Adding support for PoE

Köry Maincent
kory.maincent@bootlin.com

Köry Maincent

- Embedded Linux engineer at Bootlin
  - Embedded Linux, U-Boot, Linux kernel, Yocto, Buildroot **expertise**
  - **Development**, consulting and training
  - Strong open-source focus
- Open-source contributor
- Living in **Toulouse**, France

Agenda

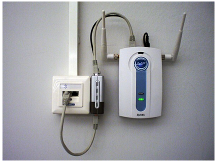
- Introduction to Power over Ethernet
  - Principles and advantages
  - IEEE 802.3 Standards
  - Hardware description
  - PSE Basic processes
  - Power classification and class detection
- Adding support for PoE in Linux
  - What's existing?
  - Linux Netlink UAPI
  - Linux Devicetree Bindings
  - Usage of regulators framework
- Future features in the pipeline

What is Power over Ethernet

Köry Maincent
kory.maincent@bootlin.com

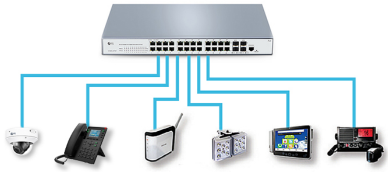
What is Power over Ethernet

- Power a device on twisted Ethernet pairs alongside data
- Remove the needs of power cable for Ethernet devices like VoIP Phone, IP Camera, router ...
- Power can be applied over 2 Ethernet pairs or 4 Ethernet pairs (PoE4 - 802.3bt standard)



PoE advantages

- Cost saving: no electricians needed anymore!
- Simple and flexible: one cable!
- No AC external power needed -> safer and no need for electrical certification!
- Reset easily the powered devices

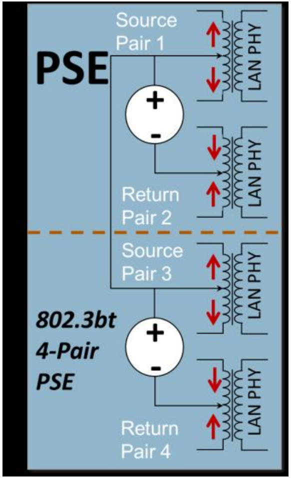


IEEE 802.3 Standards

- Clause 33
  - Power over 2 Ethernet pairs
  - IEEE 802.3af-2003
    - Power up to 15.4 W
  - IEEE 802.3at-2009
    - Power up to 25.5 W for Type 2 and Class 4 devices
  - Initially named Power via Media Dependent Interface
  - Renamed to Power over Ethernet in IEEE 802.3 2022
- Clause 145
  - Power over 4 Ethernet pairs
  - IEEE 802.3bt-2018
    - Power up to 71.3 W for Type 4 and Class 8 devices

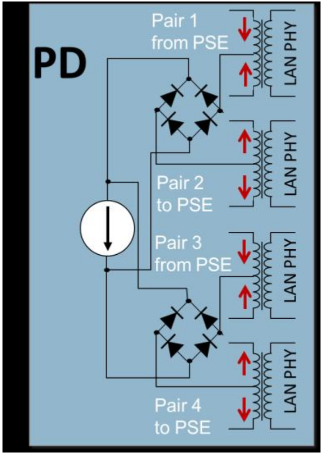
Hardware Power Sourcing Equipment (PSE) description

- Power source over 2/4 Ethernet pairs
  - One/two wire pair(s) for source current
  - One/two wire pair(s) for return current
- Ethernet are designed to work over long distance
  - Isolation transformers may be used on each pair to avoid disturbance



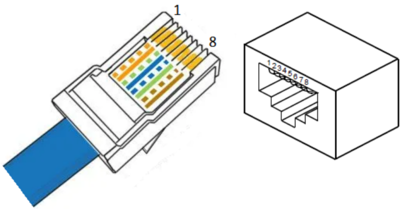
Hardware Powered Devices (PD) description

- Always 4 ports devices to be able to receive power on either pairset in either polarity
- *Single signature* if bridge outputs are combined, dual signature if separated (less common)



Power Interface (PI) pairsets and polarity

- RJ45 (8P8C)
  - 8 connectors -> 4 pairs (4 colors)
  - Straight-throught cable or Crossover cable



Power Interface (PI) pairsets and polarity

- RJ45 (8P8C)
  - 8 connectors -> 4 pairs (4 colors)
  - Straight-throught cable or Crossover cable
- Pairset: one wire pair for source current, one wire pair for return current
- 4 configurations possible with PoE2, plus two considering PoE4
- Pairsets references to Mode A or Mode B on PD side

