Clause 37 Auto-Negotiation

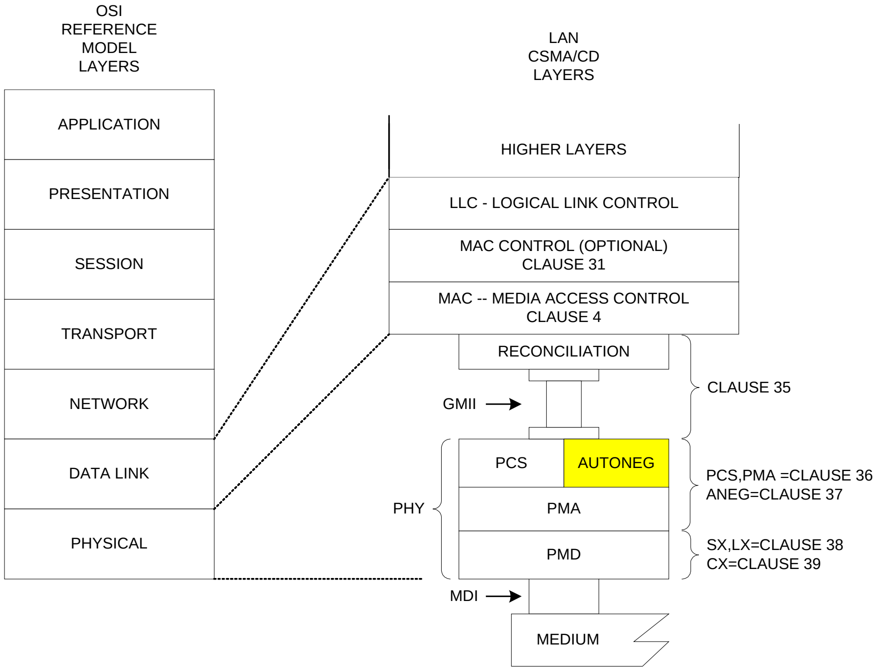
An introduction to the
Auto-Negotiation process

Presentation Overview:

- Location in the OSI Stack
- Interface to the PCS
- Auto-Negotiation functionality

OSI Stack

- Auto-Negotiation is located inside the PCS layer



1000BASE-SX
1000BASE-LX
1000BASE-CX

1000Mb/s

Inside the PCS

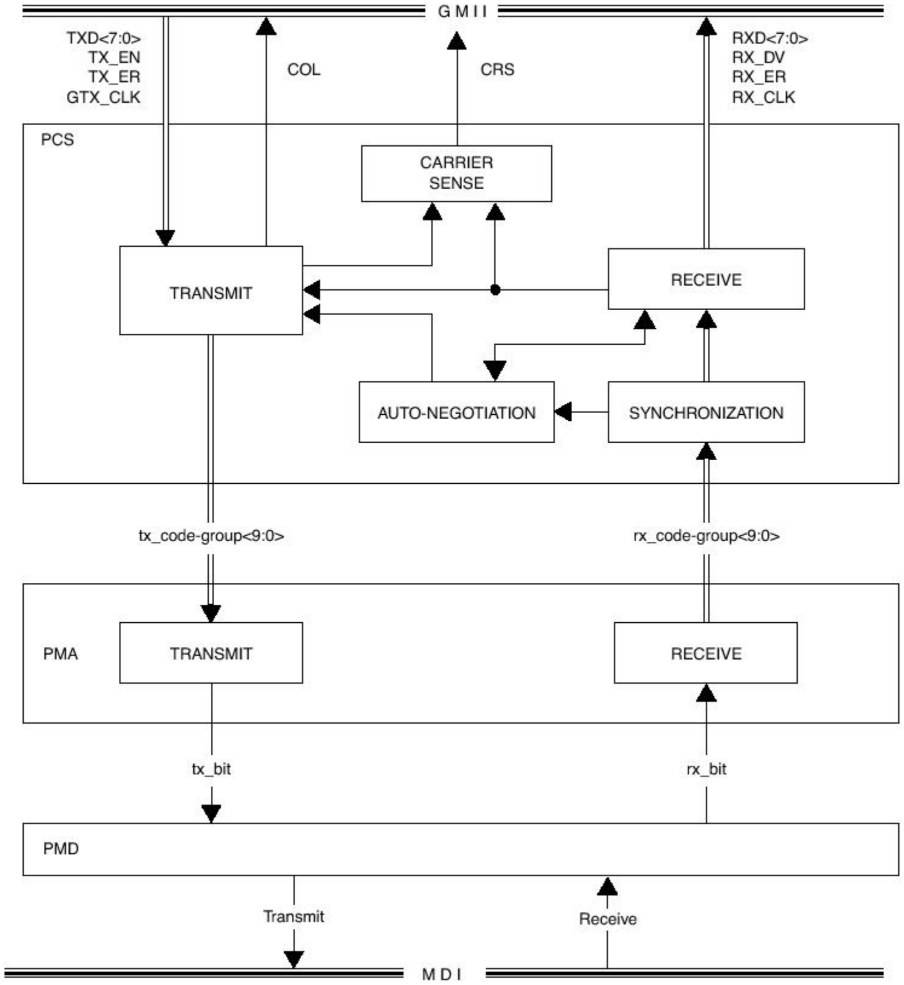


Figure 36–2—Functional block diagram

Inside the PCS

- Since Auto-Negotiation is located inside the PCS, it means that the device already needs to have established synchronization with the link partner at Gig speed (1.25 Gigabits/second on the line) before any information can be transferred

Inside the PCS

- Auto-Negotiation has control of what is sent while xmit = CONFIGURATION and IDLE
- The PCS controls what is sent when xmit = DATA
- PCS supplies link status information to the auto-negotiation layer

What is Auto-Negotiation

- Auto-Negotiation is responsible for ensuring that the optimal link is resolved between two link partners
- Information is exchanged in a 16-bit word called a configuration register (Config_Reg), which is part of a /C/ Ordered Set
- Duplex and Pause modes are considered when determining the optimal link
- Auto-Negotiation is also responsible for exchanging remote fault and any additional information

Ordered Sets

- An ordered set is defined as a Special (K) code group followed by one or more Data (D) code groups
- There are four ordered sets that are used in Auto-Negotiation
- Two are Idle (/I/) Ordered Sets and two are Configuration (/C/) Ordered Sets

Idle Ordered Sets

- The two possible Idle Ordered Sets are known as I1 and I2
- I1 is /K28.5/D5.6/
- I2 is /K28.5/D16.2/
- I1 is used to flip Running Disparity, while I2 is used to maintain Running Disparity

/C/ Ordered Sets

- The two possible /C/ ordered sets are known as C1 and C2
- C1 is /K28.5/D21.5/Config_Reg
- C2 is /K28.5/D2.2/Config_Reg
- The Config_Reg is made of two additional Data code groups

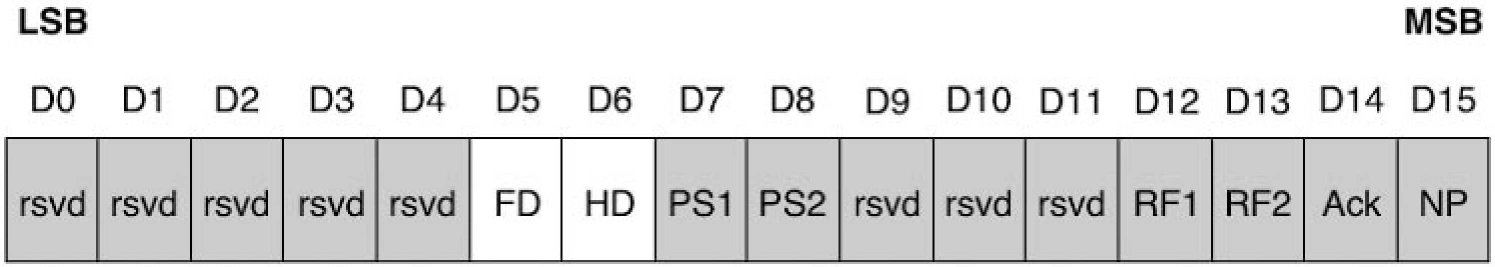
Configuration Register

- The Config_Reg is a 16-bit word that contains the information to be transferred
- The Config_Reg can be used to create several types of pages, such as base pages and next pages
- The base page looks like:

LSB															MSB
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

Duplexes

- The supported duplexes are advertised in bits 5 and 6 of the base page
- There are two possible duplexes
  - Full Duplex – Bit 5 – information is passed bi-directionally
  - Half Duplex – Bit 6 – information can only be passed one way at a time



LSB															MSB
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

Pause Modes

- The supported pause modes are advertised in bits 7 and 8 of the base page
- There are two possible pause modes
  - Symmetric pause - Bit 7 – The device can support pause frames flowing in both directions
  - Asymmetric pause – Bit 8 – The device can only support pause frames flowing in one direction

LSB MSB

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

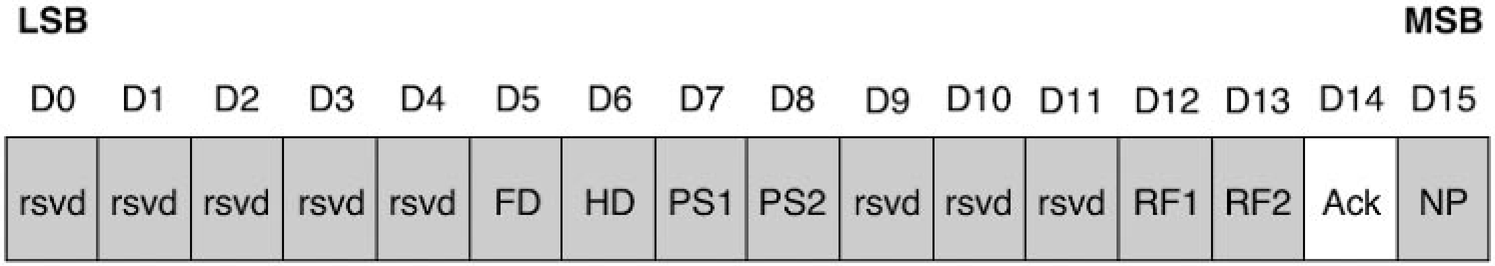
Remote Fault

- The remote fault bits are passed in bits 12 and 13 of the base page
- Remote Fault passes link failure information from one device to another

LSB															MSB
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

Acknowledge

- The Acknowledge bit is bit 14 of the base page
- The Acknowledge bit lets the link partner know that the device has received the link partner’s config_reg
- There is an ability_match function that must be true before the Acknowledge bit can be set to one.



LSB															MSB
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

Next Page

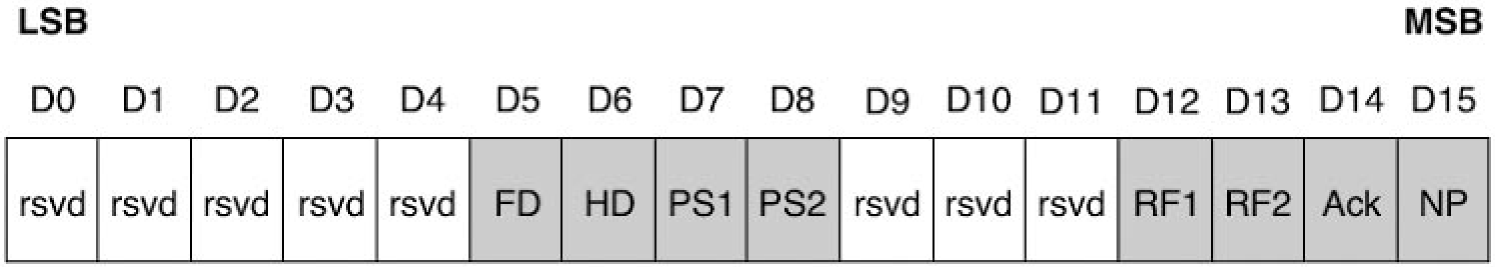
- Next page is a method of transferring additional information during the auto-negotiation process
- The Next Page bit is bit 15 of the base page
- The Next Page bit can only be set to one if the device supports next page exchange and wants to send a next page

LSB MSB

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

Reserved Bits

- The Reserved Bits are bits 0-4 and 9-11 of the base page
- These bits are reserved for future technologies to use
- All reserved bits must be set to zero until a new technology is implemented



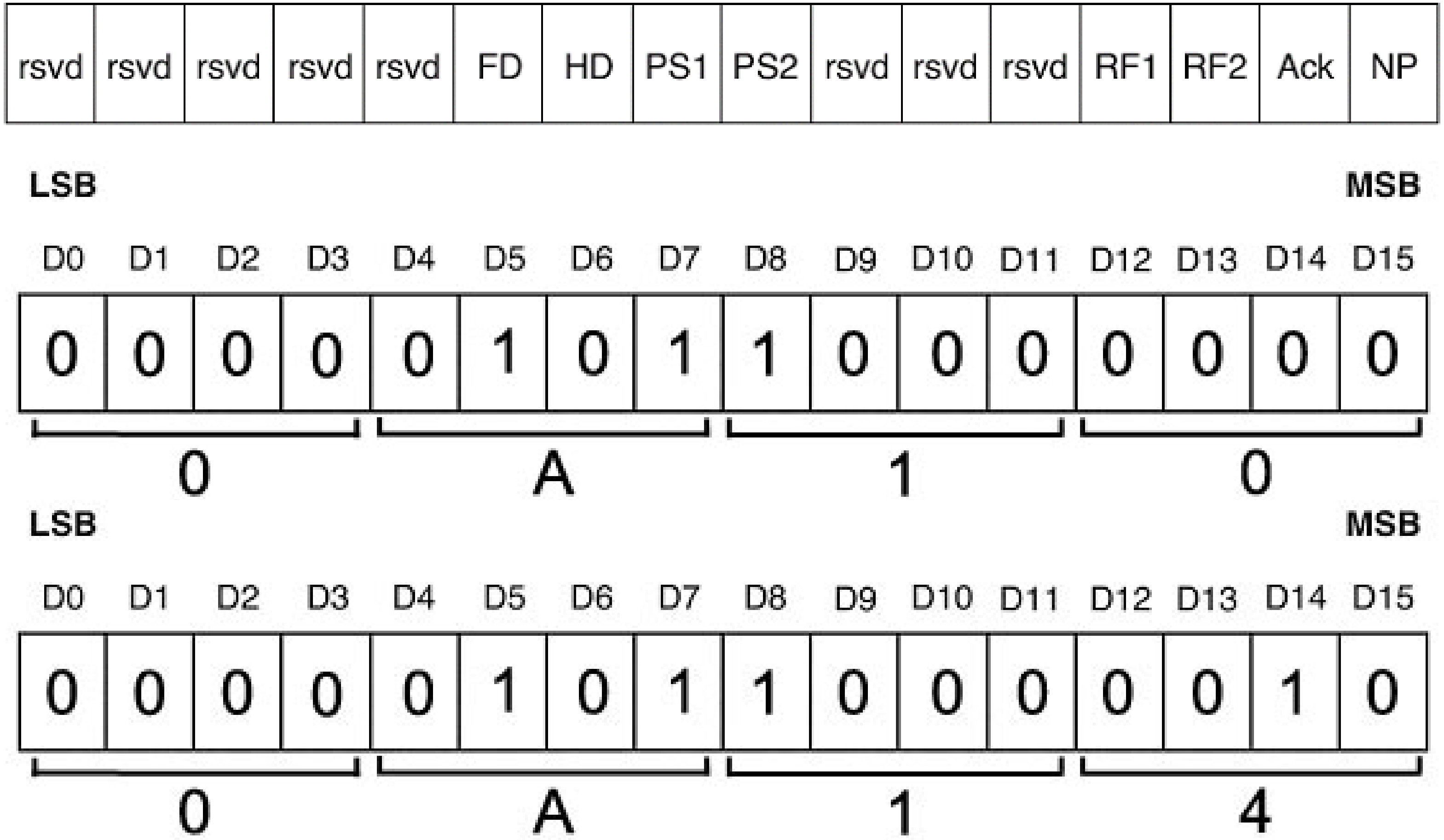
LSB															MSB
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

