AN timeout and restart mechanism

Comments #234, #282

Adee Ran, Cisco

Supporters

- Geoff Zhang, AMD
- Brent Rothermel, Cornelis Networks
- Leon Bruckman, NVIDIA
- Upen Reddy Kareti, Cisco
- Itamar Levin, Altera

Preface

- In D1.4 we have link_fail_inhibit_timer of 60 seconds (adopted in January 2025)
- The issue of AN timing and its relationship to ILT has been discussed at length.
  - Consensus building efforts reported in lusted_3dj_adhoc_02a_250206 and lusted_3dj_adhoc_01a_250220.
  - It was noted that AN has no restart mechanism other than through timeout.
  - This means that a device cannot unilaterally decide to “retry” before the timeout expires.
- With the current value and AN definitions, 60 seconds is both **maximum time-to-link** and **minimum time-to-retry…**
  - This provides a lot of flexibility to management and ILT implementations
  - but is bad for user experience and debugging.
- This presentation proposes a solution to this problem.

Scope and base assumptions

- This presentation addresses CR and KR links that use AN.
  - All optical links (for which AN is not defined), even if they include ILT on some ISLs, are not addressed.
  - CR/KR links in which AN is disabled (mr_autoneg_enable=false) are not addressed.
- It is assumed that the intent of the adopted 60 seconds is to allow a long time for ILT.
  - We want to keep that possible.
- It is assumed that non-time-critical activities are handled by "management"; the actions defined by state diagrams are not necessarily implemented in hardware.
- "**AN restart**" in this presentation means "**a transition from AN GOOD CHECK to TRANSMIT DISABLE**".

New goal (comment #234)

- We would like to **enable restarting the link-up process (going back to AN) by management on either side**, after a reasonable time (<< 60 seconds).
  - In many cases the link-up process will be fast enough.
  - If the link does not come up within the expected time, it might be due to a random ILT problem (which could be solved by a restart)…
  - or it could be a configuration problem (bug) that a restart will not solve.
- The decision of whether (and when) to restart AN if the link doesn't come up can be left to application or implementation choice
  - Preferences may differ, and we can have long discussions
  - There is room for innovation in this area
  - This is a not the topic of this presentation!

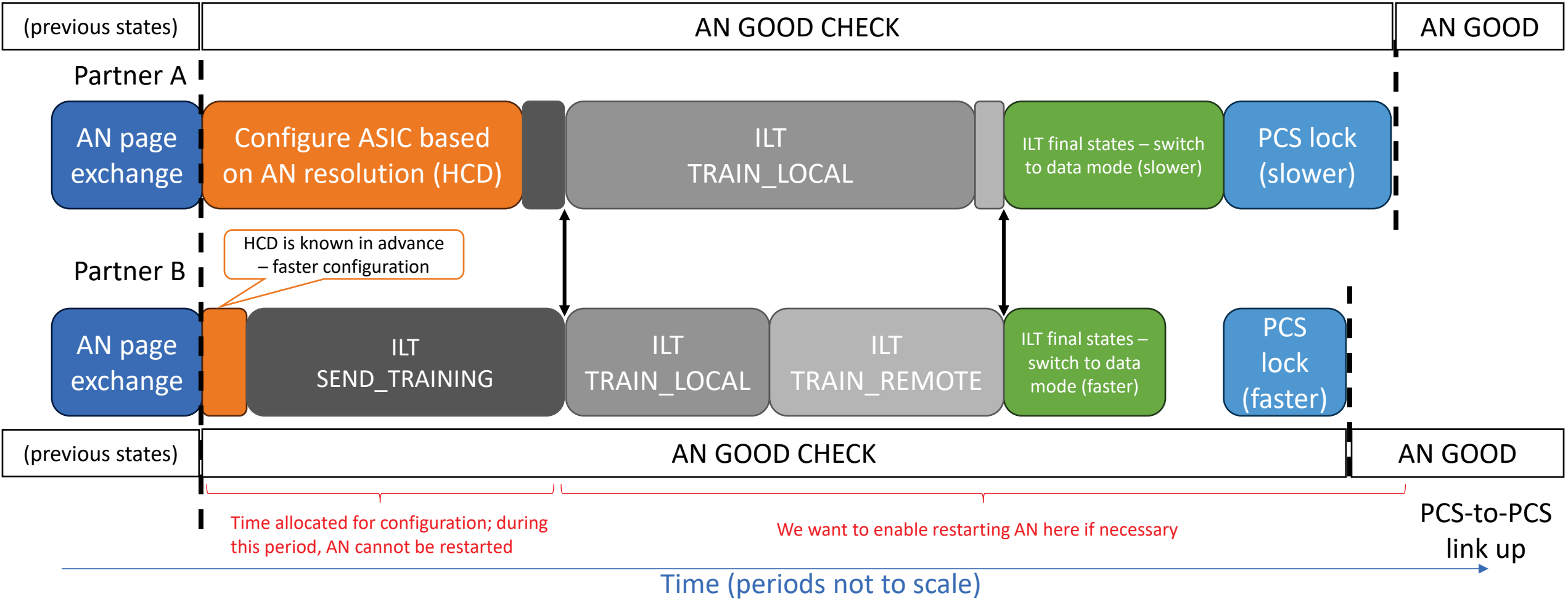
Timing considerations of AN – 3dj Era
(Used in some electrical links)

