BACKPLANE ETHERNET CONSORTIUM

Clause 72 10GBASE-KR
Startup Training Test Suite

Version 0.3

Technical Document

Last Updated: January 23, 2015 3:48 PM

Backplane Ethernet Consortium

University of New Hampshire
InterOperability Laboratory

121 Technology Drive, Suite 2
Durham, NH 03824
Phone: (603) 862-0090
Fax: (603) 862-4181
http://www.iol.unh.edu/consortiums/bp

TABLE OF CONTENTS

MODIFICATION RECORD

Revision	Release Date	Author	Comments
0.1	06/17/2009	Matthew Senerchia	• Initial Preliminary Draft
0.11	06/19/2009	Matthew Senerchia	• Added all possible test names
0.2	09/11/2009	Jon Beckwith	• Removed many excess tests
0.21	09/14/2009	Jon Beckwith	• Added state diagram figures
0.22	09/15/2009	Jon Beckwith	• Major update. • Added nearly all discussions, test procedures, and observable results • Left comments in for 4 tests.
0.23	09/16/2009	Jon Beckwith	• Editorial changes • Added text for max limit and min limit • Removed comments • Added “coefficient update priority” test
0.24	09/16/2009	Jon Beckwith	• Editorial changes
0.25	09/25/2009	Jon Beckwith	• Minor formatting changes • Added Appendix 72.A text • Released for external review
0.30	01/22/2015	Jonathan Leverone	• Minor formatting changes • Changed tests 72.3.2b, 72.3.3c, 72.3.4b, and 72.3.4c to Informative • Editorial changes • Included possible problems for tests 72.3.2, 72.3.3 and 72.3.4

ACKNOWLEDGMENTS

The University of New Hampshire would like to acknowledge the efforts of the following individuals in the development of this test suite.

Jon Beckwith	UNH InterOperability Laboratory
Jonathan Leverone	UNH InterOperability Laboratory
Matthew Senerchia	UNH InterOperability Laboratory

INTRODUCTION

The University of New Hampshire's InterOperability Laboratory (IOL) is an institution designed to improve the interoperability of standards based products by providing an environment where a product can be tested against other implementations of a standard. This particular suite of tests has been developed to help implementers evaluate the functionality of the PMD Control sublayer of their Backplane Ethernet products.

These tests are designed to determine if a product conforms to specifications defined in Clause 72 of the IEEE 802.3-2008 Standard. Successful completion of all tests contained in this suite does not guarantee that the tested device will operate with other devices. However, combined with satisfactory operation in the IOL's interoperability test bed, these tests provide a reasonable level of confidence that the Device Under Test (DUT) will function properly in many Backplane Ethernet environments.

The tests contained in this document are organized in such a manner as to simplify the identification of information related to a test, and to facilitate in the actual testing process. Tests are organized into groups, primarily in order to reduce setup time in the lab environment, however the different groups typically also tend to focus on specific aspects of device functionality. A three-part numbering system is used to organize the tests, where the first number indicates the clause of the IEEE 802.3 standard on which the test suite is based. The second and third numbers indicate the test's group number and test number within that group, respectively. This format allows for the addition of future tests to the appropriate groups without requiring the renumbering of the subsequent tests.

The test definitions themselves are intended to provide a high-level description of the motivation, resources, procedures, and methodologies pertinent to each test. Specifically, each test description consists of the following sections:

Purpose
The purpose is a brief statement outlining what the test attempts to achieve. The test is written at the functional level.

References
This section specifies source material *external* to the test suite, including specific subclauses pertinent to the test definition, or any other references that might be helpful in understanding the test methodology and/or test results. External sources are always referenced by number when mentioned in the test description. Any other references not specified by number are stated with respect to the test suite document itself.

Resource Requirements
The requirements section specifies the test hardware and/or software needed to perform the test. This is generally expressed in terms of minimum requirements, however in some cases specific equipment manufacturer/model information may be provided.

Last Modification
This specifies the date of the last modification to this test.

Discussion
The discussion covers the assumptions made in the design or implementation of the test, as well as known limitations. Other items specific to the test are covered here.

Test Setup
The setup section describes the initial configuration of the test environment. Small changes in the configuration should not be included here, and are generally covered in the test procedure section, below.

Test Procedure
The procedure section of the test description contains the systematic instructions for carrying out the test. It provides a cookbook approach to testing, and may be interspersed with observable results.

Observable Results

This section lists the specific observables that can be examined by the tester in order to verify that the DUT is operating properly. When multiple values for an observable are possible, this section provides a short discussion on how to interpret them. The determination of a pass or fail outcome for a particular test is generally based on the successful (or unsuccessful) detection of a specific observable.

Possible Problems

This section contains a description of known issues with the test procedure, which may affect test results in certain situations. It may also refer the reader to test suite appendices and/or whitepapers that may provide more detail regarding these issues.

GROUP 1: TRAINING FRAME STRUCTURE

Overview:

The tests defined in this section verify the transmission of 10GBASE-KR Startup training frames used in the Backplane Ethernet PMD sublayer defined in Clause 72 of IEEE 802.3-2008.

These tests are designed to verify that the device under test transmits acceptable training frames, which are properly formed with acceptable content.

Test 72.1.1 – Training Frame Encoding

Purpose: To verify that training frames are properly formatted, including a 4 octet frame marker, 32 octet control channel, and 512 octet training pattern.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10 – PMD control function
[2] IEEE Std 802.3-2008, subclause 72.6.10.2 – Training frame structure
[3] IEEE Std 802.3-2008, subclause 72.6.10.2.2 – Control channel encoding

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:

Reference [1] describes the control function and start-up training mechanism that is employed by 10GBASE-KR devices in order to facilitate timing recovery and equalization. This protocol uses the training frame in reference [2] and shown in figure 1 below to convey coefficient information, supply the remote station with a marker to maintain frame synchronization, and provide an example known waveform (a PRBS 11 training pattern) from which the timing recovery and equalization can be obtained. This test verifies proper formatting of the Training frames themselves.