Config_reg in Hex

- To make it easier to specify a device’s abilities, the config_reg can be encoded to hexadecimal
- This makes discussing a device’s abilities much easier
- Instead of saying the device supports full duplex, symmetric pause, and asymmetric pause; one can say the device’s abilities are 01a0

Config_Reg in Hex

- Here the mapping of 01a0 and 41a0 is illustrated



rsvd	rsvd	rsvd	rsvd	rsvd	FD	HD	PS1	PS2	rsvd	rsvd	rsvd	RF1	RF2	Ack	NP

LSB MSB

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

0 A 1 0

LSB MSB

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

0 A 1 4

Breaklink

- A config_reg of all zeros is known as breaklink
- Breaklink is used to restart auto-negotiation
- A device responds to any error received during auto-negotiation by transmitting breaklink and restarting the entire auto-negotiation process

How Auto-Negotiation Works

- The four key functions in auto-negotiation are ability_match, acknowledge_match, consistency_match, and Idle_match
- The only timer for clause 37 auto-negotiation is link_timer

Ability_Match

- Ability_match is responsible for ensuring that the device receives a reliable config_reg from the link partner
- Ability_match = TRUE when the device has received three consecutive and consistent (ignoring the ACK bit) config_regs from the link partner
- The device sets its ACK bit to one when ability_match = TRUE

Acknowledge_Match

- Acknowledge_match is similar to ability_match, except that the device must receive 3 consecutive and consistent config_regs with the ACK bit set to one

Consistency_Match

- Consistency_match ensures that the link partner transmitted the same abilities before and after it set its ACK bit
- Consistency_match compares the abilities that were used to set ability_match = TRUE to the abilities that were used to set acknowledge_match = TRUE
- All bits are examined except for the ACK bit

Idle_Match

- Idle_match is also similar to ability_match, except that it counts the Idle codes received
- The device cannot establish a link until it has successfully received 3 Idle codes from its link partner

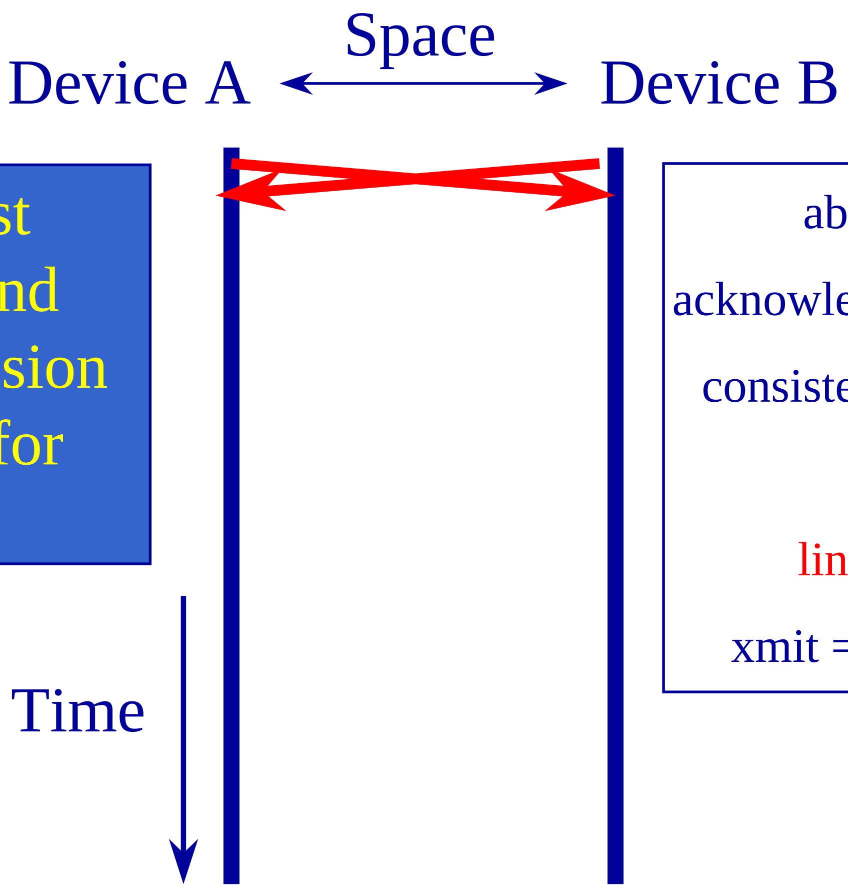
Link_Timer

- Link_timer is the only timer used in clause 37 auto-negotiation
- It is defined as 10 ms with a tolerance of +10 ms
- Link_timer is used throughout the auto-negotiation process to help verify that the link partner has enough time to see everything that happens before the device transitions to the next state

Basic Flow of Auto-Negotiation

- The basic flow of auto-negotiation is as follows:
  - Breaklink
  - Abilities
  - Abilities with the ACK bit set
  - Idle
  - PCS has control

