University of New Hampshire InterOperability Laboratory
Gigabit Ethernet Consortium

Superseded

As of July 31st, 2002 the Gigabit Ethernet Consortium Clause 36 Physical Coding Sublayer Conformance Test Suite Version October 1997 has been superseded by the release of the Clause 36 Physical Coding Sublayer Conformance Test Suite Version 1.8. 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Please refer to the following site for both current and superseded test suites: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
Version October 1997

Technical Document

Last Updated: October, 1997

InterOperability Laboratory
Research Computing Center
University of New Hampshire

121 Technology Drive, Suite 2
Durham, NH 03824
Phone: (603) 862-4822
Fax: (603) 862-0898

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

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 /K28.5/.

References:

- IEEE 802.3z/D3.2 - Subclause 36.2.5.2.6 and Figure 36-9: Synchronization state diagram.
- Appendix 36A

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: October 6th, 1997

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.

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 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
rx_even	—	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
sync_status	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	OK

Table 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
rx_even	—	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
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
rx_even	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
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.

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. 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 transmit 200 consecutive 10-bit code_groups consisting of all zeros. Instruct the testing station to then transmit three consecutive /I2/ ordered_sets followed by an echo request packet followed by a continuous stream of /IDLE/.
3) Repeat steps one and two by replacing /I2/ with /I1/.

Observable Results:

a) When xmit=DATA, the DUT should continuously be transmitting /I/ ordered_sets. After receiving the echo request packet from the testing station, the DUT should send a reply.

Possible Problems:

1) If signal_detect is set to FAIL, the DUT will fail to acquire synchronization.

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:

- IEEE 802.3z/D3.2 - Subclause 36.2.5.2.6 and Figure 36-9: Synchronization state diagram.
- Appendix 36A

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: October 7th, 1997

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.

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 1: Loss of Synchronization

code_group	—	/K28.5/	/K28.5/	/D/	/D/	/D/
state	SA1	SA1	SA2	SA2A	SA2A	SA2A
rx_even	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
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
rx_even	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
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 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 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 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 5: Code_groups used for the generation of sequences

label	code_group
/COMMA/	/K28.5/
/INVALID/	000000 0000

The echo request packet must be preceded by at least one /I/ ordered_set. This is because the PCS receive process, figure 36-7a, 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 Appendix 36A.
2) When xmit=DATA, instruct the testing station to transmit sequence 1, using the mapping from table 5.
3) Repeat step 2 for sequences 2 through 8.
4) The tester may use different mappings for sequences 1 through 8 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. For example, the tester may wish to try:

Table 6: Example sequence

alignment	ODD	EVEN	ODD	…
code_group	/INVALID/	cggood	/COMMA/	/I/×4 followed by echo request

Observable Results:

a) The DUT should reply to every echo request packet transmitted within test sequences 1-8.

Possible Problems:

1) 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.
2) 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:

- IEEE 802.3z/D3.2 - Subclause 36.2.5.2.6 and Figure 36-9: Synchronization state diagram.
- Appendices 36A

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: October 7th, 1997

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 which contains a comma.

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 which 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 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 1: Sequences which should cause the PCS to lose synchronization.

Alignment	**EVEN**	**ODD**	**EVEN**	**ODD**	**EVEN**	**…**
sequence 1	…	/COMMA/	/INVALID/	/COMMA/	/INVALID/	/GOOD/ × 10
sequence 2	…	/COMMA/	/INVALID/	/INVALID/	/INVALID/	/GOOD/ × 10
sequence 3	…	/INVALID/	/INVALID/	/COMMA/	/INVALID/	/GOOD/ × 10
sequence 4	/INVALID/	/COMMA/	/INVALID/	/COMMA/	/GOOD/	/GOOD/ × 9
sequence 5	/INVALID/	/INVALID/	/INVALID/	/COMMA/	/GOOD/	/GOOD/ × 9
sequence 6	/INVALID	/COMMA/	/INVALID/	/.INVALID/	/GOOD/	/GOOD/ × 9
sequence 7	/INVALID	/INVALID/	/INVALID/	/.INVALID/	/GOOD/	/GOOD/ × 9

This is not an exhaustive list of packets 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 2: Code_groups used for the generation of sequences

label	code_group
/COMMA/	/K28.5/
/INVALID/	000000 0000
/GOOD/	/D3.2/
/GOOD/ × n	n repetitions of /GOOD/

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 code_group /GOOD/ was chosen specifically to not allow the synchronization process to regain synchronization. This is ensured as long as no commas are transmitted in the place of /GOOD/.

