Quarkslab

Hydradancer

Using USB3 to improve USB hacking with Facedancer

Thiébaud Fuchs | kauwua | PTS 2024

What's USB

Universal Serial Bus: some facts

- USB 1.0 released in 1996
- Universal: power, data, "just works"
- Non-profit organization
- Apple, HP, Intel, Microsoft, Renesas, STMicroelectronics, Texas Instruments, ...

USB-C®

USB4®

HI SPEED CERTIFIED USB®

CERTIFIED USB® CHARGER

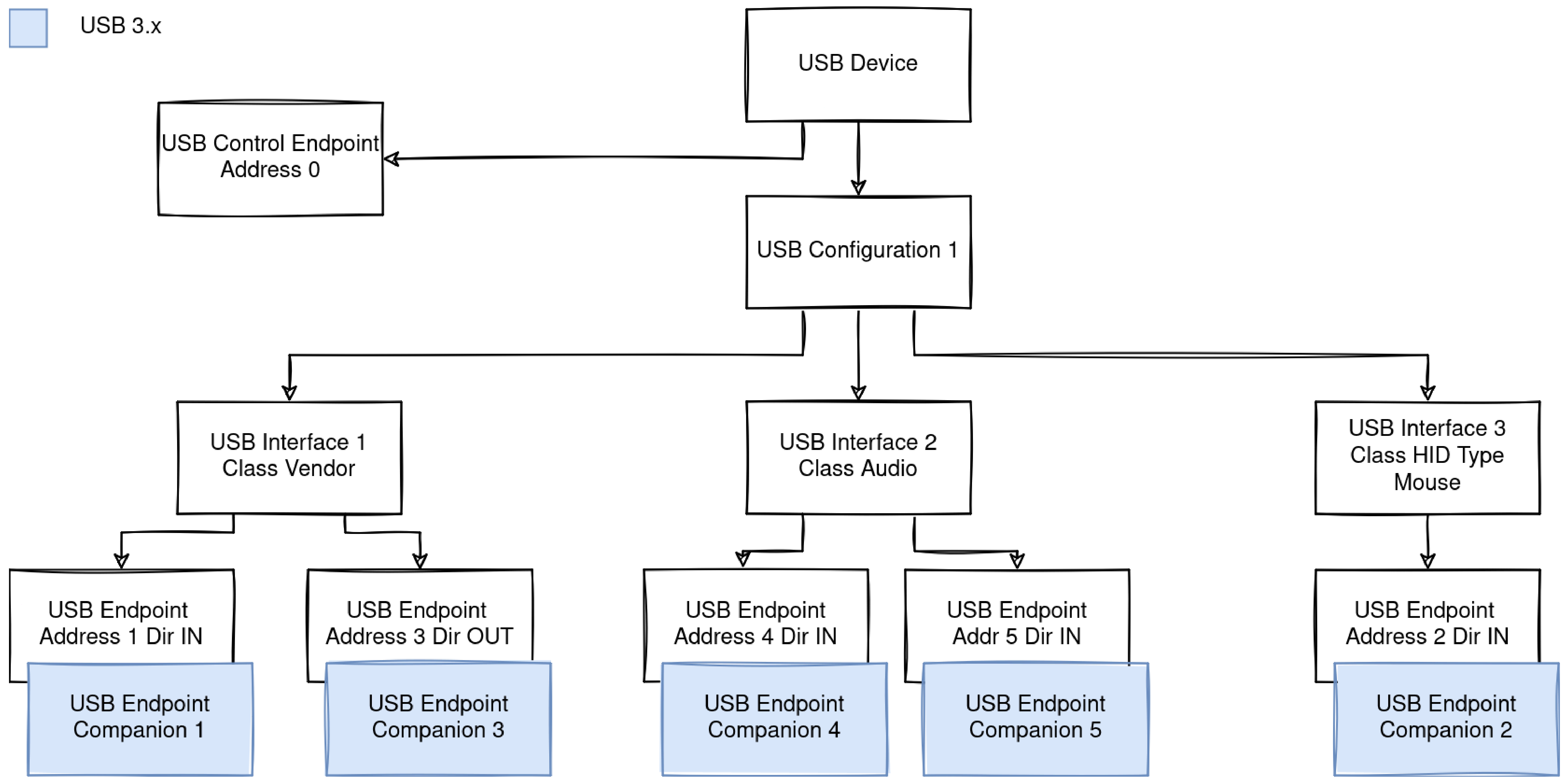
CERTIFIED USB 80Gbps™

Facedancer: creating USB peripherals in Python

Facedancer's history

- Created in 2012 by Travis Goodspeed
- Now maintained by Great Scott Gadgets (GreatFET, HackRF, ...)
- Recently released v3.0 with a new API !

USB Descriptors

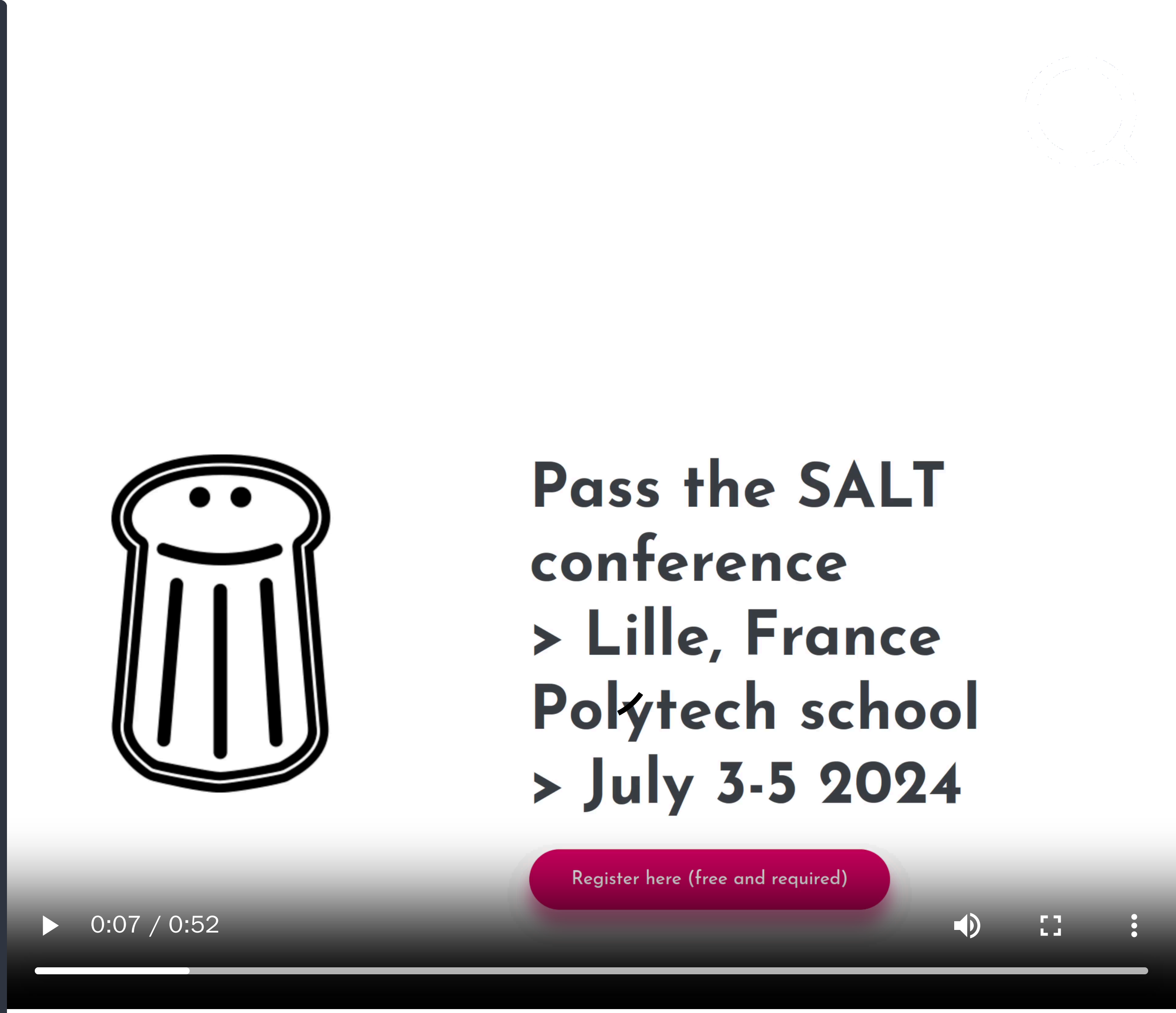

USB 3.x
USB Device
USB Control Endpoint Address 0
USB Configuration 1
USB Interface 1 Class Vendor
USB Interface 2 Class Audio
USB Interface 3 Class HID Type Mouse
USB Endpoint Address 1 Dir IN
USB Endpoint Companion 1
USB Endpoint Address 3 Dir OUT
USB Endpoint Companion 3
USB Endpoint Address 4 Dir IN
USB Endpoint Companion 4
USB Endpoint Addr 5 Dir IN
USB Endpoint Companion 5
USB Endpoint Address 2 Dir IN
USB Endpoint Companion 2

