University of New Hampshire
InterOperability Laboratory

Gigabit Ethernet Test Suite

Clause 36
Physical Coding Sublayer (PCS)

Technical Document
Version 3.1 – January 5, 2018

Gigabit Ethernet Consortium
University of New Hampshire
InterOperability Laboratory

21 Madbury Rd #100
Durham, NH 03824
Phone: +1-603-862-0090
http://www.iol.unh.edu/services/testing/ge/

MODIFICATION RECORD

- January 5, 2018 Added Test #36.3.5
- May 30, 2007 Updates to
  - Major revision of all tests.
  - Added Test# 36.1.5
- January 25, 2005 Updated references to reflect IEEE Std. 802.3, 2005 Edition
- June 9, 2003 Updates to
  - Test# 36.1.1
  - Test# 36.3.2
  - ANNEX B
- August 6, 2002 Updates to
  - Test# 36.1.1
  - Test# 36.1.2
  - Test# 36.1.3
  - Test# 36.1.4
- July 31, 2002 Updates to
  - Change document type HTML to PDF
  - Restructured Annexes
  - Test# 36.1.1
  - Test# 36.1.2
  - Test# 36.1.3
  - Test# 36.1.4
  - Test# 36.2.1
  - Test# 36.3.2
- December 3, 1998 Update to Test# 36.3.2
- September 16, 1998 Updates to
  - Test# 36.1.1
  - Test# 36.1.2
  - Test# 36.1.3
  - Test# 36.1.4
- August 26, 1998 Update to Test# 36.2.4
- August 3, 1998 Update to Test# 36.3.2
- June 24, 1998 Update to Test 36.2.4
- June 8, 1998 Updates to
  - Test#36.2.1
  - Test#36.2.2
  - Test#36.2.3
  - Test#36.3.1
  - Test#26.3.4
- March 5, 1998 Minor Adjustments
- February 18, 1998 Version 1.0 Released

ACKNOWLEDGMENTS

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

Aldobino M. Braga	University of New Hampshire InterOperability Lab
Jon Frain	University of New Hampshire InterOperability Lab
Michael S. Henninger	University of New Hampshire InterOperability Lab
Robert E. Noseworthy	University of New Hampshire InterOperability Lab
Matthew Plante	University of New Hampshire InterOperability Lab

INTRODUCTION

Overview

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 suite of tests has been developed to help implementers evaluate the functionality of their Clause 36 Physical Coding Sublayer (PCS) based products. The tests do not determine if a product conforms to IEEE 802.3 standard, nor are they purely interoperability tests. Rather, they provide one method to isolate problems within a PCS device. 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 well in most environments.

Organization of Tests

The tests contained in this document are organized to simplify the identification of information related to a test and to facilitate in the actual testing process. Each test contains an identification section that describes the test and provides cross-reference information. The discussion section covers background information and specifies why the test is to be performed. Tests are grouped in order to reduce setup time in the lab environment. Each test contains the following information:

Test Number

The Test Number associated with each test follows a simple grouping structure. Listed first is the Test Group Number followed by the test's number within the group. This allows for the addition of future tests within the appropriate groups of the test suite without requiring the renumbering of the subsequent tests.

Purpose

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

References

The references section lists cross-references to the pertaining IEEE 802.3-2015 Standard and other documentation that might be helpful in understanding and evaluating the test and results.

Resource Requirements

The requirements section specifies the hardware, and test equipment that will be needed to perform the test. The items contained in this section are special test devices or other facilities, which may not be available on all devices.

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 configuration of the test environment. Small changes in the configuration should be included in the test procedure.

Procedure

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

Observable Results

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

Possible Problems

This section contains a description of known issues with the test procedure, which may affect test results in certain situations.

TABLE OF CONTENTS

GROUP 1: Synchronization

Scope: The following tests cover PCS operations specific to the synchronization state diagram as defined in Clause 36 of the IEEE 802.3 standard.

Overview: The PCS synchronization process continuously monitors the code-groups conveyed through the PMA_UNITDATA.indicate primitive and determines whether or not the underlying receive channel is reliable. It passes each code-group, unaltered, to the PCS receive process and it communicates the status of the underlying receive channel to other PCS processes through the sync_status variable.

For convenience, we use the notation LOS to denote the LOSS_OF_SYNC state, CDx to denote the COMMA_DETECT_x state, AS_y to denote the ACQUIRE_SYNC_y state, SAz to represent the SYNC_ACQUIRED_z state and SAzA to represent the SYNC_ACQUIRED_zA state. Also note that alignment is used instead of rx_even in many cases because rx_even is somewhat contrary to the logical understanding of state transition.

Test #36.1.1 - Acquire Synchronization

Purpose: To verify that the device under test (DUT) acquires synchronization upon the reception of three even aligned code-groups containing a comma, and no INVALID code-groups.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.5.2.6: Synchronization, Figures 36-9: Synchronization state diagram, and Figure 36-7a: PCS receive state diagram, part a
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: The synchronization process is used to determine when the receive channel is in a state that will allow for the proper reception of /C/ ordered_sets or data. The receive channel is determined to be in an acceptable state (sync_status=OK) after it has received 3 even aligned commas and no invalid code-groups. This is illustrated in *Figure 36-9 Synchronization State Diagram* in the transitions from the LOSS_OF_SYNC state to the SYNC_ACQUIRED_1 state.

Note: In the PCS receive state diagram, part a will remain in the LINK_FAILED state as long as sync_status=FAIL. This prevents packets or /C/ ordered_sets from being received while sync_status=FAIL. Once sync_status=OK a packet must be preceded by at least one /I/ ordered_set, before the packet can be received. This /I/ ensures that the device is in the IDLE_D state when /S/ is received.

The following tables give examples of the state transitions that are made in the synchronization state machine when receiving /I/ ordered_sets or /C/ ordered_sets.

Table 36.1.1.1: Acquiring Synchronization with /I/ ordered_sets

code-group	alignment	transition to	sync_status
/D/		LOSS_OF_SYNC	FAIL
/K28.5/	Even	COMMA_DETECT_1	FAIL
/D16.2/	Odd	ACQUIRE_SYNC_1	FAIL
/K28.5/	Even	COMMA_DETECT_2	FAIL
/D16.2/	Odd	ACQUIRE_SYNC_2	FAIL
/K28.5/	Even	COMMA_DETECT_3	FAIL
/D16.2/	Odd	ACQUIRE_SYNC_1	OK

Table 36.1.1.2: Acquiring Synchronization with /C/ ordered_sets

code-group	alignment	transition to	sync_status
/D/		LOSS_OF_SYNC	FAIL
/K28.5/	Even	COMMA_DETECT_1	FAIL
/D21.5/	Odd	ACQUIRE_SYNC_1	FAIL
/Dx.y/	Even	ACQUIRE_SYNC_1	FAIL
/Dx.y/	Odd	ACQUIRE_SYNC_1	FAIL
/K28.5/	Even	COMMA_DETECT_2	FAIL
/D2.2/	Odd	ACQUIRE_SYNC_2	FAIL
/Dx.y/	Even	ACQUIRE_SYNC_2	FAIL
/Dx.y/	Odd	ACQUIRE_SYNC_2	FAIL
/K28.5/	Even	COMMA_DETECT_3	FAIL
/D21.5/	Odd	SYNC_ACQUIRED_1	OK
/Dx.y/	Even	SYNC_ACQUIRED_1	OK
/Dx.y/	Odd	SYNC_ACQUIRED_1	OK

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the DUT and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A:

1. Bring the DUT to the LOSS_OF_SYNC state.
2. Instruct the testing station to transmit 2 repetitions of test sequence 1 followed by one /I/ ordered_set, which is defined in the **/I/ after** column in the table below.

Sequence	Description	EVEN	ODD	/I/ after
1	/I1/	/K28.5/	/D5.6/	**/I1/**
2	/I2/	/K28.5/	/D16.2/	**/I2/**
3	/I1/I2/I2… (repeating /I2/)	/K28.5/	/D5.6/ or /D16.2/	**/I2/**
4	Alternate /I1/ and /I2/	/K28.5/	/D5.6/ or /D16.2/	**Continue pattern**
5	Should be treated as Idle	/K28.5/	/D0.0/	**/I/**
6	/K28.1/ contains a comma	/K28.1/	/D0.0/	**/I/**
7	/C1/ - break link	/K28.5/	/D21.5/	**/I/**
		/D0.0/	/D0.0/	
8	/C2/ - break link	/K28.5/	/D2.2/	**/I/**
		/D0.0/	**/D0.0/**	

9	This sequence should cause the DUT to stay in each ACQUIRE _SYNC state for 3 code-groups	/K28.5/ /D0.0/	/D0.0/ /D0.0/	/I/
10	This sequence should cause the DUT to stay in each ACQUIRE _SYNC state for 5 code-groups	/K28.5/ /D0.0/ /D0.0/	/D0.0/ /D0.0/ /D0.0/	/I/

3. Instruct the testing station to transmit a valid packet and a continuous stream of /I/.
4. Observed the output from the DUT.
5. Repeat steps 1 to 4 with all other test sequences.
6. Repeat step 1 through 5, increasing the number of repetitions until the DUT acquires synchronization and replies to the frame.