Detailed Flow of Auto-Negotiation



Device A just powered up and begins transmission of breaklink for link_timer

ability_match = FALSE

acknowledge_match = FALSE

consistency_match = FALSE

Idle_match = FALSE

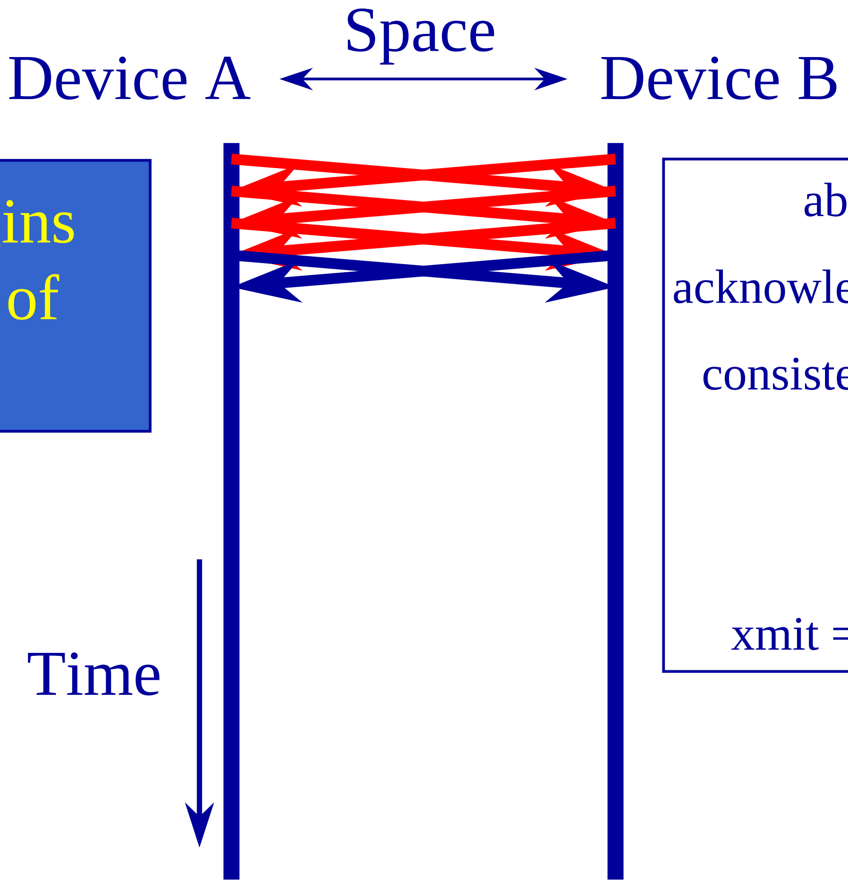
link_timer = STARTED

xmit = CONFIGURATION

<u>Legend</u>

0000	01a0
41a0	Idle
Data	

Detailed Flow of Auto-Negotiation



Device A begins transmission of abilities

ability_match = FALSE

acknowledge_match = FALSE

consistency_match = FALSE

Idle_match = FALSE

link_timer = DONE

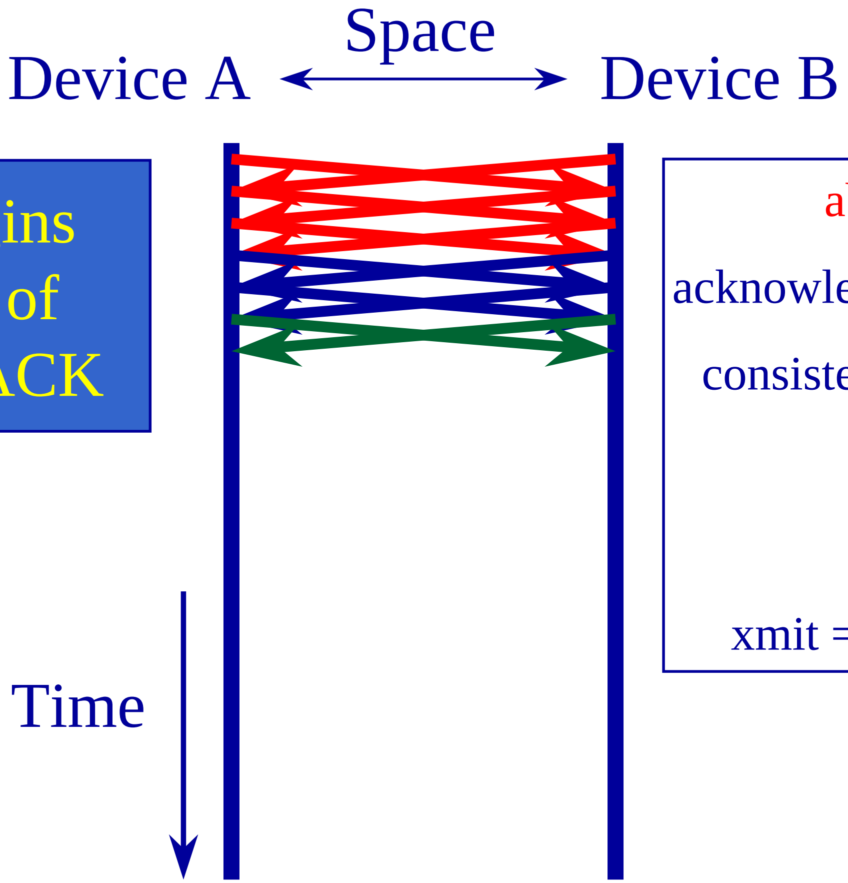
xmit = CONFIGURATION

Legend

0000	01a0
41a0	Idle
Data	

Time

Detailed Flow of Auto-Negotiation



Device A begins transmission of abilities with ACK

ability_match = TRUE

acknowledge_match = FALSE

consistency_match = FALSE

Idle_match = FALSE

link_timer = DONE

xmit = CONFIGURATION

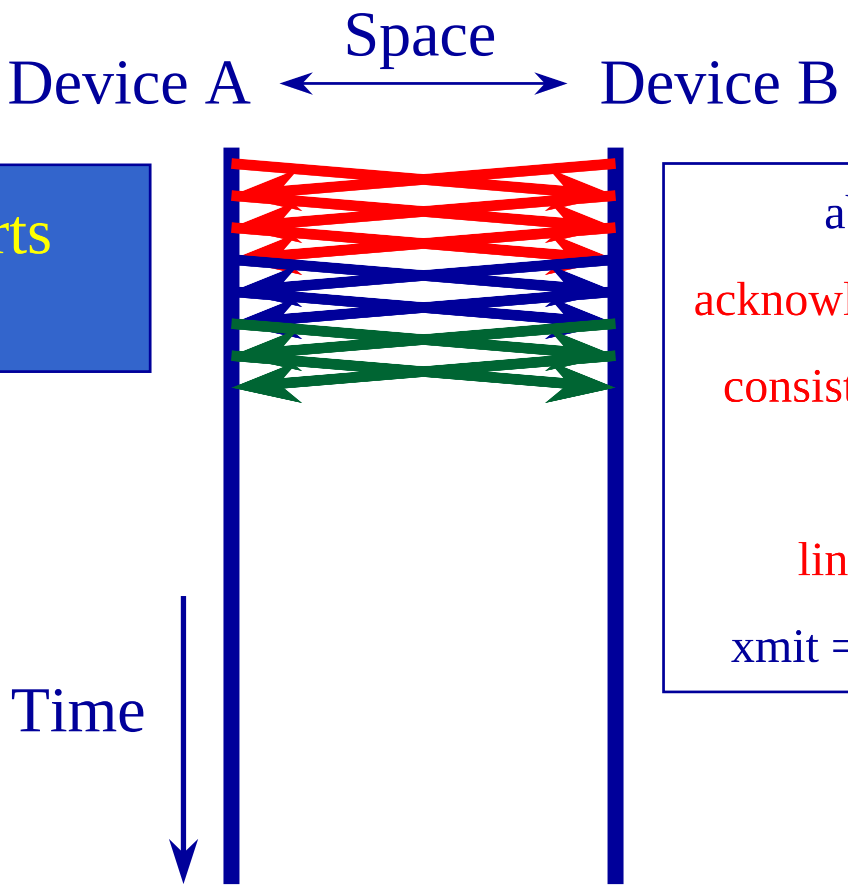
Legend

0000 01a0

41a0 Idle

Data

Detailed Flow of Auto-Negotiation



Device A starts link_timer

ability_match = TRUE

acknowledge_match = TRUE

consistency_match = TRUE

Idle_match = FALSE

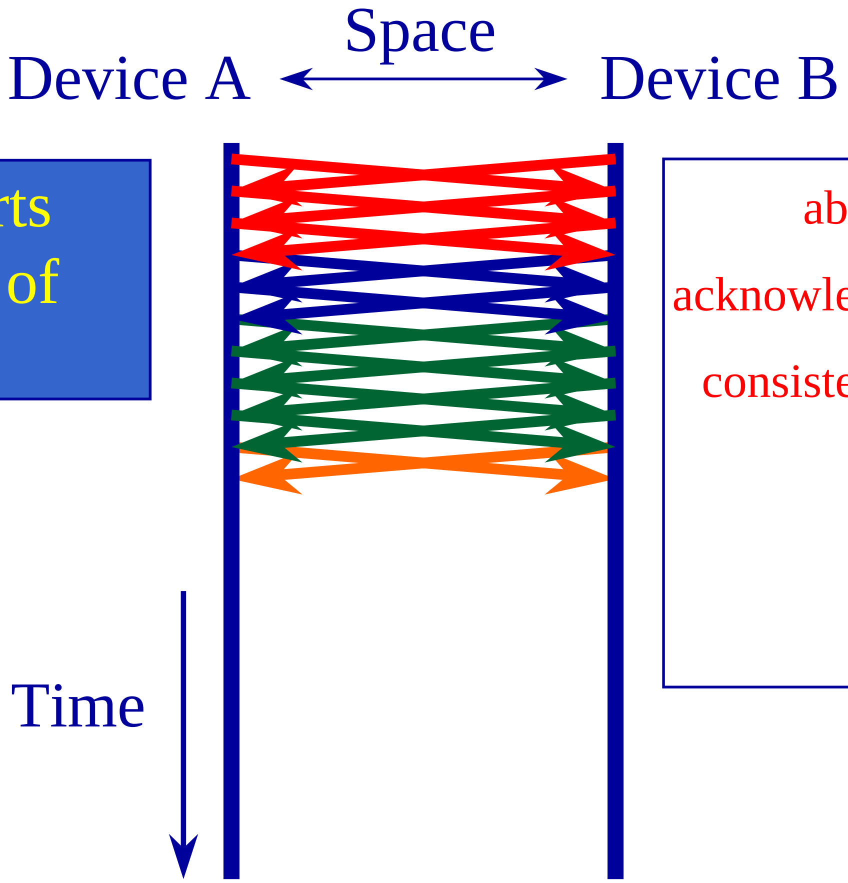
link_timer = STARTED

xmit = CONFIGURATION

Legend

0000	01a0
41a0	Idle
Data	

Detailed Flow of Auto-Negotiation



Device A starts transmission of Idle

ability_match = FALSE

acknowledge_match = FALSE

consistency_match = FALSE

Idle_match = FALSE

link_timer = DONE

xmit = IDLE

Legend

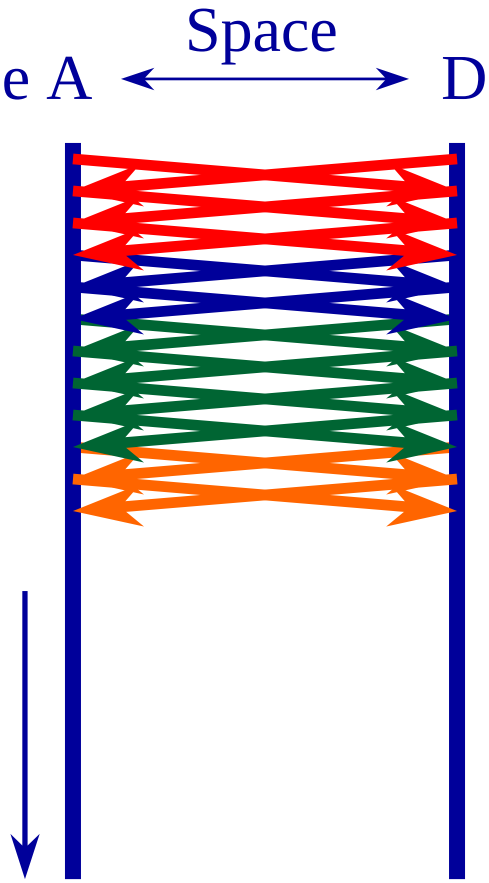
0000 01a0

41a0 Idle

Data

Detailed Flow of Auto-Negotiation

Device A



Device B

Device A starts link_timer

ability_match = FALSE

acknowledge_match = FALSE

consistency_match = FALSE

Idle_match = TRUE

link_timer = STARTED

xmit = IDLE

Legend

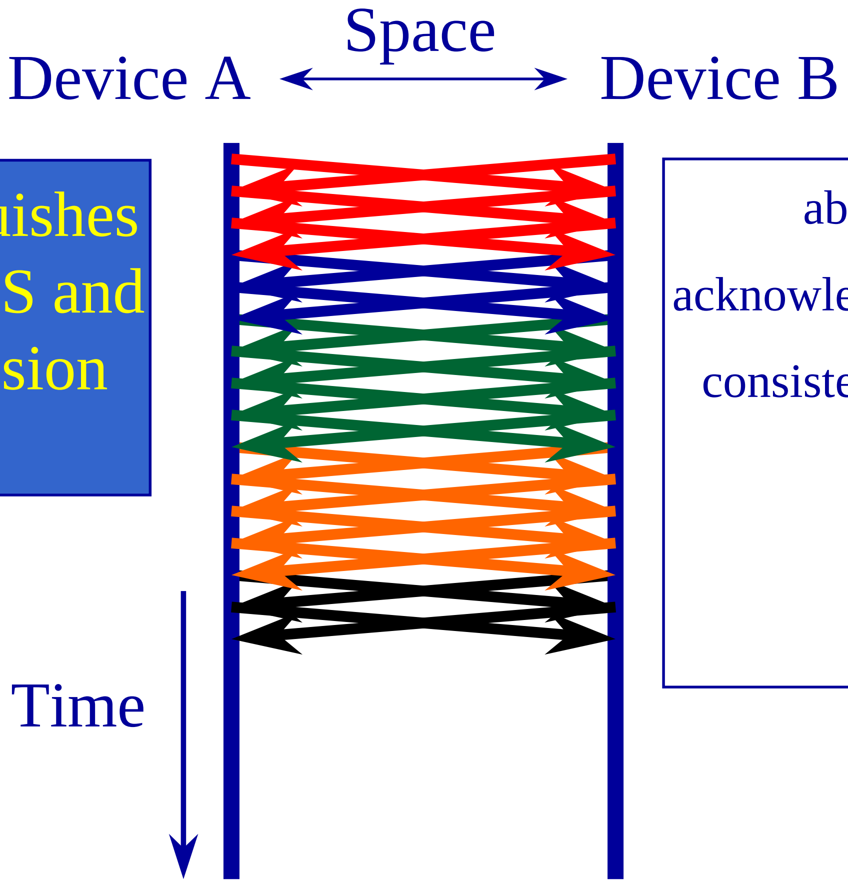
0000 01a0

41a0 Idle

Data

Time

Detailed Flow of Auto-Negotiation



Device A relinquishes control to the PCS and begins transmission of data

Legend

0000 01a0

41a0 Idle

Data

ability_match = FALSE

acknowledge_match = FALSE

consistency_match = FALSE

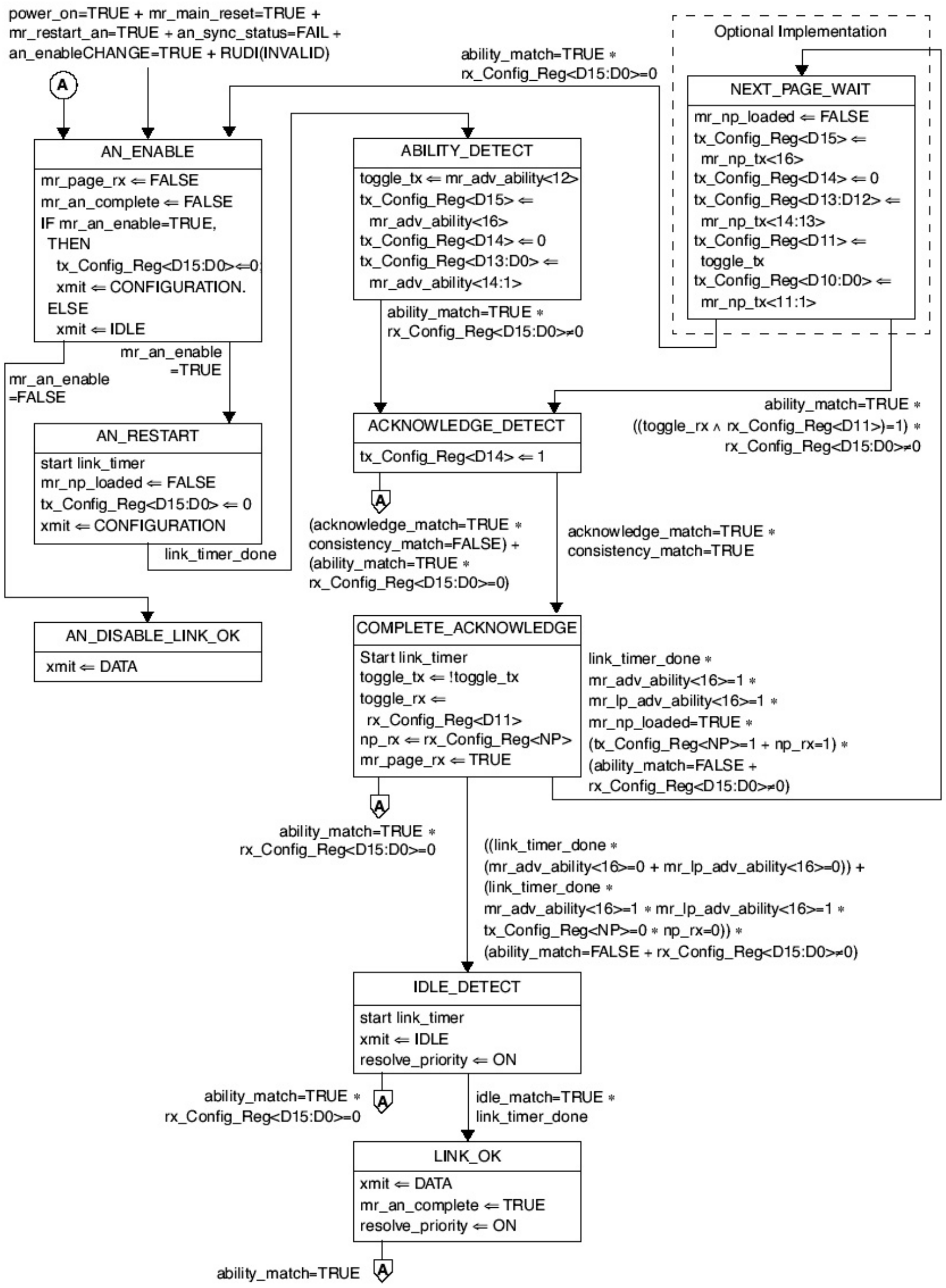
Idle_match = TRUE

link_timer = DONE

xmit = DATA

State Diagram

- The Auto-Negotiation State Diagram is the basis of clause 37 auto-negotiation



power_on=TRUE + mr_main_reset=TRUE +
mr_restart_an=TRUE + an_sync_status=FAIL +
an_enableCHANGE=TRUE + RUDI(INVALID)

A

AN_ENABLE
mr_page_rx ⇐ FALSE
mr_an_complete ⇐ FALSE
IF mr_an_enable=TRUE,
THEN
tx_Config_Reg<D15:D0>⇐0
xmit ⇐ CONFIGURATION.
ELSE
xmit ⇐ IDLE

mr_an_enable
=FALSE

mr_an_enable
=TRUE

AN_RESTART
start link_timer
mr_np_loaded ⇐ FALSE
tx_Config_Reg<D15:D0> ⇐ 0
xmit ⇐ CONFIGURATION

link_timer_done

AN_DISABLE_LINK_OK
xmit ⇐ DATA

ability_match=TRUE ∗
rx_Config_Reg<D15:D0>=0

ABILITY_DETECT
toggle_tx ⇐ mr_adv_ability<12>
tx_Config_Reg<D15> ⇐
mr_adv_ability<16>
tx_Config_Reg<D14> ⇐ 0
tx_Config_Reg<D13:D0> ⇐
mr_adv_ability<14:1>

ability_match=TRUE ∗
rx_Config_Reg<D15:D0>≠0

Optional Implementation

NEXT_PAGE_WAIT
mr_np_loaded ⇐ FALSE
tx_Config_Reg<D15> ⇐
mr_np_tx<16>
tx_Config_Reg<D14> ⇐ 0
tx_Config_Reg<D13:D12> ⇐
mr_np_tx<14:13>
tx_Config_Reg<D11> ⇐
toggle_tx
tx_Config_Reg<D10:D0> ⇐
mr_np_tx<11:1>

ability_match=TRUE ∗
((toggle_rx ∧ rx_Config_Reg<D11>)=1) ∗
rx_Config_Reg<D15:D0>≠0

ACKNOWLEDGE_DETECT
tx_Config_Reg<D14> ⇐ 1

A

(acknowledge_match=TRUE ∗
consistency_match=FALSE) +
(ability_match=TRUE ∗
rx_Config_Reg<D15:D0>=0)

acknowledge_match=TRUE ∗
consistency_match=TRUE

COMPLETE_ACKNOWLEDGE
Start link_timer
toggle_tx ⇐ !toggle_tx
toggle_rx ⇐
rx_Config_Reg<D11>
np_rx ⇐ rx_Config_Reg<NP>
mr_page_rx ⇐ TRUE

link_timer_done ∗
mr_adv_ability<16>=1 ∗
mr_lp_adv_ability<16>=1 ∗
mr_np_loaded=TRUE ∗
(tx_Config_Reg<NP>=1 + np_rx=1) ∗
(ability_match=FALSE +
rx_Config_Reg<D15:D0>≠0)

