Energy Efficient Ethernet in the Linux Kernel

Oleksij Rempel – ore@pengutronix.de

my_self = kzalloc()

- Oleksij Rempel, Linux Kernel Hacker
- Expertise in: Medical, Industrial and Agricultural devices
- Addressing challenges: Limited CPU/bandwidth, power efficiency, diagnostic
- Prioritizing long-term sustainable, secure and Open Source Embedded Linux (mainline).

Reducing power consumption - EEE

- Energy Efficient Ethernet
- On some systems, EEE saves 0.2W per port
- One Watt Initiative – reduce standby power under one watt.

Trouble shooting EEE

- Current state of EEE support in Linux kernel v6.10 is different. Some drivers do it properly. (Last talk was about 9.4 :))
- Drivers or even HW doc may provide not enough or not proper information.
- Use oscilloscope!
- This talk is to inspire more kernel hackers to explore this functionality

State of EEE Linux Kernel implementation

- drivers/net/dsa/b53/b53_common.c - OK
  - Fixed by Florian Fainelli – v6.9-rc4
- drivers/net/dsa/mt7530.c - OK
  - Implemented René van Dorst – v5.13
- drivers/net/ethernet/broadcom/genet/bcmmii.c - OK
  - Fixed by Florian Fainelli – v6.4-rc4
- drivers/net/ethernet/freescale/fec_main.c – Partially
  - Mostly fixed by Andrew Lunn – v6.8-rc6
  - Delay configurations seems to be broken

State of EEE Linux Kernel implementation

- drivers/net/ethernet/marvell/mvneta.c - OK
  - Implemented by Russell King – v4.15-rc5
- drivers/net/ethernet/microchip/lan743x_main.c
  - Fixed by Andrew Lunn - v6.9-rc2
- drivers/net/ethernet/samsung/sxgbe/sxgbe_main.c
  - Looks broken. phy_init_eee() is only on open, not on link_up.
- drivers/net/ethernet/stmicro/stmmac/stmmac_main.c
  - Looks ok

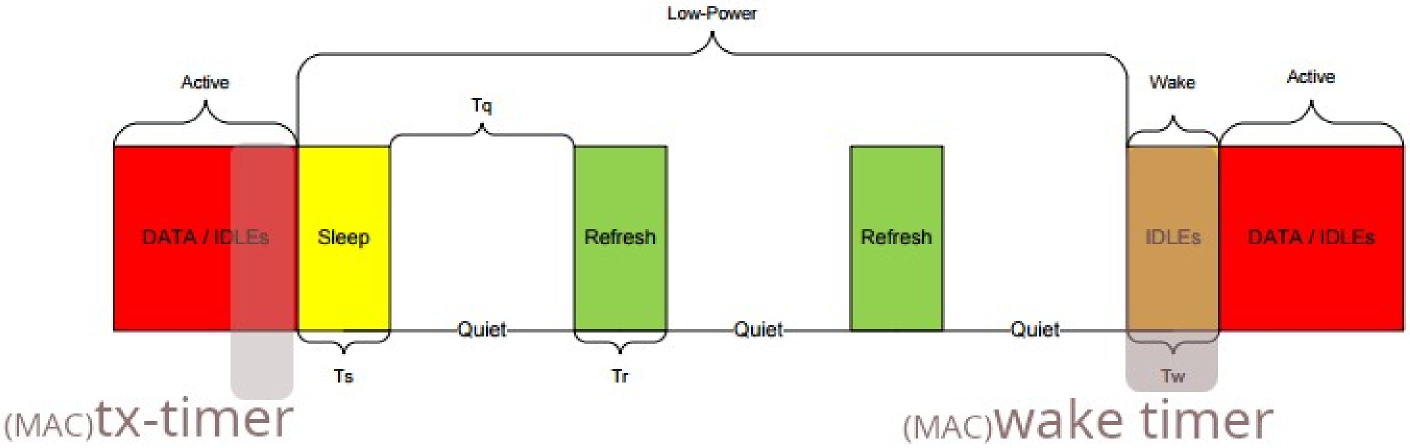
How kernel EEE support is expected to work

- MAC driver is attaching PHY device
- If MAC supports Low Power Idle mode it calls phy_support_eee()
- PHYlib framework is deciding if EEE can be enabled. If yes, phydev→enable_tx_lpi == true
- MAC driver should use phydev→enable_tx_lpi to configure LPI mode on link_up() or adjust_link()
- phydev→enable_tx_lpi or phy_init_eee() can be used

Typical bugs

- phy_init_eee() or phydev→enable_tx_lpi are not used on link_up() or link_adjust()
- EEE/LPI is configured only over ethtool interface, no phy_init_eee() or phydev→enable_tx_lpi are used
- LPI or Wake timers are not calculated against actual clock frequency
- LPI and Wake timers are set from the ethtool tx-timer value
- Without using phy_support_eee(), EEE will not be advertised.