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 Appendix 36A.
2) Once xmit=DATA, instruct the testing station to transmit sequence 1 (using the mapping from table 5) followed by one /I/ ordered_set and then an echo request packet. Follow the echo request packet 50 /I/ ordered_sets followed by the same echo request packet.
3) Repeat step 2 for sequences 2 through 7.
4) 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. 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. For example, the tester may wish to try the test sequence in table 3.

Table 6: Example sequence

alignment	**ODD**	**EVEN**	**ODD**	**EVEN**	**ODD**	**…**
code_group	/COMMA/	/VALID/	/INVALID/	/INVALID/	/COMMA/	/GOOD/ × 10

Observable Results:

a) After xmit=DATA the DUT should be transmitting /I/ ordered_sets. The DUT should not respond to the first echo request packet and it should respond to the second echo request packet.

Possible Problems:

1) 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.
2) It is possible for the DUT to have moved to the LOS 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 packet sequences which should not allow it to acquire synchronization.

References:

- IEEE 802.3z/D3.1 - Subclause 36.2.5.2.6 and Figure 36-9: Synchronization state diagram.
- Appendices 36A

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: October 7th, 1997

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 which contains a comma.

PCS Test #36.1.1 contains more details on acquiring synchronization. Tables 1 and 2 demonstrate how a PCS acquires synchronization as it transitions through the PCS synchronization process when receiving /I/ and /C/ ordered_sets.

Table 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
rx_even	—	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
sync_status	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	OK

Table 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
rx_even	—	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
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
rx_even	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
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 3.

Table 3: Code_group sequences which 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	/COMMA/	/D/	/COMMA/	/COMMA/	…	…
sequence 5	/COMMA/	/D/	/COMMA/	/D/	/INVALID/	…
sequence 6	/COMMA/	/D/	/COMMA/	/D/	/COMMA/	/COMMA/
sequence 7	/COMMA/	/D/	/COMMA/	/D/	/COMMA/	/INVALID/

This is not an exhaustive list of packets which when received while in the SA1 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.

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 Appendix 36A.
2) Once xmit=DATA, instruct the testing station to transmit four consecutive /INVALID/ code_groups followed by sequence 1 (using the mapping from table 5) to the DUT for 1 minute. Follow sequence 1 with one /I/ ordered_set and an echo request packet. Follow the echo request packet with 50 /I/ ordered_sets and then another echo request packet.
3) Repeat steps 1 and 2 for sequences 2 through 7.
4) 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. For example, the tester may wish to try:

Table 6: Example sequence

alignment	**ODD**	**EVEN**	**ODD**	**EVEN**	**ODD**
code_group	/COMMA/	/D/	/D/	/INVALID/	/COMMA/

Observable Results:

a) The DUT should not respond to the first echo request packet and it should respond to the second echo request packet.

Possible Problems:

1) 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.

Test #36.2.1 8B/10B Encoding

Purpose: To verify that the device under test selects the proper encoding for transmitted code_groups.

References:

- IEEE 802.3z/D3.2 - Subclauses 36.2.4.4 and 36.2.4.5; Tables 36-1 and 36-2.
- Appendix 36A

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: October 9th, 1997

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.

Subclause 36.2.4.4 describes the calculation of running disparity.

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 1: EPD for a running disparity value of RD+.

code_group	/D/	/T/	/R/	/K28.5/	/D5.6/	/K28.5/
running disparity	—	RD+	RD+	RD+	RD-	RD-

Table 2: EPD for a running disparity value of RD-.

code_group	/D/	/T/	/R/	/K28.5/	/D16.2/	/K28.5/
running disparity	—	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 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 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 Appendix 36A.
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. Ensure that the running disparity value is negative after the transmission of the last data byte.
3) Force the DUT to transmit a packet containing both forms of every valid data code_group listed in table 36-1. Ensure that the running disparity value is positive after the transmission of the last data byte.
4) 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.

5) 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 negative. 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.

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:

- IEEE 802.3z/D3.2 - Subclause 36.2.4.12 and Figure 36-6
- Appendix 36A

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: October 9th, 1997

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 Appendix 36A.
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:

- IEEE 802.3z/D3.2 - Subclause 36.2.4.15.1; Figures 36-5 and 36-6
- Appendix 36A

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: October 9th, 1997

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 a odd 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 even 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 1: Normal EPD

case 1	…	/T/	/R/	/I/
case 2	/T/	/R/	/R/	/I/
tx_even*	TRUE	FALSE	TRUE	FALSE

* indicates the value of tx_even prior to the transmission of the code_group in that column

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 Appendix 36A.
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.

