Server platforms: experiment with your expensive hardware too!

Jeremy Kerr

jk@codeconstruct.com.au / @CodeConstruct

code construct

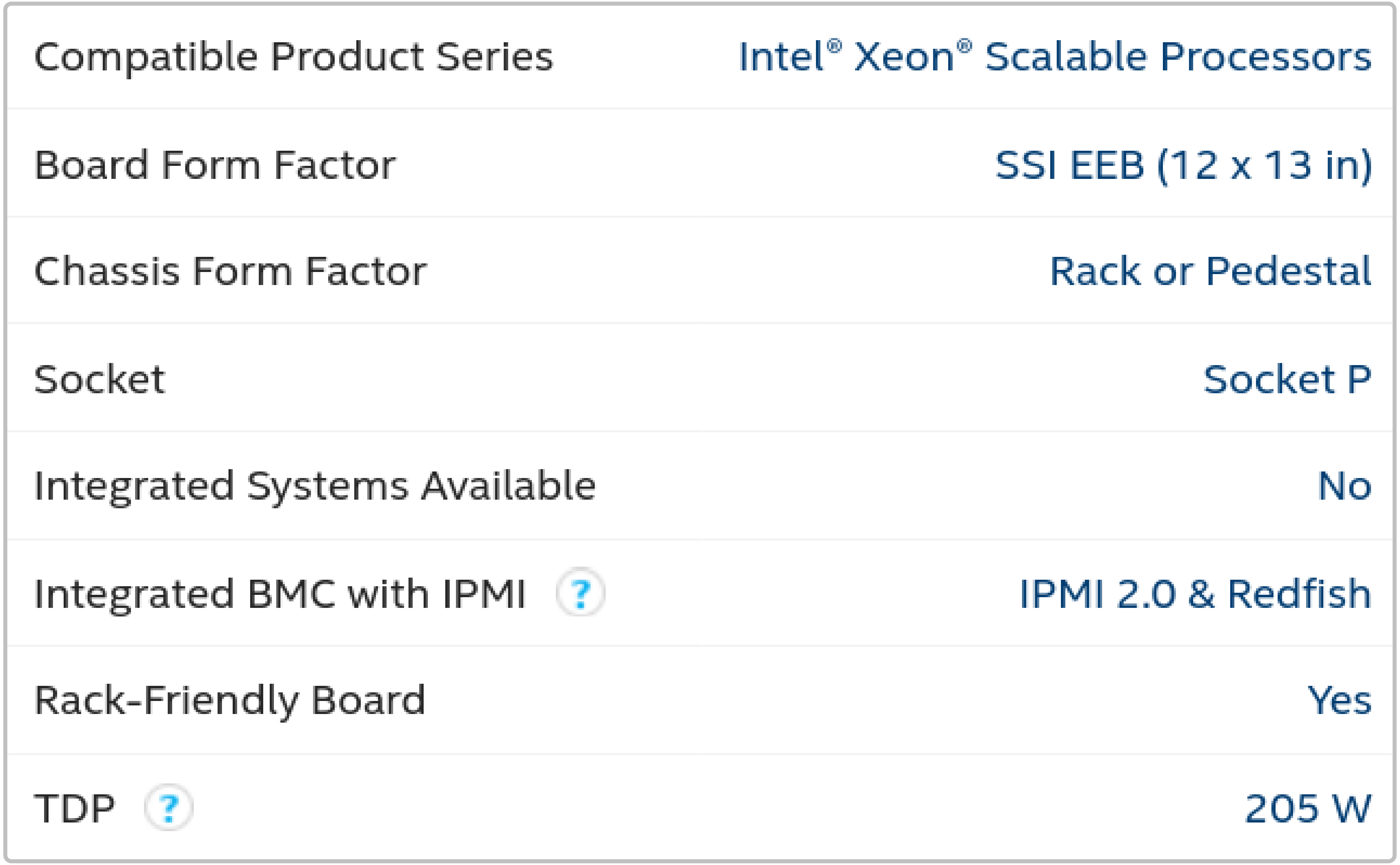
Server platforms: ~~experiment with~~ break your expensive hardware too!

Jeremy Kerr

jk@codeconstruct.com.au / @CodeConstruct

code construct

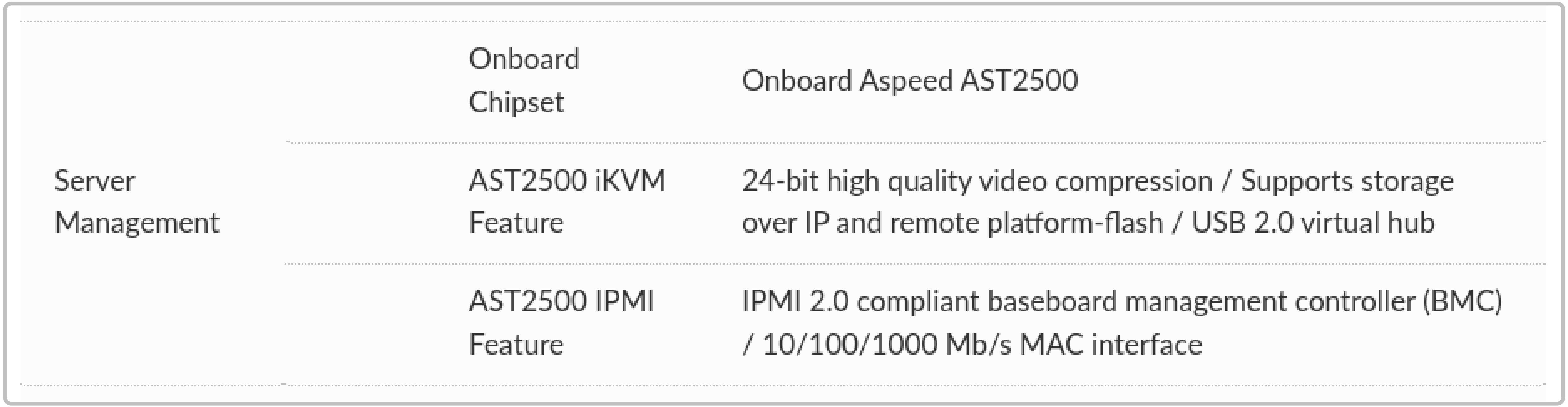
Servers?



Compatible Product Series	Intel® Xeon® Scalable Processors
Board Form Factor	SSI EEB (12 x 13 in)
Chassis Form Factor	Rack or Pedestal
Socket	Socket P
Integrated Systems Available	No
Integrated BMC with IPMI ?	IPMI 2.0 & Redfish
Rack-Friendly Board	Yes
TDP ?	205 W

On-Board Devices

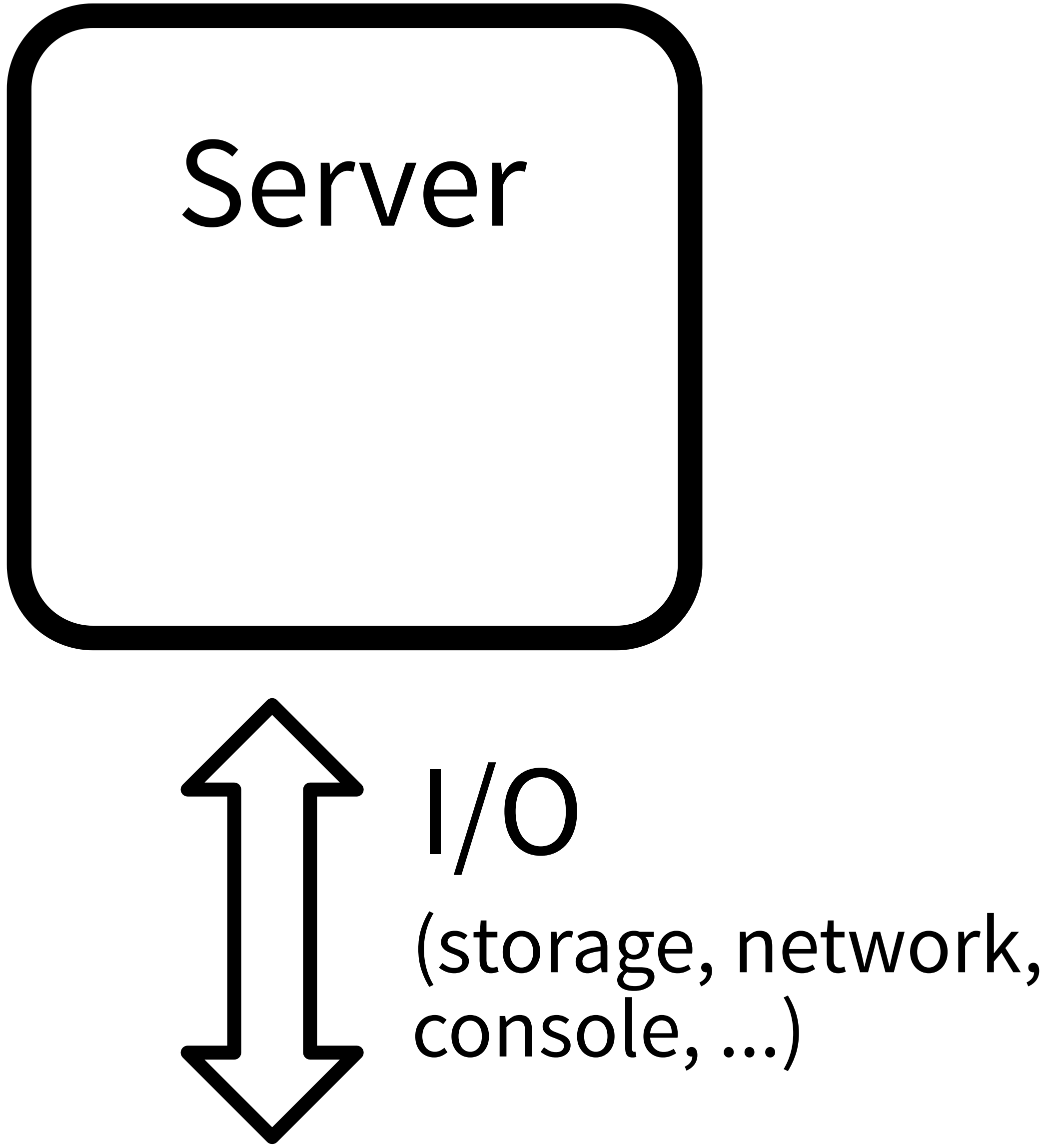
SATA	▪ SATA3 (6 Gbps)
IPMI	▪ Support for Intelligent Platform Management Interface v. 2.0 ▪ IPMI 2.0 with virtual media over LAN and KVM-over-LAN support ▪ ASPEED AST2500 BMC
Network Controllers	▪ Provided via Ultra Riser Card ▪ 1 Realtek RTL8211F PHY (dedicated IPMI)
VGA	▪ ASPEED AST2500 BMC