A

ability_match=TRUE ∗
rx_Config_Reg<D15:D0>=0

((link_timer_done ∗
(mr_adv_ability<16>=0 + mr_lp_adv_ability<16>=0)) +
(link_timer_done ∗
mr_adv_ability<16>=1 ∗ mr_lp_adv_ability<16>=1 ∗
tx_Config_Reg<NP>=0 ∗ np_rx=0)) ∗
(ability_match=FALSE + rx_Config_Reg<D15:D0>≠0)

IDLE_DETECT
start link_timer
xmit ⇐ IDLE
resolve_priority ⇐ ON

ability_match=TRUE ∗
rx_Config_Reg<D15:D0>=0

A

idle_match=TRUE ∗
link_timer_done

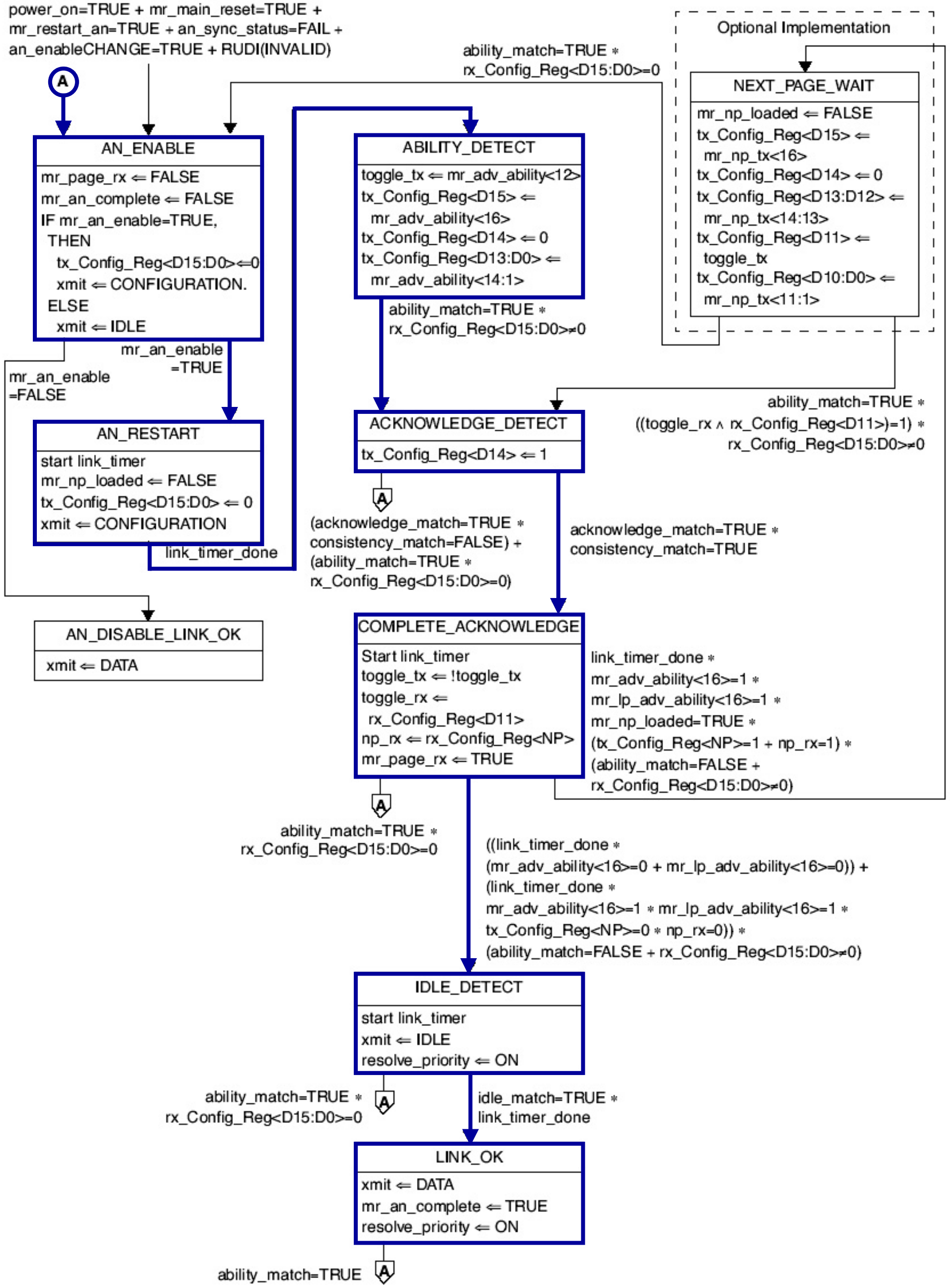
LINK_OK
xmit ⇐ DATA
mr_an_complete ⇐ TRUE
resolve_priority ⇐ ON

ability_match=TRUE

A

State Diagram

- The blue arrows are typically followed when two link partners auto-negotiate
- The example shown here does not include next pages



Clause 37 Auto-Negotiation

AN_ENABLE

This is the starting point for Auto-Negotiation. Every time a device restarts auto-negotiation it enters this state.

The device transmits breaklink.

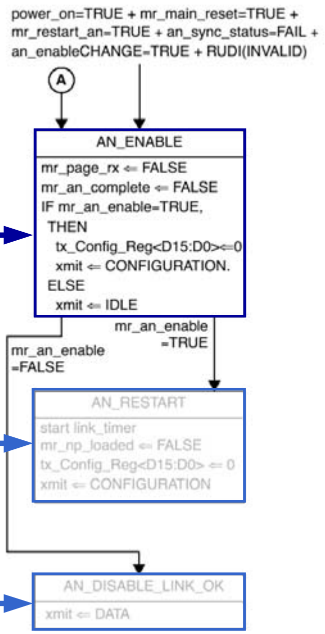
Exit Cases

mr_an_enable = TRUE

- Exit if Auto-Negotiation is enabled

mr_an_enable = FALSE

- Exit if Auto-Negotiation is disabled. Here the device will transmit Idle and relinquish control to the PCS.



AN_RESTART

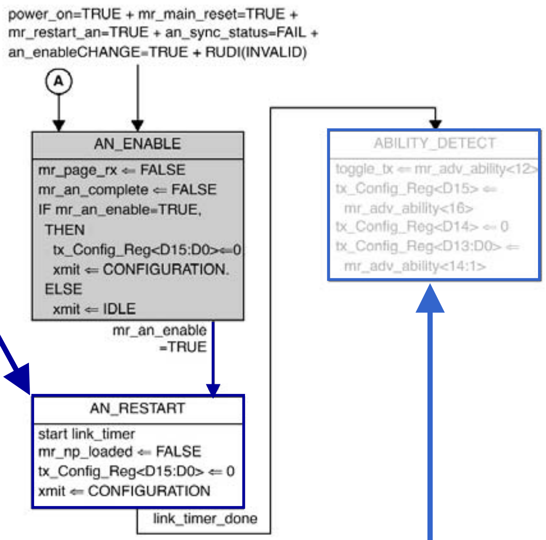
This state ensures that the link partner sees that the device is restarting Auto-Negotiation

The device transmits breaklink for link_timer

Exit Cases

link_timer_done

- This is a status bit that indicates that the device has been in this state for a duration of link_timer. After link_timer is done, the device transitions to ABILITY_DETECT



ABILITY_DETECT

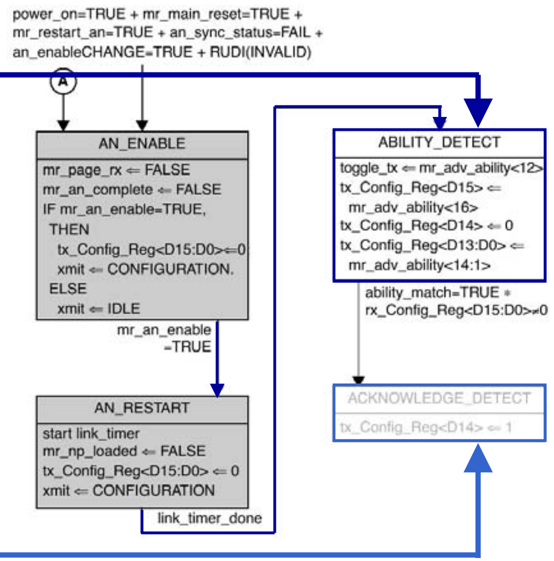
This state ensures that the device is receiving proper abilities from the link partner

The device transmits its own abilities

Exit Cases

ability_match = TRUE and rx_Config_Reg <D15:D0> ≠ 0

- Exit if the device has received 3 consecutive and consistent Config_Regs that are not breaklink



ACKNOWLEDGE_DETECT

This state ensures that the link partner has acknowledged that it has seen the devices abilities

The device transmits abilities with the ACK bit set to one.

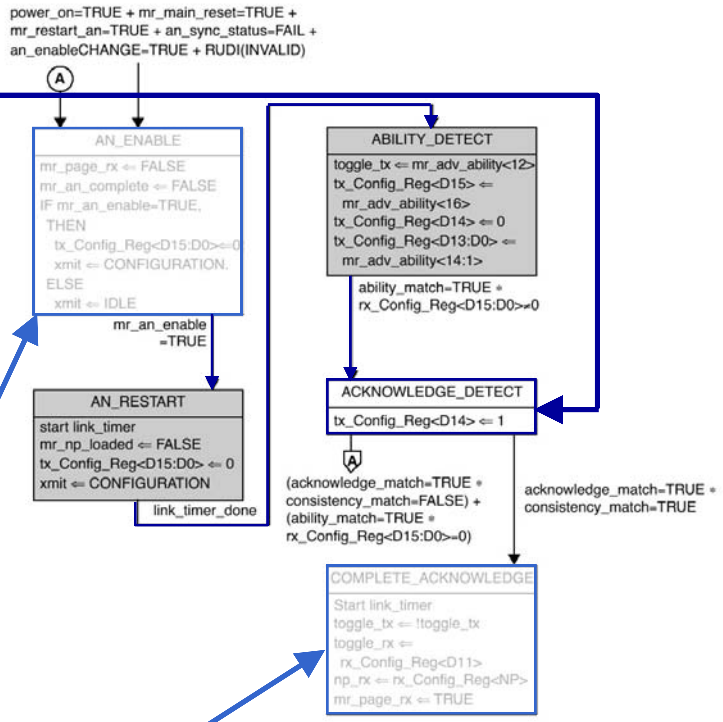
Exit Cases

(acknowledge_match = TRUE and consistency_match = FALSE) or (ability_match = TRUE and rx_Config_Reg <D15:D0> = 0)

- Exit if the device receives 3 consecutive config_regs encoded with breaklink or if the abilities received in ACKNOWLEDGE_DETECT are different than the abilities that were received in ABILITY_DETECT

acknowledge_match = TRUE and consistency_match = TRUE

- Exit if the abilities received in ACKNOWLEDGE_DETECT are the same as the abilities that were received in ABILITY_DETECT



COMPLETE_ACKNOWLEDGE

This state ensures that the link partner has time to see that the device has set its ACK bit to one

The device transmits abilities with the ACK bit set to one.

Exit Cases

ability_match = TRUE and rx_Config_Reg <D15:D0> = 0

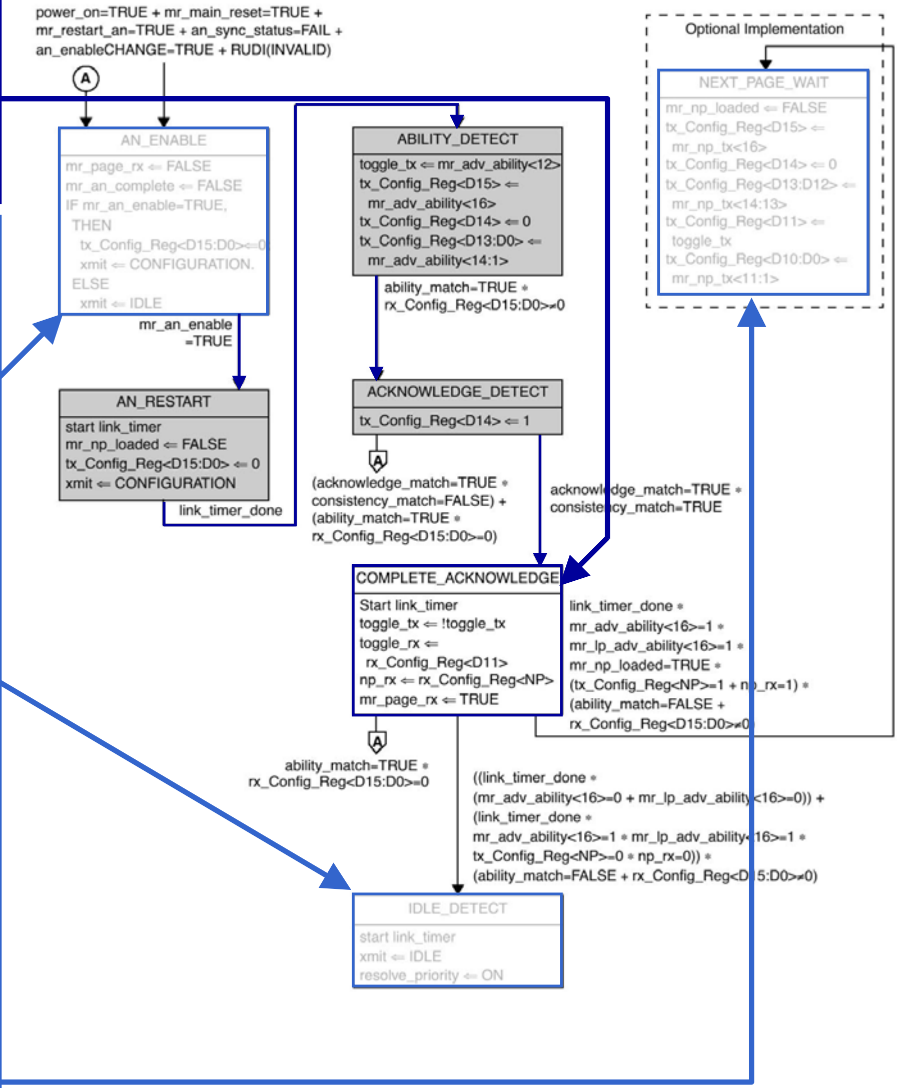
- Exit if the device receives 3 consecutive config_regs encoded with breaklink