EEE MDI (cable) view

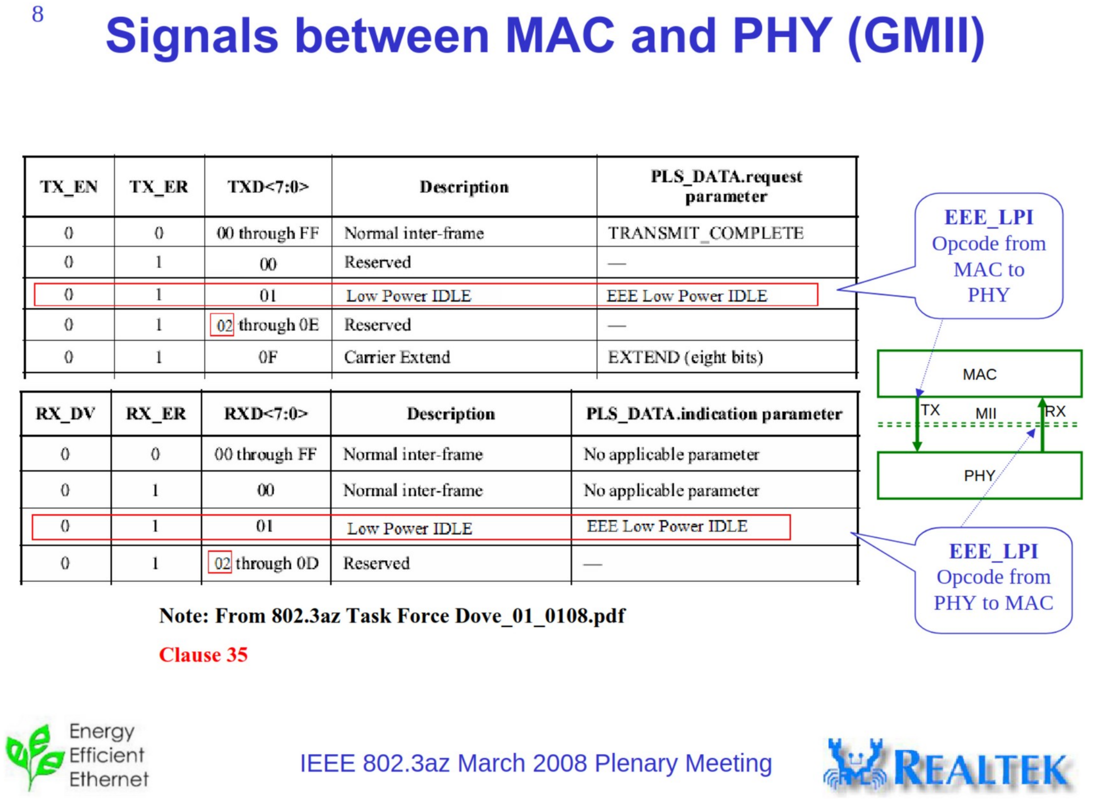


- Different components are involved. Signaling between MAC and PHY over xMII interface
- PHY timers: Ts, Tq, Tr (if wrong, link drop)
- MAC timers: tx-timer, wake timer (Tw)
- Tx-timer – idle time between last data and LPI mode: if too low – performance drop; too high – no energy savings
- wake timer – idle time between LPI and data: if too low – frame corruption, too high – performance drop

Fixing timers

- PHY related timers like Ts, Tq, Tr need potentially better equipment and better PHY documentations. Related registers are usually not documented.
- MAC related timers like tx-timer and wake timer (Tw) are usually part of MAC drivers and easier to debug with not expensive scope by measuring xMII lines.

Signaling Low Power Idle



8

Signals between MAC and PHY (GMII)

TX_EN	TX_ER	TXD<7:0>	Description	PLS_DATA.request parameter
0	0	00 through FF	Normal inter-frame	TRANSMIT_COMPLETE
0	1	00	Reserved	—
0	1	01	Low Power IDLE	EEE Low Power IDLE
0	1	02 through 0E	Reserved	—
0	1	0F	Carrier Extend	EXTEND (eight bits)

RX_DV	RX_ER	RXD<7:0>	Description	PLS_DATA.indication parameter
0	0	00 through FF	Normal inter-frame	No applicable parameter
0	1	00	Normal inter-frame	No applicable parameter
0	1	01	Low Power IDLE	EEE Low Power IDLE
0	1	02 through 0D	Reserved	—