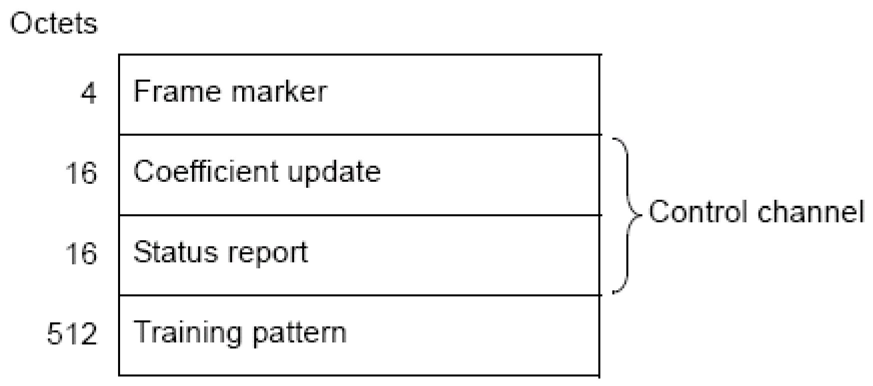


Figure 1: Training frame structure

Test Setup: See Appendix 72.A

Test Procedure:

1. Resolve a 10GBASE-KR link using Clause 73 ANEG.
2. Capture and decode several training frames

Observable Results:

a. The training frame structure should consist of:
    i. A 4 octet frame marker (FFFF0000)
    ii. A 32 octet control channel, data dependent
    iii. A 512 octet training pattern

Possible Problems: None

Test 72.1.2 – Control Channel Encoding

Purpose: To verify that the control channel is properly encoded using differential Manchester encoding.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.2 – Training frame structure
[2] IEEE Std 802.3-2008, subclause 72.6.10.2.2 – Control channel encoding

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:

Reference [1] provides the structure for training frames used in the 10GBASE-KR start-up training process. Reference [2] describes the encoding specific to the control channel portion of the training frame. This data is transmitted at a lower data rate to allow for reception over non-optimally equalized channels. Differential Manchester encoding (DME) is used to guarantee transition density and maintain DC balance. This test verifies proper differential Manchester encoding and control channel formatting.

R	R	P	I	R	R	R	R	R	R	$c_u(+1)$		$c_u(0)$		$c_u(-1)$	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 2: Coefficient update field

rr	R	R	R	R	R	R	R	R	R	$c_s(+1)$		$c_s(0)$		$c_s(-1)$	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3: Status report field

R	Reserved
P	Preset
I	Initialize
c_u	Coefficient Update
rr	RX Ready
c_s	Coeffcient Status

Test Setup: See Appendix 72.A

Test Procedure:

1. Resolve a 10GBASE-KR link using Clause 73 ANEG.
2. Capture and decode several training frames

Observable Results:

a. The control channel should be encoded using differential Manchester encoding
b. The format should reflect that shown in figures 2 and 3 above
c. Reserved bits should be set to 0
d. Bit 15 of the coefficient update and coefficient status fields should be transmitted first

Possible Problems: None

Test 72.1.3 – Training Pattern

Purpose: To verify that the training pattern sent in the training pattern field is derived from the proper polynomial.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.2.2 – Control channel encoding
[2] IEEE Std 802.3-2008, subclause 72 6.10.2.6 – Training pattern

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:

. Reference [1] provides the structure for training frames used in the 10GBASE-KR start-up training process. Reference [2] describes the training pattern used in the training frame, and specifies that it consists of 4094 bits from a PRBS11 generator using the polynomial shown in figure 4, followed by two zeros. This test verifies that the DUT uses the proper polynomial and that it uses a random seed at the start of the training pattern.

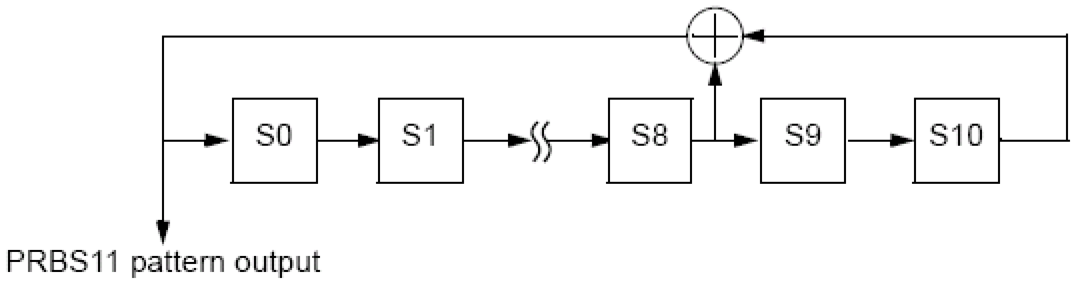


Figure 4: PRBS11 pattern generator used for training pattern

Test Setup: See Appendix 72.A

Test Procedure:

1. Resolve a 10GBASE-KR link using Clause 73 ANEG.
2. Capture and decode several training frames

Observable Results:

a. The training pattern should be generated using the generator shown in figure 4 above.
b. The training pattern seed shall have a random seed at the start of the training pattern.

Possible Problems: None

GROUP 2: STATE DIAGRAM VARIABLES, TIMERS AND FUNCTIONS

Overview:

The tests defined in this section cover operation specific to the reception of 10GBASE-KR training frames used in 10GBASE-KR Startup training defined in Clause 72 of IEEE 802.3-2008.

These tests are designed to verify that the DUT properly implements the various state diagram variables, timers, and counters as defined in subclause 72.6.10.3.1, 72.6.10.3.2, and 72.6.10.3.3 respectively.

Test 72.2.1 – Initialize

Purpose: To verify that the DUT properly implements the INITIALIZE control.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.2.3.2 – Initialize
[2] IEEE Std 802.3-2008, subclause 72.6.10.4.2 – Training
[3] IEEE Std 802.3-2008, subclause 72.7.1.11 – Transmitter output waveform requirements

Resource Requirements: See Appendix 72.A

Last Modification: September 14, 2009

Discussion:

Reference [1] describes the initialize control is sent to configure the transmit equalizer to the INITIALIZE state, where the transmit output waveform meets the requirements found in reference [2].

Test Setup: See Appendix 72.A

Test Procedure:

Parts a-e:

1. Use a traffic generator to configure the DUT for initialize operation.

Part f:

2. Use a traffic generator sending updated messages until max_wait_timer expires.

Observable Results:

a. The initialize control shall only be initially sent when all coefficient status fields indicate not_updated, and continue to be sent until no coefficient status field indicates not_updated.
b. During initialization, R_{pre} should be 1.29 +/- 10%
c. During initialization, R_{pst} should be 2.57 +/- 10%.
d. At the start of training, c(0) shall be set such that the constraints found in reference [3] are satisfied.
e. The peak-to-peak differential output voltage should be greater than 800 mV for a 1010 pattern.
f. The DUT should never issue an INITIALIZE request.

Possible Problems: None

Test 72.2.2 – Preset

Purpose: To verify that the DUT properly implements the preset control

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.2.3.1 – Preset
[2] IEEE Std 802.3-2008, Figure 72.6.10.4 – State Diagrams

Resource Requirements: See Appendix 72.A

Last Modification: September 11, 2009

Discussion:

Reference [1] describes the preset control as a state where equalization is turned off. In this state, the pre-cursor c(-1) and post-cursor c(+1) are turned off, and the main tap c(0) is set to its' maximum value.