USB Descriptors

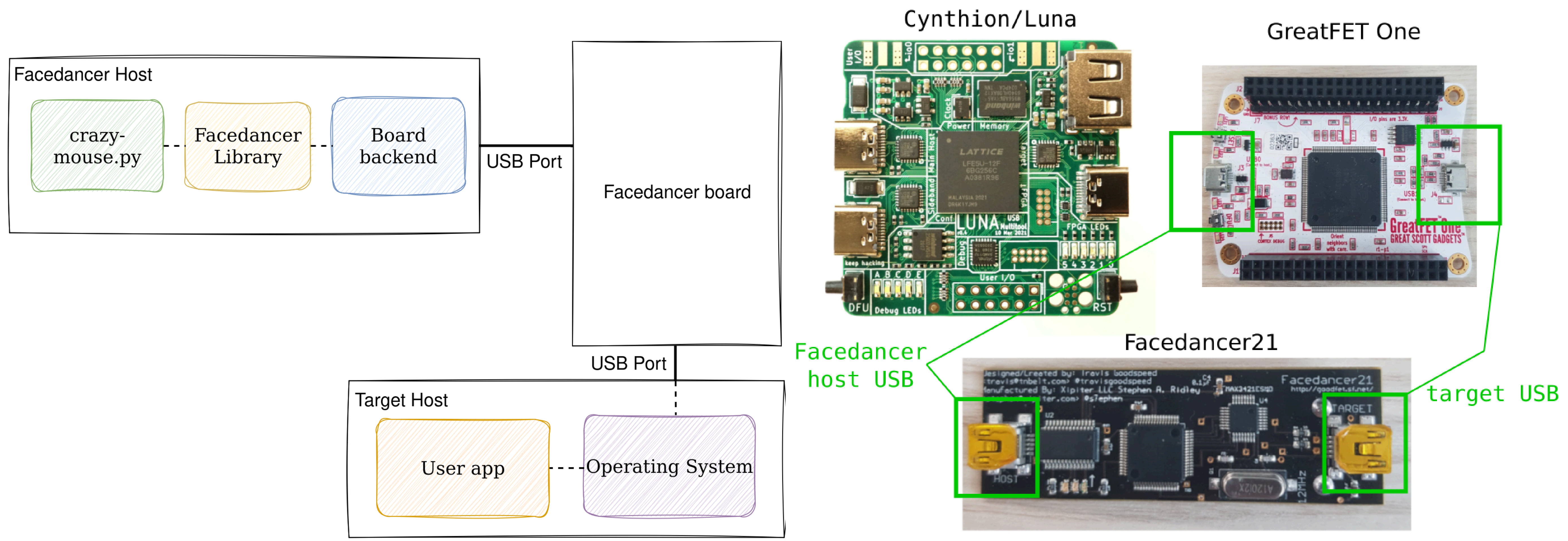
DEMO: USB mouse goes crazy

```python
class CrazyMouse(USBDevice):
    def __init__(self):
        super().__init__(
            vendor_id=0x610b,
            product_id=0x4653,
            product_string="Non-suspicious mouse"
        )

        configuration = USBConfiguration()
        self.add_configuration(configuration)

        interface = USBInterface()
        configuration.add_interface(interface)

        in_endpoint = USBEndpoint(number=3, direction=USBDirection.IN)
        interface.add_endpoint(in_endpoint)

    def handle_data_requested(self, endpoint: USBEndpoint):
        logging.info(f"Sending mouse data {data} on {endpoint}.")
```

```
$ pyt
```


Pass the SALT conference
> Lille, France
Polytech school
> July 3-5 2024

How it works



Facedancer Principle

Hydradancer: more stability and speed for Facedancer

USB protocol speeds

- USB2: LS (Low-speed, ~200KB/s), FS (Full-speed, ~1.5MB/s), HS (High-speed, ~50MB/s)
- USB3: SuperSpeed (5Gb/s), SuperSpeed+ (10Gb/s), ...
- USB4: up to 120Gb/s

Current limitations of Facedancer

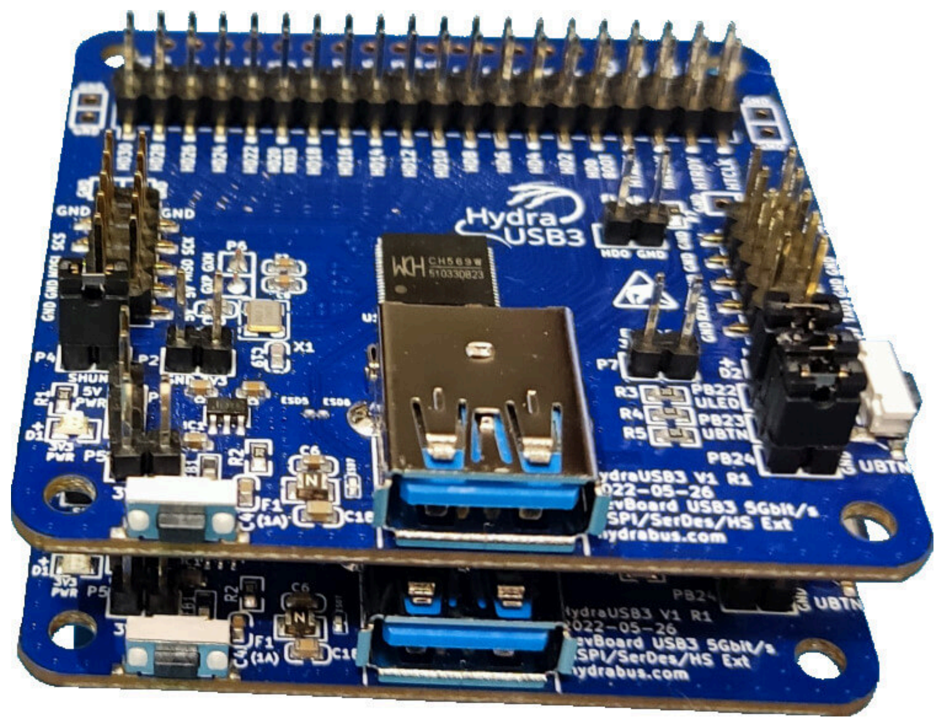
Board	Maximum speed	Number of endpoints (not EP0)	Host mode
Facedancer21/Raspdancer	USB2 Full-speed	EP1 OUT, EP2 IN, EP3 IN	yes
GreatFET One	USB2 Full-speed	3 IN / 3 OUT	yes
Hydradancer	USB2 High-speed	5 IN / 5 OUT	no
(Cynthion/LUNA)(delivery June 2024)	(USB2 High-speed)	(15 IN / 15 OUT)	(yes)

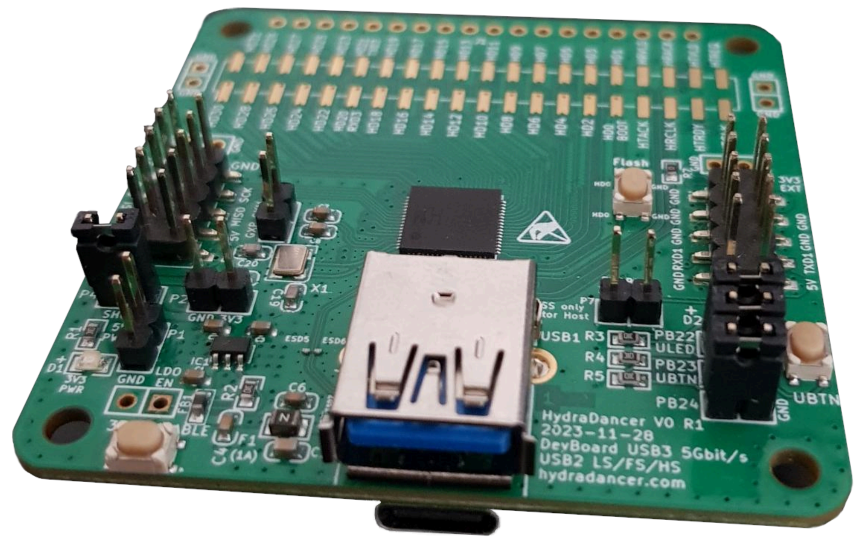
Facedancer backends functionalities

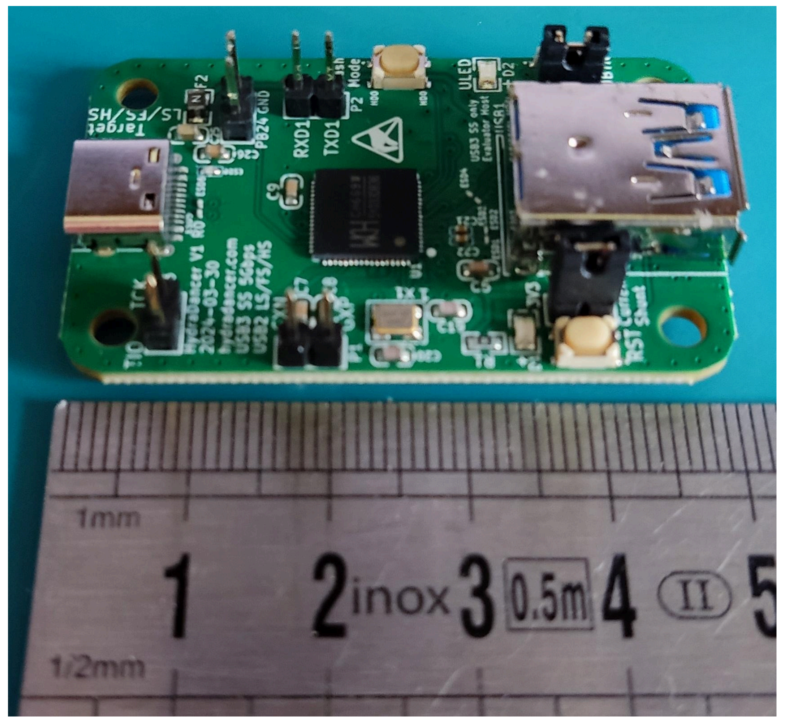
	Write average estimate	Read average estimate
GreatFET One Full-speed (one by one) (git-v2021.2.1-64-g2409575 firmware)	32.42±0.85 KB/s	33.07±1.10 KB/s
Facedancer21 Full-speed (2014-07-05 firmware)	0.697±0.000 KB/s	0.682±0.000 KB/s