Server Management	Onboard Chipset	Onboard Aspeed AST2500
	AST2500 iKVM Feature	24-bit high quality video compression / Supports storage over IP and remote platform-flash / USB 2.0 virtual hub
	AST2500 IPMI Feature	IPMI 2.0 compliant baseboard management controller (BMC) / 10/100/1000 Mb/s MAC interface

BMC?

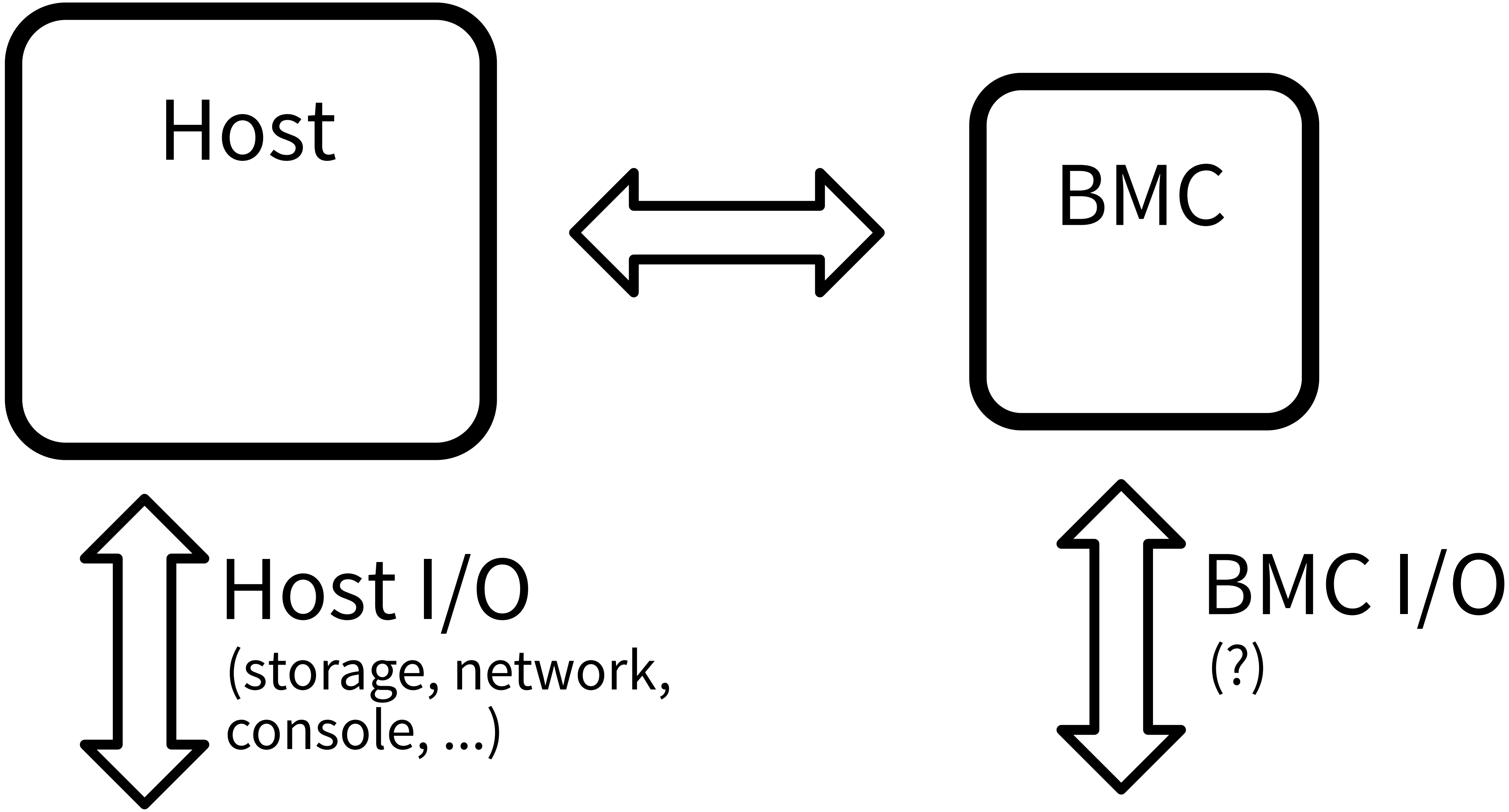
“Server-style” functionality



Server

I/O

(storage, network,
console, ...)

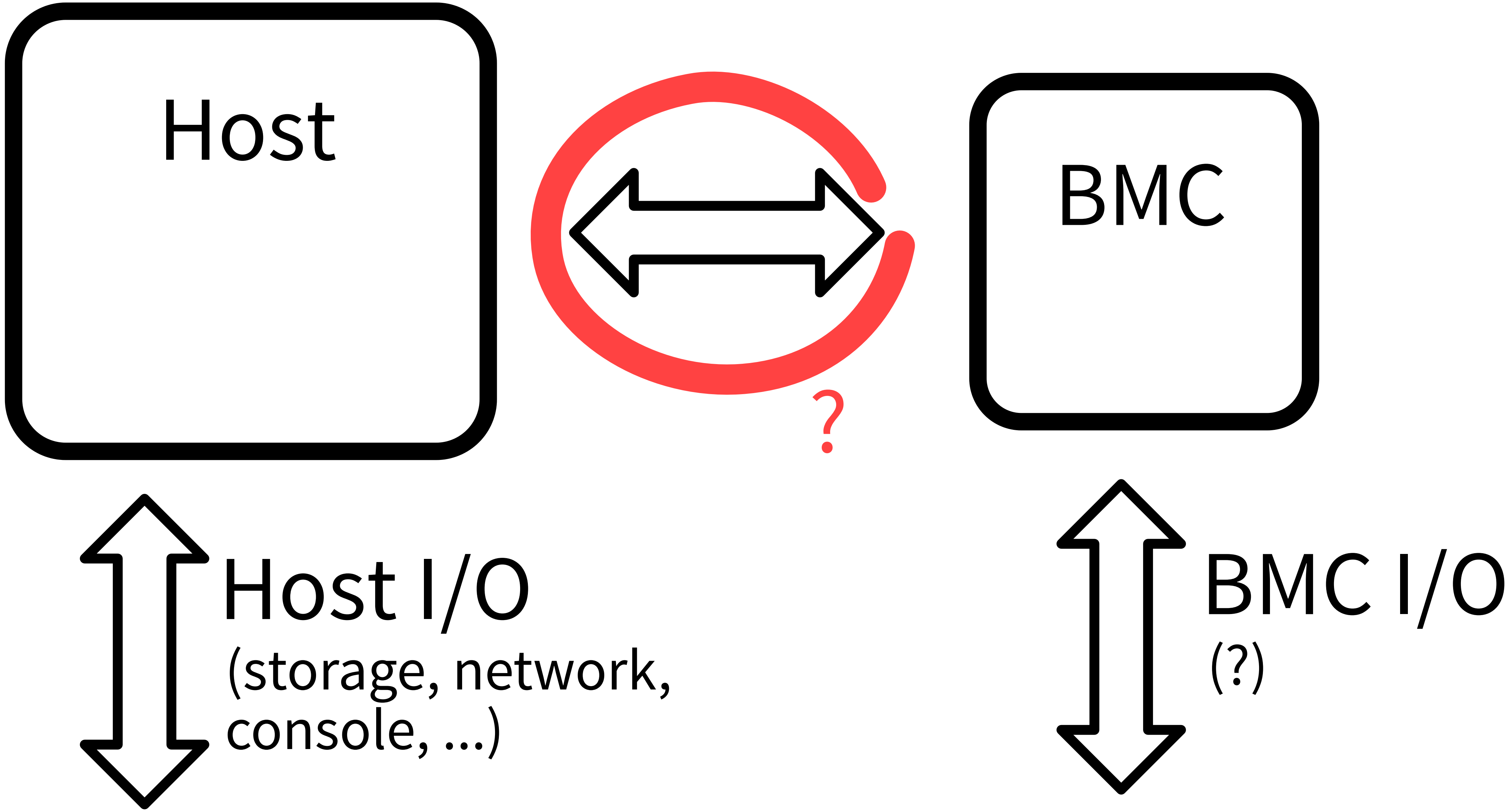


Host

BMC

Host I/O
(storage, network,
console, ...)

BMC I/O
(?)



