all outgoing Messages].
11.1.2	This Message [ASSIGN_LOGICAL_ADDRESS] **shall** be sent only by the active Manager.
11.1.4	This Message [CHANGE_LOGICAL_ADDRESS] **shall** be sent only by the active Manager.

State Machine requirements:

State Machine Name: **LAAssigmentError**

The state machine shall be resettable.

When interrogated, the state machine shall report the last Logical Address Message Code where an error has been detected in Source Address. The state machine is not reset by a valid Source Address value. Only a reset can clear the state machine.

3.6.6. BP6 - Logical Address Uniqueness

Test Purpose: Verify the uniqueness of the Logical Address assignment

The Manager is the only SLIMbus entity allowed assign Logical Addresses to the SLIMbus devices. To each Device corresponds a unique Logical Address. Note that from one run to another run, the couple (EA; LA) can differ but two (or more) devices cannot get the same Logical Address.

Note: The SLIMbus specification allows Devices with same EA to enumerate on turn (not concurrently). That means that a given EA can receive more than one LA.

Success Criterion:

Verify that the Manager does not allocate the same Logical Address to different Devices.

Coverage:

Section	Statement
10.10	The active Manager **shall** ensure that Logical Addresses are unique at bus level so that no two Devices try to use the same Logical Address at the same time.

State Machine requirements:

State Machine Name: **LAUniquenessError**

The state machine shall be resettable.

Build a list of the LA assigned by the Manager (by using the ASSIGN_LOGICAL_ADDRESS or CHANGE_LOGICAL ADDRESS message). Verify that the Manager does not assign twice or more the same LA.

When interrogated, the state machine shall report the last Logical Address value wrongly assigned by the Manager. The state machine is not reset by a valid Logical Address value assignment. Only a reset can clear the state machine.

3.6.7. BP7 - Active Manager Logical Address

Test Purpose: Verify that the active Manager uses Logical Address 0xFF

The active Manager gets assigned automatically the Logical Address 0xFF and nothing else. This Logical Address must be used as source address in all the messages that can be sent only by the active Manager.

Msg Code	Message Description
0x02	ASSIGN_LOGICAL_ADDRESS (LA)
0x04	RESET_DEVICE ()
0x08	CHANGE_LOGICAL_ADDRESS (LA)
0x09	CHANGE_ARBITRATION_PRIORITY (AP)
0x0C	REQUEST_SELF_ANNOUNCEMENT ()
0x40	BEGIN_RECONFIGURATION ()
0x44	NEXT_ACTIVE_FRAMER (LAIF, NCo, NCi)
0x45	NEXT_SUBFRAME_MODE (M)
0x46	NEXT_CLOCK_GEAR (G)
0x47	NEXT_ROOT_FREQUENCY (F)
0x4A	NEXT_PAUSE_CLOCK (RT)
0x4B	NEXT_RESET_BUS ()
0x4C	NEXT_SHUTDOWN_BUS ()
0x50	NEXT_DEFINE_CHANNEL (CN, TP, SD, SL)
0x51	NEXT_DEFINE_CONTENT (CN, FL, PR, AF, DT, CL, DL)
0x54	NEXT_ACTIVATE_CHANNEL (CN)
0x55	NEXT_DEACTIVATE_CHANNEL (CN)
0x58	NEXT_REMOVE_CHANNEL (CN)
0x5F	RECONFIGURE_NOW ()

Success Criterion:

Verify that all the messages that can be sent by the active Manager use 0xFF as source address.

Coverage:

Section	Statement
8.2.5.2	The logical address 0xFF **shall** be assigned by default to the Active Manager.

State Machine requirements:

State Machine Name: **ManagerAddressError**

The state machine shall be resettable.

When interrogated, the state machine shall report the last Message Code where the source address was erroneous. The state machine is not reset by a valid Logical Address value. Only a reset can clear the state machine.

3.6.8. BP8 - Manager Assigned Core Messages

Test Purpose: Verify that only the Manager is sending some defined Core Messages

The Manager is the only SLIMbus entity allowed to send some of the Core Messages.

Success Criterion:

Verify that the value of the source address of a RESET_DEVICE message, CHANGE_ARBITRATION_PRIORITY message or of a REQUEST_SELF_ANNOUNCEMENT message is equal to 0xFF.

Coverage:

Section	Statement
11.1.3	This Message [RESET_DEVICE] **shall** be sent only by the active Manager.
11.1.5	This Message [CHANGE_ARBITRATION_PRORITY] **shall** be sent only by the active Manager.
11.1.6	This Message [REQUEST_SELF_ANNOUCEMENT] **shall** be sent only by the active Manager.

State Machine requirements:

State Machine Name: **CoreMsgSourceAddressError**

The state machine shall be resettable.

When interrogated, the state machine shall report the last Message Code where an error has been detected in Source Address. The state machine is not reset by a valid Source Address value. Only a reset can clear the state machine.

3.6.9. BP9 - Manager Does Not Report Present

Test Purpose: Verify that the active Manager does not report present

The active Manager is the only Core Device that gets assigned automatically a Logical Address (0xFF). Therefore, the active Manager Device must not send a REPORT_PRESENT message.

Success Criterion:

Verify that Enumeration Address of the active Manager does not appear as the source address of a REPORT_PRESENT message.

Coverage:

Section	Statement
10.9.2	Following the completion of the boot process for its Component, the active Manager **shall** bypass the TryingToAnnounce and WaitingForLA states shown in Figure 77.

State Machine requirements:

State Machine Name: **ManagerReportPresentError**

The state machine shall be resettable.

When interrogated, the state machine shall report a value equal to 1 if a REPORT_PRESENT message has been sent by the active Manager, 0 else. Only a reset can clear the state machine.

3.6.10. BP10 - Report Present to non Manager device

Test Purpose: Verify that the REPORT_PRESENT message is always aimed at the active Manager

When a Device is entering the Operational State, it must identify itself on the bus by sending to the active Manager a REPORT_PRESENT message. Only the active Manager can be the destination of this message, not any other devices.

Success Criterion:

Verify that the Destination Address of the REPORT_PRESENT message is always equal to 0xFF (the LA of the active Manager).

Coverage:

Section	Statement
11.1.1	This Message [REPORT_PRESENT] **shall** be sent only to the active Manager.

State Machine requirements:

State Machine Name: **ReportPresentDestAddrError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the REPORT_PRESENT Destination Address. The state machine is not reset by a valid Destination Address. Only a reset can clear the state machine.

3.6.11. BP11 - Report Absent to non Manager device

Test Purpose: Verify that the REPORT_ABSENT message is always aimed at the active Manager

When a Device is exiting bus operation, it may send to the active Manager a REPORT_ABSENT message. Only the active Manager can be the destination of this message, not any other devices.

Success Criterion:

Verify that the Destination Address of the REPORT_ABSENT message is always equal to 0xFF (the LA of the active Manager).

Coverage:

Section	Statement
11.1.7	This Message [REPORT_ABSENT] **shall** be sent only to the active Manager.

State Machine requirements:

State Machine Name: **ReportAbsentDestAddrError**

The state machine shall be resettable.

When interrogated, the state machine shall report the number of errors that occurred in the REPORT_ABSENT Destination Address. The state machine is not reset by a valid Destination Address. Only a reset can clear the state machine.

3.6.12. BP12 - Data Channel Overlap

Test Purpose: Verify that the Data Channels are not overlapping

Data channels are built on a set of contiguous slots (called segment) repeating with a constant interval. The Channel offset and repeat interval are defined by the Segment Distribution (SD) parameter of the NEXT_DEFINE_CHANNEL message.

The active Manager must compute all the SD parameters in such a way that none of the defined data channels will have common slots.

Success Criterion:

Record all the SD parameters and verify that none of them is leading to overlapping data channels.

Coverage:

Section	Statement
6.4.2	A Slot **shall** be allocated to only one Data Channel at a time.
6.4.2	Data Channels **shall** not overlap.

State Machine requirements:

State Machine Name: **DataChannelOverlapError**

The state machine shall be resettable.

When interrogated, the state machine shall report the ID of the channel that is causing data channel slot overlap. The state machine is not reset by a valid Segment Distribution value. Only a reset can clear the state machine.

3.6.13. BP13 - Data Channel and Control Space Overlap

Test Purpose: Verify that a Data Channel is not overlapping with the control space

Data channels are built on a set of contiguous slots (called segment) repeating with a constant interval. The Channel offset and repeat interval are defined by the Segment Distribution (SD) parameter of the NEXT_DEFINE_CHANNEL message.

The active Manager must compute all the SD parameters in such a way that none of the defined data channels is overlapping with the control space. This is not a requirement as such in the SLIMbus specification but is treated as an error case by the Components. It is therefore not a desired behavior.

Success Criterion:

Record all the SD parameters and verify that none of them is leading to overlapping data channels.

Coverage:

Section	Statement
6.4.2	A Component with an output Port **shall** not allow that Port to drive the DATA line during Slots allocated to Control Space, even if the Component receives a Data Channel definition that overlaps Control Space.
10.6.1.2	If the new definition of the Message Channel overlaps with existing Data Channels, the active Manager **shall** move or cancel the Data Channels before it broadcasts the RECONFIGURE_NOW Message.
10.6.1.3	In this case [conflicts between data channels and Message channel] the Component in which this conflict occurs **shall** give the Message Channel priority over the Data Channel.

State Machine requirements:

State Machine Name: **DataChannelOverlapMCError**

The state machine shall be resettable.

When interrogated, the state machine shall report the ID of the channel that is overlapping with the control space. The state machine is not reset by a valid Segment Distribution value. Only a reset can clear the state machine.

3.6.14. BP14 - Data Channel Source Uniqueness

Test Purpose: Verify that a Data Channel has only one data source

Data channels have one data source at a time. They might have multiple data sinks but only one data source. This is not dependent on the transport protocol in use.

Note: The data channel source might change over the lifetime of the data channel. A DISCONNECT_PORT message followed by a CONNECT_SOURCE message (and the necessary channel definition reconfiguration sequence) will proceed to a data source handover.

Success Criterion:

Record all CONNECT_SOURCE messages. Update the look-up table with the relevant DISCONNECT_PORT messages. Verify that a given channel does only get connected one and one only data source at a time.

Coverage:

Section	Statement
10.13.1.1	For the Isochronous, Pushed, Pulled and Locked Transport Protocols, the Data Channel **shall** have exactly one Port associated to it by the CONNECT_SOURCE Message.
10.13.1.1	For the Asynchronous and Extended Asynchronous Transport Protocols, the transport **shall** have exactly one primary owner associated to it by the CONNECT_SOURCE Message, and one secondary owner associated to it by the CONNECT_SINK Message.

State Machine requirements:

State Machine Name: **DataChannelSourceUniquenessError**

The state machine shall be resettable.

When interrogated, the state machine shall report the ID of the channel that is getting assigned more than one data source plus 1 (ID+1), 0 else. Only a reset can clear the state machine.

3.6.15. BP15 - Data Channel Sink Uniqueness

Test Purpose: Verify that a Data Channel has only one data sink

Data channels using flow controlled data transmission can have one data source at a time. These channels are using the Pulled protocol and all flavors of the Asynchronous protocol. The table 47 in the SLIMbus specification shows these protocols as being unicast (means point to point: one source, one sink).

TP	Protocol Name	Type	# of TAG field Slots
0	Isochronous Protocol	Multicast	0
1	Pushed Protocol	Multicast	1
2	Pulled Protocol	**Unicast**	1
3	Locked Protocol	Multicast	0
4	Asynchronous Protocol – Simplex	**Unicast**	1
5	Asynchronous Protocol – Half-duplex	**Unicast**	1
6	Extended Asynchronous Protocol – Simplex	**Unicast**	2
7	Extended Asynchronous Protocol – Half-duplex	**Unicast**	2
8 to 13	Reserved	-	-
14	User Defined Protocol 1	-	1
15	User Defined Protocol 2	-	2

Note: The data channel ink might change over the lifetime of the data channel. A DISCONNECT_PORT message followed by a CONNECT_SINK message (and the necessary channel definition reconfiguration sequence) will proceed to a data sink handover.

Success Criterion:

Record all CONNECT_SINK messages. Update the look-up table with the relevant DISCONNECT_PORT messages. Verify that a given channel using the Pulled protocol or one of the Asynchronous protocols does only get connected one and one only data sink at a time.

Coverage:

Section	Statement
10.13.1.1	For the Asynchronous and Extended Asynchronous Transport Protocols, the transport **shall** have exactly one primary owner associated to it by the CONNECT_SOURCE Message, and one secondary owner associated to it by the CONNECT_SINK Message.

State Machine requirements:

State Machine Name: **DataChannelSourceUniquenessError**

The state machine shall be resettable.

When interrogated, the state machine shall report the ID of the channel that is getting assigned more than one data sink plus 1 (ID+1), 0 else. Only a reset can clear the state machine.

3.6.16. BP16 - Interface Device Class Allowed Tx Message

Test Purpose: Verify that the Interface Device does not transmit other messages than the ones allowed by its class description

The Interface device must not transmit any other messages than the ones listed in the Interface Device Core Messages table (mandatory and optional).

Mandatory	Optional
REPLY_INFORMATION	CHANGE_CONTENT
REPORT_INFORMATION	CHANGE_VALUE
REPORT_PRESENT	CLEAR_INFORMATION
	REPLY_VALUE
	REPORT_ABSENT
	REQUEST_CHANGE_VALUE
	REQUEST_CLEAR_INFORMATION
	REQUEST_INFORMATION
	REQUEST_VALUE

Success Criterion:

Verify that the messages transmitted by the Interface Device are allowed.

Coverage:

Section	Statement
12.1	The Device **shall** not transmit any other Source-referred Class specific Message

State Machine requirements:

State Machine Name: **InterfaceTxCoreMsgError**

The state machine shall be resettable.

When interrogated, the state machine shall report the message code of the "not allowed" message transmitted by the Interface Device. Only a reset can clear the state machine.

3.6.17. BP17 - Manager Device Class Allowed Tx Message

Test Purpose: Verify that the Manager Device does not transmit other messages than the ones allowed by its class description

The Manager device must not transmit any other messages than the ones listed in the Manager Device Core Messages table (mandatory and optional).

Mandatory	Optional
ASSIGN_LOGICAL_ADDRESS	CHANGE_ARBITRATION_PRIORITY
BEGIN_RECONFIGURATION	CHANGE_CONTENT
CHANGE_LOGICAL_ADDRESS	CHANGE_VALUE
CLEAR_INFORMATION	NEXT_PAUSE_CLOCK
CONNECT_SINK	NEXT_SHUTDOWN_BUS
CONNECT_SOURCE	REPLY_VALUE
DISCONNECT_PORT	REPORT_ABSENT
NEXT_ACTIVATE_CHANNEL	REPORT_INFORMATION
NEXT_ACTIVE_FRAMER	REQUEST_CHANGE_VALUE
NEXT_CLOCK_GEAR	REQUEST_SELF_ANNOUNCEMENT
NEXT_DEACTIVATE_CHANNEL	REQUEST_VALUE
NEXT_DEFINE_CHANNEL	
NEXT_DEFINE_CONTENT	
NEXT_REMOVE_CHANNEL	
NEXT_RESET_BUS	
NEXT_ROOT_FREQUENCY	
NEXT_SUBFRAME_MODE	
RECONFIGURE_NOW	
REPLY_INFORMATION	
REPORT_PRESENT	
REQUEST_CLEAR_INFORMATION	
REQUEST_INFORMATION	
RESET_DEVICE	

Success Criterion:

Verify that the messages transmitted by the Manager Device are allowed.

Coverage:

Section	Statement
12.1	The Device **shall** not transmit any other Source-referred Class specific Message

State Machine requirements:

State Machine Name: **ManagerTxCoreMsgError**

The state machine shall be resettable.

When interrogated, the state machine shall report the message code of the “not allowed” message transmitted by the Manager Device. Only a reset can clear the state machine.

3.6.18. BP18 - Framer Device Class Allowed Tx Message

Test Purpose: Verify that the Framer Device does not transmit other messages than the ones allowed by its class description

The Framer device must not transmit any other messages than the ones listed in the Framer Device Core Messages table (mandatory and optional).

Mandatory	Optional
REPLY_INFORMATION REPORT_PRESENT	CHANGE_CONTENT CHANGE_VALUE CLEAR_INFORMATION REPLY_VALUE REPORT_ABSENT REPORT_INFORMATION REQUEST_CHANGE_VALUE REQUEST_CLEAR_INFORMATION REQUEST_INFORMATION REQUEST_VALUE

Success Criterion:

Verify that the messages transmitted by the Framer Device are allowed.

Coverage:

Section	Statement
12.1	The Device **shall** not transmit any other Source-referred Class specific Message

State Machine requirements:

State Machine Name: **FramerTxCoreMsgError**

The state machine shall be resettable.

When interrogated, the state machine shall report the message code of the "not allowed" message transmitted by the Framer Device. Only a reset can clear the state machine.

3.6.19. BP19 - Generic Device Class Allowed Tx Message

Test Purpose: Verify that the Generic Device does not transmit other messages than the ones allowed by its class description

The Generic device must not transmit any other messages than the ones listed in the Generic Device Core Messages table (mandatory and optional).

Mandatory	Optional
REPLY_INFORMATION REPORT_PRESENT	CHANGE_CONTENT CHANGE_VALUE CLEAR_INFORMATION REPLY_VALUE REPORT_ABSENT REPORT_INFORMATION REQUEST_CHANGE_VALUE REQUEST_CLEAR_INFORMATION REQUEST_INFORMATION REQUEST_VALUE

Success Criterion:

Verify that the messages transmitted by the Framer Device are allowed.

Coverage:

Section	Statement
12.1	The Device **shall** not transmit any other Source-referred Class specific Message

State Machine requirements:

State Machine Name: **GenericTxCoreMsgError**

The state machine shall be resettable.

When interrogated, the state machine shall report the message code of the "not allowed" message transmitted by the Generic Device. Only a reset can clear the state machine.

4. Natural flow of events

In this section, we will bring a component in the operational state and proceed to all the applicable bus management operation.

On top of the verification listed in the *Success Criterion* section of the test, all the checks described in the Continuous Monitoring chapter shall also be performed. Any failure of one or more of these checks signifies a test failure.

4.1. Boot

The requirements for the boot sequence depend on the type of component to test. The Clock Sourcing Components will generate the boot sequence through the Framer device while the Clock Receiving Components will use the boot sequence to reach the operational state.

4.1.1. Clock Sourcing Component Boot Sequence

The Framer (and therefore the Clock Sourcing Component) is an essential part of SLIMbus. Any deviation from the SLIMbus specification will very likely lead to a non-functional system.

The boot process of the Framer is split in 4 phases and an initial state:

- *Undefined* state
- *CheckingDataLine*
- *StartingClock*
- *StartingData*
- *Booted*

Each of these states / phases has own set of requirements and will therefore get one or many specific tests.

The tests are organized in such a way that they can be played sequentially. Any failure to a test will cause the test sequence to stop, as it will result from major design issues.

4.1.1.1. CSB1 - Contention during the *CheckingDataLine* state

Test Description:

Verify that the active Framer under test will generate the proper boot sequence.

Initial conditions:

The SLIMbus must be idle. There must not be any activity on the clock line and the data line. The clock line and data line must be contention free.

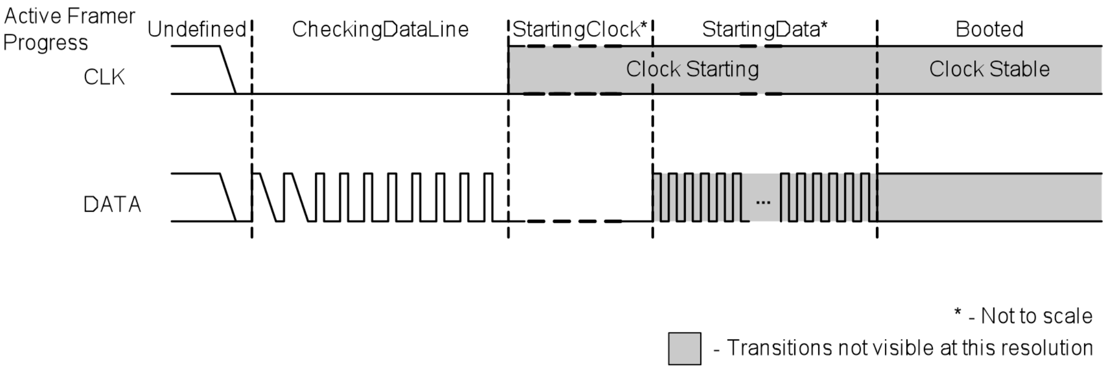
Stimulus:

Attach the data line to ground (GND) via a resistor of 1kΩ.

Trigger the framer boot sequence. The method is unfortunately device dependent (power on, software control, hardware pin control, …). The Framer vendor must provide a simple start method to run the test.

Success Criterion:

Because of the resistor of 1kΩ pulling down the level of the data line to GND when left to the bus hold, the framer will observe unexpected data transitions. Verify that the framer never moves to the *StartingClock* state and never drives the clock line. The minimum amount of time to search for a clock transition is equal to the duration of 9 frames.



Coverage:

Section	Statement
10.1	Clock Sourcing Components **shall** synchronize according to sections 10.1.3.
10.1.1.2	The active Framer **shall** not move to the StartingClock state until there have been eight consecutive Frames with no DATA line-check failures.

4.1.1.2. CSB2 - *CheckingDataLine* state

Test Description:

Verify that the active Framer under test will generate the proper boot sequence.

Initial conditions:

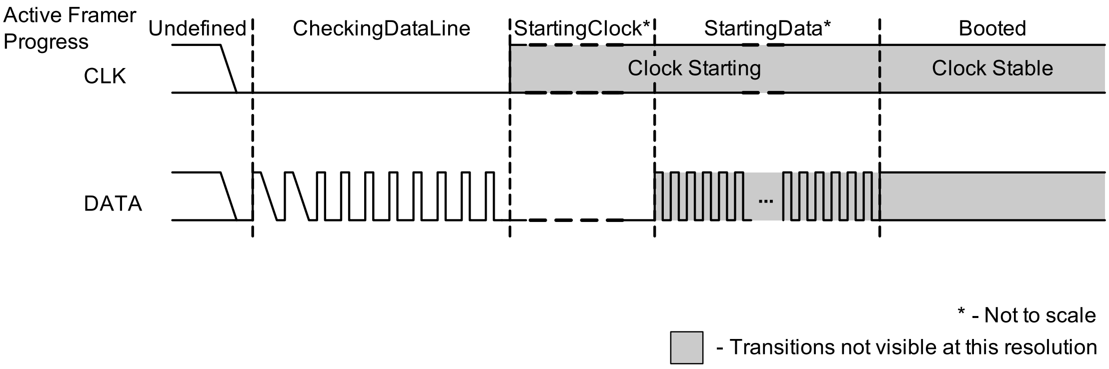
The SLIMbus must be idle. There must not be any activity on the clock line and the data line. The clock line and data line must be contention free.

Stimulus:

Trigger the framer boot sequence. The method is unfortunately device dependent (power on, software control, hardware pin control, ...). The Framer vendor must provide a simple start method to run the test.

Success Criterion:

During the *CheckingDataLine* phase, the clock line stays below VOL (no transitions). The Framer under test writes during 8 consecutive frames the Guide Byte and the framing information.



Verify that the Framer writes:

- 8 times the bit pattern 0b1011 representing the frame sync symbol.
- 8 times a 4 bit pattern that is likely be not recurring and that happens right in between 2 frame-sync symbols.
- A Guide Byte with GV=0, ENT=0 and parity check bits equal to 1. The Guide byte value is equal to 0b00000011, eventually split into 2 non-contiguous slots, depending on the initial value of the subframe mode used at boot time by the Framer (provided in the Framer data sheet).

Coverage:

Section	Statement
10.1	The active Framer **shall** boot according to section 10.1.1.
10.1.1	During the boot process, the active Framer **shall** progress through five different states: Undefined, CheckingDataLine, StartingClock, StartingData, and Booted.
10.1.1.2	In the CheckingDataLine state the active Framer **shall** continuously drive the CLK line below VOL and transmit the Framing Channel and the Guide Channel on the DATA line using its internal clock for timing.
10.1.1.2	The active Framer **shall** not move to the StartingClock state until there have been eight consecutive Frames with no DATA line-check failures.

4.1.1.3. CSB3 - *StartingClock* state

Test Description:

Verify that the active Framer under test will generate the proper boot sequence.

Initial conditions:

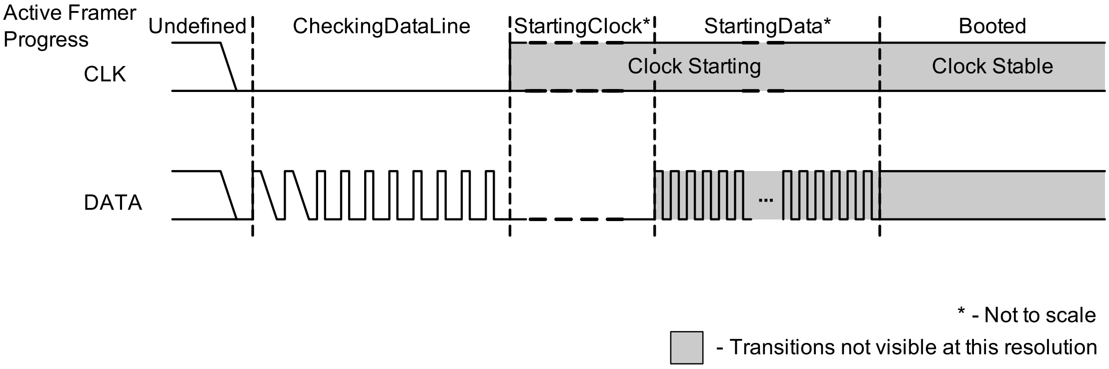
The active Framer successfully passed the *CheckingDataLine* tests and is entering the *StartingClock* state.

Stimulus:

None.

Success Criterion:

In the *StartingClock* phase, the active Framer toggles the clock line 3072 times while not driving any information on the data line.



Verify that there is not any transition on the data line during 3072 clock cycles.

Verify that there is a data line transition at the 3073rd clock cycle, indicating a transition to the *StartingData* state.

Coverage:

Section	Statement
10.1.1.3	In the *StartingClock* state, the active Framer **shall** start clocking (toggling) the CLK line.
10.1.1.3	The Clock Sourcing Component **shall** refrain from sending any data (including the Framing Channel) on the DATA line, for four Frames.
10.1.1.3	After four Frames, the active Framer **shall** change to the *StartingData* state.

4.1.1.4. CSB4 - *StartingData* state

Test Purpose:

Verify that the active Framer under test will generate the proper boot sequence. This state is also the first one in the boot sequence where the Framing Channel , the Guide Byte and the clock are transmitted concurrently.

Initial conditions:

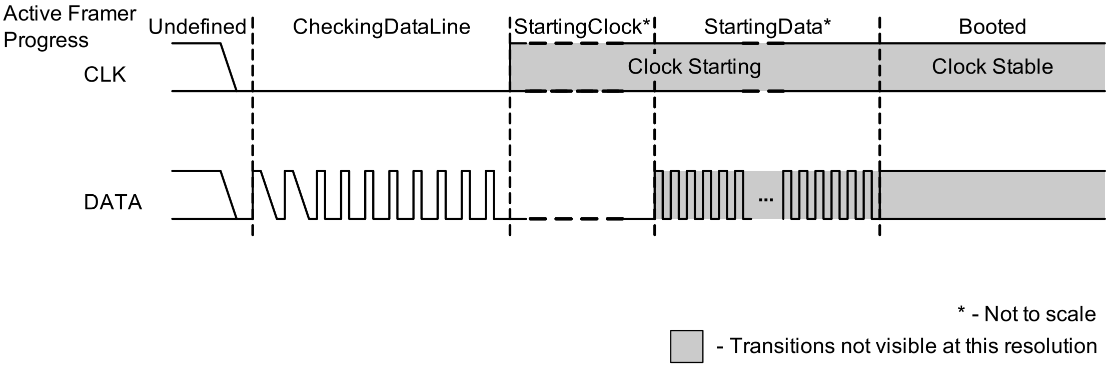
The active Framer is entering the *StartingData* state.