Facedancer backends speeds

From HydraUSB3 to the new Hydradancer dongle

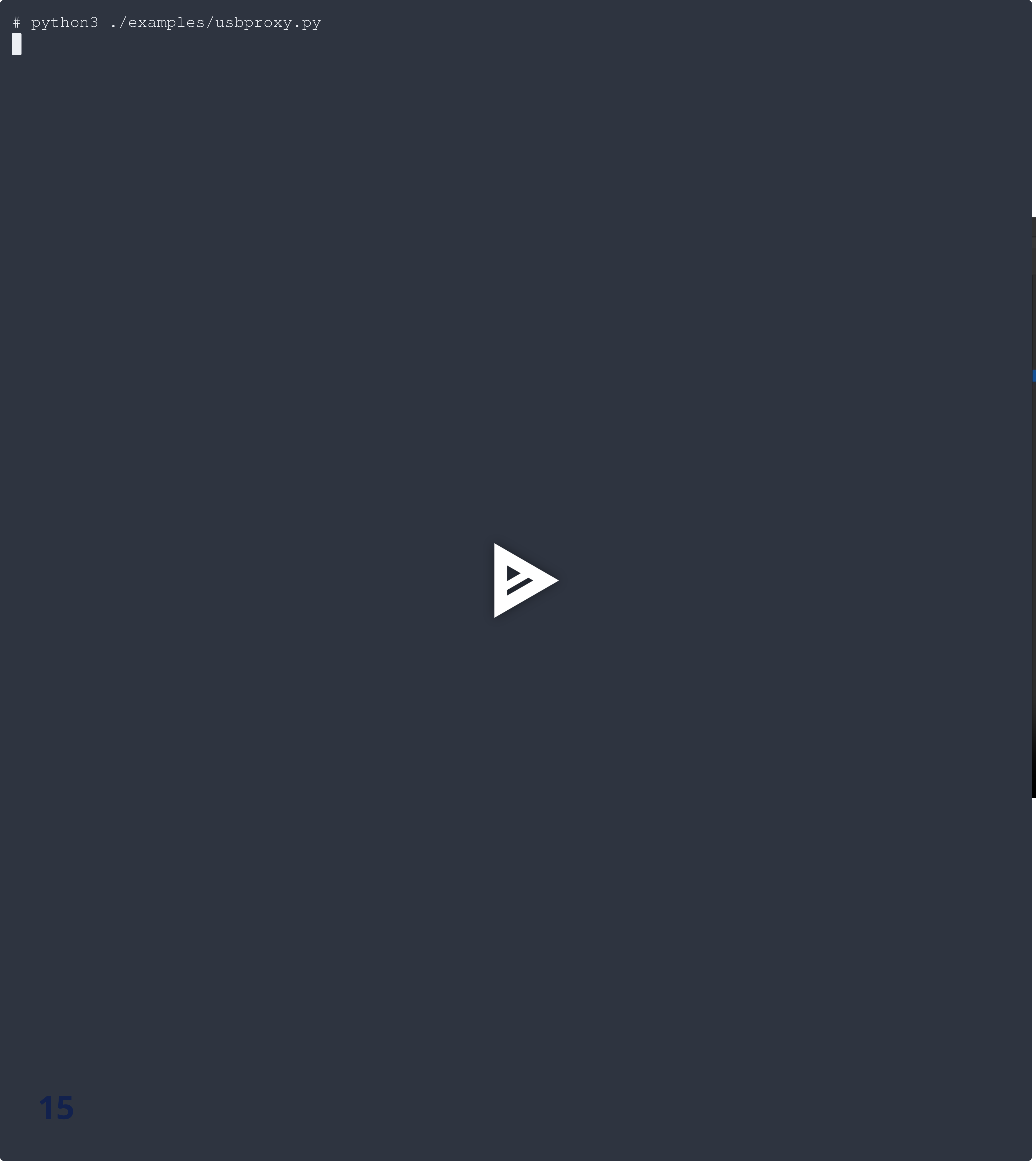




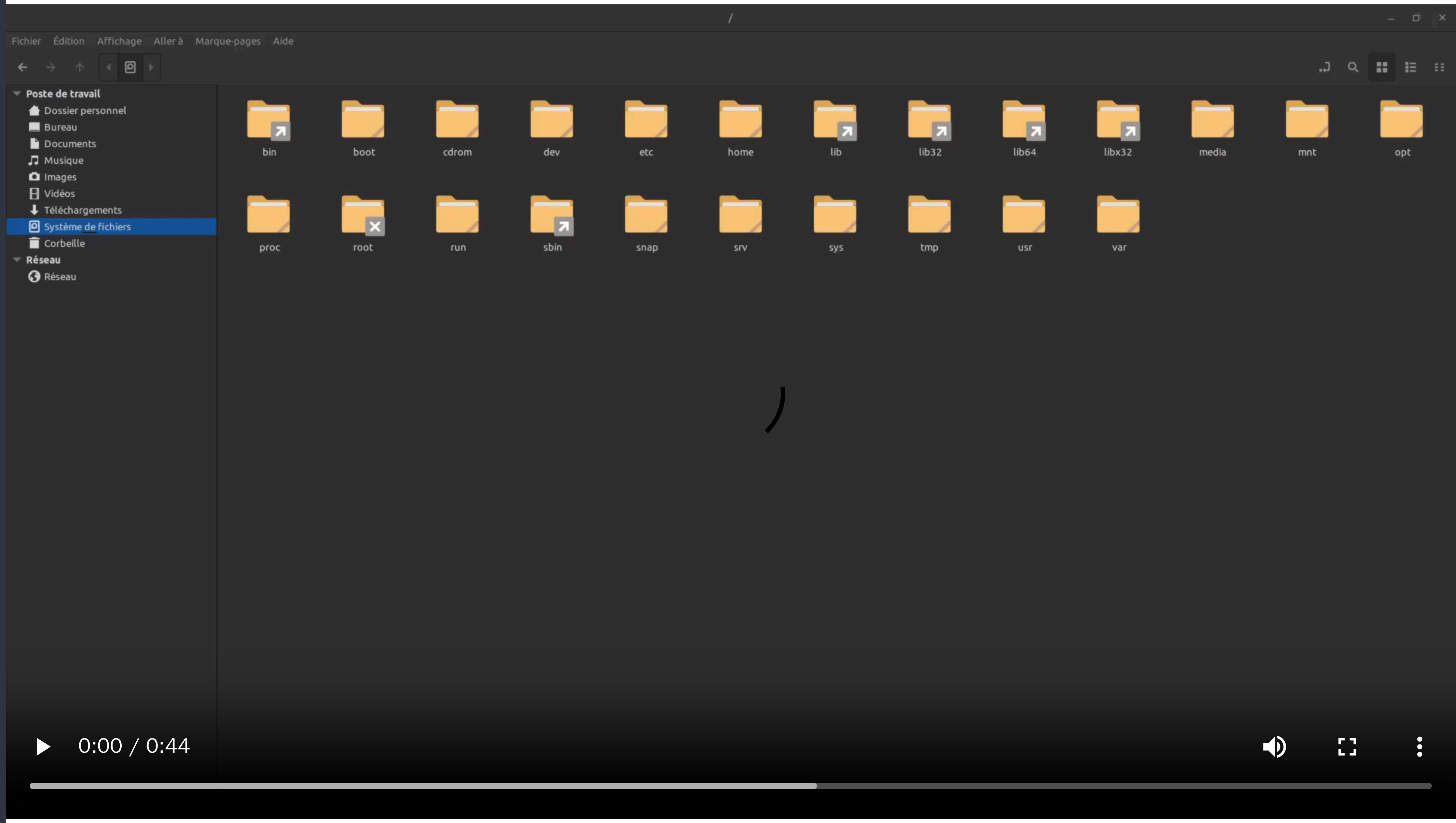


Boards created by Benjamin Vernoux. Dual-HydraUSB3/Hydradancer prototype/Hydradancer dongle