Test Setup: See Appendix 72.A

Test Procedure:

Parts a-d:

1. Use a traffic generator to send the DUT PRESET request frames.

Part e:

2. Use a traffic generator sending updated messages until max_wait_timer expires.

Observable Results:

a. The DUT should turn off equalization, setting c(-1) and c(+1) to 0, and c(0) to its' maximum value.
b. The initialize field shall be set to zero.
c. Maximum status shall be returned when the main coefficient is updated.
d. Maximum status shall be returned for the pre- and/or post-cursor coefficients when the coefficient is updated and zero is its' maximum supported value. Updated status shall be returned when it supports additional settings above zero.
e. The DUT should never issue a PRESET request.

Possible Problems: None

Test 72.2.3 – Max Limit

Purpose: To verify that the DUT properly sets a max limit for each coefficient.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.3.1 – State Diagram variables
[2] IEEE Std 802.3-2008, subclause 72.7.1.11 – Transmitter output waveform requirements
[3] IEEE Std 802.3-2008, subclause 72.7.1.4 – Output amplitude

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:

Reference [1] describes that the max_limit variable contains the maximum tap coefficient value. This variable is subject to the constraints in reference [2]. Reference [2] states that any coefficient increment that would result in a waveform amplitude violation (reference [3]) shall return a status value of maximum for that coefficient. This test observes the output amplitude when an increment request is issued when the status returned is maximum.

Test Setup: See Appendix 72.A

Test Procedure:

1. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to increment c(-1).
2. Keep incrementing until the DUT reports maximum in its' coefficient status.
3. Observe the updated waveform
4. Repeat for c(0) and c(1).

Observable Results:

a. Any coefficient update equal to increment that would result in a violation of 72.7.1.4 shall return a coefficient status value maximum for that coefficient

Possible Problems: None

Test 72.2.4 – Min Limit

Purpose: To verify that the DUT properly sets a min limit for each coefficient.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.3.1 – State Diagram variables
[2] IEEE Std 802.3-2008, subclause 72.7.1.11 – Transmitter output waveform requirements
[3] IEEE Std 802.3-2008, subclause 72.7.1.4 – Output amplitude

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:

Reference [1] describes that the min_limit variable contains the minimum tap coefficient value. This variable is subject to the constraints in reference [2]. Reference [2] states that any coefficient decrement that would result in a waveform amplitude violation (reference [3]) shall return a status value of minimum for that coefficient. This test observes the output amplitude when a decrement request is issued when the status returned is maximum.

Test Setup: See Appendix 72.A

Test Procedure:

1. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to decrement c(-1).
2. Keep decrementing until the DUT reports minimum in its' coefficient status.
3. Observe the updated waveform
4. Repeat for c(0) and c(1).

Observable Results:

a. Any coefficient update equal to decrement that would result in a violation of 72.7.1.4 shall return a coefficient status value minimum for that coefficient

Possible Problems: None

Test 72.2.5 – Update Status

Purpose: To verify that the correct update status is returned.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.2.4.5 – Coefficient (k) status
[2] IEEE Std 802.3-2008, subclause 72.6.10.3.1 – Variables
[3] IEEE Std 802.3-2008, Figure 72-6 – Coefficient update state diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:

Reference [1] describes four encodings which convey the current status of the three coefficients undergoing optimization. They are: not_updated, updated, maximum, and minimum. When the DUT is not in a maximum or minimum state, a request to increment should yield a status of updated. When an increment request is sent when the DUT is already at Maximum, a status of maximum should be reported. When a decrement request is sent when the DUT is already at Minimum, a status of minimum should be reported. After processing a coefficient update and a hold request is sent, the DUT should report not_updated.

Test Setup: See Appendix 73.A

Test Procedure:

Part a: Updated:

1. Send the DUT an increment or decrement request, when the DUT is not at its' maximum or minimum setting.

Part b: Maximum:

2. Send the DUT an increment request, when the DUT is at its' maximum setting

Part c: Minimum:

3. Send the DUT a decrement request, when the DUT is at its' minimum setting..

Part d: Not_updated

4. With the DUT already in updated, maximum, or minimum, send the DUT a hold request.

Observable Results:

a. The DUT should report updated
b. The DUT should report maximum
c. The DUT should report minimum
d. The DUT should report not_updated

Possible Problems: None

Test 72.2.6 – Max Wait Timer

Purpose: To verify that the DUT's max_wait_timer

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.3.2 – Timers
[2] IEEE Std 802.3-2008, Figure 72-5 – Training state diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:

Max wait timer is defined as the amount of time the DUT attempts to train before entering the TRAINING_FAILURE state. This can easily be confused with link_fail_inhibit_timer from Clause 73 Auto-Negotiation. The time allocated for max_wait_timer (495-505ms) is less than that allocated for link_fail_inhibit_timer (500-510ms) to allow for initialization of the 10GBASE-KR PHY after Auto-Negotiation completes.

Test Setup: See Appendix 72.A

Test Procedure:

1. Using Clause 73 ANEG, resolve a 10GBASE-KR link.
2. Measure the amount of time the DUT transmits KR training signaling.
3. Repeat multiple times and take the maximum of the observed times.

Observable Results:

a. The value of max_wait_timer should be between 495 – 505 ms.

Possible Problems:

If the DUT's max_wait_timer and the link_fail_inhibit_timer values are between 500-505ms, there is a possibility link_fail_inhibit_timer could mask the max_wait_timer value.

Test 72.2.7 – Wait Timer

Purpose: To verify that the DUT transmits wait_timer additional training frames after the remote receiver is ready to receive data.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.3.2 – Timers
[2] IEEE Std 802.3-2008, Figure 72-5 – Training state diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 11, 2009

Discussion:

Reference [1] provides an overview of the timers used during the 10GBASE-KR startup training process. During this process, once the DUT sees remote_rx_ready, then it should enter LINK_READY and transmit wait_timer frames, unless the remote station sends !remote_rx_ready.

Test Setup: See Appendix 72.A

Test Procedure:

1. Using the traffic generator, send training frames with the Receiver ready bit set to true.
2. Wait for the DUT to set the receiver ready bit to true.
3. When the DUT sets its' receiver ready bit, begin counting the number of frames, starting with the second frame.

Observable Results:

a. The value for wait_timer should be between 100 to 300 training frames.