Host

BMC

?

Host I/O

(storage, network, console, ...)

BMC I/O

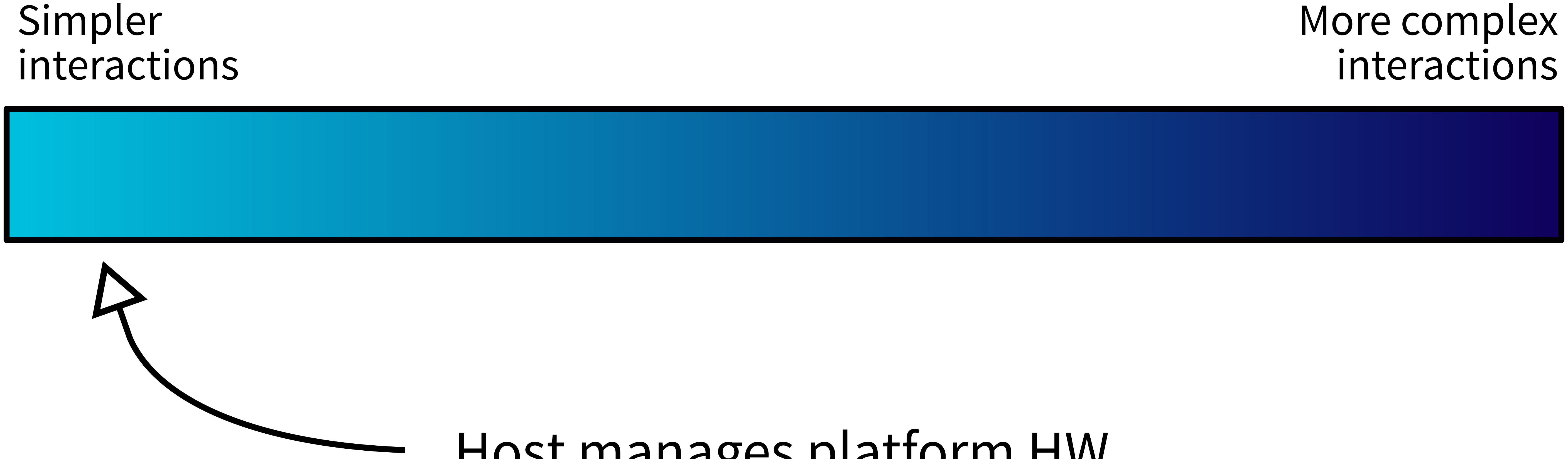
(?)

Host/BMC interactions

Simpler interactions

More complex interactions

Host/BMC interactions

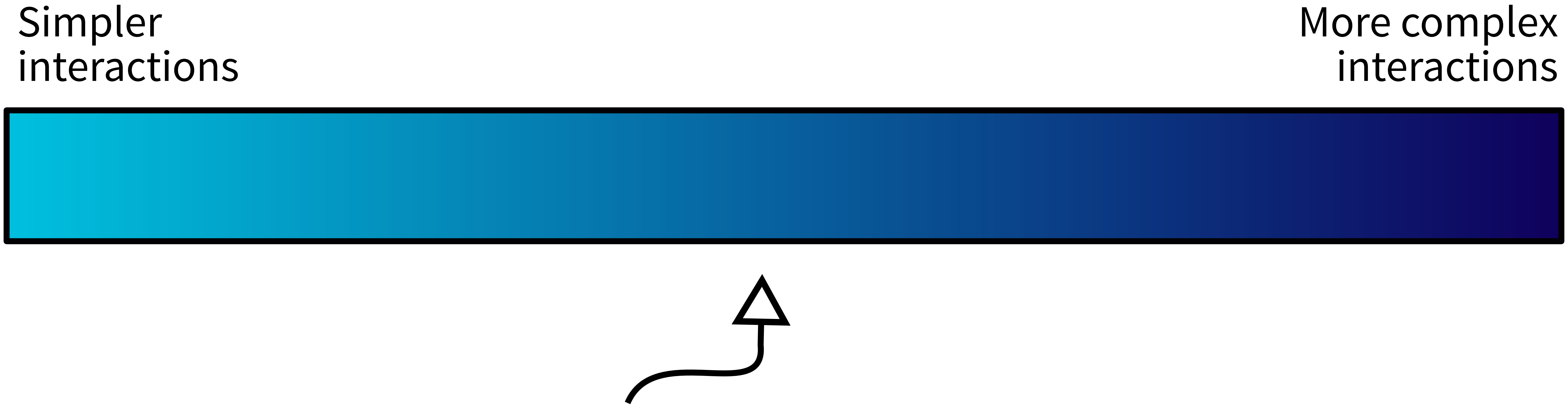


Simpler interactions

More complex interactions

Host manages platform HW
BMC remote power & console only

Host/BMC interactions

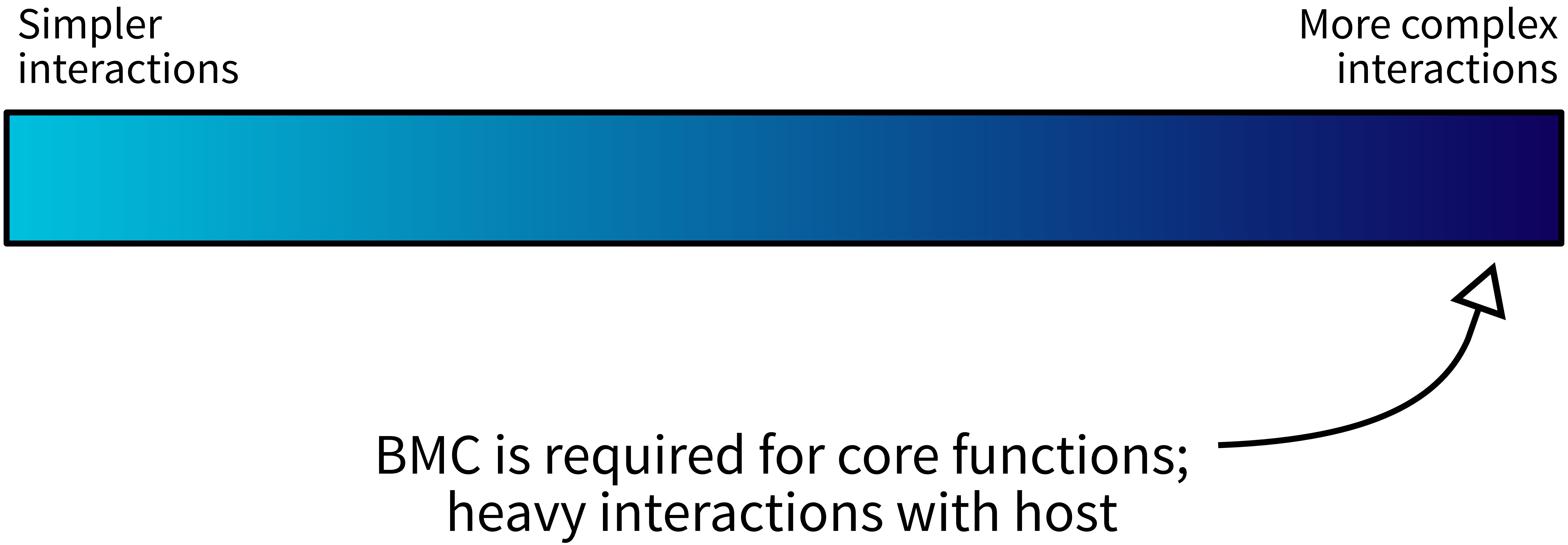


Simpler interactions

More complex interactions

Host manages most core functions
BMC performs some HW management (eg cooling)