Observable Results:

a. The DUT should respond to or forward valid frames received after at least 3 even aligned commas and no invalid code-groups (plus one /I/ ordered_set) as described in Part A. Frames preceded by fewer than 3 even aligned commas should not be received.

Possible Problems:

- If signal_detect is set to FAIL, the DUT will fail to acquire synchronization.
- This test cannot be performed, if the DUT doesn't lose its link when it receives a long stream of invalid code-groups.
- If the DUT sets mr_main_reset to true when synchronization is lost then it will be impossible to determine whether or not the DUT required the minimum number of each test sequence to acquire synchronization.
- It is difficult to determine how the DUT will interpret the reception of the first ordered_sets transmitted by the testing station. Due to code-group slippage and other issues, the DUT may miss the first couple of code-groups transmitted by the testing station. To get around this problem we will allow the DUT to acquire synchronization and then lose synchronization. This test makes the assumption that the DUT is capable of acquiring synchronization upon the reception of a long continuous stream of valid /I/ ordered_sets. This test also assumes that the DUT will lose synchronization after receiving a long continuous stream of invalid code-groups. The invalid code-groups sent will ensure the DUT remains in the LOSS_OF_SYNC state.

Test #36.1.2 - Maintain Synchronization

Purpose: To verify that the device under test (DUT) is able to maintain synchronization for a specific set of invalid code-group sequences.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.5.2.6: Synchronization, Figure 36-9: Synchronization state diagram, and Figure 36-7a: PCS receive state diagram, part a
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: Once sync_status is set to OK, the synchronization process begins counting the number of invalid code-groups received. That count is incremented for every code-group received that is invalid or contains a comma when rx_even is TRUE. That count is decremented for every four consecutive valid code-groups received (a comma received when rx_even is FALSE is considered valid). If the count reaches four, sync_status is set to FAIL and the DUT should enter the LINK_FAILED state in Figure 36-7a state diagram. The test is designed to verify that the DUT will properly receive frames while the number of INVALID code-groups counted is less than four; or, in other words, while the DUT is in any of the SYNC_ACQUIRED states. The table below is an example of how a device should transition between these states.

Table 36.1.2.1: Maintaining and losing Synchronization

code-group	alignment	transition to	good_cgs	sync_status
	Even	SYNC_ACQUIRED_1		OK
/INVALID/	Odd	SYNC_ACQUIRED_2	0	OK
/K28.5/	Even	SYNC_ACQUIRED_2A	1	OK
/D/	Odd	No transition	2	OK
/D/	Even	No transition	3	OK
/D/	Odd	SYNC_ACQUIRED_1	4	OK
/INVALID/	Even	SYNC_ACQUIRED_2	0	OK
/INVALID/	Odd	SYNC_ACQUIRED_3	0	OK
/INVALID/	Even	SYNC_ACQUIRED_4	0	OK
/D/	Odd	SYNC_ACQUIRED_4A	1	OK
/INVALID/	Odd	LOSS_OF_SYNC	0	FAIL

Note: In the PCS receive state diagram, part a will remain in the LINK_FAILED state as long as sync_status=FAIL. This prevents packets or /C/ ordered_sets from being received while sync_status=FAIL. Once sync_status=OK a packet must be preceded by at least one /I/

ordered_set, before the packet can be received. This /I/ ensures that the device is in the IDLE_D state when /S/ is received.

Note 2: A code-group is considered /INVALID/ if it is (1) an odd aligned comma, or (2) any 10-bit pattern not found in the column corresponding to the receiver's current running-disparity. Both cases will be used during this test.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the DUT and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A:

1. Bring the DUT to the SYNC_ACQUIRED_1 state and continuously transmit /I/ ordered_sets.
2. Instruct the testing station to transmit test sequence 1 to the DUT.

Sequence	Description	EVEN	ODD
1	Enter SYNC_ACQUIRED_2	/K28.5/	**/INVALID/**
2	Enter SYNC_ACQUIRED_2-3	/INVALID/	**/INVALID/**
3	Enter SYNC_ACQUIRED_2-4	/K28.5/	**/INVALID/**
		/INVALID/	**/INVALID/**
4	Enter SYNC_ACQUIRED_2-4 and SYNC_ACQUIRED_2A-4A with good_cgs = 1	/K28.5/	**/INVALID/**
		/K28.5/	**/INVALID/**
		/K28.5/	**/INVALID/**
5	Enter SYNC_ACQUIRED_2-4 and SYNC_ACQUIRED_2A-4A with good_cgs = 2	/K28.5/	**/INVALID/**
		/K28.5/	**/D16.2/ or /D5.6/**
		/INVALID/	**/D16.2/**
		/K28.5/	**/INVALID/**
6	Enter SYNC_ACQUIRED_2-4 and SYNC_ACQUIRED_2A-4A with good_cgs = 3	/K28.1/	**/INVALID/**
		/K28.5/	**/D16.2/ or /D5.6/**
		/K28.5/	**/INVALID/**
		/K28.5/	**/D16.2/ or /D5.6/**
		/K28.5/	**/INVALID/**
7	Enter SYNC_ACQUIRED_1-4, return to SYNC_ACQUIRED_3 via SYNC_ACQUIRED_4A and re-enter SYNC_ACQUIRED_4	/INVALID/	**/INVALID/**
		/INVALID/	**/D0.0/**
		/K28.5/	**/D16.2/ or /D5.6/**
		/D0.0/	**/INVALID/**

3. Instruct the testing station to transmit one /I/ ordered_set and a valid frame to the DUT.
4. Observe the output from the DUT.
5. Repeat steps 1 through 4 with all other test sequences. The tester may also try additional sequences that contain up to three valid code-groups (cggood) between code-groups that are invalid or contain commas when rx_even is TRUE.

Observable Results:

a. The DUT should receive each frame.

Possible Problems:

- It is not possible to test all 7 sequences defined in this test with every invalid 10-bit sequence. While the DUT is observed to pass for some 10-bit sequences, it may fail for others.
- If at any time signal_detect is set to FAIL, the DUT will lose synchronization.

Test #36.1.3 - Loss of Synchronization

Purpose: To verify that the DUT will lose synchronization after the reception of 4 INVALID code-group sequences each separated by no more than 3 valid code-groups.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.5.2.6: Synchronization, Figure 36-9: Synchronization state diagram, and Figure 36-7a: PCS receive state diagram, part a
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: The DUT, if in the SYNC_ACQUIRED_1 state, must return to the LOSS_OF_SYNC state if it receives a total of four invalid code-groups and/or misaligned commas. These 4 INVALID code-groups must be separated by no more than 3 valid code-groups or else the DUT will not lose synchronization. While xmit=DATA, a DUT should lose synchronization after continually receiving any of the test sequences described in Part A of this test. If auto-negotiation is enabled, the transmission of /C/ ordered_sets when xmit=DATA is an indication that synchronization has been lost. This is only true if synchronization is lost for the duration of link_timer. If auto-negotiation is disabled, the loss of a frame preceded by one /I/ ordered_set is an indication that synchronization has been lost. The PCS receive process is incapable of receiving a packet when sync_status=FAIL, as the PCS receive process is incapable of receiving a frame while not in on of the SYNC_ACQUIRED states.

Note: In the PCS receive state diagram, part a will remain in the LINK_FAILED state as long as sync_status=FAIL. This prevents packets or /C/ ordered_sets from being received while sync_status=FAIL. Once sync_status=OK a packet must be preceded by at least one /I/ ordered_set, before the packet can be received. This /I/ ensures that the device is in the IDLE_D state when /S/ is received.

Note 2: A code-group is considered /INVALID/ if it is (1) an odd aligned comma, or (2) any data or special code-group that is not found in the column corresponding to the receiver's current running-disparity. Both cases will be used in determining whether the DUT maintained synchronization.

Test Setup: Connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station using the appropriate medium (e.g. multi-mode fiber, balanced copper).

Procedure:

Part A (Verify that the DUT can lose synchronization):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to transmit a continuous stream of invalid code-groups (such that the DUT is in the LOS state).
3. Instruct the testing station to transmit one /I/ ordered_set followed by a valid packet.
4. Instruct the testing station to transmit a long string of /I/ ordered_sets followed by a second valid packet.
5. Observe the output from the DUT.
6. If the DUT replies to the packet in step 3 modify the stream of INVALID code-groups until the DUT discards the packet.

Part B (Verify that the DUT loses synchronization from the correct number of invalids):

1. Bring the DUT to the state where xmit=DATA.
2. Instruct the testing station to transmit test sequence 1, starting each on the even code-group position.

