University of New Hampshire InterOperability Laboratory
Gigabit Ethernet Consortium

Superseded

As of August 6th, 2002 the Gigabit Ethernet Consortium Clause 36 Physical Coding Sublayer Conformance Test Suite Version 1.8 has been superseded by the release of the Clause 36 Physical Coding Sublayer Conformance Test Suite Version 1.9. This document along with earlier versions, are available on the Gigabit Ethernet Consortium test suite archive page.

Please refer to the following site for both current and superseded test suites:
http://www.iol.unh.edu/testsuites/gec/

GIGABIT ETHERNET

Clause 36
Physical Coding Sublayer (PCS) Test Suite

Technical Document

Last Updated: July 31, 2002 11:00 AM

InterOperability Laboratory
Research Computing Center
University of New Hampshire

121 Technology Drive, Suite 2
Durham, NH 03824
Phone: (603) 862-0166
Fax: (603) 862-0898

http://www.iol.unh.edu/consortiums/index.html

MODIFICATION RECORD

- 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 802.3-2000 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

LIST OF TABLES

GROUP 1: Synchronization

Scope: The following tests cover PCS operations specific to the synchronization state diagram.

Overview: These tests are designed to verify that the device under test properly complies to and implements the PCS state diagrams as defined in Clause 36 of the IEEE 802.3 standard.

Test #36.1.1 - Acquire Synchronization

Purpose: To verify that the device under test (DUT) acquires synchronization upon the reception of three ordered_sets each starting with a code-group containing a comma.

References:

[1] 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
[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: July 31, 2002

Discussion: 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. Furthermore, we will use CDx to represent the state COMMA_DETECT_x, ASy to represent ACQUIRE_SYNC_y, and SAz to represent SYNC_ACQUIRED_z. 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. In the synchronization state diagram a comma received in the even position will be represented by rx_even=FALSE * PUDI(/COMMA/) until the state is entered where the comma is verified and rx_even is set to TRUE.

The process begins in the LOS state where sync_status is set to FAIL. When a code-group containing a comma is detected, the process transitions to the CD1 state, the variable rx_even is set to TRUE, and the next code-group is examined. If the code-group is a valid data code-group, the process transitions to the AS1 state and sets rx_even to FALSE. If the code-group is not a valid data code-group, the process returns to the LOS state.

While in the AS1 state, the process examines each new code-group. If the code-group is a valid data code-group, the process toggles the rx_even variable. If the code-group contains the comma character and rx_even is FALSE, the process transitions to the CD2 state and toggles rx_even. If the code-group does not satisfy either of these conditions,

then the variable cgbad is asserted by the PCS and the process returns to the LOS state.

The same mechanism transports the process to the CD3 state or returns it to the LOS state. If the process enters the CD3 state and the next code-group is a valid data code-group, the process will transition to the SA1 state where sync_status is set to OK. Otherwise, the process will return to the LOS state.

Thus, synchronization is achieved upon the reception of three ordered_sets each starting with a code-group containing a comma. Each comma must be followed by an odd number of valid data code-groups. No invalid code-groups can be received prior to the reception of the three ordered_sets. The following tables give examples of the state transitions that are made in the synchronization state machine.

Table 36.1.1.1: Acquiring Synchronization with /I/ ordered_sets

code-group	/D/	/K28.5/	/D16.2/	/K28.5/	/D16.2/	/K28.5/	/D16.2/
state	LOS	CD1	AS1	CD2	AS2	CD3	SA1
alignment		EVEN	ODD	EVEN	ODD	EVEN	ODD
sync_status	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	OK

Table 36.1.1.2: Acquiring Synchronization with /C/ ordered_sets

code-group	/D/	/K28.5/	/D21.5/	/D0.0/	/D0.0/	/K28.5/	/D2.2/
state	LOS	CD1	AS1	AS1	AS1	CD2	AS2
alignment		EVEN	ODD	EVEN	ODD	EVEN	ODD
sync_status	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL

code-group	/D0.0/	/D0.0/	/K28.5/	/D21.5/	/D0.0/	/D0.0/	/K28.5/
state	AS2	AS2	CD3	SA1	SA1	SA1	SA1
alignment	EVEN	ODD	EVEN	ODD	EVEN	ODD	EVEN
sync_status	FAIL	FAIL	FAIL	OK	OK	OK	OK

The PCS receive process will remain in the LINK_FAILED state as long as sync_status=FAIL. This prevents a packet from being received while sync_status=FAIL. The packet must be preceded by at least one /I/ ordered_set. This is a result of the PCS receive process, figure 36-7a, which checks for the start of packet after the reception of an /I/ ordered_set.