EEE_LPI Opcode from MAC to PHY

EEE_LPI Opcode from PHY to MAC

Note: From 802.3az Task Force Dove_01_0108.pdf

Clause 35

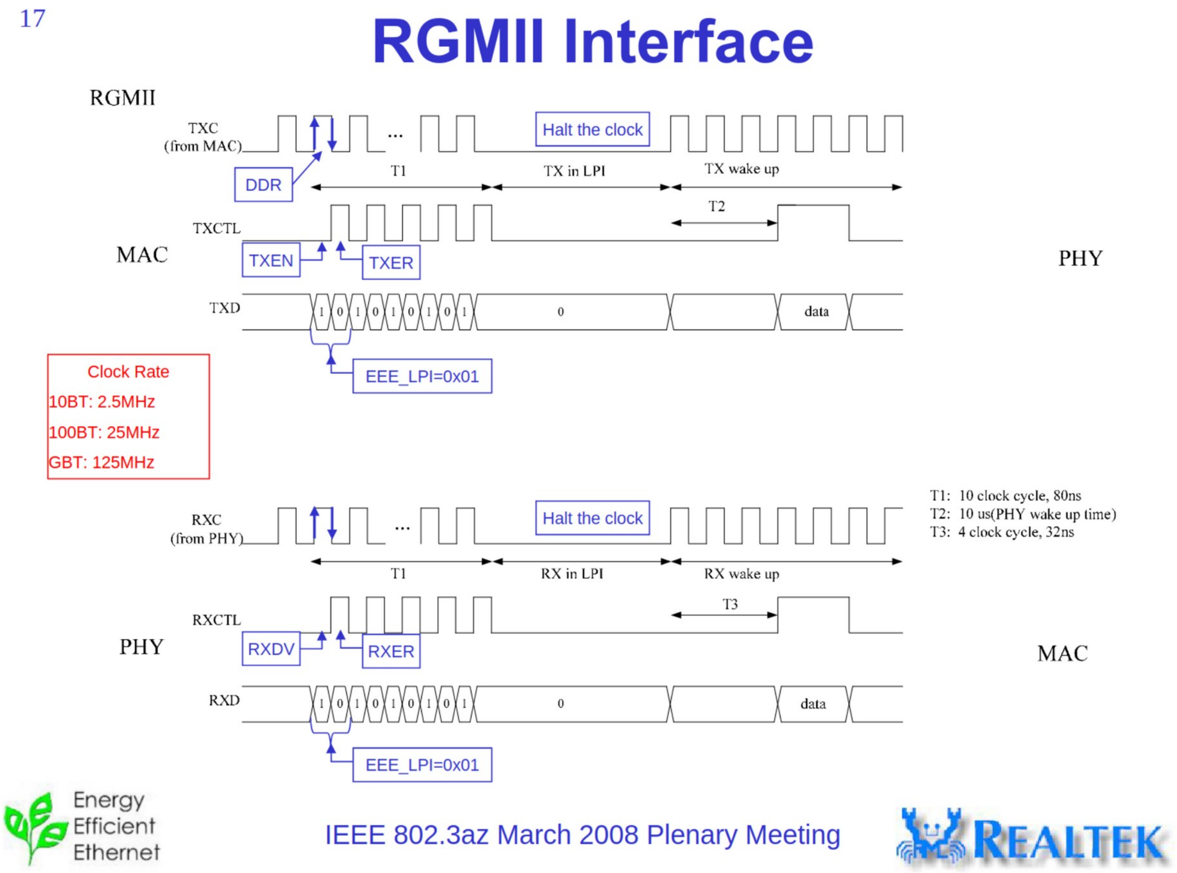
Energy Efficient Ethernet

IEEE 802.3az March 2008 Plenary Meeting

REALTEK

https://www.ieee802.org/3/az/public/may08/chou_03_0508.pdf

Signaling Low Power Idle



17

RGMII Interface

Clock Rate
10BT: 2.5MHz
100BT: 25MHz
GBT: 125MHz

T1: 10 clock cycle, 80ns
T2: 10 us(PHY wake up time)
T3: 4 clock cycle, 32ns

IEEE 802.3az March 2008 Plenary Meeting

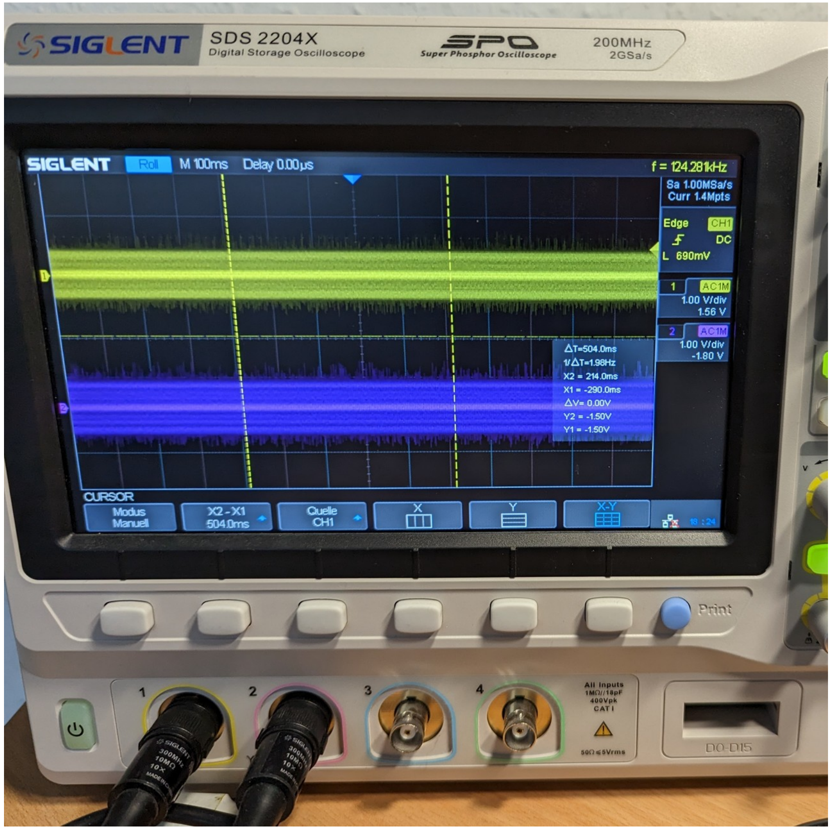
https://www.ieee802.org/3/az/public/may08/chou_03_0508.pdf

Beware of PHYs with PHY mode EEE!!