Sequence	Description	EVEN	ODD
1	0 good_cgs between each INVALID and INVALIDS start in the odd alignment	/K28.5/	/INVALID/
		/INVALID/	/INVALID/
		/INVALID/	/D16.2/ or /D5.6/
2	0 good_cgs between each INVALID and INVALIDS start in the even alignment	/INVALID/	/INVALID/
		/INVALID/	/INVALID/
3	1 good_cgs between each INVALID	/INVALID/	/D0.0/
		/INVALID/	/D0.0/
		/INVALID/	/D0.0/
		/INVALID/	/D0.0/
4	2 good_cgs between each INVALID	/INVALID/	/D0.0/
		/K28.5/	/INVALID/
		/K28.5/	/D16.2/ or /D5.6/
		/INVALID/	/D0.0/
		/K28.5/	/INVALID/
5	3 good_cgs between each INVALID	/INVALID/	/D0.0/
		/K28.5/	/D16.2/ or /D5.6/
		/INVALID/	/D0.0/
		/K28.5/	/D16.2/ or /D5.6/
		/INVALID/	/D0.0/
		/K28.5/	/D16.2/ or /D5.6/
		/INVALID/	/D0.0/

3. Instruct the testing station to transmit one /I/ ordered_set and a valid packet to the DUT.

4. Instruct the testing station to transmit a long stream of idle and a second valid packet.
5. Observe the output from the DUT.
6. Repeat steps 1 through 5 for each test sequence defined. The tester may also try additional sequences that contain up to three valid code-groups between code-groups that are invalid or contain commas when rx_even is TRUE.

Observable Results:

a. The DUT should not accept the packet in step 3 but should accept the packet in step 4. If no sequence can be found it is unlikely that Part B can be performed.
b. The DUT should not accept the packet in step 5 but should accept the packet in step 6.

Possible Problems:

- If at any point during the test signal_detect=FAIL, the DUT will lose synchronization. If this occurs for the duration of link_timer, auto-negotiation will be restarted.
- It is possible for the DUT to have moved to the LOS state from the SA1, SA2 or SA3 state rather than from the SA4 state. This would not be evident upon the results of this test, but it may be seen in PCS Test #36.1.2.

Test #36.1.4 - Fail to Acquire Synchronization

Purpose: To verify that a station in the LOSS_OF_SYNC state will not acquire synchronization if it receives code-group sequences that should not cause it to acquire synchronization.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.5.2.6: Synchronization, Figures 36-9: Synchronization state diagram, and Figure 36-7a: PCS receive state diagram, part a
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: A PCS in the LOSS_OF_SYNC state is required to receive 3 consecutive, valid ordered_sets each beginning with an evenly aligned comma in order to acquire synchronization. If an invalid code-group or a misaligned comma is received prior to achieving synchronization, the PCS Synchronization process returns to the LOSS_OF_SYNC state. The DUT should fail to acquire synchronization when continually receiving any of the code-group sequences described in part A of this test.

Note: In the PCS receive state diagram, part a will remain in the LINK_FAILED state as long as sync_status=FAIL. This prevents packets or /C/ ordered_sets from being received while sync_status=FAIL. Once sync_status=OK a packet must be preceded by at least one /I/ ordered_set before the packet can be received. This /I/ ensures that the device is in the IDLE_D state when /S/ is received.

Note 2: A code-group is considered /INVALID/ if it is (1) an odd aligned comma, or (2) any data or special code-group that is not found in the column corresponding to the receiver's current running-disparity. Both cases will be used in determining whether the DUT maintained synchronization.

Test Setup: Connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station using the appropriate medium (e.g. multi-mode fiber, balanced copper).

Procedure:

Part A:

1. Bring the DUT to the LOSS_OF_SYNC state.
2. Instruct the testing station to transmit 100 repetitions of sequence 1.

Sequence	Description	EVEN	ODD
1	INVALID in COMMA_DETECT_1	/K28.5/	/INVALID/
2	INVALID in ACQUIRE_SYNC_1	/K28.5/	/D16.2/
		/INVALID/	
3	INVALID in COMMA_DETECT_2	/K28.5/	/D16.2/
		/K28.5/	/INVALID/
4	INVALID in ACQUIRE_SYNC_2	/K28.5/	/D16.2/
		/K28.5/	/D16.2/
		/INVALID/	
5	INVALID in COMMA_DETECT_3	/K28.5/	/D16.2/
		/K28.5/	/D16.2/
		/K28.5/	/INVALID/
6	/C1/C2/ followed by an INVALID (in COMMA_DETECT_3)	/K28.5/	/D2.2/
		/D0.0/	/D0.0/
		/K28.5/	/D21.5/
		/D0.0/	/D0.0/
		/K28.5/	/INVALID/
7	INVALID in ACQUIRE_SYNC_1 after a long stream of /D/ code-groups.	/K28.5/	/D0.0/
		/D0.0/	/D0.0/
		/D0.0/	/D0.0/
		/D0.0/	/INVALID/
8	INVALID in COMMA_DETECT_3 with 5 /D/ code-groups, between each comma.	/K28.5/	/D0.0/
		/D0.0/	/D0.0/
		/D0.0/	/D0.0/
		/K28.5/	/D0.0/
		/D0.0/	/D0.0/
		/D0.0/	/D0.0/
		/K28.5/	/INVALID/

3. Instruct the testing station to transmit one /I/ ordered_set and a valid packet to the DUT.
4. Instruct the testing station to transmit a long stream of idle and a second valid packet.
5. Observe the output from the DUT.
6. Repeat steps 1 through 5 for each test sequence defined. The tester may use different sequences for sequences 1 through 8 as long as mapped sequences will not permit the DUT to move from LOS to SA1 prior to the reception of the first packet.

Observable Results:

a. The DUT should not accept the packet in step 3 but should accept the packet in step 4.

Possible Problems: A signal_detect=FAIL will cause the DUT to constantly transmit /C/ ordered_sets with /D0.0/ contained within the last two /D/ code-groups; however, you must ensure that the DUT is receiving a signal.

Test #36.1.5 – Reception of /INVALID/ Codes

Purpose: To verify that the DUT properly detects and reacts to the reception of various invalid code-groups during the synchronization process.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.4.6, 36.2.5.1.3, 36.2.5.2.6: Synchronization, Figures 36-9: Synchronization state diagram, and Figure 36-7a: PCS receive state diagram, part a
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: The synchronization process determines whether the underlying receive channel is ready for operation. In order for the DUT to acquire synchronization, it must receive 3 evenly aligned commas (see *Test #36.1.1 - Acquire Synchronization*). Once synchronization has been acquired the DUT will lose synchronization if the receive channel becomes unusable. This occurs if the DUT receives four bad code-groups within a predetermined period of time (see *Test #36.1.3 - Loss of Synchronization*).

The DUT should consider the following as bad code-groups (defined as *cgbad* in Clause 36.2.5.1.3): (1) odd aligned commas, or (2) 10-bit pattern not found in the column corresponding to the receiver's current running-disparity.

Test Setup: Connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station using the appropriate medium (e.g. multi-mode fiber, balanced copper).

Procedure:

Part A (Loss of Synchronization from INVALID code-groups):

1. If possible, disable auto-negotiation on the DUT.
2. Bring the DUT to the SYNC_ACQUIRED_1 state and instruct the testing station to transmit a continuous stream of idle (/I2/ shall be used to ensure a negative running disparity at the start of each test sequence).
3. Instruct the testing station to transmit 2 repetitions of test sequence 1.

Sequence	Description	EVEN	ODD
1	Undefined sequence (1111111111)	1111111111	1111111111
2	Undefined sequence (0000000000)	0000000000	0000000000
3	Misaligned comma (K28.5)	/D0.0/	/K28.5/
		/D0.0/	/K28.5/

Sequence	Description	EVEN	ODD
4	Misaligned comma (K28.1)	/D0.0/	/K28.1/
		/D0.0/	/K28.1/
5	Running disparity error – DATA	/+10.2/	/-10.2/
6	Running disparity error – COMMA	/+K28.5/	/-D10.2/
		/+K28.5/	/-D10.2/
7	Running disparity error – Special code-group	/+V/	/-V/
8	Running disparity error – Special code-group	/+T/	/-T/

4. Instruct the testing station to transmit one /I/ ordered_set and a valid packet to the DUT.
5. Instruct the testing station to transmit a long stream of idle and a second valid packet.
6. Observed the output from the DUT.
7. Repeat steps 1 through 6 for each test sequence defined.

Observable Results:

a. The DUT should lose synchronization upon the reception of each test sequence. The DUT should not accept the packet in step 3 but should accept the packet in step 4.

Possible Problems:

- The DUT may require more than 4 invalid code-groups to lose synchronization. If this is the case, change the number of times each sequence is transmits from 4 to the number determined to cause the DUT to lose synchronization as found in *Test #36.1.3 - Loss of Synchronization*.

GROUP 2: Transmission

Scope: The following tests cover PCS operations specific to generation and transmission of 10-bit code-groups.