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 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 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:

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to transmit a long stream of /I/ ordered_sets followed by an ARP request.
3. Instruct the testing station to transmit a continuous stream of invalid code-groups followed by one /I/ ordered_set followed by an ARP request.
4. Instruct the testing station to transmit 200 consecutive invalid code-groups with a rich transition density. Instruct the testing station to then transmit each of the following code-group sequences:
    a. /I1/I1/I1/I1/
    b. /I2/I2/I2/I2/
    c. /I1/I2/I2/I2/
    d. /I1/I2/I1/I2/
    e. /K28.5/D0.0/ (repeated 3 time) followed by an /I/ ordered_set
    f. /K28.1/D0.0/ (repeated 3 time) followed by an /I/ ordered_set
    g. /K28.5/D21.5/D0.0/D0.0/ (repeated 3 time) followed by an /I/ ordered_set
    h. /K28.5/D2.2/D0.0/D0.0/ (repeated 3 time) followed by an /I/ ordered_set
    i. /K28.5/D0.0/D0.0/D0.0/ (repeated 3 time) followed by an /I/ ordered_set
    j. /K28.5/D0.0/D0.0/D0.0/D0.0/D0.0/ (repeated 3 time) followed by an /I/ ordered_set

    following each sequence with an ARP request and a continuous stream of /I/ ordered_sets.

Observable Results:

a. The DUT should send a reply to the ARP request from step 2.
b. The DUT should lose link with the testing station after receiving the long stream of invalid code-groups in step 3. The DUT should not respond to the ARP request.
c. The DUT should respond to every ARP requests from step 4.
d. If the DUT does not respond to the ARP requests, the tester should increase the number of times the code-group sequence is sent until the device responds to the ARP request in step 4.
e. The tester should also find the minimum number of times the code-group sequences from step 4 need to be sent in order to get the DUT to respond to the ARP requests.
f. The DUT should not respond to ARP requests preceded by anything less than the sequences listed in step 4.

Possible Problems:

- If signal_detect is set to FAIL, the DUT will fail to acquire synchronization.
- If the DUT doesn't lose its link when it receives a long stream of invalid code-groups, this test cannot be performed.

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, 2000 Edition - Subclause 36.2.5.2.6: Synchronization, Figures 36-9: Synchronization state diagram, and Figure36-7a: 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: July 31, 2002

Discussion: For convenience, we use the notation LOS to denote the LOSS_OF_SYNC state, SAz to represent the SYNC_ACQUIRED_z state and SAzA to represent the SYNC_ACQUIRED_zA state. We assume that the synchronization process has reached the SA1 state (refer to discussion from test 36.1.1) and that sync_status has been set to OK. 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. In the synchronization state diagram a comma received in the even position will be represented by rx_even=FALSE * PUDI(/COMMA/) until the state is entered where the comma is verified and rx_even is set to TRUE.

While in the SA1 state, the PCS synchronization process examines each new code-group. If the code-group is a valid data code-group or contains a comma when rx_even is FALSE, the PCS asserts the variable cggood and the synchronization process toggles the rx_even variable. Otherwise, the PCS asserts the variable cgbad and the process moves to the SA2 state, toggles the rx_even variable, and sets the variable good_cgs to 0.

If the next code-group is a valid code-group, which causes the PCS to assert the variable cggood, the process transitions to the SA2A state, toggles the rx_even variable, and increments good_cgs. Otherwise it continues on to the SA3 state.

While in the SA2A state, the process examines each new code-group. For each code-group, which causes the PCS to assert cggood, the variable good_cgs is incremented. If good_cgs reaches three and if the next code-group received asserts cggood, the process returns to the SA1 state. Otherwise, the process transitions to the SA3 state.

Once in the SA3 state, the process may return to the SA2 state via the SA3A state using the same mechanisms that take the process from the SA2 state to the SA1 state.

However, another invalid code-group or comma received when rx_even is TRUE will take the process to the SA4 state.

If the process fails to return to the SA3 state via the SA4A state, it will transition to LOS where sync_status is set to FAIL.

Thus, 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). The count never goes below zero and if it reaches four, sync_status is set to FAIL.

Table 36.1.2.1: Loss of Synchronization

code-group		/K28.5/	/K28.5/	/D/	/D/	/D/
state	SA1	SA1	SA2	SA2A	SA2A	SA2A
alignment	EVEN	ODD	EVEN	ODD	EVEN	ODD
good_cgs			0	1	2	3
sync_status	OK	OK	OK	OK	OK	OK

code-group	/D/	/INVALID/	/INVALID/	/D/	/K28.5/	/INVALID/
state	SA1	SA2	SA3	SA3A	SA4	LOS
alignment	EVEN	ODD	EVEN	ODD	EVEN	ODD
good_cgs	3	0	0	1	0	0
sync_status	OK	OK	OK	OK	OK	FAIL

While xmit = DATA, the DUT shall maintain synchronization while continuously receiving the sequences listed below.

