10 Mb/s Single Twisted Pair Ethernet

10BASE-T1L EEE

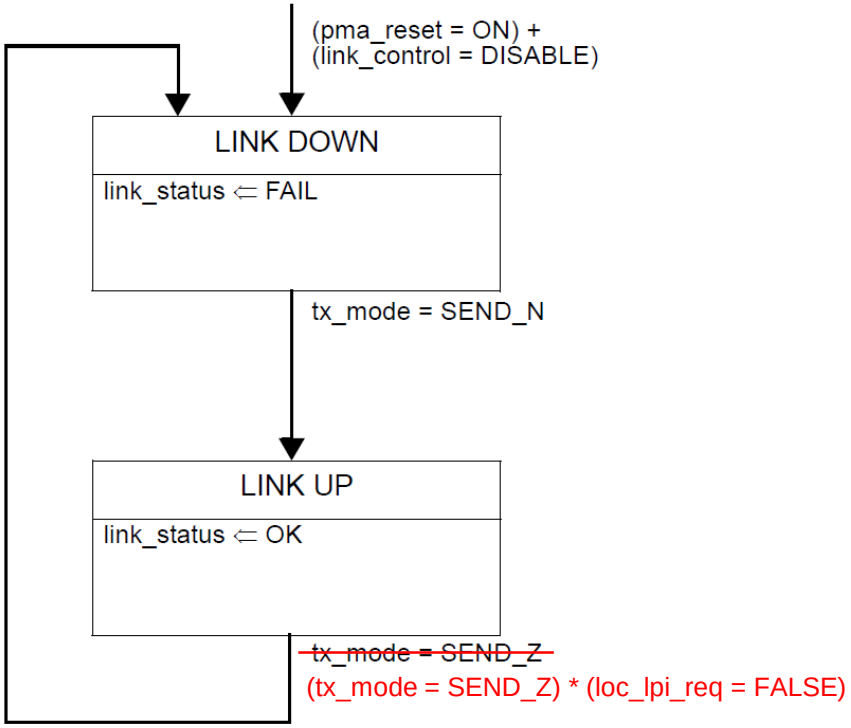
(Comments #574, #575, #579, #582)