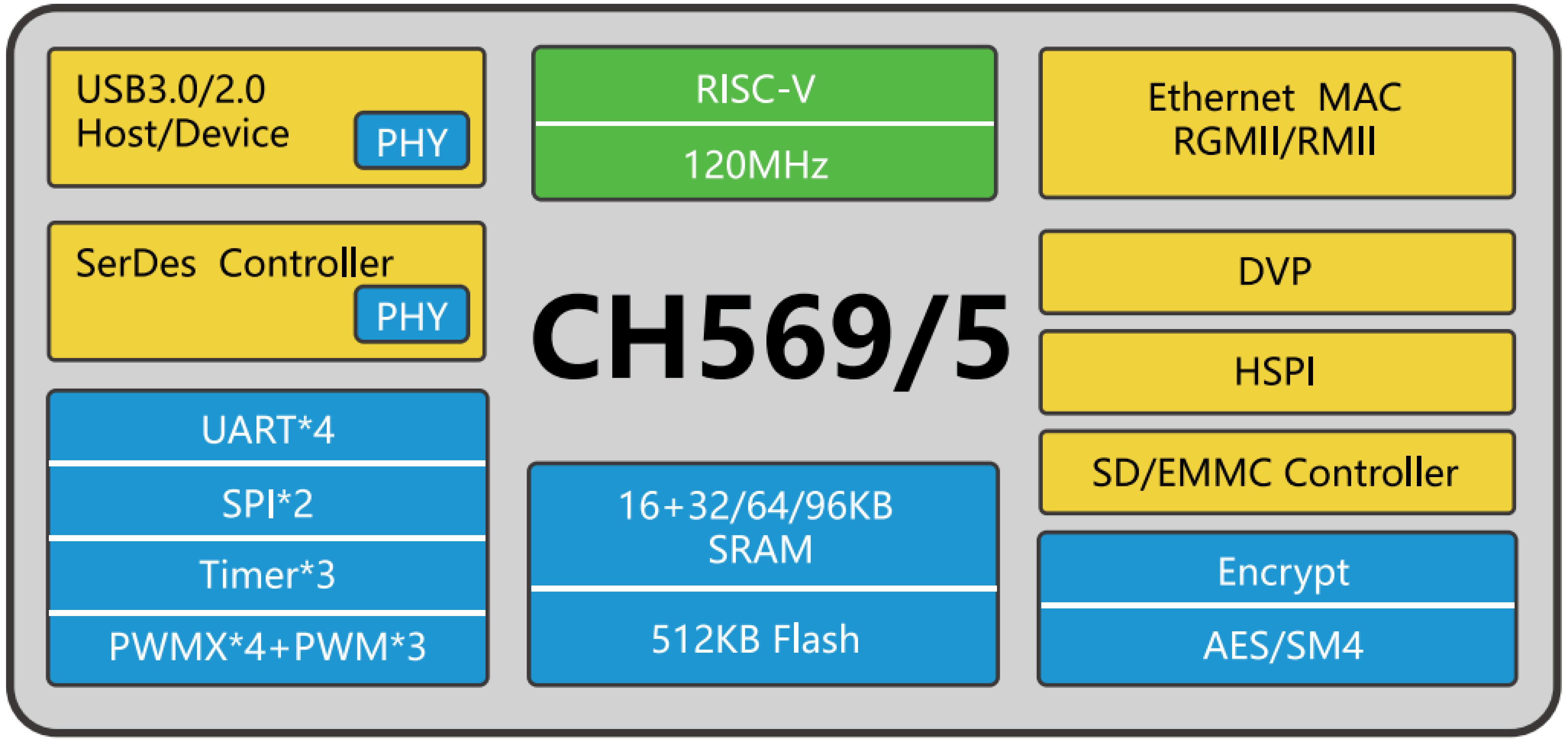
DEMO: mass storage proxy, high-speed

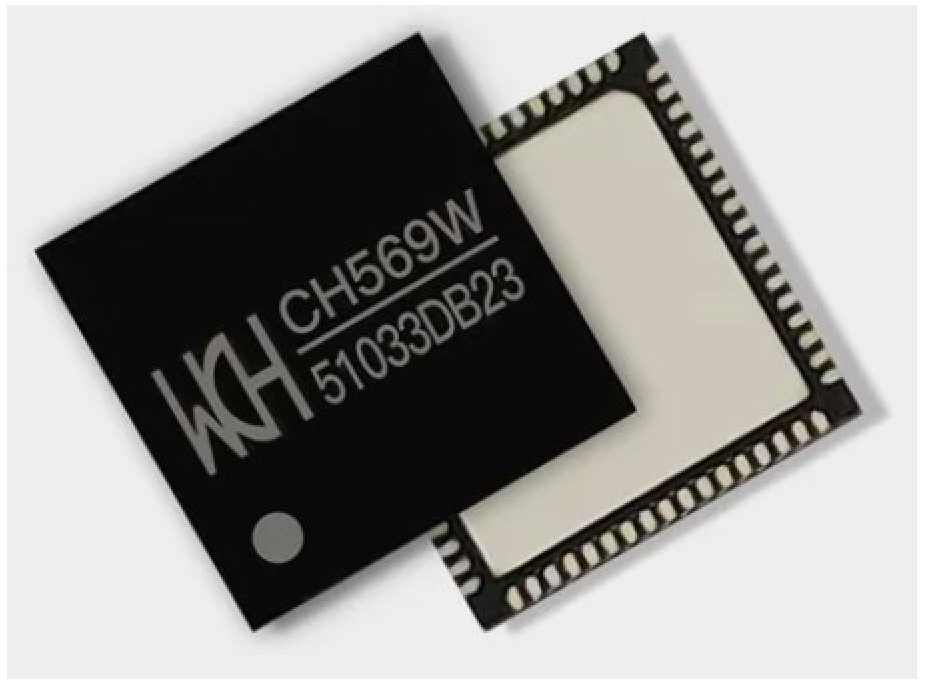


```
# python3 ./examples/usbproxy.py
```


/
Fichier Édition Affichage Aller à Marque-pages Aide
Poste de travail
Dossier personnel
Bureau
Documents
Musique
Images
Vidéos
Téléchargements
Système de fichiers
Corbeille
Réseau
Réseau
bin boot cdrom dev etc home lib lib32 lib64 libx32 media mnt opt
proc root run sbin snap srv sys tmp usr var
0:00 / 0:44

WCH-CH569W: why?

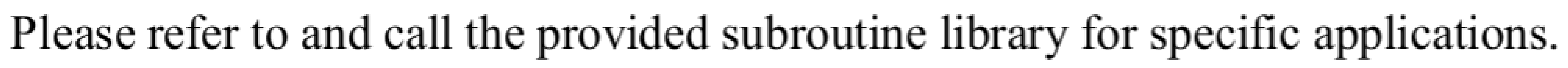




https://www.wch-ic.com

WCH-CH569W: why??

- No USB3 or SerDes documentation (examples, binary blobs)


Please refer to and call the provided subroutine library for specific applications.

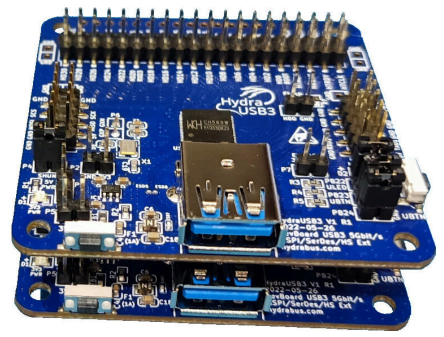
Extract from the USB3 section of the CH569 datasheet

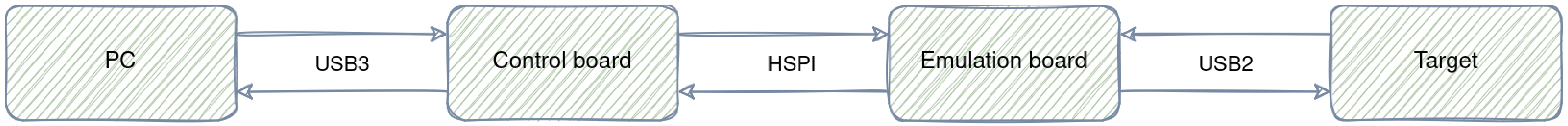
- Undocumented behavior of USB2 and HSPI hardware
- No international forums
- Incomplete examples
  - no USB2 FS/LS
  - no variable size packets in USB3
  - no examples of NAK

wch-ch56x-bsp and wch-ch56x-lib

- wch-ch56-bsp: "Reverse Engineering of advanced RISC-V MCU with USB3 & High Speed peripherals" Benjamin Vernoux, GreHack2022
  - Unified SDK with open-source USB3/SerDes Interrupt Handlers
  - Various examples and tests
- wch-ch56x-lib: Pushing the limits of the CH569 by experimentation and testing
  - Higher-level SDK: USB abstraction, extended USB3/USB2 functionnalities, interrupt processing queue, logging
  - Additionnal tests

First architecture: dual-HydraUSB3

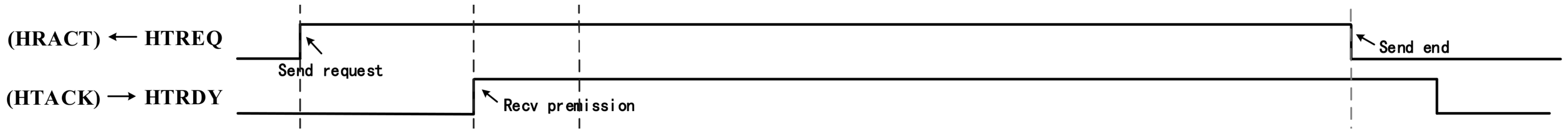




Dual-HydraUSB3 architecture

HSPI (High-Speed Parallel Interface) issues

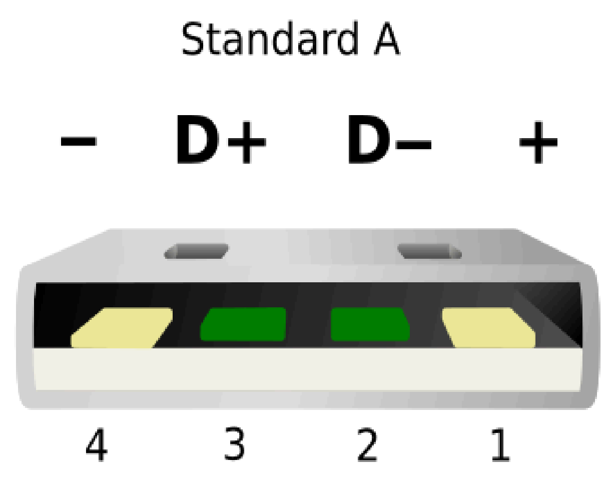
Hardware does not wait during interrupt, even if it's technically possible