Host/BMC interactions



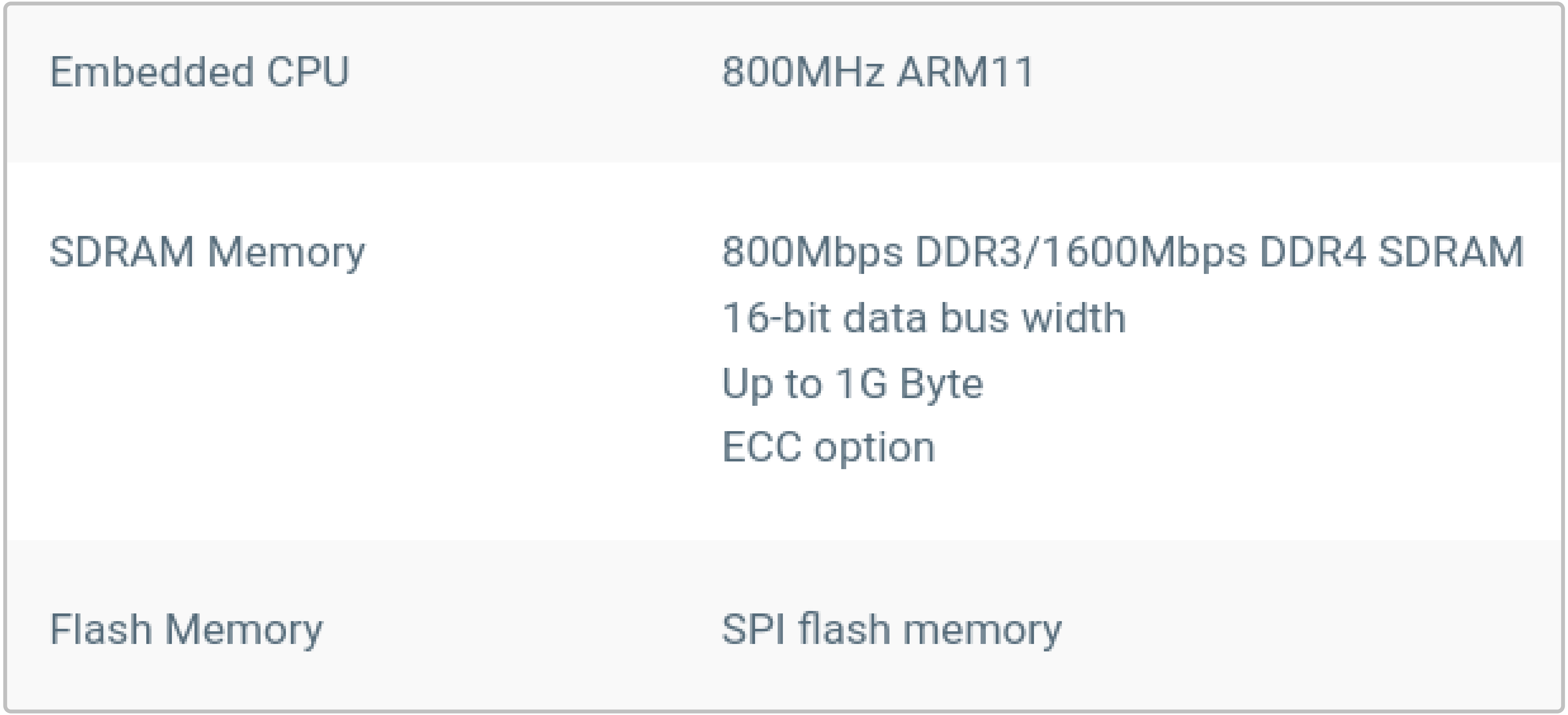
Simpler interactions

More complex interactions

BMC is required for core functions; heavy interactions with host

AST2500?

- ARM system-on-chip
- … plus hardware features for BMC functions



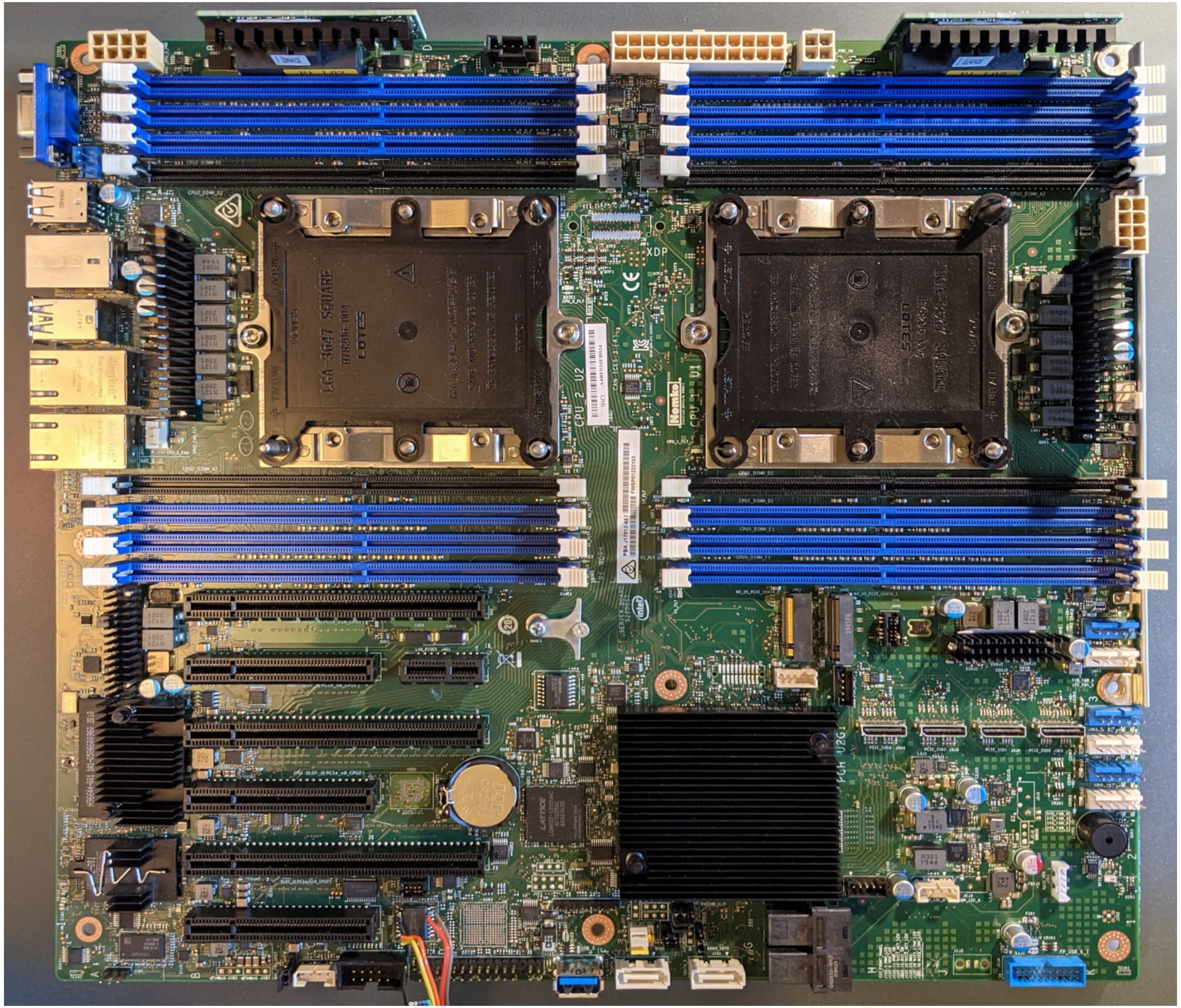
Embedded CPU	800MHz ARM11
SDRAM Memory	800Mbps DDR3/1600Mbps DDR4 SDRAM 16-bit data bus width Up to 1G Byte ECC option
Flash Memory	SPI flash memory