Table 36.1.2.2: Sequences for transition from SA1 to SA2 and back to SA1.

alignment	ODD	…
sequence 1	/INVALID/	/I/ followed by echo request
sequence 2	/COMMA/	/I/ followed by echo request

Table 36.1.2.3: Sequences for transition from SA1 to SA3 and back to SA1.

alignment	EVEN	ODD	…
sequence 3	/INVALID/	/INVALID/	/I/ followed by echo request
sequence 4	/INVALID/	/COMMA/	/I/ followed by echo request

Table 36.1.2.4: Sequences for transition from SA1 to SA4 and back to SA1.

alignment	ODD	EVEN	ODD	…
sequence 5	/COMMA/	/INVALID/	/COMMA/	/I/ followed by echo request
sequence 6	/COMMA/	/INVALID/	/INVALID/	/I/ followed by echo request
sequence 7	/INVALID/	/INVALID/	/COMMA/	/I/ followed by echo request
sequence 8	/INVALID/	/INVALID/	/INVALID/	/I/ followed by echo request

Table 36.1.2.5: Code-groups used for the generation of sequences

label	code-group
/COMMA/	/K28.5/
/INVALID/	Code-group of wrong running disparity. /K28.5/ if in an EVEN position and /D0.0/ if in an ODD position.
/I/	/I1/ if current running disparity is positive. /I2/ if current running disparity is negative.

The echo request packet must be preceded by at least one /I/ ordered_set. This is a result of the PCS receive process, figure 36-7a, which checks for the start of packet after the reception of an /I/ 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 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. When xmit=DATA, instruct the testing station to transmit a long stream of /I/ ordered_sets followed by an ARP request.
3. Instruct the testing station to transmit a continuous stream of invalid code-groups followed by one /I/ ordered_set followed by an ARP request.
4. Instruct the testing station to transmit each of the following code-group sequences, starting each on the even code-group position:
    a. /K28.5/invalid/
    b. /K28.5/comma/
    c. /invalid/invalid/
    d. /invalid/comma/
    e. /K28.5/comma/invalid/comma/
    f. /K28.5/comma/invalid/invalid/
    g. /K28.5/invalid/invalid/comma/
    h. /K28.5/invalid/invalid/invalid/
    i. /K28.5/invalid/K28.5/invalid/K28.5/invalid/
    j. /K28.5/invalid/I/invalid/D0.0/K28.5/invalid/
    k. /K28.5/invalid/I/K28.5/invalid/I/K28.5/invalid/
    l. /invalid/invalid/invalid/D0.0/I/D0.0/invalid/

    following each sequence with an /I/ ordered_set and echo request packet.
5. The tester may use different mappings for sequences a through h as long as mapped sequences will not permit the DUT to move from LOS to SA1. The tester may also try additional sequences that insert 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 reply to the echo request packet transmitted in step 2
b. The DUT should not reply to the echo request packet transmitted in step 3
c. The DUT should reply to every echo request packet transmitted in step 4 and 5.

Possible Problems:

- It is not possible to test every valid mapping for every valid sequence. While the DUT is observed to pass for some mapped 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 a station will lose synchronization after the reception of code-group sequences which should cause it to return to the LOSS_OF_SYNC state.

References:

[1] IEEE Std. 802.3, 2000 Edition - Subclause 36.2.5.2.6: Synchronization, Figure 36-9: Synchronization state diagram, and Figure36-7a: 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: July 31, 2002

Discussion: For convenience, we use the notation LOS to denote the LOSS_OF_SYNC state. Furthermore, we will use SAz to represent the SYNC_ACQUIRED_z state, and SAzA to represent the SYNC_ACQUIRED_zA state. A misaligned comma will be used when describing a code-group in an odd code-group position that contains a comma. 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. In the synchronization state diagram a comma received in the even position will be represented by rx_even=FALSE * PUDI(/COMMA/) until the state is entered where the comma is verified and rx_even is set to TRUE.

As specified in PCS Test #36.1.2, a DUT in the SA1 state will return to the LOS state if it receives a total of four invalid code-groups and/or misaligned commas within a string of code-groups that don't contain four consecutive valid and/or properly aligned commas.

While xmit=DATA, a DUT should lose synchronization after continually receiving any of the packets listed in Table 36.1.3.1. The loss of synchronization for the duration of link_timer is evident on the DUT by the transmission of /C/ ordered_sets when xmit=DATA.