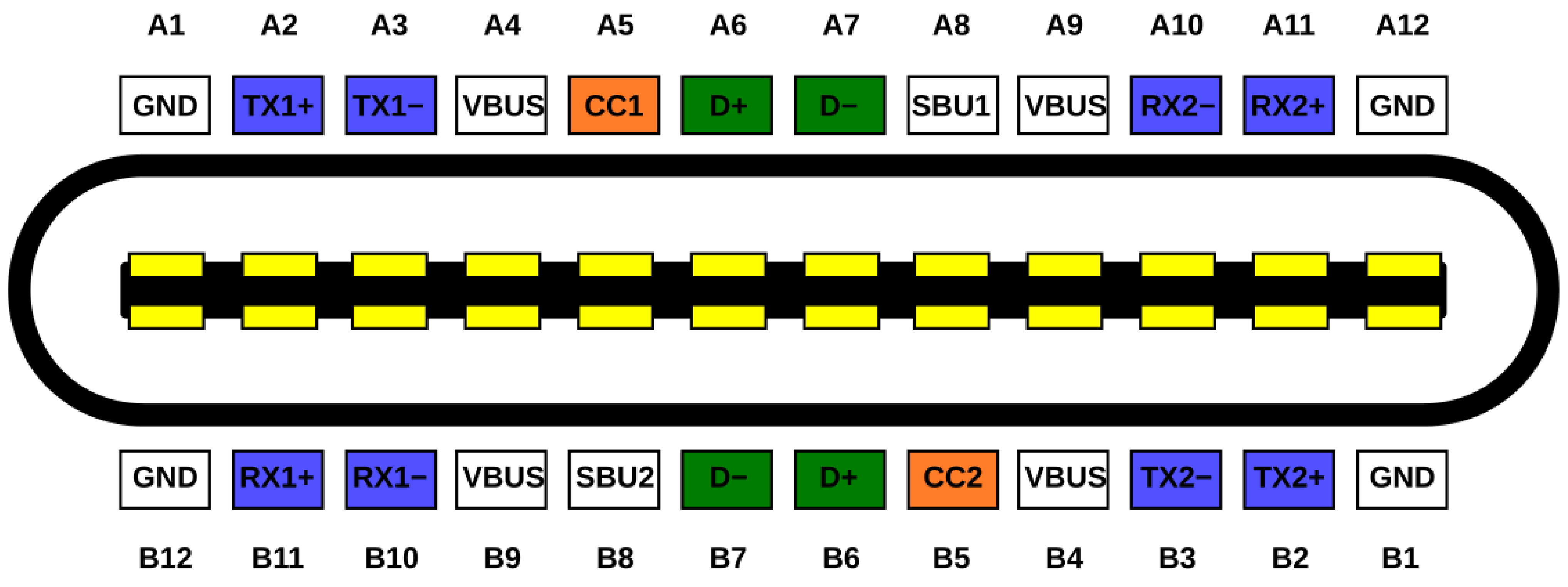


HSPI timing diagram, WCH CH569 Datasheet

Type-C

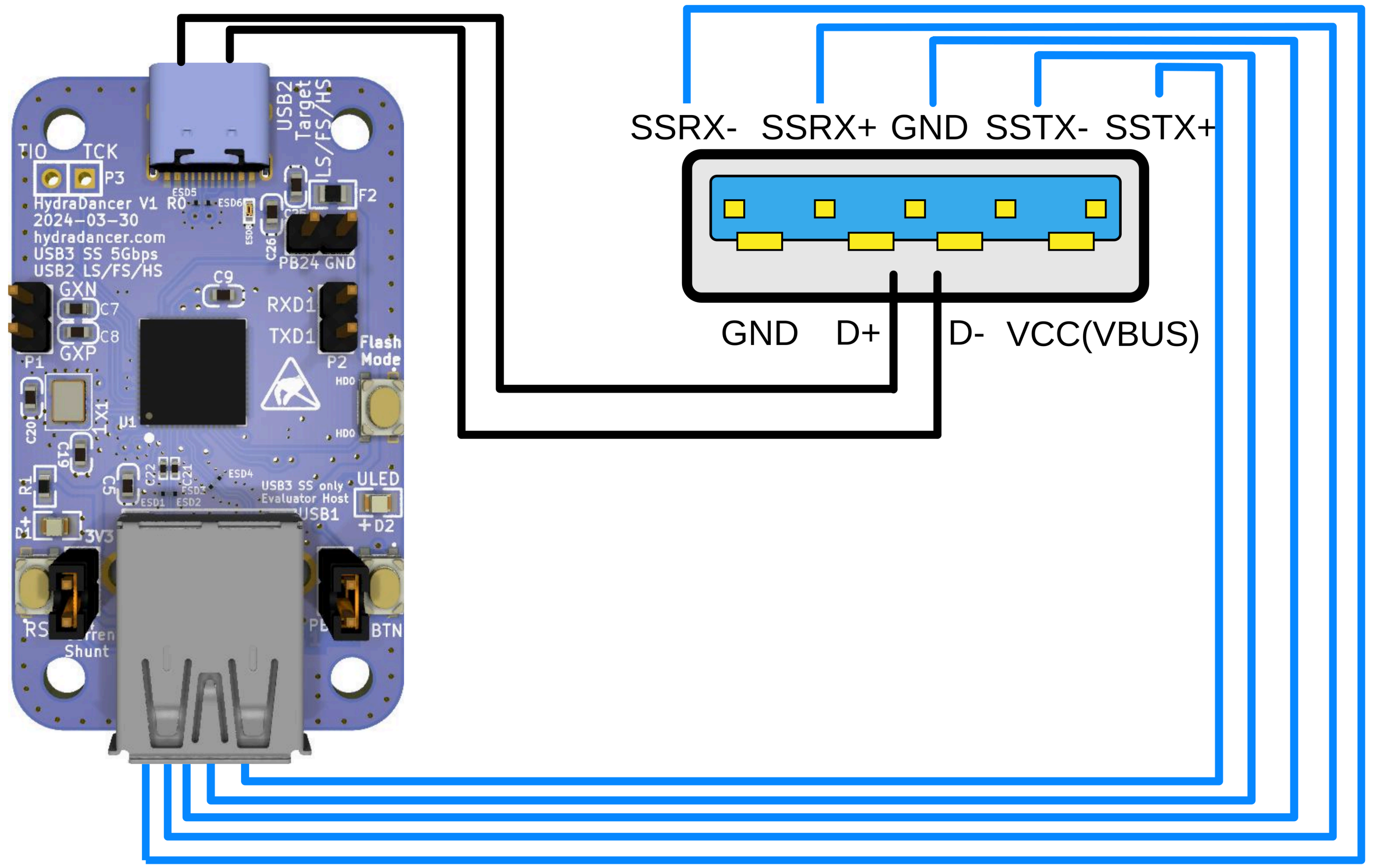
- 🟧 **USB-C Configuration, USB PD (Power Delivery)**
- 🟩 **USB 1.x/2.x**
- 🟦 **USB 3.x/4.x/Alternate modes (HDMI/DisplayPort/Thunderbolt/MHL)**





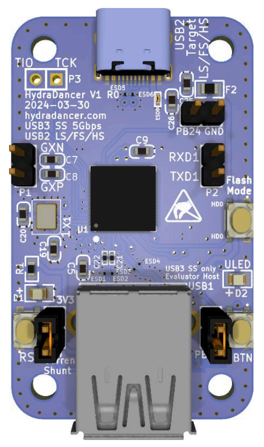
USB-C: CC BY-SA 4.0, Wikipedia, Chindi.ap ; USB-A: CC BY-SA 3.0, Wikipedia, Simon Eugster