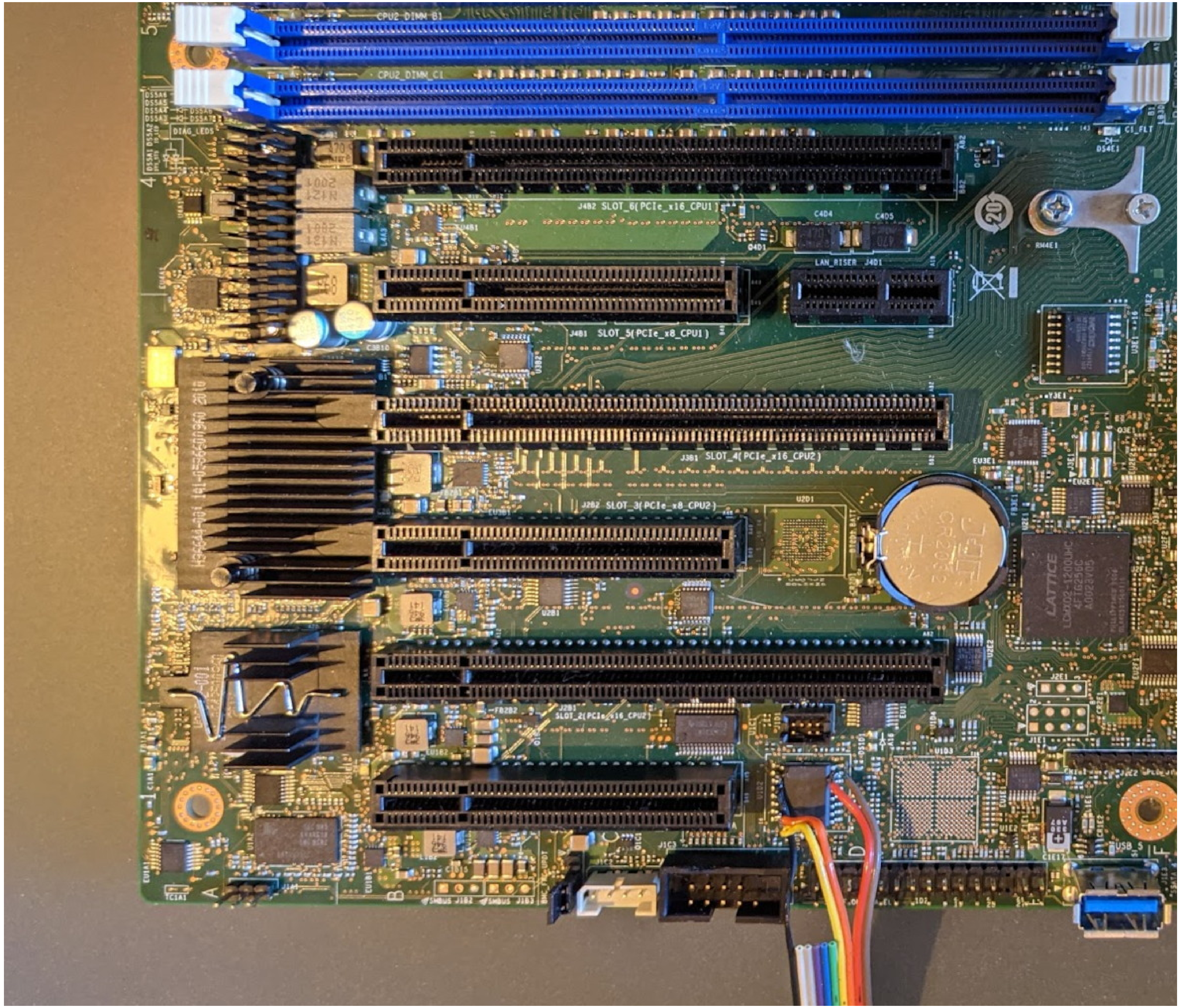
aspeedtech.com/server_ast2500/

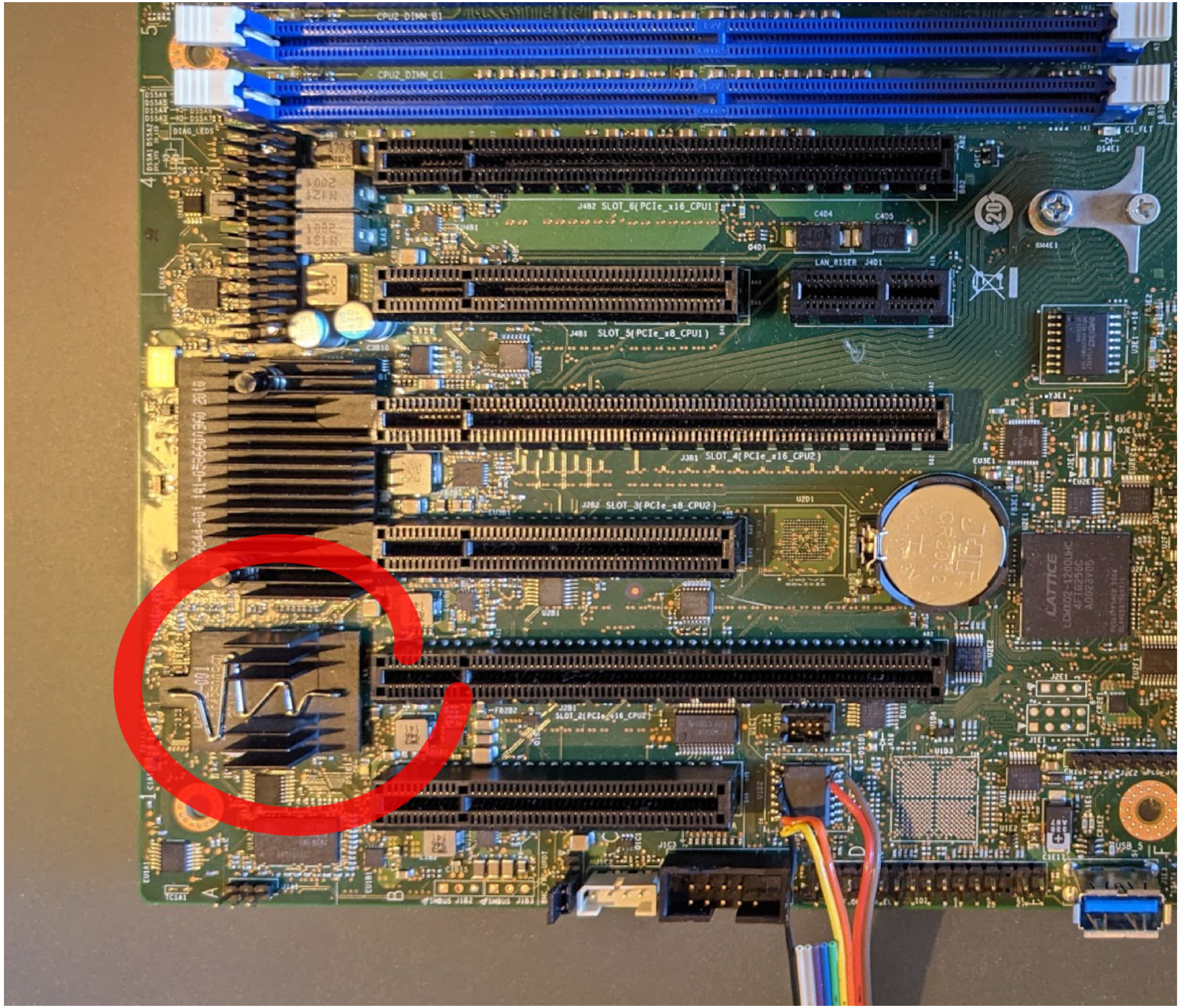
BMC	Yes

OS? Linux!

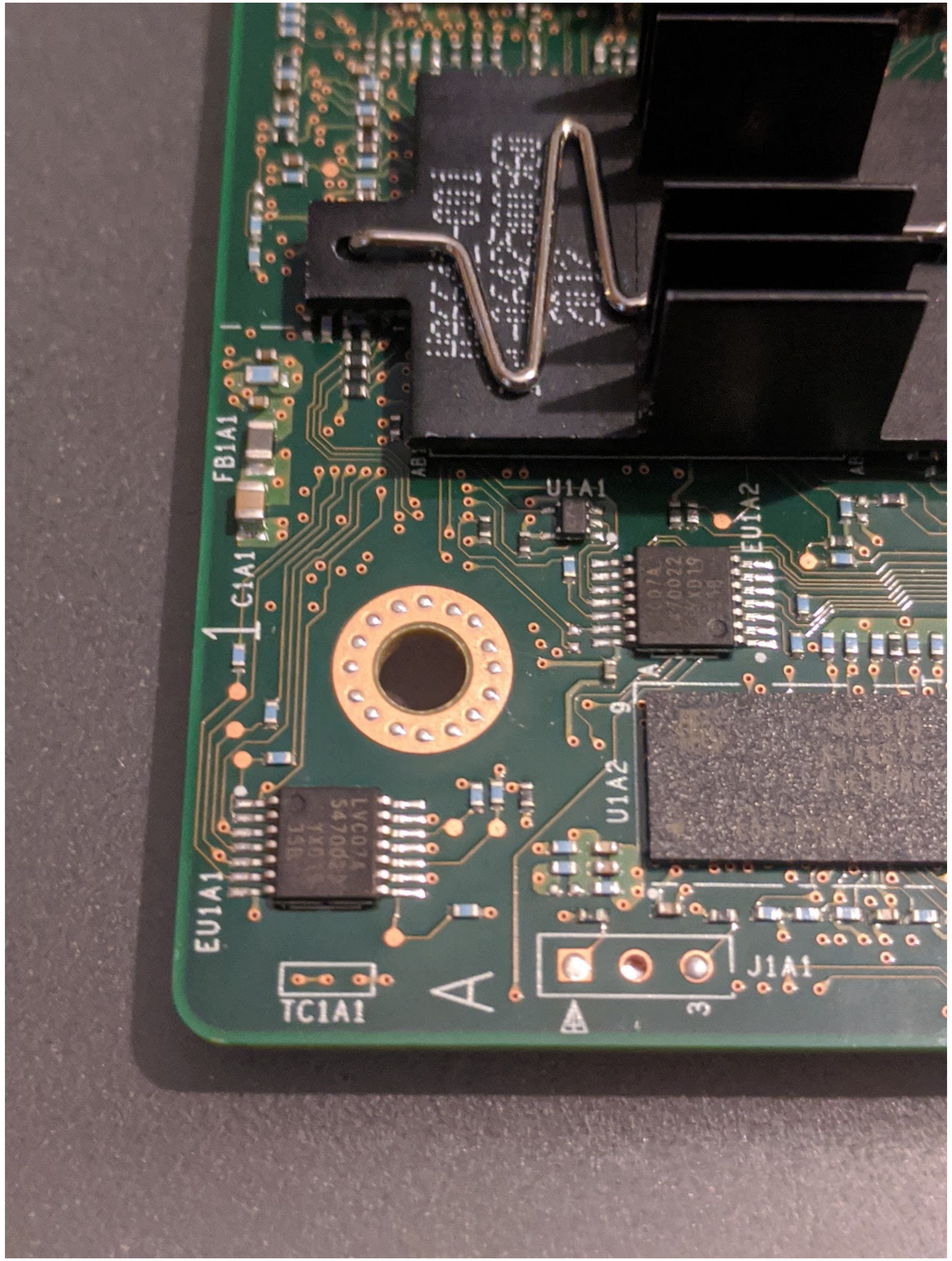
- with upstream kernel support!

