https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

USB Debug Capability (DbC) Support on FreeBSD

Hiroki Sato <hrs@FreeBSD.org>

EuroBSDcon2023 / 2023.9.16

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Outline

- **Who am I?**
    - A Japanese FreeBSD committer since 2000, working in various areas
- **Outline of This Talk**
    - Background
    - USB Debug Capability
        - High-level Overview
        - USB Host/Device Controller Basics
            - Pipes and Endpoints
            - TRBs
        - Implementation Details
    - Demo and Future Work

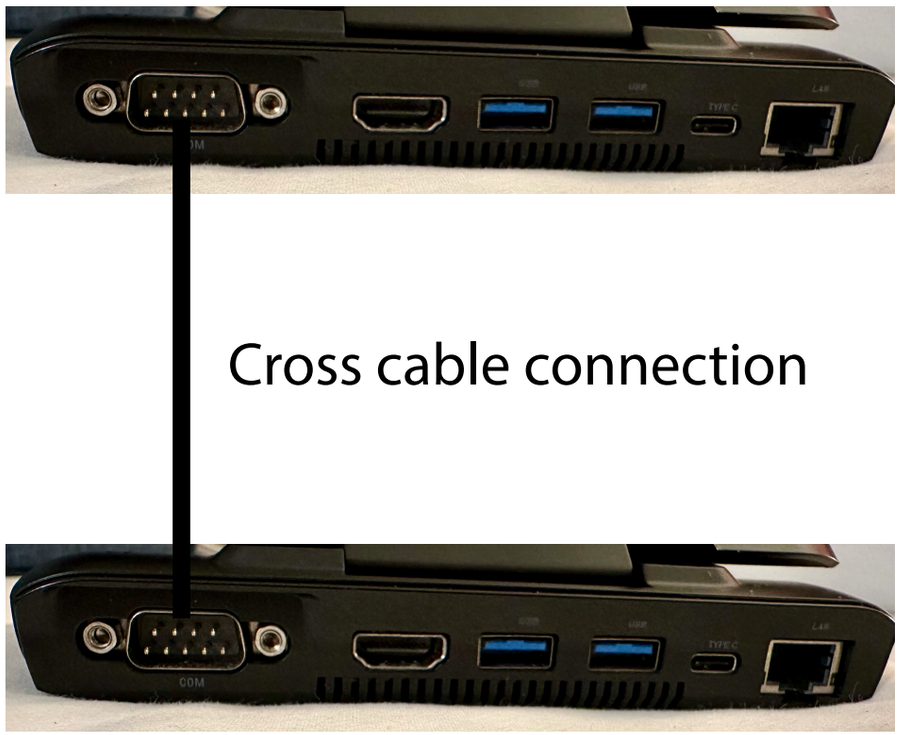
https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Background

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Background

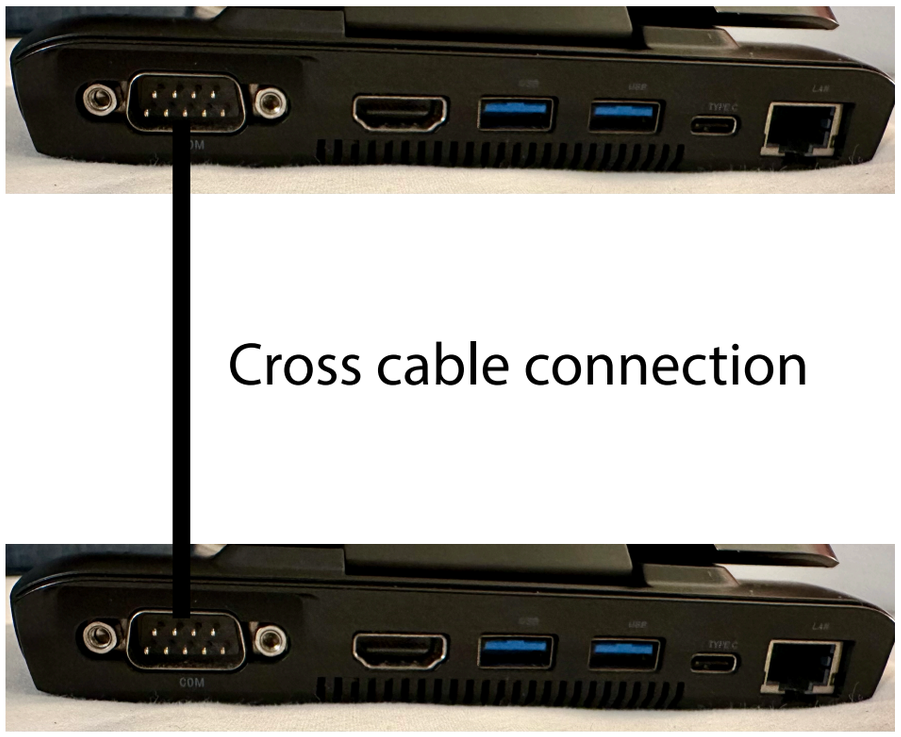
- **Debugging work using serial console:**
  - Remote access to a headless machine, including firmware (BIOS/UEFI) configuration
  - Device driver hacking
    - Remote GDB session



https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Background

- **Debugging work using serial console:**
  - Remote access to a headless machine, including firmware (BIOS/UEFI) configuration
  - Device driver hacking
    - Remote GDB session