Table 36.1.3.1: Sequences that should cause the PCS to lose synchronization.

alignment	EVEN	ODD	EVEN	ODD	EVEN
sequence 1	…	/COMMA/	/INVALID/	/COMMA/	/INVALID/
sequence 2	…	/COMMA/	/INVALID/	/INVALID/	/INVALID/
sequence 3	…	/INVALID/	/INVALID/	/COMMA/	/INVALID/
sequence 4	/INVALID/	/COMMA/	/INVALID/	/COMMA/	/COMMA/

sequence 5	/INVALID/	/INVALID/	/INVALID/	/COMMA/	/COMMA/
sequence 6	/INVALID/	/COMMA/	/INVALID/	/INVALID/	/COMMA/
sequence 7	/INVALID/	/INVALID/	/INVALID/	/INVALID/	/COMMA/

This is not an exhaustive list of sequences which when received, while in the SA1 state, will cause the DUT to lose synchronization. But it provides a good number of tests to help verify that the DUT conforms to the PCS Synchronization state diagram.

For the purposes of this test, the following mapping will be used:

Table 36.1.3.2: Code-groups used for the generation of sequences

label	code-group
/COMMA/	/K28.5/
/INVALID/	Code-group of wrong running disparity. /K28.5/ if in an EVEN position and /D0.0/ if in an ODD position.
/I/	/I1/ if current running disparity is positive. /I2/ if current running disparity is negative.

These code-groups were chosen arbitrarily, and it is possible that even though a DUT may pass the test for the code-groups chosen above, it may not pass for other equally appropriate code-groups.

The PCS receive process is incapable of receiving a packet when sync_status=FAIL. The PCS receive process is also incapable of receiving /S/ unless it is preceded by /R/ within a burst of packets or unless it is preceded by /I/ for the first packet of a burst or the only packet of a single packet transmission.

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:

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 a continuous stream of invalid code-groups followed by one /I/ ordered_set followed by an ARP request.
3. Instruct the testing station to transmit a continuous stream of 100 /I/ ordered_sets followed by the same ARP request packet.
4. Instruct the testing station to transmit each of the following code-group sequences, starting each on the even code-group position:

a. /K28.5/comma/invalid/comma/invalid/
b. /K28.5/comma/invalid/invalid/invalid/
c. /K28.5/invalid/invalid/comma/invalid/
d. /invalid/comma/invalid/comma/comma/
e. /invalid/invalid/invalid/comma/comma/
f. /invalid/comma/invalid/invalid/comma/
g. /invalid/invalid/invalid/invalid/comma/
h. /invalid/D0.0/invalid/D0.0/invalid/D0.0/invalid/D0.0/
i. /invalid/D0.0/K28.5/invalid/I/invalid/D0.0/K28.5/invalid/
j. /invalid/D0.0/I/invalid/D0.0/I/invalid/D0.0/I/invalid/D0.0/

following each sequence with an /I/ ordered_set, an echo request packet, 100 /I/ ordered_sets, and the same echo request packet.

5. The tester may use different mappings for sequences a through j as long as mapped sequences will not permit the DUT to move from LOS to SA1. The tester may also try additional sequences that insert up to three valid code-groups between code-groups that are invalid or contain commas when rx_even is TRUE.

Observable Results:

a. After xmit=DATA the DUT should be transmitting /I/ ordered_sets. The DUT should not respond to the first ARP request and it should respond to the second ARP request in step 2.
b. The DUT should only respond to the second ARP request packet in step 3.

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 would be seen in PCS Test #36.1.2. But only for the code-groups used in that test.

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 allow it to acquire synchronization.

References:

[1] 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
[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: July 31, 2002

Discussion: For convenience, we use the notation LOS to denote the LOSS_OF_SYNC state. Furthermore, we will use CDx to represent the COMMA_DETECT_x state, ASy to represent the ACQUIRE_SYNC_y state, and SAz to represent the SYNC_ACQUIRED_z state. A misaligned comma will be used when describing a code-group in an odd code-group position that contains a comma. 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. In the synchronization state diagram a comma received in the even position will be represented by rx_even=FALSE * PUDI(/COMMA/) until the state is entered where the comma is verified and rx_even is set to TRUE.

PCS Test #36.1.1 contains more details on acquiring synchronization. Table 36.1.4.1 and Table 36.1.4.2 demonstrate how a PCS acquires synchronization as it transitions through the PCS synchronization process when receiving /I/ and /C/ ordered_sets.