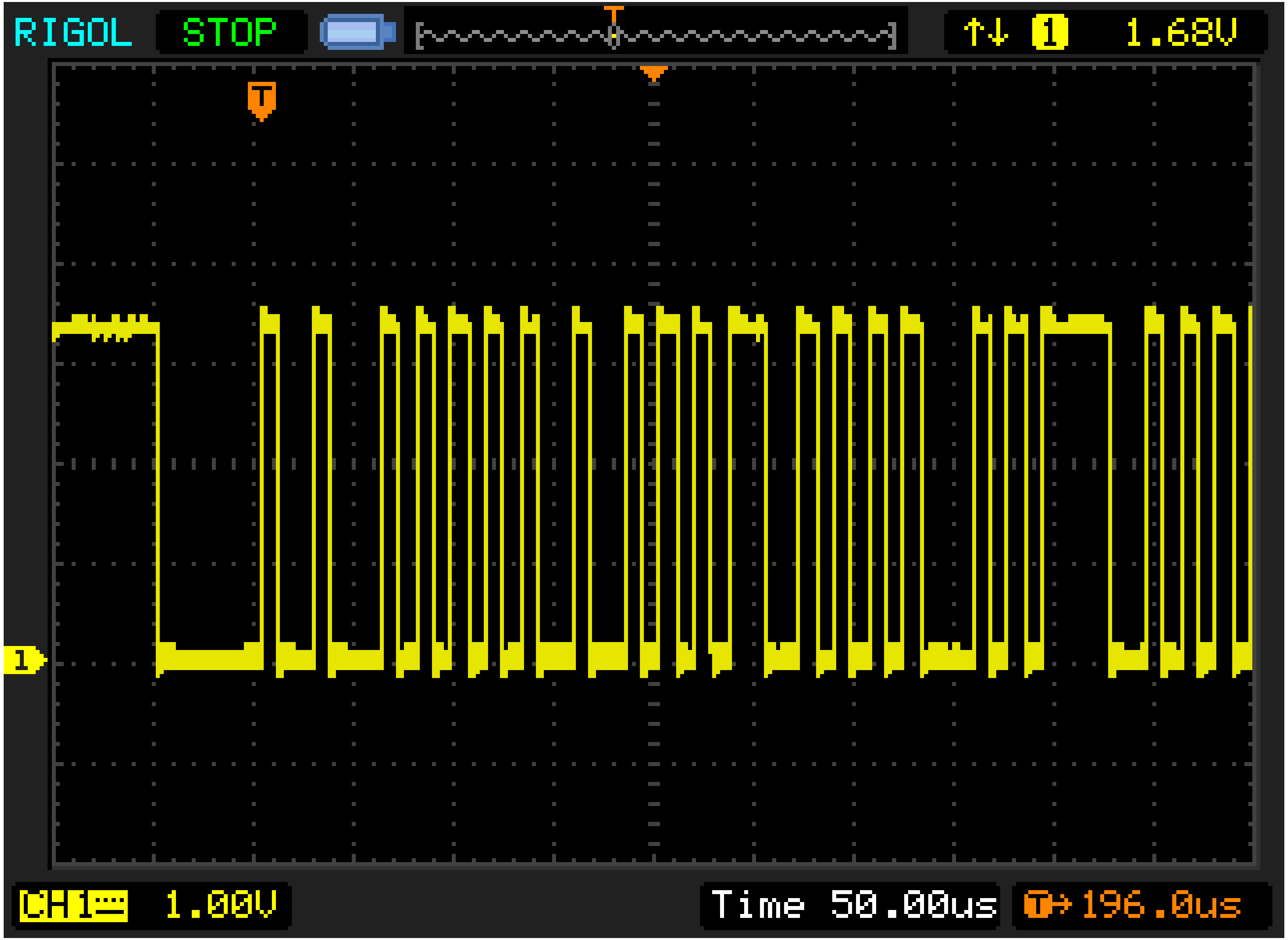

LGA 3647 SQUARE
LOTES
XDP
CPU_2_U2
CPU_1_U1
Nemko
PCH U2G

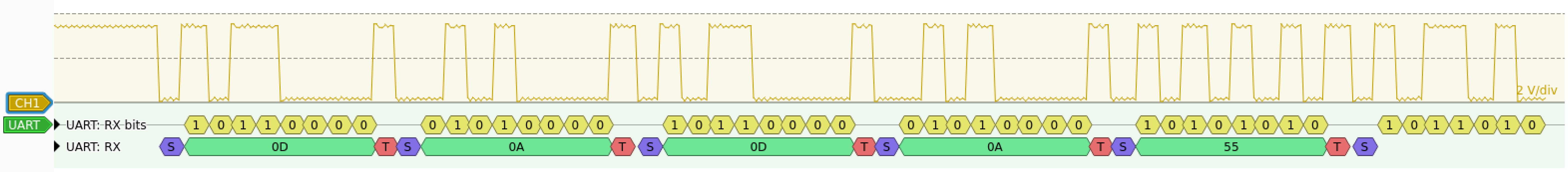

CPU2_DIMM_B1
CPU2_DIMM_C1
DIAG_LEDS
J4B2 SLOT_6(PCIe_x16_CPU1)
LAN_RISER J4D1
J4B1 SLOT_5(PCIe_x8_CPU1)
J3B1 SLOT_4(PCIe_x16_CPU2)
J2B2 SLOT_3(PCIe_x8_CPU2)
J2B1 SLOT_2(PCIe_x16_CPU2)
CR2032
LATTICE
SMBUS
USB


CPU2_DIMM_B1
CPU2_DIMM_C1
DIAG_LEDS
J4B2 SLOT_6(PCIe_x16_CPU1)
LAN_RISER J4D1
J4B1 SLOT_5(PCIe_x8_CPU1)
J3B1 SLOT_4(PCIe_x16_CPU2)
J2B2 SLOT_3(PCIe_x8_CPU2)
J2B1 SLOT_2(PCIe_x16_CPU2)
CR2032
LATTICE

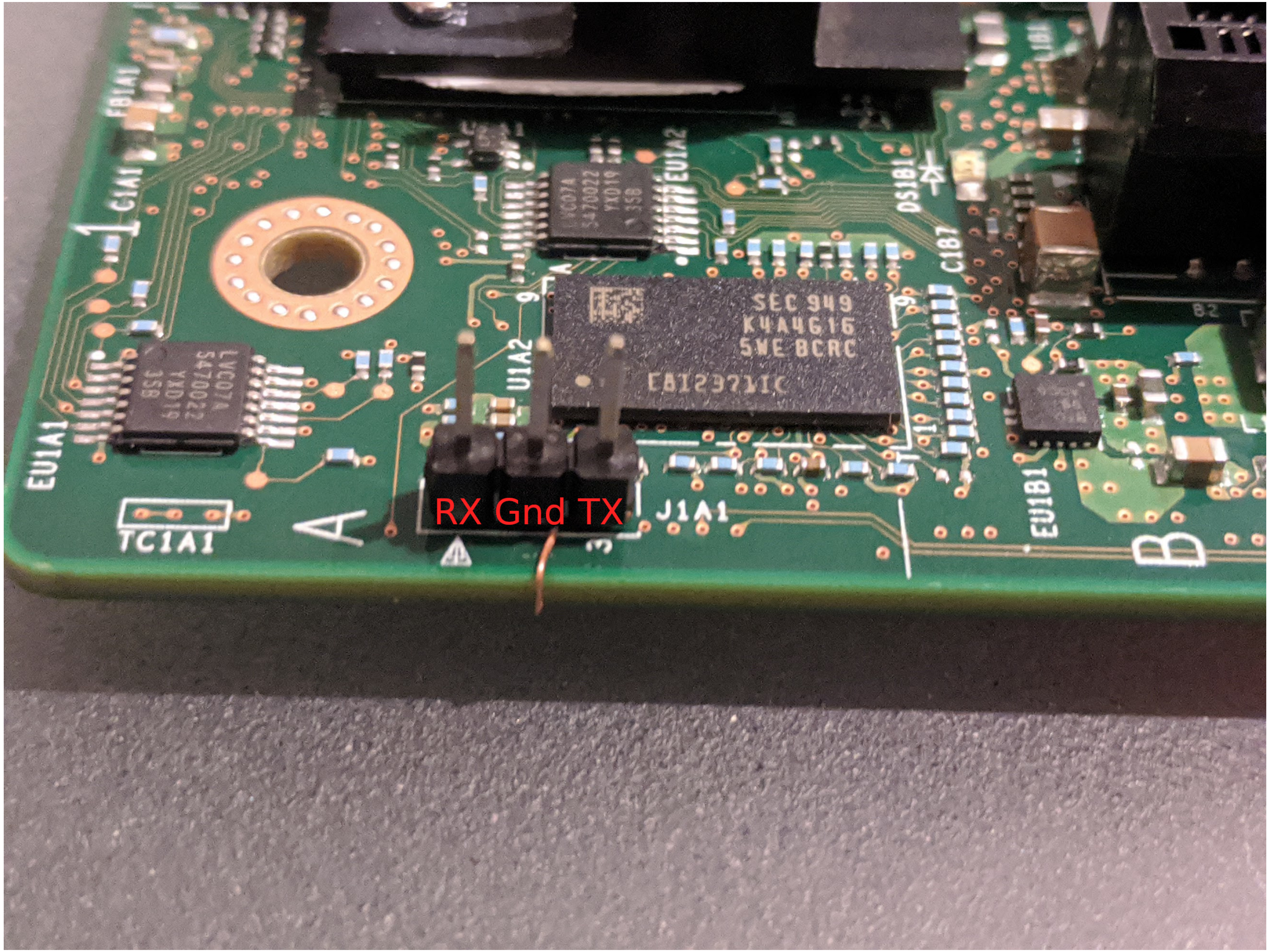
ok, but now what?