- When AN page exchange is done, the PCS-to-PCS link is known to be physically connected from end to end
- Management on either side may need to configure the ASIC and possibly a local retimer according to the chosen ability (HCD)
  - This could take a long time depending on both the retimer and the management software
  - Management processor can service many ports in parallel, and have other duties
  - If the HCD is known in advance, it can be much faster
- ILT can run only after the ASICs (and possibly retimers) have been configured
  - It may be long, and may not be performed in all ISLs in parallel
  - But in many cases, it will be fast and parallel
- The time required to bring up the end-to-end path can be much longer than the time consumed by ILT.
  - But in many practical cases ILT will be the dominant period.

From [lusted_3dj_adhoc_01a_250220]

AN timing illustration – link w/o AUIs

(see backup for example with AUI)



AN arbitration state diagram

With the existing AN arbitration state diagram, link_fail_inhibit_timer is both "max **time-to-link**" and "min **time-to-retry**".

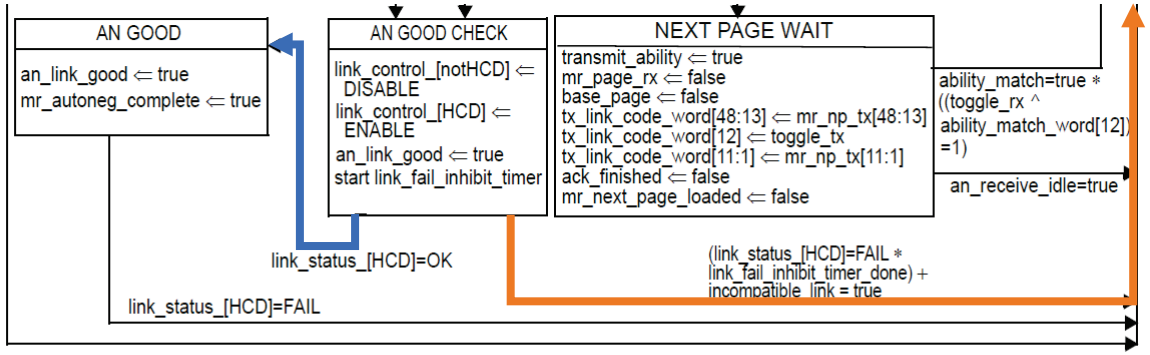


Figure 73–11—Arbitration state diagram

The choice between these paths is done by the link_status parameter of AN_SIGNAL.indication (generated by the PCS).

The current definition of link_status allows only OK and FAIL, e.g. in 119.6:

119.6 Auto-Negotiation

The following requirements apply to a PCS used with a 200GBASE-CR4 or 200GBASE-KR4 PMD where support for the Auto-Negotiation process defined in Clause 73 is mandatory. The PCS shall support the AN_LINK.indication(link_status) primitive (see 73.9). The parameter link_status shall take the value FAIL when PCS_status=false and the value OK when PCS_status=true. The primitive shall be generated when the value of link_status changes.