Overview: These tests are designed to verify that the device under test properly generates and transmits 10-bit code-groups and ordered_sets.

Test #36.2.1 - 8B/10B Encoding

Purpose: To verify that the device under test (DUT) selects the proper encoding for transmitted code-groups.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclauses 36.2.4.4: Running disparity rules, 36.2.4.5: Generating code-groups, Table 36-1: Valid data code-groups, and Table 36-2: Valid special code-groups
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: The PCS transmit process updates its running disparity value after each code-group is sent. The current value of the running disparity is used to select the proper encoding of each transmitted code-group. In order to adequately test the 8B/10B encoding process, it is necessary to have the device under test transmit all valid data code-groups, /K28.5/, /S/, /T/, /R/, and /V/ for both the positive and negative running disparity. *Both forms of each valid data code-group can be generated as part of normal packet data.*

Since properly encoded /T/ and /R/ ordered_sets do not change the value of the running disparity, the forms of /T/, /R/, and /K28.5/ are defined by the running disparity value after the last byte of packet data. See examples 1 and 2 below.

Since /S/ must be preceded by /I/, and /I/ ensures that the running disparity assumes its negative value, only the negative running disparity encoding of /S/ will normally be observed. The only exception to this is when /S/ appears in the second or later packet in a packet burst. In this case, /S/ follows /R/ and the form of /S/ is defined by the last byte of data in the preceding packet. See example 2 below.

/V/ ordered_sets may be observed as the jam pattern a device under test (DUT) sends to enforce a collision within carrier extension. In such cases, the form of /V/ depends on the value of the running disparity following the transmission of the last data byte. See examples 4 and 5 below.

Example	Description	EVEN	ODD
1	EPD for a frame ending with a positive running disparity (/EPD/I1/I2/…)	/+T/	/+R/
		/+K28.5/	/-D5.6/
		/-K28.5/	/-D16.2/
2	EPD for a frame ending with a negative running disparity (/EPD/I2/…)	/-T/	/-R/
		/-K28.5/	/+D16.2/
		/-K28.5/	/+D16.2/

Example	Description	EVEN	ODD
3	SPD (/S/) with a positive running disparity as seen as the second or later packet in a burst. (/EPD/R/…/R/S/)	/+T/	/+R/
		/+R/	/+R/
		…	…
		/+S/	/+D/
4	Error_Propagation (/V/) with a positive running disparity as seen when a collision occurs during the transmission of Carrier_Extend (/EPD/R/V/)	/+T/	/+R/
		/+R/	/+V/
		/+V/	/+V/
5	Error_Propagation (/V/) with a negative running disparity as seen when a collision occurs during the transmission of Carrier_Extend (/EPD/R/V/)	/-T/	/-R/
		/-R/	/-V/
		/-V/	/-V/

Note: Test 36.2.2 and test 36.2.3 verify that the DUT transmits the proper IDLE and proper number of /R/ ordered_sets in the EPD, respectively.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the DUT and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A (Verify all data code-groups):

1. Bring the DUT to the state where xmit=DATA. If the DUT is not capable of manual configuration, instruct the testing station to auto-negotiate with the DUT.
2. Once xmit=DATA, force the DUT to transmit a packet containing both forms of every valid data code-group listed in Table 36-1.
3. Observe the output from the DUT.

Part B (Verify /T/, /R/ and /-S/):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, force the DUT to transmit two packets. The first packet should end with negative running disparity while the second packet should end with positive running disparity.
3. Observe the output from the DUT.

Part C (Verify /+S/, only applicable in Half Duplex mode):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, force the DUT to transmit a burst of two or more packets. Ensure that the running disparity after the last byte of data in the first packet is positive.
3. Observe the output from the DUT.

Part D (verify /V/, only applicable in Half Duplex mode):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, force the DUT to issue a packet that requires extension. Ensure that the running disparity after the last byte of data is positive.
3. Instruct the testing station to collide with the packet while extension is being sent.

4. Repeat step 2 and 3, ensuring that the running disparity after the last byte of data issued by the DUT is negative.
5. Observe the output from the DUT.

Observable Results:

a. At all times, the DUT should transmit the correct encoding of the appropriate data code-group.
b. The DUT should properly transmit a /+T/ and at least one /+R/ at the end of the packet ending with a positive running disparity, a /-T/ and at least one /-R/ at the end of the packet ending with a negative running disparity, and both packets should start with a /-S/.
c. The DUT should start the second frame in the burst with a /+S/.
d. The DUT should transmit /+V/ after /+R/ and /-V/ after /-R/ when a collision occurs during the transmission of extension.

Possible Problems: Most devices will not support half duplex mode. In this case, test parts C and D cannot be performed.

Test #36.2.2 - /I/ Generation

Purpose: To verify that the first /I/ ordered_set following the EPD or /C/ ordered_set ensures that the running disparity is negative.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.4.12: IDLE (/I/), Figure 36-5: PCS transmit ordered_set state diagram, and Figure 36-6: PCS transmit code-group state diagram
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: /I/ ordered_sets are transmitted during normal inter-packet gap. /I/ may be transmitted as /I1/ or /I2/. The /I1/ ordered_set is defined so that the running disparity following the ordered_set is opposite the running disparity at the beginning of the ordered_set. The /I2/ ordered_set is defined so that the running disparity following the ordered_set is identical to the running disparity at the beginning of the ordered_set.

/I/ ensures that the running disparity assumes its negative value. Thus, if the running disparity at the beginning of the inter-packet gap is positive, /I/ will consist of /I1/ followed by /I2/ for the duration of /I/. If the running disparity is negative, /I/ will consist solely of /I2/.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the DUT and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A (Ensure the DUT transmits one /I1/ then /I2/s after an EPD with a positive running disparity):

1. Bring the DUT to the state where xmit=DATA. If the DUT is not capable of manual configuration, instruct the testing station to auto-negotiate with the DUT.
2. Once xmit=DATA, force the DUT to transmit a packet such that the running disparity following the last byte of data is positive.
3. Observe the output from the DUT.

Part B (Ensure the DUT transmit /I2/s after an EPD with a negative running disparity):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, force the DUT to transmit a packet such that the running disparity following the last byte of data is negative.
3. Observe the output from the DUT.

Observable Results:

a. If the running disparity at the start of transmission of /I/ is positive, /I/ shall consist of /I1/ followed by repeated /I2/ ordered_sets.
b. If the running disparity at the start of transmission of /I/ is negative, /I/ shall consist solely of repeated /I2/ ordered_sets.

Possible Problems: None

Test #36.2.3 - /I/ Alignment

Purpose: To verify that the device under test (DUT) transmits the correct number of /R/ code-groups so that /I/ begins in an even code-group position.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.4.15.1: Carrier_Extend rules, Figure 36-5: PCS transmit ordered_set state diagram, and Figure 36-6: PCS transmit code-group state diagram
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: /I/ ordered_sets are transmitted during normal inter-packet gap. Since /I/ ordered_sets begin with /K28.5/ and /K28.5/ must fall in an even code-group position, /I/ always begins in an even code-group position.

If /T/ is transmitted in an even code-group position, there must be an odd number of /R/ code-groups for /I/ to begin in an even code-group position (see example 1). If /T/ is transmitted in an odd code-group position, there must be an even number of /R/ code-groups for /I/ to begin in an even code-group position (see example 2). This behavior should still be observed after the last packet in a frame burst.

Example	Description	EVEN	ODD
1	EPD with /T/ in the even position	/T/	/R/
		/K28.5/	/D5.6/ or /D16.2/
2	EPD with /T/ in the odd position	/D/	/T/
		/R/	/R/
		/K28.5/	/D5.6/ or /D16.2/

If a packet is transmitted with carrier extension, the DUT should still ensure that idle begins in even code-group position by transmitting two or three /R/ ordered_sets after TX_ER is set to 0.

Test Setup: Connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station using the appropriate medium (e.g. multi-mode fiber, balanced copper).

Procedure:

Part A (/T/ in the even position):

1. Bring the DUT to the state where xmit=DATA. If the DUT is not capable of manual configuration, instruct the testing station to auto-negotiate with the DUT.
2. Once xmit=DATA, force the DUT to transmit a packet which does not require extension and requires /T/ to be transmitted in an even code-group position.
3. Observe the output from the DUT.

Part B (/T/ in the odd position):

4. Repeat part A, forcing the DUT to transmit a packet which does not require extension and requires /T/ to be transmitted starting in an odd code-group position in step 2.

Part C (/T/ in the even position with extension, only applicable in Half Duplex mode):

5. Repeat part A, forcing the DUT to transmit a packet that requires extension and requires /T/ to be transmitted in an even code-group position in step 2.

Part D (/T/ in the odd position with extension, only applicable in Half Duplex mode):

6. Repeat part A, forcing the DUT to transmit a packet that requires extension and requires /T/ to be transmitted in an odd code-group position in step 2.