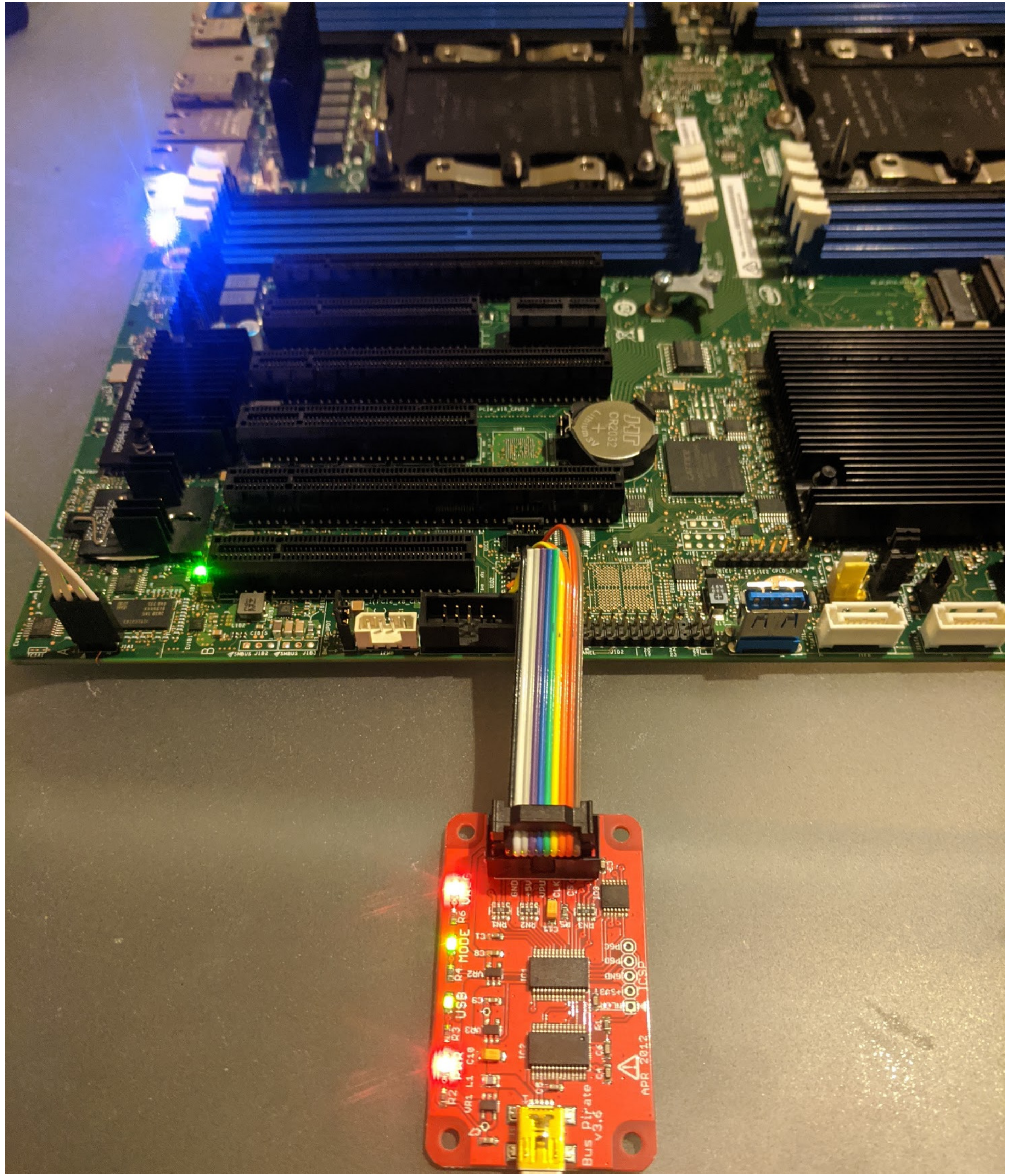

FB1A1
C1A1
AB1
U1A1
EU1A2
9
U1A2
EU1A1
TC1A1
A
J1A1
3


RIGOL
STOP
T
↑↓ 1 1.68V
T
1
CH1 1.00V
Time 50.00us
T→ 196.0us



CH1
2 V/div
UART
UART: RX bits
1 0 1 1 0 0 0 0
0 1 0 1 0 0 0 0
1 0 1 1 0 0 0 0
0 1 0 1 0 0 0 0
1 0 1 0 1 0 1 0
1 0 1 1 0 1 0
UART: RX
S 0D T S 0A T S 0D T S 0A T S 55 T S


RX Gnd TX
J1A1
U1A2
EU1A1
EU1A2
TC1A1
C1A1
FB1A1
DS1B1
C1B7
EU1B1
A
B
SEC 949
K4A4G16
5WE BCRC


CR2032
MODE
USB
Bus pirate v3.6
APR 2012

BMC firmware?

kernel

```
aspeed_g5_defconfig
```

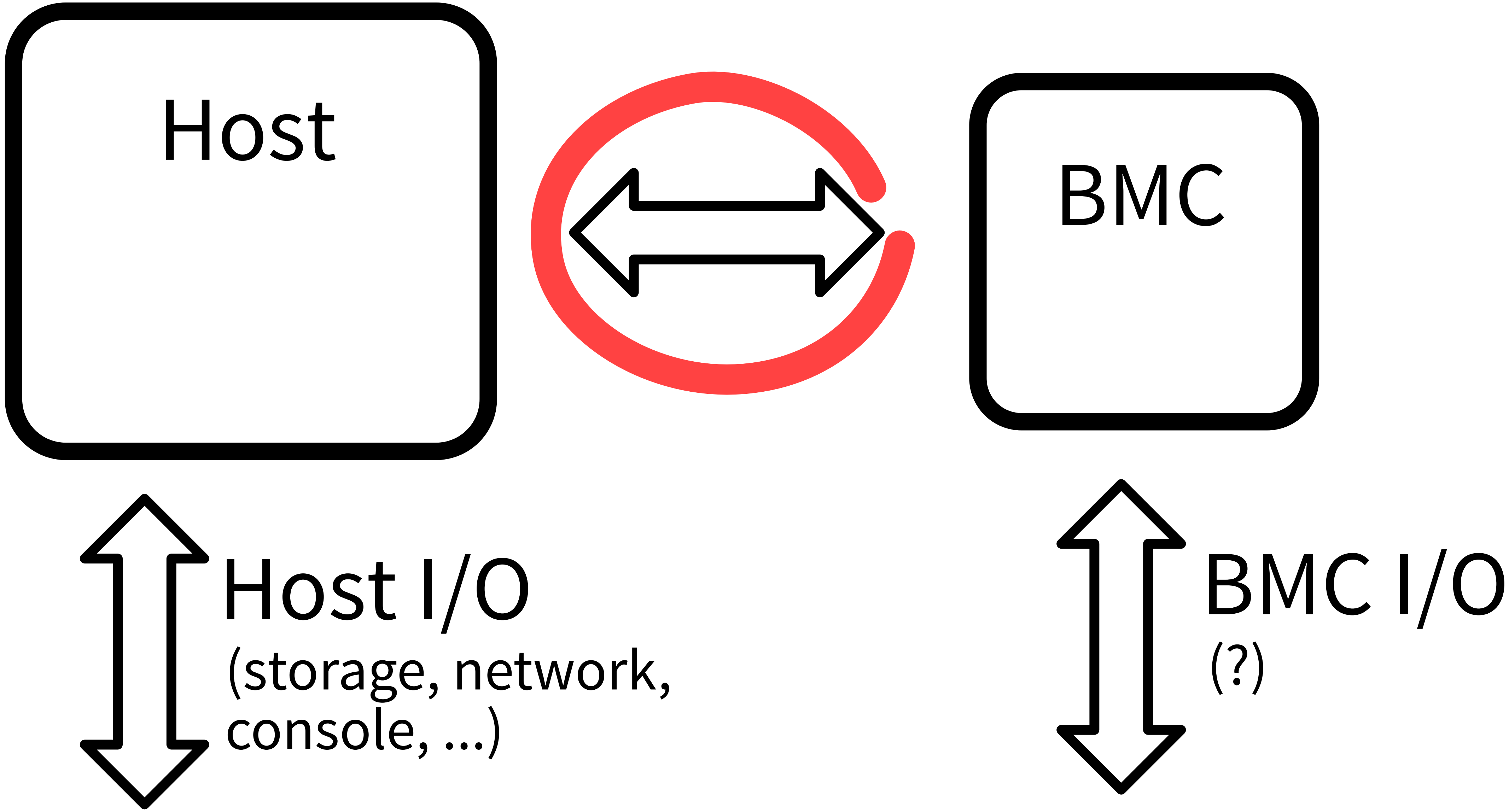
buildroot

```
BR2_arm=y
BR2_arm1176jz_s=y
BR2_TARGET_ROOTFS_CPIO=y
BR2_TARGET_ROOTFS_CPIO_XZ=y
```