PCS_status is defined in 119.2.6.2.2:

PCS_status

A Boolean variable that is true when align_status is true and is false otherwise.

So link_status is essentially align_status of the PCS.

We don't want to change the state diagram!
but we can assign new meaning to link_status in new PHYs
(as we did with the SIGNAL_OK parameter of IS_SIGNAL)

Solution (AN side)

- Add a third possible value, IN_PROGRESS.
- link_status=IN_PROGRESS will keep the state diagram in “AN GOOD CHECK” state, preventing AN from restarting.
- The PCS generates IN_PROGRESS while ILT has not completed (and thus the PCS does not receive valid data and cannot align).
  - This is indicated in the PCS by having SIGNAL_OK of either IN_PROGRESS or READY in PMA:IS_SIGNAL.indication into the PCS.
- When ILT is completed everywhere, RTS propagation will cause SIGNAL_OK to become OK.
  - This will cause signal_ok = true in the PCS.

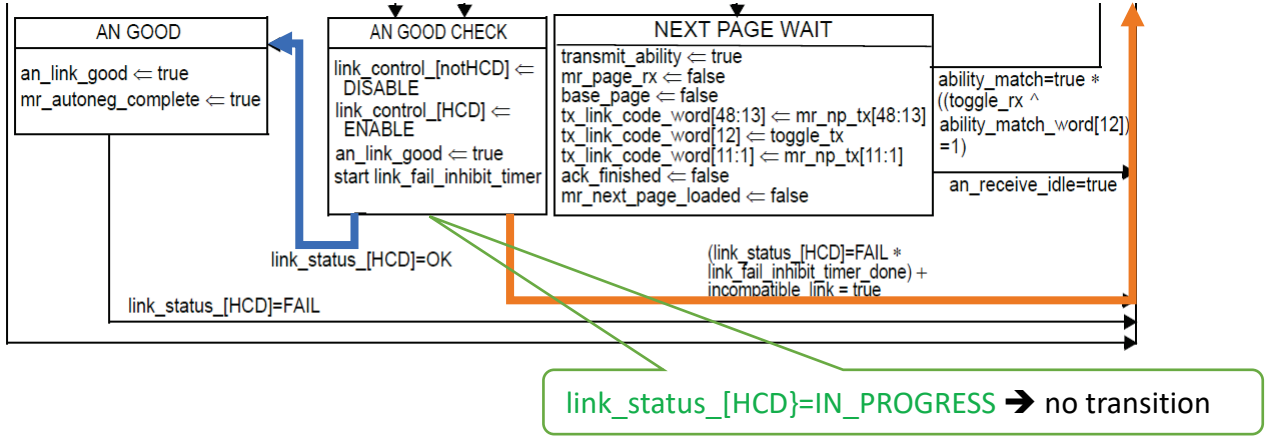


Figure 73–11—Arbitration state diagram

Solution (PCS side)

- When the PCS gets signal_ok = true it starts the AM lock and synchronization processes (see backup slide).
  - Although there is no specified timing, these processes are assumed to be fast.
- If the PCS completes these processes successfully, then align_status = true ➔ link_status = OK.
- If the deskew process fails, or if multiple uncorrectable codewords are detected (link down), the PCS asserts restart_lock; this should cause link_status = FAIL and AN restart.
  - **This also means PCS reset can be used by management to restart AN at any time**, but it is not mandatory**.**
- In all other cases, the PCS should set link_status = IN_PROGRESS (new value).
  - Since this is a new feature for existing PCSs, it should be conditional on a management variable.
  - Using IN_PROGRESS instead of FAIL (as defined above) would be interoperable with any link partner, and can be valuable in PHYs other than those in this project, so it should be allowed.

```
if (reset + restart_lock) == true
      link_status = FAIL
else if align_status == true
      link_status = OK
else if use_in_progress
      link_status = IN_PROGRESS
else
      link_status = FAIL
```

use_in_progress
Boolean variable indicating support of the value IN_PROGRESS for link_status. It is true for a PCS in the same package as a SM-PMA. Otherwise, its value is implementation dependent.