Stimulus:

None.

Success Criterion:

In the *StartingData* phase, the active Framer drives both the clock line and the data line during 16 frames (12288 clock cycles).



Verify that none of the state machines listed in the following table reports an error.

Test ID	State Machine Name
FS1	FrameSyncError
FS2	SuperFrameSyncError
FS3	FramingExtError
FS4	FramingInfoResvBitError
FS6	ClockGearMismatch
FS7	RootFrequencyMismatch
GB1	GuideByteIntegrityError
GB2	GuideByteENTbitError
GB3	GuideValueError

Coverage:

Section	Statement
10.1.1.4	In the StartingData state, the active Framer **shall** continue clocking the bus.
10.1.1.4	The Framing Channel and Guide Channel **shall** be written by the active Framer while in the StartingData state, enabling rapid synchronization of the Devices on the bus.
10.1.1.4	The active Framer **shall** not transmit any Messages while in this state.
10.1.1.4	Starting at the Superframe boundary, the active Framer **shall** transmit sixteen Frames and then change to the Booted state.
10.1.1.5	While the Clock Sourcing Component is not in Message synchronization (section 8), the active Framer **shall** write the ENT bit and the Guide Value as zero.

4.1.2. Clock Receiving Component Boot Sequence

The Clock Receiving Components do not have much visible effects during boot. When successful the Boot process results in transmitting a REPORT_PRESENT message to the active Manager. However, some conditions must be tested.

4.1.2.1. CRB1 - *SM field monitoring*

Test Purpose:

Verify that the device under test does not reach the SeekingMessageSync when the Framing Extention bit (FE) is equal to 0 and the Subframe Mode (SM) field does not exhibit identical values over two consecutive superframes.

Initial conditions:

The active Framer is entering the *StartingData* state.

Stimulus:

The Framing Extension (FE) bit is equal to 0. The Subframe Mode (SM) takes the following values:

- Superframe 0: SM=25 This superframe is part of the *StartingData* state
- Superframe 1: SM=24 This superframe is part of the *StartingData* state
- Superframe 2: SM=25
- Superframe 3: SM=24
- Superframe 4: SM=25
- Superframe 5: SM=24
- Superframe 6: SM=25
- Superframe 7: SM=24
- Superframe 8: SM=25
- Superframe 9: SM=25

All following superframes use SM=25.

Success Criterion:

Verify that there is no REPORT_PRESENT message sent while the SM value in the superframes is not constant over time.

Verify that a REPORT_PRESENT message is sent as soon as 2 consecutive superframes are showing identical Subframe Mode (SM) values.

Coverage:

Section	Statement
10.1.2.4	For Superframes with FE=0, a Component **shall** not change to the SeekingMessageSync state until it has observed the contents of SM field as being identical across two Superframes.

4.1.2.2. CRB2 - *CG field monitoring*

Test Purpose:

Verify that the device under test does not reach the SeekingMessageSync when the Framing Extention bit (FE) is equal to 0 and the Clock Gear (CG) field does not exhibit identical values over two consecutive superframes. Note that not all devices will monitor the CG field. It must be indicated in data sheet.

Initial conditions:

The active Framer is entering the *StartingData* state.

Stimulus:

The Framing Extension (FE) bit is equal to 0. The Subframe Mode (SM) takes the following values:

- Superframe 0: CG=9 This superframe is part of the *StartingData* state
- Superframe 1: CG=8 This superframe is part of the *StartingData* state
- Superframe 2: CG=9
- Superframe 3: CG=8
- Superframe 4: CG=9
- Superframe 5: CG=8
- Superframe 6: CG=9
- Superframe 7: CG=8
- Superframe 8: CG=9
- Superframe 9: CG=9

All following superframes use CG=9.

Success Criterion:

Verify that there is no REPORT_PRESENT message sent while the CG value in the superframes is not constant over time.

Verify that a REPORT_PRESENT message is sent as soon as 2 consecutive superframes are showing identical Clock Gear (CG) values.

Coverage:

Section	Statement
10.1.2.4	In addition, if the Component uses the CG or RF field it **shall** read the used field.
10.1.2.4	In addition, if the Component uses the CG or RF field it **shall** not change to the SeekingMessageSync state until it has observed the contents as being identical across two consecutive occurrences of the field.

4.1.2.3. CRB3 - *RF field monitoring*

Test Purpose:

Verify that the device under test does not reach the SeekingMessageSync when the Framing Extention bit (FE) is equal to 0 and the Root Frequency (RF) field does not exhibit identical values over two consecutive superframes. Note that not all devices will monitor the RF field. It must be indicated in data sheet.

Initial conditions:

The active Framer is entering the *StartingData* state.

Stimulus:

The Framing Extension (FE) bit is equal to 0. The Root Frequency (RF) takes the following values:

- Superframe 0: RF=1	This superframe is part of the *StartingData* state
- Superframe 1: RF=2	This superframe is part of the *StartingData* state
- Superframe 2: RF=1	
- Superframe 3: RF=2	
- Superframe 4: RF=1	
- Superframe 5: RF=2	
- Superframe 6: RF=1	
- Superframe 7: RF=2	
- Superframe 8: RF=1	
- Superframe 9: RF=1	

All following superframes use RF=1.

Success Criterion:

Verify that there is no REPORT_PRESENT message sent while the RF value in the superframes is not constant over time.

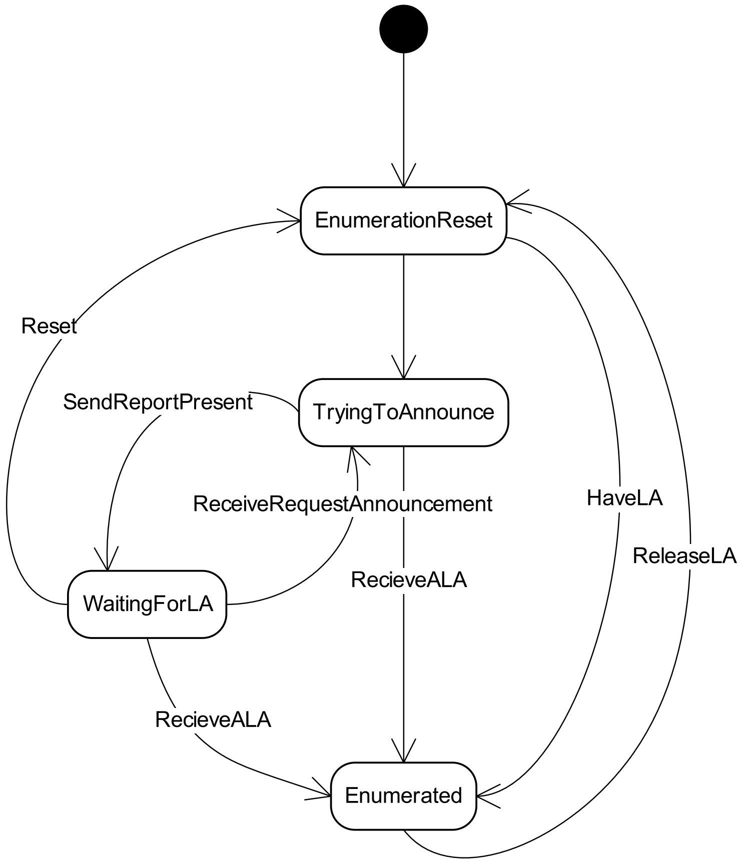
Verify that a REPORT_PRESENT message is sent as soon as 2 consecutive superframes are showing identical Root Frequency (RF) values.

Coverage:

Section	Statement
10.1.2.4	In addition, if the Component uses the CG or RF field it **shall** read the used field.
10.1.2.4	In addition, if the Component uses the CG or RF field it **shall** not change to the SeekingMessageSync state until it has observed the contents as being identical across two consecutive occurrences of the field.

4.2. Device Enumeration

Once the Boot process is finished, the Enumeration phase starts. Devices will announce themselves on the bus and will wait for the manager to configure them. Once they get assigned their Logical address, the devices are ready for bus operations.



We will test first the typical path:

- Transition to the *TryingToAnnounce* state
- Transition to the *WaitingForLA* state by sending a REPORT_PRESENT message
- Transition to the *Enumerated* state by receiving a ASSIGN_LOGICAL_ ADDRESS message

In a later stage, the direct transition from *EnumerationReset* to *Enumerated* will be tested.

These are a fundamental steps. If a device is failing one of the following tests, it will not be able to reliably participate to any bus transaction.

4.2.1. DE1 - REPORT_PRESENT message validity

Test Description:

When the Boot sequence is completed, the Clock Sourcing Component will enter the *Booted* state and Clock Receiving Component will enter the *Operational* state. These 2 states are equivalent as the Devices of the components are ready for nominal operations.

The First expected behavior of a properly booted device is the transmission of a REPORT_PRESENT message. This implies that the devices have acquired all the synchronization levels (Frame, Superframe and Message sync) and that the subframe mode is correctly decoded.

The REPORT_PRESENT Messages will be first NACKed to trigger message retransmission. The SLIMbus specification mandates a device to keep retransmitting the REPORT_PRESENT message till it gets a positive Acknowledge from the active Manager. However, it is not possible to fully test this requirement as it would require an infinite amount of time to execute the test. We will make the assumption that if the retransmission happens 10 times, it will be valid.

Initial conditions:

The Clock Sourcing Component is entering the *Booted* state or the Clock Receiving Component is entering the *Operational* state.

Stimulus:

Negatively acknowledge (NACK) the REPORT_PRESENT messages sent by all the Devices 10 times in a row and then send a Positive Acknowledge (PACK) to all the Devices.

Success Criterion:

Verify that all the devices present in the Component are issuing their REPORT_PRESENT message.

Verify that all the REPORT_PRESENT messages are retransmitted as long as they receive a negative acknowledge (NACK).

Verify that a REPORT_PRESENT message of the **Interface** device is transmitted and contains the Interface Device Class code (0xFD) and the Interface Device Class version (0x01). Verify that only one device reports a DC=0xFD.

If a **Framer** Device is present in the Component, Verify that the REPORT_PRESENT message of the active **Framer** device is transmitted and contains the Framer Device Class code (0xFE) and the Framer Device Class version (0x01).

If a (inactive) **Manager** device is present in the Component, verify that the REPORT_PRESENT message of a **Manager** device is present and contains the Generic Device Class code (0xFF) and the Generic Device Class version (0x01).

If a **Generic** device is present in the Component, verify that the REPORT_PRESENT message of a **Generic** device is present and contains the Generic Device Class code (0x00) and the Generic Device Class version (0x01).

Verify that each device has a unique Enumeration Address (which de facto includes a unique Device Index)

Verify that the Manufacturer ID, Product Code and the Instance Value are identical for all the devices that reported Present.

Verify that the Manufacturer ID has been assigned by the Mipi Alliance.

Coverage:

Section	Statement
12.1	A Device **shall** support the following Core Information Elements: DATA_TX_COL, **DEVICE_CLASS, DEVICE_CLASS_VERSION**, UNSPRTD_MSG
12.4	A SLIMbus Framer **shall** have Device Class code 0xFE.
12.4	A SLIMbus Framer that implements this version of the Device Class definition **shall** have Device Class Version code 0x01.
12.4.2	A Framer **shall** be able to send all Core Messages in the "Mandatory" column, and may also send any Core Message in the "Optional" column, of the "As Source" section of Table 73.
12.3	A SLIMbus Manager **shall** have Device Class code 0xFF.
12.3	A SLIMbus Manager that implements this version of the Device Class definition **shall** have Device Class Version code 0x01.
12.3.2	A Manager **shall** be able to send all Core Messages in the "Mandatory" column, and may also send any Core Message in the "Optional" column, of the "As Source" section of Table 70.
12.2	A SLIMbus Interface Device **shall** have Device Class code 0xFD.
12.2	A SLIMbus Interface Device that implements this version of the Device Class definition **shall** have Device Class Version code 0x01.
12.2	A Device of the Interface Device Class (Interface Device) **shall** be able to send all Core Messages in the "Mandatory" column, and may also send any Core Message in the "Optional" column, of the "As Source" section of Table 67.
12.5	A Member of the Generic Device Class **shall** have Device Class code 0x00.
12.5	In the absence of a separate Generic Device Class definition, a member of the Generic Device Class **shall** have Device Class Version code 0x01.
12.5.2	A Device of the Generic Device Class (Generic Device) **shall** be able to send all Core Messages in the "Mandatory" column, and may also send any Core Message in the "Optional" column, of the "As Source" section of Table 77.
4.4	A Component **shall** have one, and only one, Interface Device, but may have many Devices of other Device Classes
8.2.1	During Long Arbitration, a Device **shall** use its Enumeration Address when sending the arbitration sequence.
8.2.5.1	Only Manufacturer IDs assigned by the MIPI Alliance **shall** be used in this field.
8.2.5.1	Each Device within a Component **shall** have a unique Device Index.
8.2.5.1	The Manufacturer ID, Product Code and the Instance Value **shall** be identical for all Devices in the same Component.
8.2.5.1	Enumeration Addresses **shall** be unique across all Devices that can enumerate concurrently.
10.1	Clock Receiving Components **shall** synchronize according to sections 10.1.2.
10.1	A Component **shall** be capable of joining the bus by following the appropriate boot and synchronization processes.
10.1.2.5	Once the Component has achieved Message synchronization, it **shall** enter the Operational state.
10.9	If a Device is *Unenumerated*, has not yet received a Logical Address and gains Message synchronization, it **shall** enter the *TryingToAnnounce* state and send a REPORT_PRESENT Message to the active Manager.
10.9	The payload of the REPORT_PRESENT Message **shall** contain the Device Class code and the Device Class Version (DCV) of the Device.
10.9.1.2	While in the *TryingToAnnounce* state, a Device **shall** attempt to send the REPORT_PRESENT Message.
10.9.1.2	The Device **shall** repeat the REPORT_PRESENT Message until it is positively acknowledged by the active Manager.
10.9.1.2	After receiving the positive acknowledgement from the active Manager, the Device **shall** move to the WaitingForLA state.
11.1.1	This Message [REPORT_PRESENT] **shall** be sent by an unenumerated Device after it has gained Message synchronization as described in section 10.9.1.3.
12.1	A Device **shall** be capable of sending the following Core Messages: REPLY_INFORMATION [and REPORT_PRESENT]

4.2.2. DE2 - Request Self Announcement to non enumerated devices

Test Description:

Once a device has sent its REPORT_PRESENT message and got a positive acknowledge to it, it is in the *WaitingForLA* state. At this stage, a device can either accept an ASSIGN_LOGICAL_ADDRESS and move to the *Enumerated* state or receive a REQUEST_SELF_ANNOUNCEMENT message and move back to the *TryingToAnnounce* state.

Verify that all the devices behaves accordingly to the specification when receiving a REQUEST_SELF_ANNOUNCEMENT message.

Initial conditions:

The Component has been properly booted. It has sent the REPORT_PRESENT messages and got positive acknowledgments.

Stimulus:

Transmit the REQUEST_SELF_ANNOUNCEMENT message on the bus.

Assign the Logical Address 0x00 to the Interface Device.

Transmit the REQUEST_SELF_ANNOUNCEMENT message on the bus.

Success Criterion:

Verify that all the devices present in the Component are issuing their REPORT_PRESENT message.

Verify that a REPORT_PRESENT message of the **Interface** device is transmitted and contains the Interface Device Class code (0xFD) and the Interface Device Class version (0x01).

If a **Framer** Device is present in the Component, Verify that the REPORT_PRESENT message of the active **Framer** device is transmitted and contains the Framer Device Class code (0xFE) and the Framer Device Class version (0x01).

If a (inactive) **Manager** device is present in the Component, verify that the REPORT_PRESENT message of a **Manager** device is present and contains the Generic Device Class code (0xFF) and the Generic Device Class version (0x01).

If a **Generic** device is present in the Component, verify that the REPORT_PRESENT message of a **Generic** device is present and contains the Generic Device Class code (0x00) and the Generic Device Class version (0x01).

Verify that each device has a unique Enumeration Address (which de facto includes a unique Device Index).

Verify that only the non enumerated devices are reporting present.

Coverage:

Section	Statement
10.9.1	In this state [WaitingForLA], the Device **shall** wait until it receives an ASSIGN_LOGICAL_ADDRESS or a REQUEST_SELF_ANNOUNCEMENT Message.
10.9.1	The REQUEST_SELF_ANNOUNCEMENT Message **shall** cause the Device to return to the TryingToAnnounce state.

4.2.3. DE3 - Request Self Announcement to enumerated devices

Test Description:

Once a device has been properly enumerated (got a LA), it must ignore the REQUEST_SELF_ANNOUNCEMENT message.

Verify that all the devices behaves accordingly to the specification when receiving a REQUEST_SELF_ANNOUNCEMENT message.

Initial conditions:

The Component has been properly booted and enumerated.

Stimulus:

Transmit the REQUEST_SELF_ANNOUNCEMENT message on the bus.

Success Criterion:

Verify that none of the devices present in the Component are issuing their REPORT_PRESENT message.

Coverage:

Section	Statement
10.9.1.4	A Device in the Enumerated state **shall** ignore all REQUEST_SELF_ANNOUNCEMENT Messages.
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: ASSIGN_LOGICAL_ADDRESS, CHANGE_LOGICAL_ADDRESS, CLEAR_INFORMATION, REQUEST_CLEAR_INFORMATION, REQUEST_INFORMATION, **REQUEST_SELF_ANNOUNCEMENT**, RESET_DEVICE.

4.2.4. DE4 - Logical Address Assignment to None

Test Description:

When a device has sent a REPORT_PRESENT message, the active Manager will send an ASSIGN_LOGICAL_ADDRESS message to define the logical address the device (used in any further transactions with that device). We must verify that the device under test will not positively acknowledge a message for which he is not the destination.

Initial conditions:

The Component has been properly booted and all of its devices have reported present and have been positively acknowledged (PACK).

Stimulus:

Send an ASSIGN_LOGICAL_ADDRESS to a non existing device. Use EA=0x01C100000000 (it will never be used in a product).

Success Criterion:

Verify that the ASSIGN_LOGICAL_ADDRESS message is not acknowledged (read a NORE).

Coverage:

Section	Statement
8.4.2	A Device that is not a destination Device **shall** not acknowledge the Message.
8.4.2.7	A Device that is not a destination **shall** not write to the Response Field.

4.2.5. DE5 - Logical Address Assignment

Test Description:

When a device has sent a REPORT_PRESENT message, the active Manager will send an ASSIGN_LOGICAL_ADDRESS message to define the logical address the device (used in any further transactions with that device). We must verify that the device under test properly receive the newly assigned logical address. However, while in the *WaitingForLA* state, the device must not tranmit any message on the bus. We will test that by forcing the non enumerated device to lose the superframe sync. An enumerated Interface device would sent a REPORT_INFORMATION(EC=0x4008).

Initial conditions:

The Component has been properly booted and all of its devices have reported present and have been positively acknowledged (PACK).

Stimulus:

Corrupt the sync symbol of two consecutive superframes.

Transmit 1000 Superframes with no SMC traffic.

Send an ASSIGN_LOGICAL_ADDRESS to all the devices present in the component (start at value 0x01 and increase the LA by one unit on each iteration). Assign Logical Address 0x01 to the Interface Device.

Success Criterion:

Verify that the devices do not transmit any messages in the 1000 idle superframes. Note that this cannot be taken as a formal proof. The test duration should be infinite, which is not possible. 1000 superframes will be a good enough approximation of infinity in this particular case.

Verify that the all the ASSIGN_LOGICAL_ADDRESS messages are all positively acknowledged (PACK).

Verify that the REPORT_INFORMATION (if any) is sent after the Logical Address assignment.

Coverage:

Section	Statement
10.9.1.3	While in the WaitingForLA state, a Device **shall** not send any Messages, but act only upon receiving an ASSIGN_LOGICAL_ADDRESS Message or a REQUEST_SELF_ANNOUNCEMENT Message.
8.4.2	A destination Device **shall** respond to the source Device by sending an appropriate acknowledgement in the Response field.
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: **ASSIGN_LOGICAL_ADDRESS**, CHANGE_LOGICAL_ADDRESS, CLEAR_INFORMATION, REQUEST_CLEAR_INFORMATION, REQUEST_INFORMATION, REQUEST_SELF_ANNOUNCEMENT, RESET_DEVICE.

4.2.6. DE6 - Logical Address Assignment Verification

Test Description:

Once all the devices have been properly enumerated, we shall verify that the devices under test properly received the newly assigned logical address and moved to the *Enumerated* state.

Initial conditions:

The Component has been properly booted and all of its devices have reported present. Each Device of the Component has received a Logical Address. The Interface device gets a Logical Address = 0x01.

Stimulus:

Send a REQUEST_INFORMATION(TID=255, EC=0x0098) message to a non assigned logical address (use 0xEF by default).

Send a REQUEST_INFORMATION(TID=i, EC=0x0098) message to each of the devices in the component. The TID value **i** used in the REQUEST_ message must be equal to the Logical Address of the destination device.

Positively acknowledge the REPLY_INFORMATION messages.

Success Criterion:

Verify that the REQUEST_INFORMATION to the non-assigned LA is not acknowledged (read NORE).

Verify that the REQUEST_INFORMATION to enumerated devices are all positively acknowledged (read PACK).

Verify that all the devices are transmitting a REPLY_INFORMATION message with a TID that is equal to the Logical Address of the source device.

Verify that the DEVICE_CLASS code of the REPLY_INFORMATION message correspond to the device class of the device that got assigned the logical address that is sending back the message (code=0xFD for LA=0x01, for instance).

Coverage:

Section	Statement
8.2.1	During Short Arbitration, a Device **shall** use its Logical Address when sending the arbitration sequence.
8.4.2	A Device that is not a destination Device **shall** not acknowledge the Message.
8.4.2	A Device **shall** temporarily record the Logical Address of the source Device.
10.9.1	The ASSIGN_LOGICAL_ADDRESS Message **shall** cause the Device to change to the Enumerated state.
10.9.1.2	In addition, the Device **shall** act upon an ASSIGN_LOGICAL_ADDRESS Message sent to the Enumeration Address of the Device by changing to the Enumerated state.
10.10	That Logical Address **shall** be used by the destination Device during the arbitration sequence of all outgoing Messages.
10.10	In addition, the ASSIGN_LOGICAL_ADDRESS Message also informs a Device that it **shall** be the destination Device of all Messages that contain that specific Logical Address in their Header.

10.10	A Device **shall** use the assigned Logical Address, until a Component Reset (see section 10.4) forces the Device to discard the address, the Device detaches from the bus (see section 10.9.1.5), or until a CHANGE_LOGICAL_ADDRESS Message is received.
10.12.1	A destination Device of a REQUEST Message **shall** send a corresponding REPLY Message.
10.12.2	The REPLY Message **shall** contain the same Transaction ID as the REQUEST Message.
10.12.2	The REPLY Message **shall** have the Logical Address of the source of the REQUEST Message as its Destination Address.
10.12.2	A Device **shall** support at least one open transaction, and may support more than one.
12.1	A Device **shall** be capable of sending the following Core Messages: REPLY_INFORMATION and REPORT_PRESENT
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: ASSIGN_LOGICAL_ADDRESS, CHANGE_LOGICAL_ADDRESS, CLEAR_INFORMATION, REQUEST_CLEAR_INFORMATION, **REQUEST_INFORMATION**, REQUEST_SELF_ANNOUNCEMENT, RESET_DEVICE.

4.2.7. DE7 - Discard Second Logical Address Assignment

Test Description:

Once all the devices have been properly enumerated, we shall verify that the devices under test properly discard a second ASSIGN_LOGICAL_ADDRESS message. The proper method to modify the Logical Address of an Enumerated device is to use the CHANGE_LOGICAL_ADDRESS message.

Initial conditions:

The Component has been properly booted and enumerated. The Interface device gets a Logical Address = 0x01.

Stimulus:

Send a ASSIGN_LOGICAL_ADDRESS (LA=i+10) to each of the devices present where i is the currently assigned logical address.

Send a REQUEST_INFORMATION(TID=i+10, EC=0x0098) message to each of the devices in the component. The TID value **i** used in the REQUEST_ message must be equal to the Logical Address of the last valid ASSIGN_LOGICAL_ADDRESS messages.

Success Criterion:

Verify that none of REQUEST_INFORMATION messages is positively acknowledged (PACK).

Coverage:

Section	Statement
10.10	A Device **shall** ignore a second ASSIGN_LOGICAL_ADDRESS Message.
10.10	A Device **shall** not use the Logical Address in the payload of that second [ASSIGN_LOGICAL_ADDRESS] Message.

4.2.8. DE8 - Manager Logical Address Assignment Verification

Test Description:

The active Manager must automatically receive the Logical Address 0xFF.

Initial conditions:

The Component in which the active Manager resides has been properly booted. All of its devices have reported present and have been positively acknowledged (PACK) by the Active Manager. Eventually the active Manager will proceed with Device Enumeration. Wait for the first positively acknowledged REPORT_PRESENT message before starting the test.

Stimulus:

Send an REQUEST_INFORMATION (TID=1, EC=0x0098) to the active Manager.

Success Criterion:

Verify that the active Manager transmits the corresponding REPLY_INFORMATION (TID=1, EC=0x0098) and that the DC and DCV Information Elements correspond to the active Manager (DC=0xFF and DCV=0x01).

Coverage:

Section	Statement
10.9.2	Following the completion of the boot process for its Component, the active Manager **shall** use the Logical Address 0xFF.

4.2.9. DE9 - Change of Logical Address

Test Description:

Once a device is in the *Enumerated* state, its Logical Address can be changed by the Active Manager by using the CHANGE_LOGICAL_ADDRESS message. After reception of that message, the device must use the new Logical Address for all its message transactions.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send an CHANGE_LOGICAL_ADDRESS(LA=0x10) message to the Interface device.

Send an REQUEST_INFORMATION(TID=1,EC=0x0098) message to the previous Logical Address of the Interface device.

Send an REQUEST_INFORMATION(TID=2,EC=0x0098) message to the new Logical Address of the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=2).

Success Criterion:

Verify that the first REQUEST_INFORMATION(TID=1) message is NOT acknowledged (NORE).

Verify that the second REQUEST_INFORMATION(TID=2) message is positively acknowledged (PACK).

Verify that the reported Device Class code is equal to 0xFD.

Coverage:

Section	Statement
10.10	A Device **shall** use the assigned Logical Address, until a Component Reset (see section 10.4) forces the Device to discard the address, the Device detaches from the bus (see section 10.9.1.5), or until a CHANGE_LOGICAL_ADDRESS Message is received.
10.10	The CHANGE_LOGICAL_ADDRESS Message **shall** be used by the active Manager to inform a Device of a change in the Logical Address that it shall use during the Short arbitration sequence of all outgoing Messages.
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: ASSIGN_LOGICAL_ADDRESS, **CHANGE_LOGICAL_ADDRESS**, CLEAR_INFORMATION, REQUEST_CLEAR_INFORMATION, REQUEST_INFORMATION, REQUEST_SELF_ANNOUNCEMENT, RESET_DEVICE.

4.2.10. DE10 - Speed Enumeration

Test Description:

If a Device receives a Logical Address before it has a chance to send its REPORT_PRESENT message, it must cancel the transmission of the message. Its a direct transition from the *TryingToAnnounce* state to the *Enumerated* state (see figure 77 of the SLIMbus specification).

Initial conditions:

The Component is in the *TryingToAnnounce* state.

Stimulus:

Directly at the end of the boot sequence, assign a Logical Address to the devices present in the Component.

Success Criterion:

Verify that the devices positively acknowledge the ASSIGN_LOGICAL_ADDRESS messages.

Verify that the Devices do not send the REPORT_PRESENT messages.