Part E (/T/ in the even position for last frame in a burst, only applicable in Half Duplex mode)

7. Repeat part A, forcing the DUT to transmit a burst of two or more packets where the last packet requires /T/ to be transmitted in an even code-group position in step 2.

Part F (/T/ in the odd position for last frame in a burst, only applicable in Half Duplex mode):

8. Repeat part A, forcing the DUT to transmit a burst of two or more packets where the last packet requires /T/ to be transmitted in an odd code-group position in step 2.

Observable Results:

a. The DUT shall ensure that /I/ begins in an even code-group position after the packet.
b. The DUT shall ensure that /I/ begins in an even code-group position after the packet.
c. The DUT shall ensure that /I/ begins in an even code-group position after the packet with extension.
d. The DUT shall ensure that /I/ begins in an even code-group position after the packet with extension.
e. The DUT shall ensure that /I/ begins in an even code-group position after the burst.
f. The DUT shall ensure that /I/ begins in an even code-group position after the burst.

Possible Problems: Most devices will not support half duplex mode. In this case test parts C through F cannot be performed.

Test #36.2.4 - /C/ Transmission Order

Purpose: To verify that the device under test (DUT) transmits /C/ ordered_sets as alternating /C1/ and /C2/ ordered_sets.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.4.10: Configuration (/C/), Figure 36-5: PCS transmit ordered_set state diagram, and Figure 36-6: PCS transmit code-group state diagram
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: /C/ ordered_sets are used to convey 16-bit configuration registers to the link partner. Figure 36-5, the PCS transmit ordered_set state diagram, shows that while xmit is set to CONFIGURATION, tx_o_set will be set to /C/. Figure 36-6, the PCS transmit code-group state diagram, shows that while tx_o_set is set to /C/, ordered_sets /C1/ and /C2/ are transmitted one after the other.

Note that the /C1/ ordered_set is defined so that the running disparity following the transmission of the first two code-groups is opposite the running disparity at the beginning of the ordered_set. Also note that the /C2/ ordered_set is defined so that the running disparity following the transmission of the first two code-group is identical to the running disparity at the beginning of the ordered_set.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A:

1. Disconnect the receiver of the DUT from the transmitter of the testing station. Force the DUT to restart auto-negotiation.
2. Observe the output from the DUT.

Part B:

3. Disconnect the receiver of the DUT from the transmitter of the testing station.
4. Reconnect the receiver of the DUT to the transmitter of the testing station. Force the testing station to transmit /I/ ordered_sets.
5. Observe the output from the DUT.

Observable Results:

a. When the DUT isn't receiving any signal from the testing station it is expected to constantly send break link. The DUT shall transmit /C/ ordered_sets as alternating /C1/ and /C2/ ordered_sets. The testing station should observe this for all of the /C/ ordered_sets sent while the DUT is sending break link.
b. When the DUT is receiving idle from the testing station it is expected to initially transmit break link for link timer and then proceed to send /C/ ordered_sets containing its abilities. Because it never receives break link from the testing station it is expected to constantly send /C/ ordered_sets with its abilities. The DUT shall transmit /C/ ordered_sets as alternating /C1/ and /C2/ ordered_sets. The testing station should observe this for all of the /C/ ordered_sets sent while the DUT is receiving /I/ ordered_sets from the testing station. The testing station should also observe the DUT maintain the alternating transmission of /C1/ and /C2/ ordered_sets when transitioning from break link to abilities.

Possible Problems: If the DUT does not support Auto-Negotiation this test cannot be performed.

GROUP 3: Reception

Scope: The following tests cover PCS operations specific to the reception of 10-bit code-groups.

Overview: These tests are designed to verify that the device under test properly receives 10-bit code-groups and ordered_sets.

Test #36.3.1 - 8B/10B Decoding

Purpose: To verify that the device under test (DUT) properly decodes /D/ code-groups using the DECODE function, and properly handles valid special code-groups.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclauses 35.2.1.5: Response to error indications from GMII, 36.2.4.3: Valid and invalid code-groups, 36.2.4.4: Running disparity rules, 36.2.4.6: Checking the validity of received code-groups, 36.2.5.2.4 Code-group stream decoding, Table 36-1: Valid data code-groups, Table 36-2: Valid special code-groups, and Table 36-3: Defined ordered_sets

[2] ANNEX A - Table of Acronym Definitions and Abbreviations

[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: At all times the PCS should be checking the validity of all received code-groups. A code-group is considered invalid if it is (1) an odd aligned comma, or (2) a 10-bit pattern not found in the column corresponding to the receiver's current running-disparity.

This test is designed to verify that the DUT and decode valid received code-groups. This test has two test cases. The first verifies that the DUT can decode all valid data code-groups when received in a frame. The second verifies that the DUT can receive a variety of different valid code-groups during the synchronization process (i.e. verifying that valid code-groups are not considered cgbad).

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A (valid data code-groups during a frame):

1. Bring the DUT to the state where xmit=DATA. If the DUT is not capable of manual configuration, instruct the testing station to auto-negotiate with the DUT.
2. Once xmit=DATA, instruct the testing station to transmit an ICMP echo request containing both forms of every valid data code-group listed in Table 36-1.
3. Observe any activity from the DUT.

Part B (valid special code-groups during idle):

1. Bring the DUT to the SYNC_ACQUIRED_1 state and continuously transmit /I/ ordered_sets so that the running disparity at the beginning of each test sequence is negative.
2. Instruct the testing station to transmit 100 repetitions of test sequence 1 to the DUT.

Sequence	Description	EVEN	ODD
1	Both running disparities of /K28.0/	/K28.0/	/D16.2/
2	Both running disparities of /K28.1/ (comma)	/K28.1/	/D5.6/
3	Both running disparities of /K28.2/	/K28.2/	/D5.6/
4	Both running disparities of /K28.3/	/K28.3/	/D5.6/
5	Both running disparities of /K28.4/	/K28.4/	/D16.2/
6	Both running disparities of /K28.5/	/K28.6/	/D5.6/
7	Both running disparities of /K28.7/ (comma)	/K28.7/	/D5.6/
8	Both running disparities of /K23.7/	/K23.7/	/D16.2/
9	Both running disparities of /K27.7/	/K27.7/	/D16.2/
10	Both running disparities of /K29.7/	/K29.7/	/D16.2/
11	Both running disparities of /K30.7/	/K30.7/	/D16.2/
12	Both running disparities of an arbitrary data code-group*	/D/	/D16.2/

3. Instruct the testing station to transmit one /I/ ordered_set and a valid packet to the DUT.
4. Observe the output from the DUT.
5. Repeat steps 1 through 4 for each test sequence defined.

* Some code-groups have the same 10 bit value for each running disparity (see /D10.2/ for an example). These code-groups should not be used for this test case.

Observable Results:

a. The DUT should reply to or forward the ICMP echo request. If the reply contains both forms of every valid data code-group it is assumed that the DUT properly decoded each code-group.
b. The DUT should reply to or forward all ICMP echo requests.

Possible Problems: None.

Test #36.3.2 - Carrier Event Handling

Purpose: To verify that the device under test (DUT) detects carrier events and handles them properly.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 36.2.4.16: Error_Propagation (/V/), 36.2.5.2.3: State variable function carrier_detect(x), Figure 36-5: PCS transmit ordered_set state diagram, and Figure 36-7: PCS receive state diagram, parts a and b
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements
[4] ANNEX C – Carrier Event Handling Sequences

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: The variable carrier_detect is set to TRUE upon the reception of a code-group that is two or more bits different from both encodings of /K28.5/ and is in an even code-group position. If this code-group is 10 bits different from expected /K28.5/ (based on the current running disparity) then the carrier_detect should be set to FALSE. This will only occur if the DUT receives a /K28.5/ with a running disparity error. The PCS receive process checks carrier_detect upon exit from the IDLE_D state. If it is TRUE, the process moves into the CARRIER_DETECT state. If it is FALSE, the process goes back to the RX_K state.

Upon entry to the CARRIER_DETECT state, the variable receiving is set to TRUE and the code-group that set carrier_detect to TRUE is examined. If it is not /S/, the process transitions to the FALSE_CARRIER state and remains there until /K28.5/ is received in an even code-group position. This will take the process to the RX_K state where receiving is set to FALSE.