Solution (editor's version)

In each of the PCS clauses (119, 172, 175) and in clause 45:

- Add a definition for a new variable **use_in_progress** (as in the previous slide) and MDIO register mapping.
- Change the definition of **link_status** in the Auto-negotiation subclauses (119.6, 172.6, and 175.7). For example, in 119.6:

  The following requirements apply to a PCS used with a 200GBASE-CR4, 200GBASE-CR2, 200GBASE-KR4, 200GBASE-KR2, 400GBASE-CR4, or 400GBASE-KR4 PMD where support for the Auto-Negotiation process defined in Clause 73 is mandatory.

  The PCS shall support the AN_LINK.indication(link_status) primitive (see 73.9). The parameter link_status shall take ~~the value FAIL when PCS_status=false and the value OK when PCS_status=true~~ one of the values FAIL, IN_PROGRESS, or OK, according to Table 119-2a.

  The primitive shall be generated when the value of link_status changes.

Table 119-2a--AN_LINK.indication(link_status) generation

reset + restart_lock	align_status	use_in_progress	link_status
True	Don't care	Don't care	FAIL
False	False	False	FAIL
		True	IN_PROGRESS
	True	Don't care	OK

In clause 73 (AN): change the semantics of AN_LINK.indication (73.9.1.1) as follows:

The link_status parameter shall assume one of ~~two~~ three values: OK, IN_PROGRESS, or FAIL, indicating ~~whether the underlying receive channel is intact and enabled (OK) or not intact (FAIL)~~ the status of the PCS alignment, as defined in the PCS clause.

Restarting AN

- AN can be restarted by either partner after link_fail_inhibit_timer expires, by resetting the PCS.
- This causes the following process
  - In the PCS: PCS reset ➔ link_status=FAIL
  - In AN arbitration (Figure 73-11): link_status=FAIL when link_fail_inhibit_timer_done ➔ transition to TRANSMIT DISABLE
  - In TRANSMIT DISABLE the local PHY is disabled (link_control_[all]= disable), which will reset all components, including ILT
  - TRANSMIT DISABLE will break the link for the link partner too:
    - If the partner is in DATA mode – PCS will lose AM lock and cause link_status = FAIL; AN will restart "automatically"
    - If the partner has any ISL in TRAINING mode, it will lose TF lock and the training control state diagram will go to FAIL; this will be propagated via SIGNAL_OK (see backup slide) across all ISLs, up to the ILT function in the PMA below the PCS.
    - The partner's management can respond to the FAIL status in the ILT by resetting its own PCS, which would restart AN.
    - If the link partner has not acquired TF lock yet, it will not cause FAIL (but that's ok).

What about link_fail_inhibit_timer

- A PCS that supports IN_PROGRESS will report FAIL when it is in reset
  - But once out of reset it will not report FAIL unless it reaches DESKEW_FAIL.
  - This means link_fail_inhibit_timer must be longer than the time that the PCS is in reset. We can assume this is short duration (<1 s).
- Management should not reset the PCS again before link_fail_inhibit_timer expires (it may have various effects on the link partner until its own timer expires)
  - This means link_fail_inhibit_timer will be the minimum time to restart.
  - The terminal count can be set to 12 s, as in the 802.3ck PHYs, for the PHYs defined in this project (suggested for consistency – although it could be shorter with no adverse effects).
- When (and after) this timer expires:
  - If training is in progress on any ISL (SIGNAL_OK is either IN_PROGRESS or READY), the link will not fail, and AN will not restart.
  - If ILT fails in any ISL the training control state diagram will be in the FAIL (terminal) state, and set SIGNAL_OK to FAIL, which propagates to the ILT adjacent to the PCS. Management should detect it and cause AN restart by resetting the PCS (AN does not restart “automatically”).
  - If ILT succeeds in all ISLs but the PCS AM lock or deskew fails, AN will restart “automatically”.