Coverage:

none

4.3. Bus Reconfiguration

When a Component has reached the operational state and is ready for operation (understand able to receive and transmit messages), the various bus reconfiguration messages can be exercised.

4.3.1. BR1 - Subframe mode Change

Test Description:

A device must be capable of supporting all the Subframe modes. The modes are modified through a bus reconfiguration sequence. After a Subframe mode change, a device must keep all the synchronization levels.

Note that the mode change as such does not always produce visible changes if messages are not transmitted. To amplify the effects of a subframe mode change, it is necessary to cycle through all subframe modes with a given order.

There are 2 parameters implicitly given by the subframe mode value: the controls space size in slots and the subframe length in slots. When the control space size is equal to 1 or 2 slots, the Guide Byte slots are not contiguous and not happening always at the same position in the superframe. If the decoding of the subframe mode is not properly done, the device might not look at the right place for the Guide Byte and may fail the integrity checks. This will trigger the Interface device to set the LOST_MS IE and to try to transmit a REPORT_INFORMATION message with the relevant information (EC=0x4008). Note that such transmission might just fail because of the wrong subframe mode decoding.

A REQUEST_INFORMATION message sent to the device right after the reconfiguration boundary will indicate if the device is still in message sync. A PACK followed by a REPLY_INFORMATION message is a proof of good sync.

Initial conditions:

The Component is in the *Enumerated* state and the LOST_SF, LOST_SFS and LOST_MS IE are all cleared. The FE bit (of the Framing Information) is equal to 0.

Stimulus:

Send a Subframe mode change reconfiguration sequence. Right after the reconfiguration boundary, send a REQUEST_INFORMATION(TID=1, EC=0x4008) message to the Interface Device of the Component.

Repeat the test 27 times with the following sequence in the subframe mode (SM) parameter value: 0, 4, 13, 21, 5, 12, 22, 6, 15, 23, 7, 14, 25, 8, 16, 24, 9, 17, 26, 10, 18, 27, 28, 11, 19, 29, 31, 7, 0.

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged (PACK).

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer updates the content of the SM field in the Framing Channel.

Coverage:

Section	Statement
6.1.3	The Subframe length is programmable and **shall** be six, eight, twenty-four or thirty-two contiguous Slots (24, 32, 96 or 128 Cells)
6.3.1.2	If a Component receives a Superframe with FE=0, it **shall** obey all specified behaviors based on the SM, CG, and RF fields.
10.1.2.4	For Superframes with FE=0, a Component **shall** read the SM field of the Framing Information.
10.3.1	The active Manager **shall** indicate the start of a Reconfiguration Sequence by broadcasting the BEGIN_RECONFIGURATION Message.
10.3.1	The active Manager **shall** end a Reconfiguration Sequence with the RECONFIGURE_NOW Message.
10.3.1.1	Pending changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the RECONFIGURE_NOW Message.
10.3.1.1	A Device within the Component **shall** not discard the announced changes.
10.3.1.1	A Component **shall** behave as though all Reconfiguration Messages are executed at the associated Reconfiguration Boundary.
10.3.1.2	At that moment [the reconfiguration boundary], the Component **shall** change its behavior to reflect all of the pending Messages.
10.6	A Component **shall** configure its Frame Layer according to the pending Subframe Mode at the associated Reconfiguration Boundary.
10.6.1.1	The change of Subframe Mode has no influence on the location and width of the Framing Channel, but the active Framer **shall** adapt the content of the Framing Channel to match the new bus parameters.
10.12.1	A destination Device of a REQUEST Message **shall** send a corresponding REPLY Message.
10.12.2	The REPLY Message **shall** contain the same Transaction ID as the REQUEST Message.
10.12.2	The REPLY Message **shall** have the Logical Address of the source of the REQUEST Message as its Destination Address.

4.3.2. BR2 - Clock Gear Change

Test Description:

A device must be capable of supporting all the Clock Gears. The gears are modified through a bus reconfiguration sequence. After a Clock Gear change, a Device must keep all the synchronization levels.

Initial conditions:

The Component is in the *Enumerated* state and the LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a Clock Gear mode change reconfiguration sequence. Right after the reconfiguration boundary, send a REQUEST_INFORMATION(TID=1, EC=0x4008) message to the Interface Device of the Component. Repeat the test for 10 clock gear (CG) values: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,10

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer updates the content of the CG field in the Framing Channel and effectively modifies the clock frequency according to the requested value.

Coverage:

Section	Statement
6.2.1	Increasing to the next Clock Gear without changing the Root Frequency **shall** double the SLIMbus CLK frequency.
10.3.1.1	Pending changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the RECONFIGURE_NOW Message.
10.3.1.1	A Device within the Component **shall** not discard the announced changes.
10.3.1.1	A Component **shall** behave as though all Reconfiguration Messages are executed at the associated Reconfiguration Boundary.
10.3.1.2	At that moment [the reconfiguration boundary], the Component **shall** change its behavior to reflect all of the pending Messages.
10.5	A Component **shall** configure its Frame Layer according to the pending Clock Gear at the associated Reconfiguration Boundary.
10.5	The active Manager **shall** announce a Clock Gear change by sending the NEXT_CLOCK_GEAR Message using the broadcast mechanism.
10.5.1.1	[framing] Channel location in the Frame and Superframe for the Framing Channel **shall** remain constant over a Clock Gear change.
10.5.1.1	After the Reconfiguration Boundary, the active Framer **shall** transmit the new Clock Gear value in the Framing Channel as instructed by the active Manager.
10.5.1.2	A NEXT_CLOCK_GEAR Message shall not influence the location and parameters of the Message Channel. [equivalent to say that the Subframe Mode does not change]

4.3.3. BR3 - Root Frequency Change

Test Description:

A device must be capable of supporting all the Root Frequencies specified its data sheet. The frequency is modified through a bus reconfiguration sequence (of messages). After a Root Frequency change, a Device must keep all the synchronization levels.

Note that in case of the component would loose the message sync, the Interface device of the component must transmit a REPORT_INFORMATION(EC=4008) message to signify to the Manager a loss of sync.

Initial conditions:

The Component is in the *Enumerated* state and the LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a Root Frequency change reconfiguration sequence. Right after the reconfiguration boundary, send a REQUEST_INFORMATION(TID=1, EC=0x4008) message to the Interface Device of the Component.

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged (PACK).

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer updates the content of the RF field in the Framing Channel and effectively modifies the clock frequency according to the requested value.

Coverage:

Section	Statement
10.3.1.1	Pending changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the RECONFIGURE_NOW Message.
10.3.1.1	A Device within the Component **shall** not discard the announced changes.
10.3.1.1	A Component **shall** behave as though all Reconfiguration Messages are executed at the associated Reconfiguration Boundary.
10.3.1.2	At that moment [the reconfiguration boundary], the Component **shall** change its behavior to reflect all of the pending Messages.
10.7	At the Reconfiguration Boundary, the active Framer **shall** change the CLK line to match the announced Root Frequency and Clock Gear.
10.7	A Component **shall** configure its Frame Layer according to the pending Root Frequency at the associated Reconfiguration Boundary.
10.7	The active Manager **shall** announce a Root Frequency change by sending the NEXT_ROOT_FREQUENCY Message using the broadcast mechanism.
10.7.1.1	At the associated Reconfiguration Boundary, the active Framer **shall** change the contents of the Root Frequency (RF) field in the Framing Channel to reflect the new Root Frequency.

4.3.4. BR4 - Stacking reconfigurations

Test Description:

A device must be capable of supporting multiple reconfiguration messages in a single reconfiguration sequence. After the reconfiguration sequence is executed, the Device framing layer must be configured accordingly to the reconfiguration messages.

Note that in case of the component would loose the message sync, the Interface device of the component must transmit a REPORT_INFORMATION(EC=4008) message to signify to the Manager a loss of sync.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is equal to 9 and the subframe mode equal to 25. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a Clock Gear mode (CG=1) and a Subframe mode (SM=7) reconfiguration sequence. Right after the reconfiguration boundary, send a REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface Device of the Component.

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged (PACK).

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer updates the content of the SM and CG fields in the Framing Channel and effectively modifies the clock frequency according to the requested values.

Coverage:

Section	Statement
10.3.1.1	Pending changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the RECONFIGURE_NOW Message.
10.3.1.1	A Component **shall** behave as though all Reconfiguration Messages are executed at the associated Reconfiguration Boundary.
10.3.1.2	At that moment [the reconfiguration boundary], the Component **shall** change its behavior to reflect all of the pending Messages.

4.3.5. BR5 - Reconfiguration Message Execution Order

Test Description:

A device must execute the messages in the order they arrive. When transmitting many times the same reconfiguration message with different parameters in a reconfiguration sequence, the device must execute the last received message and discard the other ones.

Note that in case of the component would lose the message sync, the Interface device of the component must transmit a REPORT_INFORMATION(EC=4008) message to signify to the Manager a loss of sync.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is equal to 9 and the subframe mode equal to 25. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a Subframe mode reconfiguration sequence with many NEXT_SUBFRAME_MODE messages. Use SM=31, SM=7 and SM=19. Right after the reconfiguration boundary, send a REQUEST_INFORMATION(TID=1, EC=0x4008) message to the Interface Device of the Component.

The subframe mode must change to value 19 without going through 31 and 7.

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer properly updates the content of the SM value in the Framing Channel.

Coverage:

Section	Statement
8.4.2	The destination Device **shall** behave as though the Messages were executed in the order in which they were received.
10.3.1	Just like all the other Messages, Reconfiguration Messages **shall** behave as though they are executed in order.
10.3.1.1	Pending changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the RECONFIGURE_NOW Message.
10.3.1.1	A Device within the Component **shall** not discard the announced changes.
10.3.1.1	A Component **shall** behave as though all Reconfiguration Messages are executed at the associated Reconfiguration Boundary.
10.3.1.2	At that moment [the reconfiguration boundary], the Component **shall** change its behavior to reflect all of the pending Messages.
10.3.1.4	In this case [of reception of many reconfiguration messages affecting the same parameters], the most recently received value **shall** be used to update the parameter.

4.3.6. BR6 - Aborted Reconfiguration Sequence

Test Description:

Upon reception of a BEGIN_RECONFIGURATION, a device must discard any announce changes. Verify that a device can support an aborted reconfiguration sequence.

Note that in case of the component would loose the message sync, the Interface device of the component must transmit a REPORT_INFORMATION(EC=4008) message to signify to the Manager a loss of sync.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode (SM) value is equal to 25. The clock gear (CG) value is equal to 9. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send the following reconfiguration message sequence

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE (SM=7)
- NEXT_CLOCK_GEAR (CG=8)
- BEGIN_RECONFIGURATION
- RECONFIGURE_NOW

Right after the reconfiguration boundary, send the following messages:

- REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface Device

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged (PACK).

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer does not modify the SM and CG field value and does not modify the frequency of the bus clock.

Coverage:

Section	Statement
10.3.1.3	The reception of a BEGIN_RECONFIGURATION Message **shall** clear all pending bus state changes that have been communicated by Reconfiguration Messages.

4.3.7. BR7 - Reconfiguration Error - Missing BEGIN_RECONFIGURATION

Test Description:

A valid Reconfiguration sequence must always start with a Positively acknowledged BEGIN_RECONFIGURATION message. If this message is negatively acknowledged or is missing, the reconfiguration sequence can't go ahead. Any following Reconfiguration messages must be ignored.

At the reception of the RECONFIGURE_NOW, the Clock Receiving Component must lose superframe sync and eventually set the EX_ERROR IE.

The Interface device of the Clock Receiving Component must send a REPORT_INFORMATION(EC=0x4008) message to indicate a loss of superframe sync.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode (SM) value is equal to 25. The clock gear (CG) value is equal to 9. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send the following Reconfiguration messages:

- NEXT_SUBFRAME_MODE (SM=7)
- NEXT_CLOCK_GEAR (CG=8)
- RECONFIGURE_NOW

If the Component is a Clock Receiving Component, after reception of the REPORT_INFORMATION message from the Interface device, send a REQUEST_INFORMATION (TID=1, EC=0x0008) to the Interface Device.

If the Component is a Clock Sourcing Component, send a REQUEST_INFORMATION (TID=1, EC=0x0008) to the Interface Device right after the Reconfiguration Boundary.

Success Criterion:

If the Component is a Clock Receiving Component, verify that the REPORT_INFORMATION message is transmitted (and PACK it).

If the Component is a Clock Receiving Component, verify that the LOST_SFS Information Element is equal to 1.

Verify that the REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

If supported by the Interface device, verify that the EX_ERROR Information Element is equal to 1.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer does not modify the SM and CG field value and does not modify the frequency of the bus clock.

Coverage:

Section	Statement
10.3.1	No Reconfiguration Messages shall be transmitted without first ending the BEGIN_RECONFIGURATION Message.
10.3.1	If a Component receives a RECONFIGURE_NOW Message without first receiving a corresponding BEGIN_RECONFIGURATION Message, or if a Component loses Message synchronization between the BEGIN_RECONFIGURATION and RECONFIGURE_NOW Messages, the Component **shall** set the EX_ERROR Information Element as defined in section 7.1.2.3 if the EX_ERROR Information Element is implemented
10.3.1	In addition, a Clock Receiving Component **shall** lose Superframe synchronization at the Reconfiguration Boundary (Figure 57, SuperframeSyncLost transition).

4.3.8. BR8 - Reconfiguration Error - NACKed BEGIN_RECONFIGURATION

Test Description:

A valid Reconfiguration sequence must always start with a Positively acknowledged BEGIN_RECONFIGURATION message. If this message is negatively acknowledged or is missing, the reconfiguration sequence can't go ahead. Any following Reconfiguration messages must be ignored.

At the reception of the RECONFIGURE_NOW, the Clock Receiving Component must lose superframe sync and eventually set the EX_ERROR IE.

The Interface device of the Clock Receiving Component must send a REPORT_INFORMATION(EC=0x4008) message to indicate a loss of superframe sync.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode (SM) value is equal to 25. The clock gear (CG) value is equal to 9. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION with a NACK value in Message Response field
- NEXT_SUBFRAME_MODE (SM=7)
- NEXT_CLOCK_GEAR (CG=8)
- RECONFIGURE_NOW

If the Component is a Clock Receiving Component, after reception of the REPORT_INFORMATION message from the Interface device, send a REQUEST_INFORMATION (TID=1, EC=0x0008) to the Interface Device.

If the Component is a Clock Sourcing Component, send a REQUEST_INFORMATION (TID=1, EC=0x0008) to the Interface Device right after the Reconfiguration Boundary.

Success Criterion:

If the Component is a Clock Receiving Component, verify that the REPORT_INFORMATION message is transmitted (and PACK it).

If the Component is a Clock Receiving Component, verify that the LOST_SFS Information Element is equal to 1.

Verify that the REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

If supported by the Interface device, verify that the EX_ERROR Information Elements is equal to 1.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer does not modify the SM and CG field value and does not modify the frequency of the bus clock.

Coverage:

Section	Statement
10.3.1	No Reconfiguration Messages shall be transmitted without first ending the BEGIN_RECONFIGURATION Message.
10.3.1	If a Component receives a RECONFIGURE_NOW Message without first receiving a corresponding BEGIN_RECONFIGURATION Message, or if a Component loses Message synchronization between the BEGIN_RECONFIGURATION and RECONFIGURE_NOW Messages, the Component **shall** set the EX_ERROR Information Element as defined in section 7.1.2.3 if the EX_ERROR Information Element is implemented
10.3.1	In addition, a Clock Receiving Component **shall** loose Superframe synchronization at the Reconfiguration Boundary (Figure 57, SuperframeSyncLost transition).

4.3.9. BR9 - Reconfiguration Error - NACKed RECONFIGURE_NOW

Test Description:

When a RECONFIGURE_NOW message gets a NACK in the Message Response filed, it must be considered as non existing. The Reconfiguration Sequence will only be completed at the reception of a PACKed RECONFIGURE_NOW message. The announced changes are not reset by a NACKed RECONFIGURE_NOW message.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode (SM) value is equal to 25. The clock gear (CG) value is equal to 9. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE (SM=7)
- NEXT_CLOCK_GEAR (CG=8)
- RECONFIGURE_NOW with a NACK value in Message Response field

Right after the (non happening) reconfiguration boundary, send the following messages:

- REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface Device
- RECONFIGURE_NOW with a NORE value in Message Response field

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all cleared.

Verify that the announced changes are executed at the Reconfiguration Boundary.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer does not modify the SM and CG field value and does not modify the frequency of the bus clock after the first invalid RECONFIGURE_NOW.

Coverage:

Section	Statement
10.3.1.1	Note, a Component that does not observe a PACK in response to a RECONFIGURE_NOW Message **shall** not execute the Reconfiguration Messages.
10.3.1.1	[when seeing a NACKed RECONFIGURE_NOW] A Device within the Component **shall** not discard the announced changes.

4.3.10. BR10 - Reconfiguration Error - NACKed Reconfiguration messages

Test Description:

When a Reconfiguration Message gets a NACK in its Message Response field, it must be discarded by all the devices. Therefore, the announced changes must not be implemented.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode (SM) value is equal to 25. The clock gear (CG) value is equal to 9. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE (SM=7) with a NACK value in Message Response field
- NEXT_CLOCK_GEAR (CG=1) with a NACK value in Message Response field
- RECONFIGURE_NOW

Right after the (virtual) reconfiguration boundary, send the following messages:

- REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface Device

Success Criterion:

Verify that the REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer does not modify the SM and CG field value and does not modify the frequency of the bus clock.

Coverage:

Section	Statement
10.3.1.4	Reconfiguration Messages that receive a negative acknowledge **shall** be discarded by a Component.

4.3.11. BR11 - Delayed Reconfiguration Boundary

Test Description:

When the end of a RECONFIGURE_NOW message happens right at the end of a superframe, the reconfiguration boundary is postponed to the beginning of the next superframe.

Obviously, the only way to do that is to have the subframe mode set to 100% control space (SM=0).

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode (SM) value is equal to 0. The clock gear (CG) value is equal to 9. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE (SM=24)
- RECONFIGURE_NOW ending right at the end of a superframe

At the beginning of the superframe following the end of the RECONFIGURE_NOW message, send:

- REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface Device

At the beginning of the next superframe (which is the actual Reconfiguration Boundary), send:

- REQUEST_INFORMATION(TID=2, EC=0x4008) to the Interface Device

Success Criterion:

Verify that the first REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=1.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

Verify that the second REQUEST_INFORMATION message is positively acknowledged.

Verify that the REPLY_INFORMATION message is sent with the right TID=2.

Verify that the LOST_MS, LOST_SFS and LOST_FS Information Elements are all equal to 0.

Coverage:

Section	Statement
10.3.1.1	Pending changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the RECONFIGURE_NOW Message.

4.3.12. BR12 - Bus Shutdown

Test Description:

The bus can be stopped by a reconfiguration sequence containing the NEXT_SHUTDOWN_BUS message. At the Reconfiguration Boundary, the active Framer will stop clocking the bus. This is the only observable result of a bus shutdown.

So the test can only apply to a Clock Sourcing Component.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_SHUTDOWN_BUS
- RECONFIGURE_NOW

Success Criterion:

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer stops the clock activity at the superframe boundary.

Coverage:

Section	Statement
10.3.3	The active Framer **shall** respond to this command [NEXT_BUS_SHUTDOWN] by moving to the Undefined state (Figure 55) at the associated Reconfiguration Boundary.

4.3.13. BR13 - Reserved Subframe Modes

Test Description:

If, for any reasons, a reserved Subframe Mode is broadcasted, the device should behave accordingly to the following table:

Subframe mode	Equivalent to
0b00001 (1)	0b00101 (5) = 1/8
0b00010 (2)	0b00000 (0) = 8/8
0b00011 (3)	0b00111 (7) = 1/32
0b10100 (20)	0b00000 (0) = 8/8
0b11110 (30)	0b00000 (0) = 8/8

A device should still be able to participate to message transactions, even with the reserved Subframe Modes.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE(SM=...)
- RECONFIGURE_NOW

At the Reconfiguration Boundary, send:

- REQUEST_INFORMATION(TID=1, EX=0x4008) to the Interface Device of the Component.

Positively acknowledge the coming REPLY message.

Repeat the sequence for all the reserved Subframe Modes listed in the table.

Success Criterion:

Verify that the REQUEST messages are positively acknowledged (PACK).

Verify that the REPLY messages are sent with the proper TID.

Verify that the IE LOST_MS, LOST_SFS and LOST_FS are all cleared.

Coverage:

Section	Statement
6.3.1.4	A Component that observes a "Reserved" Subframe Mode coding in the Framing Information **shall** behave as specified in the "Comments" column in Table 15.

4.3.14. BR14 - Clock Pause and message reception

Test Description:

A device must stay synchronized when a clock pause happens. More specifically, if a message is crossing the superframe boundary where the clock pause happens, the message transmission will resume at the end of the clock pause and the receiving device must properly receive and process the message.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode is 19 (4/32).

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_PAUSE_CLOCK(RT=2)
- RECONFIGURE_NOW

Send a message REQUEST_INFORMATION(TID=1, EX=0x4008) to the Interface Device of the Component. That message must cross the superframe boundary.

Positively acknowledge the coming REPLY message.

Success Criterion:

Verify that the REQUEST_INFORMATION(TID=1, EC=0x04008) message is positively acknowledged (PACK).

Verify that the REPLY message is sent with the proper TID.

Verify that the IE LOST_MS, LOST_SFS and LOST_FS are all cleared.

Coverage:

Section	Statement
10.3.2	If a Message is interrupted by a Clock Pause, i.e. only part of the Message has been sent at the Superframe boundary, the source Device shall resume sending the rest of the Message in the first Message Slot after the Framer has resumed toggling the CLK line.

4.3.15. BR15 - Clock Pause and message transmission

Test Description:

A device must stay synchronized when a clock pause happens. More specifically, if a message is crossing the superframe boundary where the clock pause happens, the message transmission will resume at the end of the clock pause and the transmitting device must not introduce spurs or errors in the transmission.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode is 19 (4/32).

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_PAUSE_CLOCK(RT=2)
- RECONFIGURE_NOW

Send a message REQUEST_INFORMATION(TID=1, EX=0x4008) to the Interface Device of the Component in such a way that the REPLY message will cross the superframe boundary.

Positively acknowledge the coming REPLY message.

Success Criterion:

Verify that the REQUEST_INFORMATION(TID=1, EC=0x04008) message is positively acknowledged (PACK).

Verify that the REPLY message is sent with the proper TID and does not contains any errors.

Verify that the IE LOST_MS, LOST_SFS and LOST_FS are all cleared.

Coverage:

Section	Statement
10.3.2	If a Message is interrupted by a Clock Pause, i.e. only part of the Message has been sent at the Superframe boundary, the source Device **shall** resume sending the rest of the Message in the first Message Slot after the Framer has resumed toggling the CLK line.

4.4. Information Element and Value Element Management

These tests will verify the Information Elements and Value Elements specific management. Basic functionalities have been tested yet in the previous sections. Note that the Byte access will be used to test the IE and VE functionalities. The Elemental Access being optional, it will not be addressed in this document revision.

4.4.1. IE1 - Illegal IE Address

Test Description:

Element codes of a REQUEST message in the range 0xC000 to 0xFFFF are pointing to a Reserved portion of the Information Map and are forbidden. Such a request shall trigger the transmission of a REPLY_INFORMATION message with an abbreviated response (no Information Slice, just a TID).

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following message to the Interface Device of the DUT:

- REQUEST_INFORMATION(TID=1,EC=0xC008)

Success Criterion:

Verify that the Interface Device sends a REPLY_INFORMATION without any Information Slice, just a TID equal to 1.

Coverage:

Section	Statement
10.12.2	If the Element Code of a REQUEST Message is in the range 0xC00 to 0xFFF or is an elemental access that the target does not support, the target **shall** abbreviate the REPLY Message such that its payload is just the Transaction ID.

4.4.2. IE2 - Non existing IE Address

Test Description:

Element codes of a REQUEST message pointing to a non-existing IE shall trigger the transmission of a REPLY_INFORMATION message with an abbreviated response (no Information Slice, just a TID).

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following message to the Interface Device of the DUT:

- REQUEST_INFORMATION(TID=1,EC=0x00A8)

Success Criterion:

Verify that the Interface Device sends a REPLY_INFORMATION without any Information Slice, just a TID equal to 1.

Coverage:

Section	Statement
10.12.2	If the Element Code of a REQUEST Message is in the range 0xC00 to 0xFFF or is an elemental access that the target does not support, the target **shall** abbreviate the REPLY Message such that its payload is just the Transaction ID.

4.4.3. IE3 - Vacant Part of IEs

Test Description:

When using Byte Access method, the vacant part of the byte (or bytes) shall appear as 0. For the test purpose, the most efficient way to test this is to request the content of the core IE byte address 0. Only the 4 least significant bits are used.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following message to the Interface Device of the DUT:

- REQUEST_INFORMATION(TID=1,EC=0x0008)

Success Criterion:

Verify that the Interface Device sends a REPLY_INFORMATION with TID equal to 1 and with the four most significant bit of the Information Slice equal to 0.

Coverage:

Section	Statement
7.1	When vacant parts of the Information Map are transferred using byte-based accesses, they **shall** appear as zeros.

4.4.4. IE4 - Core Information Element Attribute

Test Description:

The Information Elements can be either clearable or read-only. The tests consists in clearing read-only IE and clearable IEs. However, not all clearable IEs can be easily set so the tests won't be exhaustive.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Clear the DEVICE_CLASS Core IE at address 0x009 containing the Device Class code. Then proceed to a value read back. Send the following messages to the Interface Device of the DUT:

- CLEAR_INFORMATION(EC=0x0098, CM=0xFF)
- REQUEST_INFORMATION(TID=1, EC=0x0098)

Clear the DEVICE_CLASS_VERSION Core IE at address 0x008 containing the Device Class code. Then proceed to a value read back. Send the following messages to the Interface Device of the DUT:

- CLEAR_INFORMATION(EC=0x0088, CM=0xFF)
- REQUEST_INFORMATION(TID=1, EC=0x0088)

Set the UNSPRTD_MSG IE by sending a CONNECT_SOURCE (or any other unsupported message) to the Interface Device of the DUT. Read the content of the byte address 0x000. Clear the content by using the Clear Mask value 0x0B. Read again the content.

- CONNECT_SOURCE(PN=0,CN=0)
- REQUEST_INFORMATION(TID=1, EC=0x0008)
- CLEAR_INFORMATION(EC=0x0008, CM=0x0B)
- REQUEST_INFORMATION(TID=2, EC=0x0008)

Success Criterion:

Verify that reported value for the DC is equal to 0xFD (and not 0x00).

Verify that reported value for the DCV is equal to 0x01 (and not 0x00).

Verify first that the UNSPRTD_MSG IE is set, then verify that it is cleared.

Coverage:

Section	Statement
7.1.2	Each of the Core Information Elements **shall** have the attributes identified for it in Table 22.
7.1.2	Each clearable Information Element in Table 22 **shal**l be cleared by the Information Element being a target of an appropriately formulated REQUEST_CLEAR_INFORMATION or CLEAR_INFORMATION Message
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: ASSIGN_LOGICAL_ADDRESS, CHANGE_LOGICAL_ADDRESS, **CLEAR_INFORMATION**, REQUEST_CLEAR_INFORMATION, REQUEST_INFORMATION, REQUEST_SELF_ANNOUNCEMENT, RESET_DEVICE.

4.4.5. IE5 - Core Information Element Reset

Test Description:

The clearable Information Elements are reset by a Component undergoing a boot process. This can happen when the component is reset or when the SLIMbus is reset. However, not all clearable IEs can be easily set so the tests won't be exhaustive.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Set the UNSPRTD_MSG IE by sending a CONNECT_SOURCE (or any other unsupported message) to the Interface Device of the DUT. Read the content of the byte address 0x000.

- CONNECT_SOURCE(PN=0,CN=0)
- REQUEST_INFORMATION(TID=1, EC=0x0008)

Reset the component by sending a RESET_DEVICE message to the Interface Device of the DUT.