This test will determine whether or not the DUT detects and properly handles carrier events by prepending a normal packet with a special two code-group sequence in place of a normal /I2/. Four test cases are verified: (1) the /K28.5/ of idle is replaced with each 10 bit sequence 2-bit different from both /K28.5/ encodings, (2) the /K28.5 of idle is replaced with each 10 bit sequence 1-bits different from both /K28.5/ encodings, (3) /K28.5/ is transmitted with /K28.5/ of the opposite running disparity, and (4) the /D/ code-group of idle is replaced with code-groups that are not /D16.2/, /D5.6/, /D2.2/ or /D21.5/. The 10bit sequences for Parts A to D can be found in *ANNEX C – Carrier Event Handling Sequences*.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station

Procedure:

Part A (2-bits different from /-K28.5/):

1. Bring the DUT to the state where xmit=DATA. If the DUT is not capable of manual configuration, instruct the testing station to auto-negotiate with the DUT.
2. Once xmit=DATA, instruct the testing station to transmit a long stream of /I2/ with negative running disparity.
3. Instruct the testing station to transmit a packet prepended with a two code-group sequence. The first code-group of the sequence replaces /-K28.5/ with 0b011111001, which is two bits different from /-K28.5/. The second code-group should be /D16.2/.
4. Instruct the testing station to transmit the minimum allowed interPacketGap (6 /I/ ordered_sets) and a second packet.
5. Observe the output from the DUT.
6. Repeat steps 1 through 5 for all sequences that are 2-bit different from /-K28.5/. There are a total of 45 test sequences.

Part B (2-bits different from /+K28.5/)

1. Repeat part A modifying step #3 as follows. Instruct the testing station to transmit a packet prepended with a two code-group sequence. The first code-group of the sequence replaces /-K28.5/ with a sequence that is two bits different from /+K28.5/. The second code-group should be /D16.2/.
2. Repeat for all sequences that are 2-bit different from /+K28.5/. There are a total of 45 test sequences.

Part C (1-bit different from /-K28.5/):

1. Repeat part A modifying step #3 as follows. Instruct the testing station to transmit a packet prepended with a two code-group sequence. The first code-group of the sequence replaces /-K28.5/ with a sequence that is one bit different from /-K28.5/. The second code-group should be /D16.2/.
2. Repeat for all sequences that are 1-bit different from /-K28.5/. There are a total of 10 test sequences.

Part D (1-bit different from /+K28.5/):

1. Repeat part A modifying step #3 as follows. Instruct the testing station to transmit a packet prepended with a two code-group sequence. The first code-group of the sequence replaces /-K28.5/ with a sequence that is one bit different from /+K28.5/. The second code-group should be /D16.2/.
2. Repeat for all sequences that are 1-bit different from /+K28.5/. There are a total of 10 test sequences.

Part E (10-bits different from /-K28.5/):

1. Repeat part A modifying step #3 as follows. Instruct the testing station to transmit a packet prepended with a two code-group sequence. The first code-group of the sequence replaces /-K28.5/ with /+K28.5/, which is 10 bits different. The second code-group should be /D16.2/.

Part F (Not /D16.2/, /D5.6/, /D2.2/ or /D21.5/):

1. Repeat part A modifying step #3 as follows. Instruct the testing station to transmit a packet prepended with a two code-group sequence. The first code-group should be /K28.5/. The second code-group should be sequence 1.

Sequence	code-group
1	/D5.6/
2	/D6.6/
3	/D10.1/
4	/D3.3/
5	/D27.7/
6	/D3.0/
7	/D30.2/
8	/D12.4/

Sequence	code-group
9	/D8.6/
10	/D13.7/
11	/S/
12	/T/
13	/R/
14	/V/
15	Other /D/

2. Repeat for each sequence listed above.

Observable Results:

a. For each sequence, the DUT should discard the first packet and accept the second packet.
b. For each sequence, the DUT should discard the first packet and accept the second packet.
c. For each sequence, the DUT should accept both packets.
d. For each sequence, the DUT should accept both packets.
e. For each sequence, the DUT should accept both packets.
f. For each sequence, the DUT should accept both packets.

Possible Problems: If the test packet is not discarded, it is impossible to determine whether carrier_status was not set to true or whether the PCS receive process failed to stay in the FALSE_CARRIER state.

Test #36.3.3 - Detecting End of Packet

Purpose: To verify that the device under test (DUT) can distinguish valid EPDs from invalid EPDs and detect the premature end of a packet.

References:

[1] IEEE Std. 802.3, 2005 Edition - Subclause 35.2.1.5: Response to error indications from GMII, 36.2.4.14.1: EPD rules, and Figure 36-7b: PCS receive state diagram, part b
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements:

A testing station capable of sending (receiving) 10-bit code-groups using the signaling method specified in clause 38 or 39.

Last Modification: May 30, 2007

Discussion: The End_of_Packet delimiter (EPD) is used to delineate the ending boundary of a packet. The EPD can assume one of two forms and the form used depends on whether /T/ was transmitted in an even or odd code-group position. See table below.

Example	Description	EVEN	ODD
1	Valid EPD with /T/ in the even position.	/T/	/R/
		/K28.5/	/D5.6/ or /D16.2/
2	Valid EPD with /T/ in the odd position.	/D/	/T/
		/R/	/R/
		/K28.5/	/D5.6/ or /D16.2/

The PCS receive process searches for the EPD using the check_end function. The check_end function returns the most recently received code-group and the two code-groups that will follow it. If check_end returns /T/R/R/ or /T/R/K28.5/ (with /K28.5/ being in an even code-group position), the PCS receive process recognizes that the EPD is about to be received and terminates the packet without error. Invariably, if the most recent code-group received is not valid data and the check_end function does not verify that a valid EPD is about to be received, the PCS receive process will guarantee that the MAC receives the packet with an error. It accomplishes this by setting the GMII signal RX_ER to TRUE which, when RX_DV is TRUE, will cause the Reconciliation Sublayer to force the MAC to return frameCheckError for the given packet.

A special case of this occurs when /K28.5/ is received before the EPD. /K28.5/ is used exclusively in /I/ and /C/ ordered sets and its reception indicates that the packet has reached an early end. If check_end returns /K28.5/D/K28.5/ (/K28.5/ falling in an even code-group position), the process assumes /I/ has been received. If check_end returns /K28.5/ followed by /D21.5/ or /D2.2/ and another /D/ code-group (as always, /K28..5/ falls in an even code-group position), the process

assumes that a /C/ ordered_set was received. In either case, the process sets RX_ER to TRUE and two code-groups later finds its way to the IDLE_D state. Note that a single /C/ ordered_set received in the middle of a packet is not sufficient to restart auto-negotiation.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A (Bad EPD where /T/ should be in even alignment):

1. Bring the DUT to the state where xmit=DATA. If the DUT is not capable of manual configuration, instruct the testing station to auto-negotiate with the DUT.
2. Once xmit=DATA, instruct the testing station to transmit a packet appended with sequence 1 instead of a valid EPD.

Sequence	Description	EVEN	ODD
1	No /R/ (RX_DATA_ERROR)	/T/	!/R/
		/K28.5/	/D/
2	No /I/ after /T/R/ (RX_DATA_ERROR)	/T/	/R/
		!/K28.5/ or !/R/**	/D/
3	No /T/ (EARLY_END_EXT)	/R/	/R/
		/R/	/D/
4	/I/ ordered_sets (EARLY_END)	/K28.5/	/D16.2/ or /D5.6/
		/K28.5/	/D16.2/ or /D5.6/
5	/C1/ ordered_set (EARLY_END)	/K28.5/	/D21.5/
		/D0.0/	/D0.0/
6	/C2/ ordered_set (EARLY_END)	/K28.5/	/D2.2/
		/D0.0/	/D0.0/

** An /R/ order_set should not be used to replace the /K28.5/ because this will force the DUT to enter the EXTEND_ERR state and thus receive the frame with 1 byte of extension and 3 bytes of extension error. Such behavior is outside the scope of this test suite.

3. Instruct the testing station to transmit minimum inter packet worth of /I/ and a second packet.
4. Observe the output from the DUT.
5. Repeat steps 1 through 4 for all sequences list above.

Part B (Bad EPD where /T/ should be in odd alignment):

1. Repeat part A. Using the sequences defined in the table below.

Sequence	Description	EVEN	ODD
1			/T/

Sequence	Description	EVEN	ODD
	Not enough /R/ (RX_DATA_ERROR)	/R/	/K28.5/
2	/D/ in place of first /R/ (RX_DATA_ERROR)		/T/
		/D/	/R/
3	/D/ in place of second /R/ (RX_DATA_ERROR)		/T/
		/R/	/D/
4	No /T/ (EARLY_END_EXT)		/R/
		/R/	/R/

Observable Results:

a. The DUT should discard the first packet and report frameCheckError (Informatively). The second packet should be accepted.
b. The DUT should discard the first packet and report frameCheckError (Informatively). The second packet should be accepted.

Possible Problems: In the cases where there are multiple encodings for an EPD element (e.g. !/R/), the DUT may pass the test for certain encodings but fail for others.

Test #36.3.4 - Reception of /C/ during IDLE

Purpose: To verify the device under test (DUT) restarts the auto-negotiation process after receiving a /C/ ordered_set during the reception of /I/ ordered_sets (while xmit=DATA).