- Make sure LPI signal on xMII interface do actually correspond to the idle on MDI (cable).
- If timing changes do not affect end result on MDI side, may be you have PHY mode EEE.
- There are many PHYs from different PHY vendors supporting this mode (Atheros, Broadcom, Realtek,..)
- It is not always publicly documented.

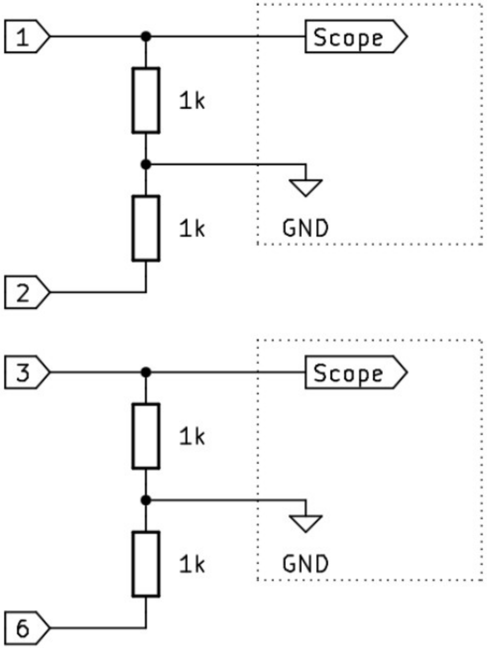
Oscilloscope

- No upper budget limit
- Let’s reduce budget to get more hackers on board :)
- No 1000BaseT or 100BaseT decoder support is need
- It is enough to presence of the signal, not exact form of it.
- 2x channels is enough



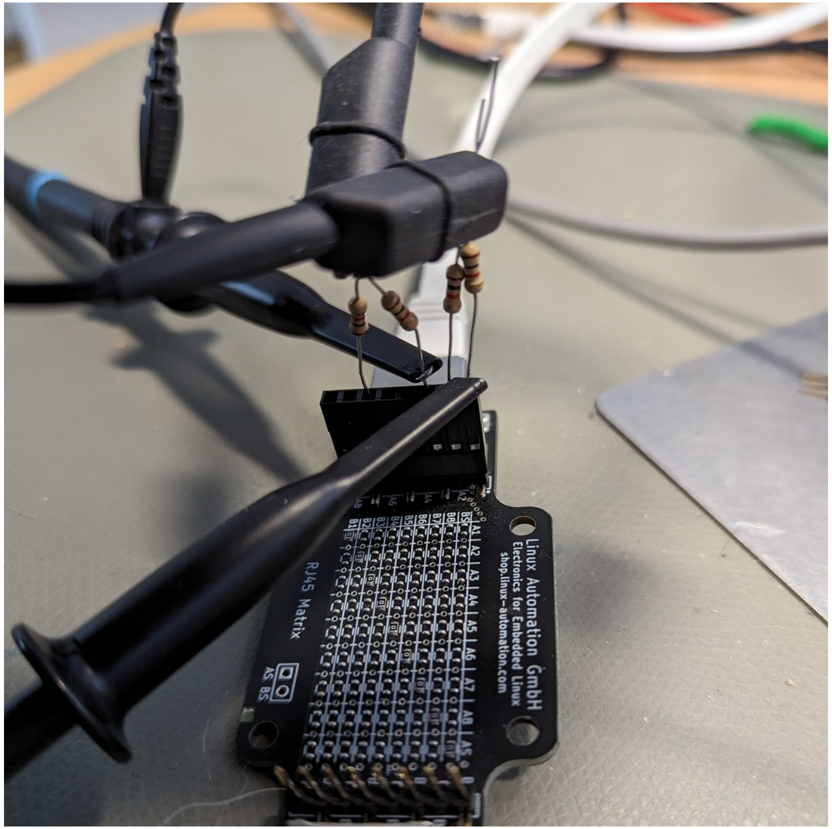
Probes

- Normally - differential probe is needed
- But we are doing low budget setup, so let’s use bunch of 1kOhm
- If you know your HW setup it should be less risky to do so.
- Be careful to avoid HW damage!!
- Make sure no PoE or PoDL is in use!!!



Probes – reducing noise

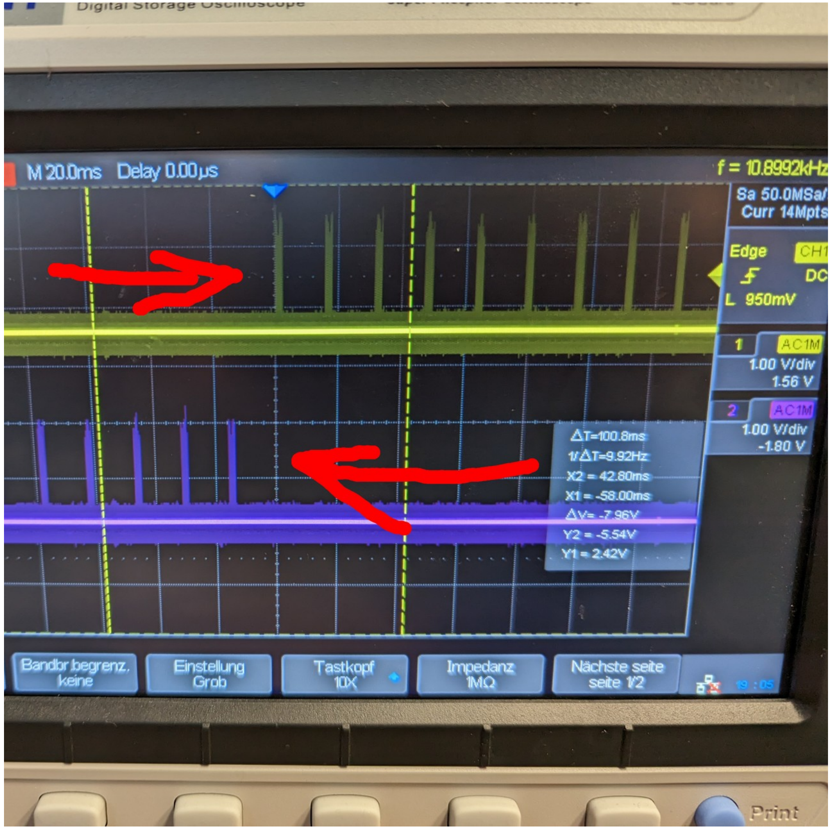
- Without differential probes and too long wires there will be too much noise
- Optimizing it a bit will make this setup more usable.