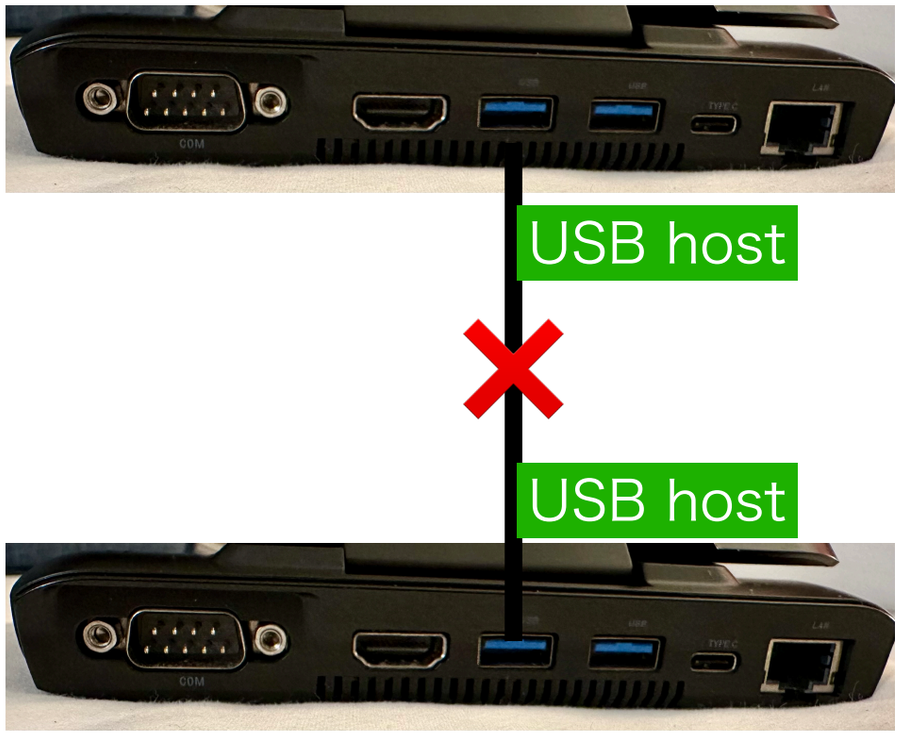


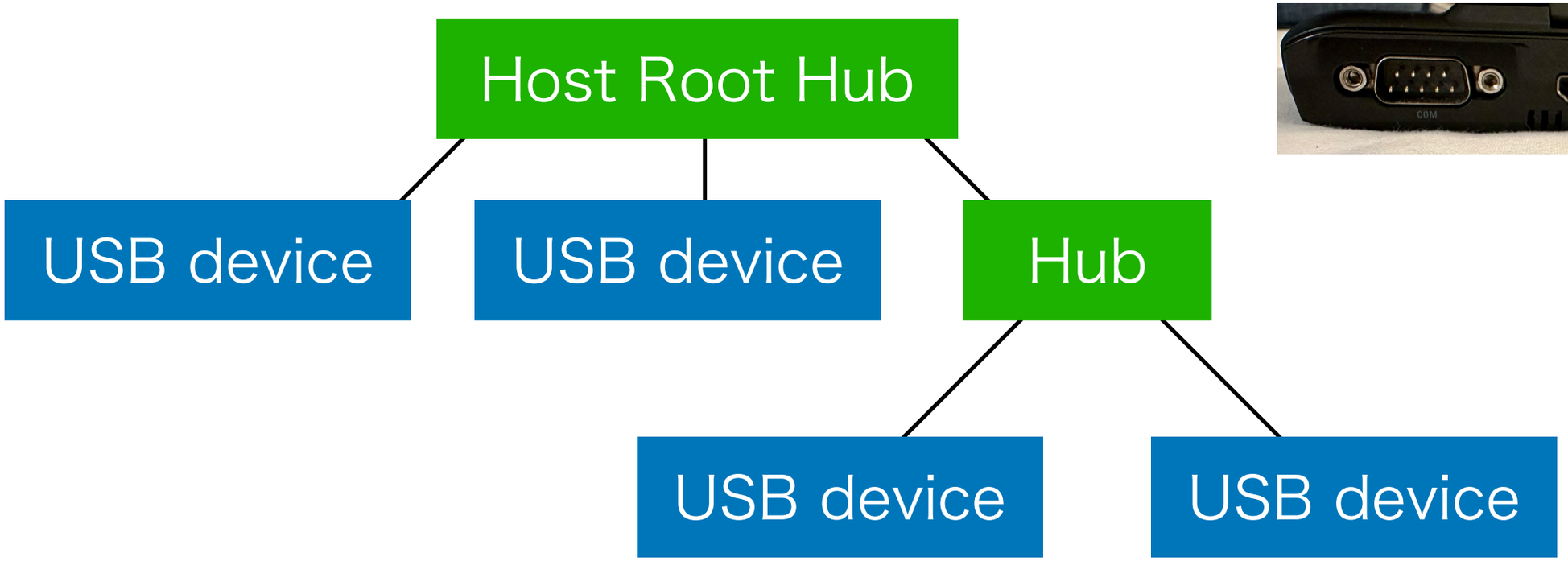
- **No serial port on modern hardware, however...**
  - A legacy interface
  - Server-grade machines have BMC with "console redirection"
    - BMC: baseboard management controller
      - An embedded processor that runs independently
      - Provides virtual serial ports over IPMI SoL (Serial-over-LAN, 623/udp)

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Background

- **USB is the replacement of legacy interfaces including the serial ports**
  - USB basically requires tiered star topology
  - No direct connection of USB hosts is allowed