Possible Problems: None

Test 72.2.8 – Control Channel violation

Purpose: To verify that the DUT properly ignores the contents of the control channel if a differential Manchester coding violation is found.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.2.2 – Control channel encoding

Resource Requirements: See Appendix 72.A

Last Modification: September 14, 2009

Discussion:

If the control channel portion of the training frame is improperly encoded using differential Manchester encoding, then the contents of the control channel for that frame should be ignored. The data cell length shall be 8 10GBASE-KR UI. While "improperly encoded" could have infinite interpretations, here the scope is limited to varying the length of the data cell.

Test Setup: See Appendix 72.A

Test Procedure:

1. Send the DUT PRESET frame requests using a control channel that uses a data cell length of 4 10GBASE-KR UI.
2. Monitor the DUT's response.
3. Repeat for a data cell length of 8.

Observable Results:

a. The DUT should only respond to the PRESET request for a data cell length of 8 10GBASE-KR UI.

Possible Problems: None

Test 72.2.9 – Coefficient Update priority

Purpose: To verify that the DUT follows the appropriate priority if multiple actions are requested.

References:

[1] IEEE Std 802.3-2008, subclause 72.6.10.2.3.3 – Coefficient (k) update
[2] IEEE Std 802.3-2008, subclause 72.6.10.3.4 – Functions

Resource Requirements: See Appendix 72.A

Last Modification: September 64, 2009

Discussion:

Reference [1] states that a preset/initialize request shall not be sent in combination with a coefficient increment/decrement request. Reference [2] provides a list showing the priority the COEFF_UPDATE function should follow if multiple actions are requested. The priority is: (1) preset, (2) initialize, and (3) inc/dec.

Test Setup: See Appendix 72.A

Test Procedure:

Part a: Preset, Initialize, and Coefficient Update

1. Send the DUT a frame containing a PRESET and INTIALIZE request, and a request to increment all coefficients
2. Monitor the DUT's response.

Part b: Preset and Initialize

3. Send the DUT a frame containing a PRESET and INTIALIZE request.
4. Monitor the DUT's response.

Part c: Preset and Coefficient Update

5. Send the DUT a frame containing a PRESET request and a request to increment all coefficients
6. Monitor the DUT's response.

Part d: Initialize and Coefficient Update

7. Send the DUT a frame containing a INITIALIZE request and a request to increment all coefficients
8. Monitor the DUT's response.

Observable Results:

a. The DUT should configure its' transmitter for PRESET operation
b. The DUT should configure its' transmitter for PRESET operation
c. The DUT should configure its' transmitter for PRESET operation
d. The DUT should configure its' transmitter for INITIALIZE operation

Possible Problems: None

GROUP 3: FRAME LOCK STATE DIAGRAM

Overview:
The tests defined in this section cover operation specific to the reception of Training Frames used in Backplane Ethernet Startup Training protocol defined in Clause 72.6.10 of IEEE 802.3-2008.

These tests are designed to verify that the DUT properly transitions through the states in the Frame Lock state diagram, which is defined in Figure 72-4.

Test 72.3.1 – TEST_MARKER

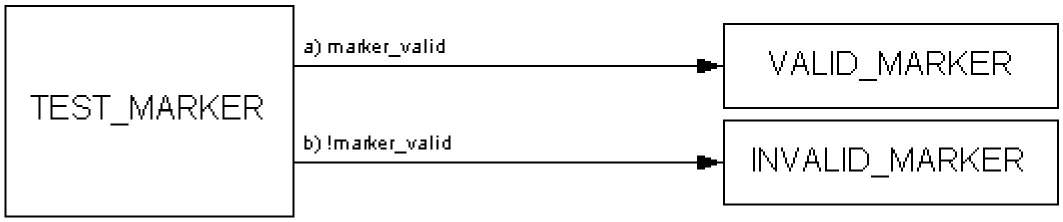
Purpose: To verify that the DUT properly exits the TEST MARKER state under the appropriate condition.

References:
[1] IEEE Std 802.3-2008, Figure 72-4 – Frame Lock State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:
Inside the frame lock state diagram, frame synchronization is achieved through the use of a frame marker – a series of 16 ones followed by 16 zeros. The TEST_MARKER state is to determine whether the received frame marker is valid or not.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Valid markers
1. Send the DUT PRESET frame requests using a valid marker.

Part b: Invalid markers
2. Send the DUT PRESET frame requests using an invalid marker consisting of 8 ones and 8 zeros.

Observable Results:
a. The DUT should process the request and c(0) should report maximum.
b. The DUT should not lock accept the marker, and should not process the request

Possible Problems: None

Test 72.3.2 – VALID_MARKER

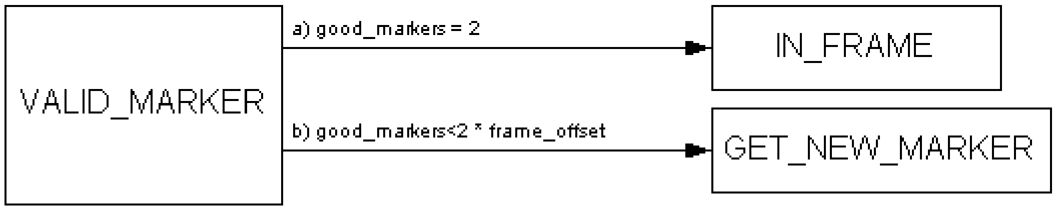
Purpose: To verify that the DUT properly exits the VALID MARKER state under the appropriate conditions.

References:
[1] IEEE Std 802.3-2008, Figure 72-4 – Frame Lock State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 14, 2009

Discussion:
In order to process any frame data, a frame containing a valid frame marker must be received at least 2 times. This marker must also be followed by 544 octets of data.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Valid markers
1. Send the DUT 2 frames containing PRESET requests

Part b: Valid markers after one invalid marker
2. Send the DUT an alternating sequence consisting of:
   i. 1 PRESET request containing a valid frame marker
   ii. 1 PRESET request with an invalid frame marker (8 ones and 8 zeros)

Observable Results:
a. The DUT should process the requests, indicating a transition from VALID_MARKER to IN_FRAME.
b. INFORMATIVE: The DUT may or may not process the request

Possible Problems: According to IEEE 802.3 2008, there is no link between figures 72-4 (Frame lock diagram) and 72-6 (Coefficient update state diagram). This means that frame_lock is not required to be true in order for a DUT to process a preset request. Currently Part c of this test is reported on a information basis only.