Reassign a logical address to the Interface Device and read back the UNSPRTD_MSG IE.

Set the UNSPRTD_MSG IE by sending a CONNECT_SOURCE (or any other unsupported message) to the Interface Device of the DUT. Read the content of the byte address 0x000.

- CONNECT_SOURCE(PN=0,CN=0)
- REQUEST_INFORMATION(TID=1, EC=0x0008)

Reset the component by sending a RESET_DEVICE message to the Interface Device of the DUT.

- BEGIN_RECONFIGURATION
- NEXT_BUS_RESET
- RECONFIGURE_NOW

Reassign a logical address to the Interface Device and read back the UNSPRTD_MSG IE.

Success Criterion:

Verify that the UNSPRTD_MSG IE is cleared by a Component reset.

Verify that the UNSPRTD_MSG IE is cleared by a Bus reset.

Coverage:

Section	Statement
7.1.2	Each clearable Information Element in Table 22 **shall** be cleared by the Device being reset by a Device Reset, a Component Reset or a Bus Reset (section 10.4)
7.1.2.6	The Device **shall** set UNSPRTD_MSG whenever it receives a non-broadcast Message that it does not implement as a Destination.
11.3.3	Any unused bits **shall** be filled with zeros.
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: ASSIGN_LOGICAL_ADDRESS, CHANGE_LOGICAL_ADDRESS, CLEAR_INFORMATION, REQUEST_CLEAR_INFORMATION, REQUEST_INFORMATION, REQUEST_SELF_ANNOUNCEMENT, **RESET_DEVICE**.

4.4.6. IE6 - Core Information Element Value Verification

Test Description:

The stored Device Class code and Device Class version read by a REQUEST_INFORMATION message must match the values reported by the REPORT_PRESENT message. This must be verified for all the devices in a Component.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send to every Device the following message sequence:

- REQUEST_INFORMATION(TID=1, EC=0x0098)
- REQUEST_INFORMATION(TID=2, EC=0x0088)

Positively acknowledge the REPLY messages.

Success Criterion:

Verify that the DEVICE_CLASS IE reported by TID=1 corresponds to the Logical Device.

Verify that the DEVICE_CLASS_VERSION IE reported by TID=2 corresponds to the Logical Device.

Coverage:

Section	Statement
7.1.2.1	The DEVICE_CLASS Information Element **shall** be equal to the Device Class Code of the Device.
7.1.2.2	The DEVICE_CLASS_VERSION Information Element **shall** be equal to the version of the Device Class implemented by the Device.
11.3.3	In all other cases [payload containing more than just the TID] the Payload **shall** contain the Information Slice that was identified in the corresponding REQUEST Message.
11.3.3	The Information Slice **shall** be placed in the LSBs of the relevant payload byte(s).
12.2.5	An Interface Device **shall** support byte-based requests and clears of its Class specific Information Elements using 1, 2 and 4-byte Slice Sizes.
12.3.5	A Manager **shall** support byte-based requests and clears of its Class specific Information Elements using 1, 2 and 4-byte Slice Sizes.
12.4.5	A Framer **shall** support byte-based requests and clears of its Class specific Information Elements using 1, 2 and 4-byte Slice Sizes.
12.5.3	A Generic Device **shall** support byte-based requests and clears of its Core Information Elements using 1, 2 and 4-byte Slice Sizes.

4.4.7. IE7 - REQUEST_CLEAR_INFORMATION support

Test Description:

Any SLIMbus device must support the REQUEST_CLEAR_INFORMATION message and perform the required actions:

- send a REPLY_INFORMATION message with the content of the Information Element
- clear the content of the Information Element.

Additionally, a device must support an abbreviated Clear Mask (CM) or a longer than necessary Clear Mask.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following messages to the Interface device:

- CONNECT_SOURCE(PN=0,CN=0) or any other unsupported message.
- REQUEST_CLEAR_INFORMATION(TID=1, EC=0x0008, CM=0x0B)
- REQUEST_INFORMATION(TID=2, EC=0x0008)

Send the following messages to the Interface device:

- CONNECT_SOURCE(PN=0,CN=0) or any other unsupported message.
- REQUEST_CLEAR_INFORMATION(TID=3, EC=0x0008) - No CM (0 byte)
- REQUEST_INFORMATION(TID=4, EC=0x0008)

Send the following messages to the Interface device:

- CONNECT_SOURCE(PN=0,CN=0) or any other unsupported message.
- REQUEST_CLEAR_INFORMATION(TID=5, EC=0x0008, CM=0xFF0B)
- REQUEST_INFORMATION(TID=6, EC=0x0008)

Success Criterion:

Verify that the REPLY_INFORMATION(TID=1) message reports UNSPRTD_MSG IE set.

Verify that the REPLY_INFORMATION(TID=2) message reports UNSPRTD_MSG IE cleared.

Verify that the REPLY_INFORMATION(TID=3) message reports UNSPRTD_MSG IE set.

Verify that the REPLY_INFORMATION(TID=4) message reports UNSPRTD_MSG IE cleared.

Verify that the REPLY_INFORMATION(TID=5) message reports UNSPRTD_MSG IE set.

Verify that the REPLY_INFORMATION(TID=6) message reports UNSPRTD_MSG IE cleared.

Coverage:

Section	Statement
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: ASSIGN_LOGICAL_ADDRESS, CHANGE_LOGICAL_ADDRESS, CLEAR_INFORMATION, **REQUEST_CLEAR_INFORMATION**, REQUEST_INFORMATION, REQUEST_SELF_ANNOUNCEMENT, RESET_DEVICE.
11.3.2	A Device **shall** change to their reset values, those bits of the identified Information Slice for which the corresponding Clear Mask bit is 1.
11.3.2	If the size of Clear Mask is smaller than the size of the Information Slice, the Device **shall** pad the MSBs of Clear Mask with ones.
11.3.2	If the size of the Clear Mask is zero bytes, the Device **shall** change all bits of the Information Slice to their Reset Value.
11.3.2	If the size of Clear Mask is larger than the size of the Information Slice, the Device **shall** change only the portion of the Information Map corresponding to the identified Information Slice.
11.3.3	The Destination Address field **shall** be the Logical Address of the Device that sent the REQUEST_INFORMATION or REQUEST_CLEAR_INFORMATION Message.
11.3.3	The Transaction ID TID[7:0] **shall** be the Transaction ID of the corresponding REQUEST_INFORMATION or REQUEST_CLEAR_INFORMATION Message.

4.4.8. IE8 - CLEAR_INFORMATION support

Test Description:

Any SLIMbus device must support the CLEAR_INFORMATION message and perform the required actions:

- clear the content of the Information Element.

Additionally, a device must support an abbreviated Clear Mask (CM) or a longer than necessary Clear Mask.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following messages to the Interface device:

- CONNECT_SOURCE(PN=0,CN=0) or any other unsupported message.
- REQUEST_INFORMATION(TID=1, EC=0x0008)
- CLEAR_INFORMATION(EC=0x0008, CM=0x0B)
- REQUEST_INFORMATION(TID=2, EC=0x0008)

Send the following messages to the Interface device:

- CONNECT_SOURCE(PN=0,CN=0) or any other unsupported message.
- REQUEST_INFORMATION(TID=3, EC=0x0008)
- CLEAR_INFORMATION(EC=0x0008) - No CM (0 byte)
- REQUEST_INFORMATION(TID=4, EC=0x0008)

Send the following messages to the Interface device:

- CONNECT_SOURCE(PN=0,CN=0) or any other unsupported message.
- REQUEST_INFORMATION(TID=5, EC=0x0008)
- CLEAR_INFORMATION(EC=0x0008, CM=0xFF0B)
- REQUEST_INFORMATION(TID=6, EC=0x0008)

Success Criterion:

Verify that the REPLY_INFORMATION(TID=1) message reports UNSPRTD_MSG IE set.

Verify that the REPLY_INFORMATION(TID=2) message reports UNSPRTD_MSG IE cleared.

Verify that the REPLY_INFORMATION(TID=3) message reports UNSPRTD_MSG IE set.

Verify that the REPLY_INFORMATION(TID=4) message reports UNSPRTD_MSG IE cleared.

Verify that the REPLY_INFORMATION(TID=5) message reports UNSPRTD_MSG IE set.

Verify that the REPLY_INFORMATION(TID=6) message reports UNSPRTD_MSG IE cleared.

Coverage:

Section	Statement
12.1	A Device **shall** be capable of receiving the following Core Messages and shall perform the action specified for the Message: ASSIGN_LOGICAL_ADDRESS, CHANGE_LOGICAL_ADDRESS, **CLEAR_INFORMATION**, REQUEST_CLEAR_INFORMATION, REQUEST_INFORMATION, REQUEST_SELF_ANNOUNCEMENT, RESET_DEVICE.
11.3.4	A Device **shall** change to their reset values, those bits of the identified Information Slice for which the corresponding Clear Mask bit is 1.
11.3.4	If the size of Clear Mask is smaller than the size of the Information Slice, the Device **shall** pad the MSBs of Clear Mask with ones.
11.3.4	If the size of Clear Mask is zero bytes, the Device **shall** change all bits of the Information Slice to their reset values.
11.3.4	If the size of Clear Mask is larger than the size of the Information Slice, the Device shall change only the portion of the Information Map corresponding to the identified Information Slice.
11.3.4	[CLEAR_INFORMATION] The CM field **shall** be placed in the LSBs of the relevant payload byte(s).

4.4.9. IE9 - DATA_TX_COL Core Information Element Verification

Test Description:

The DATA_TX_COL IE indicates that a collision happened in a data channel segment. In order to be able to run the test, the DUT must implement a source data port. If not, the DUT must return 0 for the DATA_TX_COL IE.

To generate a collision, the DUT must be configured to source data in a data channel and the Traffic Generator (TG) shall be configured to send data in the same segments. The TG will output a fixed 0xFFFF value to maximize the probability to cause a collision.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Configure the DUT source port to use the channel 0.

Configure data channel 0 to use the Isochronous transport protocol, 48 kHz channel rate, 48 kHz Presence Rate and a Segment Length of 4 slots. Use Subframe Mode 19, CLock Gear 9 and Root Frequency 1. The Segment Distribution is equal to 3140.

Configure the Traffic Generator to act as a source on data channel 0. The channel has now 2 sources (which is strictly forbidden in nominal operation).

After 100 superframes, deactivate the channel 0 and send a REQUEST_INFORMATION message to the data sourcing logical device:

- REQUEST_INFORMATION(TID=1, EC=0x0008)

send a REQUEST_INFORMATION message to the Interface Device:

- REQUEST_INFORMATION(TID=2, EC=0x0008)

After the REPLY_INFORMATION is received, clear the DATA_TX_COL IE and verify its value again to check the clear operation.

- CLEAR_INFORMATION(TID=1, EC=0x0008,CM=0x0B)
- REQUEST_INFORMATION(TID=1, EC=0x0008)

Success Criterion:

Verify that the DATA_TX_COL IE is equal to 1 for the data source device.

Verify that the DATA_TX_COL IE is equal to 0 for the Interface device.

Verify that the DATA_TX_COL IE of the data source device is equal to 0 after the CLEAR_INFORMATION message.

Coverage:

Section	Statement
7.1.2.5	The Device **shall** set DATA_TX_COL whenever the Component the Device resides within detects a DATA line Collision while one of the Device's Ports is writing to a Data Channel.
7.1.2.5	DATA_TX_COL **shall** always return FALSE if a Device does not implement any Ports.

4.4.10. VE1 - Illegal VE Address

Test Description:

Element codes of a REQUEST message in the range 0xC000 to 0xFFFF are pointing to a Reserved portion of the Value Map and are forbidden. Such a request shall trigger the transmission of a REPLY_VALUE message with an abbreviated response (no Value Slice, just a TID).

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following message to the Interface Device of the DUT:

- REQUEST_VALUE(TID=1,EC=0xC008)

Success Criterion:

Verify that the Interface Device sends a REPLY_INFORMATION without any Value Slice, just a TID equal to 1.

Coverage:

Section	Statement
10.12.2	If the Element Code of a REQUEST Message is in the range 0xC00 to 0xFFF or is an elemental access that the target does not support, the target **shall** abbreviate the REPLY Message such that its payload is just the Transaction ID.

4.4.11. VE2 - Non existing VE Address

Test Description:

Element codes of a REQUEST message pointing to a non-existing IE shall trigger the transmission of a REPLY_VALUE message with an abbreviated response (no Information Slice, just a TID).

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following message to the Interface Device of the DUT:

- REQUEST_VALUE(TID=1,EC=0x00A8)

Success Criterion:

Verify that the Interface Device sends a REPLY_VALUE without any Value Slice, just a TID equal to 1.

Coverage:

Section	Statement
10.12.2	If the Element Code of a REQUEST Message is in the range 0xC00 to 0xFFF or is an elemental access that the target does not support, the target **shall** abbreviate the REPLY Message such that its payload is just the Transaction ID.

4.4.12. VE3 - REQUEST_CHANGE_VALUE support

Test Description:

SLIMbus devices may support the REQUEST_CHANGE_VALUE message. The actions are:

- send a REPLY_VALUE message with the content of the Value Element
- change the content of the Value Element.

Additionally, a device must support an abbreviated Value Update (VU) or a longer than necessary Value Update.

Initial conditions:

The Component is in the *Enumerated* state. The Value Element used for the test is 1 byte long and is cleared. The address used in the text is 0x000. Use the appropriate one.

Stimulus:

Send the following messages to the device under test:

- REQUEST_CHANGE_VALUE(TID=1, EC=0x0008, VU=0x1F)
- REQUEST_VALUE(TID=2, EC=0x0008)

Send the following messages to the device under test:

- REQUEST_CHANGE_VALUE(TID=3, EC=0x0008)
- REQUEST_VALUE(TID=4, EC=0x0008)

Send the following messages to the device under test:

- REQUEST_CHANGE_VALUE(TID=5, EC=0x0008, VU=0xFF1F)
- REQUEST_VALUE(TID=6, EC=0x0008)

Give positive acknowledgment to all the REPLY messages.

Success Criterion:

Verify that the REPLY_VALUE(TID=1) message reports a VU=0x00.

Verify that the REPLY_VALUE(TID=2) message reports a VU=0x1F.

Verify that the REPLY_VALUE(TID=3) message reports a VU=0x1F.

Verify that the REPLY_VALUE(TID=4) message reports a VU=0x00.

Verify that the REPLY_VALUE(TID=5) message reports a VU=0x00.

Verify that the REPLY_VALUE(TID=6) message reports a VU=0x1F.

Coverage:

Section	Statement
11.5.2	The VU field **shall** be placed in the LSBs of the relevant payload byte(s).
11.5.2	A Device **shall** overwrite the identified Value Slice with Value Update.
11.5.2	If the size of Value Update is smaller than the size of the Value Slice, the Device **shall** pad the MSBs of Value Update with zeroes. For example, if the size of Value Update is zero bytes, the Device **shall** change all bits of the Value Slice to zeroes.
11.5.2	If the size of Value Update is larger than the size of the Value Slice, the Device shall change only the portion of the Value Map corresponding to the identified Value Slice.
11.5.3	The Destination Address field shall be the Logical Address of the Device that sent the REQUEST_VALUE or REQUEST_CHANGE_VALUE Message.

11.5.3	The Transaction ID TID[7:0] shall be the Transaction ID of the corresponding REQUEST_VALUE or REQUEST_CHANGE_VALUE Message
11.5.3	In all other cases the payload shall contain the Value Slice that was identified in the corresponding REQUEST Message.
11.5.3	The Value Slice shall be placed in the LSBs of the relevant payload byte(s).

4.4.13. VE4 - CHANGE_VALUE support

Test Description:

SLIMbus devices may support the CHANGE_VALUE message. The actions are:

- change the content of the Value Element.

Additionally, a device must support an abbreviated Value Update (VU) or a longer than necessary Value Update.

Initial conditions:

The Component is in the *Enumerated* state. The Value Element used for the test is 1 byte long and is cleared. The address used in the text is 0x000. Use the appropriate one.

Stimulus:

Send the following messages to the device under test:

- REQUEST_VALUE(TID=1, EC=0x0008)
- CHANGE_VALUE(EC=0x0008, VU=0x1F)
- REQUEST_VALUE(TID=2, EC=0x0008)

Send the following messages to the device under test:

- REQUEST_VALUE(TID=3, EC=0x0008)
- CHANGE_VALUE(EC=0x0008)
- REQUEST_VALUE(TID=4, EC=0x0008)

Send the following messages to the device under test:

- REQUEST_VALUE(TID=5, EC=0x0008)
- CHANGE_VALUE(EC=0x0008, VU=0xFF1F)
- REQUEST_VALUE(TID=6, EC=0x0008)

Give positive acknowledgment to all the REPLY messages.

Success Criterion:

Verify that the REPLY_VALUE(TID=1) message reports a VU=0x00.

Verify that the REPLY_VALUE(TID=2) message reports a VU=0x1F.

Verify that the REPLY_VALUE(TID=3) message reports a VU=0x1F.

Verify that the REPLY_VALUE(TID=4) message reports a VU=0x00.

Verify that the REPLY_VALUE(TID=5) message reports a VU=0x00.

Verify that the REPLY_VALUE(TID=6) message reports a VU=0x1F.

Coverage:

Section	Statement
11.5.4	The VU field **shall** be placed in the LSBs of the relevant payload byte(s).
11.5.4	A Device **shall** overwrite the identified Value Slice with Value Update.
11.5.4	If the size of Value Update is smaller than the size of the Value Slice, the Device **shall** pad the MSBs of Value Update with zeroes. For example, if the size of Value Update is zero bytes, the Device **shall** change all bits of the Value Slice to zeroes.
11.5.4	If the size of Value Update is larger than the size of the Value Slice, the Device **shall** change only the portion of the Value Map corresponding to the identified Value Slice.

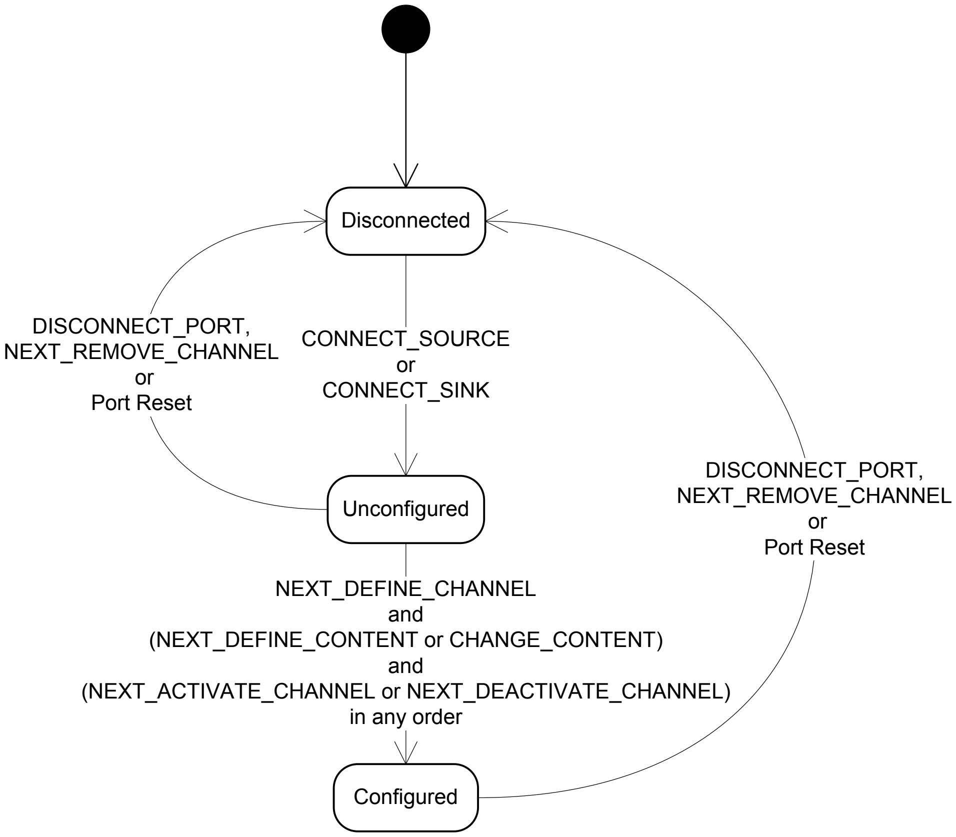
5. Data Channel Management

5.1. Data Channel Generic Behavior

The tests in this section will verify the overall behavior of a component when a data channel is in operation. The transport protocols themselves will not be tested. The Isochronous protocol will systematically be used because it does not imply any specific behavior.

All the tests described in this section will need to be customized to reflect the actual configuration of the Device Under Test (Port numbering and capabilities).

The tests will mainly address the data port behavior and the way devices control their data ports. The main reference for the data port behavior is the Port State Diagram as found in figure 80of the SLIMbus specification.



Ports are configured by a collection of messages.

The tests will divide in 2 sections: the ones for the source (transmitter) ports and the one for the sink (receiver) ports.

5.1.1. Source Port

5.1.1.1. DGT1 - Isochronous Data Channel Setup

Test Description:

The test will verify the proper setup of an Isochronous data channel. The Isochronous transport protocol is the simplest protocol in SLIMbus. It does not use any TAG bits (no flow control, no flow information). The device under test can either be a data source (for instance an ADC), a data sink (for instance a DAC DAC) or both (for instance a Mixer).

In this test, we assume that the DUT is capable of sourcing data. The actual port number depends on the DUT. We will use Port 0 by default. We will create one data channel with a channel rate equal to 48 kHz.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)

Wait 200 superframes then send:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Coverage:

Section	Statement
9.1	It [the channel definition] also determines the Transport Protocol that **shall** be used in the Data Channel.
9.2	The Channel Definition and Content Definition **shall** be broadcast through the Message Channel.
9.3.1	There are no TAG bits used [in the Isochronous protocol], therefore the length of the TAG field **shall** be zero Slots.
9.5	The audio samples on the DATA line **shall** use the offset sign and magnitude number format.
10.13.1	Initially, a Port is in the Disconnected state, where it **shall** not transmit or receive any data.
10.13.1	As long as a Port is in this state [Unconfigured], it **shall** not transmit or receive any data.
10.13.1.1	The Connection Messages shall cause the destination Device to start capturing all channel configuration Messages (Table 64) that contain the same Channel Number as the Connection Message.

10.13.1.1	A Port **shall** capture and act upon the Messages that contain the same Channel Number as the last received Connection Message, regardless of its current state.
10.13.1.2.1	After the associated Reconfiguration Boundary, a Device **shall** ensure that the Segment Distribution, Transport Protocol, and Segment Length from the payload of this Message are used for the Data Channel.
10.13.1.2.2	If the Port is active, or becomes active at that moment, the new Content Definition **shall** be used.
10.13.1.2.2	If the Port is in the Configured state and is active, the Content Definition **shall** be used by the Port from the appropriate time
10.13.1.2.4	As soon as a Port has received the Channel Definition, the Content Definition, and either the NEXT_ACTIVATE_CHANNEL or the NEXT_DEACTIVATE_CHANNEL Message, it **shall** move to the Configured state at the appropriate time, which **shall** be the moment at which the last change takes effect.
11.2.1	[CONNECT_SOURCE] The Port **shall** act as the data source for the Data Channel.
12.1	A Device that supports at least one Transport Protocol **shall** also be capable of receiving the following Core Messages and shall perform the action specified for the Message: BEGIN_RECONFIGURATION, CHANGE_CONTENT, CONNECT_SINK (Sink Devices Only), CONNECT_SOURCE (Source Devices Only), DISCONNECT_PORT, NEXT_ACTIVATE_CHANNEL, NEXT_DEACTIVATE_CHANNEL, NEXT_DEFINE_CHANNEL, NEXT_DEFINE_CONTENT, NEXT_REMOVE_CHANNEL, RECONFIGURE_NOW

5.1.1.2. DGT2 - Message Sync Loss and Data Channel operation

Test Description:

When a channel is in operation, a loss of message sync shall not impact the data streaming.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 superframe, then send a REQUEST_INFORMATION(TID=1, EC=0x4008) message with a UDEF in the message response field to the DUT to trigger the loss of message sync.

Positively acknowledge the REPORT_INFORMATION message coming from the Interface device.

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Verify that the data stream is not interrupted by the loss of message sync.

Coverage:

Section	Statement
8.3.3	The [Message Synchronization loss] error **shall** have no effect on the way the Component is using the Data Space.

5.1.1.3. DGT3 - Frame Sync Loss and Data Channel operation

Test Description:

When a channel is in operation, a loss of Frame sync shall reset the data port and stop the data streaming. The data port will move to the *Disconnected* state Re-enabling the data channel without the proper CONNECT_SOURCE message shall fail and remain without any effect.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 superframe, then corrupt the frame sync symbol of 2 consecutive frames.

Positively acknowledge the REPORT_INFORMATION message coming from the Interface device.

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Verify that the data stream is interrupted by the loss of frame sync.

Verify that the second reconfiguration sequence does not re-enable the data stream.

Coverage:

Section	Statement
10.1.2.3	The Component [that fails checking the frame sync over 2 consecutive frames] **shall** cause all of its Devices to reset their Ports, as defined in section 10.4.4.
10.1.2.4	A Component **shall** cause all Devices within the Component to reset their Ports as defined in section 10.4.4 when moving from the SeekingMessageSync or Operational state to the SeekingSuperframeSync state.
10.4.4	[when reset] The Port **shall** move to the Disconnected state in Figure 80.

5.1.1.4. DGT4 - Superframe Sync Loss and Data Channel operation

Test Description:

When a channel is in operation, a loss of Superframe sync shall reset the data port and stop the data streaming. The data port will move to the *Disconnected* state Re-enabling the data channel without the proper CONNECT_SOURCE message shall fail and remain without any effect.

Initial conditions:

The Component is in the *Enumerated* state. The Clock Gear is 9, the Subframe Mode is 19 (4/32) and the Root Frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 Superframe, then corrupt the Superframe sync symbol of 2 consecutive Superframes.

Positively acknowledge the REPORT_INFORMATION message coming from the Interface device.

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Verify that the data stream is interrupted by the loss of Superframe sync.

Verify that the second Reconfiguration Sequence does not re-enable the data stream.

Coverage:

Section	Statement
10.1.2.4	A Component **shall** cause all Devices within the Component to reset their Ports as defined in section 10.4.4 when moving from the SeekingMessageSync or Operational state to the SeekingSuperframeSync state.
10.4.4	[when reset] The Port **shall** move to the Disconnected state in Figure 80.

5.1.1.5. DGT5 - Device Reset and Data Channel operation

Test Description:

When a channel is in operation, a device reset shall also reset the data ports and stop any data streaming from that device. The data port will move to the *Disconnected* state Re-enabling the data channel without the proper CONNECT_SOURCE message shall fail and remain without any effect.

Initial conditions:

The Component is in the *Enumerated* state. The Clock Gear is 9, the Subframe Mode is 19 (4/32) and the Root Frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 Superframe, then send a RESET_DEVICE message to the DUT.
Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Verify that the data stream is interrupted by the device reset.

Verify that the second Reconfiguration Sequence does not re-enable the data stream.

Coverage:

Section	Statement
10.4.3	[device reset] If the Device contains any Ports, the Device **shall** reset these Ports, and as a result stop participating in any ongoing transports.
10.4.3	A Device Reset s**hall** result in the Device performing a Port Reset for all of its Ports (section 10.4.4), if any, and performing a reset of all implemented Core Information Elements (section 7.1.2).
10.4.4	[when reset] The Port **shall** move to the Disconnected state in Figure 80.

5.1.1.6. DGT6 - Multiple Reconfiguration Sequences

Test Description:

A data channel can be setup by one or more Reconfiguration sequences (max 3). The number of sequence shall not make any difference in the creation and activation of the channel. In this test the DUT is source. It could be made sink if the DUT cannot be source.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes and reset the DUT by sending a RESET_DEVICE message.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes and reset the DUT by sending a RESET_DEVICE message.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION

- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes and reset the DUT by sending a RESET_DEVICE message.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level in the 4 sequences, at the end of the last reconfiguration of the group of messages.