Table 36.1.4.1: Synchronization with /I/ ordered_sets

code-group	/D/	/K28.5/	/D16.2/	/K28.5/	/D16.2/	/K28.5/	/D16.2/
state	LOS	CD1	AS1	CD2	AS2	CD3	SA1
alignment		EVEN	ODD	EVEN	ODD	EVEN	ODD
sync_status	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	OK

Table 36.1.4.2: Synchronization with /C/ ordered_sets

code-group	/D/	/K28.5/	/D21.5/	/D0.0/	/D0.0/	/K28.5/	/D2.2/
state	LOS	CD1	AS1	AS1	AS1	CD2	AS2
alignment		EVEN	ODD	EVEN	ODD	EVEN	ODD
sync_status	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL

code-group	/D0.0/	/D0.0/	/K28.5/	/D21.5/	/D0.0/	/D0.0/	/K28.5/
state	AS2	AS2	CD3	SA1	SA1	SA1	SA1
alignment	EVEN	ODD	EVEN	ODD	EVEN	ODD	EVEN
sync_status	FAIL	FAIL	FAIL	OK	OK	OK	OK

A PCS in the LOS 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 LOS state. The DUT should fail to acquire synchronization when continually receiving any of the code-group sequences listed in Table 36.1.4.3.

Table 36.1.4.3: Code-group sequences that should not allow synchronization to be acquired

rx_even	…	TRUE	FALSE	TRUE	FALSE	TRUE
sequence 1	/COMMA/	INVALID/	…	…	…	…
sequence 2	/COMMA/	/COMMA/	…	…	…	…
sequence 3	/COMMA/	/D/	/INVALID/	…	…	…
sequence 4	/COMMA/	/D/	/COMMA/	/INVALID/	…	…
sequence 5	/COMMA/	/D/	/COMMA/	/COMMA/	…	…
sequence 6	/COMMA/	/D/	/COMMA/	/D/	/INVALID/	…
sequence 7	/COMMA/	/D/	/COMMA/	/D/	/COMMA/	/COMMA/
sequence 8	/COMMA/	/D/	/COMMA/	/D/	/COMMA/	/INVALID/

This is not an exhaustive list of sequences which when received while in the LOS state will prevent the DUT from acquiring synchronization. But it provides a good number of tests to help verify that the DUT conforms to the PCS Synchronization state diagram.

Table 36.1.4.4: Code-groups used for the generation of sequences

label	code-group
/COMMA/	/K28.5/
/INVALID/	Code-group of wrong running disparity. /K28.5/ if in an EVEN position and /D0.0/ if in an ODD position.
/I/	/I1/ if current running disparity is positive. /I2/ if current running disparity is negative.

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:

1. Bring the DUT to the state where xmit=DATA.
2. Once xmit=DATA, instruct the testing station to transmit a long stream of /I/ ordered_sets followed by an ARP request.
3. Instruct the testing station to transmit a continuous stream of invalid code-groups followed by one /I/ ordered_set followed by an ARP request.
4. Instruct the testing station to transmit 200 consecutive /INVALID/ code-groups followed by 100 transmissions of the following sequences. Follow each sequence with one /I/ ordered_set and an echo request packet. Follow the echo request packet with 100 /I/ ordered_sets and then another echo request packet.
    a. /comma/invalid/
    b. /comma/comma/
    c. /comma/D0.0/invalid/
    d. /comma/D0.0/comma/invalid/
    e. /comma/D0.0/comma/comma/
    f. /comma/D0.0/comma/D0.0/invlaid/
    g. /comma/D0.0/comma/D0.0/comma/comma/
    h. /comma/D0.0/comma/D0.0/comma/invalid/
    i. /K28.5/D2.2/D0.0/D0.0/K28.5/D21.5/D0.0/D0.0/K28.5/invalid/
    j. /K28.5/D0.0/D0.0/D0.0/D0.0/D0.0/D0.0/invalid/
    k. /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/
5. The tester may use different mappings for sequences 1 through 7 as long as mapped sequences will not permit the DUT to move from LOS to SA1 prior to the reception of the first echo request packet. The tester may also try additional sequences that substitute up to three valid /D/ code-groups for each /D/ code-group.

Observable Results:

a. The DUT should respond to the echo request packet from step 2.
b. The DUT should not respond to the echo request packet from step 3.

c. The DUT should respond only to the second echo request packet from steps 4 and 5.

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. You must ensure that the DUT is receiving a signal.