https://shop.linux-automation.com/lxa_tools-S02-R01-V01-C00

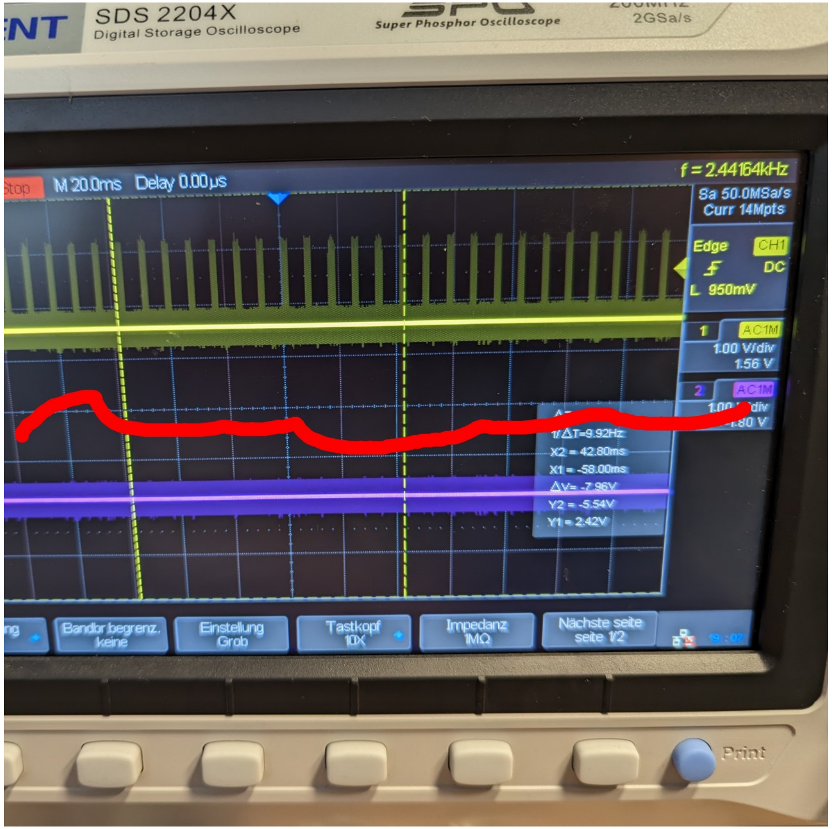
Get MDI-X under control

- First attach only one device
- ip l s dev eth0 up
- If pulse on both channels auto MDI-X is active
- Disable it to make things predictable



ethtool -s eth0 advertise 0x008 mdix on

- Some pre-configuration
- “advertise 0x008” – advertise only 100BaseT/Full. It is easier to debug with low budget setup
- “mdix on” – force MDI-X configuration. Not auto MDI-X. Link partner should stay Auto or depending on cable “mdix off”
- If mdix off/on is not working. Send patches :)