Coverage:

Section	Statement
10.13.1	If the necessary parameters are communicated as part of multiple Reconfiguration Sequences, the Port **shall** move to the Configured state at the Reconfiguration Boundary that belongs to the sequence that contains the last set of parameters.
10.13.1.2.2	If the Port remains inactive, the Content Definition **shall** be stored for future use.

5.1.1.7. DGT7 - Port Already Configured

Test Description:

When a port receives a CONNECT_SOURCE message that does not modify the configuration of the port, the port must continue to operate as if the message was never received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 200 Superframes, send to the DUT:

- CONNECT_SOURCE(PN=0, CN=0).

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Coverage:

Section	Statement
10.13.1.1	When an already connected Port receives a repeated Connection Message (a Connection Message that does not change the Port connection), the Port **shall** continue to function as if the repeated Connection Message had not been received.

5.1.1.8. DGT8 - NEXT_ACTIVATE_CHANNEL missing

Test Description:

If the port does not receive the NEXT_ACTIVATE_CHANNEL message, it cannot participate to the channel transactions, even if the other channel setup messages were properly received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- RECONFIGURE_NOW

Wait 200 superframes then send a reconfiguration sequence to activate the channel.

- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel remains inactive after the first Reconfiguration Sequence

Verify that the data channel effectively carries a clean sine wave with the right level after the second Reconfiguration Sequence.

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port **shall** remain inoperable.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port shall not transmit or receive any information in the Data Channel.

5.1.1.9. DGT9 - NEXT_DEFINE_CHANNEL missing

Test Description:

If the port does not receive the NEXT_DEFINE_CHANNEL message, it cannot participate to the channel transactions, even if the other channel setup messages were properly received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel remains inactive after the first Reconfiguration Sequence

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port **shall** remain inoperable.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port shall not transmit or receive any information in the Data Channel.

5.1.1.10. DGT10 - NEXT_DEFINE_CONTENT missing

Test Description:

If the port does not receive the NEXT_DEFINE_CONTENT message, it cannot participate to the channel transactions, even if the other channel setup messages were properly received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel remains inactive after the first Reconfiguration Sequence

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port **shall** remain inoperable.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port shall not transmit or receive any information in the Data Channel.

5.1.1.11. DGT11 - Data Channel Deactivation

Test Description:

Once a data channel is created, it can be paused by sending a NEXT_DEACTIVATE_CHANNEL message. The data ports will keep the channel settings but will stop data transmission or reception. The data channel can simply be reactivated by sending a NEXT_ACTIVATE_CHANNEL message.

This test will focus on the behavior of the source device which must stop driving the cells of the channel segment.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, deactivate the channel 0 and activate channel 1 with the same parameters. Assign another source than DUT to channel 1 (Traffic Generator for instance). No need to have a sink for channel 1. Setup the channel 1 source to output a sine wave at 500 Hz and -6 dBFs level.

- CONNECT_SOURCE(PN=0, CN=1) (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEACTIVATE_CHANNEL(CN=0)
- NEXT_DEFINE_CHANNEL(CN=1, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=1, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=1)
- RECONFIGURE_NOW

After 500 superframes, deactivate the channel 1 and reactivate channel 0.

- BEGIN_RECONFIGURATION
- NEXT_DEACTIVATE_CHANNEL(CN=1)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the DUT feeds a 1 Khz tone at -3 dBFs level in channel 0 during 500 superframes till the deactivation of channel 0.

Verify that the channel 1 carries contention free samples of a tone at 500 Hz and -6 dBFs till the deactivation of channel 1. At the same moment, channel 0 shall resume and the DUT must source the 1kHz and -3 dBFs level in channel 0.

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.3	To temporarily suspend Data Channel operation, the NEXT_DEACTIVATE_CHANNEL **shall** be broadcast by the active Manager.
10.13.3	A Port **shall** not transmit or receive data in a deactivated Data Channel.
10.13.3	A Port **shall** maintain its knowledge of the Data Channel and Content definition, so that a NEXT_ACTIVATE_CHANNEL will resume Data Channel operation.

5.1.1.12. DGT12 - Data Channel Removal

Test Description:

Once a data channel is created, it can be deleted by using the message NEXT_REMOVE_CHANNEL. In this case, the ports using the channel must be reset and must stop all transaction with the channel. The data ports are moving to the *Disconnected* stateThis test will focus on the behavior of the source device which must stop driving the cells of the channel segment.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, remove the channel 0.

- BEGIN_RECONFIGURATION
- NEXT_REMOVE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, send a Reconfiguration sequence to activate channel 0.

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT feeds a 1 KHz tone at -3 dBFs level in channel 0 during 500 Superframes till the removal of channel 0.

Verify that the DUT still does not feed any sample in channel 0 after the channel activation sequence is sent.

Coverage:

Section	Statement
10.13.4	To release the bandwidth allocated to a Data Channel, the active Manager shall broadcast a NEXT_REMOVE_CHANNEL with the relevant Channel Number during one of the reconfiguration sequences.
10.13.4	At the associated Reconfiguration Boundary, all Ports involved in the transport **shall** be reset and return to the Disconnected state (Figure 80).
11.4.13	[NEXT_REMOVE_CHANNEL] A Device shall disconnect and reset any Ports connected to the specified Data Channel at the associated Reconfiguration Boundary.

5.1.1.13. DGT13 - Port Disconnect

Test Description:

When a data port gets disconnected from a data channel, it stops operation on that channel. A source port will not transmit anything anymore on the bus and will reset its configuration.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, send a DISCONNECT_PORT(PN=0) message to the DUT.

After 500 superframes, send a channel reconfiguration sequence without any CONNECT_SOURCE message.

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT feeds a 1 KHz tone at -3 dBFs level in channel 0 during 500 superframes till the disconnection of the port.

Verify the the transmission stops at the beginning of the superframe following the end of the DISCONNECT_PORT message.

Verify that the DUT does not transmit any data after the last reconfiguration sequence.

Coverage:

Section	Statement
10.13.2	The DISCONNECT_PORT Message instructs a Port that it **shall** no longer be involved with a particular Data Channel.
10.13.2	Regardless of the Port state, it **shall** reset and return to the Disconnected state.
10.13.2	This implies that the Port **shall** cease to transmit or receive data in the Data Channel.
10.13.2	In addition, the Port **shall** discard any knowledge of the Data Channel Definition and Content and shall discard the Channel Number
10.13.2	These changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the DISCONNECT_PORT Message.

5.1.1.14. DGT14 - Data Channel Move in Data Space

Test Description:

Once a data channel is created, it can be moved in the data space to better organize the free bandwidth. This is done by using the NEXT_DEFINE_CHANNEL message. The data stream must not be interrupted by such a data channel move. It is implicitly understood that only the segment offset is modified. The channel rate, transport protocol and segment length must remain unchanged.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, move the channel 0.

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, **SD=3148**, TP=0, SL=4)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT feeds a 1 KHz tone at -3 dBFs level in channel 0 during 500 Superframes.

Verify that the DUT still feeds samples in channel 0 after the channel move sequence is sent and that there is no stream interruption around the Reconfiguration Boundary.

Coverage:

Section	Statement
10.13.6	When a Data Channel has to be moved in the Data Space to free some bandwidth or to fit in the new available space left after a subframe mode change, the active Manager **shall** broadcast the NEXT_DEFINE_CHANNEL Message to notify the Devices involved in the transport of the new channel parameters.
11.4.9	[NEXT_DEFINE_CHANNEL] If the Data Channel, CN, is already defined, its attributes **shall** be modified according to the parameters in the Payload field.

5.1.1.15. DGT15 - Channel Parameters and Clock Gear Change

Test Description:

Once a data channel is created, its parameters (Segment Distribution, Segment Length and Transport Protocol) are not modified. For instance, a channel having a rate of 48 kHz in clock gear 9 will have a rate of 24 kHz in clock gear 8 because the segment interval (and therefore the number of segments per Superframe) does not change while the bus frequency is halved.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 Superframes, change the Clock Gear to 8. The bus frequency is halved.

- BEGIN_RECONFIGURATION
- NEXT_CLOCK_GEAR(CG=8)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT feeds a 1 KHz tone at -3 dBFs level in channel 0 during 500 Superframes.

Verify that the DUT still feeds samples in channel 0 after the clock gear change and that the new effective sampling rate is equal to 24 kHz.

Coverage:

Section	Statement
10.5.1.3	In absence of explicit Messages from the active Manager changing the definition of a Data Channel, the Data Channel parameters shall remain constant over a Clock Gear change.

5.1.1.16. DGT16 - Reserved Transport Protocol

Test Description:

A port connected to a channel using a Reserved transport protocol shall not participate to any transactions in that channel. A source port must not write any data in the channel segments. The Reserved values are 8, 9, 10, 11, 12 and 13.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=x, SL=4) for 48 kHz/16 bits - x takes all values from 8 to 13.
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 500 superframes.

Repeat the sequence for each reserved transport protocol.

Success Criterion:

Verify that the DUT never feeds a 1 KHz tone at -3 dBFs level in channel 0.

Coverage:

Section	Statement
9.3	A Device **shall** treat “Reserved” Transport Protocol codes as unsupported Transport Protocols.

5.1.1.17. DGT17 - Reserved Auxiliary Bit Format

Test Description:

A port connected to a channel using a Reserved auxiliary bit format must treat it as unsupported. A source port must write 0 in the AUX slots. The Reserved values are 2 to 10 for a single AUX slot and 12 to 14 for 2 slots.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=5) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=x, DT=1, CL=0, DL=4) - x takes all Reserved value values
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 500 superframes.

Repeat the sequence for each reserved Auxiliary bit formats. For the values 12, 13 and 14, set SL=6 to count for the 2 slots AUX bits.

Success Criterion:

Verify that the DUT writes 0 in the AUX slot(s) and writes the adequate samples value (a sine wave at 1kHz, -3dBFs).

Coverage:

Section	Statement
9.4.3	Device that receives a “Reserved” AUX format codes **shall** behave as if it received an unsupported AUX format code.
9.4.3	If a Data Source does not support the AUX format as specified for a Data Channel, it **shall** fill the AUX field with zeroes.

5.1.1.18. DGT18 - Clock Pause and Streaming

Test Description:

The data port operation shall not be impacted by a clock pause and clock wake-up. Unless a channel reconfiguration was broadcasted in the same Reconfiguration Sequence as the one of the clock pause, the channel will keep the same parameters.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=5) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=x, DT=1, CL=0, DL=4) - x takes all Reserved value values
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 500 superframes, then send a clock pause sequence:

- BEGIN_RECONFIGURATION
- NEXT_PAUSE_CLOCK(RT=2)
- RECONFIGURE_NOW

Wake-up the bus (automatic wake-up from TG, manual wake-up on Audio Bridge).

Success Criterion:

Verify that the DUT transmits a sine wave at 1kHz, -3dBFs before and after the clock pause.

Coverage:

Section	Statement
10.3.2	A channel that is not deactivated by the active Manager **shall** remain active over a Clock Pause.
10.3.2	A Component, its Devices and their Ports **shall** resume normal operation after wake-up as though the Clock Pause never happened.

5.1.1.19. DGT19 - CHANGE_CONTENT instead of NEXT_DEFINE_CONTENT

Test Description:

The NEXT_DEFINE_CONTENT and CHANGE_CONTENT messages have the same effect: they define the content properties of the stream. These message usually do not have visible effect on SLIMbus. However, when a data port is in the *Unconfigured* state, the reception of a CHANGE_CONTENT message (together with the other mandatory messages) will allow the port to move in the Configured state and start transmitting data.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=5) for 48 kHz/16 bits
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes, then send a CHANGE_CONTENT message.

- CHANGE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)

Success Criterion:

Verify that the DUT transmits a sine wave at 1kHz, -3dBFs at the beginning of the Superframe following the end of the CHANGE_CONTENT message.

Coverage:

Section	Statement
10.13.1	If a CHANGE_CONTENT Message is used to broadcast some of the necessary parameters, and the remaining parameters are sent through a Reconfiguration Sequence, the Port **shall** move to the Configured state as soon as all parameters have taken effect, either at the associated Reconfiguration Boundary, or at the first Superframe boundary that occurs at least two Slots after the end of the CHANGE_CONTENT Message, whichever comes last.
10.13.5	The announced content parameters **shall** be used by the source Device from the first Superframe boundary at least two Slots after the transmission of the CHANGE_CONTENT Message.

5.1.1.20. DGT20 - Segment Structure

Test Description:

All the fields of a segment must be left aligned. If the segment is longer than necessary, the slots after the DATA field are unused and shall not be driven by the source port. However, this last requirement is currently not testable and will not be covered by this test. The segment length will be 32 bits while the data length will be 16 bits.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, **SL=8**) for 48 kHz/32 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT transmits the 16 bits samples on the left side of the segment.

Verify that the unused bits of the segments are equal to 0.

Coverage:

Section	Statement
6.4.1	The Segment structure **shall** be MSB aligned within the Segment.
6.4.1	If the Segment Length is longer than the combined length of the TAG, AUX and DATA fields then the Slots after the DATA field are unused. The source Device **shall** not drive the unused Slots.

5.1.1.21. DGT21 - Segment and Message Channel Overlap

Test Description:

A data port must not drive slots that belong to the message channel (control slots), even if the Segment Distribution requires it.

This test is creating a data channel that has some segments whose the two most significant slots are overlapping the control slots.

Note that there should not be any collision as the port will disable its transmission without having to wait for a collision detection.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 24 (8/24) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the DUT to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3142, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the DUT feeds a 1 KHz tone at -3 dBFs level in channel 0, except in the slots overlapping the message channel

Coverage:

Section	Statement
6.4.2	A Component with an output Port **shall** not allow that Port to drive the DATA line during Slots allocated to Control Space, even if the Component receives a Data Channel definition that overlaps Control Space.
10.6.1.3	In this case [conflicts between data channels and Message channel] the Component in which this conflict occurs **shall** give the Message Channel priority over the Data Channel.
10.6.1.3	This implies that a Component **shall** never overwrite the Message Channel with a Segment, even though the overlap is the result of explicit programming by the active Manager.
12.2.5.1	An Interface Device of a Component **shall** set DATA_SLOT_OVERLAP whenever it suppresses the output of a Port within the Component due to Segment overlap with the Control Space or with the output of another Port.
12.2.6	If an Interface Device detects a Data Space Slot overlap, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager at the earliest opportunity.

5.1.2. Sink Port

5.1.2.1. DGR1 - Isochronous Data Channel Setup

The test will verify the proper setup of an Isochronous data channel. The Isochronous transport protocol is the simplest protocol in SLIMbus. It does not use any TAG bits (no flow control, no flow information). The device under test can either be a data source (for instance an ADC), a data sink (for instance a DAC DAC) or both (for instance a Mixer).

In this test, we assume that the DUT is capable of sinking data. The actual port number depends on the DUT. We will use Port 0 by default. We will create one data channel with a channel rate equal to 48 kHz.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to DUT

Wait 200 superframes then send:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level and that the DUT outputs the channel content in the proper way.

Coverage:

Section	Statement
9.1	It [the channel definition] also determines the Transport Protocol that **shall** be used in the Data Channel.
9.2	The Channel Definition and Content Definition **shall** be broadcast through the Message Channel.
9.3.1	There are no TAG bits used [in the Isochronous protocol], therefore the length of the TAG field **shall** be zero Slots.
9.5	The audio samples on the DATA line **shall** use the offset sign and magnitude number format.
10.13.1	Initially, a Port is in the Disconnected state, where it **shall** not transmit or receive any data.
10.13.1	As long as a Port is in this state [Unconfigured], it **shall** not transmit or receive any data.
10.13.1.1	The Connection Messages shall cause the destination Device to start capturing all channel configuration Messages (Table 64) that contain the same Channel Number as the Connection Message.
10.13.1.1	A Port **shall** capture and act upon the Messages that contain the same Channel Number as the last received Connection Message, regardless of its current state.

10.13.1.2.1	After the associated Reconfiguration Boundary, a Device **shall** ensure that the Segment Distribution, Transport Protocol, and Segment Length from the payload of this Message are used for the Data Channel.
10.13.1.2.2	If the Port is active, or becomes active at that moment, the new Content Definition **shall** be used.
10.13.1.2.2	If the Port is in the Configured state and is active, the Content Definition **shall** be used by the Port from the appropriate time
11.2.2	[CONNECT_SINK] The Port **shall** act as a data sink for the Data Channel.
12.1	A Device that supports at least one Transport Protocol shall also be capable of receiving the following Core Messages and shall perform the action specified for the Message: BEGIN_RECONFIGURATION, CHANGE_CONTENT, CONNECT_SINK (Sink Devices Only), CONNECT_SOURCE (Source Devices Only), DISCONNECT_PORT, NEXT_ACTIVATE_CHANNEL, NEXT_DEACTIVATE_CHANNEL, NEXT_DEFINE_CHANNEL, NEXT_DEFINE_CONTENT, NEXT_REMOVE_CHANNEL, RECONFIGURE_NOW

5.1.2.2. DGR2 - Message Sync Loss and Data Channel operation

Test Description:

When a channel is in operation, a loss of message sync shall not impact the data streaming.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 superframe, then send a REQUEST_INFORMATION(TID=1, EC=0x4008) message with a UDEF in the message response field to the DUT to trigger the loss of message sync.

Positively acknowledge the REPORT_INFORMATION message coming from the Interface device.

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level and that the DUT outputs the content of the channel.

Verify that the data stream is not interrupted by the loss of message sync and that the DUT continues to outputs the content of the channel.

Coverage:

Section	Statement
8.3.3	The [Message Synchronization loss] error **shall** have no effect on the way the Component is using the Data Space.

5.1.2.3. DGR3 - Frame Sync Loss and Data Channel operation

Test Description:

When a channel is in operation, a loss of Frame sync shall reset the data port and stop the data streaming. Re-enabling the data channel with only a NEXT_ACTIVATE_CHANNEL message shall fail and remain without any effect.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 superframe, then corrupt the frame sync symbol of 2 consecutive frames.

Positively acknowledge the REPORT_INFORMATION message coming from the Interface device.

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Verify that the DUT stops outputting the channel content after the frame sync loss.

Verify that the second reconfiguration sequence does not re-enable the data output.

Coverage:

Section	Statement
10.1.2.3	The Component [that fails checking the frame sync over 2 consecutive frames] **shall** cause all of its Devices to reset their Ports, as defined in section 10.4.4.
10.1.2.4	A Component **shall** cause all Devices within the Component to reset their Ports as defined in section 10.4.4 when moving from the SeekingMessageSync or Operational state to the SeekingSuperframeSync state.
10.4.4	[when reset] The Port **shall** move to the Disconnected state in Figure 80.

5.1.2.4. DGR4 - Superframe Sync Loss and Data Channel operation

Test Description:

When a channel is in operation, a loss of Superframe sync shall reset the data port and stop the data streaming. Re-enabling the data channel with only a NEXT_ACTIVATE_CHANNEL message shall fail and remain without any effect.

Initial conditions:

The Component is in the *Enumerated* state. The Clock Gear is 9, the Subframe Mode is 19 (4/32) and the Root Frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 Superframe, then corrupt the Superframe sync symbol of 2 consecutive Superframes.

Positively acknowledge the REPORT_INFORMATION message coming from the Interface device.

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AS=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Verify that the DUT stops outputting the channel content after the superframe sync loss.

Verify that the second reconfiguration sequence does not re-enable the data output.

Coverage:

Section	Statement
10.1.2.4	A Component **shall** cause all Devices within the Component to reset their Ports as defined in section 10.4.4 when moving from the SeekingMessageSync or Operational state to the SeekingSuperframeSync state.
10.4.4	[when reset] The Port **shall** move to the Disconnected state in Figure 80.

5.1.2.5. DGR5 - Device Reset and Data Channel operation

Test Description:

When a channel is in operation, a device reset shall also reset the data port and stop the data streaming. Re-enabling the data channel with only a NEXT_ACTIVATE_CHANNEL message shall fail and remain without any effect.

Initial conditions:

The Component is in the *Enumerated* state. The Clock Gear is 9, the Subframe Mode is 19 (4/32) and the Root Frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AS=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 100 Superframe, then send a RESET_DEVICE message to the DUT.
Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level.

Verify that the DUT stops outputting the channel content after the DEVICE_RESET is sent.

Verify that the second reconfiguration sequence does not re-enable the data output.

Coverage:

Section	Statement
10.4.3	[device reset] If the Device contains any Ports, the Device **shall** reset these Ports, and as a result stop participating in any ongoing transports.
10.4.3	A Device Reset s**hall** result in the Device performing a Port Reset for all of its Ports (section 10.4.4), if any, and performing a reset of all implemented Core Information Elements (section 7.1.2).
10.4.4	[when reset] The Port **shall** move to the Disconnected state in Figure 80.

5.1.2.6. DGR6 - Multiple Reconfiguration Sequences

Test Description:

A data channel can be setup by one or more Reconfiguration sequences (max 3). The number of sequence shall not make any difference in the creation and activation of the channel. In this test the DUT is source. It could be made sink if the DUT cannot be source.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes and reset the DUT by sending a RESET_DEVICE message.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes and reset the DUT by sending a RESET_DEVICE message.
Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- RECONFIGURE_NOW

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes and reset the DUT by sending a RESET_DEVICE message.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to the DUT.
- CONNECT_SINK(PN=0,CN=0) sent to test gear (optional if Traffic Generator)
- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel effectively carries a clean sine wave with the right level in the 4 sequences and that the DUT outputs the content of the channel, at the end of the last reconfiguration of the group of messages.

Coverage:

Section	Statement
10.13.1	If the necessary parameters are communicated as part of multiple Reconfiguration Sequences, the Port **shall** move to the Configured state at the Reconfiguration Boundary that belongs to the sequence that contains the last set of parameters.

5.1.2.7. DGR7 - Port Already Configured

Test Description:

When a port receives a CONNECT_x message that does not modify the configuration of the port, the port must continue to operate as if the message was never received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 Superframes then send:

- CONNECT_SINK(PN=0, CN=0) sent to the DUT.

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the DUT outputs the sine wave without any interruption.

Coverage:

Section	Statement
10.13.1.1	When an already connected Port receives a repeated Connection Message (a Connection Message that does not change the Port connection), the Port **shall** continue to function as if the repeated Connection Message had not been received.

5.1.2.8. DGR8 - NEXT_ACTIVATE_CHANNEL missing

Test Description:

If the port does not receive the NEXT_ACTIVATE_CHANNEL message, it cannot participate to the channel transactions, even if the other channel setup messages were properly received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. The TG must transmit the data irrespectively of the transmitted messages. The goal is to verify that the DUT will not read and use data.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- RECONFIGURE_NOW

wait 1000 superframes
Send a reconfiguration sequence to activate the channel.

- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel remains inactive after the first Reconfiguration Sequence and that the DUT does not output anything.

Verify that the data channel effectively carries a clean sine wave with the right level after the second Reconfiguration Sequence and that the DUT outputs the content of the data channel.

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port **shall** remain inoperable.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port shall not transmit or receive any information in the Data Channel.

5.1.2.9. DGR9 - NEXT_DEFINE_CHANNEL missing

Test Description:

If the port does not receive the NEXT_DEFINE_CHANNEL message, it cannot participate to the channel transactions, even if the other channel setup messages were properly received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. The TG must transmit the data irrespectively of the transmitted messages. The goal is to verify that the DUT will not read and use data.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel remains inactive after the Reconfiguration Sequence and that the DUT does not output anything.

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port **shall** remain inoperable.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port shall not transmit or receive any information in the Data Channel.

5.1.2.10. DGR10 - NEXT_DEFINE_CONTENT missing

Test Description:

If the port does not receive the NEXT_DEFINE_CONTENT message, it cannot participate to the channel transactions, even if the other channel setup messages were properly received.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. The TG must transmit the data irrespectively of the transmitted messages. The goal is to verify that the DUT will not read and use data.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the data channel remains inactive after the Reconfiguration Sequence

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port **shall** remain inoperable.
10.13.1.2.3	If, for any reason, one or more of the channel definition, content definition and channel activation Messages have not been received, the Port shall not transmit or receive any information in the Data Channel.

5.1.2.11. DGR11 - Data Channel Deactivation

Test Description:

Once a data channel is created, it can be paused by sending a NEXT_DEACTIVATE_CHANNEL message. The data ports will keep the channel settings but will stop data transmission or reception. The data channel can simply be reactivated by sending a NEXT_ACTIVATE_CHANNEL message.

This test will focus on the behavior of the sink device which must stop reading the channel segments.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. The TG must transmit the data irrespectively of the transmitted messages. One solution is to create channel 1 with the same parameters than channel 0 and to activate it at the same moment channel 0 is deactivated. Channel 1 is only fed by the Traffic Generator. Obviously, when reactivating channel 0, channel 1 must be deactivated.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, deactivate the channel 0

- BEGIN_RECONFIGURATION
- NEXT_DEACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, reactivate channel 0.

- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that all the messages are positively acknowledged.

Verify that the DUT outputs a 1 KHz tone at -3 dBFs level from channel 0 during 500 Superframes till the deactivation of channel 0.

500 Suprframes later, verify that the channel 0 resumes and that the DUT outputs the 1kHz tone at -3 dBFs level it gets from channel 0.

Coverage:

Section	Statement
10.13.1.2.3	The active Manager **shall** broadcast the NEXT_ACTIVATE_CHANNEL Message as part of a Reconfiguration Sequence to notify the Devices involved in a transport that the Data Channel shall be activated at the associated Reconfiguration Boundary.
10.13.3	To temporarily suspend Data Channel operation, the NEXT_DEACTIVATE_CHANNEL **shall** be broadcast by the active Manager.
10.13.3	A Port **shall** not transmit or receive data in a deactivated Data Channel.
10.13.3	A Port **shall** maintain its knowledge of the Data Channel and Content definition, so that a NEXT_ACTIVATE_CHANNEL will resume Data Channel operation.

5.1.2.12. DGR12 - Data Channel Removal

Test Description:

Once a data channel is created, it can be deleted by using the message NEXT_REMOVE_CHANNEL. In this case, the ports using the channel must be reset and stop all transaction with the channel. This test will focus on the behavior of the sink device which must stop reading the channel segments.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. The TG must transmit the data irrespectively of the transmitted messages. The goal is to verify that the DUT will not read and use data.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, remove the channel 0.

- BEGIN_RECONFIGURATION
- NEXT_REMOVE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, send a Reconfiguration sequence to activate channel 0.

- BEGIN_RECONFIGURATION
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT outputs a 1 KHz tone at -3 dBFs level it gets from channel 0 during 500 Superframes till the removal of channel 0.

500 Suprframes later, verify that the channel 0 does not resumes and that the DUT still does not output any data.

Coverage:

Section	Statement
10.13.4	To release the bandwidth allocated to a Data Channel, the active Manager **shall** broadcast a NEXT_REMOVE_CHANNEL with the relevant Channel Number during one of the reconfiguration sequences.
10.13.4	At the associated Reconfiguration Boundary, all Ports involved in the transport **shall** be reset and return to the Disconnected state (Figure 80).
11.4.13	[NEXT_REMOVE_CHANNEL] A Device **shall** disconnect and reset any Ports connected to the specified Data Channel at the associated Reconfiguration Boundary.

5.1.2.13. DGR13 - Port Disconnect

Test Description:

When a data port gets disconnected from a data channel, it stops operation on that channel. A sink port will not capture anything anymore on the bus and will reset its configuration.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. The TG must transmit the data irrespectively of the transmitted messages. The goal is to verify that the DUT will not read and use data.

Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, send a DISCONNECT_PORT(PN=0) message to the DUT.

After 500 superframes, send a channel reconfiguration sequence without any CONNECT_SINK message.

- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT receives a 1 KHz tone at -3 dBFs level from channel 0 during 500 Superframes till the disconnection of the port.

Verify that the DUT stops receiving data at the beginning of the Superframe following the end of the DISCONNECT_PORT message.