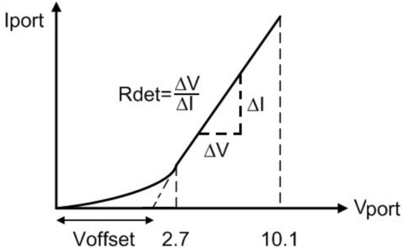
Conductor	Alternative A (MDI-X)	Alternative A (MDI)	Alternative B(X)	Alternative B(S)
1	Negative V_{PSE}	Positive V_{PSE}	—	—
2	Negative V_{PSE}	Positive V_{PSE}	—	—
3	Positive V_{PSE}	Negative V_{PSE}	—	—
4	—	—	Negative V_{PSE}	Positive V_{PSE}
5	—	—	Negative V_{PSE}	Positive V_{PSE}
6	Positive V_{PSE}	Negative V_{PSE}	—	—
7	—	—	Positive V_{PSE}	Negative V_{PSE}
8	—	—	Positive V_{PSE}	Negative V_{PSE}

PSE Basic processes and constraints

- Ethernet Devices plugged
  - Discriminating Powered Devices from other devices that might be damaged if PoE voltages were applied
  - Assessing the basic power requirements
  - Conducting mutual discovery and power negotiation
  - Supporting surge (or inrush) power required to start up the PD
- Powered Devices at runtime
  - Supporting peak power demands from the PD
  - Supporting link-layer (LLPD) for specific configurations
  - Reacting to PD's that are drawing more power than they should
  - Supporting unbalanced load currents between pairsets
  - Limiting maximum possible current for safety
- Ethernet Devices unplugged
  - Remove power quickly enough

PoE detection and classification

- Detection
  - Periodic low-voltage pulse between 2.8V and 10V
  - Measurement of Rdet



PoE detection and classification

- Detection
  - Periodic low-voltage pulse between 2.8V and 10V
  - Measurement of Rdet
- Classification
  - Series of one to five short pulses called “class events” between 15.5V and 20.5V
  - Current draw variation of the PD on these pulses

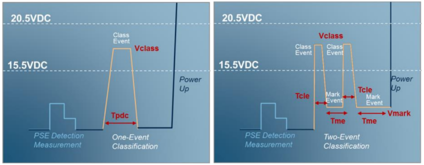


Figure 2.6 PD Classification under the **802.3at** Specification

PoE detection and classification

- Detection
  - Periodic low-voltage pulse between 2.8V and 10V
  - Measurement of Rdet
- Classification
  - Series of one to five short pulses called "class events" between 15.5V and 20.5V
  - Current draw variation of the PD on these pulses
  - One long first class event for Type 3 and 4 PSEs (Class > 4)

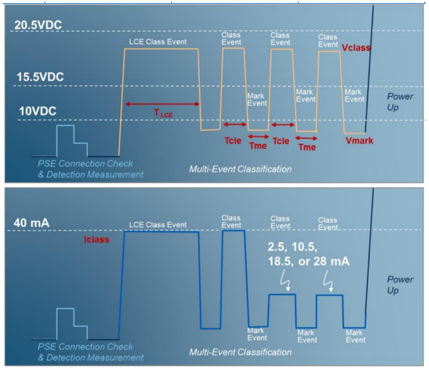


Figure 2.7 PD Classification under the 802.3bt Specification

PoE classification pulses

PD Class	Events 1 & 2	Events 3-5	Power Request at the PD	Units
Class 1	10.5 mA		3.84	Watts Total on 2-Pairs or 4-Pairs
Class 2	18.5 mA		6.5	
Class 3	28.0 mA		13.0	
Class 4	40.0 mA		25.5	
Class 5	40.0 mA	2.5 mA	40.0	Watts Total on 4-Pairs
Class 6	40.0 mA	10.5 mA	51.0	
Class 7	40.0 mA	18.5 mA	62.0	
Class 8	40.0 mA	28.0 mA	71.3	
Class 1 (Dual)	10.5 mA	2.5 mA	3.94	Watts per Pairset
Class 2 (Dual)	18.5 mA	2.5 mA	6.5	
Class 3 (Dual)	28.0 mA	2.5 mA	13.0	
Class 4 (Dual)	40.0 mA	2.5 mA	25.5	
Class 5 (Dual)	40.0 mA	28.0 mA	35.6	

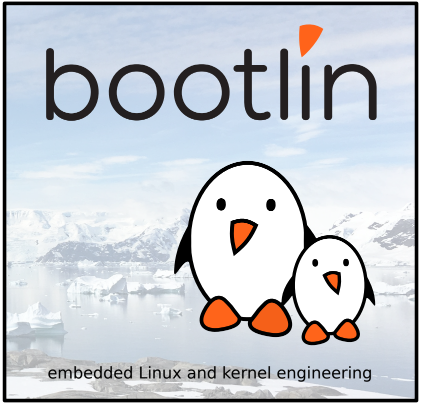
PoE classes

Power levels available[40][41]