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, 2000 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: July 31, 2002

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. For example:

Table 36.2.1.1: EPD for a running disparity value of RD+.

code-group	/D/	/T/	/R/	/K28.5/	/D5.6/	/K28.5/
running disparity	XX	RD+	RD+	RD+	RD-	RD-

Table 36.2.1.2: EPD for a running disparity value of RD-.

code-group	/D/	/T/	/R/	/K28.5/	/D16.2/	/K28.5/
running disparity	XX	RD-	RD-	RD-	RD+	RD-

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. For example:

Table 36.2.1.3: Encoding of /S/ for a running disparity value of RD+.

code-group	/D/	/T/	/R/	/R/	…	/S/
running disparity		RD+	RD+	RD+	RD+	RD+

/V/ ordered_sets may be observed as the jam pattern a device under test (DUT) sends to enforce a collision during 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. For example:

Table 36.2.1.4: Encoding of /V/ for a running disparity value of RD+.

code-group	/D/	/T/	/R/	/R/	/V/	/V/
running disparity		RD+	RD+	RD+	RD+	RD+

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:

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, force the DUT to transmit a packet containing both forms of every valid data code-group listed in Table 36-1.
3. If the DUT is capable of packet bursting, 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. Repeat, ensuring that the running disparity after the last byte of data issued by the DUT is negative.
4. If the DUT performs carrier extension, force the DUT to issue a packet that requires extension. Ensure that the running disparity after the last byte of data is positive. Instruct the testing station to collide with the packet while extension is being sent. Repeat, ensuring that the running disparity after the last byte of data issued by the DUT is negative.

Observable Results:

a. At all times, the DUT should transmit the correct encoding of the appropriate code-group as specified in step 2.
b. The DUT should transmit the correct encoding of the appropriate code-group as specified in step 3.

c. The DUT should transmit the correct encoding of the appropriate code-group as specified in step 4.

Possible Problems: None

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, 2000 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: July 31, 2002

Discussion: /I/ ordered_sets are transmitted during the normal inter-packet gap. /I/ follows the last /C/ ordered_set sent, the last /R/ of the EPD or extension, and precedes the transmission of /S/ for a single packet transmission or the first packet of a burst. /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:

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, force the DUT to transmit a packet such that the running disparity following the last byte of data is positive.

3. Force the DUT to transmit a packet such that the running disparity following the last byte of data is negative.

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, 2000 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: July 31, 2002

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. Figure 36-5, the PCS transmit ordered_set state diagram, illustrates this requirement. When the transmit process enters the EPD2_NOEXT state, an /R/ code-group is transmitted and the status of the tx_even variable is checked. If tx_even is TRUE, the process moves to the EPD3 state, another /R/ code-group is transmitted, and then the process moves on to the XMIT_DATA state. If tx_even is FALSE, the process goes directly to the XMIT_DATA state. While in the XMIT_DATA state, the PCS transmit process sends /I/. Note that the value of tx_even is toggled following the transmission of each code-group.

Thus, 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. 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. Consider a normal EPD:

Table 36.2.3.1: Normal EPD

case 1	…	/T/	/R/	/I/
case 2	/T/	/R/	/R/	/I/
alignment	TRUE	FALSE	TRUE	FALSE

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:

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, 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. Force the DUT to transmit a packet which does not require extension and requires /T/ to be transmitted starting in an odd code-group position.
4. If the DUT is capable of performing frame extension, force the DUT to transmit a packet that requires extension and requires /T/ to be transmitted in an even code-group position.
5. If the DUT is capable of performing frame extension, force the DUT to transmit a packet that requires extension and requires /T/ to be transmitted in an odd code-group position.
6. If the DUT is capable of performing packet bursting, force 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.
7. If the DUT is capable of performing packet bursting, force 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.

Observable Results:

a. The DUT shall ensure that /I/ begins in an even code-group position after replying to the echo request packets from step 2 though 7.

Possible Problems: None

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, 2000 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 receiving arbitrary sequences of 10-bit code-groups using the signaling method of clause 38 or clause 39.

Last Modification: July 31, 2002

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:

1. Disconnect the receiver of the DUT from the transmitter of the testing station. Force the DUT to restart auto-negotiation. The testing station will observe any activity from the DUT.
2. Reconnect the receiver of the DUT to the transmitter of the testing station. Force the testing station to transmit /I/ ordered_sets. The testing station will observe any activity 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 to maintain the alternating transmission of /C1/ and /C2/ ordered_sets when transitioning from break link to abilities.

Possible Problems:

- The DUT may not support Auto-Negotiation.

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 – Currently not performed

Purpose: To verify that the device under test (DUT) can distinguish between valid and invalid code-groups.

References:

[1] IEEE Std. 802.3, 2000 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, 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: July 31, 2002

Discussion: To be considered valid, code-groups received by the PCS receive process must be located in the column of Table 36-1 or 36-2 that corresponds to the receiver's current running disparity. The running disparity is calculated for each code-group received, regardless of its validity. For example, if the receiver's current running disparity value is negative and 000101 1010 is received, then the current running disparity value will remain negative and the RD- column will be used to check the next incoming code-group.