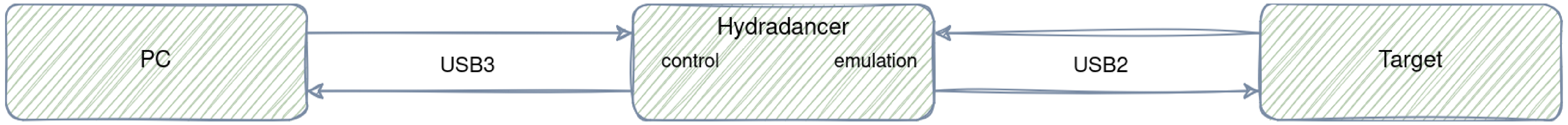
Splitting the USB3 connector in two



"Regular" USB3 to USB3-only and USB2 connectors

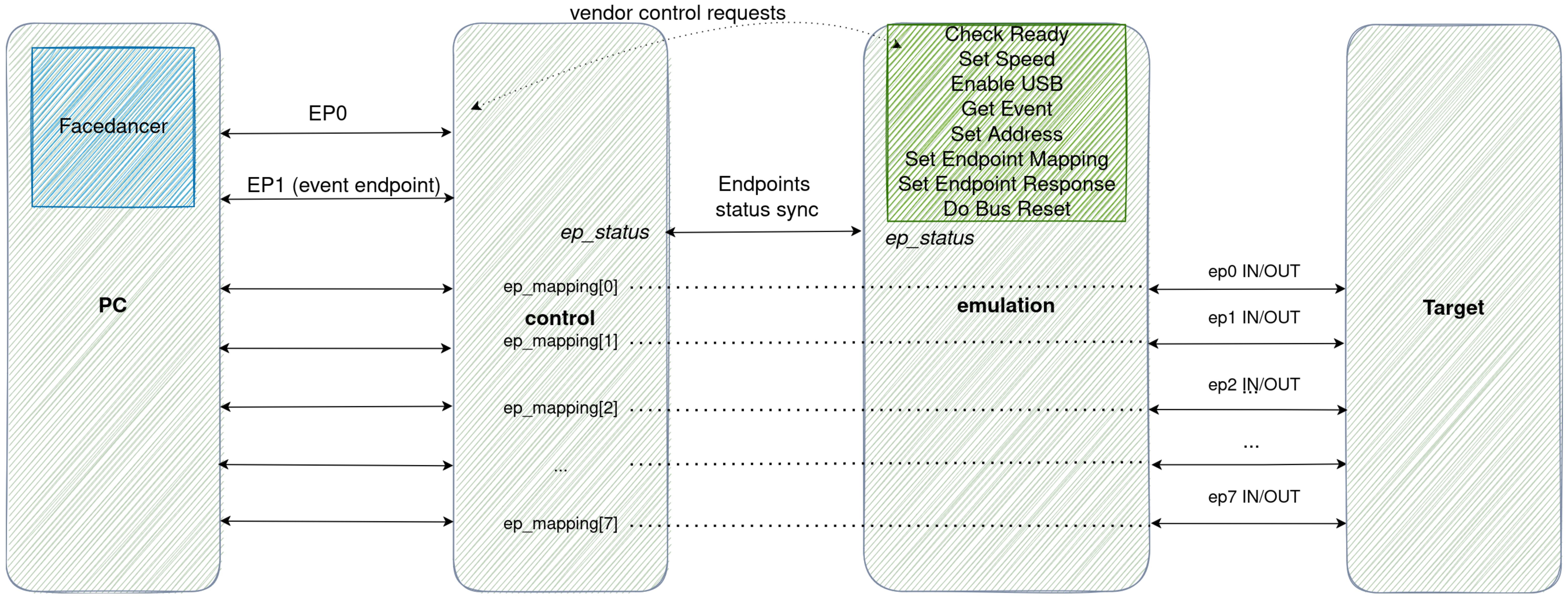
New architecture: Hydradancer dongle





Hydradancer dongle architecture

New architecture: more details

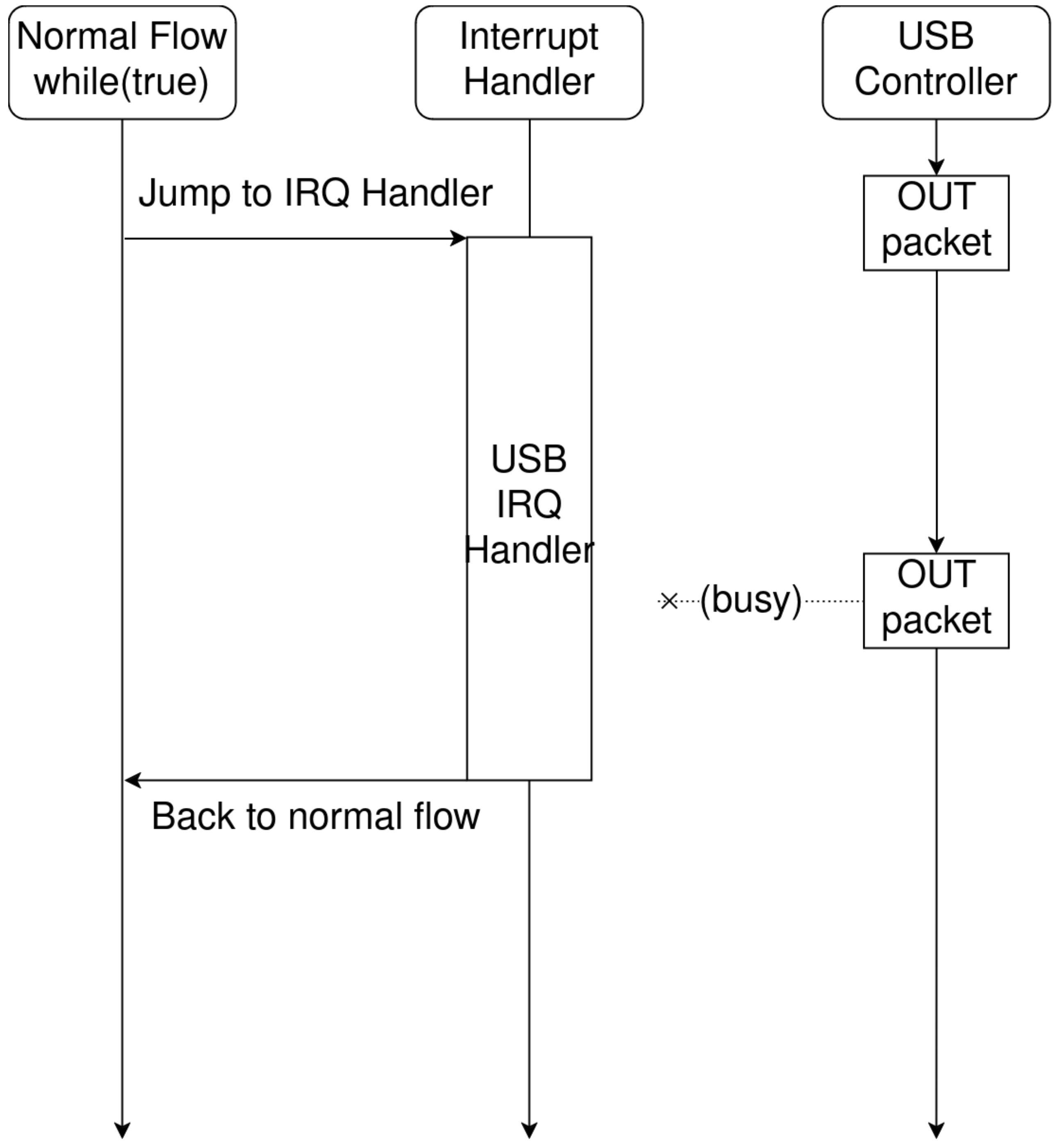


Facedancer backend

Interrupt hell

Saving time for interrupts to happen

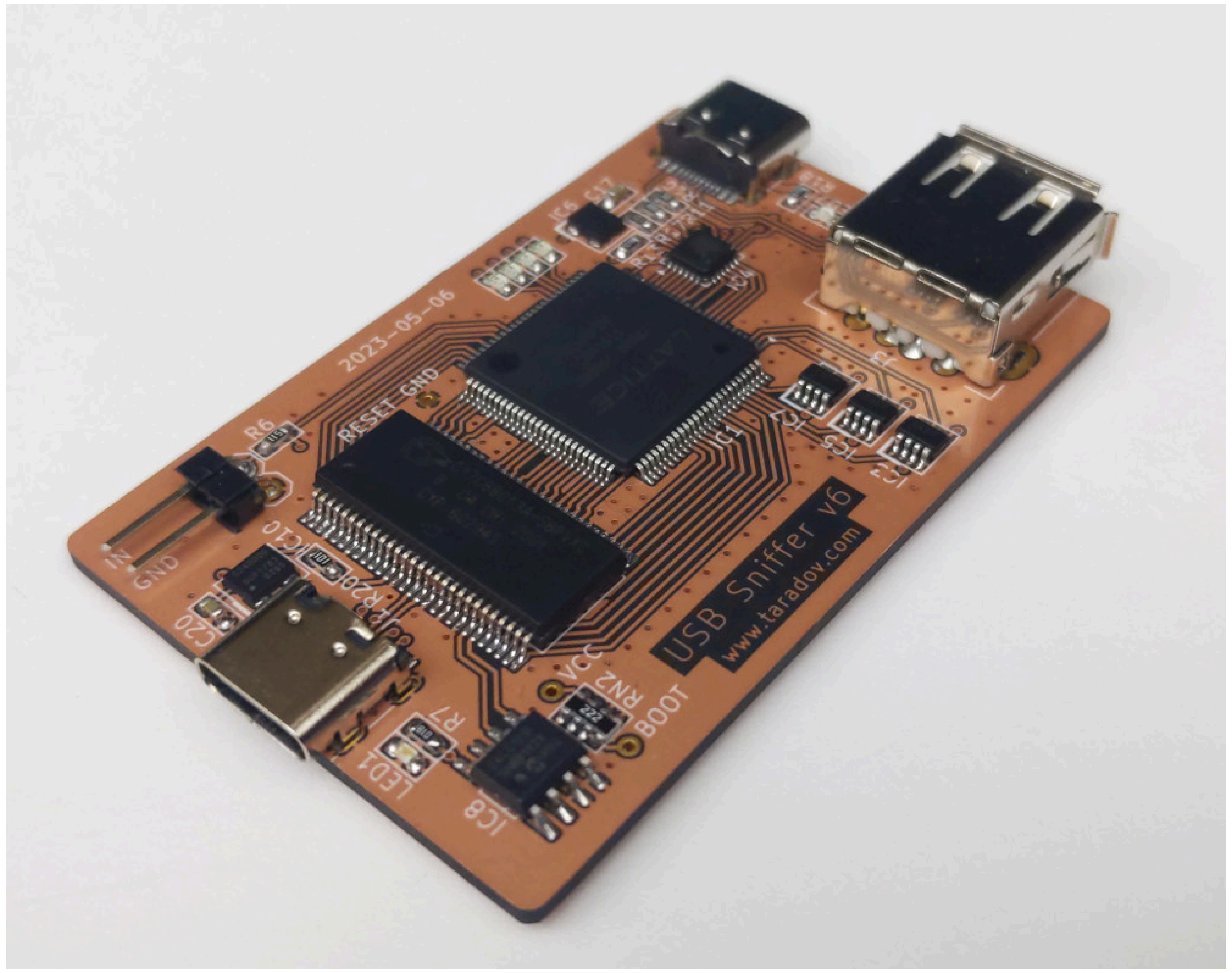
- Zero-copy (no memcpy)
- Store data, handle in normal flow
- Hardware busy while in interrupt



Dealing with interrupts

Debugging USB on the WCH CH569

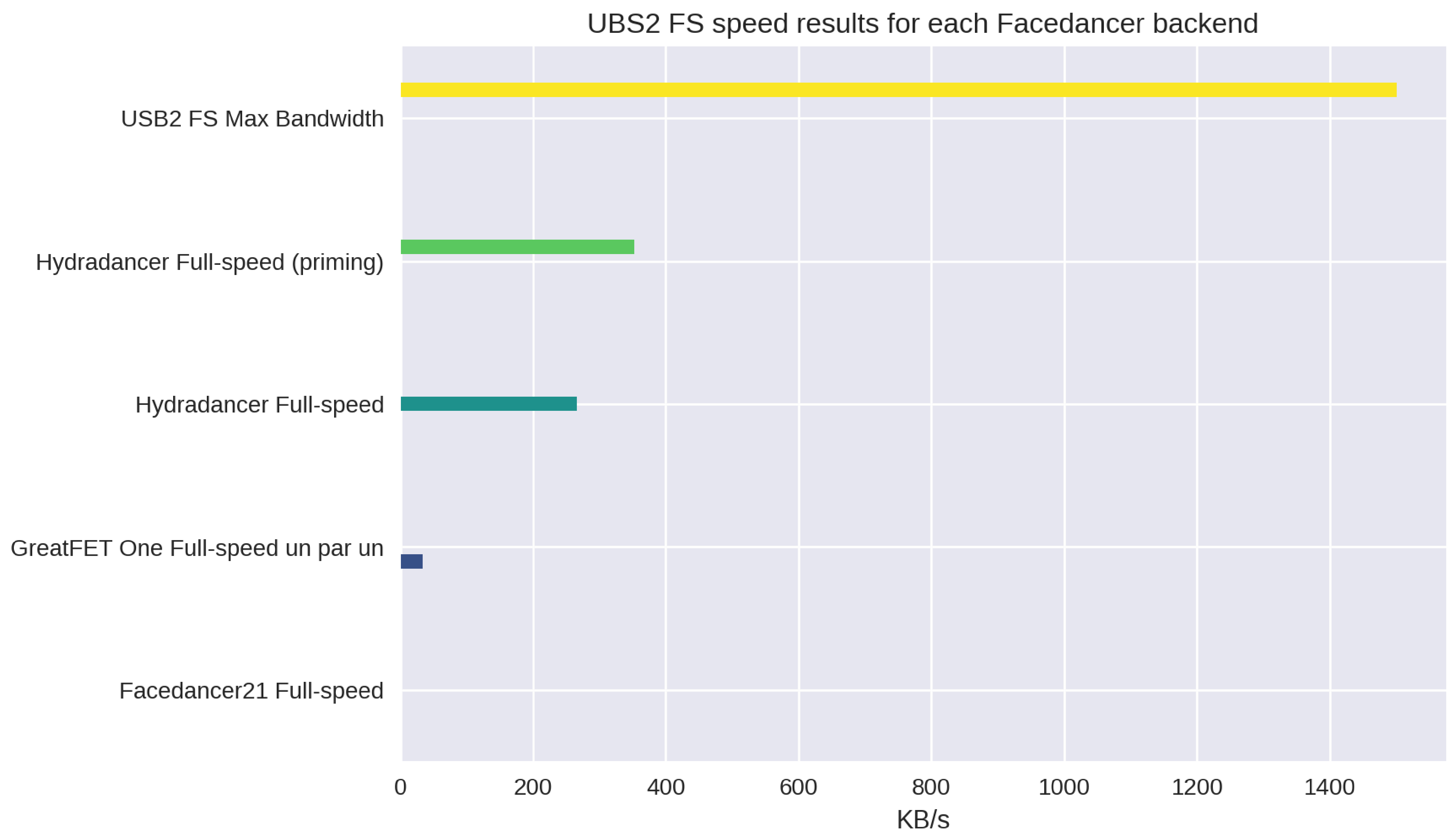
- USB protocol analyzer required
  - A.Tadarov USB Sniffer ($60), open-source, Wireshark plugin
  - Beagle USB 480 ($$$$1,295)
- Wireshark/usbmon: USB transfers (not packet level)
- `lsusb -v -d vid:pid`
- `dmesg`
- `udevadm monitor`
- UART logs, beware the interrupts