Relationship to ILT – comment #252

- ILT is currently defined without a timeout
  - This was defined to enable multi-ISL systems where some ISLs may not be functional for a long time.
- A timer could be added in the training algorithm (TRAIN_LOCAL state)…
  - There is no consensus on whether a timeout should be considered a failure
  - There is no consensus on timeout value
  - The timeout length may be different between clauses/annexes
  - But there is merit in formalizing a timer and enabling management to configure and check it.
  - A default value will serve as industry guidance.
- Should the timer also cover TRAIN_REMOTE?
  - The proposal on the next slide is that it is only for TRAIN_LOCAL, because the link partner's timing is not known in general.

ILT training timer (comment #282)

178B.14.3.1 Variables

mr_training_timer_duration

Unsigned integer variable that controls the terminal count of training_timer in seconds. A value of 0 corresponds to an infinite time. The default value of this variable is defined by the PMD clause or AUI annex.

178B.14.3.3 Timers

quiet_timer

This timer is started when the training control state diagram on a lane enters the QUIET state (see Figure 178B–8). The terminal count of this timer is between 100 ms and 200 ms.

propagation_timer

This timer is started when the training control state diagram on a lane enters the PATH_READY state (see Figure 178B–8). The terminal count of this timer is between 100 ms and 200 ms.

recovery_timer

This timer is started when the training control state diagram on a lane enters the RECOVERY state (see Figure 178B–8). The terminal count of this timer is between 20 ms and 30 ms.

training_timer

This timer is started when the training control state diagram on a lane enters the TRAIN_LOCAL state (see Figure 178B–8). The terminal count of this timer is controlled by the management variable mr_training_timer_duration. The effect of expiration of this timer is implementation dependent.

Add MDIO mapping for **mr_training_timer_duration** (RW) and **training_timer_done** (RO) in Table 178B–6 and in Clause 45.
Add default values of mr_training_timer_duration: in clauses 178 and 179 – 60, in annexes 176C and 176D – 0.