For the purposes of this test, the following code-groupsare considered invalid:

a. A code-group not found in the column coresponding the receiver's current running disparity.
b. The reserved special code-groups of Table 36-2.
c. /V/, the Error_Propagation ordered_set from Table 36-3.

If any invalid code-group is received within a packet, the PCS receive process will guarantee that the MAC receives that 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.

This test will substitute every valid code-group (both positive and negative running disparity encodings) for each invalid code-group. Each packet will contain only one invalid code-group, the running disparity of the remaining packet will be consistent with the invalid code-group, and the FCS will contain the CRC value for the packet prior to the substitution.

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 a three packet sequence where each packet is separated by the minimum inter-packet gap. The first and third packets shall be valid echo request packets. The second packet shall be a valid echo request packet with one valid code-group substituted with an invalid one.
3. Repeat step 2 until every invalid code-group has been substituted for every valid code-group (both positive and negative running disparity encodings).

Observable Results:

a. The DUT should reply to the first and third echo request packets. The DUT shall report frameCheckError for the second packet in the sequence.

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, 2000 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

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: July 31, 2002

Discussion: The variable carrier_detect is set to TRUE upon the reception of a code-group that is two or more bits different from /K28.5/ and is in an even code-group position. 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.

The following code-groups do not have at least a two bit difference from /K28.5/. If they form a valid /D/ or /K/ code-group that code-group is specified. Otherwise, /INVALID/ is placed next to the code-group to mark it as an /INVALID/ code-group.

Table 36.3.2.1: Code-Groups which have less than a two bit difference from /K28.5/

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	/INVALID/	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/

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.
4. Instruct the testing station to transmit an ARP request followed by idle and then 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 preceded with a two code-group sequence. The second packet shall be a valid echo request packet. The first code-group shall be /K28.5/ and the second code-group shall be any data code-group other than /D21.5/ or /D2.2/, this code-group sequence should be considered as part of the inter-packet gap. Repeat until every valid code-group other than /D2.2/ or /D21.5/ has been transmitted as the second code-group.

Observable Results:

a. 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.
b. The DUT should reply to all three packets sent in step 3.
c. The DUT should reply to all three packets sent in step 4

Possible Problems:

- If the second packet of the sequence fails to be lost for any of the appropriate sequences, it cannot be determined whether carrier_status was not set to true or if 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, 2000 Edition - Subclause 35.2.1.5: Response to error indications from GMII, 36.2.4.14: 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: July 31, 2002

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. Table 36.3.3.1 illustrates the valid EPD encodings.

Table 36.3.3.1: Valid EPDs

Alignment	EVEN	ODD	EVEN	ODD
EPD 1	…	/T/	/R/	/R/
EPD 2	/T/	/R/	/K28.5/	/D/*

* additional /D/ is used to force the following /I/ to begin in an even code-group position with the correct running disparity.

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 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.

Table 36.3.3.2 contains a list of EPDs that should cause the PCS receive process to invalidate the preceding packet.

Table 36.3.3.2: Invalid EPDs

alignment	EVEN	ODD	EVEN	ODD
EPD 3	…	/T/	/R/	/K28.5/
EPD 4	…	/T/	!/R/	/R/
EPD 5	/T/	!/R/	/K28.5/	/D/*
EPD 6	…	/T/	/R/	!/R/
EPD 7	/T/	/R/	!/K28.5/	/D/*
EPD 8	/R/	/R/	/R/	/D/*
EPD 9	…	/R/	/R/	/R/
EPD 10	/K28.5/	/D/	/K28.5/	/D/*
EPD 11	/K28.5/	/D21.5/	/D0.0/	/D/*
EPD 12	/K28.5/	/D2.2/	/D0.0/	/D/*

* additional /D/ is used to force the following /I/ to begin in an even code-group position with the running disparity.

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 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 with a valid EPD. The second packet shall be a valid echo request packet with EPD 3 as given in Table 36.3.3.2.

3. Repeat step 2 for every invalid EPD given in Table 36.3.3.2. In cases where multiple encodings may satisfy a particular EPD element (e.g. !/R/), the tester is encouraged to use as many encodings as possible.

Observable Results:

a. The DUT should reply to the first echo request packet. The DUT shall report frameCheckError for all second packets with invalid EPDs starting with an even alignment.

b. The DUT should reply to the first echo request packet. The DUT shall report frameCheckError for all second packets with invalid EPDs starting with an odd alignment.