((link_timer = DONE and (mr_adv_ability<16>=0 or mr_lp_adv_ability<16>=0)) or (link_timer = DONE and mr_adv_ability<16>=1 and mr_lp_adv_ability<16>=1 and tx_Config_Reg<NP>=0 and np_rx=0)) and (ability_match = FALSE or rx_Config_Reg<D15:D0> ≠ 0)

- Exit if link_timer is done and one of the devices doesn't support next page exchange or both devices are done transmitting next pages and the device isn't receiving breaklink

link_timer = DONE and mr_adv_ability<16>=1 and mr_lp_adv_ability<16>=1 and mr_np_loaded = TRUE and (tx_Config_Reg<NP>=1 or np_rx=1) and (ability_match = FALSE or rx_Config_Reg<D15:D0> ≠ 0)

- Exit if link_timer is done and both devices support next page exchange and at least one device still has next pages to transmit and the device isn't receiving breaklink



IDLE_DETECT

This state ensures that the link partner is ready to establish a link before the device relinquishes control to the PCS

The device transmits Idle

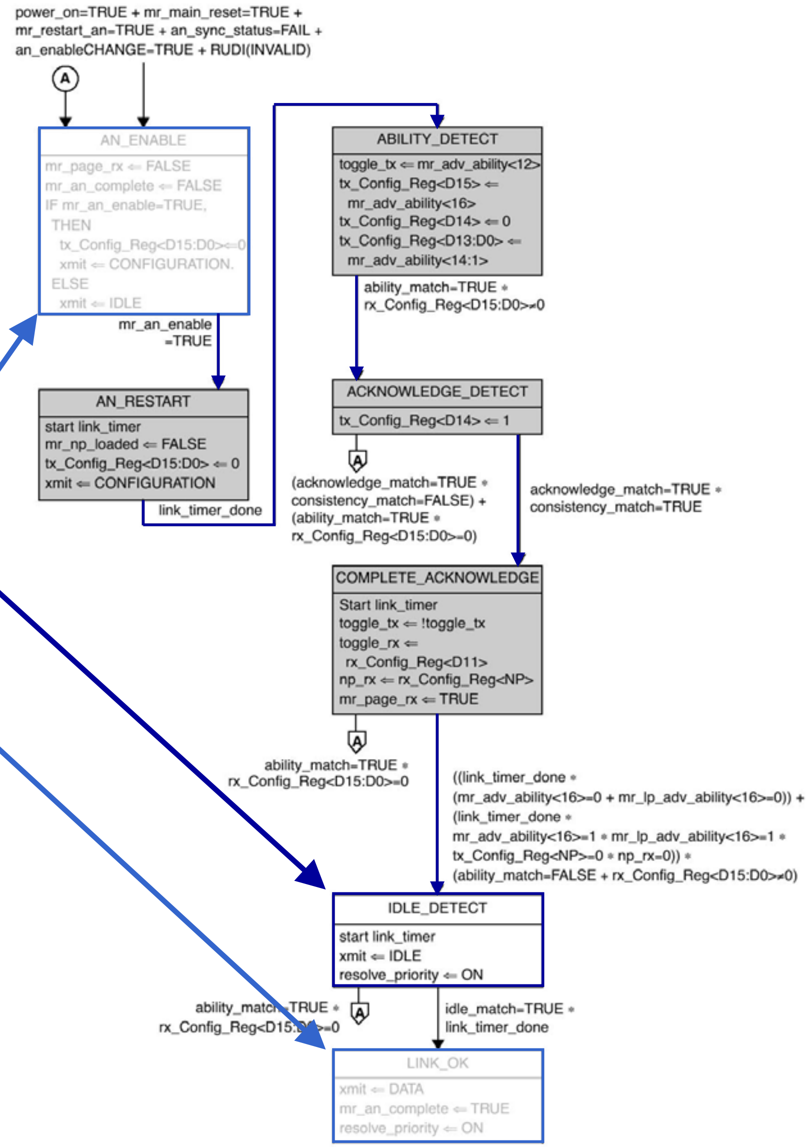
Exit Cases

ability_match = TRUE and rx_Config_Reg <D15:D0> = 0

- Exit if the device receives 3 consecutive config_regs encoded with breaklink

idle_match = TRUE and link_timer_done

- Exit if the device has received 3 consecutive Idle codes and link_timer is done



LINK_OK

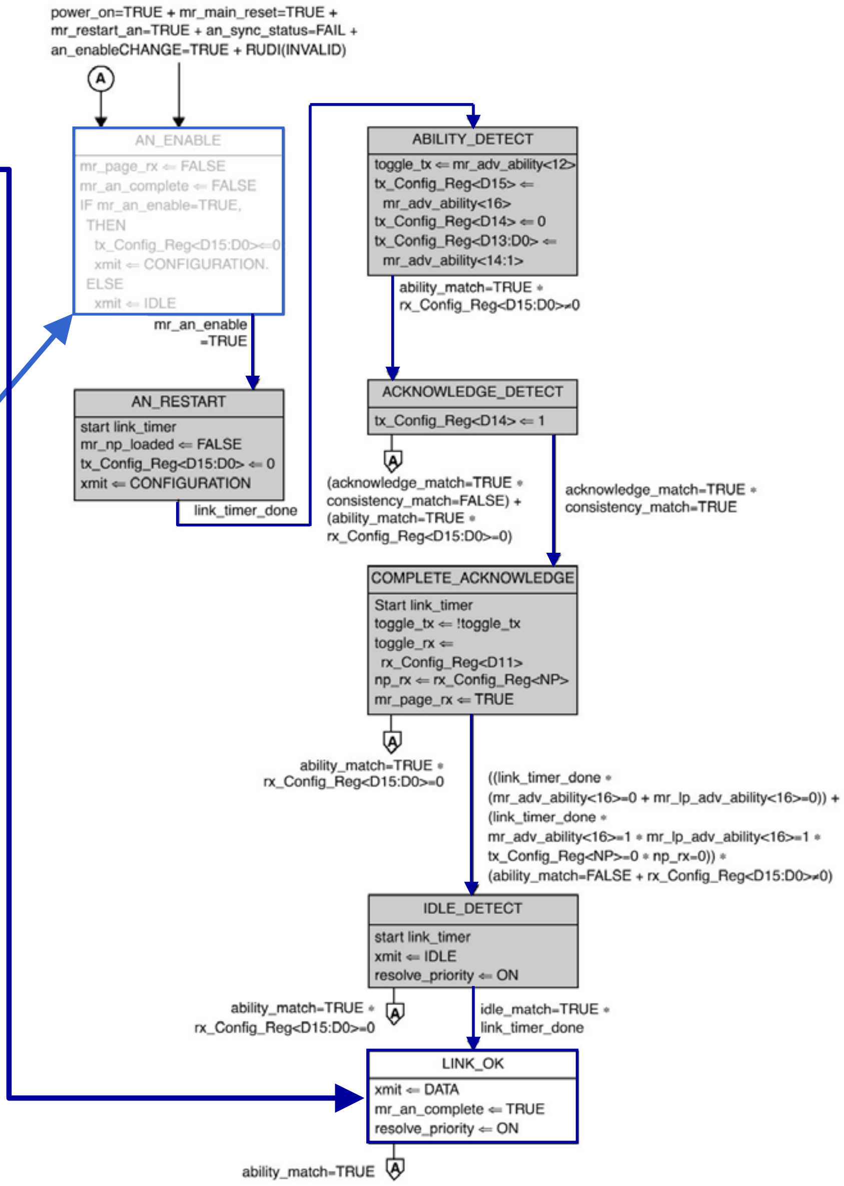
Auto-Negotiation is now complete and the PCS now has control

The PCS determines what to transmit

Exit Cases

ability_match = TRUE

- Exit if the device has received 3 consecutive and consistent Config_Regs



power_on=TRUE + mr_main_reset=TRUE +
mr_restart_an=TRUE + an_sync_status=FAIL +
an_enableCHANGE=TRUE + RUDI(INVALID)

A

AN_ENABLE
mr_page_rx ⇐ FALSE
mr_an_complete ⇐ FALSE
IF mr_an_enable=TRUE,
 THEN
  tx_Config_Reg<D15:D0>⇐0
  xmit ⇐ CONFIGURATION.
 ELSE
  xmit ⇐ IDLE

mr_an_enable
=TRUE

AN_RESTART
start link_timer
mr_np_loaded ⇐ FALSE
tx_Config_Reg<D15:D0> ⇐ 0
xmit ⇐ CONFIGURATION

link_timer_done

ABILITY_DETECT
toggle_tx ⇐ mr_adv_ability<12>
tx_Config_Reg<D15> ⇐
 mr_adv_ability<16>
tx_Config_Reg<D14> ⇐ 0
tx_Config_Reg<D13:D0> ⇐
 mr_adv_ability<14:1>

ability_match=TRUE ∗
rx_Config_Reg<D15:D0>≠0

ACKNOWLEDGE_DETECT
tx_Config_Reg<D14> ⇐ 1

A

(acknowledge_match=TRUE ∗
consistency_match=FALSE) +
(ability_match=TRUE ∗
rx_Config_Reg<D15:D0>=0)

acknowledge_match=TRUE ∗
consistency_match=TRUE

COMPLETE_ACKNOWLEDGE
Start link_timer
toggle_tx ⇐ !toggle_tx
toggle_rx ⇐
 rx_Config_Reg<D11>
np_rx ⇐ rx_Config_Reg<NP>
mr_page_rx ⇐ TRUE

A

ability_match=TRUE ∗
rx_Config_Reg<D15:D0>=0

((link_timer_done ∗
(mr_adv_ability<16>=0 + mr_lp_adv_ability<16>=0)) +
(link_timer_done ∗
mr_adv_ability<16>=1 ∗ mr_lp_adv_ability<16>=1 ∗
tx_Config_Reg<NP>=0 ∗ np_rx=0)) ∗
(ability_match=FALSE + rx_Config_Reg<D15:D0>≠0)

IDLE_DETECT
start link_timer
xmit ⇐ IDLE
resolve_priority ⇐ ON

ability_match=TRUE ∗
rx_Config_Reg<D15:D0>=0

A

idle_match=TRUE ∗
link_timer_done

LINK_OK
xmit ⇐ DATA
mr_an_complete ⇐ TRUE
resolve_priority ⇐ ON

ability_match=TRUE

A

Next Page Exchange

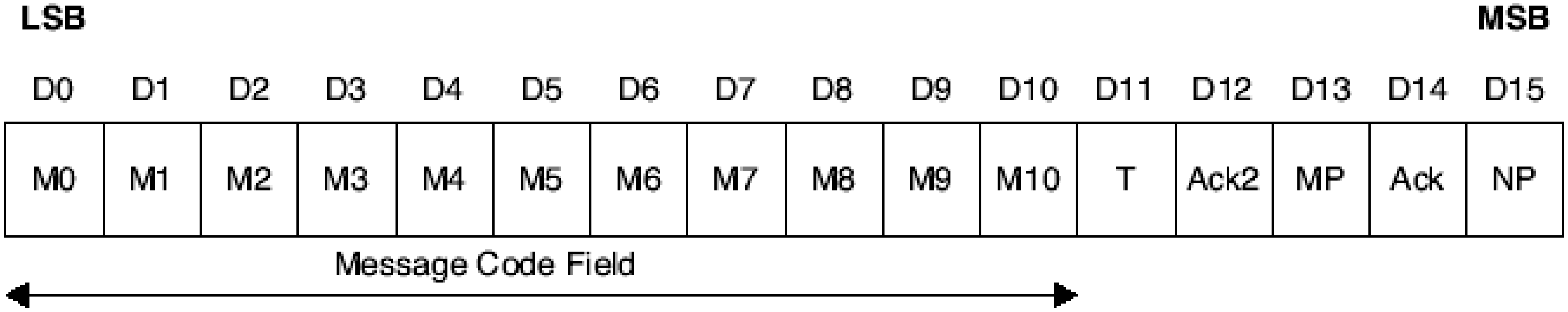
- As shown earlier, the next page capability (desire) is advertised in bit 15 of the base page
- If the bit is set to one, the device supports next page exchange and wants to transmit a next page. If the bit is set to zero, either the device doesn't support next page exchange or doesn't want to transmit one.

Why Next Page?