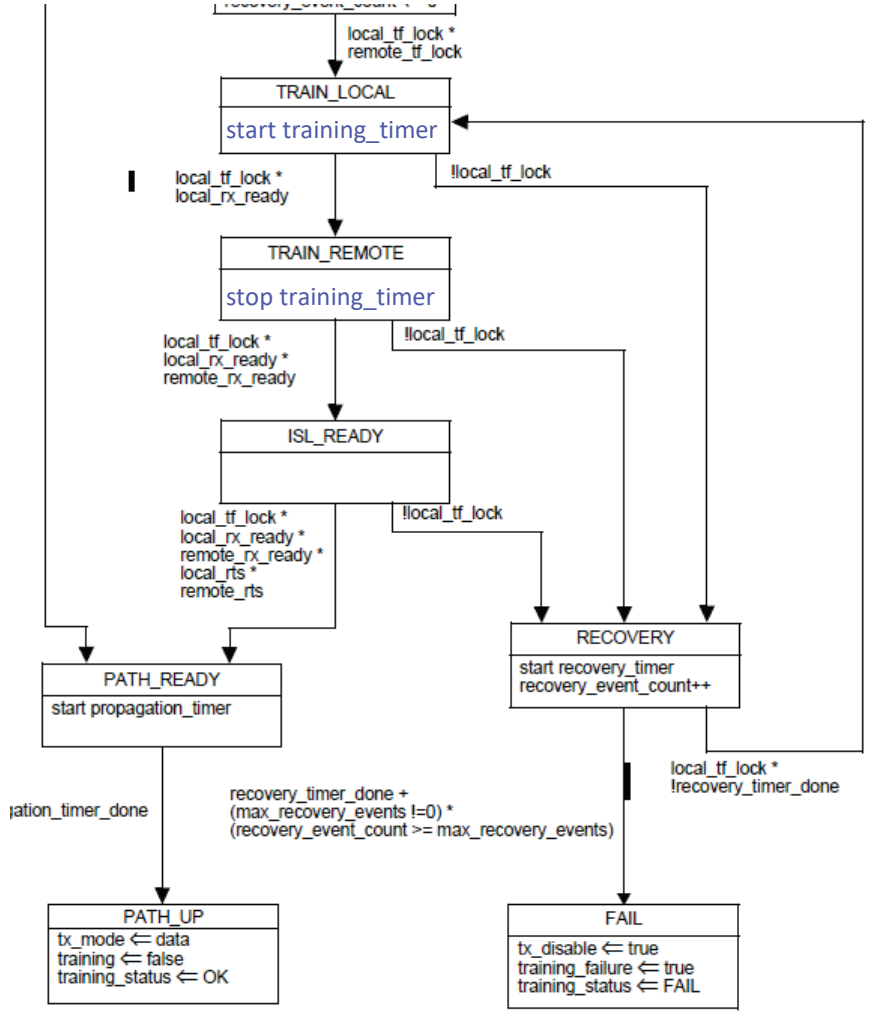


Figure 178B–8—Training control state diagram

Proposal

- Implement the changes to the PCS, AN, and MDIO clauses (119, 172, 175, 73, and 45) as shown on slide 11.
- Implement the changes to ILT (Annex 178B), CR+KR PMDs (Clauses 178 and 179) and MDIO (clause 45) as shown on slide 15.
- In Table 73–7, Add the CR/KR PHYs in this project to the existing row of the 802.3ck PHYs, such that the terminal count of link_fail_inhibit_timer is the same (min:12.3, max:12.4 seconds).

Summary

- The proposal presented enables restarting AN by either side after link_fail_inhibit_timer expires.
- Most changes included in this proposal are variable definitions that are abstract and do not imply a specific implementation.
  - The main change is the definition of a service interface parameter.
  - No new timers or state diagram changes in existing PCS clauses.
  - New ILT timer is used only by management.
- The new behavior of the AN arbitration state diagram can be implemented by management (software) using PCS and ILT status variables.

That’s all

Questions?

Backup

PCS processes

(Illustration; no changes required)