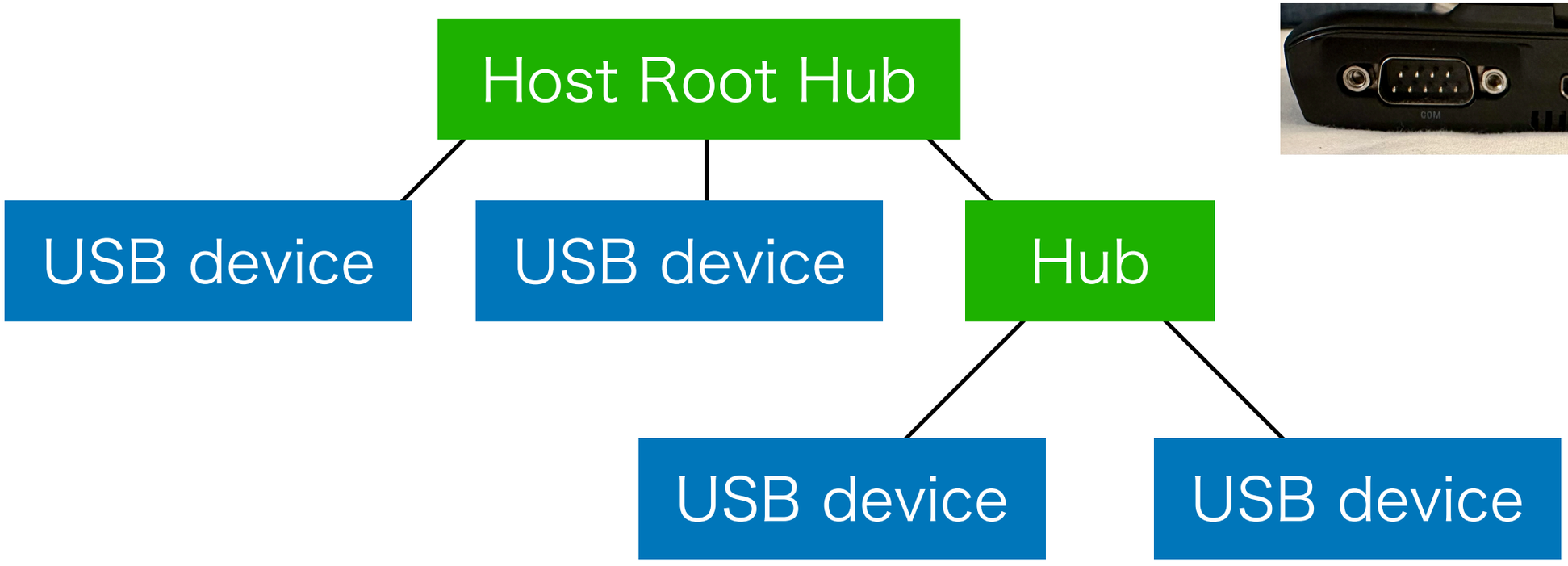
https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

USB Debug Capability

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

USB Debug Capability

- **1-sentence summary: USB DbC changes one of the USB ports on a USB host for a USB device**
  - Not a point-to-point connection
  - An optional feature in USB 3.0 Specification
  - Most of Intel xHCI controllers support it



https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

A-to-A Cable?

- **A-to-A USB3 Cross Cable is required**
  - No A-A for USB 2.0. It is not allowed.
  - USB3 spec has 5 cables including A-A. **It is always a cross cable.**
  - Note that non-standard A-A cables can be found in the market.

Reference: USB 3.1 Legacy Connector and Cable Specification

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Similar Technologies

- **IEEE 1394 (FireWire) supports point-to-point connection and physical memory access**
  - OHCI specification
  - You can read/write memory
  - dcons(4) is a serial communication driver using this
  - Firewire is considered a legacy interface

- **USB2.0 also supports debug capability**
  - EHCI specification
  - Requires a special repeater hardware