- Next page exchange allows two devices to transfer information that is not included in the base page
- This information could include additional modes of operation or remote fault information
- There are two kinds of next pages:
  - Message pages
  - Unformatted pages

Message Page

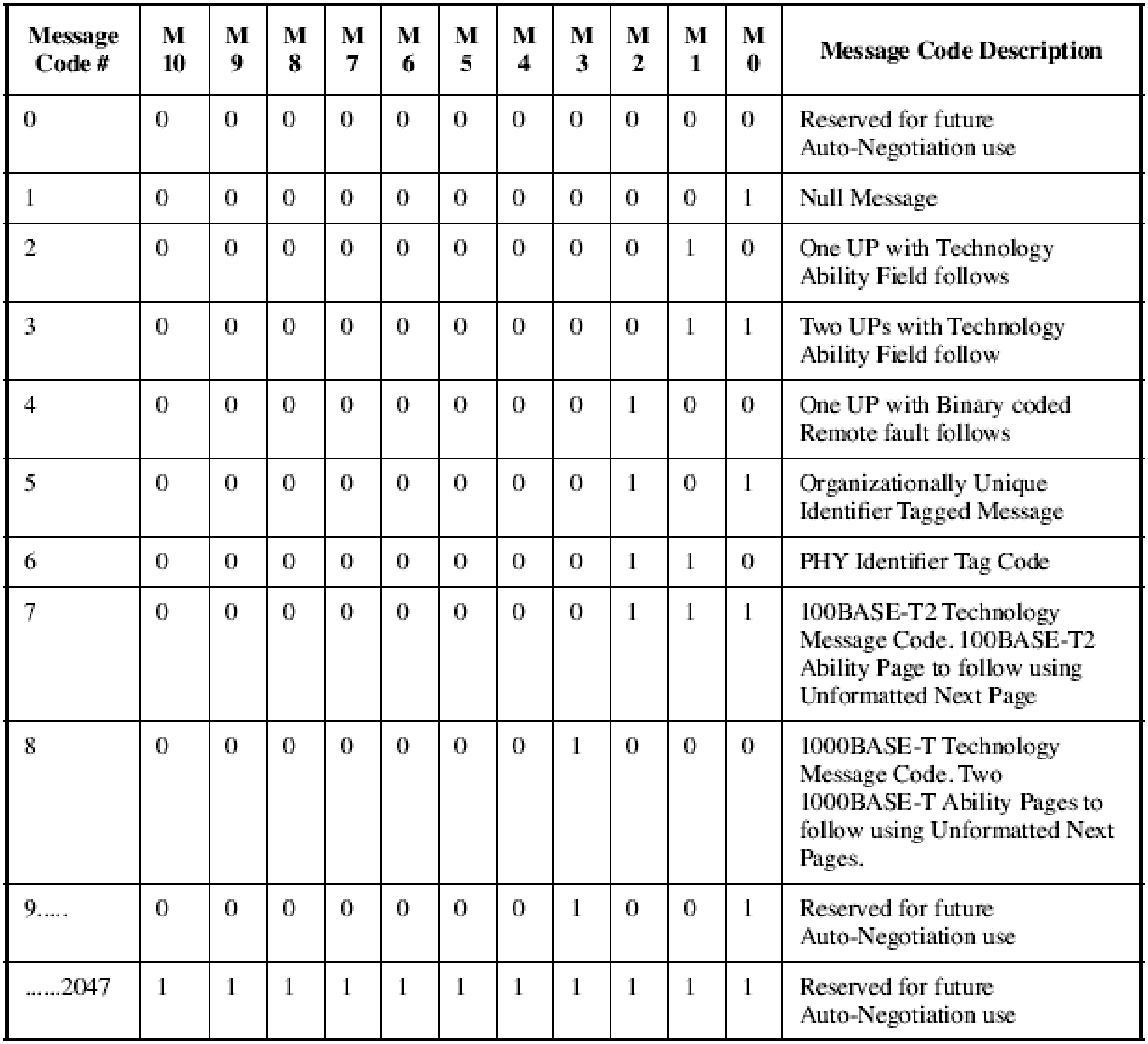
- The message page contains an 11-bit formatted Message Code Field
- The first message page determines how many unformatted pages will follow
- The encoding of the message page is:



LSB … MSB

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	T	Ack2	MP	Ack	NP

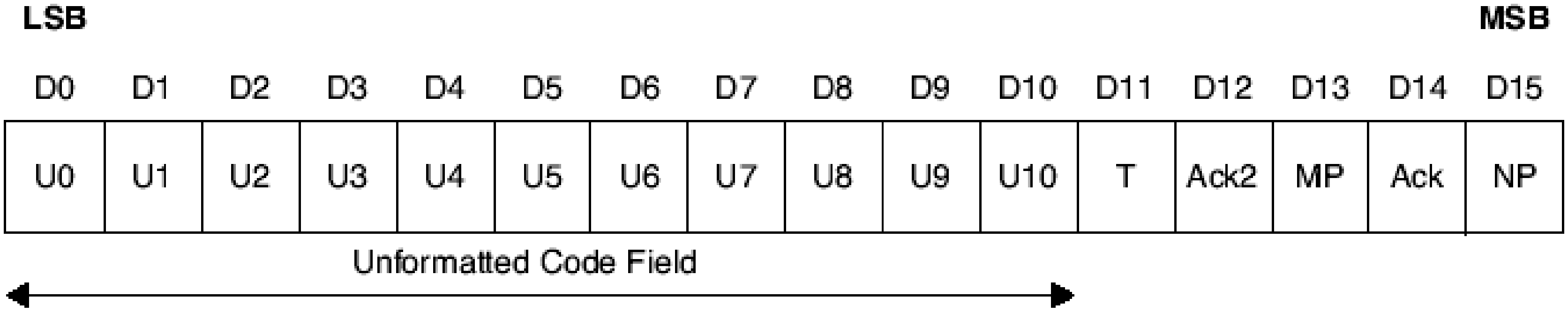
Encoding for the 11-bit message code at the beginning of a message page



Message Code #	M 10	M 9	M 8	M 7	M 6	M 5	M 4	M 3	M 2	M 1	M 0	Message Code Description
0	0	0	0	0	0	0	0	0	0	0	0	Reserved for future Auto-Negotiation use
1	0	0	0	0	0	0	0	0	0	0	1	Null Message
2	0	0	0	0	0	0	0	0	0	1	0	One UP with Technology Ability Field follows
3	0	0	0	0	0	0	0	0	0	1	1	Two UPs with Technology Ability Field follow
4	0	0	0	0	0	0	0	0	1	0	0	One UP with Binary coded Remote fault follows
5	0	0	0	0	0	0	0	0	1	0	1	Organizationally Unique Identifier Tagged Message
6	0	0	0	0	0	0	0	0	1	1	0	PHY Identifier Tag Code
7	0	0	0	0	0	0	0	0	1	1	1	100BASE-T2 Technology Message Code. 100BASE-T2 Ability Page to follow using Unformatted Next Page
8	0	0	0	0	0	0	0	1	0	0	0	1000BASE-T Technology Message Code. Two 1000BASE-T Ability Pages to follow using Unformatted Next Pages.
9.....	0	0	0	0	0	0	0	1	0	0	1	Reserved for future Auto-Negotiation use
......2047	1	1	1	1	1	1	1	1	1	1	1	Reserved for future Auto-Negotiation use

Unformatted Page

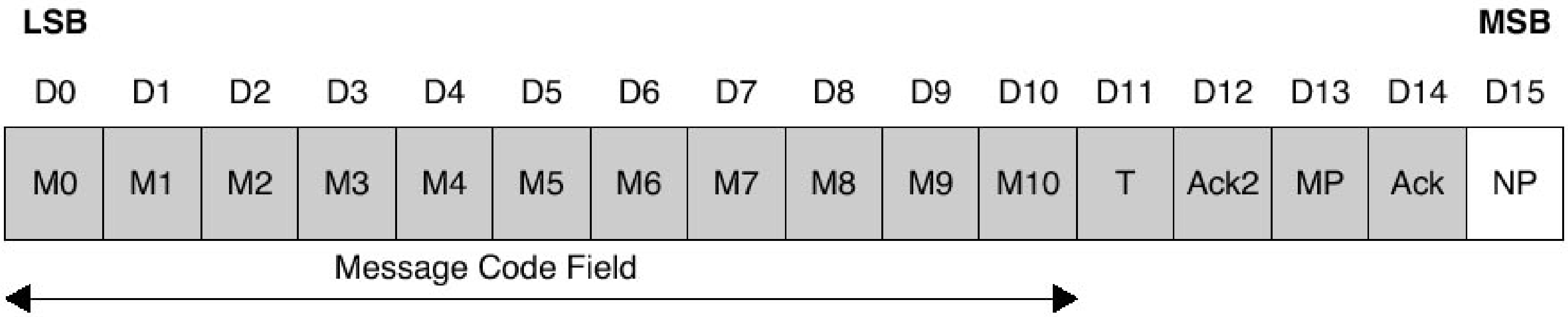
- The unformatted page contains an 11-bit Unformatted Code Field
- The first 11 bits can be any arbitrary value



LSB															MSB
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
U0	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	T	Ack2	MP	Ack	NP

Next Page Bit

- The Next Page Bit is bit 15 of all next pages
- It is used to indicate if the current page is the last page that will be exchanged
- A zero is used to indicate that this is the last page that will be exchanged

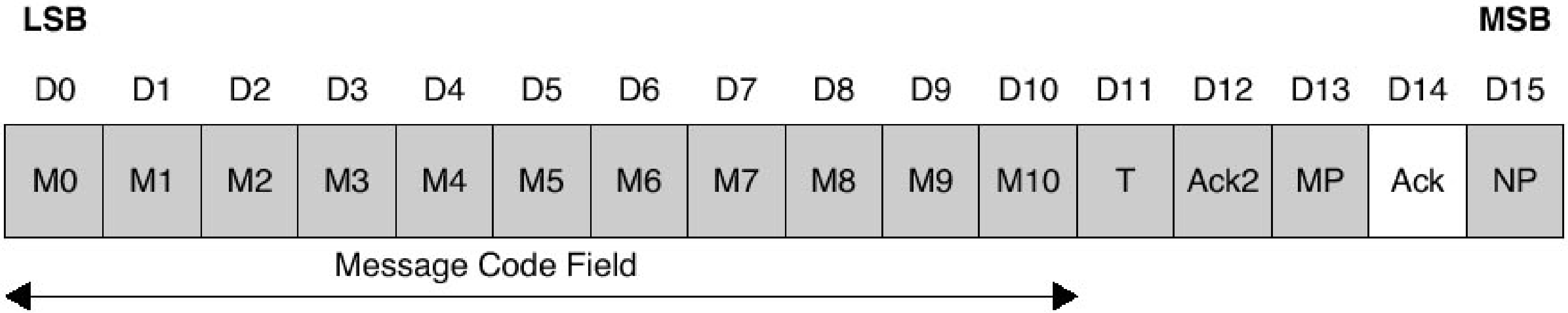


LSB															MSB
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	T	Ack2	MP	Ack	NP

Message Code Field

Acknowledge Bit

- The Acknowledge Bit is bit 14 of all next pages
- This acknowledge bit works like the acknowledge bit in the base page, ability_match must be true before the acknowledge bit is set to one
- This ensures that the link partner is correctly receiving the device’s next page



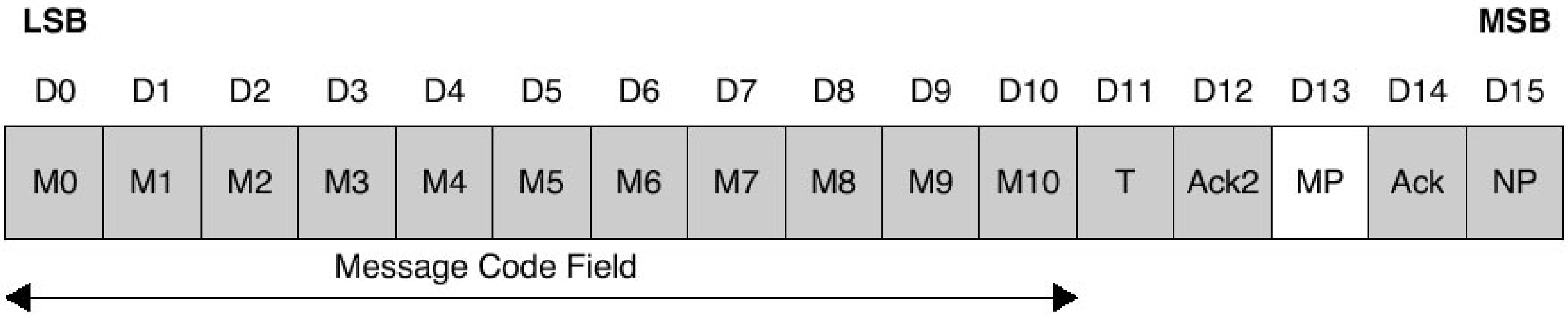
LSB MSB

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	T	Ack2	MP	Ack	NP

Message Code Field

Message Page Bit

- The Message Page Bit is bit 13 of all next pages
- This bit is used to differentiate a message page from an unformatted page
- A one indicates a message page while a zero indicates an unformatted page



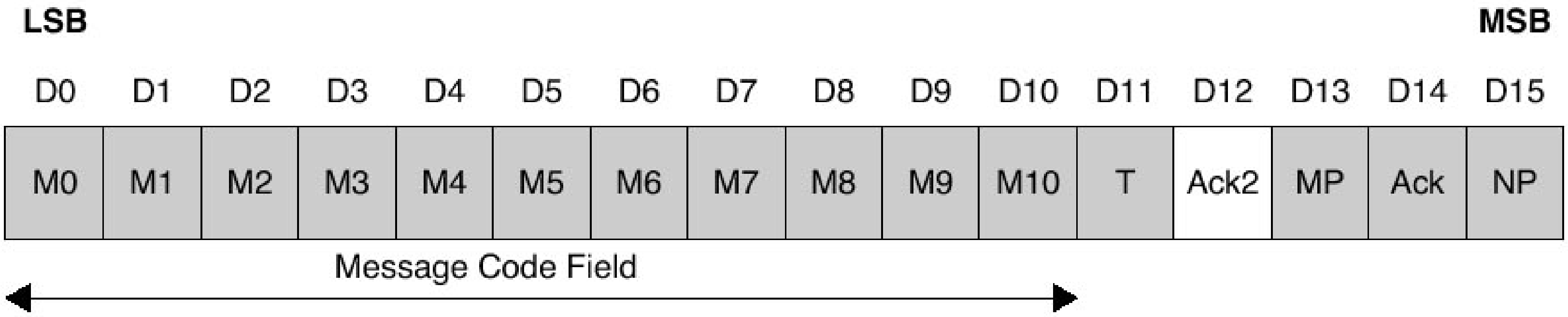
LSB … MSB

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	T	Ack2	MP	Ack	NP

Message Code Field

Acknowledge 2 Bit

- The Acknowledge 2 Bit is bit 12 of all next pages
- It is used to indicate if the link partner is capable of acting on the information that was transferred in the previous pages
- A one indicates that the link partner can comply with the message, while a zero indicates that the link partner cannot comply



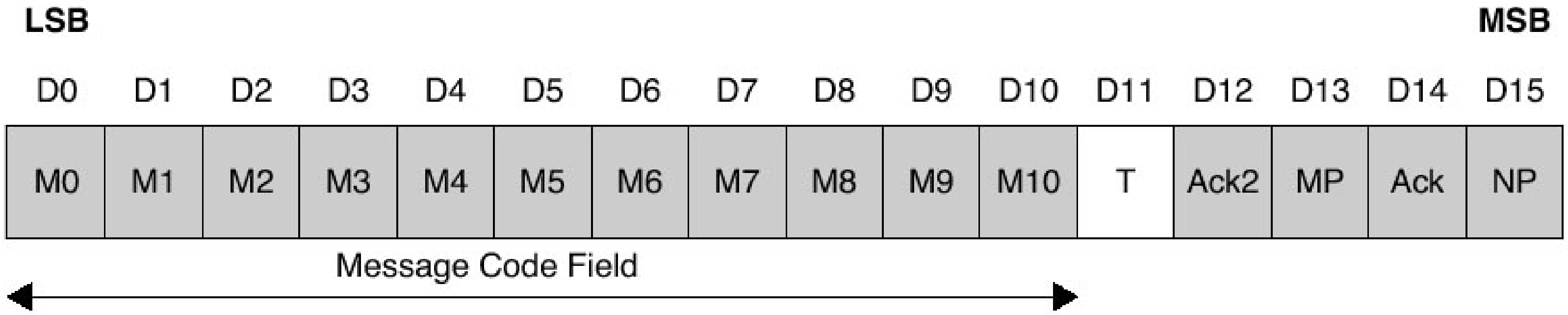
LSB MSB

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	T	Ack2	MP	Ack	NP

Message Code Field

Toggle Bit

- The Toggle bit is bit 11 for all next pages
- The Toggle Bit is used to ensure synchronization of next pages during next page exchange
- The value of the Toggle Bit is flipped for every new next page that is sourced from the device