Test 72.3.3 – INVALID_MARKER

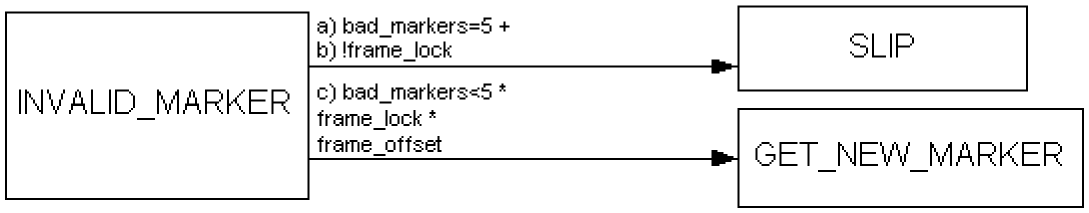
Purpose: To verify that the DUT properly exits the INVALID_MARKER state under the appropriate conditions.

References:
[1] IEEE Std 802.3-2008, Figure 72-4 – Frame Lock State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 16, 2009

Discussion:
In the frame lock state diagram, the INVALID_MARKER state handles markers when they have been flagged as a bad marker. If only one marker is flagged as bad, then the DUT should not lose frame lock but instead attempt to get a new marker.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Bad markers before request
1. Force a SLIP to occur by sending 5 training frames with an invalid frame marker.
2. Send 2 frames with a preset request

Part b: Loss of frame lock before request
3. Force a loss of frame lock by inserting (n) extra ones before the frame marker.
4. Send 2 frames with a preset request

Part c: One invalid frame marker
5. Send the DUT 1 PRESET request containing a valid frame marker, followed by 1 with an invalid frame marker, then 2 with a valid frame marker.

Observable Results:
a. The DUT should SLIP and then accept the preset request.
b. The DUT should SLIP and then accept the preset request
c. INFORMATIVE: The DUT may or may not process the preset request until after the invalid frame marker.

Possible Problems: According to IEEE 802.3 2008, there is no link between figures 72-4 (Frame lock diagram) and 72-6 (Coefficient update state diagram). This means that frame_lock is not required to be true in order for a DUT to process a preset request. Currently Part c of this test is reported on a information basis only.

Test 72.3.4 – IN_FRAME

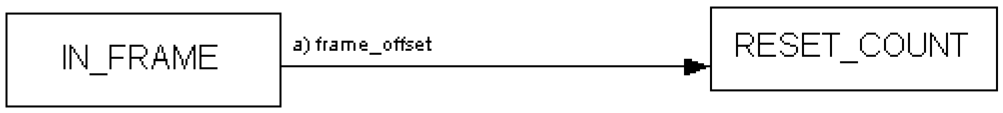
Purpose: To verify that the DUT properly exits the IN_FRAME state under the appropriate conditions.

References:
[1] IEEE Std 802.3-2008, Figure 72-4 – Frame Lock State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 16, 2009

Discussion:
If the training frame is too short and a new frame marker is received prior to the end of the frame, this will cause invalid data to be observed for the end of the training pattern, and an invalid frame marker to be detected in the next frame.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Properly formed training frame
1. Send the DUT PRESET requests using a valid length frame

Part b: Training pattern too short
2. Send the DUT 1 properly formed PRESET request
3. Send the DUT 1 PRESET request, omitting the last octet of the training pattern.
4. Send the DUT 3 properly formed PRESET requests

Part c: Training pattern too long
5. Send the DUT 1 properly formed PRESET request
6. Send the DUT 1 PRESET request, adding an extra octet to the training pattern.
7. Send the DUT 3 properly formed PRESET requests

Observable Results:
a. The DUT should accept the PRESET request
b. INFORMATIVE: The DUT may or may not accept only the last two PRESET requests
c. INFORMATIVE: The DUT may or may not accept only the last two PRESET requests

Possible Problems: According to IEEE 802.3 2008, there is no link between figures 72-4 (Frame lock diagram) and 72-6 (Coefficient update state diagram). This means that frame_lock is not required to be true in order for a DUT to process a preset request. Currently Parts b and c of this test are reported on a information basis only.

GROUP 4: TRAINING STATE DIAGRAM

Overview:

The tests defined in this section cover operation specific to the reception of Training Frames used in Backplane Ethernet Startup Training protocol defined in Clause 72.6.10 of IEEE 802.3-2008.

These tests are designed to verify that the DUT properly transitions through the states in the Training state diagram, which is defined in Figure 72-5.

Test 72.4.1 – SEND_TRAINING

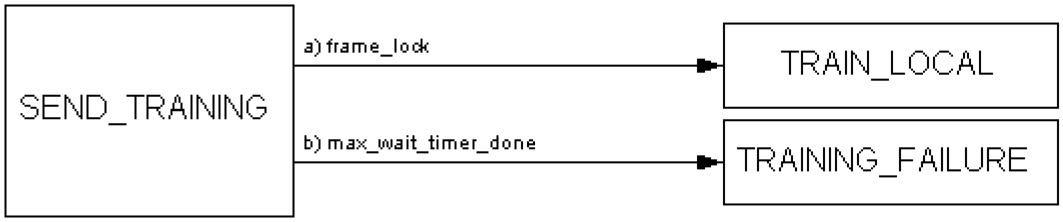
Purpose: To verify that the DUT properly exits the SEND TRAINING state under the appropriate conditions only.

References:
[1] IEEE Std 802.3-2008, Figure 72-5 – Training State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:
When KR Training has been enabled (mr_training_enable=TRUE), the DUT should proceed to SEND_TRAINING. Here it will exit when it has established frame lock or wait until max_wait_timer has expired and training has failed.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Establish frame lock
1. Once a frame lock has been established, the DUT should begin training it's receiver by requesting a preset, initialize, or increment/decrement

Part b Training failure:
2. Using Clause 73 ANEG, resolve a 10GBASE-KR link, continue to send the DUT request until max_wait_timer expires.

Observable Results:
a. Once a frame lock has been established, the DUT should begin training it's receiver by requesting a preset, initialize, or increment/decrement
b. The DUT should wait until max_wait_timer expires and enter the TRAINING_FAILURE state.

Possible Problems: None

Test 72.4.2 – TRAIN_LOCAL

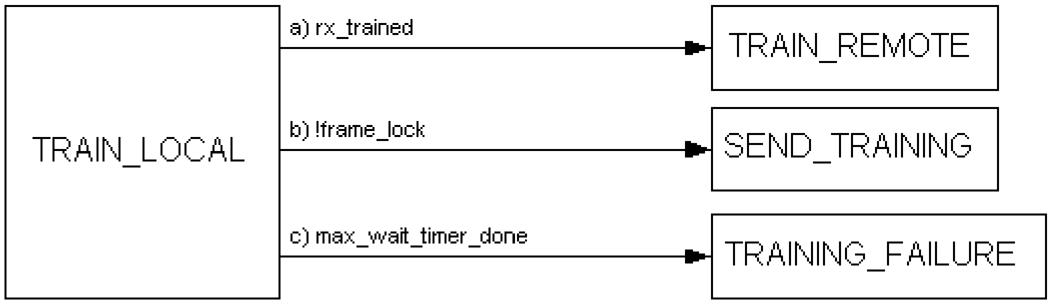
Purpose: To verify that the DUT properly exits the TRAIN_LOCAL state under the appropriate conditions only.