Verify that the DUT does not receive any data after the last reconfiguration sequence.

Coverage:

Section	Statement
10.13.2	The DISCONNECT_PORT Message instructs a Port that it **shall** no longer be involved with a particular Data Channel.
10.13.2	Regardless of the Port state, it **shall** reset and return to the Disconnected state.
10.13.2	This implies that the Port **shall** cease to transmit or receive data in the Data Channel.
10.13.2	In addition, the Port **shall** discard any knowledge of the Data Channel Definition and Content and shall discard the Channel Number
10.13.2	These changes **shall** take effect at the first Superframe boundary that comes at least two Slots after the end of the DISCONNECT_PORT Message.

5.1.2.14. DGR14 - Data Channel Move in Data Space

Test Description:

Once a data channel is created, it can be moved in the data space to better organize the free bandwidth. This is done by using the NEXT_DEFINE_CHANNEL message. The data stream must not be interrupted by such a data channel move. It is implicitly understood that only the segment offset is modified. The channel rate, transport protocol and segment length must remain unchanged.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:
- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, move the channel 0.
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, **SD=3148**, TP=0, SL=4)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT received a 1 KHz tone at -3 dBFs level from channel 0 during 500 superframes.

Verify that the DUT still receives samples from channel 0 after the channel move sequence is sent and that there is no interruption around the Reconfiguration Boundary.

Coverage:

Section	Statement
10.13.6	When a Data Channel has to be moved in the Data Space to free some bandwidth or to fit in the new available space left after a subframe mode change, the active Manager **shall** broadcast the NEXT_DEFINE_CHANNEL Message to notify the Devices involved in the transport of the new channel parameters.
11.4.9	[NEXT_DEFINE_CHANNEL] If the Data Channel, CN, is already defined, its attributes **shall** be modified according to the parameters in the Payload field.

5.1.2.15. DGR15 - Channel Parameters and Clock Gear Change

Test Description:

Once a data channel is created, its parameters (Segment Distribution, Segment Length and Transport Protocol) are not modified. For instance, a channel having a rate of 48 kHz in clock gear 9 will have a rate of 24 kHz in clock gear 8 because the segment interval (and therefore the number of segments per Superframe) does not change while the bus frequency is halved.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=4) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

After 500 superframes, change the Clock Gear to 8. The bus frequency is halved.

- BEGIN_RECONFIGURATION
- NEXT_CLOCK_GEAR(CG=8)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT receives a 1 KHz tone at -3 dBFs level in channel 0 during 500 superframes.

Verify that the DUT still receives samples in channel 0 after the clock gear change and that the new effective sampling rate is equal to 24 kHz.

Coverage:

Section	Statement
10.5.1.3	In absence of explicit Messages from the active Manager changing the definition of a Data Channel, the Data Channel parameters shall remain constant over a Clock Gear change.

5.1.2.16. DGR16 - Reserved Transport Protocol

Test Description:

A port connected to a channel using a Reserved transport protocol shall not participate to any transactions in that channel. A sink port must not read any data from the channel segments. The Reserved values are 8, 9, 10, 11, 12 and 13.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=x, SL=4) for 48 kHz/16 bits - x takes all values from 8 to 13.
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 500 superframes.

Repeat the sequence for each reserved transport protocol.

Success Criterion:

Verify that the DUT never outputs any data.

Coverage:

Section	Statement
9.3	A Device **shall** treat "Reserved" Transport Protocol codes as unsupported Transport Protocols.

5.1.2.17. DGR17 - Reserved Auxiliary Bit Format

Test Description:

A port connected to a channel using a Reserved auxiliary bit format must treat it as unsupported. A sink port must ignore the content of the AUX slots. The Reserved values are 2 to 10 for a single AUX slot and 12 to 14 for 2 slots.

We can only test the fact that the data are properly used. What happens to the AUX bits is not verifiable.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. The AUX slots will see their value cycling from 0 to 15 or from 0 to 255. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=5) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=x, DT=1, CL=0, DL=4) - x takes all Reserved value values
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 500 superframes.

Repeat the sequence for each reserved Auxiliary bit formats. For the values 12, 13 and 14, set SL=6 to count for the 2 slots AUX bits.

Success Criterion:

Verify that the DUT outputs a sine wave at 1kHz, -3dBFs.

Coverage:

Section	Statement
9.4.3	Device that receives a “Reserved” AUX format codes **shall** behave as if it received an unsupported AUX format code.
9.4.3	If a Data Sink does not support the AUX format as specified for a Data Channel, the Data Sink **shall** ignore the contents of the AUX field.

5.1.2.18. DGR18 - Clock Pause and Streaming

Test Description:

The data port operation shall not be impacted by a clock pause and clock wake-up. Unless a channel reconfiguration was broadcasted in the same Reconfiguration Sequence as the one of the clock pause, the channel will keep the same parameters.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=5) for 48 kHz/16 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=x, DT=1, CL=0, DL=4) - x takes all Reserved value values
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 500 superframes, then send a clock pause sequence:

- BEGIN_RECONFIGURATION
- NEXT_PAUSE_CLOCK(RT=2)
- RECONFIGURE_NOW

Wake-up the bus (automatic wake-up from TG, manual wake-up on Audio Bridge).

Success Criterion:

Verify that the DUT outputs a sine wave at 1kHz, -3dBFs (read from the channel segments) before and after the clock pause.

Coverage:

Section	Statement
10.3.2	A channel that is not deactivated by the active Manager **shall** remain active over a Clock Pause.
10.3.2	A Component, its Devices and their Ports **shall** resume normal operation after wake-up as though the Clock Pause never happened.

5.1.2.19. DGR19 - CHANGE_CONTENT instead of NEXT_DEFINE_CONTENT

Test Description:

The NEXT_DEFINE_CONTENT and CHANGE_CONTENT messages have the same effect: they define the content properties of the stream. These message usually do not have visible effect on SLIMbus. However, when a data port is in the *Unconfigured* state, the reception of a CHANGE_CONTENT message (together with the other mandatory messages) will allow the port to move in the Configured state and start transmitting data.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, SL=5) for 48 kHz/16 bits
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Wait 200 superframes, then send a CHANGE_CONTENT message.

- CHANGE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)

Success Criterion:

Verify that the DUT receives a sine wave at 1kHz, -3dBFs at the beginning of the Superframe following the end of the CHANGE_CONTENT message.

Coverage:

Section	Statement
10.13.1	If a CHANGE_CONTENT Message is used to broadcast some of the necessary parameters, and the remaining parameters are sent through a Reconfiguration Sequence, the Port **shall** move to the Configured state as soon as all parameters have taken effect, either at the associated Reconfiguration Boundary, or at the first Superframe boundary that occurs at least two Slots after the end of the CHANGE_CONTENT Message, whichever comes last.

5.1.2.20. DGR20 - Segment Structure

Test Description:

All the fields of a segment must be left aligned. If the segment is longer than necessary, the slots after the DATA field are unused and shall not be driven by the source port. However, this last requirement is currently not testable and will not be covered by this test. The segment length will be 32 bits while the data length will be 16 bits.

Initial conditions:

The Component is in the *Enumerated* state. The clock gear is 9, the subframe mode is 19 (4/32) and the root frequency is 1 (24.576 MHz).

Stimulus:

Configure the Traffic Generator (TG) to transmit a 1kHz sine wave at -3dBFs in channel 0. Send the following Reconfiguration messages:

- CONNECT_SOURCE(PN=0, CN=0) sent to test gear (optional if TG)
- CONNECT_SINK(PN=0,CN=0) sent to the DUT
- BEGIN_RECONFIGURATION
- NEXT_DEFINE_CHANNEL(CN=0, SD=3140, TP=0, **SL=8**) for 48 kHz/32 bits
- NEXT_DEFINE_CONTENT(CN=0, FL=1, PR=3, AF=0, DT=1, CL=0, DL=4)
- NEXT_ACTIVATE_CHANNEL(CN=0)
- RECONFIGURE_NOW

Success Criterion:

Verify that the DUT transmits the 16 bits samples on the left side of the segment.

Verify that the unused bits of the segments are equal to 0.

Coverage:

Section	Statement
6.4.1	The Segment structure **shall** be MSB aligned within the Segment.
6.4.1	If the Segment Length is longer than the combined length of the TAG, AUX and DATA fields then the Slots after the DATA field are unused. The sink Device **shall** ignore the unused Slots.

5.2. Pushed Transport Protocol

5.3. Pulled Transport Protocol

6. Specific Behavior and Error Cases

6.1. Reset Strategies

There exists many reset possibilities in SLIMbus:

- Announced bus reset through a reconfiguration sequence using the NEXT_RESET_BUS message.
- Component Reset through a RESET_DEVICE message sent to the Interface device of the component.
- Unannounced bus reset through a long enough series of clock cycles without any data activity
- A simple Device reset by using the DEVICE_RESET message.

6.1.1. RS1 - Bus Reset

Test Description:

Upon reception of a valid reconfiguration sequence containing the NEXT_RESET_BUS message, all the devices will go back to their *Reset* state at the Reconfiguration boundary and the active Framer (if any present) will start a boot sequence. At the end of the boot sequence, the devices will be again in the *EnumerationReset* state. All the devices present on the bus must send a REPORT_PRESENT message.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_RESET_BUS
- RECONFIGURE_NOW

Success Criterion:

Verify that all the devices present in the Component are issuing their REPORT_PRESENT message.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer performs adequately the boot process.

Coverage:

Section	Statement
10.4.1	At the Reconfiguration Boundary that is associated with the NEXT_RESET_BUS announcement, the active Framer **shall** leave the Booted state as shown in Figure 68 (BusReset transition) and repeat its boot sequence starting at the StartingClock state (see section 10.1.1.3 and beyond).
12.2.9	Note that if, and when, a SLIMbus Reset takes effect, the Device **shall** reset as a result of the process described in section 10.4.
12.4.9	When the active Framer receives a NEXT_RESET_BUS Message, it **shall** prepare to perform a Bus Reset as described in section 10.4.
12.5.11	When a Generic Device receives a Component Reset by the mechanisms described in section 10.4.2, it **shall** perform the actions specified in that section.

6.1.2. RS2 - Bus Reset followed by speed enumeration

Test Description:

Upon reception of a valid reconfiguration sequence containing the NEXT_RESET_BUS message, all the devices will go back to their *Reset* state at the Reconfiguration boundary and the active Framer (if any present) will start a boot sequence. At the end of the boot sequence, the devices will be again in the *EnumerationReset* state.

However, this time, instead of waiting for the REPORT_PRESENT messages, the logical address assignment will be done directly, forcing the devices to move from the *EnumerationReset* to the *Enumerated* state through the HaveLA transition (see figure 77 of the SLIMbus specification).

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_RESET_BUS
- RECONFIGURE_NOW

As soon as in the Operational state, assign Logical Addresses directly to all the devices of the Component without letting device to send a REPORT_PRESENT message.

Send a REQUEST_INFORMATION(TID=n, EC=0x0008) to each device, with n=LA.

Success Criterion:

Verify that all the REQUEST_INFORMATION messages are positively acknowledged.

Verify that the REPLY_INFORMATION messages is sent with the right TID (equal to the source LA).

Coverage:

Section	Statement
10.9	If a Device has received a Logical Address, i.e. has been enumerated, it **shall** enter the Enumerated state directly (HaveLA transition).

6.1.3. RS3 - Bus Reset followed by a Reconfiguration Sequence

Test Description:

Upon reception of a valid reconfiguration sequence containing the NEXT_RESET_BUS message, all the devices will go back to their *Reset* state at the Reconfiguration boundary and the active Framer (if any present) will start a boot sequence. At the end of the boot sequence, the devices will be again in the *EnumerationReset* state.

The devices will move to the *TryingToAnnounce* state. However, it is possible that a reconfiguration sequence takes place before a device has a chance to issue its REPORT_PRESENT message. Devices must follow the Reconfiguration sequences to be able to transmit the REPORT_PRESENT message.

While being in the *WaitingForLA* state, a device must track all the Reconfiguration sequences in order to stay in sync and be able to receive a LA assignment message.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode (SM) is equal to is equal to 19

Stimulus:

Send the following Reconfiguration messages:

- BEGIN_RECONFIGURATION
- NEXT_RESET_BUS
- RECONFIGURE_NOW

As soon as in the Operational state, transmit a Reconfiguration sequence to modify the subframe mode.

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE (7)
- RECONFIGURE_NOW

Positively Acknowledge the REPORT_PRESENT messages.

transmit a Reconfiguration sequence to modify again the subframe mode.

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE (9)
- RECONFIGURE_NOW

Assign Logical Addresses to all the devices of the Component.

Success Criterion:

Verify that all the devices present in the Component are issuing their REPORT_PRESENT message.

Verify that the ASSIGN_LOGICAL_ADDRESS messages all get a PACK in the Message Response field.

Coverage:

Section	Statement
10.9.1.2	Although a Device in the TryingToAnnounce state cannot be the destination of a unicast Message, it **shall** still observe broadcast Messages and be prepared for a change in the bus configuration, so as not to disrupt normal bus operation while sending the REPORT_PRESENT Message.
10.9.1.3	Although a Device in the WaitingForLA state cannot be the destination of a unicast Message, it **shall** still observe broadcast Messages and be prepared for changes in bus configuration.

6.1.4. RS4 - Component Reset

Test Description:

To reset a specific component, the active Manager must send a RESET_DEVICE message to the Interface device of the component.

The devices belonging to the component will move to the *Reset* state and lose their logical address (move to *EnumerationReset*).

All the devices will reacquire all levels of synchronization and send a REPORT_PRESENT message.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send the RESET_DEVICE to the Interface device of the component.

Positively Acknowledge the incoming REPORT_PRESENT messages.

Success Criterion:

Verify that all the devices present in the Component are issuing their REPORT_PRESENT message.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer performs adequately the boot process.

Coverage:

Section	Statement
10.4.2	To perform a Component Reset, the active Manager **shall** send a RESET_DEVICE Message to the Interface Device of the Component that it wants to reset.
10.4.2	A Component in the Operational state that receives this Message **shall** move to the Reset state (Figure 57).
10.4.2	The Clock Sourcing Component **shall** move its active Framer to the StartingClock state (Figure 55).
10.4.2	A Clock Receiving Component **shall** perform the Reset without disturbing the bus operation
10.4.2	Following a Component Reset, the Component **shall** follow the process defined in section 10.1.2 or 10.1.3 to resynchronize to the bus.
12.2.10	When an Interface Device receives a RESET_DEVICE Message, it **shall** perform the following actions in addition to those specified for all Devices in section 10.4.3: The Interface Device **shall** cause all Devices within the same Component to perform a reset as described in section 10.4.2.
12.2.11	When an Interface Device receives a Component Reset by the mechanisms described in section 10.4.2, it **shall** perform the actions specified in that section.

6.1.5. RS5 - Device Reset

Test Description:

To reset a specific device, the active Manager must send a RESET_DEVICE message to that device.

The reset will clear all clearable IE and reset the other ones. To test the IE reset, the only method is to force the device to set the UNSPRTD_MSG IE as this is the only one that is mandatory for all devices and that can easily be triggered by the transmission of an unsupported message.

A device reset will not cause a device to release its Logical Address.

Note that a device reset will also reset the data ports. But this not going to be tested here.

The test has to be repeated for all devices but the Interface Device.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send a Destination-Referred User message to the device under test:

- Message Type (MT) = 0b110
- Message Code (MC) = 0x01
- Destination Type (DT) = 0b00
- Payload length = 0

Send a REQUEST_INFORMATION(TID=1, EC=0x0008) to the device under test.

Send the RESET_DEVICE to the device under test.

Send a REQUEST_INFORMATION(TID=2, EC=0x0008) to the device under test.

Positively acknowledge the REPLY_INFORMATION message.

Success Criterion:

Verify that the first REPLY_INFORMATION message is sent with the right TID.

Verify that the UNSPRTD_MSG Information Element is set.

Verify that the second REPLY_INFORMATION message is sent with the right TID.

Verify that the UNSPRTD_MSG Information Element is reset.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer continues clocking the bus without any interruptions.

Coverage:

Section	Statement
10.4.3	A Device Reset **shall** result in the Device performing a Port Reset for all of its Ports (section 10.4.4), if any, and performing a reset of all implemented Core Information Elements (section 7.1.2).
10.4.3	As the result of a reset, a Device **shall** change all implemented Core Information Elements to their reset values.
10.4.3	If the Device Reset is not part of a Component Reset, the Component **shall** remain in the Operational state.
12.1	A Device **shall** support the following Core Information Elements: DATA_TX_COL, DEVICE_CLASS, DEVICE_CLASS_VERSION, **UNSPRTD_MSG**
12.2.3	An Interface Device **shall** support all Core Information Elements in the “Mandatory” column, and may also support any Core Information Element in the “Optional” column, of Table 68.
12.3.3	A Manager **shall** support all Core Information Elements in the “Mandatory” column, and may also support any Core Information Element in the “Optional” column, of Table 71.
12.4.3	A Framer **shall** support all Core Information Elements in the “Mandatory” column, and may also support any Core Information Element in the “Optional” column, of Table 74.
12.4.10	When the active Framer receives a RESET_DEVICE Message, it shall perform the following actions in addition to those specified for all Devices in section 10.4.3: The active Framer **shall** continue clocking the bus and continue writing the Framing Channel and Guide Channel.
12.5.3	A Generic Device **shall** support all Core Information Elements in the “Mandatory” column, and may also support any Core Information Element in the “Optional” column, of Table 78.

6.1.6. RS6 - Device Reset and Reconfiguration Sequence

Test Description:

When a device is reset, it must discard any announced changes. Any Reconfiguration Sequence effects must be ignored.

Changing the subframe mode is an easy way to validate the behavior. Changing from subframe mode 19 to 7 will modify the position of the Guide Byte slots. The device will lose the message sync. The Framing Information is also modified without notice from a device point of view and it will lose the superframe sync.

The Interface device of the component will report the loss of sync by sending a REPORT_INFORMATION(EC=0x4008) message.

If the device under test is the active Framer, the framing information must not reflect the announced values. This will also cause all the devices present on the bus to lose the superframe sync.

Initial conditions:

The Component is in the *Enumerated* state. The subframe mode is equal to 19.

Stimulus:

Send the following message sequence:

- BEGIN_RECONFIGURATION
- NEXT_SUBFRAME_MODE (7)
- RESET_DEVICE sent to the device under test.
- RECONFIGURE_NOW

Positively acknowledge the incoming REPORT_INFORMATION

Success Criterion:

Verify that the REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device.

Verify that the LOST_SFS Information Element is set.

If the Component is a Clock Sourcing Component (it contains an active Framer), verify that the active Framer does not modify the framing information content.

Coverage:

Section	Statement
10.4.3	[device reset] Any announced changes **shall** be lost by the Device.
10.4.3	[device reset] The Device **shall** act as though it has not received a BEGIN_RECONFIGURATION Message.

6.1.7. RS7 - Low Level Reset Mechanism

Test Description:

A component will reset if it detects DATA line inactivity for 1536 consecutive Cells. The reset can occur after a minimum of 768 cells.

The test can only be ran on Clock Receiving Component as this is not a controllable Framer behavior.

The Component will be in the *Operational* state. To move back to the *SeekingFrameSync* state, it will need to see 2 consecutive erroneous frame sync symbol. That will take 193 slots (772 bits or clock cycles). Therefore, the Component shall not reset before 772+768=1540 clock cycles but must have preformed its reset before 772+1536=2308 clock cycles.

After 1539 clock cycles without data activity, the Component will reacquire all the sync levels and the Interface Device will transmit a REPORT_INFORMATION (EC=0x4008) indicating that the frame sync was lost.

After 2308 clock cycles without data activity, the Component will be reset and will enter the boot sequence. The Component devices will issue their REPORT_PRESENT messages.

Initial conditions:

The Component is in the *Enumerated* state.

Stimulus:

Send 1539 clock cycles without any DATA line activity and then resume with frame sync, Guide Byte and Framing Information. Positively acknowledge the REPORT_INFORMATION message sent by the Interface device.

Send 2308 clock cycles without any DATA line activity and then resume with frame sync, Guide Byte and Framing Information. Positively acknowledge the REPORT_PRESENT messages sent by all the devices.

Success Criterion:

Verify that the REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device.

Verify that the IE LOST_FS is set.

Verify that the REPORT_PRESENT messages are sent by all the devices of the Component (but the active Manager if the Component has one).

Coverage:

Section	Statement
10.1.2.2	If a Component moves to the Reset state from the SeekingFrameSync or Operational states, the Component **shall** cause each Device in the Component to undergo a Device Reset (section 10.4.3) and release its Logical Address.
10.1.2.3	The Component **shall** move to the Reset state if it detects DATA line inactivity (no transitions on the DATA line) for 1536 consecutive Cells (Figure 57, NoDataLineActivity transition).
10.1.2.3	However, the Component **shall** not move to the Reset state before it detects DATA line inactivity of at least 768 consecutive Cells.

6.2. Synchronization Acquisition and Recovery

SLIMbus has 3 levels of synchronization:

- The Frame synchronization, which is the lowest level of SLIMbus synchronization. It allows the component to form nibbles (4 bits block) out of the incoming bit stream. It also indicates where the Framing Channel is located. Frame synchronization is achieved by locking on the frame sync symbol (0b1011).
- The Superframe synchronization, that allows the component to decode the Framing Information and to build a slot map (Control, Data, ...). Superframe synchronization is achieved by looking at the punctured superframe sync pattern in the Framing Information.
- The Message Synchronization, that allows a device to reliably transmit a message in the Message Channel. The Message synchronization is achieved by looking at the Guide Byte and its content.

These 3 levels of synchronization can be lost at any time by any devices. The following tests are meant to validate all the synchronization recovery mechanisms.

Note that these tests can only be run on Clock Receiving Components (not having the active Framer). The devices in a Clock Sourcing Component are directly synchronized by the on-chip Framer and therefore cannot lose synchronization.

6.2.1. SR1 - Frame Sync Pattern Error

Test Description:

A device that has acquired the frame sync must continuously monitor the frame sync symbol. If the verification fails once, it must be ignored (it can be an isolated error). If the verification fails twice (because of a clock glitch, for instance), the device must lose message sync, superframe sync and must reacquire the frame sync. The loss of sync must happen no more than 764 slots after the second corrupted frame sync symbol.

In operation, the loss of frame sync will result in the interface device to transmit a REPORT_INFORMATION (EC=0x4008). The LOST FS will be set.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The FE bit of the Framing Information is equal to 0. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Corrupt the Frame sync symbol at the beginning of a superframe. Any slot value other than **0b1011** is suitable.

Wait 10 superframes with no message traffic and no errors.

Corrupt 2 consecutive frame sync symbols at the beginning of a superframe. Any slot value other than **0b1011** is suitable.

At the beginning of the frame following corrupted frame sync symbol (764 slots later, then), transmit a REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface Device.

Positively acknowledge the incoming REPORT_INFORMATION message.

Success Criterion:

Verify that the Interface Device does not send any REPORT_INFORMATION after the isolated frame sync symbol error.

Verify that the REQUEST_INFORMATION message is not acknowledged (NORE in the message response field).

Verify that the REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device after the 2 erroneous frame sync symbols.

Verify that the LOST_FS is set.

Coverage:

Section	Statement
6.3.1	Components **shall** use the Frame Sync symbol to lock onto the Frame Structure.
10.1	A Component **shall** implement the necessary recovery mechanisms to deal with the loss of Superframe synchronization, **Frame synchronization**, and Message synchronization.
10.1.2.3	A Component that has achieved Frame synchronization **shall** continuously verify Frame Sync symbols.
10.1.2.3	If verification fails in two consecutive Frames, a Component **shall** move to the SeekingFrameSync state (Figure 57, FrameSyncLost transitions).
10.1.2.3	Isolated errors in the verification of Frame Sync symbols **shall** be ignored.
10.1.2.3	The Component [that fails checking the frame sync over 2 consecutive frames] **shall** make this move [to the SeekingFrameSync state] no more than 764 CLK periods after the second Frame Sync symbol verification failure.
11.3.5	[REPORT_INFORMATION] The Information Slice **shall** be placed in the LSBs of the relevant payload byte(s).
11.3.5	[REPORT_INFORMATION] Any unused bits **shall** be filled with zeros.
12.2.5	Each Information Element **shall** have the attributes identified for it in Table 69.
12.2.5.4	The Interface Device of a Component **shall** set LOST_FS whenever the Component moves to the SeekingFrameSync state from a state other than the Reset state (Figure 57).
12.2.6	If an Interface Device detects that its Component has lost Frame synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.2. SR2 - Superframe Sync Pattern Error

Test Description:

A device that has acquired the superframe sync must continuously monitor the superframe sync punctured symbol. If the verification fails once, it must be ignored (it can be an isolated error). If the verification fails twice (because of a clock glitch, for instance), the device must reacquire the superframe sync.

In operation, the loss of frame sync will result in the interface device to transmit a REPORT_INFORMATION (EC=0x4008). The LOST_SFS will be set.

If the superframe sync cannot be reacquired within 4 superframes (32 frames), the component is likely not locked to the right frame sync symbol (pathological case) and must transition to the *SeekingFrameSync* state to reacquire reliably the frame sync.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Corrupt the superframe sync symbol once. Any value other than **0**X**011**XXX (where X is “don’t care”) is suitable.

At the superframe boundary, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component.

Corrupt 2 consecutive superframe sync symbols. Any value other than **0**X**011**XXX (where X is “don’t care”) is suitable.

Positively acknowledge the incoming REPORT_INFORMATION message.

Corrupt 6 consecutive superframe sync symbols. Any value other than **0**X**011**XXX (where X is “don’t care”) is suitable.

Positively acknowledge the incoming REPORT_INFORMATION message.

Success Criterion:

Verify that the Interface Device positively acknowledges the REQUEST_INFORMATION message sent after the isolated superframe sync symbol error.

Verify that the first REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device after the 2 erroneous frame sync symbols.

Verify that the LOST_SFS is set.

Verify that the second REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device after the 2 erroneous superframe sync symbols.

Verify that the LOST_FS is set.

Coverage:

Section	Statement
10.1	A Component **shall** implement the necessary recovery mechanisms to deal with the loss of **Superframe synchronization**, Frame synchronization, and Message synchronization.
10.1.2.4	A Component **shall** determine from the Superframe Sync pattern which of the eight Frame boundaries is the Superframe boundary.
10.1.2.4	A Component **shall** not change to the SeekingMessageSync state before it has verified the Superframe Sync pattern over sixteen consecutive Frames.
10.1.2.4	If the Component has not verified the Superframe Sync pattern after thirty-two Frames, it **shall** change to the SeekingFrameSync state (Figure 57, TimeOut transition) and seek Frame synchronization at a different phase.
10.1.2.4	A Component that has achieved Superframe synchronization **shall** continuously verify the Superframe Sync pattern.
10.1.2.4	If verification fails in two consecutive Superframes, the Component **shall** move to the SeekingSuperframeSync state (Figure 57, SuperframeSyncLost transitions).
10.1.2.4	Isolated verification [of the superframe sync pattern] errors **shall** be ignored.
11.3.5	[REPORT_INFORMATION] The Information Slice **shall** be placed in the LSBs of the relevant payload byte(s).
11.3.5	[REPORT_INFORMATION] Any unused bits **shall** be filled with zeros.
12.2.5.3	An Interface Device **shall** set LOST_SFS whenever the Component moves to the SeekingSuperframeSync state from a state other than the SeekingFrameSync state (Figure 57).
12.2.6	If an Interface Device detects that its Component has lost Superframe synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.3. SR3 - Framing Information Error

Test Description:

A device that has acquired the superframe sync must continuously monitor the SM value. If the verification fails once, it must be ignored (it can be an isolated error). If the verification fails in 2 consecutive occurrences, the device must reacquire the superframe sync.

In operation, the loss of frame sync will result in the interface device to transmit a REPORT_INFORMATION (EC=0x4008). The LOST_SFS will be set.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared. The SM value is equal to 19. The CG value is equal to 9.