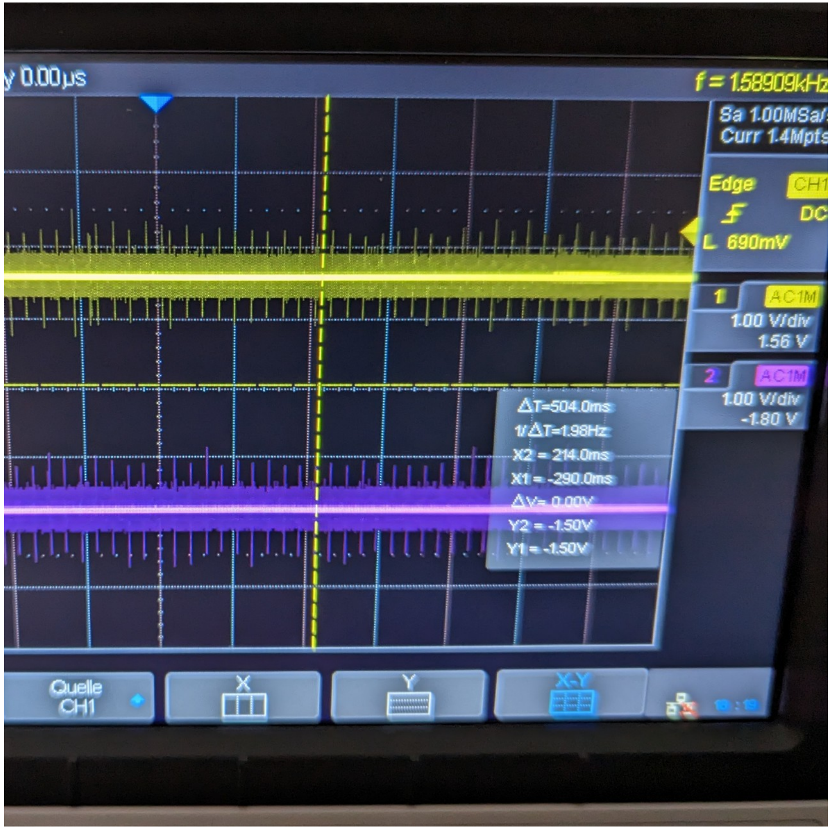


ethtool --show-eee eth0

```
EEE settings for eth0:
        EEE status: enabled - active
        Tx LPI: 500040 (us)
        Supported EEE link modes:  100baseT/Full
                                   1000baseT/Full
        Advertised EEE link modes:  100baseT/Full
                                    1000baseT/Full
        Link partner advertised EEE link modes:  100baseT/Full
                                                 1000baseT/Full
```

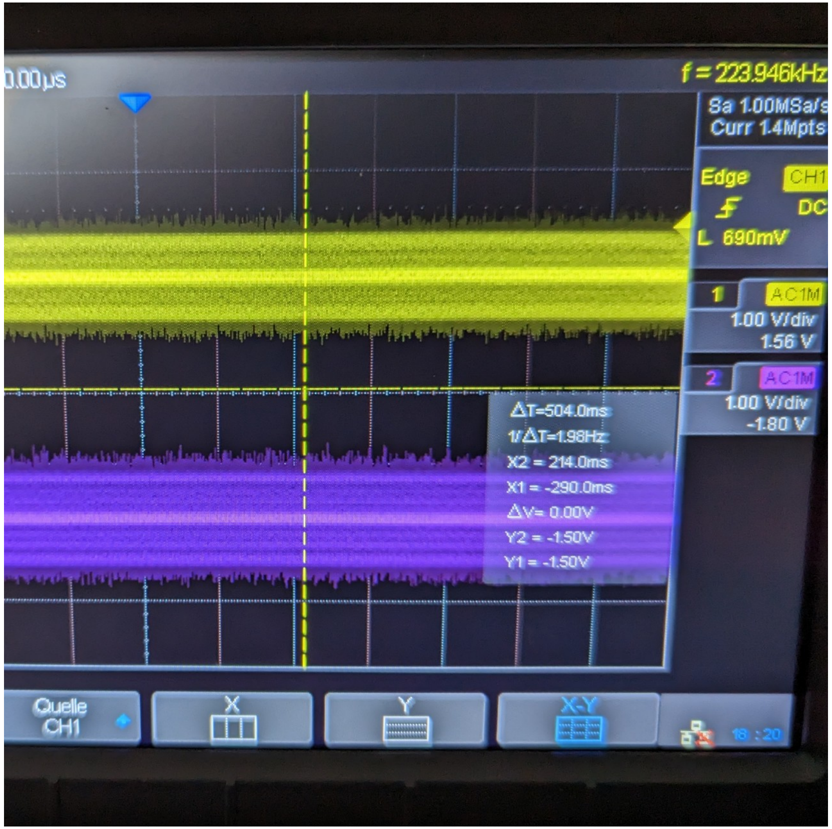
ethtool --set-eee eth0 eee on

- If “EEE status: enabled – active”. We should get some how similar picture
- There are no active transfers on the link