References:
[1] IEEE Std 802.3-2008, Figure 72-5 – Training State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 14, 2009

Discussion:
After obtaining frame lock, the DUT begins training its' receiver and the remote transmitter by means of coefficient optimization. Once the remote transmitters and the local receiver have been optimized for the interconnecting channel, rx_trained is set to TRUE, and the remote receiver begins training. If frame lock is lost, the DUT ceases training and enters SEND_TRAINING. If the local receiver and remote transmitter are unable to successfully train, then max_wait_timer expires and training has failed. It should be noted that while in the TRAIN_LOCAL state, the local device will be training the remote device.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Complete training
1. Using a traffic generator, respond to any requests (initialize, preset, increment, decrement) such that the DUT is able to complete training

Part b: Lose frame lock
2. Once the DUT has begun sending initialize, preset, or increment/decrement requests, use a traffic generator to force the DUT to lose frame lock by inserting (n) extra ones before the frame marker.

Part c: Training failure
3. Use a traffic generator to send training to the DUT, such that it is able to obtain frame lock, but un-optimized enough so that it issues an initialize, preset, or increment/decrement request.
4. Continue sending training signaling, not responding to the requests.

Observable Results:
a. After the DUT completes training of its receiver, it should set the RX Ready bit to 1.
b. If frame lock is lost during a coefficient exchange, the DUT should restart training.
c. If the DUT is not able to train its' receivers within max_wait_timer, it should report a training failure.

Possible Problems: None

Test 72.4.3 – TRAIN_REMOTE

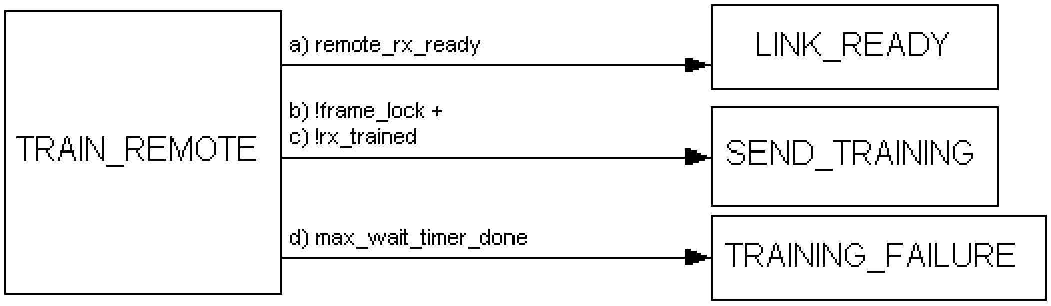
Purpose: To verify that the DUT properly exits the TRAIN_REMOTE state under the appropriate conditions only.

References:
[1] IEEE Std 802.3-2008, Figure 72-5 – Training State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:
After the local device has finished training and rx_trained=TRUE, it transitions to TRAIN_REMOTE. If the remote is finished training, remote_rx_ready=TRUE and the link is ready. If frame lock has been lost, or the local device requires retraining, training is restarted. If the remote receiver and local transmitter are unable to be trained before max_wait_timer expires, training fails.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Remote training complete
1. Use a traffic generator to enable the DUT to complete training of it's local receivers
2. Send frames with the RX ready bit set to 1. Modify transmit waveform and update status report as necessary to complete training for DUT's receiver.

Part b: Loss of frame lock
3. Use a traffic generator to enable the DUT to complete training of it's local receivers
4. Once rx_trained=TRUE, cause the DUT to lose frame lock by inserting (n) extra ones before the frame marker

Part c: Require re-training of DUT's receiver
5. Use a traffic generator to enable the DUT to complete training of it's local receivers
6. Once rx_trained=TRUE, change the transmitted waveform so that the DUT is forced to re-train its' receiver. This will cause the DUT to turn off its' Rx ready bit.

Part d: Training failure
7. Use a traffic generator to enable the DUT to complete training of it's local receivers
8. Continue sending training frames without the Rx ready bit set for a period longer than max_wait_timer.

Observable Results:
a. The DUT should exit the TRAIN_REMOTE state, enter LINK_READY, and begin wait_timer.
b. The DUT should attempt to re-lock and restart training.
c. The DUT should attempt to re-train its' receiver by turning off the Rx ready bit and issuing a preset, initialize, or coefficient increment/decrement.

d. The DUT should exit the TRAIN_REMOTE state and cease training.

Possible Problems: None

Test 72.4.4 – LINK_READY

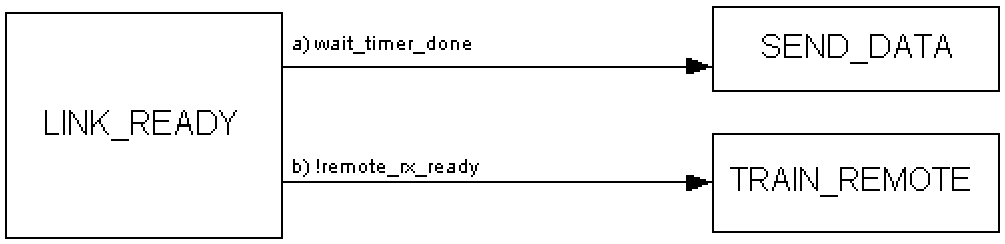
Purpose: To verify that the DUT properly exits the LINK_READY state under the appropriate condition only.

References:
[1] IEEE Std 802.3-2008, Figure 72-5 – Training State Diagram

Resource Requirements: See Appendix 72.A

Last Modification: September 15, 2009

Discussion:
After both the local and remote devices have completed training, rx_trained and remote_rx_ready are TRUE. If there is no change in remote_rx_ready, then an additional wait_timer training frames are sent before transitioning to SEND_DATA. If remote_rx_ready changes to FALSE, then the remote device is re-trained.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Successful training completion
1. Use a traffic generator to bring the DUT into the LINK READY state by training its' receiver and sending at least 3 training frames with the Rx ready bit set to one.
2. Continue sending training frames while the DUT transitions to SEND_DATA.