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. It is the responsibility of the tester to try as many encodings as possible.

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, 2000 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: July 31, 2002

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.

The code-groups in Table 36.3.4.1 are no more than one bit away from /K28.5/. If they form a valid /D/ or /K/ code-group then that code-group is specified. Otherwise, /INVALID/ is placed next to the code-group to mark it as an invalid code-group.

Table 36.3.4.1: Code-Groups which have less than a two bit difference from /K28.5/

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 Table 36.3.4.1 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.

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:

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 Appendix 36A.
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/D2.2/D0.0/D0.0/ three times followed by a continuous stream of IDLE.
4. Repeat steps 1 through 3 with /D2.2/ being replaced by /D21.5/.
5. Repeat steps 1 through 4 replacing /D0.0/ with every other /D/ code-group.

6. Repeat steps 1 through 5 replacing /K28.5/ of the first /C/ ordered_set with every code-group listed in Table 36.3.4.1.

Observable Results:

a. 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 doesn't 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.

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
/I/	IDLE ordered_set
/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/
/V/	Error_Propagation ordered_set /K30.7/

ANNEX B - Testing Requirements

Device Under Test:

Synchronization Test Requirements

Purpose: To outline the rational behind the inability to perform synchronization testing when the DUT is not observed to lose synchronization.

References:

[1] 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

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: July 31, 2002

Discussion: 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.

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 a continuous stream of invalid code-groups followed by one /I/ ordered_set followed by an ARP request.

Follow the ARP request packet with 100 /I/ ordered_sets followed by the same echo request packet.

Observable Results:

a. After xmit=DATA the DUT should be transmitting /I/ ordered_sets. The DUT should not respond to the first ARP request and it should respond to the second ARP request in step 2. If the first ARP request is responded to the Group 1 tests should not be performed.

Possible Problems: None

Acquire and Fail to Acquire Synchronization Test Requirements

Purpose: To outline the rational behind the inability to perform synchronization Tests 36.1.1 and 36.1.4.

References:

[1] 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

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: July 31, 2002

Discussion: In order to perform Test 36.1.1 - Acquire Synchronization and Test 36.1.4 - Fail to Acquire Synchronization, the device under test (DUT) will have to be capable of manual configuration. 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 DUTs 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.

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.
2. Once xmit=DATA, instruct the testing station to transmit a continuous stream of /I/ ordered_sets.

Observable Results:

a. After xmit=DATA the DUT should be transmitting /I/ ordered_sets.

Possible Problems: None

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:

[1] IEEE Std. 802.3, 2000 Edition - Subclause 37.3.1.5, Figure 37-6 - Auto-Negotiation state diagram

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. The testing station must implement or be able to emulate the auto-negotiation process described in clause 37.

Last Modification: July 31, 2002

Discussion: For convenience, we use the notation LOS to denote the LOSS_OF_SYNC state. Furthermore, we will use CDx to represent the state COMMA_DETECT_x, ASy to represent ACQUIRE_SYNC_y, and SAz to represent SYNC_ACQUIRED_z. A misaligned comma will be used when describing a code-group in an odd code-group position which contains a comma.

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 that the DUT is also capable of. 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.

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:

Ensure that the DUT is in the LOS state. This can be achieved by either disconnecting the receive wire of the DUT or by deactivating the transmitter of the testing station.
Reconnect the receive wire of the DUT or reactivate the transmitter of the testing station, depending upon what was performed in step #1.

The test station is instructed to transmit /C/ ordered_sets with the ACK bit set to one. The table below shows exactly how the last two /D/ code-groups should be filled.

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

Upon the reception of /I/ from the DUT, the testing station will transmit /I/.
While the DUT is transmitting /I/, give the DUT's MAC a packet to send. Upon the expiration of the DUT's link_timer the MAC should transmit the packet.

Observable Results:

Prior to reaching the SA1 state, the DUT will transmit /C/ ordered_sets with /D0.0/ contained within the last two /D/ code-groups.

Upon reaching the SA1 state, and after link_timer is expired, the DUT will transmit /C/ ordered_sets with its abilities contained within the last two /D/ code-groups. This should immediately be followed by the testing station's transmission of /C/ ordered_sets with its abilities and the ACK bit set to one.

Eventually the DUT will begin transmitting /C/ ordered_sets containing its abilities along with the ACK bit set to one.

After receiving three consecutive, consistent /C/ ordered_sets with the testing stations abilities and the ACK bit set to one and after the link_timer has expired, the DUT will begin transmitting /I/s.

After the expiration of link_timer, the DUT should transmit the waiting data packet signaling that the auto-negotiation process is complete.

Possible Problems: None