Stimulus:

Corrupt the SM value once with value 0.

At the superframe boundary, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component.

Positively acknowledge the incoming REPLY_INFORMATION (TID=1) message.

Corrupt 2 consecutive SM values with value 0 then set it back to value 19.

Positively acknowledge the incoming REPORT_INFORMATION message.

Success Criterion:

Verify that the Interface Device positively acknowledges the REQUEST_INFORMATION message sent after the isolated SM value error.

Verify that the first REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device after the 2 erroneous SM value.

Verify that the IE LOST_SFS is set.

Coverage:

Section	Statement
10.1.2.4	A Component **shall** also continuously verify the SM, and those of the CG and RF Framing Information fields that they use.
10.1.2.4	For each of those fields [SM and (CG or RF or nothing)], if the Component sees a verification failure in two consecutive occurrences, it **shall** move to the SeekingSuperframeSync state (Figure 57, SuperframeSyncLost transitions).
10.2.4	If, for some reason, the Framing Information changes without detecting the appropriate Message in the Message Channel, a Component **shall** assume that it has missed a vital bus Message and change to the SeekingSuperframeSync state (see Figure 57).

6.2.4. SR4 - Superframe Sync and Reserved Bits

Test Description:

The eleven Reserved bits R[10:0] of the Framing Information must have a value equal to 0. However, if the value differs from 0, it must not impact the acquisition of the superframe sync.

The R bits affect directly the strip S where the superframe sync punctured pattern is encoded.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Set all Reserved bits to 1 (R[10:0]=0b11111111111) on 4 consecutive superframes.

At the third superframe boundary, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component.

Positively acknowledge the incoming REPLY_INFORMATION (TID=1) message.

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message positively acknowledged (PACK) by the Interface device.

Verify that the REPLY_INFORMATION message is sent with the proper TID=1.

Verify that the IE LOST_MS and LOST_SFS and LOST_SF are equal to 0.

Coverage:

Section	Statement
6.3.1.1	Components reading the Framing Information **shall** not use the contents of the reserved bits except to decode the stripe S.
6.3.1.3	The Superframe Sync pattern is a punctured repeating pattern, 0X011XXX, that allows a Component to detect the beginning of a new Superframe. The active Framer **shall** encode the Superframe Sync pattern in stripe S of the Framing Information using the equations shown in Figure 24.

6.2.5. SR5 - Message Sync Lost - Invalid Guide Integrity

Test Description:

A Component gains (or regains) Message synchronization by looking at the Guide Byte. Many events can lead to a loss of Message synchronization:

- **Invalid Guide Integrity**
- Mismatch between the broadcasted Guide Value and the expected one
- Primary Integrity error (CRC failing)
- Illegal Arbitration Type
- Illegal Message Type
- UDEF Message Response
- Collision in the Message Response

The Guide Byte Integrity will be corrupted and the Message sync acquisition will be verified by the transmission of a REPORT_INFORMATION (EC=0x4008) sent by the Interface Device of the Component.

While in the SeekingMessageSync, the Component will not be able to participate to any message transaction.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

No message will cross the superframe boundary. Therefore the Guide value should be equal to 0, the ENT bit should be equal to 0 and the Guide Integrity should be equal to 0b11. Set the Guide Integrity to 0b00.

Just after the erroneous Guide Byte slots, send a valid REQUEST_INFORMATION (TID=2, EC=0x4008) to the Interface device of the Component.

Right at the beginning of the next superframe, once the device has read the new correct Guide Byte, send a REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component. Positively acknowledge the REPLY_INFORMATION message.

Positively acknowledge the incoming REPORT_INFORMATION (EC=0x4008) message sent by the Interface device of the component.

Success Criterion:

Verify that the Component regains Message sync after the corruption of the Guide Integrity:

- Verify that the REQUEST_INFORMATION (TID=2, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).
- Check the transmission of the REPLY_INFORMATION message. Verify that the TID=1 and that LOST_MS IE is set.

- Check the transmission of the REPORT_INFORMATION message. Verify that the LOST_MS IE is set.

Coverage:

Section	Statement
10.1.2.5	A Component **shall** observe the Guide Channel and try to achieve Message synchronization as mandated by section 8.3.1.
10.1.2.5	A Component that has achieved Message synchronization **shall** continuously verify the Guide Channel and the Message Channel as described in section 8.3.2.
10.1.2.5	If [a loss of message sync] happens, the Component **shall** move to the SeekingMessageSync state.
8.1.1	A Component **shall** use the following formula to validate the Guide Byte: [Guide Byte Parity Check = (ENT ^ GV[3] ^ GV[1] ^ GI[1]) & (GV[4] ^ GV[2] ^ GV[0] ^ GI[0])]
8.3.1	The Component **shall** read the upcoming Guide Byte, and check the parity.
8.3.1	If the parity check does not pass, a Device **shall** not use the Guide Byte contents.
8.3.1	If the parity check does not pass, a Device **shall** wait for the next Guide Byte.
8.3.1	In the absence of events that would cause loss of synchronization, the Component **shall** achieve Message synchronization no later than the end of the following Guide Byte.
8.3.3	While tracking Messages, a Component **shall** continue monitoring every Guide Byte.
8.3.3	If the Guide Byte Parity Check is equal to zero, the Component **shall** lose Message synchronization.
8.3.3	A Component that has lost track of the Message flow but that is still in synchronization with the Superframe **shall** wait for the next Superframe boundary and again achieve synchronization with the Guide Channel.
8.3.3	The Component [that has lost track of the Message flow] **shall** update its Core Information Elements and set its Interface Device's LOST_MS Information Element to TRUE.
11.3.5	[REPORT_INFORMATION] The Information Slice **shall** be placed in the LSBs of the relevant payload byte(s).
11.3.5	[REPORT_INFORMATION] Any unused bits **shall** be filled with zeros.
12.2.5.2	An Interface Device **shall** set LOST_MS whenever it observes any of the following: Failed Guide Byte Parity Check (section 8.3.3), Mismatch of the Guide Byte (section 8.3.2), "Illegal" Arbitration Type (section 8.2.1.1), "Illegal" Message Type (section 8.2.2.1), Failure of the Primary Integrity CRC (section 8.2.4.1), "Undefined" Message Response code (section 8.2.4), Collision in the Message Channel (section 10.2.3)
12.2.6	If an Interface Device detects that its Component has lost Message synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.6. SR6 - Message Sync Lost - Guide Value Mismatch

Test Description:

A Component gains (or regains) Message synchronization by looking at the Guide Byte. Many events can lead to a loss of Message synchronization:

- Invalid Guide Integrity
- **Mismatch between the broadcasted Guide Value and the expected one**
- Primary Integrity error (CRC failing)
- Illegal Arbitration Type
- Illegal Message Type
- UDEF Message Response
- Collision in the Message Response

The Guide Value will be corrupted and the Message sync acquisition will be verified by the transmission of a REPORT_INFORMATION (EC=0x4008) sent by the Interface Device of the Component.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

No message will cross the superframe boundary. Therefore the Guide value should be equal to 0, the ENT bit should be equal to 0 and the Guide Integrity should be equal to 0b11. Set the Guide Value to 0x01, the ENT bit to 0 and the Guide Integrity to 0b10 (Guide Byte value = 0b00000101 = 0x5). The Guide Byte looks valid but it is not.

Just after the erroneous Guide Byte slots, send a valid REQUEST_INFORMATION (TID=2, EC=0x4008) to the Interface device of the Component.

Right at the beginning of the next superframe, once the device has read the new correct Guide Byte, send a REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component. Positively acknowledge the REPLY_INFORMATION message.

Positively acknowledge the incoming REPORT_INFORMATION (EC=0x4008) message sent by the Interface device of the component.

Success Criterion:

Verify that the Component regains Message sync after the corruption of the Guide Integrity:

- Verify that the REQUEST_INFORMATION (TID=2, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).
- Check the transmission of the REPLY_INFORMATION message. Verify that the TID=1 and that LOST_MS IE is set.
- Check the transmission of the REPORT_INFORMATION message. Verify that the LOST_MS IE is set.

Coverage:

Section	Statement
10.1.2.5	A Component **shall** observe the Guide Channel and try to achieve Message synchronization as mandated by section 8.3.1.
10.1.2.5	A Component that has achieved Message synchronization **shall** continuously verify the Guide Channel and the Message Channel as described in section 8.3.2.
10.1.2.5	If [a loss of message sync] happens, the Component **shall** move to the SeekingMessageSync state.
8.3.1	The Component **shall** read the upcoming Guide Byte, and check the parity.
8.3.1	In the absence of events that would cause loss of synchronization, the Component **shall** achieve Message synchronization no later than the end of the following Guide Byte.
8.3.3	While tracking Messages, a Component **shall** continue monitoring every Guide Byte.
8.3.3	A Component that has lost track of the Message flow but that is still in synchronization with the Superframe **shall** wait for the next Superframe boundary and again achieve synchronization with the Guide Channel.
8.3.3	The Component [that has lost track of the Message flow] **shall** update its Core Information Elements and set its Interface Device's LOST_MS Information Element to TRUE.
8.3.3	Otherwise, the Component **shall** check the consistency of the Guide Byte with its internal Component state.
8.3.3	If a Component has a mismatch when comparing the Guide Byte information with its internal Message Channel tracking mechanism, the Component's Interface Device **shall** set the LOST_MS Information Element to TRUE
12.2.5.2	An Interface Device **shall** set LOST_MS whenever it observes any of the following: Failed Guide Byte Parity Check (section 8.3.3), Mismatch of the Guide Byte (section 8.3.2), "Illegal" Arbitration Type (section 8.2.1.1), "Illegal" Message Type (section 8.2.2.1), Failure of the Primary Integrity CRC (section 8.2.4.1), "Undefined" Message Response code (section 8.2.4), Collision in the Message Channel (section 10.2.3)
12.2.6	If an Interface Device detects that its Component has lost Message synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.7. SR7 - Message Sync Lost - Primary Integrity Error

Test Description:

A Component gains (or regains) Message synchronization by looking at the Guide Byte. Many events can lead to a loss of Message synchronization:

- Invalid Guide Integrity
- Mismatch between the broadcasted Guide Value and the expected one
- **Primary Integrity error (CRC failing)**
- Illegal Arbitration Type
- Illegal Message Type
- UDEF Message Response
- Collision in the Message Response

The Primary Integrity field of a message will be corrupted, leading to a loss of Message sync by the component. The Message sync acquisition will be verified by the transmission of a REPORT_INFORMATION (EC=0x4008) sent by the Interface Device of the Component.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a REQUEST_INFORMATION (TID=3, EC=0x4008) with a corrupted Primary Integrity field to the Interface device of the Component.

Just after the erroneous message, send a valid REQUEST_INFORMATION (TID=2, EC=0x4008) to the Interface device of the Component.

Right at the beginning of the next superframe, once the device has read the new correct Guide Byte, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component. Positively acknowledge the REPLY_INFORMATION message.

Positively acknowledge the incoming REPORT_INFORMATION (EC=0x4008) message sent by the Interface device of the component.

Success Criterion:

Verify that the Component regains Message sync after the corruption of the Guide Integrity:

- Verify that the REQUEST_INFORMATION (TID=3, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=2, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).
- Check the transmission of the REPLY_INFORMATION message. Verify that the TID=1 and that LOST_MS IE is set.

- Check the transmission of the REPORT_INFORMATION message. Verify that the LOST_MS IE is set.

Coverage:

Section	Statement
10.1.2.5	A Component **shall** observe the Guide Channel and try to achieve Message synchronization as mandated by section 8.3.1.
10.1.2.5	A Component that has achieved Message synchronization **shall** continuously verify the Guide Channel and the Message Channel as described in section 8.3.2.
10.1.2.5	If [a loss of message sync] happens, the Component **shall** move to the SeekingMessageSync state.
8.3.1	The Component **shall** read the upcoming Guide Byte, and check the parity.
8.3.1	In the absence of events that would cause loss of synchronization, the Component **shall** achieve Message synchronization no later than the end of the following Guide Byte.
8.3.3	While tracking Messages, a Component **shall** continue monitoring every Guide Byte.
8.3.3	A Component that has lost track of the Message flow but that is still in synchronization with the Superframe **shall** wait for the next Superframe boundary and again achieve synchronization with the Guide Channel.
8.3.3	The Component [that has lost track of the Message flow] **shall** update its Core Information Elements and set its Interface Device's LOST_MS Information Element to TRUE.
8.4.2.5	A Component **shall** read the Primary Integrity and verify the value with its internal CRC as described in section 8.2.4.1.
8.4.2.6	If the [Primary Integrity] CRC fails, the Component **shall** change to the Error state and lose Message synchronization (InvalidHeader transition in Figure 40).
12.2.5.2	An Interface Device **shall** set LOST_MS whenever it observes any of the following: Failed Guide Byte Parity Check (section 8.3.3), Mismatch of the Guide Byte (section 8.3.2), “Illegal” Arbitration Type (section 8.2.1.1), “Illegal” Message Type (section 8.2.2.1), Failure of the Primary Integrity CRC (section 8.2.4.1), “Undefined” Message Response code (section 8.2.4), Collision in the Message Channel (section 10.2.3)
12.2.6	If an Interface Device detects that its Component has lost Message synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.8. SR8 - Message Sync Lost - Illegal Arbitration Type

Test Description:

A Component gains (or regains) Message synchronization by looking at the Guide Byte. Many events can lead to a loss of Message synchronization:

- Invalid Guide Integrity
- Mismatch between the broadcasted Guide Value and the expected one
- Primary Integrity error (CRC failing)
- **Illegal Arbitration Type**
- Illegal Message Type
- UDEF Message Response
- Collision in the Message Response

A message with an illegal Arbitration Type will be sent, leading to a loss of Message sync by the component. The Message sync acquisition will be verified by the transmission of a REPORT_INFORMATION (EC=0x4008) sent by the Interface Device of the Component.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a REQUEST_INFORMATION (TID=3, EC=0x4008) with an Arbitration Type field value equal to **0b1101** (13 or 0xD) to the Interface device of the Component.

Just after the erroneous message, send a valid REQUEST_INFORMATION (TID=2, EC=0x4008) to the Interface device of the Component.

Right at the beginning of the next superframe, once the device has read the new correct Guide Byte, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component. Positively acknowledge the REPLY_INFORMATION message.

Positively acknowledge the incoming REPORT_INFORMATION (EC=0x4008) message sent by the Interface device of the component.

Success Criterion:

Verify that the Component regains Message sync after the corruption of the Guide Integrity:

- Verify that the REQUEST_INFORMATION (TID=3, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=2, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).
- Check the transmission of the REPLY_INFORMATION message. Verify that the TID=1 and that LOST_MS IE is set.

- Check the transmission of the REPORT_INFORMATION message. Verify that the LOST_MS IE is set.

Coverage:

Section	Statement
10.1.2.5	A Component **shall** observe the Guide Channel and try to achieve Message synchronization as mandated by section 8.3.1.
10.1.2.5	A Component that has achieved Message synchronization **shall** continuously verify the Guide Channel and the Message Channel as described in section 8.3.2.
10.1.2.5	If [a loss of message sync] happens, the Component **shall** move to the SeekingMessageSync state.
8.3.1	The Component **shall** read the upcoming Guide Byte, and check the parity.
8.3.1	A Component **shall** track the Arbitration Sequence, read the RL field and track the Message until the end.
8.3.1	In the absence of events that would cause loss of synchronization, the Component **shall** achieve Message synchronization no later than the end of the following Guide Byte.
8.3.3	While tracking Messages, a Component **shall** continue monitoring every Guide Byte.
8.3.3	A Component that has lost track of the Message flow but that is still in synchronization with the Superframe **shall** wait for the next Superframe boundary and again achieve synchronization with the Guide Channel.
8.3.3	The Component [that has lost track of the Message flow] **shall** update its Core Information Elements and set its Interface Device's LOST_MS Information Element to TRUE.
8.4.2	A Component **shall** always parse the Arbitration Type, so that it knows the length of the Arbitration field
8.4.2.1	[...] when a Component observes an illegal value in the Arbitration [Type] field, the Component **shall** change to the Error state in Figure 40.
8.4.2.1	In the latter case [when a Component observes an illegal value in the Arbitration field], the Component's Interface Device **shall** change its LOST_MS Information Element to TRUE
8.4.2.1	In the latter case [when a Component observes an illegal value in the Arbitration field], the Component's Interface Device **shall** lose Message synchronization as described in section 8.3.4.
12.2.5.2	An Interface Device **shall** set LOST_MS whenever it observes any of the following: Failed Guide Byte Parity Check (section 8.3.3), Mismatch of the Guide Byte (section 8.3.2), "Illegal" Arbitration Type (section 8.2.1.1), "Illegal" Message Type (section 8.2.2.1), Failure of the Primary Integrity CRC (section 8.2.4.1), "Undefined" Message Response code (section 8.2.4), Collision in the Message Channel (section 10.2.3)
12.2.6	If an Interface Device detects that its Component has lost Message synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.9. SR9 - Message Sync Lost - Illegal Message Type

Test Description:

A Component gains (or regains) Message synchronization by looking at the Guide Byte. Many events can lead to a loss of Message synchronization:

- Invalid Guide Integrity
- Mismatch between the broadcasted Guide Value and the expected one
- Primary Integrity error (CRC failing)
- Illegal Arbitration Type
- **Illegal Message Type**
- UDEF Message Response
- Collision in the Message Response

A message with an illegal Message Type will be sent, leading to a loss of Message sync by the component. The Message sync acquisition will be verified by the transmission of a REPORT_INFORMATION (EC=0x4008) sent by the Interface Device of the Component.

Note that the test is about illegal values, not reserved values.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a REQUEST_INFORMATION (TID=3, EC=0x4008) with an Message Type field value equal to **0b011** to the Interface device of the Component.

Just after the erroneous message, send a valid REQUEST_INFORMATION (TID=2, EC=0x4008) to the Interface device of the Component.

Right at the beginning of the next superframe, once the device has read the new correct Guide Byte, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component. Positively acknowledge the REPLY_INFORMATION message.

Positively acknowledge the incoming REPORT_INFORMATION (EC=0x4008) message sent by the Interface device of the component.

Success Criterion:

Verify that the Component regains Message sync after the corruption of the Guide Integrity:

- Verify that the REQUEST_INFORMATION (TID=3, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=2, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).
- Check the transmission of the REPLY_INFORMATION message. Verify that the TID=1 and that LOST_MS IE is set.

- Check the transmission of the REPORT_INFORMATION message. Verify that the LOST_MS IE is set.

Coverage:

Section	Statement
10.1.2.5	A Component **shall** observe the Guide Channel and try to achieve Message synchronization as mandated by section 8.3.1.
10.1.2.5	A Component that has achieved Message synchronization **shall** continuously verify the Guide Channel and the Message Channel as described in section 8.3.2.
10.1.2.5	If [a loss of message sync] happens, the Component **shall** move to the SeekingMessageSync state.
8.3.1	The Component **shall** read the upcoming Guide Byte, and check the parity.
8.3.1	In the absence of events that would cause loss of synchronization, the Component **shall** achieve Message synchronization no later than the end of the following Guide Byte.
8.3.3	While tracking Messages, a Component **shall** continue monitoring every Guide Byte.
8.3.3	A Component that has lost track of the Message flow but that is still in synchronization with the Superframe **shall** wait for the next Superframe boundary and again achieve synchronization with the Guide Channel.
8.3.3	The Component [that has lost track of the Message flow] **shall** update its Core Information Elements and set its Interface Device's LOST_MS Information Element to TRUE.
8.4.2	A Component **shall** always parse the Message Type, some MT codes causes the Component to lose Message synchronization
8.4.2.2	A Component that observes a Message with an Illegal Message Type **shall** lose Message synchronization (InvalidHeader transition in Figure 40).
12.2.5.2	An Interface Device **shall** set LOST_MS whenever it observes any of the following: Failed Guide Byte Parity Check (section 8.3.3), Mismatch of the Guide Byte (section 8.3.2), “Illegal” Arbitration Type (section 8.2.1.1), “Illegal” Message Type (section 8.2.2.1), Failure of the Primary Integrity CRC (section 8.2.4.1), “Undefined” Message Response code (section 8.2.4), Collision in the Message Channel (section 10.2.3)
12.2.6	If an Interface Device detects that its Component has lost Message synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.10. SR10 - Message Sync Lost - UDEF

Test Description:

A Component gains (or regains) Message synchronization by looking at the Guide Byte. Many events can lead to a loss of Message synchronization:

- Invalid Guide Integrity
- Mismatch between the broadcasted Guide Value and the expected one
- Primary Integrity error (CRC failing)
- Illegal Arbitration Type
- Illegal Message Type
- **UDEF Message Response**
- Collision in the Message Response

A message that gets UDEF in the Message Response field will cause the transmitting device to lose Message sync. The Message sync acquisition will be verified by the transmission of a REPORT_INFORMATION (EC=0x4008) sent by the Interface Device of the Component.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared.

Stimulus:

Send a REQUEST_INFORMATION (TID=3, EC=0x4008) message with **0b0100** in the Message Response field. Set the source address to be 0x20. Destination address shall be 0x00.

Just after the erroneous message, send a valid REQUEST_INFORMATION (TID=2, EC=0x4008) to the Interface device of the Component.

Right at the beginning of the next superframe, once the device has read the new correct Guide Byte, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) to the Interface device of the Component. Positively acknowledge the REPLY_INFORMATION message.

Positively acknowledge the incoming REPORT_INFORMATION (EC=0x4008) message sent by the Interface device of the component.

Success Criterion:

Verify that the Component regains Message sync after the corruption of the Guide Integrity:

- Verify that the REQUEST_INFORMATION (TID=3, EC=0x4008) message has the value 0xE (0b1010 **OR** 0b0100 = 0b1110).
- Verify that the REQUEST_INFORMATION (TID=2, EC=0x4008) message is not acknowledged (NORE).
- Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).
- Check the transmission of the REPLY_INFORMATION message. Verify that the TID=1 and that LOST_MS IE is set.

- Check the transmission of the REPORT_INFORMATION message. Verify that the LOST_MS IE is set.

Coverage:

Section	Statement
10.1.2.5	A Component **shall** observe the Guide Channel and try to achieve Message synchronization as mandated by section 8.3.1.
10.1.2.5	A Component that has achieved Message synchronization **shall** continuously verify the Guide Channel and the Message Channel as described in section 8.3.2.
10.1.2.5	If [a loss of message sync] happens, the Component **shall** move to the SeekingMessageSync state.
8.3.1	The Component **shall** read the upcoming Guide Byte, and check the parity.
8.3.1	A Component **shall** track the Message until the end.
8.3.1	In the absence of events that would cause loss of synchronization, the Component **shall** achieve Message synchronization no later than the end of the following Guide Byte.
8.3.2	In order to remain synchronized to the Message Channel, a Component **shall** track the Messages appearing in the Message Channel.
8.3.3	While tracking Messages, a Component **shall** continue monitoring every Guide Byte.
8.3.3	A Component that has lost track of the Message flow but that is still in synchronization with the Superframe **shall** wait for the next Superframe boundary and again achieve synchronization with the Guide Channel.
8.3.3	The Component [that has lost track of the Message flow] **shall** update its Core Information Elements and set its Interface Device's LOST_MS Information Element to TRUE.
8.4.2	While sending its acknowledgement, a Device **shall** use Logical-OR signaling as described in section 5.1.
8.4.2.7	When the Issued Response is a NORE and the Read Response is a UDEF, the Device **shall** cause its Component to lose Message synchronization and return to the SeekingMessageSync state.
8.4.2.7	When the Issued Response is a NORE and the Read Response is a UDEF, the Device **shall** cause its Component to assert LOST_MS
12.2.5.2	An Interface Device **shall** set LOST_MS whenever it observes any of the following: Failed Guide Byte Parity Check (section 8.3.3), Mismatch of the Guide Byte (section 8.3.2), “Illegal” Arbitration Type (section 8.2.1.1), “Illegal” Message Type (section 8.2.2.1), Failure of the Primary Integrity CRC (section 8.2.4.1), “Undefined” Message Response code (section 8.2.4), Collision in the Message Channel (section 10.2.3)
12.2.6	If an Interface Device detects that its Component has lost Message synchronization, it **shall** send a REPORT_INFORMATION Message, including byte 0x400 of the Information Map, to the active Manager once it has regained Message synchronization.

6.2.11. SR11 - Framing Extension (FE) effects

Test Description:

When the Framing Extension (FE) bit is set, the SM, CG, RF and R fields are not present. Therefore, the FE bit will affect the way a device is trying to gain synchronization. But the FE bit will not impact devices that are already synchronized.

The superframe sync pattern must be verified over 16 frames (2 superframes). The value of the Subframe Mode (SM) field must also have the same value over 2 superframes

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The LOST_SF, LOST_SFS and LOST_MS IE are all cleared. The SM value = 19, the CG value = 9.

Stimulus:

Set the FE bit value to 1 over 4 consecutive superframes. The SM value should be set to 0.

At the beginning of the 2nd superframe, send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device.

Positively acknowledge the REPLY_INFORMATION (TID=1) message sent by the Interface device of the component.

At the end of the 4 superframes, corrupt the superframe sync symbol in 2 consecutive superframes to trigger a loss of superframe sync. The FE bit value is don't care (set it to 0 by default).

At the end of the 2 superframes, transmit 10 superframes with a valid sync pattern but with the FE bit equal to 1 (it should prevent the device to regain superframe sync).

At the beginning of the 8th superframe, send a valid REQUEST _INFORMATION (TID=2, EC=0x4008) message to the Interface device.

At the end of the 10 superframes, set the FE bit value to 0 (it should allow the device to regain superframe sync and message sync) and the SM value to 19.

Positively acknowledge the incoming REPORT_INFORMATION (EC=0x4008) message sent by the Interface device of the component.

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).

Verify that the REPLY_INFORMATION (TID=1) message is sent by the Interface device.

Verify that the LOST_MS, LOST_SFS and LOST_FS IE are all cleared.

Verify that the REQUEST _INFORMATION (TID=2, EC=0x4008) message is not acknowledged (NORE).

Verify that the REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device.

Verify that the LOST_SFS IE is set.

Coverage:

Section	Statement
6.3.1.2	If a Component receives a Superframe with FE=1, it **shall** assume the RF, CG, SM and R fields are absent in the current Superframe.
6.3.1.2	The FE field **shall** have no effect on the mechanisms verification of the Frame synchronization.
6.3.1.2	A Component that locks to the Superframe Sync pattern when FE=1 cannot determine the current Subframe Mode (SM) and therefore **shall** defer the process of acquiring Superframe synchronization and Message synchronization until the Subframe Mode has been indicated in the Framing Information.
6.3.1.2	The FE field shall have no effect on the mechanisms for verification of Message synchronization.
6.3.1.2	The FE field **shall** have no effect on the mechanisms for verification of Message synchronization.
10.1.2.4	For Superframes with FE=0, a Component **shall** not change to the SeekingMessageSync state until it has observed the contents of SM field as being identical across two Superframes.

6.3. Message Protocol Check

6.3.1. MPC1 - Unsupported Source Class Specific Message

Test Description:

A device that receives a Source Class Specific Message and that has no knowledge about that specific class shall treat such a message as an unsupported message. Therefore, the UNSPRTD_MSG IE will be set.

Initial conditions:

The device is in the *Enumerated* state. The UNSPRTD_MSG IE is cleared.

Stimulus:

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device with the Message Type value set to 0b101 (5).

Send a message REQUEST_INFORMATION(TID=2, EC=0x0008) to the Interface device

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).

Verify that there is no REPLY_INFORMATION with TID=1.

Verify that the REQUEST _INFORMATION (TID=2, EC=0x0008) message is positively acknowledged (PACK).

Verify that the REPPLY_INFORMATION (TID=2, EC=0x4008) message is sent by the Interface device.

Verify that the UNSPRTD_MSG IE is set.