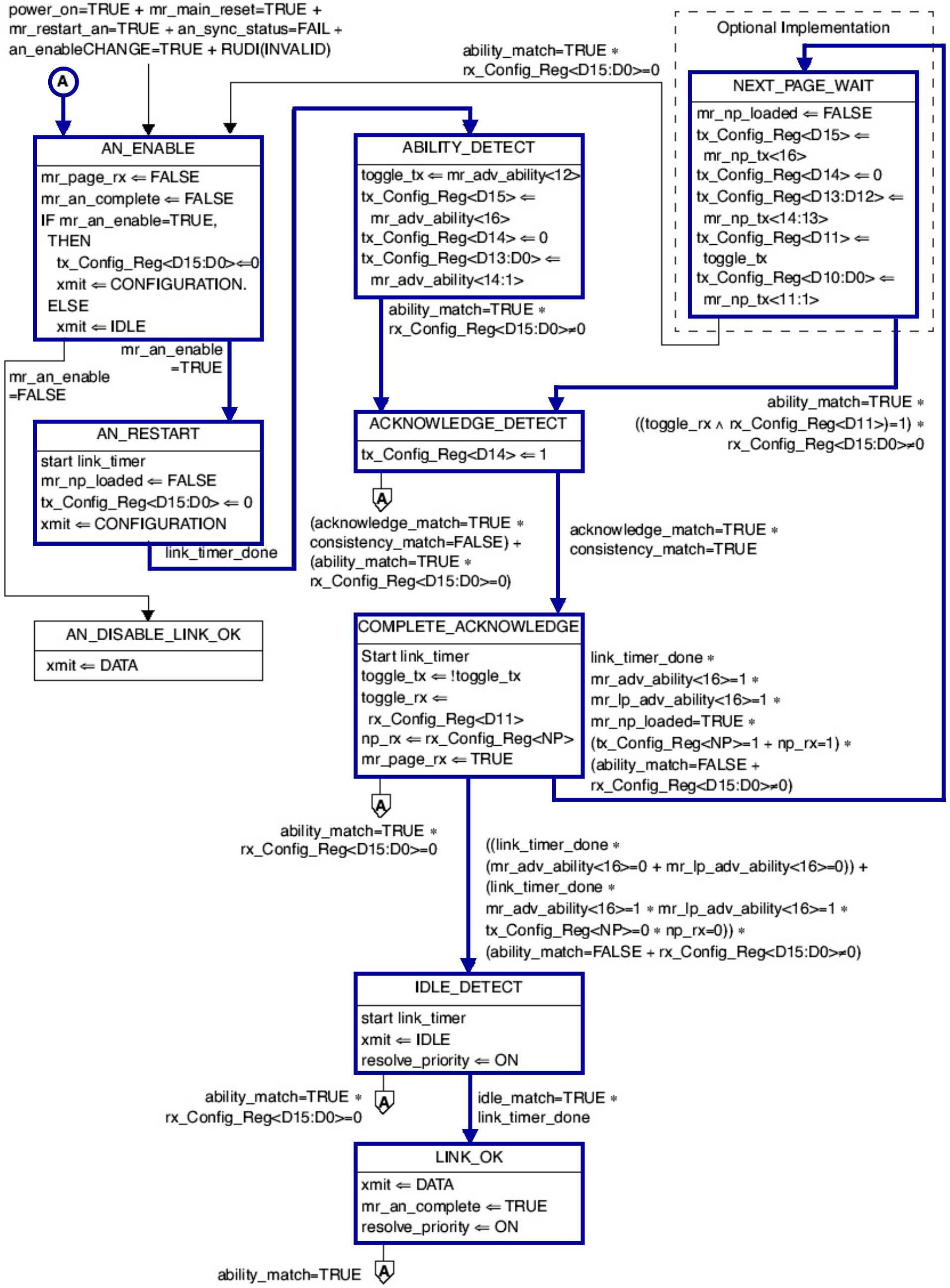
LSB … MSB

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	T	Ack2	MP	Ack	NP

Message Code Field

State Diagram with Next Page

- The blue lines show how the device flows through the state diagram with next page exchange
- The loop consisting of the ACKNOWLEDGE_DETECT, COMPLETE_ACKNOWLEDGE, and NEXT_PAGE_WAIT states is entered once for each different next page exchanged
- In the following slides, the state transitions for a next page exchange will be examined, beginning in the COMPLETE_ACKNOWLEDGE state.



power_on=TRUE + mr_main_reset=TRUE + mr_restart_an=TRUE + an_sync_status=FAIL + an_enableCHANGE=TRUE + RUDI(INVALID)

AN_ENABLE

mr_page_rx ⇐ FALSE
mr_an_complete ⇐ FALSE
IF mr_an_enable=TRUE,
THEN
tx_Config_Reg<D15:D0>⇐0
xmit ⇐ CONFIGURATION.
ELSE
xmit ⇐ IDLE

mr_an_enable =TRUE

mr_an_enable =FALSE

AN_RESTART

start link_timer
mr_np_loaded ⇐ FALSE
tx_Config_Reg<D15:D0> ⇐ 0
xmit ⇐ CONFIGURATION

link_timer_done

AN_DISABLE_LINK_OK

xmit ⇐ DATA

ability_match=TRUE ∗ rx_Config_Reg<D15:D0>=0

ABILITY_DETECT

toggle_tx ⇐ mr_adv_ability<12>
tx_Config_Reg<D15> ⇐ mr_adv_ability<16>
tx_Config_Reg<D14> ⇐ 0
tx_Config_Reg<D13:D0> ⇐ mr_adv_ability<14:1>

ability_match=TRUE ∗ rx_Config_Reg<D15:D0>≠0

ACKNOWLEDGE_DETECT

tx_Config_Reg<D14> ⇐ 1

(acknowledge_match=TRUE ∗ consistency_match=FALSE) + (ability_match=TRUE ∗ rx_Config_Reg<D15:D0>=0)

acknowledge_match=TRUE ∗ consistency_match=TRUE

Optional Implementation

NEXT_PAGE_WAIT

mr_np_loaded ⇐ FALSE
tx_Config_Reg<D15> ⇐ mr_np_tx<16>
tx_Config_Reg<D14> ⇐ 0
tx_Config_Reg<D13:D12> ⇐ mr_np_tx<14:13>
tx_Config_Reg<D11> ⇐ toggle_tx
tx_Config_Reg<D10:D0> ⇐ mr_np_tx<11:1>

ability_match=TRUE ∗ ((toggle_rx ∧ rx_Config_Reg<D11>)=1) ∗ rx_Config_Reg<D15:D0>≠0

COMPLETE_ACKNOWLEDGE

Start link_timer
toggle_tx ⇐ !toggle_tx
toggle_rx ⇐ rx_Config_Reg<D11>
np_rx ⇐ rx_Config_Reg<NP>
mr_page_rx ⇐ TRUE

link_timer_done ∗ mr_adv_ability<16>=1 ∗ mr_lp_adv_ability<16>=1 ∗ mr_np_loaded=TRUE ∗ (tx_Config_Reg<NP>=1 + np_rx=1) ∗ (ability_match=FALSE + rx_Config_Reg<D15:D0>≠0)

ability_match=TRUE ∗ rx_Config_Reg<D15:D0>=0

((link_timer_done ∗ (mr_adv_ability<16>=0 + mr_lp_adv_ability<16>=0)) + (link_timer_done ∗ mr_adv_ability<16>=1 ∗ mr_lp_adv_ability<16>=1 ∗ tx_Config_Reg<NP>=0 ∗ np_rx=0)) ∗ (ability_match=FALSE + rx_Config_Reg<D15:D0>≠0)

IDLE_DETECT

start link_timer
xmit ⇐ IDLE
resolve_priority ⇐ ON

ability_match=TRUE ∗ rx_Config_Reg<D15:D0>=0

idle_match=TRUE ∗ link_timer_done

LINK_OK

xmit ⇐ DATA
mr_an_complete ⇐ TRUE
resolve_priority ⇐ ON

ability_match=TRUE

COMPLETE_ACKNOWLEDGE

This state ensures that the link partner has time to see that the device has set its ACK bit to one

The device transmits abilities with the ACK bit set to one.

Exit Cases

ability_match = TRUE and rx_Config_Reg <D15:D0> = 0

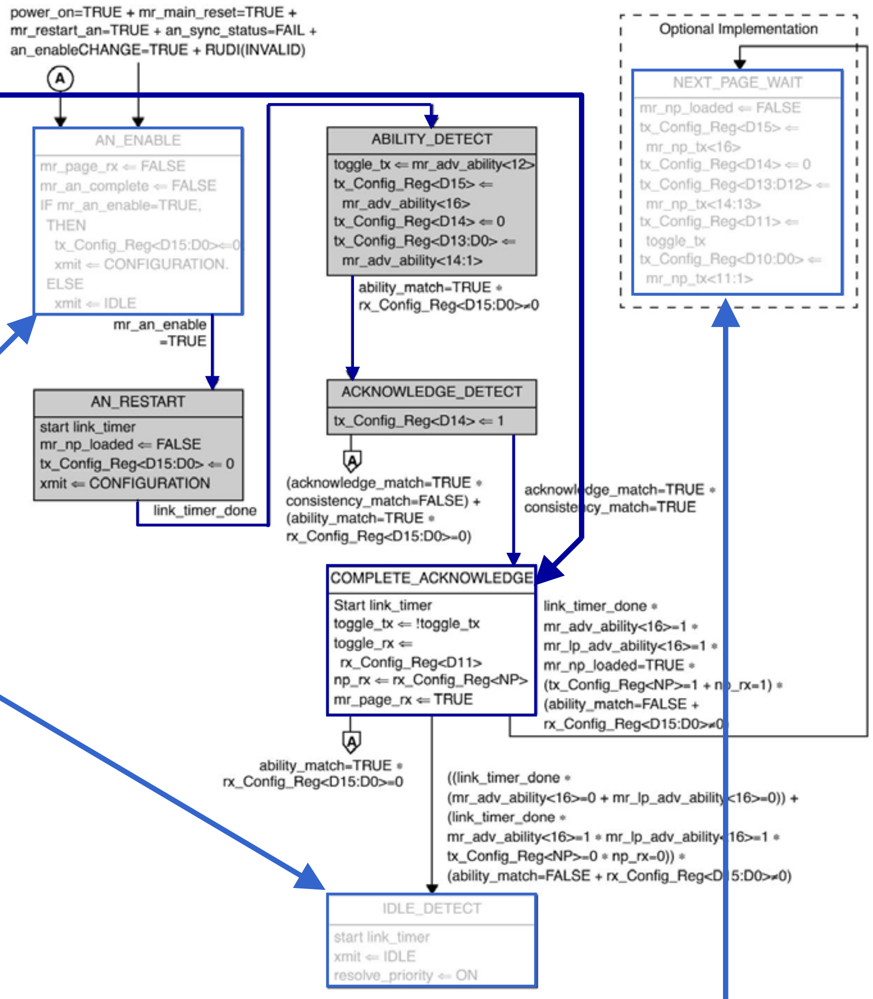
- Exit if the device receives 3 consecutive config_regs encoded with breaklink

((link_timer = DONE and (mr_adv_ability<16>=0 or mr_lp_adv_ability<16>=0)) or (link_timer = DONE and mr_adv_ability<16>=1 and mr_lp_adv_ability<16>=1 and tx_Config_Reg<NP>=0 and np_rx=0)) and (ability_match = FALSE or rx_Config_Reg<D15:D0> ≠ 0)

- Exit if link_timer is done and one of the devices doesn't support next page exchange or both devices are done transmitting next pages and the device isn't receiving breaklink

link_timer = DONE and mr_adv_ability<16>=1 and mr_lp_adv_ability<16>=1 and mr_np_loaded = TRUE and (tx_Config_Reg<NP>=1 or np_rx=1) and (ability_match = FALSE or rx_Config_Reg<D15:D0> ≠ 0)

- Exit if link_timer is done and both devices support next page exchange and at least one device still has next pages to transmit and the device isn't receiving breaklink



NEXT_PAGE_WAIT

This state is similar to the ABILITY_DETECT state, the device stays in this state until it receives 3 consecutive and consistent Config_Regs

The device begins transmission of a next page

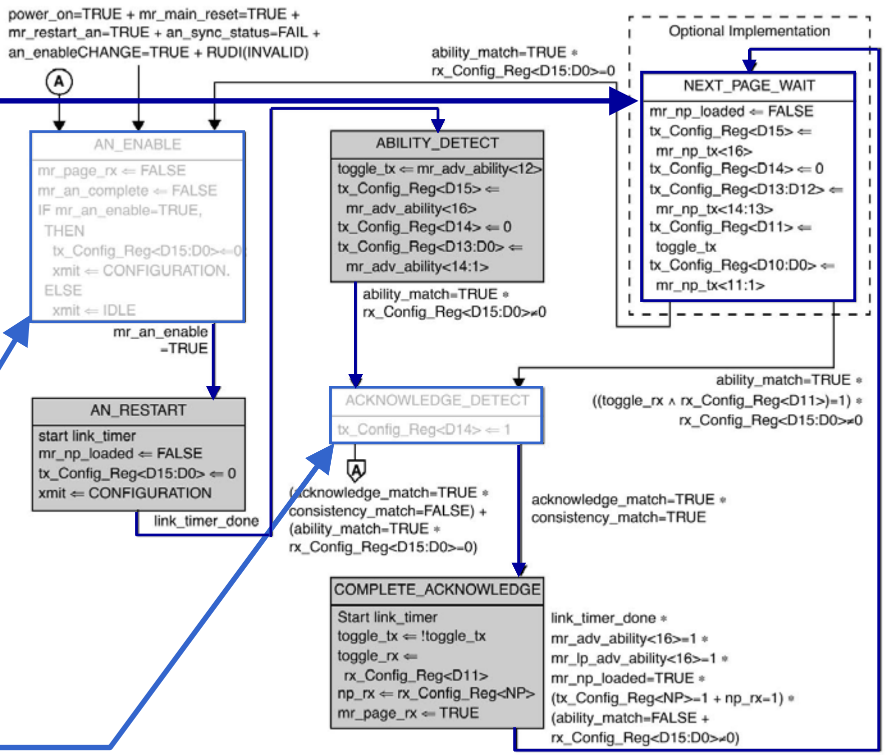
Exit Cases

ability_match = TRUE and rx_Config_Reg <D15:D0> = 0

- Exit if the device receives 3 consecutive Config_Regs encoded with breaklink

ability_match = TRUE and ((toggle_rx xor rx_Config_Reg<D11>)=1) and rx_Config_Reg<D15:D0> ≠ 0

- Exit if the device has received 3 consecutive and consistent next pages and the toggle bit has been changed since the previous next page and the device is not receiving breaklink



ACKNOWLEDGE_DETECT

This state ensures that the link partner has acknowledged that it has seen the devices Config_Reg

The device transmits abilities with the ACK bit set to one.