Class	Usage	Classification current (mA)	Power range at PD (W)	Max power from PSE (W)	Class description
0	Default	0–5	0.44–12.94	15.4	Classification unimplemented
1	Optional	8–13	0.44–3.84	4.00	Very Low power
2	Optional	16–21	3.84–6.49	7.00	Low power
3	Optional	25–31	6.49–12.95	15.4	Mid power
4	Valid for Type 2 (802.3at) devices, not allowed for 802.3af devices	35–45	12.95–25.50	30	High power
5	Valid for Type 3 (802.3bt) devices	36–44 & 1–4	40 (4-pair)	45	
6		36-44 & 9–12	51 (4-pair)	60	
7	Valid for Type 4 (802.3bt) devices	36–44 & 17–20	62 (4-pair)	75	
8		36–44 & 26–30	71.3 (4-pair)	99	

PoE Linux support

Köry Maincent
kory.maincent@bootlin.com



What's existing

- poed - https://github.com/dentproject/poed
  - Userspace tool written in python
  - Need to install python on the target
  - Only support pd69200 controller
- No generic support of PoE, each constructor has its own example of code
- https://www.ti.com/product/TPS23881#software-development

State of Linux - Power over Data Line

- PoDL (IEEE 802.3 clause 30) is like PoE for Single-Pair Ethernet
- PoDL was added to Linux at the end of 2022 thanks to Oleksij Rempel
- PoDL has similar behavior as PoE in its PSE management
  - Core PSE framework support
  - Netlink UAPI description following the IEEE 802.3 standards
  - Ethtool PoDL support
- Simple PSE PoDL regulator driver and bindings added

Linux support fo PoE

- PoE development not yet merged mainline, currently in its ninth version
- https://lore.kernel.org/r/20240417-feature_poe-v9-0-242293fd1900@bootlin.com
- Adds support for two PSE controller drivers
  - Microchip PD692x0
  - TI TPS23881

Linux PoE Netlink UAPI

- Enhance `netlink` UAPI with PoE messages following the standards

```
--- a/include/uapi/linux/ethtool_netlink.h
+++ b/include/uapi/linux/ethtool_netlink.h
@@ -895,6 +895,9 @@ enum {
        ETHTOOL_A_PODL_PSE_ADMIN_STATE,         /* u32 */
        ETHTOOL_A_PODL_PSE_ADMIN_CONTROL,       /* u32 */
        ETHTOOL_A_PODL_PSE_PW_D_STATUS,         /* u32 */
+       ETHTOOL_A_C33_PSE_ADMIN_STATE,          /* u32 */
+       ETHTOOL_A_C33_PSE_ADMIN_CONTROL,        /* u32 */
+       ETHTOOL_A_C33_PSE_PW_D_STATUS,          /* u32 */

--- a/include/uapi/linux/ethtool.h
+++ b/include/uapi/linux/ethtool.h
+enum ethtool_c33_pse_admin_state {
+       ETHTOOL_C33_PSE_ADMIN_STATE_UNKNOWN = 1,
+       ETHTOOL_C33_PSE_ADMIN_STATE_DISABLED,
+       ETHTOOL_C33_PSE_ADMIN_STATE_ENABLED,
+};
+
+enum ethtool_c33_pse_pw_d_status {
+       ETHTOOL_C33_PSE_PW_D_STATUS_UNKNOWN = 1,
+       ETHTOOL_C33_PSE_PW_D_STATUS_DISABLED,
+       ETHTOOL_C33_PSE_PW_D_STATUS_SEARCHING,
+       ETHTOOL_C33_PSE_PW_D_STATUS_DELIVERING,
+       ETHTOOL_C33_PSE_PW_D_STATUS_TEST,
+       ETHTOOL_C33_PSE_PW_D_STATUS_FAULT,
+       ETHTOOL_C33_PSE_PW_D_STATUS_OTHERFAULT,
+};
```

Linux PSE PoDL devicetree binding

- PoDL PI composed by only one pair -> does not need much information
- The PSE port number is enough

```
pse_t1l1: ethernet-pse {
  compatible = "podl-pse-foo-controller";
  pse-supply = <&reg_t1l1>;
  #pse-cells = <1>;
};

&ethernet_phy1 {
  pses = <&pse_t1l1 0>;
}

&ethernet_phy2 {
  pses = <&pse_t1l1 1>;
}
```