Part b: Remote rx requires re-training
3. Use a traffic generator to bring the DUT into the LINK READY state by training its' receiver and sending 3 training frames with the Rx ready bit set to one.
4. Set the Rx ready bit to zero.

Observable Results:
a. The DUT should send wait_timer (100-300) additional training frames and then begin sending data.
b. The DUT should continue to send its' ready frame and attempt to train the remote..

Possible Problems: None

GROUP 5: COEFFICIENT UPDATE STATE DIAGRAM

Overview:

The tests defined in this section cover operation specific to the processing of coefficient update requests of Training Frames used in Backplane Ethernet Startup protocol defined in Clause 72.6.10 of IEEE 802.3-2008.

These tests are designed to verify that the DUT properly transitions through the states in the Coefficient Update state diagram, which is defined in Figure 72-6.

Test 72.5.1 – NOT_UPDATED

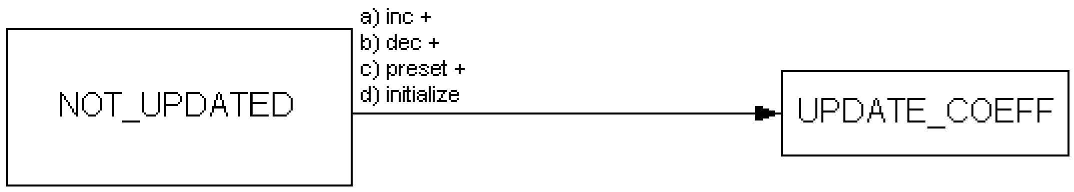
Purpose: To verify that the DUT properly transitions out of the NOT_UPDATED state.

References:
[1] IEEE Std 802.3-2008, subclause 72.6.10 – PMD control function

Resource Requirements: See Appendix 72.A

Last Modification: September 11, 2009

Discussion:
The default state for all taps is not_updated. When a new request to initialize, preset, increment, or decrement is received, it should be processed accordingly.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Increment
1. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to increment c(-1), but making sure that it will not cause the DUT to reach its maximum limit..
2. Observe the updated waveform
3. Repeat for c(0) and c(1).

Part b: Decrement
4. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to decrement c(-1), but making sure that it will not cause the DUT to reach its minimum limit..
5. Observe the updated waveform
6. Repeat for c(0) and c(1)

Part c: Preset
7. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to preset its' transmitter.
8. Observe the waveform

Part d: Initialize
9. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to initialize its' transmitter.
10. Observe the waveform

Observable Results:
a. The DUT should increment only the appropriate coefficient
b. The DUT should decrement only the appropriate coefficient.
c. The DUT should configure its transmitter for preset operation.
d. The DUT should configure its transmitter for initialize operation.

Possible Problems: None

Test 72.5.2 – UPDATE_COEFF

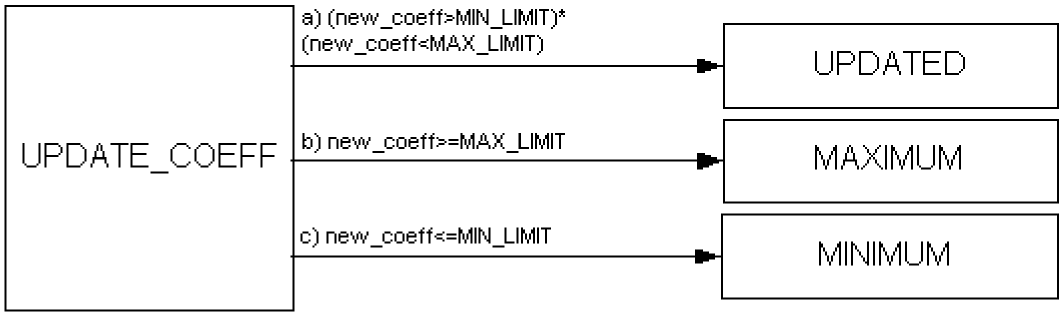
Purpose: To verify that the DUT properly transitions out of the UPDATE_COEFF state..

References:
[1] IEEE Std 802.3-2008, subclause 72.6.10 – PMD control function

Resource Requirements: See Appendix 72.A

Last Modification: September 16, 2009

Discussion:
When a DUT reaches the maximum or minimum, it should properly report it and not update.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Verifying proper updates.
1. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to increment c(-1), but making sure that it will not cause the DUT to reach its maximum limit..
2. Observe the updated waveform
3. Repeat for c(0) and c(1).

Part b: Testing maximum status
4. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to increment c(-1).
5. Keep incrementing until the DUT reports maximum in its' coefficient status.
6. Observe the updated waveform
7. Repeat for c(0) and c(1).

Part c: Testing minimum status
8. Configure the traffic generator to send a training frame to the DUT, instructing the DUT to decrement c(-1).
9. Keep incrementing until the DUT reports minimum in its' coefficient status.
10. Observe the updated waveform
11. Repeat for c(0) and c(1)

Observable Results:
a. The DUT should increment only the appropriate coefficient
b. The DUT should not update its waveform
c. The DUT should not update its waveform.

Possible Problems: None

Test 72.5.3 – MAXIMUM

Purpose: To verify that once the DUT enters the MAXIMUM state, it will not leave the state until it receives a hold.

References:

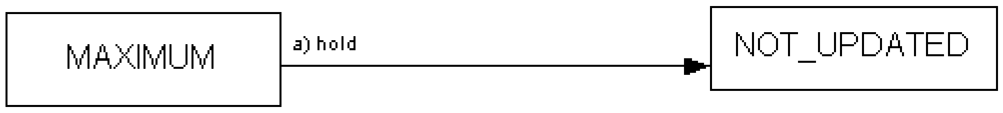
[1] IEEE Std 802.3-2008, subclause 72.6.10 – PMD control function

Resource Requirements: See Appendix 72.A

Last Modification: September 16, 2009

Discussion:

Once the coefficient has been updated and the status field has been changed to maximum, the DUT should not change the coefficients unless a hold is received. Hold is only set to TRUE when the coefficient update field is hold and neither preset nor initialize are activated.