Signal Name	Wire Number
VBUS	1
D−	2
D+	3
GND	4

By Nicola02nb - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=117668227

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

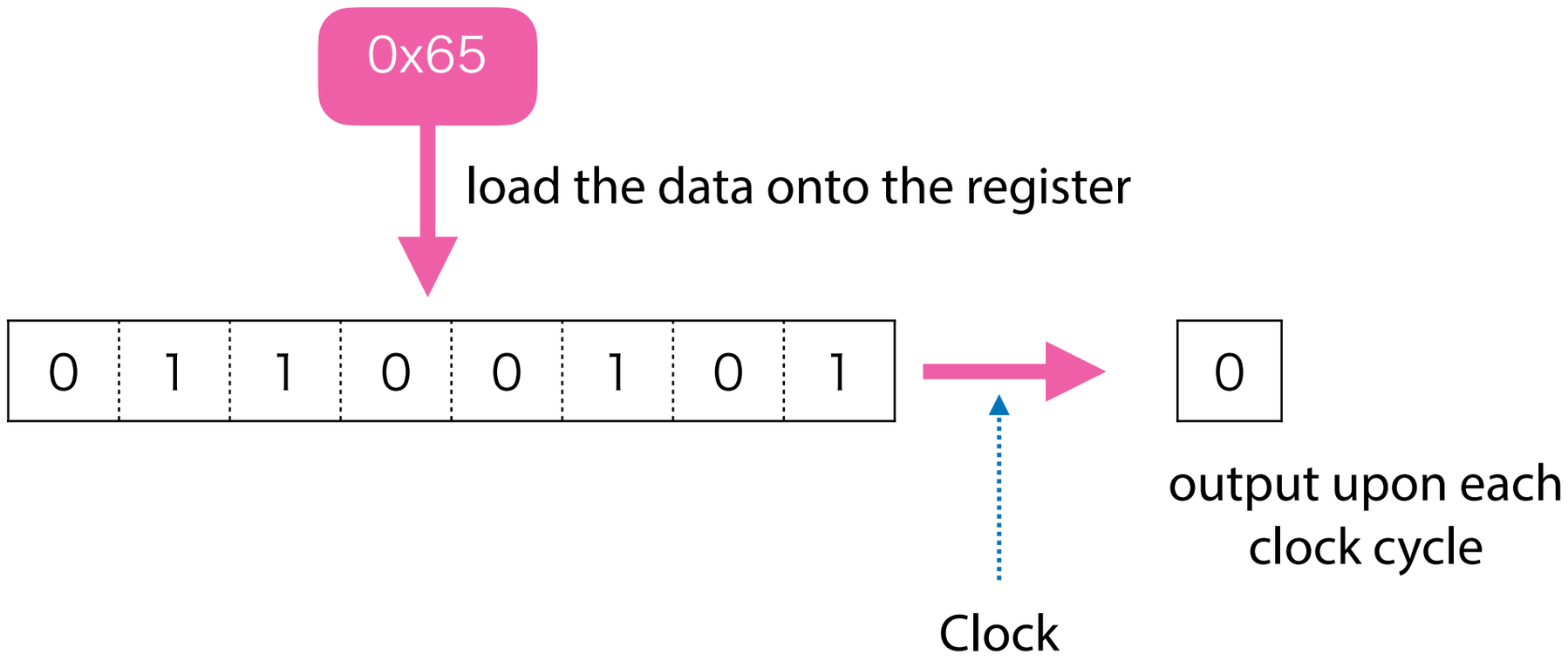
Implementation Details

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

USB Host/Device Controller Basics

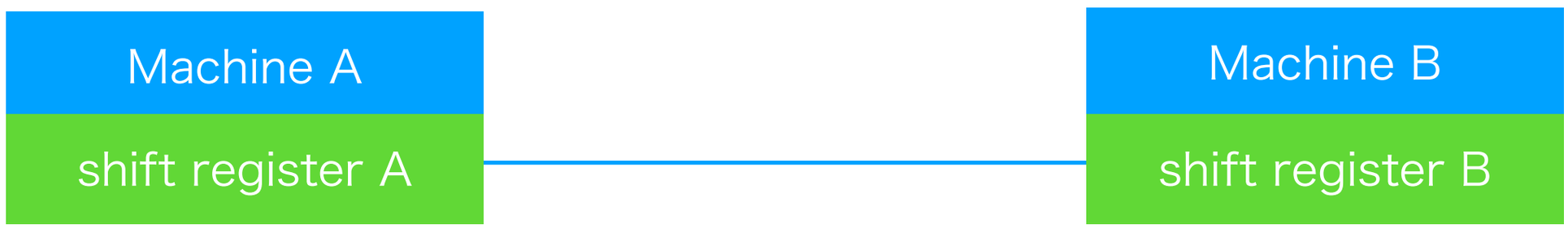
- **Serial communication over the legacy serial ports**

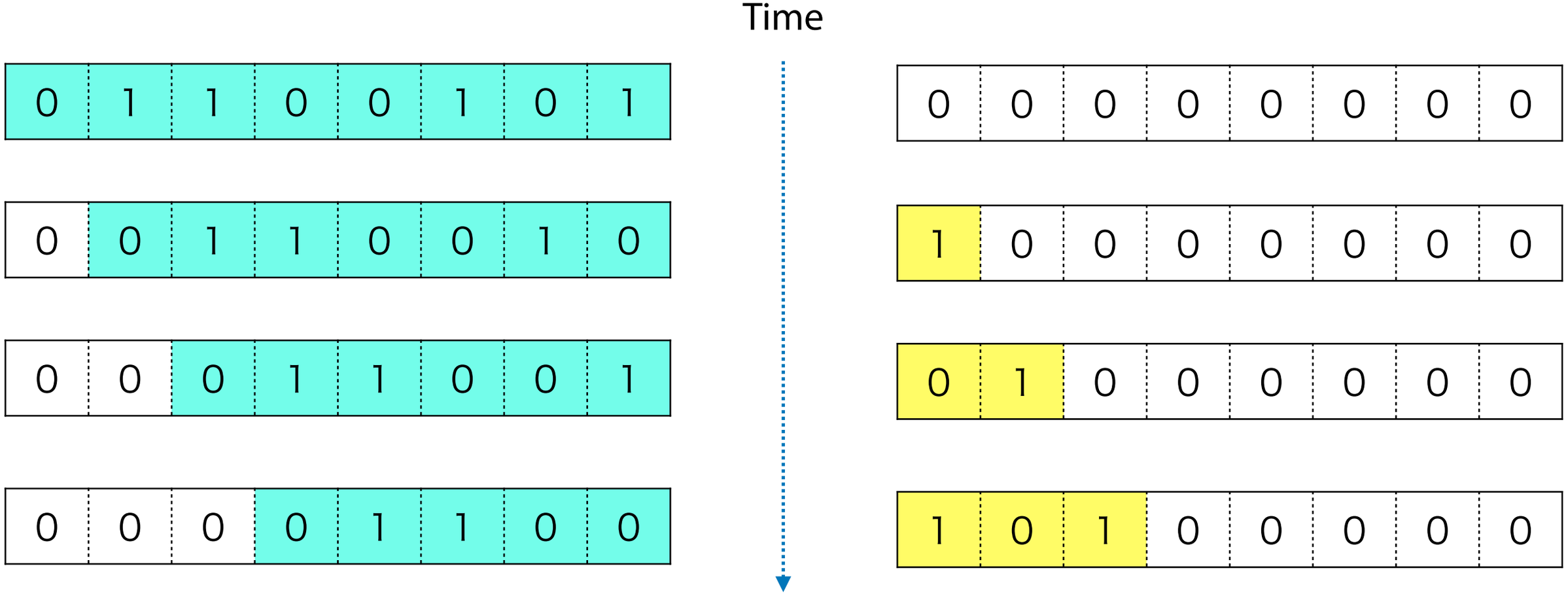
Shift register to convert data into a pulse sequence