Steffen Graber
Pepperl+Fuchs

Content

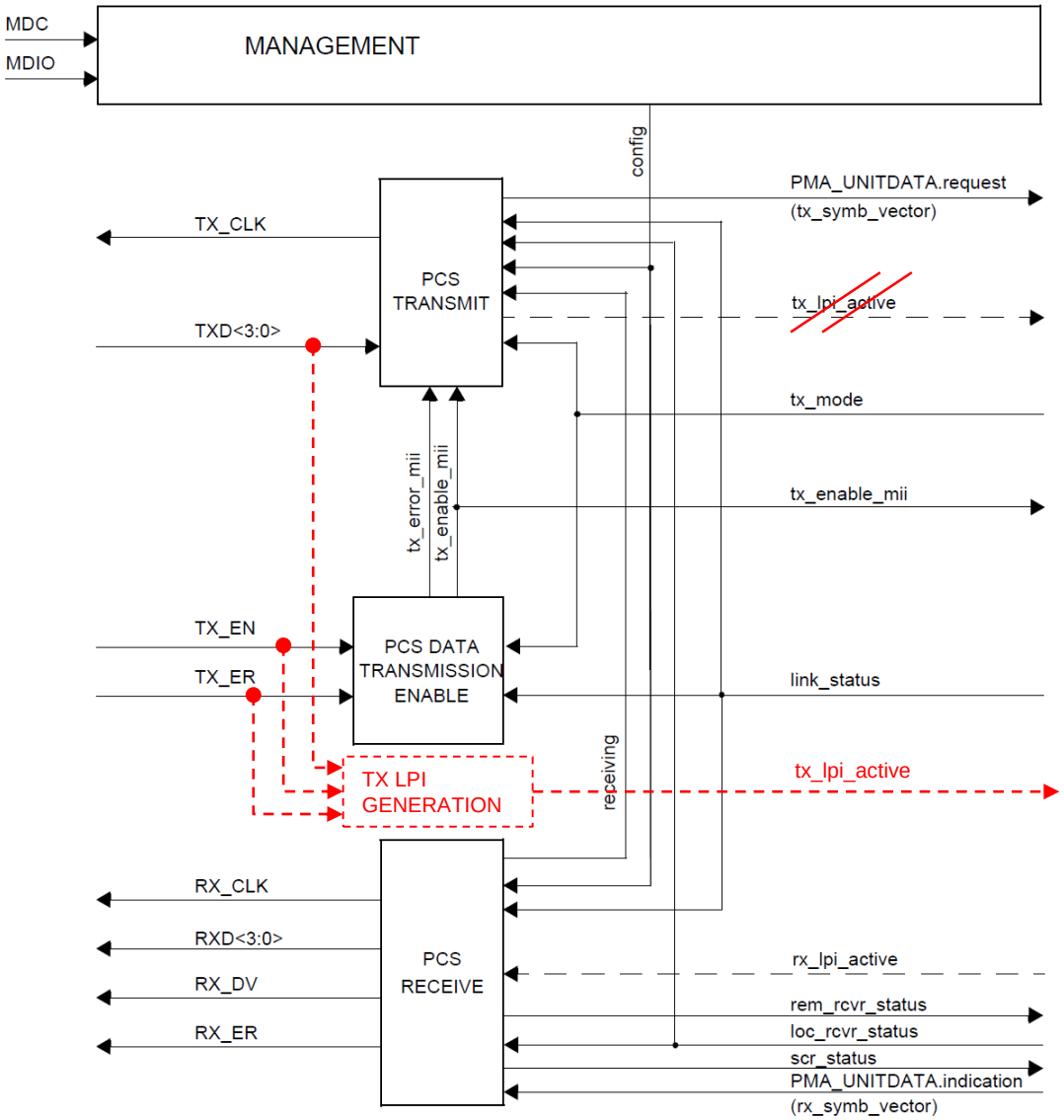
- This presentation contains suggestions for handling the following comments related to EEE:
  - Comment #575: Link Monitor
  - Comment #579: PCS Reference Diagram
  - Comment #582: PHY Control State Diagram
  - Comment #574: Symmetric/Asymmetric LPI and Synchronization (input is needed here)

Comment #575: Link Monitor



- The Link Monitor state diagram as in draft D2.0 will unintentionally generate a link down, when being in QUIET state.
- This is prevented by adding (loc_lpi_req = FALSE) condition as shown above, as the variable loc_lpi_req is TRUE in this case (while LPI mode is active).