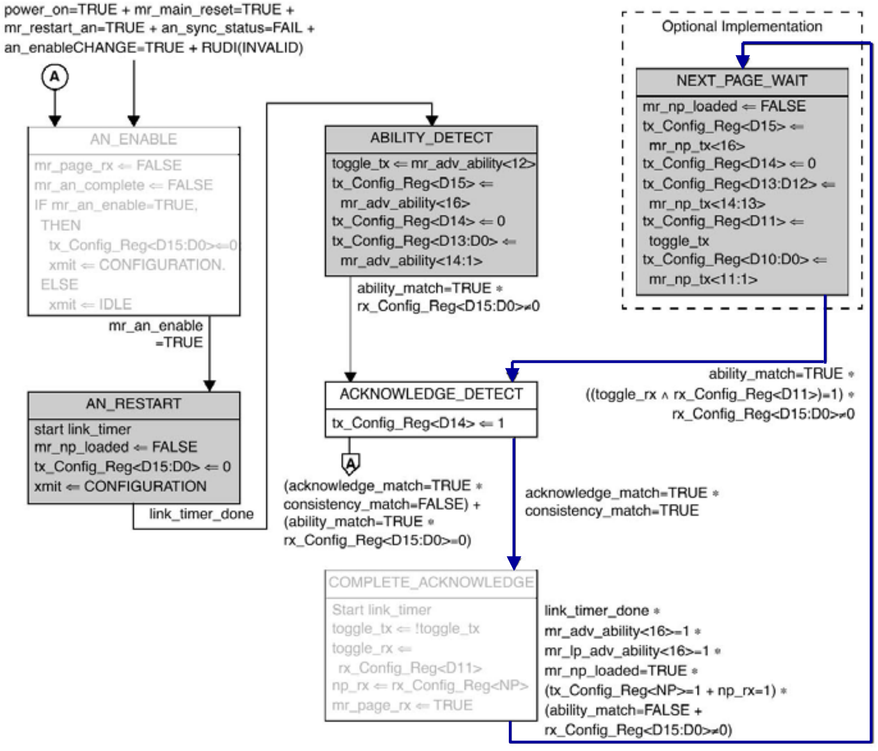
Exit Cases

(acknowledge_match = TRUE and consistency_match = FALSE) or (ability_match = TRUE and rx_Config_Reg <D15:D0> = 0)

- Exit if the device receives 3 consecutive config_regs encoded with breaklink or if the Config_Reg received in ACKNOWLEDGE_DETECT is different than the Config_Reg that was received in NEXT_PAGE_WAIT

acknowledge_match = TRUE and consistency_match = FALSE

- Exit if the Config_Reg received in ACKNOWLEDGE_DETECT is the same as the Config_Reg that was received in NEXT_PAGE_WAIT



COMPLETE_ACKNOWLEDGE

This state ensures that the link partner has time to see that the device has set its ACK bit to one

The device transmits abilities with the ACK bit set to one.

Exit Cases

ability_match = TRUE and rx_Config_Reg <D15:D0> = 0

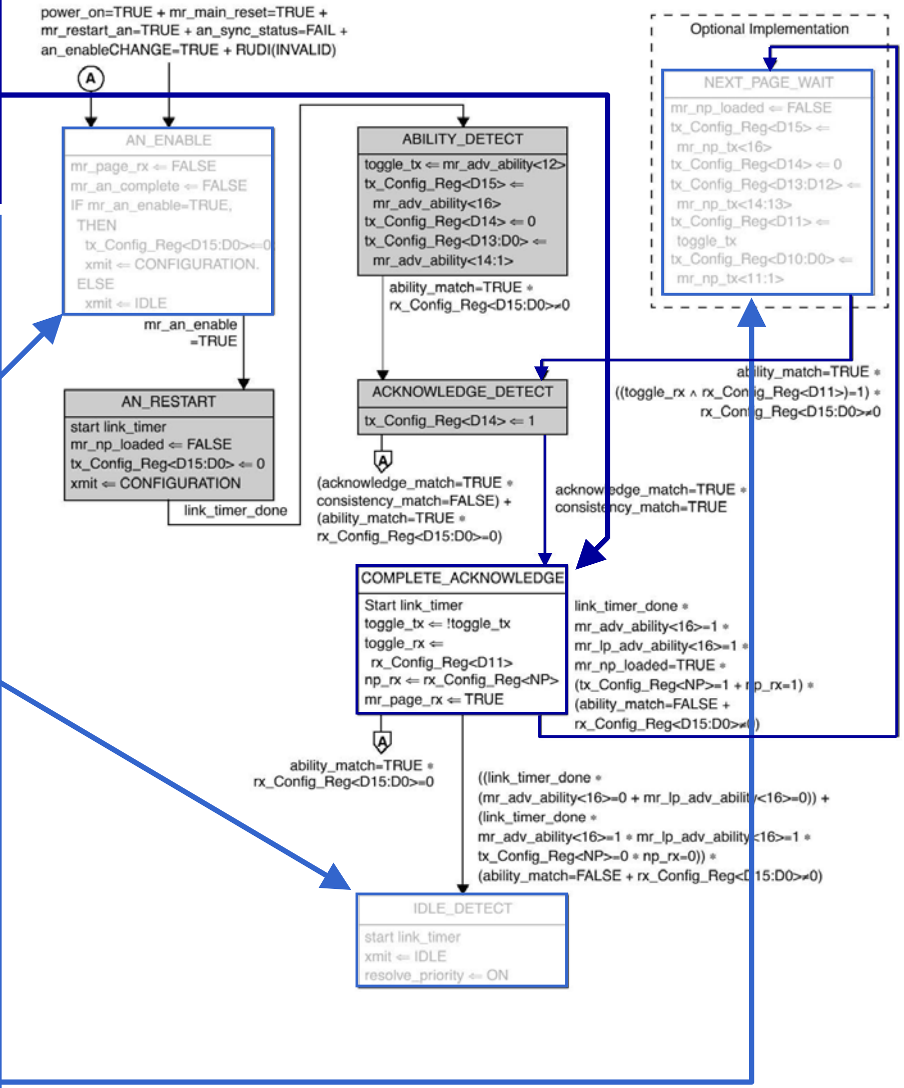
- Exit if the device receives 3 consecutive config_regs encoded with breaklink

((link_timer = DONE and (mr_adv_ability<16>=0 or mr_lp_adv_ability<16>=0)) or (link_timer = DONE and mr_adv_ability<16>=1 and mr_lp_adv_ability<16>=1 and tx_Config_Reg<NP>=0 and np_rx=0)) and (ability_match = FALSE or rx_Config_Reg<D15:D0> ≠ 0)

- Exit if link_timer is done and one of the devices doesn't support next page exchange or both devices are done transmitting next pages and the device isn't receiving breaklink

link_timer = DONE and mr_adv_ability<16>=1 and mr_lp_adv_ability<16>=1 and mr_np_loaded = TRUE and (tx_Config_Reg<NP>=1 or np_rx=1) and (ability_match = FALSE or rx_Config_Reg<D15:D0> ≠ 0)

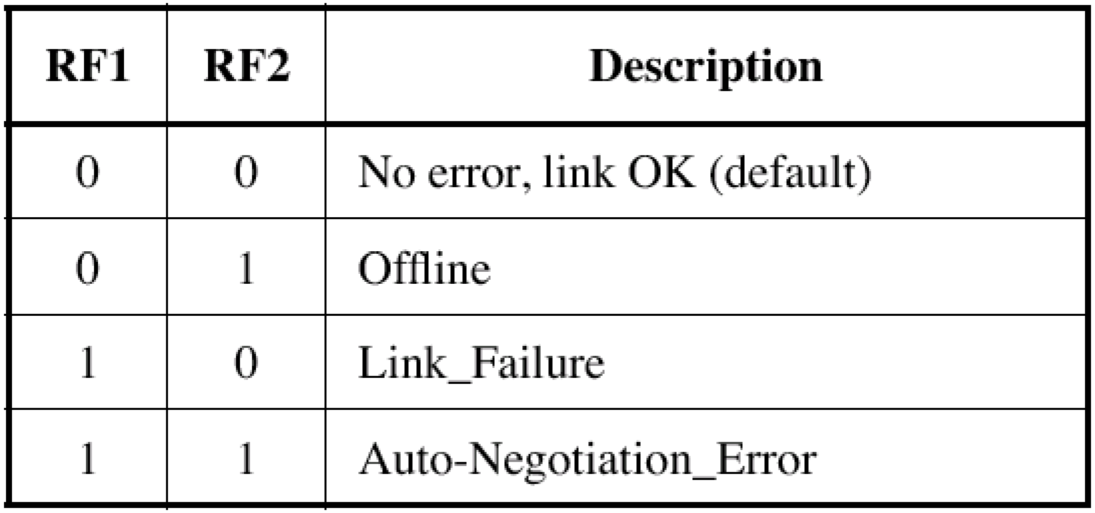
- Exit if link_timer is done and both devices support next page exchange and at least one device still has next pages to transmit and the device isn't receiving breaklink



power_on=TRUE + mr_main_reset=TRUE + mr_restart_an=TRUE + an_sync_status=FAIL + an_enableCHANGE=TRUE + RUDI(INVALID)
A
AN_ENABLE
mr_page_rx ⇐ FALSE
mr_an_complete ⇐ FALSE
IF mr_an_enable=TRUE,
THEN
tx_Config_Reg<D15:D0>⇐0
xmit ⇐ CONFIGURATION.
ELSE
xmit ⇐ IDLE
mr_an_enable =TRUE
AN_RESTART
start link_timer
mr_np_loaded ⇐ FALSE
tx_Config_Reg<D15:D0> ⇐ 0
xmit ⇐ CONFIGURATION
link_timer_done
ABILITY_DETECT
toggle_tx ⇐ mr_adv_ability<12>
tx_Config_Reg<D15> ⇐ mr_adv_ability<16>
tx_Config_Reg<D14> ⇐ 0
tx_Config_Reg<D13:D0> ⇐ mr_adv_ability<14:1>
ability_match=TRUE * rx_Config_Reg<D15:D0>≠0
ACKNOWLEDGE_DETECT
tx_Config_Reg<D14> ⇐ 1
A
(acknowledge_match=TRUE * consistency_match=FALSE) + (ability_match=TRUE * rx_Config_Reg<D15:D0>=0)
acknowledge_match=TRUE * consistency_match=TRUE
Optional Implementation
NEXT_PAGE_WAIT
mr_np_loaded ⇐ FALSE
tx_Config_Reg<D15> ⇐ mr_np_tx<16>
tx_Config_Reg<D14> ⇐ 0
tx_Config_Reg<D13:D12> ⇐ mr_np_tx<14:13>
tx_Config_Reg<D11> ⇐ toggle_tx
tx_Config_Reg<D10:D0> ⇐ mr_np_tx<11:1>
ability_match=TRUE * ((toggle_rx ∧ rx_Config_Reg<D11>)=1) * rx_Config_Reg<D15:D0>≠0
COMPLETE_ACKNOWLEDGE
Start link_timer
toggle_tx ⇐ !toggle_tx
toggle_rx ⇐ rx_Config_Reg<D11>
np_rx ⇐ rx_Config_Reg<NP>
mr_page_rx ⇐ TRUE
link_timer_done * mr_adv_ability<16>=1 * mr_lp_adv_ability<16>=1 * mr_np_loaded=TRUE * (tx_Config_Reg<NP>=1 + np_rx=1) * (ability_match=FALSE + rx_Config_Reg<D15:D0>≠0)
A
ability_match=TRUE * rx_Config_Reg<D15:D0>=0
((link_timer_done * (mr_adv_ability<16>=0 + mr_lp_adv_ability<16>=0)) + (link_timer_done * mr_adv_ability<16>=1 * mr_lp_adv_ability<16>=1 * tx_Config_Reg<NP>=0 * np_rx=0)) * (ability_match=FALSE + rx_Config_Reg<D15:D0>≠0)
IDLE_DETECT
start link_timer
xmit ⇐ IDLE
resolve_priority ⇐ ON

Remote Fault

- The remote fault bits of the base page are used to transmit link fault and error information to the link partner
- There are 4 possible encodings for the 2 bits:



RF1	RF2	Description
0	0	No error, link OK (default)
0	1	Offline
1	0	Link_Failure
1	1	Auto-Negotiation_Error

No Error, Link OK

- This is the default encoding for the remote fault bits
- It is used by devices that either don’t support remote fault detection and devices that support remote fault detection but haven’t detect any errors or faults

Offline

- The offline encoding is used to signal that the device is about the go offline
- An offline indication could be caused by the device powering off, running transmitter tests, or removing the device from the active configuration

Link_Failure

- The link_failure indication is used to signal either loss of synchronization or reception of breaklink for a duration longer than link_timer

Auto-Negotiation_Error

- The auto-negotiation_error encoding is used to indicate that the device could not resolve a mode of operation between the link partner and itself
- For example, if one device only supported full duplex and its link partner only supported half duplex then both devices would encode the remote fault bits with auto-negotiation_error

Summary

- Auto-Negotiation is located inside the PCS
- Auto-Negotiation transfers information to the link partner in a 16-bit Config_Reg to resolve the optimal link
- When Auto-Negotiation is complete, the PCS controls what is transmitted