USB Host/Device Controller Basics

- **Serial communication over the legacy serial ports**

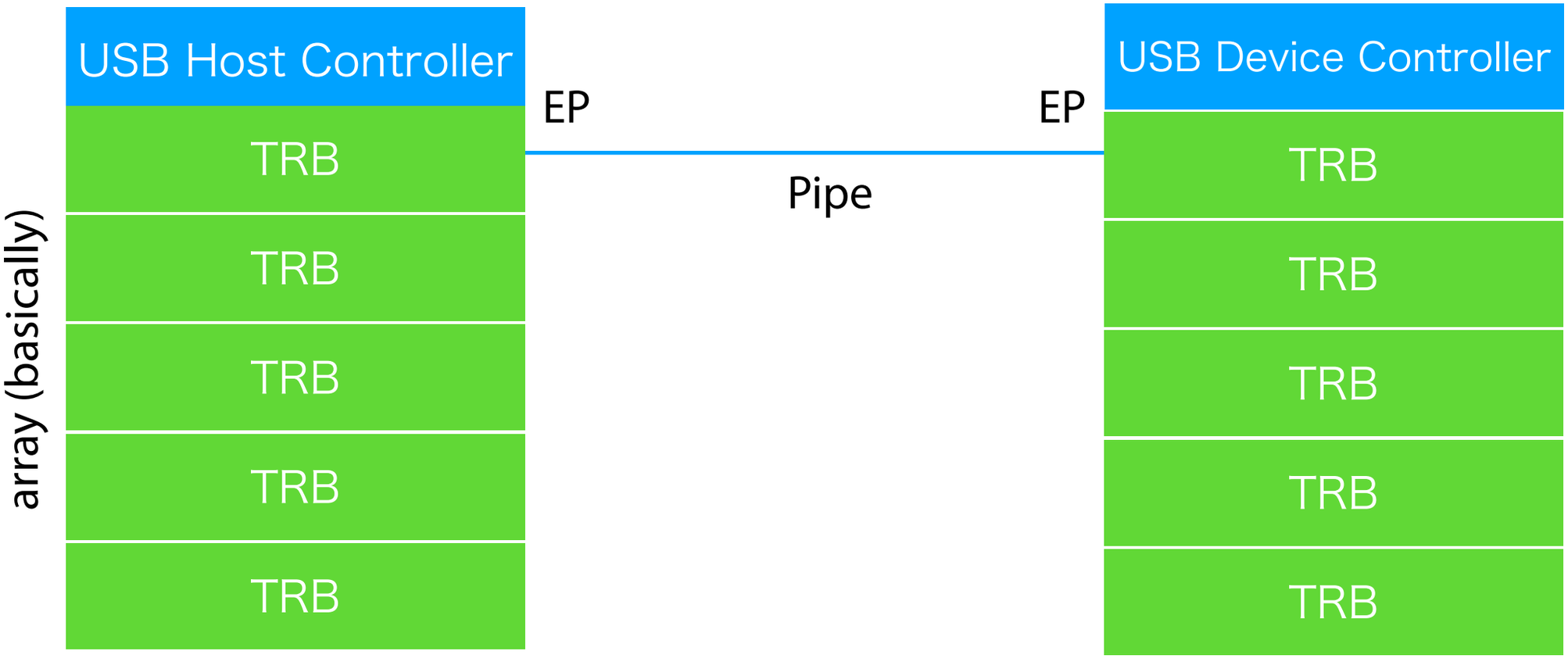




https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

USB Host/Device Controller Basics

- **Serial communication over USB using xHCI**

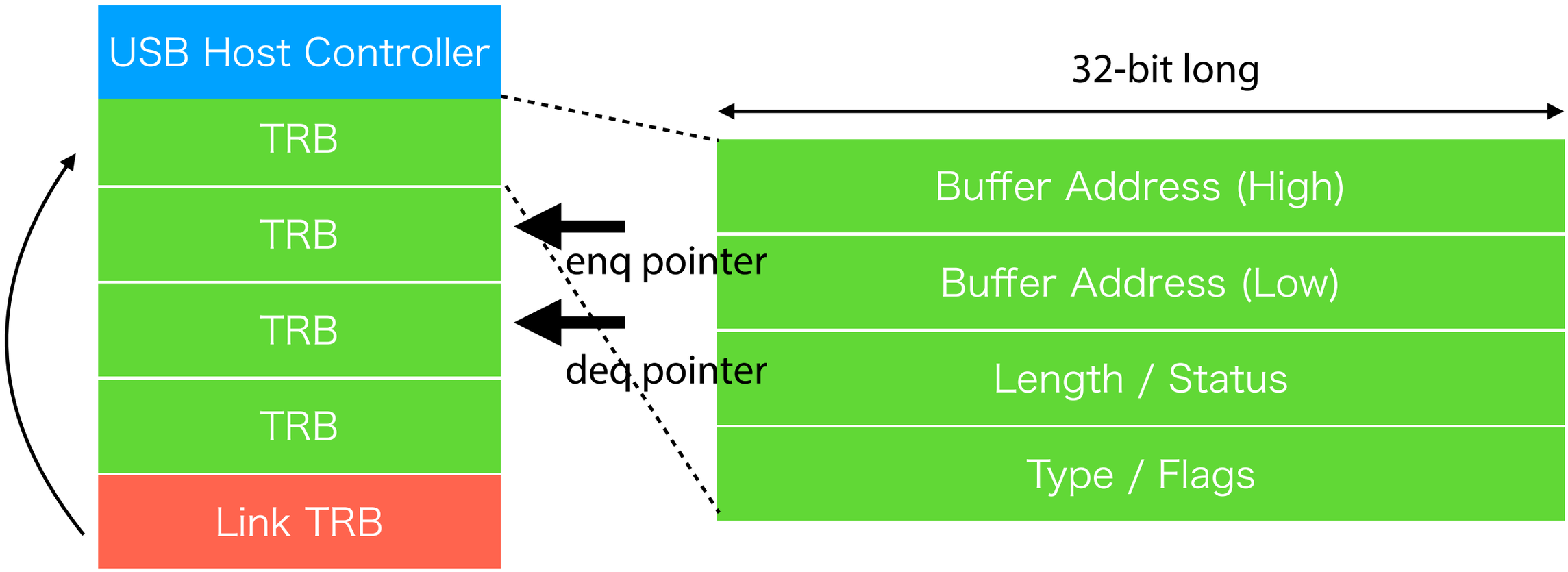


- xHCI uses ring buffers of TRBs (Transfer Request Block)
- Data on a TRB will be transferred to another end by the controllers.
- Multiple virtual serial communications are managed by EPs (End Point)

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

USB Host/Device Controller Basics

- **TRB and ring structure**



- A 16-byte TRB for transfer holds a pointer
  - Normal TRB type is used to specify data transfe
  - Link TRB type can point another TRB as the next one
    - A segmented TRB buffer helps when memory is non-contiguous

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Functions of USB DbC

- A virtual "device-side" controller with the minimal functionality on one of the ports on "host-side" controller:
    - Two pipes: IN and OUT
    - SuperSpeed (5Gbps) at least.
    - The max size of USB packet is 1024 bytes
    - **The host controller does not see the port after initialization**

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Functions of USB DbC

- A virtual "device-side" controller with the minimal functionality on one of the ports on "host-side" controller:
    - Two pipes: IN and OUT
    - SuperSpeed (5Gbps) at least.
    - The max size of USB packet is 1024 bytes
    - **The host controller does not see the port after initialization**