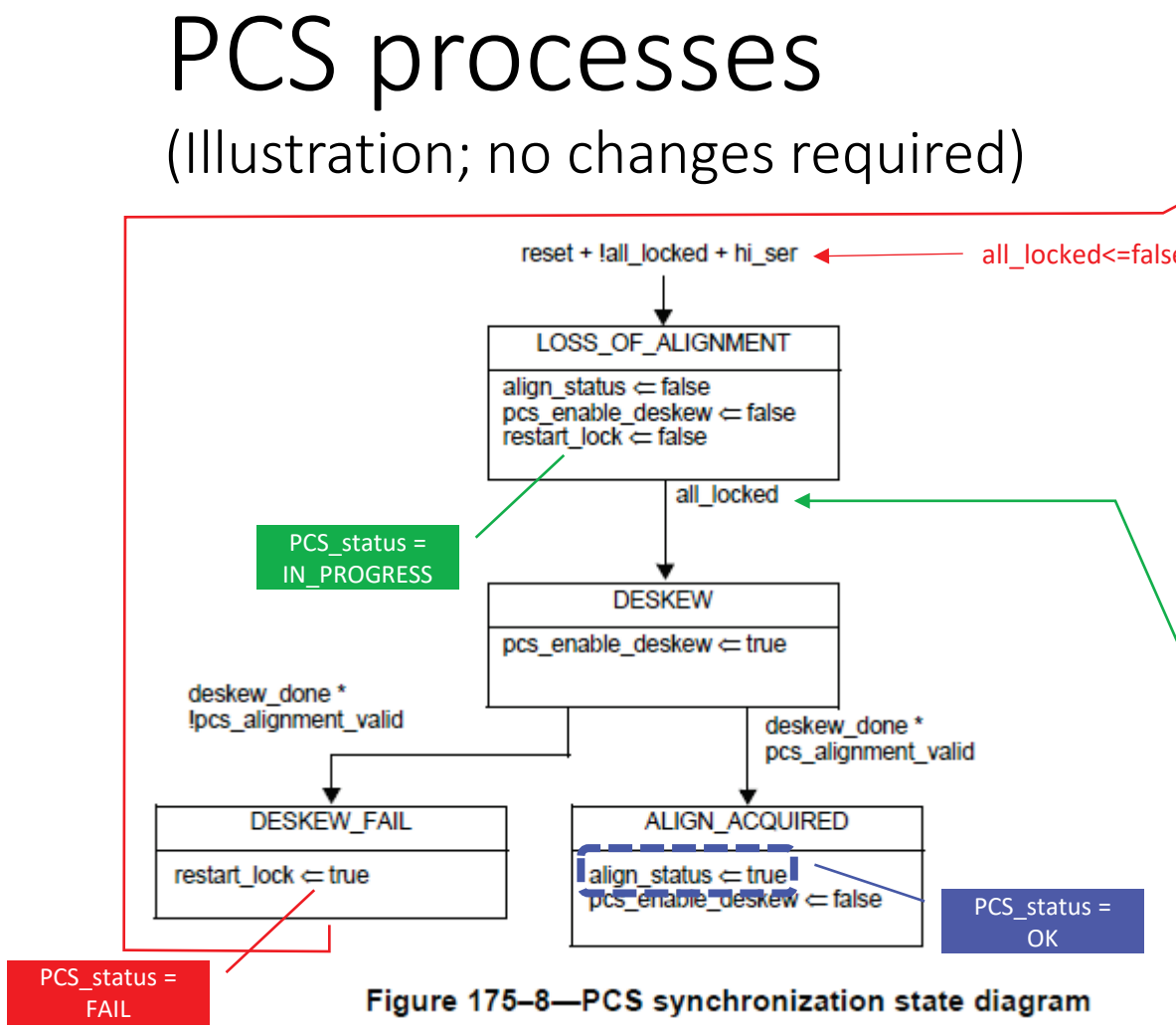


Figure 175–8—PCS synchronization state diagram

Note: clause 119 and clause 172 have different PCS synchronization state diagrams, but all these diagrams have the same conditions for LOSS_OF_ALIGNMENT and generate the **align_status** and **restart_lock** variables.