https://github.com/ataradov/usb-sniffer

```
#
```

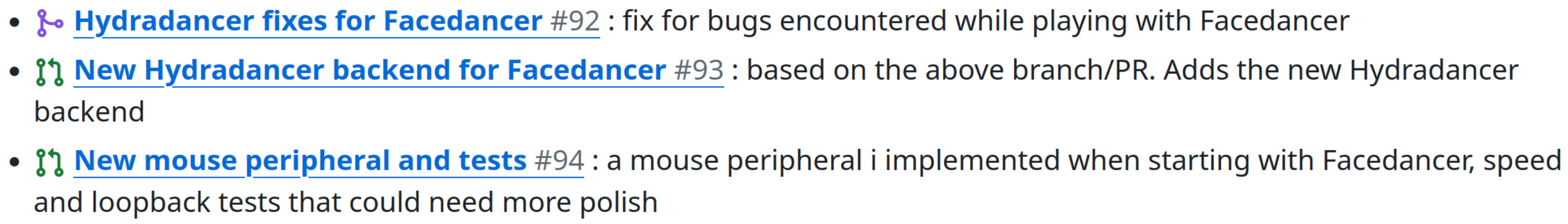
Results and comparison: speedtests



Results and comparison: speedtests

	Write average estimate	Read average estimate
Hydradancer High-speed	3911±151 KB/s	2653±96 KB/s
Hydradancer High-speed (priming)	3788±194 KB/s	2962±118 KB/s
Hydradancer Full-speed (priming)	369.80±2.46 KB/s	352.35±6.66 KB/s
Hydradancer Full-speed	369.66±4.98 KB/s	266.64±7.32 KB/s
GreatFET One Full-speed (one by one) (git-v2021.2.1-64-g2409575 firmware)	32.42±0.85 KB/s	33.07±1.10 KB/s
Facedancer21 Full-speed (2014-07-05 firmware)	0.697±0.000 KB/s	0.682±0.000 KB/s

Pull-request, fixing bugs



- **Hydradancer fixes for Facedancer** #92 : fix for bugs encountered while playing with Facedancer
- **New Hydradancer backend for Facedancer** #93 : based on the above branch/PR. Adds the new Hydradancer backend
- **New mouse peripheral and tests** #94 : a mouse peripheral i implemented when starting with Facedancer, speed and loopback tests that could need more polish

https://github.com/greatscottgadgets/facedancer/issues/95

USB as a pentester target: probing hosts for supported peripherals

Existing tools

- umap2: host fuzzing and scanning.
  Includes many peripherals BUT buggy, unmaintained for 3 years, same for the kitty fuzzing framework.
  Facedancer files included in project, not as Python module.
- nu-map: umap2 translation to modern Facedancer (Facedancer as Python module), "from friends of @greatscottgadgets".
  Fuzzing framework still unmaintained, mostly same bugs and incomplete.

There's a need for new fuzzing and scanning tools based on Facedancer!

USBScan: unreleased Python tool to scan USB hosts

- Fixed umap2 peripherals
- Fixed bugs in Facedancer (PR merged)
- Inject detection in the `USBDevice` object, "transparent"
- JSON-based
- USB classes/devices lists from linux-hardware.org/usb.org
- USB class/device/vendor scan

When is a USB peripheral trully handled by the system? Some hints

- Successful configuration is not enough: any USB peripheral can do it
- Detection based on Class/Vendor/Reserved requests
- Detection based on endpoint activity (excluding priming)

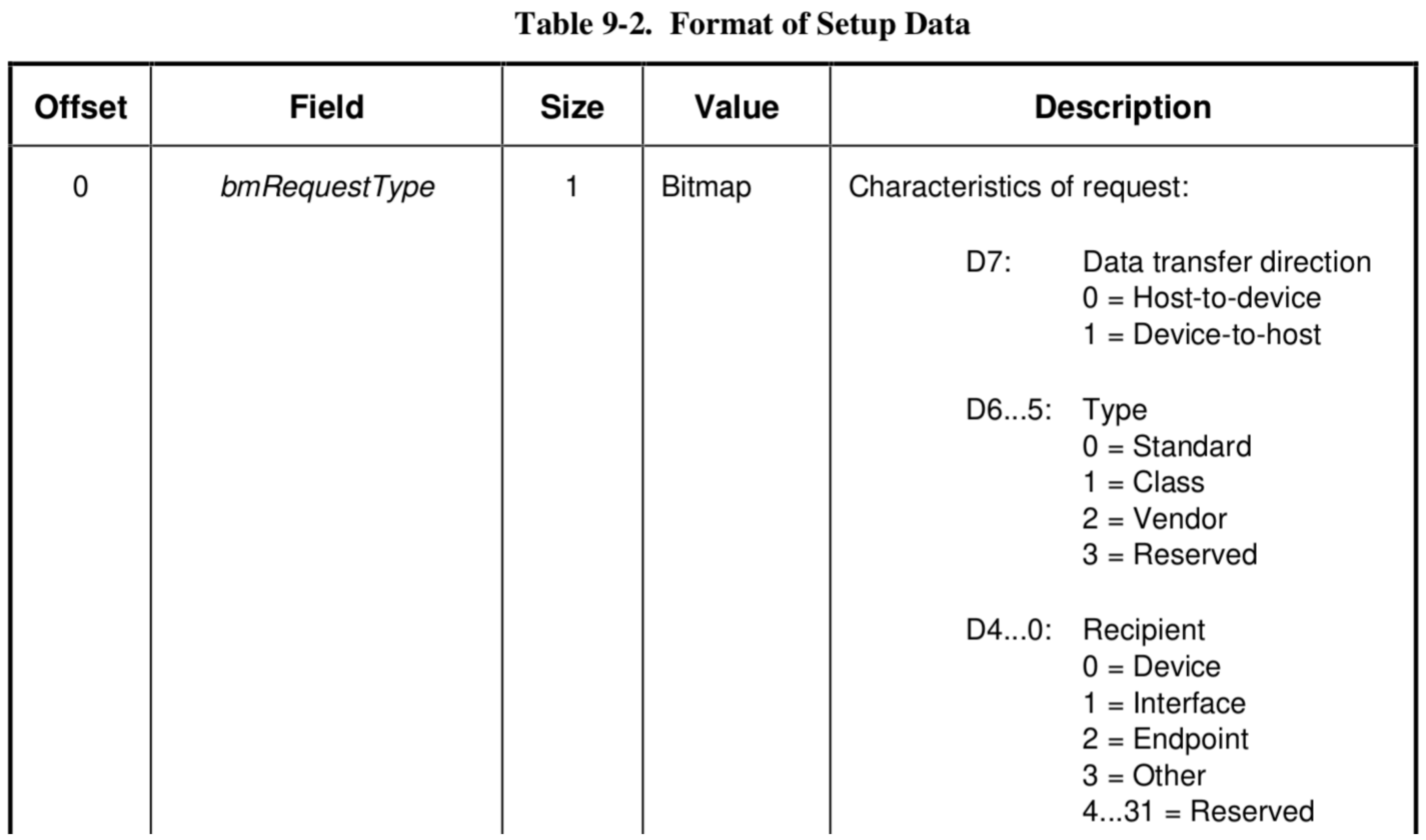
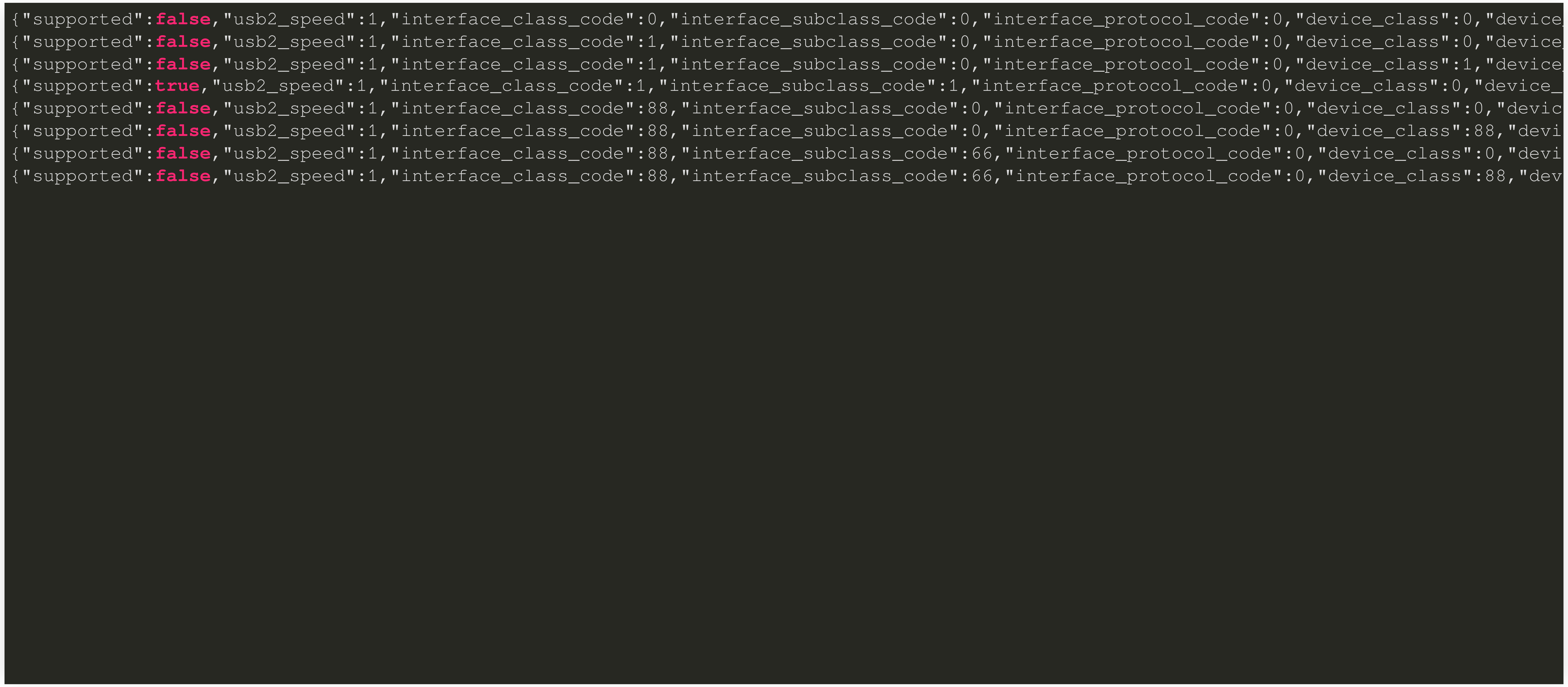