References:

[1] IEEE Std. 802.3, 2005 Edition - Figure 36-7a: PCS receive state diagram, part a
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements: A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: May 30, 2007

Discussion: According to figure 36-7a, when xmit=DATA and the PCS receive process is receiving /I/ ordered_sets as it transitions between the RX_K and IDLE_D states. If /K28.5/ (or any code-group within one bit of /K28.5/) is received while in the IDLE_D state, the process moves into the RX_K state. When in the RX_K state, if /D2.2/ or /D21.5/ are received, the process assumes that it has received a /C/ ordered_set and transitions to the RX_CB state. If it receives a /D/ code-group other than /D2.2/ or /D21.5/, then it assumes it has received an /I/ ordered_set and transitions back to IDLE_D. Otherwise it moves into the RX_INVALID state and waits for the reception of /K28.5/ in an even code-group position.

RD-	code-group	RD+	code-group
001111 1010	/K28.5/	110000 0101	/K28.5/
001111 1011	/INVALID/	110000 0100	/INVALID/
001111 1000	/K28.7/	110000 0111	/K28.7/
001111 1110	/INVALID/	110000 0001	/INVALID/
001111 0010	/K28.4/	110000 1101	/K28.4/
001110 1010	/D14.5/	110001 0101	/D3.2/
001101 1010	/D12.5/	110010 0101	/D19.2/
001011 1010	/D20.5/	110100 0101	/D11.2/
000111 1010	/D7.5/	111000 0101	/D7.2/
011111 1010	/INVALID/	100000 0101	/INVALID/
101111 1010	/INVALID/	010000 0101	/INVALID/

The code-groups in the table above should ensure that carrier_detect remains FALSE. If any of these code-groups are received in the IDLE_D state and are followed by either /D2.2/ or /D21.5/, the PCS receive process will move to the RX_CB state. Only the first /C/ ordered_set may be received without starting with /K28.5/. The transitions required in going back to the RX_K state after receiving a /C/ ordered_set require that /K28.5/ be received in an even code-group position.

For a properly implemented device, auto-negotiation will be restarted when xmit=DATA, after the reception of three consecutive /C/ ordered_sets with consistent abilities.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station.

Procedure:

Part A (/C1/ only):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to continuously transmit /I/ ordered_sets in the form of /K28.5/D16.2/.
3. Instruct the testing station to send a /C/ ordered_set in the form of /K28.5/D21.5/D0.0/D0.0/ three times.
4. Instruct the testing station to continuously transmit /I/ ordered_sets.
5. Observe the output from the DUT.

Part B (/C2/ only):

1. Repeat part A with /D21.5/ being replaced by /D2.2/.

Part C (all /D/):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to continuously transmit /I/ ordered_sets.
3. Instruct the testing station to send a /C1/ ordered_set with /D0.1/ and /D0.2/ in the configuration register. Send two more /C/ ordered_sets, alternating between /C1/ and /C2/.
4. Instruct the testing station to continuously transmit /I/ ordered_sets.
5. Observe the output from the DUT.
6. Repeat steps 1 through 5 for all /D/ code-groups.

Part D (carrier_detect in first /C/ ordered_set):

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to continutously transmit /I/ ordered_sets.
3. Instruct the testing station to send three /C1/ ordered_sets with the /K28.5/ replaced with a test sequence 1-bit different.
4. Instruct the testing station to continuously transmit /I/ ordered_sets.
5. Observe the output from the DUT.
6. Repeat steps 1 through 5 for all 10 bit sequences 1 bit different from /-K28.5/ and /+K28.5/. See *ANNEX C – Carrier Event Handling Sequences* for all 10 bit sequences 1 bit different from /-K28.5/.

Observable Results:

a-d. After the DUT has received the test sequence it shall transmit /C/ ordered_sets with Config_Reg set to zero.

Possible Problems: If the DUT does not restart auto-negotiation, it is impossible to determine whether or not the failure is the result of an improper implementation of the Receive State Diagram or the Auto-Negotiation State Diagram.

Test #36.3.5 – Reception of a code with wrong running disparity within a frame

Purpose: To verify that the device under test (DUT) detects codes with running disparity errors in a frame.

References:

[1] IEEE Std. 802.3, 2000 Edition - Subclause 36.2.4.6: Checking the validity of received code-groups, and Figure 36-7: PCS receive state diagram, parts a and b
[2] ANNEX A - Table of Acronym Definitions and Abbreviations
[3] ANNEX B - Testing Requirements

Resource Requirements:

A testing station capable of transmitting and receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: March 26, 2004

Discussion: Codes received with the incorrect running disparity should be considered /INVALID/. The PCS receive process checks carrier_detect upon exit from the IDLE_D state. If it is TRUE, the process moves into the CARRIER_DETECT state. If it is FALSE, the process goes back to the RX_K state.

Upon entry to the CARRIER_DETECT state, the variable receiving is set to TRUE and the code-group that set carrier_detect to TRUE is examined. If it is not /S/, the process transitions to the FALSE_CARRIER state and remains there until /K28.5/ is received in an even code-group position. This will take the process to the RX_K state where receiving is set to FALSE. RX_K transitions to IDLE_D when a /D/ code-group is received unless the received code-group is /D21.5/ or /D2.2/. This special case is examined in test 36.3.4. If a non-/D/ code-group is received within the RX_K state, the receive process transitions to the RX_INVALID state. The RX_INVALID state is generally used to inform the auto-negotiation process that an erroneous /C/ or /I/ ordered_set has been received. This is true when xmit=DATA. When xmit=DATA, the receive process waits for the reception of a code-group in an even code-group position.

Note that when receiving is set to TRUE, the PCS carrier sense process causes the GMII signal CRS to be asserted. The Reconciliation Sublayer maps CRS to the signal the MAC uses to determine if the underlying medium is busy. If the MAC is operating in half-duplex mode and if it is given a packet to send while CRS is asserted, it will defer the transmission of the packet until CRS is de-asserted and the minimum inter-packet gap has passed.

This provides us one means of verifying that the DUT is receiving false carriers. The testing station will send a valid echo request packet followed by a false carrier. If the DUT detects the false carrier properly, it will defer the transmission of the reply until the false carrier is completed and the minimum inter-packet gap has passed. Obviously, the false carrier would have to be made sufficiently long enough to account for the time it will take the DUT to generate a reply.

This test will determine whether or not the DUT detects and properly handles carrier events by prepending a normal packet with a special two code-group sequence. For example, the code-group sequence /D16.9/D/ (given that distance between /K28.5/ and /D16.9/ is greater than two and /D16.9/ falls in an even code-group position) will put the PCS receive process in the FALSE_CARRIER state and hold it there until /I/ ordered_sets are received. If this sequence prepends a packet, that packet will be lost as part of the false carrier whether it begins with /S/ or not.

Test Setup: Using the appropriate medium (e.g. multi-mode fiber, balanced copper), connect the transmitter of the testing station to the receiver of the device under test (DUT) and the transmitter of the DUT to the receiver of the testing station

Procedure:

1. Bring the DUT to the state where xmit=DATA. If the DUT is not capable of manual configuration, instruct the testing station to auto-negotiate with the DUT. If the testing station does not implement auto-negotiation, it may emulate the process using the procedure given in ANNEX B.
2. Once xmit=DATA, instruct the testing station to transmit an ARP request followed by a two packet sequence where each packet is separated by the minimum inter-packet gap. The first packet shall be a valid echo request packet prepended with a two code-group sequence. The first code-group of the two code-group sequence replaces /K28.5/ with a code-group that has a two-bit difference from /K28.5/. The second code-group may be anything other than /D21.5/ or /D2.2/. Repeat until /K28.5/ of the negative running disparity has been replaced by every code-group that has a two-bit difference from it.
3. Repeat step #2 by replacing /K28.5/ of the negative running disparity with every code-group that has a one-bit difference from it.

Observable Results:

1. The DUT should reply to the ARP and second echo request packet in steps 2. The DUT should not reply to the first echo request packet for each of the sequences sent in step 2.
2. The DUT should reply to all three packets sent in step 3.
3. The DUT should reply to all three packets sent in step 4

Possible Problems: None.