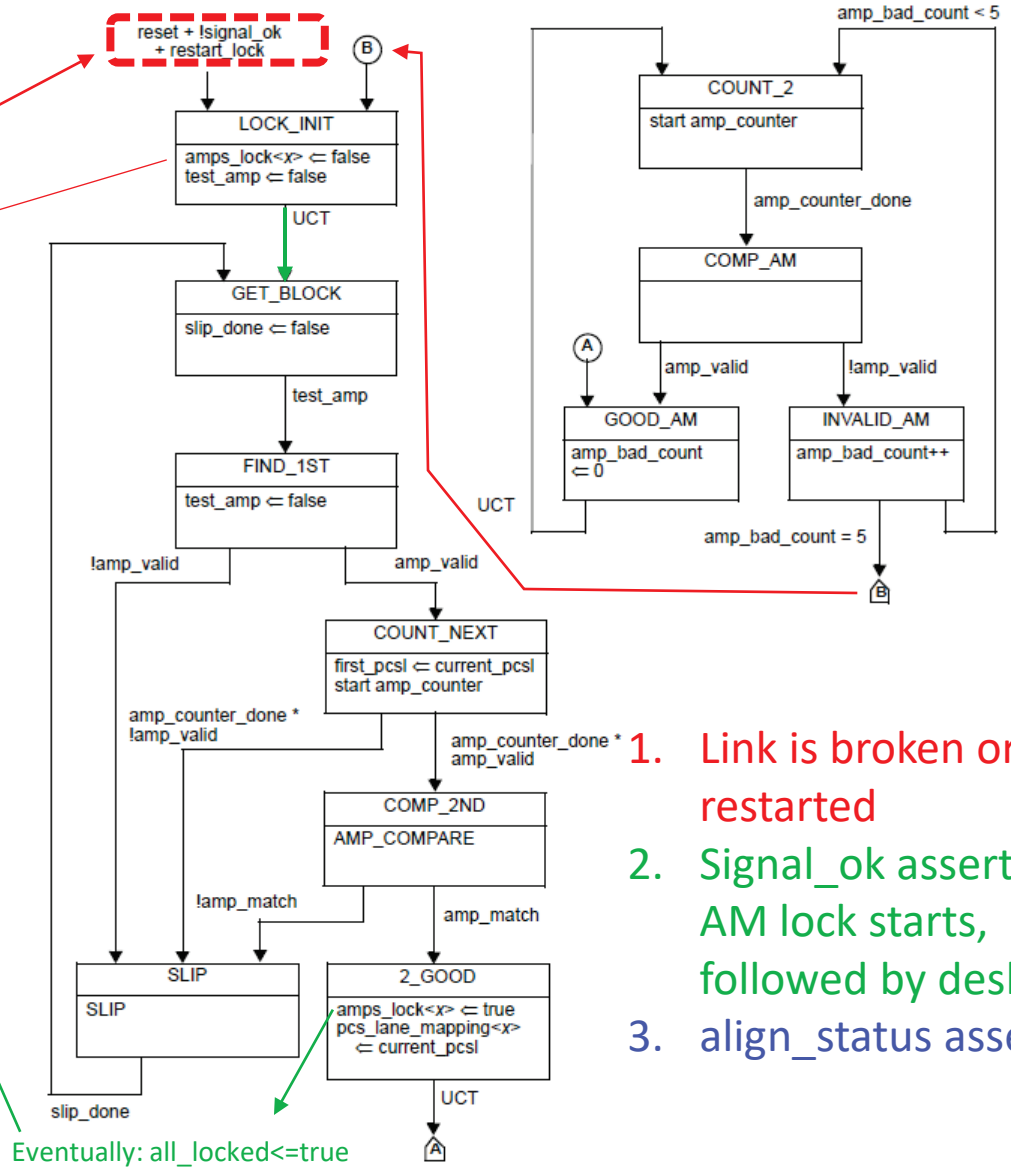


Figure 119–12—Alignment marker lock state diagram

1. Link is broken or restarted
2. Signal_ok asserted; AM lock starts, followed by deskew
3. align_status asserted

PCS_status affects AN

- The state diagrams could be implemented in a way that will cause restart_lock to be asserted only momentarily (see red arrows in the previous slides).
  - This will cause pcs_status to become FAIL only momentarily.
  - Formally, this “momentary FAIL” still affects the AN arbitration state diagram.
- In practice, there may be various ways for passing the PCS status to the AN sublayer such that the desired behavior (e.g., restarting AN) is maintained.
  - “(link_status) indication is relayed from the device with the PCS sublayer to the device with the AN sublayer by means at the discretion of the implementer…”

ILT failure

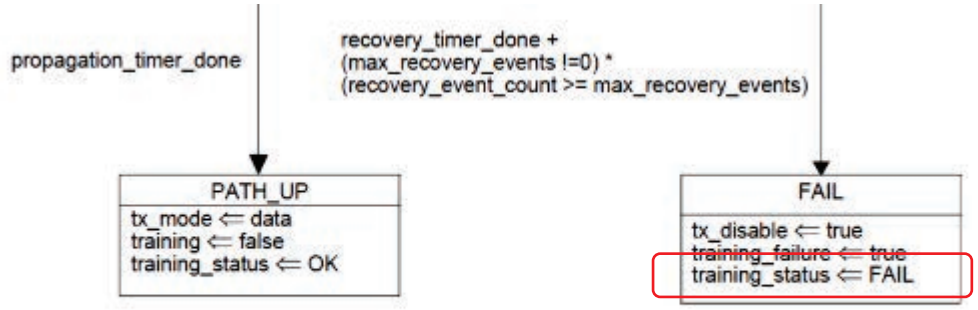


Figure 178B–8—Training control state diagram

- When ILT fails on any ISL it will set training_status to FAIL
- This will affect SIGNAL_OK according to the service interface definitions of the PMD or AUI (e.g. 176D.4):

  The SIGNAL_OK parameter of the PMA:IS_SIGNAL.indication (for an AUI component above the AUI channel) or PMA:IS_SIGNAL.request (for an AUI component below the AUI channel) corresponds to the variable training_status of the ILT function, as defined in 178B.14.2.1. When SIGNAL_OK is either IN_PROGRESS or FAIL, the corresponding tx_symbol parameters on all lanes are unspecified.

- SIGNAL_OK=FAIL propagates through ISLs and reaches the PCS, where it will result in link_status=FAIL and restart AN
- No changes required

AN timing illustration – AUI on one side