Table 9-2. Format of Setup Data

Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved

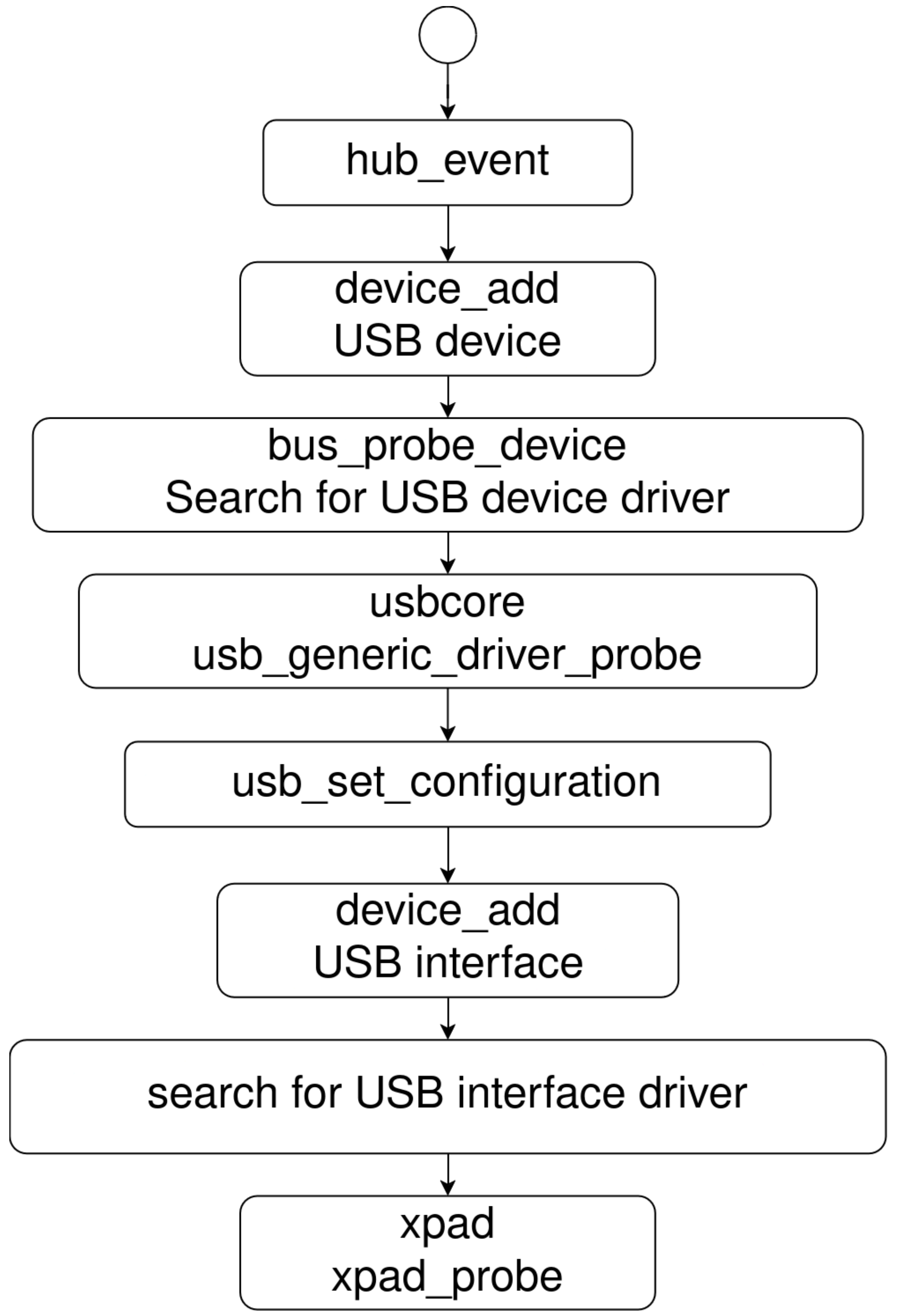
USB 2.0 specification, 9.3

DEMO: scan of a computer

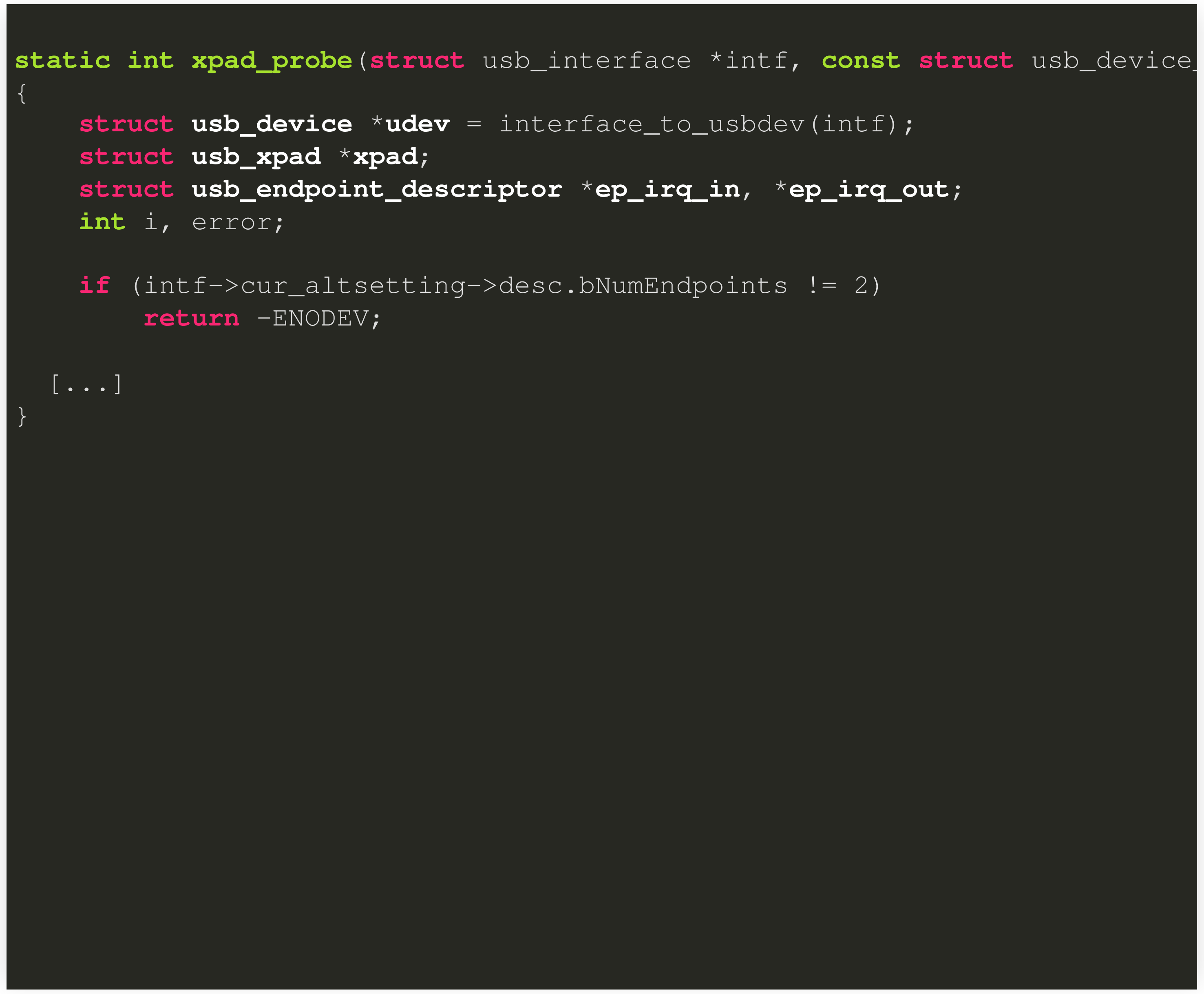


```
{"supported":false,"usb2_speed":1,"interface_class_code":0,"interface_subclass_code":0,"interface_protocol_code":0,"device_class":0,"device
{"supported":false,"usb2_speed":1,"interface_class_code":1,"interface_subclass_code":0,"interface_protocol_code":0,"device_class":0,"device
{"supported":false,"usb2_speed":1,"interface_class_code":1,"interface_subclass_code":0,"interface_protocol_code":0,"device_class":1,"device
{"supported":true,"usb2_speed":1,"interface_class_code":1,"interface_subclass_code":1,"interface_protocol_code":0,"device_class":0,"device_
{"supported":false,"usb2_speed":1,"interface_class_code":88,"interface_subclass_code":0,"interface_protocol_code":0,"device_class":0,"devic
{"supported":false,"usb2_speed":1,"interface_class_code":88,"interface_subclass_code":0,"interface_protocol_code":0,"device_class":88,"devi
{"supported":false,"usb2_speed":1,"interface_class_code":88,"interface_subclass_code":66,"interface_protocol_code":0,"device_class":0,"devi
{"supported":false,"usb2_speed":1,"interface_class_code":88,"interface_subclass_code":66,"interface_protocol_code":0,"device_class":88,"dev
```

Not a silver bullet: xpad, the Linux driver for XBox controllers



Simplified Linux USB driver stack



```
static int xpad_probe(struct usb_interface *intf, const struct usb_device_
{
    struct usb_device *udev = interface_to_usbdev(intf);
    struct usb_xpad *xpad;
    struct usb_endpoint_descriptor *ep_irq_in, *ep_irq_out;
    int i, error;

    if (intf->cur_altsetting->desc.bNumEndpoints != 2)
        return -ENODEV;

  [...]
}
```

Conclusion

- Renewed interest in Facedancer: v3.0, USB2 High-Speed with Cynthion and Hydradancer
- https://github.com/HydraDancer/hydradancer_fw: open to contributions and issues
- https://twitter.com/hydrabus: Hydradancer dongle will be announced there
- USBScan: might be open-sourced
- We need new USB fuzzing tools based on Facedancer!

Thanks! Questions?

quarkslab

tfuchs@quarkslab.com

raw-gadget: USB3 in Facedancer one day?

What's raw-gadget?

- raw-gadget: used in Google's syzkaller to fuzz the Linux USB drivers
- Similar to `usbfs` driver/libusb but for USB devices
- Not yet USB3, but not limited by technology
- Need a UDC (USB Device Controller) in your system

raw-gadget and Facedancer

There's a prototype Facedancer backend!