Comment #579: PCS Reference Diagram

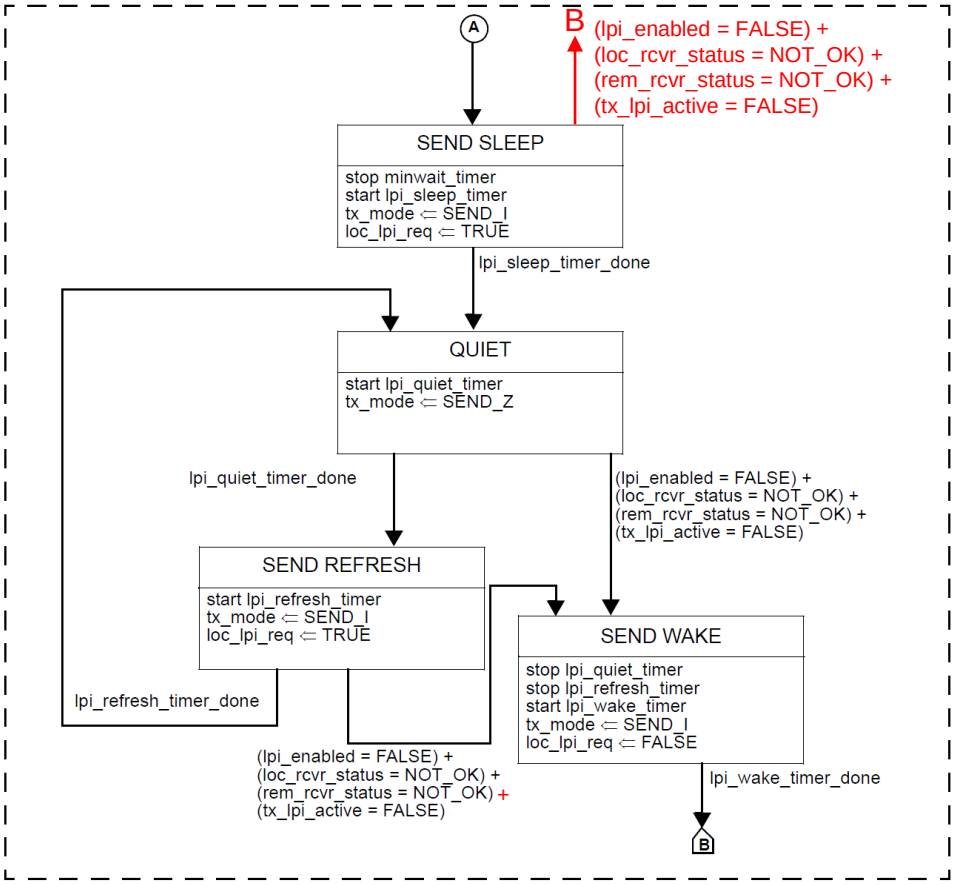


According to the description of variable tx_lpi_active, this variable is set to TRUE, if „Assert Low Power Idle“ condition on the MII is active.

Figure 146-3 shows the variable being generated within PCS TRANSMIT, which does not directly get the control signals from the MII.

To match Figure 146-3 to the intended functionality of tx_lpi_active, a new functional block „TX LPI GENERATION“ should be added, which directly gets the signals from the MII.

Comment #582: PHY Control State Diagram



- Adding above shown changes allows the PHY to go back to normal operation faster, if tx_lpi_active is TRUE for only a short time duration, and the PHY not yet went into QUIET state.
- As it is an improvement at practically no cost, suggestion is to just add it.

Comment #574: Symmetric/Asymmetric LPI

- Currently an asymmetric LPI scheme is being described in Draft D2.0.
- Background is, that the typical 10BASE-T1L data transfer in a lot of industrial, building or IOT applications is quite asymmetric.
- E.g. industrial control sensors, after being configured, just transmit their measurement data in a cyclic way, but do not receive data at all.

- **Therefore the suggestion is to keep an asymmetric LPI scheme, besides there are really good arguments for changing it to a symmetric LPI scheme.**

Comment #574: LPI Timing

- Currently the refresh timing is very conservative (about 1 : 10).
- The background of the conservative refresh timing is to allow a phase shift due to oscillator tolerances between the Master and the Slave PHY of up to approx. 7 to 8 degree assuming an oscillator frequency difference between the Master and the Slave PHY of up to 5 ppm.
- With our FPGA based PHY implementation, doing a 7.5 degree phase shift (48 possible steps) on the Master PHY side shows a visible effect on the SNR for a short time, but does not cause any bit error (so this value was assumed to work in conjunction with the LPI mode).
- The 5 ppm are easy to reach, even in a discrete implementation; in an integrated implementation, likely a significantly reduced clock tolerance is possible.
- If there is input from the task force, that significantly smaller oscillator tolerances can be assumed, then the refresh timing can be more relaxed, providing (significantly) higher energy savings.

- **Information related to timing values or clock tolerances, which can be assumed for integrated silicon, would be really appreciated here**.

Comment #574: LPI Synchronization

- Currently an asymmetric LPI scheme without synchronization between the Master and the Slave PHY is being used (as it was the most simple solution possible).
- Therefore the receivers of the PHYs need to stay active, as it is not known, when the other PHY starts to send its refresh, thus limiting the possible energy efficiency.
- Also the training while in LPI mode can be more complicated, as it can happen, that no PHY, only one PHY or both PHYs are transmitting the same time and the training has to be able to handle these different conditions.
- Thinking about synchronization of the Master and the Slave PHY, it is important to know, that for a 10BASE-T1L PHY during Idle mode, there is currently only one bit (loc_lpi_req) available, which can be transmitted to the remote PHY signaling the LPI status (theoretically an additional bit would be available, but this would significantly reduce the reliability of Idle mode detection, if an SSD has been missed, so it is not really recommended to use this bit).

- **If there is need to add synchronization between the Master and the Slave PHY during LPI mode, technical input (or a proposal) on how to implement this is required.**

Thank You