Linux PSE PoE devicetree binding

- The PoE PI has more information due to the 4 pairs:
- PSE Port number, pairset configuration, polarity supported

```
ethernet-pse@20 {
  compatible = "ti,tps23881";
  reg = <0x20>;

  channels {
    #address-cells = <1>;
    #size-cells = <0>;

    phys0: channel@0 {
      reg = <0>;
    };
    phys1: channel@1 {
      reg = <1>;
    };
    phys2: channel@2 {
      reg = <2>;
    };
  };
  ...
```

```
  ...
  pse-pis {
    #address-cells = <1>;
    #size-cells = <0>;

    pse_pi0: pse-pi@0 {
      reg = <0>;
      #pse-cells = <0>;
      pairset-names = "alternative-a", "alternative-b";
      pairsets = <&phys0>, <&phys1>;
      polarity-supported = "MDI", "S";
      vpwr-supply = <&vpwr>;
    };
    pse_pi1: pse-pi@1 {
      reg = <1>;
      #pse-cells = <0>;
      pairset-names = "alternative-a";
      pairsets = <&phys2>;
      polarity-supported = "MDI";
      vpwr-supply = <&vpwr>;
    };
  };
};
&ethernet_phy0 {
  pses = <&pse_pi0>;
}
&ethernet_phy1 {
  pses = <&pse_pi1>;
}
```

Usage of regulator framework

- PSE is like a regulator but for Ethernet devices
- Add code complexity with function wrappers
- Benefit from the already written regulator API
  - Power limit
  - Power/voltage status
  - Temperature limit
  - Sysfs description
- New features like power priorities could also benefit regulator API

Usage of regulator framework

```
static int
devm_pse_pi_regulator_register(struct pse_controller_dev *pcdev,
                               char *name, int id)
{
...
        rdesc->id = id;
        rdesc->name = name;
        rdesc->type = REGULATOR_CURRENT;
        rdesc->ops = &pse_pi_ops;
        rdesc->owner = pcdev->owner;

        rinit_data->constraints.valid_ops_mask = REGULATOR_CHANGE_STATUS;
        rinit_data->supply_regulator = "vpwr";

        rconfig.dev = pcdev->dev;
        rconfig.driver_data = pcdev;
        rconfig.init_data = rinit_data;

        rdev = devm_regulator_register(pcdev->dev, rdesc, &rconfig);
...
        pcdev->pi[id].rdev = rdev;

        return 0;
}
```

Usage of regulator framework

```
psec->ps = devm_regulator_get_exclusive(pcdev->dev,
                                        rdev_get_name(pcdev->pi[index].rdev));
if (IS_ERR(psec->ps)) {
        ret = PTR_ERR(psec->ps);
        goto put_module;
}

ret = regulator_is_enabled(psec->ps);
if (ret < 0)
        goto regulator_put;

pcdev->pi[index].admin_state_enabled = ret;
```

Usage of regulator framework

```
static int pse_ethtool_c33_set_config(struct pse_control *psec,
                                      const struct pse_control_config *config)
{
	int err = 0;

	/* Look at admin_state_enabled status to not call regulator_enable
	 * or regulator_disable twice creating a regulator counter mismatch
	 */
	switch (config->c33_admin_control) {
	case ETHTOOL_C33_PSE_ADMIN_STATE_ENABLED:
		if (!psec->pcdev->pi[psec->id].admin_state_enabled)
			err = regulator_enable(psec->ps);
		break;
	case ETHTOOL_C33_PSE_ADMIN_STATE_DISABLED:
		if (psec->pcdev->pi[psec->id].admin_state_enabled)
			err = regulator_disable(psec->ps);
		break;
	default:
		err = -EOPNOTSUPP;
	}

	return err;
}
```

What's next?

Köry Maincent
kory.maincent@bootlin.com

Lack of a Linux Ethernet port abstraction

- Linux current state with this PoE development
  - The PHY driver `get`s the PSE PI
  - No way to get a PSE from a Network Interface Card driver
  - The PSE PIs regulator provider and consumer are currently the same
- In reality the PSE PIs are only related to the RJ45 ports
- Maxime Chevallier is working on the Linux Ethernet port abstraction

Next features

- Set and read Power limit
- Read power and/or voltage status
- Read PoE class
- Read failure reason
- Configure PSE port priority
- Configure persistent configuration

Demo time!

Köry Maincent
kory.maincent@bootlin.com

References

- References:
  - https://community.fs.com/article/power-over-ethernet-tutorial.html
  - https://en.wikipedia.org/wiki/Power_over_Ethernet
  - https://sifos.com/wp-content/uploads/IEEE-802.3-PoE-Technology-Overview-Sifos-Technologies.pdf
  - https://www.poepower.net/
- Sponsored by Dent Project

Questions? Suggestions? Comments?

Köry Maincent
kory.maincent@bootlin.com

https://bootlin.com/pub/conferences/2024/eoss/maincent-adding-support-for-poe/maincent-adding-support-for-poe.pdf