Coverage:

Section	Statement
8.4.2.3	A Source-referred Class specific Message received from a Device whose Device Class is unknown **shall** be treated as unsupported by the destination Device.

6.3.2. MPC2 - Unsupported Source User Specific Message

Test Description:

A device that receives a Source User Specific Message and that has no knowledge about that specific user message shall treat such a message as an unsupported message. Therefore, the UNSPRTD_MSG IE will be set.

Initial conditions:

The device is in the *Enumerated* state. The UNSPRTD_MSG IE is cleared.

Stimulus:

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device with the Message Type value set to 0b110 (6).

Send a message REQUEST_INFORMATION(TID=2, EC=0x0008) to the Interface device

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).

Verify that there is no REPLY_INFORMATION with TID=1.

Verify that the REQUEST _INFORMATION (TID=2, EC=0x0008) message is positively acknowledged (PACK).

Verify that the REPPLY_INFORMATION (TID=2, EC=0x4008) message is sent by the Interface device.

Verify that the UNSPRTD_MSG IE is set.

Coverage:

Section	Statement
8.4.2.3	A Source-referred User Message received from a Device whose User Messages are unknown **shall** be treated as unsupported by the destination Device.

6.3.3. MPC3 - Illegal Message Type

Test Description:

A device that receives a message with an illegal value int he Message Type field must discard the message and lose the message synchronization. The Interface device must send a REPORT_PRESENT message with the LOST_MS IE set.

There are 2 illegal values: 0b011 (3) and 0b111 (7).

Initial conditions:

The device is in the *Enumerated* state. The LOST_MS IE is cleared.

Stimulus:

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device with the Message Type value set to 0b011 (3).

Positively acknowledge the REPORT_INFORMATION(EC=0x4008) sent by the Interface device.

Send a CLEAR_INFORMATION(EC=0x4008, CM=0x1F) message to the Interface device to clear the LOST_MS IE.

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device with the Message Type value set to 0b111 (7).

Positively acknowledge the REPORT_INFORMATION(EC=0x4008) sent by the Interface device.

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message with illegal MT is not acknowledged (NORE).

Verify that there is no REPLY_INFORMATION with TID=1.

Verify that the REPORT_INFORMATION(EC=0x4008) message signifies a loss of message sync (LOST_MS=1).

Verify that the CLEAR_INFORMATION(EC=0x4008, CM=0x1F) is positively acknowledged

Verify that the REQUEST_INFORMATION (TID=2, EC=0x4008) message with illegal MT is not acknowledged (NORE).

Verify that there is no REPLY_INFORMATION with TID=2.

Verify that the REPORT_INFORMATION(EC=0x4008) message signifies a loss of message sync (LOST_MS=1).

Coverage:

Section	Statement
8.4.2.2	A Component that observes a Message with an Illegal Message Type **shall** discard the Message.
8.4.2.2	A Component that observes a Message with an Illegal Message Type **shall** lose Message synchronization (InvalidHeader transition in Figure 40).

6.3.4. MPC4 - Reserved Message Type

Test Description:

A device that receives a message with an reserved value int he Message Type field must discard (not use it for any purpose) the message but shall stay synchronized to the message channel. The message parsing must take place and an acknowledgment must be given (according the Message Integrity check).

There is 1 reserved value: 0b100 (4).

Initial conditions:

The device is in the *Enumerated* state. The LOST_MS IE is cleared.

Stimulus:

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device with the Message Type value set to 0b100 (4).

Send a valid message REQUEST_INFORMATION(TID=2, EC=0x4008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=2) sent by the Interface device.

Send a valid message REQUEST_INFORMATION(TID=3, EC=0x0008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=3) sent by the Interface device.

Success Criterion:

Verify that the (reserved message type) REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).

Verify that there is no REPLY_INFORMATION with TID=1.

Verify that the REPLY_INFORMATION(TID=2) message reports no loss of message sync (IE LOST_MS is cleared).

Verify that the REPLY_INFORMATION(TID=3) message reports an unsupported message (IE UNSPRTD_MSG is set).

Coverage:

Section	Statement
7.1.2.6	The Device **shall** set UNSPRTD_MSG whenever it receives a non-broadcast Message that it does not implement as a Destination.
8.4.2.2	A Component that observes a Message with Reserved Message Type **shal**l discard the Message.
8.4.2.2	A Component that observes a Message with Reserved Message Type **shall** stay in Message synchronization.
10.2.5	The Device **shall** not act upon an unsupported Message with the exception of setting the UNSPRTD_MSG Information Element.

6.3.5. MPC5 - Reserved Destination Type

Test Description:

A device that receives a message with an reserved value int he Destination Type field must discard the message (and not act to its content) but shall stay synchronized to the message channel.

There is 1 reserved value: 0b10 (2).

Initial conditions:

The device is in the *Enumerated* state. The LOST_MS IE is cleared.

Stimulus:

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device with the Destination Type value set to 0b10 (2).

Send a valid message REQUEST_INFORMATION(TID=2, EC=0x4008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=2) sent by the Interface device.

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is not acknowledged (NORE).

Verify that there is no REPLY_INFORMATION with TID=1.

Verify that the REPLY_INFORMATION(TID=2) message reports no loss of message sync (IE LOST_MS is cleared).

Coverage:

Section	Statement
8.4.2.4	If a Device observes a Message with a reserved value for the Destination Type, the Device **shall** discard the Message and **shall** not act on the contents.

6.3.6. MPC6 - Reserved bits and CRC verification

Test Description:

A device must use the content of the reserved cells of the header for the CRC calculation, even if these cells are not equal to 0.

The Broadcast header, the Short Header and the Long Header have the same reserved cells. The three headers must be tested separately.

Initial conditions:

The device is in the *Operational* state.

Stimulus:

Send a message ASSIGN_LOGICAL_ADDRESS(LA=0x00) to the Interface device with all the reserved bits set to 1.

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device with all the reserved bits set to 1.

Positively acknowledge the REPLY_INFORMATION(TID=1) sent by the Interface device.

Send a message BEGIN_RECONFIGURATION with all the reserved bits set to 1.

Send a message NEXT_SUBFRAME_MODE(SM=24).

Send a message RECONFIGURE_NOW.

Send a message REQUEST_INFORMATION(TID=2, EC=0x4008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=2) sent by the Interface device.

Success Criterion:

Verify that the REQUEST_INFORMATION(TID=1, EC=0x4008) message is positively acknowledged.

Verify that the REPLY_INFORMATION(TID=1) is sent by the Interface device.

Verify that the REPLY_INFORMATION(TID=1) message reports no loss of sync .

Verify that all the message from the Reconfiguration Sequence are positively acknowledged.

Verify that the REPLY_INFORMATION(TID=2) is sent by the Interface device.

Verify that the REPLY_INFORMATION(TID=2) message reports no loss of sync .

Coverage:

Section	Statement
8.2.2	The [three reserved Cells] contents **shal**l be used only for CRC generation.
8.2.4.1	Reserved Cells **shall** be included when performing Integrity CRC calculations.
11	Destinations **shal**l not use the contents of reserved Cells, except for Message integrity verification.

6.3.7. MPC7 - Message Response

<u>Test Description:</u>

Every SLIMbus message must receive an acknowledgment. Depending on the verification of the Message Integrity CRC, the message will receive a Positive ACKnowledgment (PACK) if the test passes or a Negative ACKnowledgment (NACK) if the test fails.

<u>Initial conditions:</u>

The device is in the *Enumerated* state.

<u>Stimulus:</u>

Send a valid message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=1) sent by the Interface device.

Send a message REQUEST_INFORMATION(TID=2, EC=0x4008) to the Interface device with a bad Message Integrity CRC.

<u>Success Criterion:</u>

Verify that the REQUEST_INFORMATION(TID=1, EC=0x4008) message is positively acknowledged.

Verify that the REPLY_INFORMATION(TID=1) is sent by the Interface device.

Verify that the REQUEST_INFORMATION(TID=2, EC=0x4008) message is negatively acknowledged.

Verify that there is no REPLY_INFORMATION(TID=2).

<u>Coverage:</u>

Section	Statement
8.4.2.7	A Device that is a destination of the incoming Message **shall** write to the Response Field as mandated in Table 44.
8.4.2.7	A destination Device **shall** issue PACK when the Message Integrity CRC passes, except when the Device requires retransmission of the Message.
8.4.2.7	A destination Device **shall** issue NACK if the Message Integrity CRC fails.
8.4.2.7	When the Issued Response is a PACK and the Read Response is a PACK, the destination Device **shall** start processing the received Message.
8.4.2.7	When the Read Response is a NACK, the destination Device **shall** discard the received Message.

6.3.8. MPC8 - Remaining Length Usage

Test Description:

The Remaining Length Header field tells a device the size of the message and, by consequence, the size of the message payload. The Remaining Length value might be longer or shorter than the specified value.

If it is longer, a device must read the payload as if the last, “unexpected” transmitted bytes of the payload are reserved.

If it is smaller, then the message is unusable by the device as information are missing.

Initial conditions:

The device is in the *Enumerated* state.

Stimulus:

Send a message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device that is one byte longer than normal.

Positively acknowledge the REPLY_INFORMATION(TID=1) sent by the Interface device.

Send a message REQUEST_INFORMATION(TID=2, EC=0x4008) to the Interface device that is one byte shorter than normal.

Success Criterion:

Verify that the REQUEST_INFORMATION(TID=1, EC=0x4008) message is positively acknowledged.

Verify that the REPLY_INFORMATION(TID=1) is sent by the Interface device.

Verify that the REQUEST_INFORMATION(TID=2, EC=0x4008) message is positively acknowledged.

Verify that there is no REPLY_INFORMATION(TID=2).

Coverage:

Section	Statement
8.4.2.5	If the RL field indicates a Message Payload length longer than what is needed for the necessary fields for a particular Message, a Device **shall** assume that the relevant content is located in the beginning of the Message Payload, with layout as specified for that Message.
8.4.2.5	[If the RL field indicates a Message Payload length longer than what is needed for the necessary fields for a particular Message] The unknown field(s) **shall** be considered reserved, i.e. Devices shall use these fields only for CRC calculation.
8.4.2.5	If the RL field indicates a length shorter than what is needed for that Message, a destination Device **shall** discard the Message.
11.6	If the EX_ERROR Core Information Element is implemented, it **shall** be set when the received Core Message is supported by the Device, and the indicated check procedure is implemented, and the received Core Message fails the check [Received Core message with Remaining Length Header field indicating payload shorter than specified in section 11 for the message].

6.3.9. MPC9 - Unsupported Messages

Test Description:

When a device receives a message that it does not support, it must acknowledge it according to the message verification rules but shall not act on it. Additionally, the device must set the UNSPRTD_MSG Information Element.

When device receives a Broadcast message that is optional for its class but that it does not support, it must not set the UNSPRTD_MSG IE.

Any core message sent with an unassigned message code is also unsupported.

Initial conditions:

The device is in the *Enumerated* state and UNSPRTD_MSG IE is cleared.

Stimulus:

Send a valid message CONNECT_SOURCE (or any other unsupported message) to the Interface device.

Send a message REQUEST_INFORMATION(TID=1, EC=0x0008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=1) sent by the Interface device.

Send a message CLEAR_INFORMATION(EC=0x0008, CM=0x0B) to the Interface device to clear the UNSPRTD_MSG Information Element.

Send a valid message CONNECT_SOURCE (or any other unsupported message) to the Interface device with the destination type set to Broadcast (0b11).

Send a message REQUEST_INFORMATION(TID=1, EC=0x0008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=1) sent by the Interface device.

Send a message CLEAR_INFORMATION(EC=0x0008, CM=0x0B) to the Interface device to clear the UNSPRTD_MSG Information Element.

Send a valid core message with message code equal to 0x03 (not defined in the SLIMbus specification).

Send a message REQUEST_INFORMATION(TID=2, EC=0x0008) to the Interface device.

Positively acknowledge the REPLY_INFORMATION(TID=2) sent by the Interface device.

Success Criterion:

Verify that the CONNECT_SOURCE message is positively acknowledged.

Verify that the REPLY_INFORMATION(TID=1) is sent by the Interface device and verify that the UNSPRTD_MSG IE is set.

Verify that the second CONNECT_SOURCE message is positively acknowledged.

Verify that the REPLY_INFORMATION(TID=1) is sent by the Interface device and verify that the UNSPRTD_MSG IE is cleared.

Verify the 0x03 Core message is positively acknowledged by the Interface device.

Verify that the REPLY_INFORMATION(TID=2) is sent by the Interface device and verify that the UNSPRTD_MSG IE is set.

Coverage:

Section	Statement
7.1.2.6	The Device **shall** set UNSPRTD_MSG whenever it receives a non-broadcast Message that it does not implement as a Destination.
10.2.5	A Device **shall** set the UNSPRTD_MSG Information Element if it receives a unicast Message that it does not implement as a destination Device.
10.2.5	The Device **shall** not act upon an unsupported Message with the exception of setting the UNSPRTD_MSG Information Element.
10.2.5	A destination Device **shall** not set the UNSPRTD_MSG Information Element if it receives a broadcast Message that is optional for the Device Class to which the destination Device belongs.
11	A destination Device that receives a Message with a reserved Core Message Code **shall** treat the Message as unsupported.

6.3.10. MPC10 - Arbitration Priority

Test Description:

The message arbitration of a device can be changed by the active Manager. The device is free to use the newly assigned arbitration priority or to use the Default arbitration priority. This test assumes that the device will use the newly assigned arbitration priority.

After a component reset, the arbitration priority must be reset to the device defaults.

Initial conditions:

The device is in the *Enumerated* state.

Stimulus:

Send a message REQUEST_INFORMATION(TID=1, EC=0x0008) to the Interface device. Positively acknowledge the REPLY_INFORMATION(TID=1).

Send a message CHANGE_ARBITRATION_PRIORITY(AP=3) to the Interface device. This is setting High Priority.

Send a message REQUEST_INFORMATION(TID=2, EC=0x0008) to the Interface device.

Send a reconfiguration to change the Subframe Mode:

- BEGIN_RECONFIGURATION with Default arbitration priority.
- NEXT_SUBFRAME_MODE(SM=24) with high arbitration priority.
- NEXT_SUBFRAME_MODE(SM=25) with Default arbitration priority
- RECONFIGURE_NOW

After the Reconfiguration Boundary, send a message REQUEST_INFORMATION (TID=3, EC=0x4008) to the Interface device. Positively acknowledge the REPLY_INFORMATION(TID=3).

Send a RESET_DEVICE message to the Interface device.

Success Criterion:

Verify that the REPLY_INFORMATION(TID=1) is sent by the Interface device with a Default arbitration priority.

Verify that the REPLY_INFORMATION(TID=2) is sent by the Interface device with a High arbitration priority.

After the Reconfiguration sequence, verify that the REQUEST_INFORMATION (TID=3, EC=0x4008) is positively acknowledged and followed by a REPLY_INFORMATION(TID=3) reporting no loss of sync.

Verify that the REPORT_PRESENT messages are sent with a Default arbitration priority.

Coverage:

Section	Statement
10.11	For each Message Code if a Device does not use the newly assigned Arbitration Priority, it **shall** use the default Arbitration Priority.
10.11	A Device **shall** not change the Arbitration Priority of a Message at runtime unless it receives a CHANGE_ARBITRATION_PRIORITY Message.
10.11	When arbitrating with an Enumeration Address, a Device **shall** not use the Arbitration Priority assigned by the active Manager.
10.11	A Device **shall** behave as though all Messages are executed in the order they are received, even if the Messages have different Arbitration Priorities.

6.3.11. MPC11 - REPLY retransmission

Test Description:

A REPLY message that follows a REQUEST message must not be retransmitted infinitely if it keeps receiving a NORE or a NACK in its message response field. The test uses the Information elements. If the Device Under Test has aValue Element implemented, the test can apply to it as wall.

Initial conditions:

The device is in the *Enumerated* state.

Stimulus:

Send a valid message REQUEST_INFORMATION(TID=1, EC=0x4008) to the Interface device.

Negatively acknowledge the REPLY_INFORMATION message a 100 times.

Success Criterion:

Verify that the REQUEST_INFORMATION(TID=1, EC=0x4008) message is positively acknowledged.

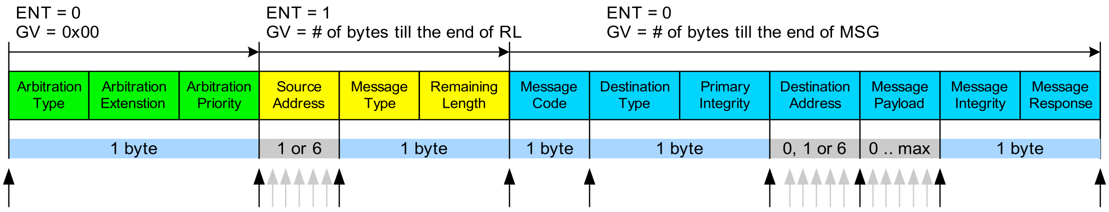
Verify that the REPLY_INFORMATION(TID=1) retransmission stops after a while.

Coverage:

Section	Statement
10.12.2	REPLY Messages **shall** not be retransmitted indefinitely.

6.4. Guide Byte Generation

These tests are meant for Clock Sourcing Components. They will mainly address Framer behaviors with messages crossing the superframe boundary.



6.4.1. GBG1 - Message crossing with ENT=1

Test Description:

In most of the case, messages are not crossing the superframe boundary. In this case, the ENT bit is equal to 0 and the Guide Value is equal to 0. The situation does not trigger any special behavior. Devices know they can simply use the first control slot available for messaging.

But, when a message crosses the superframe boundary between the beginning of the Source Address and the end of the Remaining Length (RL) field, the active Framer must set the ENT bit (equal to 1). The Active Framer must also compute the Guide Value which is equal to the number of bytes till the end of the RL field.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 19.

Stimulus:

In Subframe Mode 19, there are 174 slots available for a message. The REQUEST_INFORMATION message is 10 bytes long. To cross the superframe boundary with ENT=1, the message shall start 1 or 2 bytes before the end of the superframe. Start the message slot 1504 (2 bytes before crossing).

Send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device in such a way that it crosses the superframe boundary with ENT=1.

Positively acknowledge the REPLY_INFORMATION (TID=1) message sent by the Interface device of the component.

Success Criterion:

Verify that the ENT bit is equal to 1 and that the Guide Value is equal to 1.

Coverage:

Section	Statement
8.1.2	If the Guide Byte occurs elsewhere [not after the RL field, but before the start of the next Message], the ENT bit **shall** be set to one.
8.1.3	When ENT=1, the Guide Value **shall** be the number of bytes in the Message Channel, 1 to 7, that remains after the Guide Byte until the end of the RL field.

6.4.2. GBG2 - Message crossing with ENT=0

Test Description:

In most of the case, messages are not crossing the superframe boundary. In this case, the ENT bit is equal to 0 and the Guide Value is equal to 0. The situation does not trigger any special behavior. Devices know they can simply use the first control slot available for messaging.

But, when a message crosses the superframe boundary between the beginning of the Message Code and the end of the Message Payload,, the active Framer must reset the ENT bit (equal to 0). The Active Framer must also compute the Guide Value which is equal to the number of bytes till the end of the RL field.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 19.

Stimulus:

In Subframe Mode 19, there are 174 slots available for a message. The REQUEST_INFORMATION message is 10 bytes long. To cross the superframe boundary with ENT=0, the message shall start 3 to 8 bytes before the end of the superframe. Start the message slot 1472 (4 bytes before crossing).

Send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device in such a way that it crosses the superframe boundary with ENT=0.

Positively acknowledge the REPLY_INFORMATION (TID=1) message sent by the Interface device of the component.

Success Criterion:

Verify that the ENT bit is equal to 0 and that the Guide Value is equal to 6.

Coverage:

Section	Statement
8.1.2	If the Guide Byte occurs after the RL field, but before the start of the next Message, the ENT bit **shall** be set to zero.
8.1.3	When ENT=0, the Guide Value **shall** be the number of bytes in the Message Channel, 0 to 31, that remains after the Guide Byte until the end of the current Message.

6.4.3. GBG3 - Message ending just before the Guide Byte

Test Description:

When a message ends just before the Guide Byte, the ENT bit and the Guide value must be equal to 0.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 0 (100% control mode).

Stimulus:

In Subframe Mode 0, there are 1518 slots available for a message. The REQUEST_INFORMATION message is 10 bytes long. To get the message ending right at the end of the superframe, it should start in slot 1498 (10 bytes before crossing).

Send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device in such a way that it ends at the end of the superframe.

Positively acknowledge the REPLY_INFORMATION (TID=1) message sent by the Interface device of the component.

Success Criterion:

Verify that the ENT bit is equal to 0 and that the Guide Value is equal to 0.

Coverage:

Section	Statement
8.1.3	If the Guide Byte occurs immediately after the end of a Message, the ENT bit and the Guide Value **shall** be zero.

6.5. Guide Byte Acquisition and Tracking

These tests are meant for Clock Receiving Components. They will mainly address behaviors with messages crossing the superframe boundary.

6.5.1. GBA1 - Message crossing with ENT=1

Test Description:

In most of the case, messages are not crossing the superframe boundary. In this case, the ENT bit is equal to 0 and the Guide Value is equal to 0. The situation does not trigger any special behavior. Devices know they can simply use the first control slot available for messaging.

But, when a message crosses the superframe boundary between the beginning of the Source Address and the end of the Remaining Length (RL) field, the ENT bit is equal to 1. The Guide Value is equal to the number of bytes till the end of the RL field. The active Framer is responsible for the generation of the Guide Byte value. The goal of the test it to verify that a device effectively tracks the guide byte and uses the Guide Value and the ENT bit.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 19.

Stimulus:

In Subframe Mode 19, there are 174 slots available for a message. The REQUEST_INFORMATION message is 10 bytes long. To cross the superframe boundary with ENT=1, the message shall start 1 or 2 bytes before the end of the superframe. Start the message slot 1504 (2 bytes before crossing).

Send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device in such a way that it crosses the superframe boundary with ENT=1.

Positively acknowledge the REPLY_INFORMATION (TID=1) message sent by the Interface device of the component.

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).

Verify that the REPLY_INFORMATION (TID=1) message is sent by the Interface device.

Verify that the LOST_MS, LOST_SFS and LOST_FS IE are all cleared.

Coverage:

Section	Statement
8.3.2	In order to remain synchronized to the Message Channel, a Component **shall** track the Messages appearing in the Message Channel.
8.3.1	[ENT=1] A Component **shall** track the Arbitration Sequence, read the RL field and track the Message until the end.

6.5.2. GBA2 - Message crossing with ENT=0

Test Description:

In most of the case, messages are not crossing the superframe boundary. In this case, the ENT bit is equal to 0 and the Guide Value is equal to 0. The situation does not trigger any special behavior. Devices know they can simply use the first control slot available for messaging.

But, when a message crosses the superframe boundary between the beginning of the Message Code and the end of the Message Payload, the ENT bit is equal to 0. The Guide Value is equal to the number of bytes till the end of the message. The active Framer is responsible for the generation of the Guide Byte value. The goal of the test it to verify that a device effectively tracks the guide byte and uses the Guide Value and the ENT bit.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 19.

Stimulus:

In Subframe Mode 19, there are 174 slots available for a message. The REQUEST_INFORMATION message is 10 bytes long. To cross the superframe boundary with ENT=0, the message shall start 3 to 8 bytes before the end of the superframe. Start the message slot 1472 (4 bytes before crossing).

Send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device in such a way that it crosses the superframe boundary with ENT=0.

Positively acknowledge the REPLY_INFORMATION (TID=1) message sent by the Interface device of the component.

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message is positively acknowledged (PACK).

Verify that the REPLY_INFORMATION (TID=1) message is sent by the Interface device.

Verify that the LOST_MS, LOST_SFS and LOST_FS IE are all cleared.

Coverage:

Section	Statement
8.3.2	In order to remain synchronized to the Message Channel, a Component **shall** track the Messages appearing in the Message Channel.
8.3.1	[ENT=0] A Component **shall** track the Message until the end.

6.5.3. GBA3 - Message crossing with ENT=1 and bad GV

Test Description:

Devices shall continuously monitor the Guide Byte and compare it to their own calculation. If ENT=1, the device is capable of calculating the position of the Remaining Length field because all the necessary information are in the Arbitration and Header fields. Any deviation from the Guide Byte will cause a device to ignore the message and lose the message sync.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 19.

Stimulus:

In Subframe Mode 19, there are 174 slots available for a message. The REQUEST_INFORMATION message is 10 bytes long. To cross the superframe boundary with ENT=1, the message shall start 1 to 5 bytes before the end of the superframe. Start the message slot 1504 (2 bytes before crossing).

Send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device in such a way that it crosses the superframe boundary with ENT=1.

In the following superframe, force the Guide byte to have the value 0x8B (GV=2) instead of 0x84 (GV=1).

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message receives a NORE response.

Verify that the REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device to signify a loss of message sync (LOST_MS=1).

Coverage:

Section	Statement
8.3.3	When a Component loses the Message Synchronization, the Device within the Component that was receiving a Message (if any) **shall** not send any response to the Message
8.3.3	If a Component has a mismatch when comparing the Guide Byte information with its internal Message Channel tracking mechanism, the current Message **shall** be considered aborted.

6.5.4. GBA4 - Message crossing with ENT=0 and bad GV

Test Description:

Devices shall continuously monitor the Guide Byte and compare it to their own calculation. If ENT=0, the device is capable of calculating the position of the end of the message by using the value of the Remaining Length field. Any deviation from the Guide Byte will cause a device to ignore the message and lose the message sync.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 19.

Stimulus:

In Subframe Mode 19, there are 174 slots available for a message. The REQUEST_INFORMATION message is 10 bytes long. To cross the superframe boundary with ENT=1, the message shall start 3 to 8 bytes before the end of the superframe. Start the message slot 1472 (4 bytes before crossing).

Send a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message to the Interface device in such a way that it crosses the superframe boundary with ENT=0.

In the following superframe, force the Guide byte to have the value 0x17 (GV=5) instead of 0x18 (GV=6).

Success Criterion:

Verify that the REQUEST_INFORMATION (TID=1, EC=0x4008) message receives a NORE response.

Verify that the REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device to signify a loss of message sync (LOST_MS=1).

Coverage:

Section	Statement
8.3.3	When a Component loses the Message Synchronization, the Device within the Component that was receiving a Message (if any) **shall** not send any response to the Message
8.3.3	If a Component has a mismatch when comparing the Guide Byte information with its internal Message Channel tracking mechanism, the current Message **shall** be considered aborted.
8.4.2.5	A Component **shall** read the Remaining Length field.

6.5.5. GBA5 - DUT Message crossing with Erroneous Guide Byte

Test Description:

When a device is transmitting a message that crosses the superframe boundary, it must compare the Guide Byte with its own value. If there is a mismatch between both values, the device most use the Guide Byte as being the reference, always. The device must abort the transmission of the message and set the IE LOST_MS.

Initial conditions:

The Clock Receiving Component is in the *Enumerated* state. The Subframe Mode is equal to 19.

Stimulus:

Position the start of a valid REQUEST_INFORMATION (TID=1, EC=0x4008) message in such a way that the REPLY message will cross the superframe boundary with ENT=1.

In the following superframe, force the Guide byte to have the value 0x18 (ENT=0, GV=6) instead of 0x84 (ENT=1, GV=1).

Success Criterion:

Verify that the REPLY_INFORMATION message is aborted. There should not be any control slots driven right after the guide byte. with GV=6, the earliest we can expect to see a message starting is 12 slots after the last Guide slot.

Verify that the REPORT_INFORMATION (EC=0x4008) message is sent by the Interface device to signify a loss of message sync (LOST_MS=1).

Coverage:

Section	Statement
8.3.3	When a Component loses the Message Synchronization, the Device within the Component that was transmitting a Message (if any) **shall** stop transmission of the current Message.
8.3.3	If a Component has a mismatch when comparing the Guide Byte information with its internal Message Channel tracking mechanism, the Component **shall** update its internal tracking mechanism with the Guide Byte information.