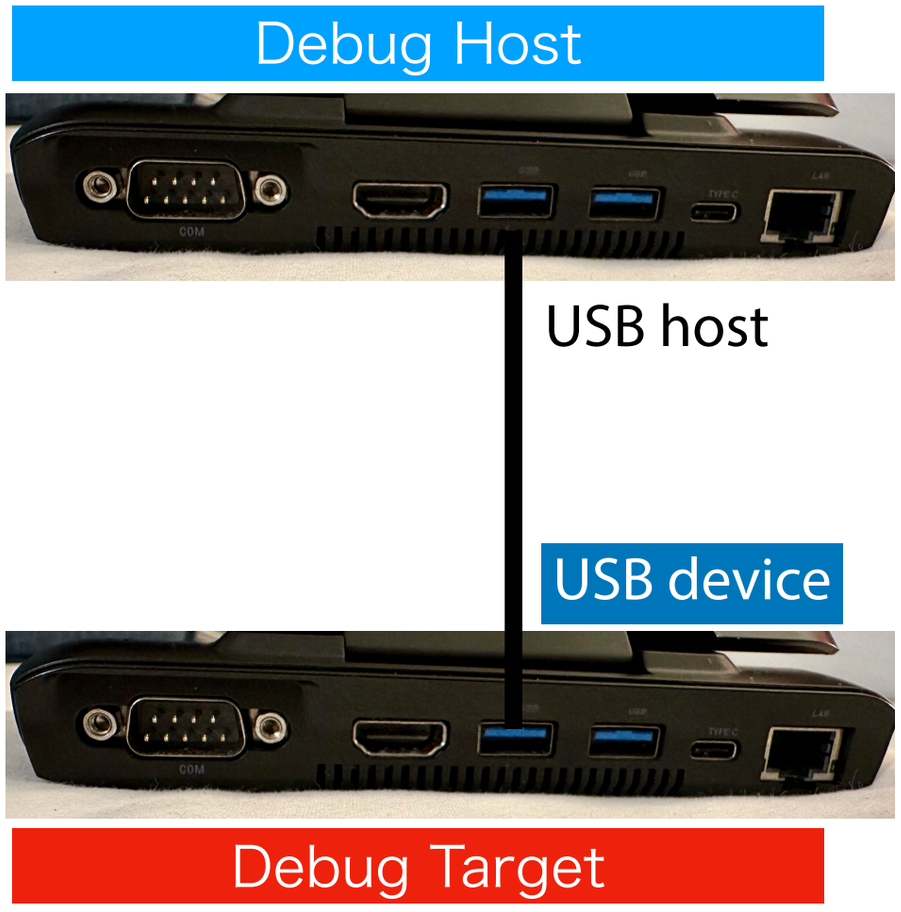
- **No full USB stack is required**
    - After specifying addresses for TRB ring buffers of the two pipes, what you have to do is to place your data into the ring buffer (or read it).

- getchar()/putchar() will be more than "`inb 0x3f8` + offset", but writing/reading the TRB ring is still simple

- DbC is designed as a transport for more sophisticated debug feature, such as JTAG and Intel DCI (exposing processor internal states and memory region)

Software Components for DbC

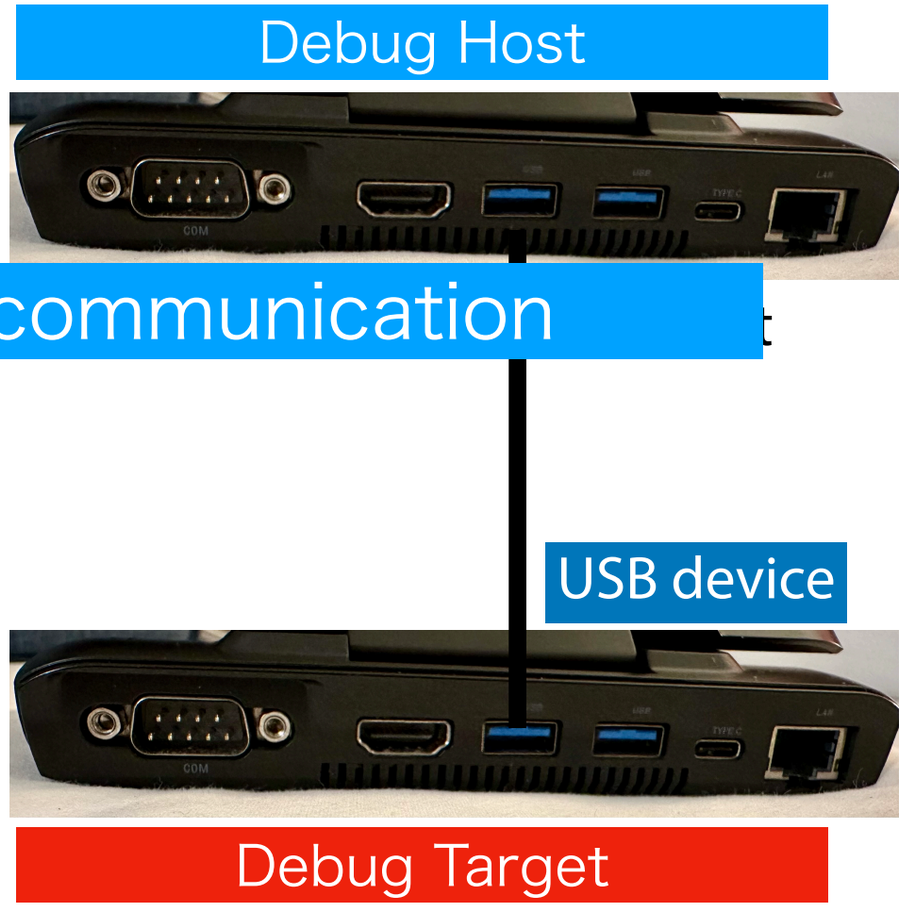
- **On the Debug Host**
  - A normal USB3 stack is sufficient. No DbC required.
  - A client driver is required. This is because the USB device has USB Debug Class (0xdc in the bInterfaceClass field)



https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Software Components for DbC