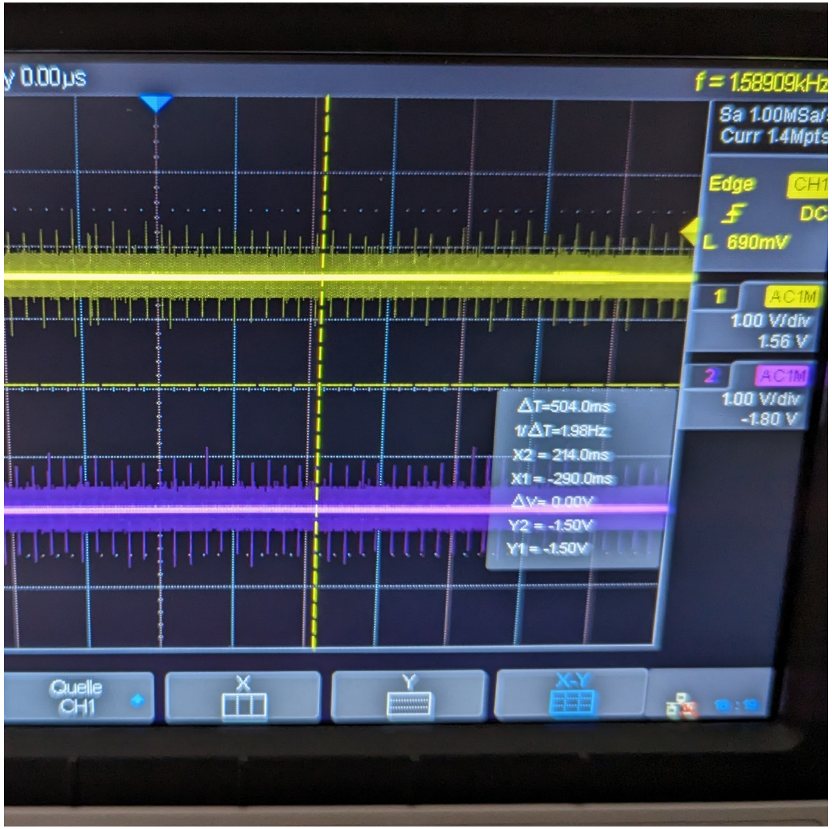


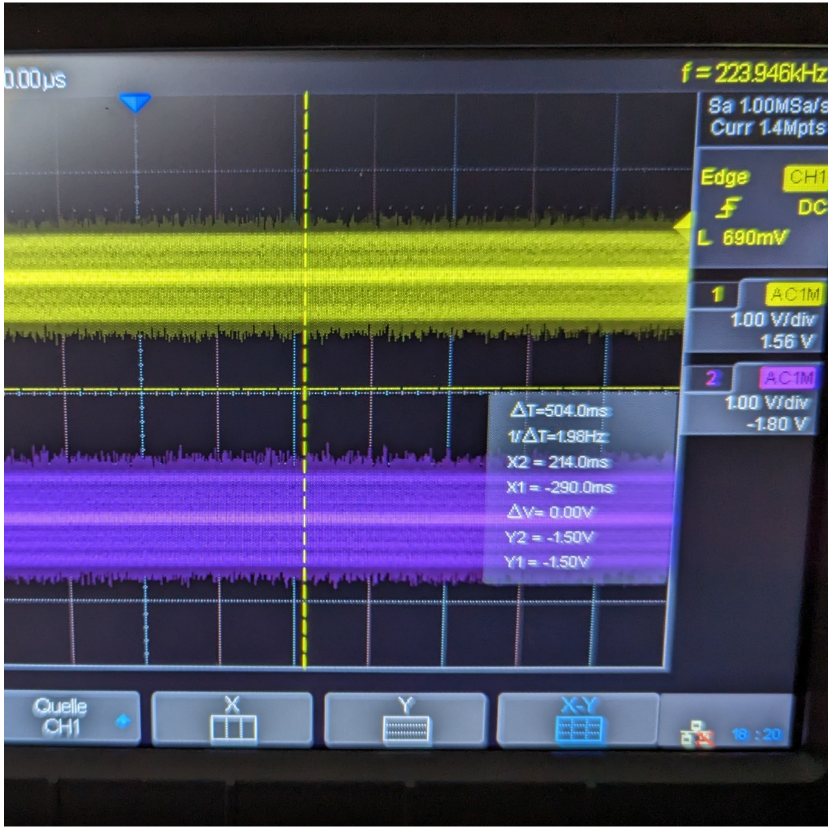
ethtool --set-eee eth0 eee off

- If “EEE status: disabled”
- Or “EEE status: enabled - inactive”