ANNEX A - Table of Acronym Definitions and Abbreviations

Table A - 1: Acronym Definitions and Abbreviations

ACK	Acknowledge usually in reference to the ACK Bit
AS_y	ACQUIRE_SYNC_y; where y is the state
/C/	Configuration ordered_set
/C1/	Configuration 1 ordered_set /K28.5/D21.5/Config_Reg
/C2/	Configuration 2 ordered_set /K28.5/D2.2/Config_Reg
CDx	COMMA_DETECT_x; where x is the state
/D/	Data code-group
DUT	device under test
EPD	End of Packet Deliminator
GMII	Gigabit Media Independent Interface; defined in clause 35 of the IEEE 802.3 standardv
/I/	IDLE ordered_set, this is either /I1/ or /I2/
/I1/	IDLE 1 ordered_set /K28.5/D5.6/
/I2/	IDLE 2 ordered_set /K28.5/D16.2/
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IOL	InterOperability Laboratory
LOS	LOSS_OF_SYNC state
MAC	Media Access Control; defined in clause 4 of the IEEE 802.3 standard
PCS	Physical Coding Sublayer; defined in clause 36 of the IEEE 802.3 standard
PMA	Physical Medium Adapter; defined in clause 36 of the IEEE 802.3 standard
/R/	Carrier_Extend1/K23.7/
RD+	Positive Running Disparity
RD-	Negative Running Disparity
/S/	Start_of_Packet ordered_set /K27.7/
SA_z	SYNC_ACQUIRED_z; where z is the state
/T/	End_of_Packet ordered_set /K29.7/
TBI	Ten Bit Interface; used to connect the PCS with the PMA
UNH	University of New Hampshire
/V/	Error_Propagation ordered_set /K30.7/

ANNEX B - Testing Requirements

Device Under Test:

Synchronization State Machine Traversal

References: IEEE Std. 802.3, 2000 Edition - Subclauses 36.2.5.2.6: Synchronization, 37.3.1.5: State Diagrams, Figure 36-9: Synchronization state diagram, Figure36-7a: PCS receive state diagram - part a, and Figure 37-6: Auto-Negotiation State Diagram

In order to perform the synchronization tests, the DUT must start in one of two states, the LOSS_OF_SYNC state or the SYNC_ACQUIRED_1 state.

Test 36.1.1 Acquire Synchronization and Test 36.1.4 Fail to Acquire Synchronization require the DUT to start in the LOSS_OF_SYNC state. In order to force the DUT into this state the following procedure may be used:

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to transmit a continuous stream of invalid code-groups. Typically 100 transmissions of /D0.0/ with the incorrect disparity is used.

Test 36.1.2 Maintain Synchronization and Test 36.1.3 Loss of Synchronization require the DUT to start in the SYNC_ACQUIRED_1 state. In order to force the DUT into this state the following procedure may be used:

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to transmit a continuous stream of valid /I/ ordered_sets. Typically 100 transmissions of /I/ ordered_sets is enough to bring the device to the SYNC_ACQUIRED_1 state.

Synchronization Test Requirements

References: IEEE Std. 802.3, 2000 Edition - Subclause 36.2.5.2.6: Synchronization, Figures 36-9: Synchronization state diagram, and Figure36-7a: PCS receive state diagram, part a.

In order to perform any of the synchronization tests described in Group 1, the device under test (DUT) must first be observed to lose synchronization. Sending a large number of invalid code-groups to the DUT is usually sufficient to place the device in the LOSS_OF_SYNC state. Should the DUT not lose synchronization off of a large stream of invalid code-groups, Test 36.1.1 - Acquire Synchronization and Test 36.1.4 - Failure to Acquire Synchronization cannot be performed, as the device will already be in a state of synchronization. Test 36.1.2 - Maintain Synchronization cannot be performed, as it is not possible to find invalid code-groups that will cause the DUT to lose synchronization, let alone drop to lower SYNC_ACQUIRED states before losing synchronization. Test 36.1.3 – Loss of Synchronization cannot be tested, as it is highly unlikely a device will lose synchronization from a small stream of invalid code-groups and not a large stream.

Acquire and Fail to Acquire Synchronization Test Requirements

References: IEEE Std. 802.3, 2000 Edition - Subclauses 36.2.5.2.6: Synchronization, 37.3.1.5: State Diagrams, Figure 36-9: Synchronization state diagram, Figure36-7a: PCS receive state diagram - part a, and Figure 37-6: Auto-Negotiation State Diagram

In order to perform Test 36.1.1 - Acquire Synchronization and Test 36.1.4 - Fail to Acquire Synchronization, the device under test (DUT) must be able to disable auto-negotiation. Both tests require the DUT to enter the LOSS_OF_SYNC state and either exit or remain in that state given the appropriate mix of valid and invalid code-groups. This is observed by the DUT's ability to respond to or discard a valid echo request packet. An issue arises when an auto-negotiating DUT enters the LOSS_OF_SYNC state. After the DUT has transmitted a link timer's worth of idle the device will transmit break link, which consists of alternating C1:0000 and C2:0000. The testing station which must also have auto-negotiation enabled, will respond by transmitting break link. This starts the auto-negotiation process, which without its completion the DUT cannot respond to packets. However, once auto-negotiation completes the DUT is already in the SYNC_ACQUIRED state and it is impossible to monitor the state transitions.

Testing Station:

Simulation of Auto-negotiation

Purpose: To outline the procedure for allowing a non-auto-negotiating device to simulate the auto-negotiation process.

References: IEEE Std. 802.3, 2000 Edition - Subclause 37.3.1.5, Figure 37-6 - Auto-Negotiation state diagram

In order for the DUT to be able to transmit data it must first auto-negotiate with the testing station. After an auto-negotiating station powers-up or exits diagnostic mode, the station begins transmitting /C/ ordered_sets with the last two /D/ code-groups being /D0.0/. When link_timer expires the station will then begin transmitting /C/ ordered_sets with its abilities contained within the last two /D/ code-groups. The ACK(knowledge) bit of the configuration register will be set to zero.

After a station receives three consecutive, consistent /C/ ordered_sets with the link partners abilities (regardless of the ACK bit), the station will transmit /C/ ordered_sets with its abilities and the ACK bit set to one. After a station receives three consecutive, consistent /C/ ordered_sets with the link partners abilities and the ACK bit set to one the station will reset link_timer and continue to send /C/ ordered _sets with its abilities and the ACK bit set to one. At the expiration of link_timer the station will again reset the link_timer and begin transmitting IDLE(/I/). The station will continue to transmit /I/ until the link_timer expires. If when the link_timer expires the station has received three consecutive, consistent /I/s, the station is now capable of transmitting data packets. The station can now transmit data packets separated by a continuous stream of /I/.

The testing station can bring the DUT to the point where it is looking for /I/s by simply transmitting /C/ ordered_sets with its abilities and the ACK bit set to one. To ensure that the DUT is able to complete the auto-negotiation process, the testing station must advertise abilities supported by the DUT. This requires that the testing station sets bits D5 and D6 each to one, which advertises both Full- and Half-Duplex capabilities. Bits D7 and D8 are used to establish flow control and the testing station will set both of these bits to one to ensure that a device requiring pause frames can send them. Bits D12 and D13 communicate any errors to the DUT. For testing purposes these bits will be set to zero, which communicates that no error exists with the link. Bits D0-D4 and bits D9-D11 are reserved for future use and they are set to zero. Bit D14 is the ACK bit and is set to one after the testing station has received three consecutive and consistent /C/ ordered_sets. Bit D15 is the Next Page bit and it is set to zero. This is illustrated in the table below:

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
0	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0

ANNEX C – Carrier Event Handling Sequences

10bit sequences 1 bit different from *negative* /K28.5/

Sequence	code-group
1	1011111010
2	0111111010
3	0001111010
4	0010111010
5	0011011010

Sequence	code-group
6	0011101010
7	0011110010
8	0011111110
9	0011111000
10	0011111011

10bit sequences 1 bit different from *positive* /K28.5/

Sequence	code-group
1	0100000101
2	1000000101
3	1110000101
4	1101000101
5	1100100101

Sequence	code-group
6	1100010101
7	1100001101
8	1100000001
9	1100000111
10	1100000100

10bit sequences 2 bit different from *negative* /K28.5/

Sequence	code-group
1	1111111010
2	1001111010
3	1010111010
4	1011011010
5	1011101010
6	1011110010
7	1011111110
8	1011111000
9	1011111011
10	0101111010
11	0110111010
12	0111011010
13	0111101010
14	0111110010
15	0111111110
16	0111111000
17	0111111011
18	0000111010
19	0001011010
20	0001101010

Sequence	code-group
24	0001111011
25	0010011010
26	0010101010
27	0010110010
28	0010111110
29	0010111000
30	0010111011
31	0011001010
32	0011010010
33	0011011110
34	0011011000
35	0011011011
36	0011100010
37	0011101110
38	0011101000
39	0011101011
40	0011110110
41	0011110000
42	0011110011
43	0011111100

21	0001110010
22	0001111110
23	0001111000

44	0011111111
45	0011111001

10bit sequences 2 bit different from *positive* /K28.5/

Sequence	code-group
1	0000000101
2	0110000101
3	0101000101
4	0100100101
5	0100010101
6	0100001101
7	0100000001
8	0100000111
9	0100000100
10	1010000101
11	1001000101
12	1000100101
13	1000010101
14	1000001101
15	1000000001
16	1000000111
17	1000000100
18	1111000101
19	1110100101
20	1110010101
21	1110001101
22	1110000001
23	1110000111

Sequence	code-group
24	1110000100
25	1101100101
26	1101010101
27	1101001101
28	1101000001
29	1101000111
30	1101000100
31	1100110101
32	1100101101
33	1100100001
34	1100100111
35	1100100100
36	1100011101
37	1100010001
38	1100010111
39	1100010100
40	1100001001
41	1100001111
42	1100001100
43	1100000011
44	1100000000
45	1100000110