- **On the Debug Host**
  - A normal USB3 stack is sufficient. No DbC required.
  - udbc(4) driver for simple serial communication because the USB device has USB Debug Class (0xdc in the bInterfaceClass field)

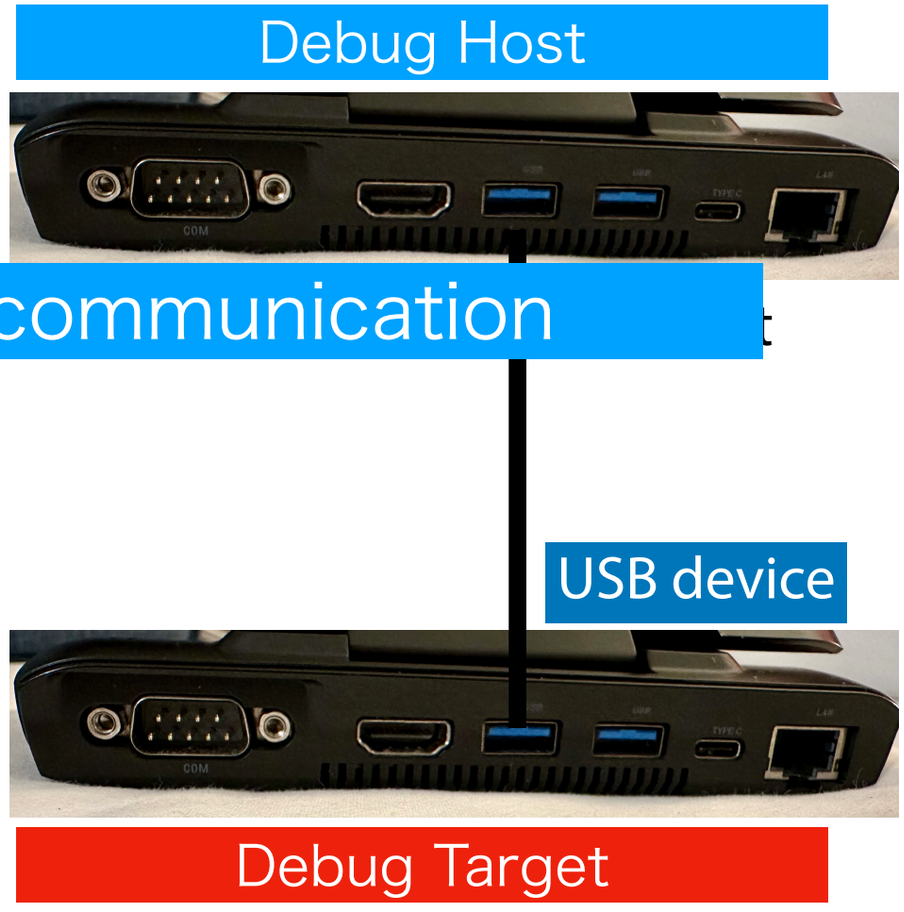


https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Software Components for DbC

- **On the Debug Host**
  - A normal USB3 stack is sufficient. No DbC required.
  - udbc(4) driver for simple serial communication because the USB device has USB Debug Class (0xdc in the bInterfaceClass field)

- **On the Debug Target**
  - Activation of DbC is required.
  - DbC has two endpoints (IN and OUT) for bulk transfer
  - TRB ring buffers for IN and OUT must be allocated in memory (DMA will handle them)

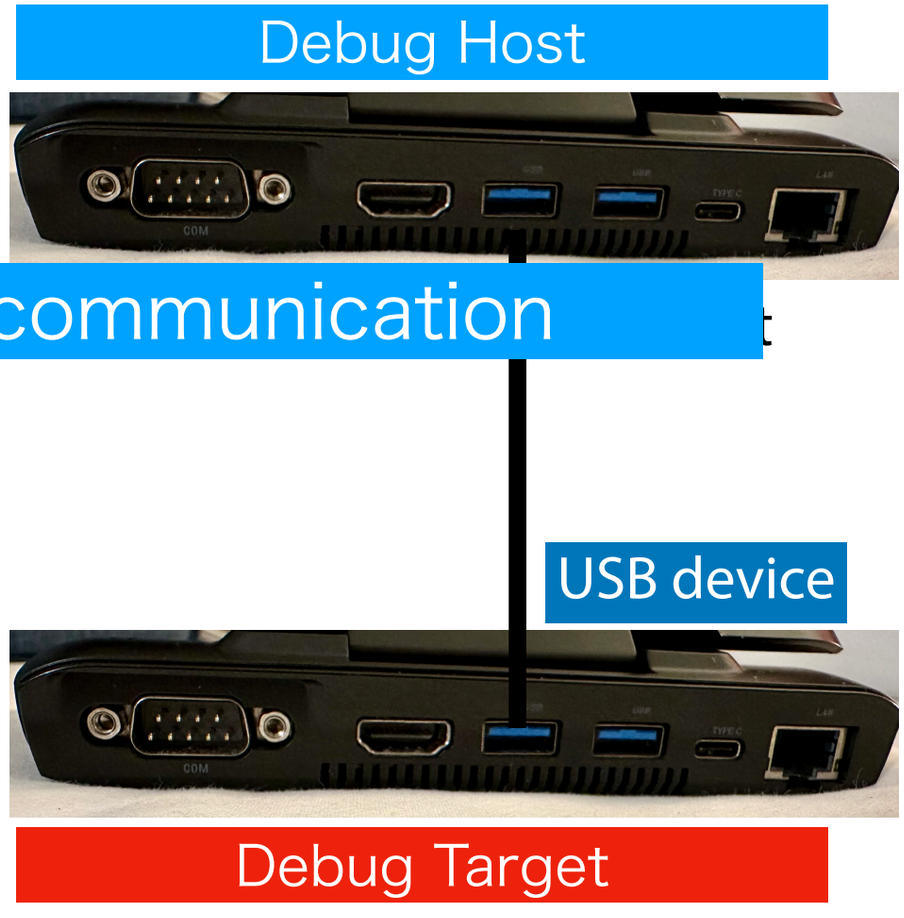


https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Software Components for DbC