Test Setup: See Appendix 72.A

Test Procedure:

Part a: Increment without holding

1. Send the DUT a request to increment c(0)
2. When the status changes to updated, request a hold.
3. Repeat steps 1 and 2 until the status changes to maximum.
4. Continue sending increment requests, without sending hold
5. Repeat for c(-1) and c(1)

Part b: Decrement without holding

6. Repeat steps 1-3
7. Send decrement requests, without sending hold
8. Repeat for c(-1) and c(1)

Part c: Preset without holding

9. Repeat steps 1-3
10. Send preset requests, without sending hold

Part d: Initialize without holding

11. Repeat steps 1-3
12. Send initialize requests, without sending hold

Observable Results:

a. The DUT should ignore the increment requests
b. The DUT should ignore the decrement requests
c. The DUT should ignore the preset requests.
d. The DUT should ignore the initialize requests

Possible Problems: None

Test 72.5.4 – UPDATED

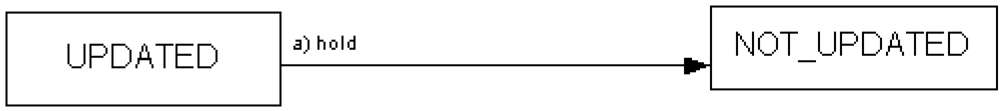
Purpose: To verify that once the DUT enters the UPDATED state, it will not leave the state until it receives a hold.

References:
[1] IEEE Std 802.3-2008, subclause 72.6.10 – PMD control function

Resource Requirements: See Appendix 72.A

Last Modification: September 14, 2009

Discussion:
Once the coefficient has been updated and the status field has been changed to updated, the DUT should not change the coefficients unless a hold is received. Hold is only set to TRUE when the coefficient update field is hold and neither preset nor initialize are activated.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Increment without holding
1. Send the DUT a request to increment c(0)
2. When the status changes to updated, request a hold.
3. Continue sending increment requests, without sending hold
4. Repeat for c(-1) and c(1)

Part b: Decrement without holding
5. Repeat steps 1-2.
6. Send decrement requests, without sending hold
7. Repeat for c(-1) and c(1)

Part c: Preset without holding
8. Repeat steps 1-3
9. Send preset requests, without sending hold

Part d: Initialize without holding
10. Repeat steps 1-3
11. Send initialize requests, without sending hold

Observable Results:
a. The DUT should ignore the increment requests.
b. The DUT should ignore the decrement requests
c. The DUT should ignore the preset requests.
d. The DUT should ignore the initialize requests

Possible Problems: None

Test 72.5.5 – MINIMUM

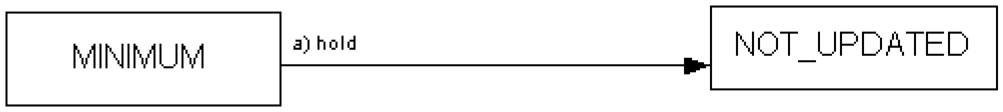
Purpose: To verify that once the DUT enters the MINIMUM state, it will not leave the state until it receives a hold

References:
[1] IEEE Std 802.3-2008, subclause 72.6.10 – PMD control function

Resource Requirements: See Appendix 72.A

Last Modification: September 14, 2009

Discussion:
Once the coefficient has been updated and the status field has been changed to minimum, the DUT should not change the coefficients unless a hold is received. Hold is only set to TRUE when the coefficient update field is hold and neither preset nor initialize are activated.



Test Setup: See Appendix 72.A

Test Procedure:
Part a: Decrement without holding
1. Send the DUT a request to decrement c(0)
2. When the status changes to updated, request a hold.
3. Continue sending decrement requests, without sending hold
4. Repeat for c(-1) and c(1)

Part b: Increment without holding
5. Repeat steps 1-2.
6. Send increment requests, without sending hold
7. Repeat for c(-1) and c(1)

Part c: Preset without holding
8. Repeat steps 1-3
9. Send preset requests, without sending hold

Part d: Initialize without holding
10. Repeat steps 1-3
11. Send initialize requests, without sending hold

Observable Results:
a. The DUT should ignore the decrement requests.
b. The DUT should ignore the increment requests
c. The DUT should ignore the preset requests.
d. The DUT should ignore the initialize requests

Possible Problems: None

APPENDICES

Overview:

Test suite appendices are intended to provide additional low-level technical detail pertinent to specific tests contained in this test suite. These appendices often cover topics that are outside of the scope of the standard, and are specific to the methodologies used for performing the measurements in this test suite. Appendix topics may also include discussion regarding a specific interpretation of the standard (for the purposes of this test suite), for cases where a particular specification may appear unclear or otherwise open to multiple interpretations.

Scope:

Test suite appendices are considered informative supplements, and pertain solely to the test definitions and procedures contained in this test suite.

Appendix 72.A – Test Setups

Purpose: To specify the measurement hardware, test fixtures, and setups used in this test suite

References:

[1] IEEE Std 802.3-2008, subclause 72.7.1.1 – Test Fixture

Last Modification: September 25, 2009

Discussion:

For tests that do not include a response from the test system, a digital storage oscilloscope will be used, as shown in figure 72.A-1. For tests involving transmitter waveform verification (such as 72.2.1 parts b and c), the DSO is a sampling oscilloscope, and has a minimum 18GHz bandwidth. For tests requiring decoding of the DUT's transmitted frame, a real-time DSO is used.

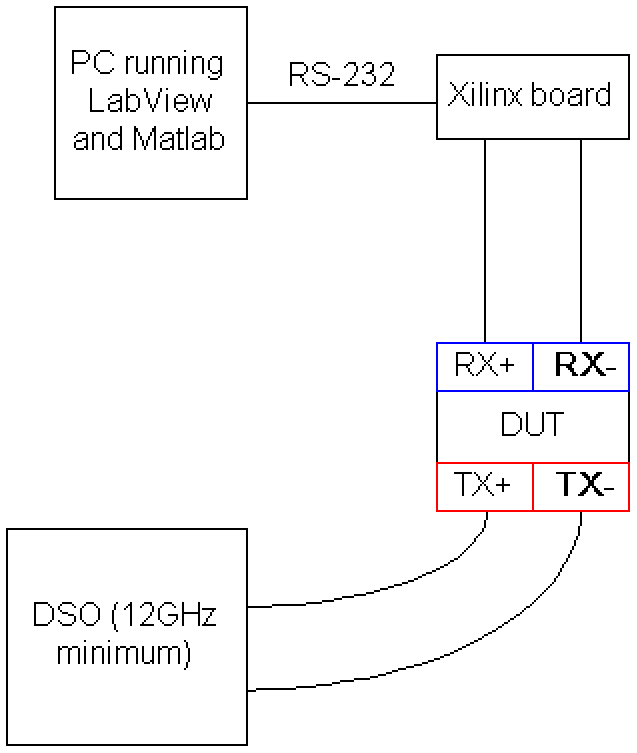


Figure 72.A-1: Setup #1 for performing startup tests

For tests that require adaptive responses based on the DUT's requests, additional processing capabilities are planned for inclusion into the Xilinx board. These modifications are in development.