Possible Problems: none

Test #36.2.4 /C/ Transmission Order

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

References:

- IEEE 802.3z/D3.2 - Subclause 36.2.5.1; Figures 36-5 and 36-6

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: October 9th, 1997

Discussion:

/C/ ordered_sets are used to convey 16-bit configuration registers to the link partner. Figure 36-5, the PCS 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) Force the DUT to restart auto-negotiation. The testing station will observe any activity from the DUT.

Observable Results:

a) After restarting auto-negotiation, the DUT shall transmit /C/ ordered_sets as alternating /C1/ and /C2/ ordered_sets.

Possible Problems: none

Test #36.3.1 8B/10B Decoding

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

References:

- IEEE 802.3z/D3.2 - Subclauses 36.2.4.3, 36.2.4.4 and 36.2.4.6 and Tables 36-1, 36-2 and 36-3
- Appendix 36A

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: October 9th, 1997

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.

Subclause 36.2.4.4 defines the calculation of running disparity.

For the purposes of this test, the following code_groups are considered invalid**:**

a) A code_group not found in the column corresponding 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 Appendix 36A.
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:

- IEEE 802.3z/D3.2 - Subclauses 36.2.4.16 and 36.2.5.3 and Figures 36-5, 36-7a and 36-7b.
- Appendix 36A

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: October 8th, 1997

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.

In order for a carrier event to be received, it must be preceded by a valid /I/ ordered_set. Otherwise, the packet will be dropped by the receive process.

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 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/

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 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 prepended with a two code_group sequence. 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 /D/ other than /D2.2/ or /D21.5/ has been transmitted as the second code_group.
3) Repeat step 2 by substituting the first code_group of the two code_group sequence with any code_group other than /S/. Repeat this substitution until the first code_group has been replaced by every code_group other than /S/. The second code_group may be anything other than /D21.5/ or /D2.2/.

Observable Results:

a) The DUT should reply to the first and third echo request packets. The DUT should reply to second echo request packet for each of the sequences sent in step 2 and each sequence from step 3 whose first code_group appears in table 1.

Possible Problems:

1) 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:

- IEEE 802.3z/D3.2 - Subclause 36.2.4.14 and Figure 36-7b
- Appendix 36A

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: October 9th, 1997

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 1 illustrates the valid EPD encodings.

Table 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

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

Table 2: Invalid EPDs

alignment	EVEN	ODD	EVEN	ODD
EPD 3	/T/	/R/	/R/	/K28.5/
EPD 4	…	/T/	/R/	/K28.5/
EPD 5	…	/T/	!/R/	/R/
EPD 6	/T/	!/R/	/K28.5/	/D/*
EPD 7	…	/T/	/R/	!/R/
EPD 8	/T/	/R/	!/K28.5/	/D/*
EPD 9	/R/	/R/	/R/	/D/*
EPD 10	…	/R/	/R/	/R/
EPD 11	/K28.5/	/D/	/K28.5/	/D/*
EPD 12	/K28.5/	/D21.5/	/D0.0/	/D/*
EPD 13	/K28.5/	/D2.2/	/D0.0/	/D/*

* additional /D/ is used to force the following /I/ to begin in an even code_group position

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 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 with valid EPDs. The second packet shall be a valid echo request packet with EPD 3 as given in table 2.
3) Repeat step 2 for every invalid EPD given in table 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 and third echo request packets. The DUT shall report frameCheckError for the second packet in the sequence.

Possible Problems:

1) 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 (while xmit=DATA)

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

References:

- IEEE 802.3z/D3.2 - Figure 36-7a
- Appendix 36A

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: October 9th, 1997

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 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 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 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 1.

Observable Results:

a) After the DUT has received the three consecutive, consistent /C/ ordered_sets it shall transmit /C/ ordered_sets with Config_Reg set to zero.

Possible Problems:

1) If the DUT doesn't restart auto-negotiation, it is impossible to determine whether or not it is the receive state diagram or the auto-negotiation state diagram that is failing.

Appendix 36A: Simulation of Auto-Negotiation

Purpose:

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

References:

- IEEE 802.3z/D3.1 - 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:

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 a /C/ ordered_set with the ACK bit set to one. 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:

1) 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.
2) Reconnect the receive wire of the DUT or reactivate the transmitter of the testing station, depending upon what was performed in step #1.
3) 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

4) Upon the reception of /I/ from the DUT, the testing station will transmit /I/.
5) 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:

a) Prior to reaching the SA1 state, the DUT will transmit /C/ ordered_sets with /D0.0/ contained within the last two /D/ code_groups .
b) 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 zero.
c) Eventually the DUT will begin transmitting /C/ ordered_sets containing its abilities along with the ACK bit set to one.
d) 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.
e) After the expiration of link_timer, the DUT should transmit the waiting data packet signaling that the auto-negotiation process is complete.