- **On the Debug Host**
  - A normal USB3 stack is sufficient. No DbC required.
  - udbc(4) driver for simple serial communication
  - because the USB device has USB Debug Class (0xdc in the bInterfaceClass field)

- **On the Debug Target**
  - Activation of DbC is required.
  - DbC has two endpoints (IN and OUT) for bulk transfer
  - Console backend in the loader and the kernel
  - TRB ring buffers for IN and OUT must be allocated in memory (DMA will handle them)



https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Physical Setup

- **A-to-A USB3 Cable between the two**
  - On the debug target, **one of the ports on Root Hub** will become USB device.
  - This means that you have to find ports associated with the Root Hub. **Any USB 2.0 ports do not work.**

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Physical Setup

- **A-to-A USB3 Cable between the two**
  - On the debug target, **one of the ports on Root Hub** will become USB device.
  - This means that you have to find ports associated with the Root Hub. **Any USB 2.0 ports do not work.**

- **I am distributing A-A cross cable + A-A extension + A-C adapter + Beastie charm for 20 EUR here.** 8 sets are available. Catch me if you are interested in them.

- kevans@ told me that he got a cable from the following URL. It should work as well:
  - https://www.datapro.net/products/usb-3-0-super-speed-a-a-debugging-cable.html

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

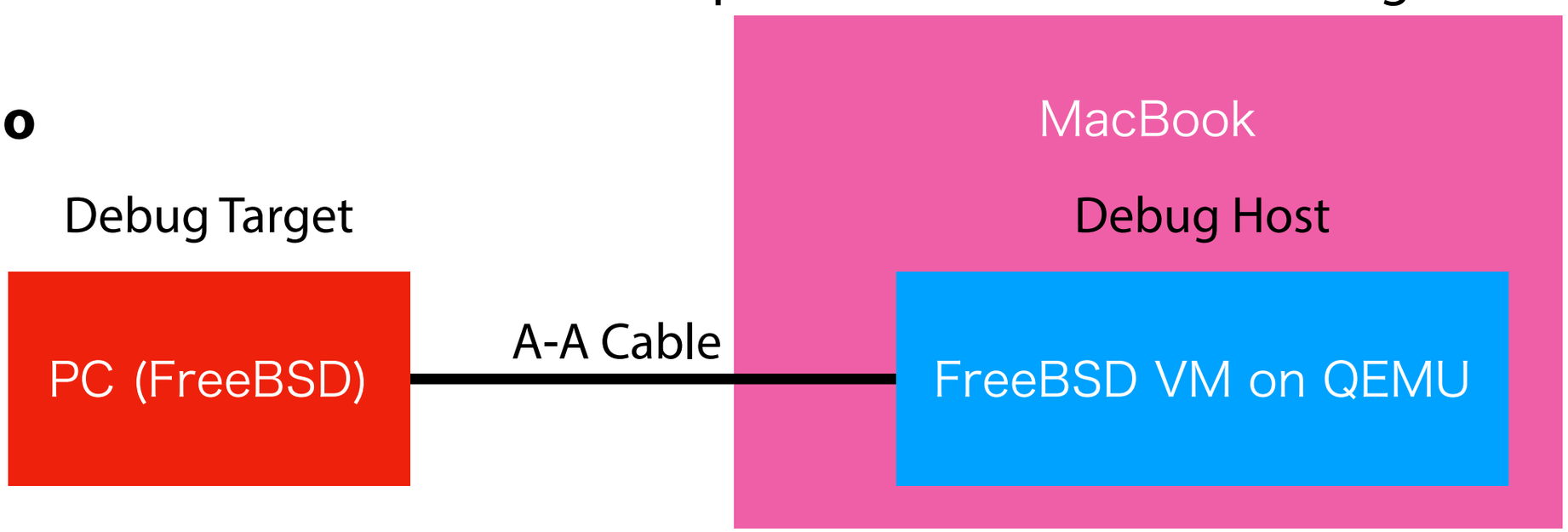
Demo and Call for Test

- **An implementation for testing is ready for you:**
  - **A unidirectional communication test from Target to Host**
    https://people.allbsd.org/~hrs/FreeBSD/udbc/20230915/
    - Build and install udbc(4) on Debug Host
    - Boot using the memstick image on Debug Target
    - Read README.udbc for more details
    - Needs more information about if DbC works on xHCI
      - **AMD xHCIs are not supported yet in the above image**.
        Send me your "`pciconf -lv | grep ^xhci`"
      - USB-C seems to have a linkup issue and I need to investigate it more

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Demo and Call for Test

- **An implementation for testing is ready for you:**
  - **A unidirectional communication test from Target to Host**
    https://people.allbsd.org/~hrs/FreeBSD/udbc/20230915/
    - Build and install udbc(4) on Debug Host
    - Boot using the memstick image on Debug Target
    - Read README.udbc for more details
    - Needs more information about if DbC works on xHCI
      - **AMD xHCIs are not supported yet in the above image**.
        Send me your "`pciconf -lv | grep ^xhci`"
      - USB-C seems to have a linkup issue and I need to investigate it more
- **Demo**



TODOs and Future Work

- I will submit patches after getting feedback about compatibility:
  - udbc(4)
  - simple serial console backend support in loader(8) and kernel
  - Both legacy loader and UEFI loader are supported.
    - This should not be x86-specific, and should be easy to port to other *BSDs.

- This can actually be extended to **mimic other type of USB devices** as long as drivers are prepared on the host side. My plans include:
  - **Target disk mode**: disk devices are exposed to the host when the loader runs. Maybe useful for installation or diagnostics.

https://people.allbsd.org/~hrs/FreeBSD/udbc/sato-20230916-usbdbc.pdf

Summary

- USB DbC is a feature to change one of the ports on a USB host for a USB device.

- The USB device has two EPs. You can receive/send any data over the IN and OUT pipes (virtual serial channels).

- A-A USB3 cable is required (again, catch me you want one). 5Gbps speed is supported at least.

- I need more information about device compatibility. Please try the test and let me know your xHCI device id and if it works or not.

Questions/Comments/Suggestions?

Please send your feedback to hrs@FreeBSD.org