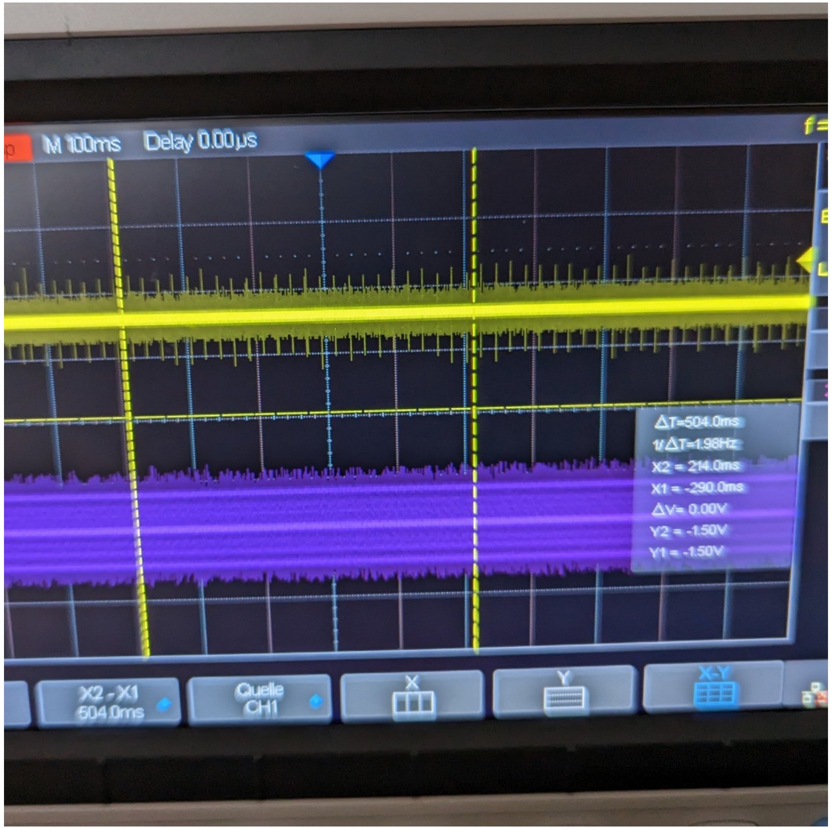
ethtool --set-eee eth0 eee on <> off





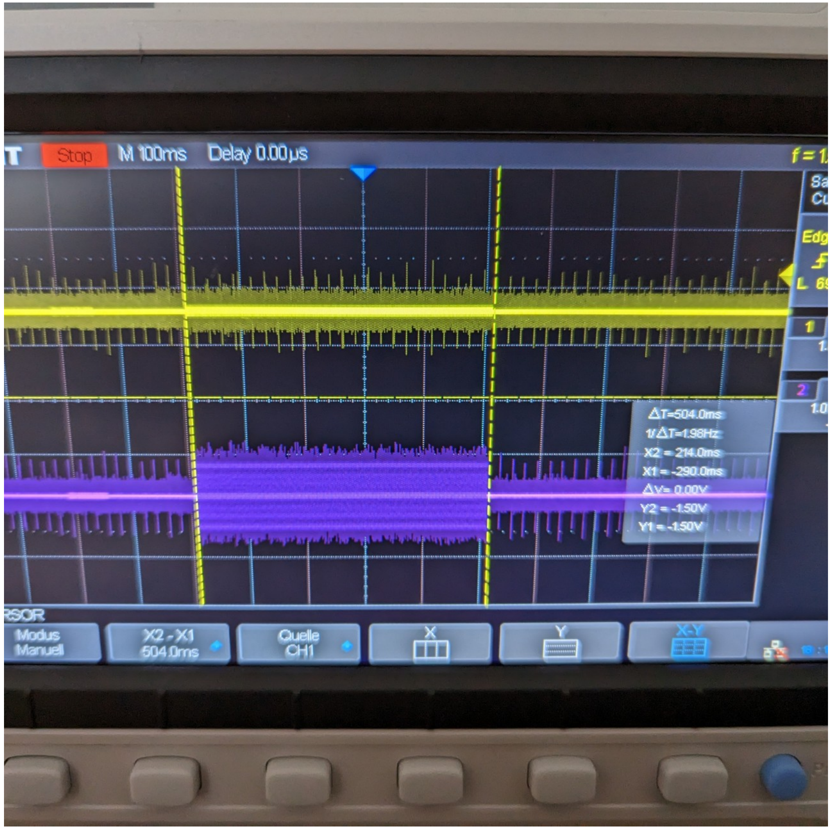
ethtool --set-eee eth0 tx-lpi off

- LPI – Low Power Idle
- It is possible to partially disable EEE
- tx-lpi off – disable TX LPI on local side
- By default - tx-lpi on



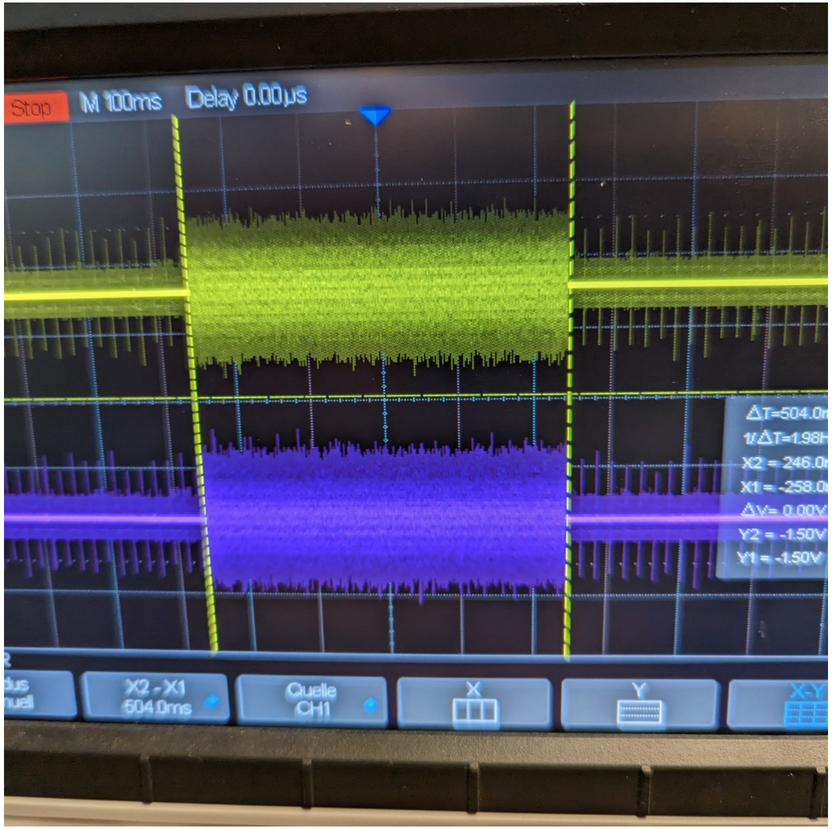
ethtool --set-eee eth0 tx-timer 500000

- Tx-timer – how long we should not enter LPI after transmission
- Send some packet to test this state. For example: mausezahn eth0 -c 1 -a rand -p 64



ethtool -s eth0 advertise 0x020

- Compare if things look similar with 1000BaseT
- advertise 0x020 – advertise support only for 1000BaseT/Full
- Note: with 1Gbit same ping will appear on both channels



Thank you!

Questions?

Pengutronix.

https://www.pengutronix.de