(optional) u-boot

github.com/openbmc/u-boot

BMC	Yes



Host

BMC

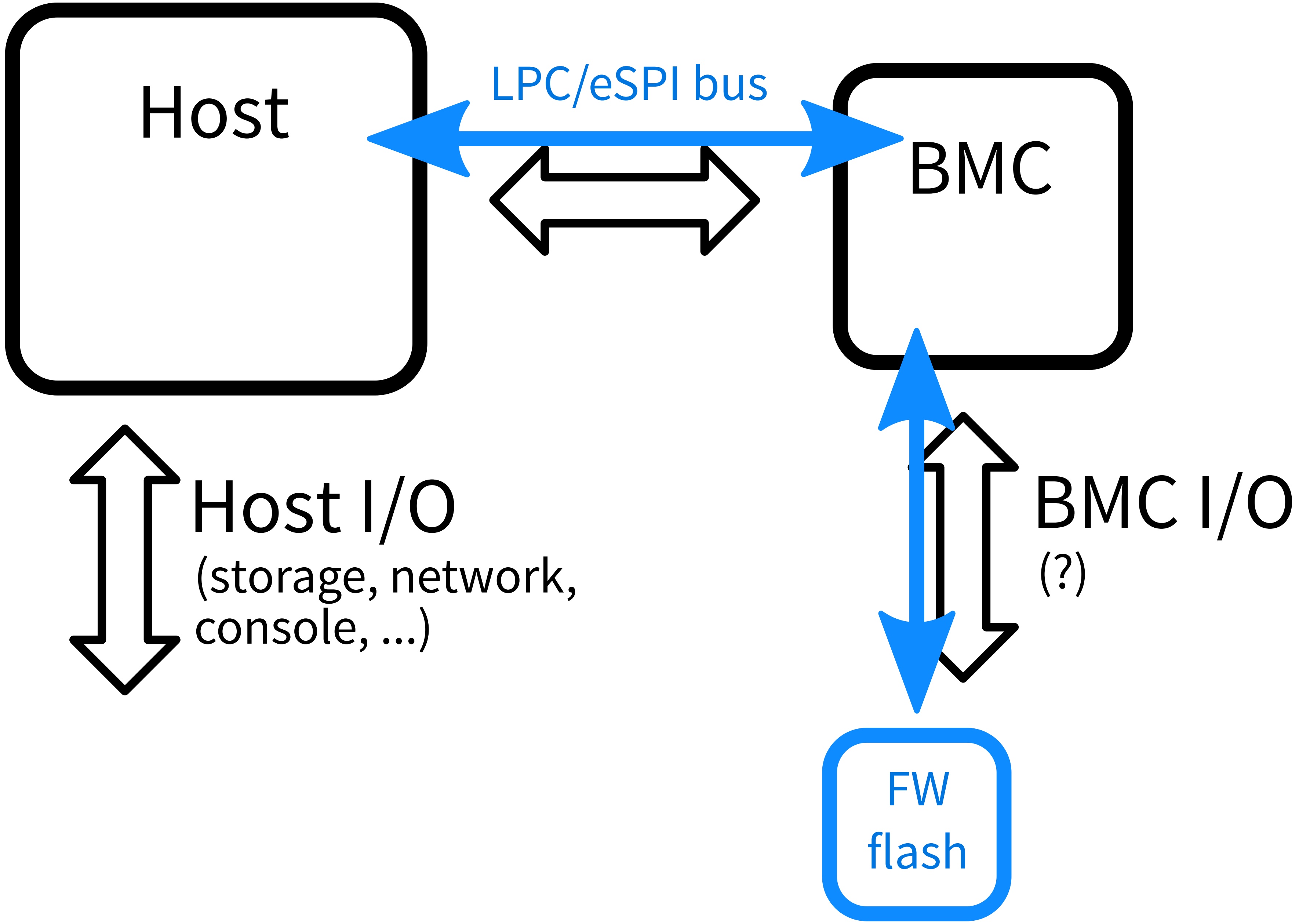
Host I/O

(storage, network, console, ...)

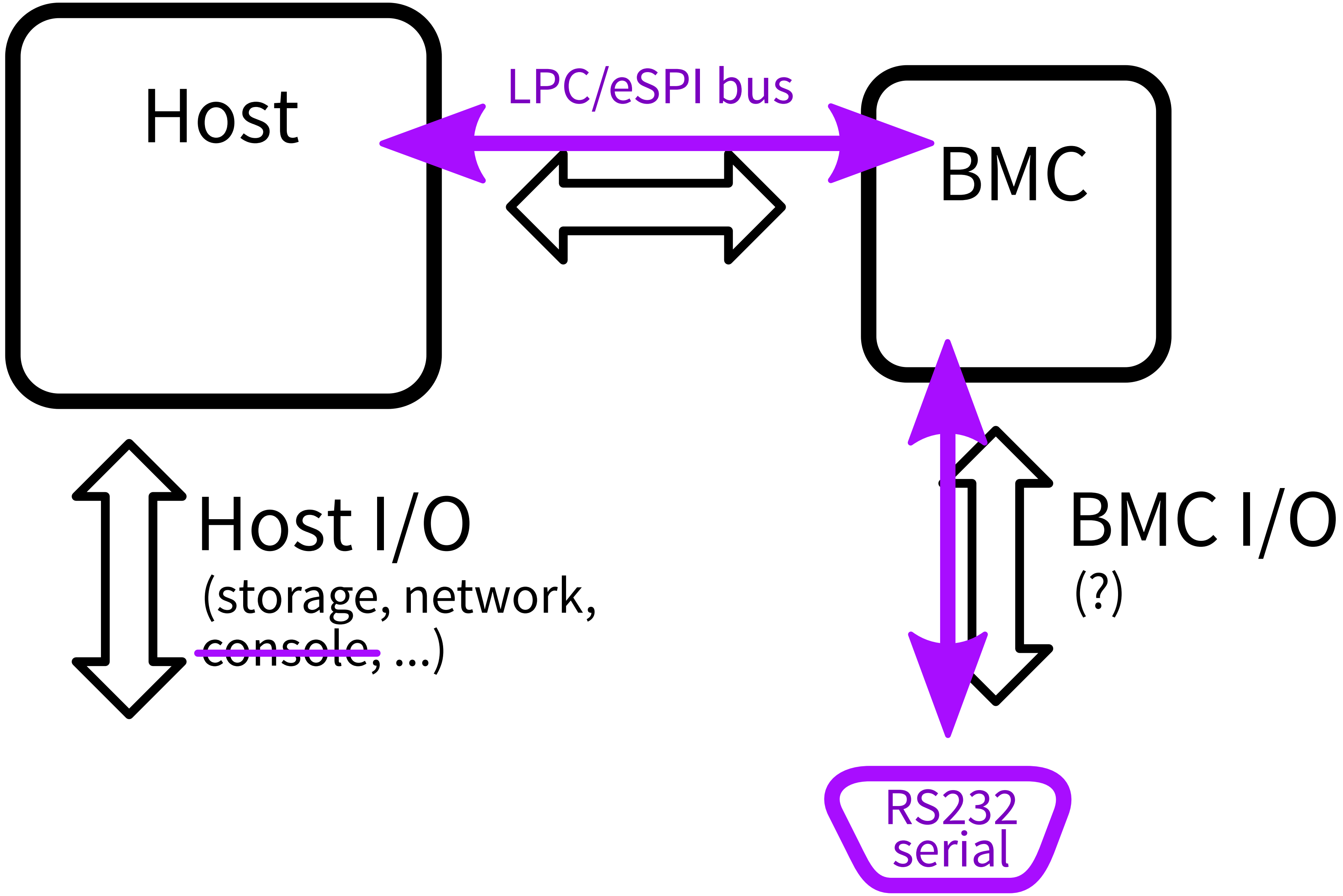
BMC I/O

(?)

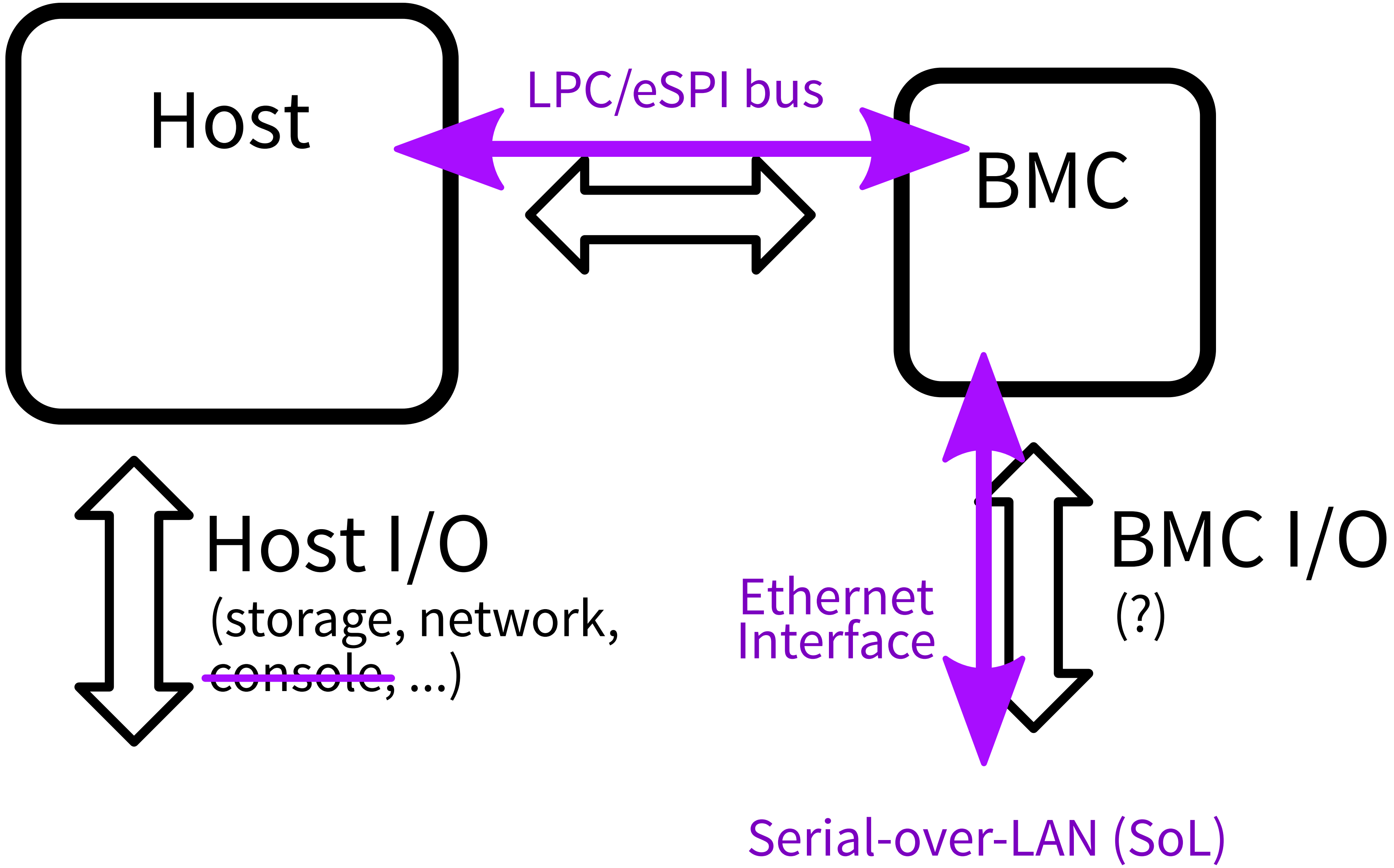
host firmware access



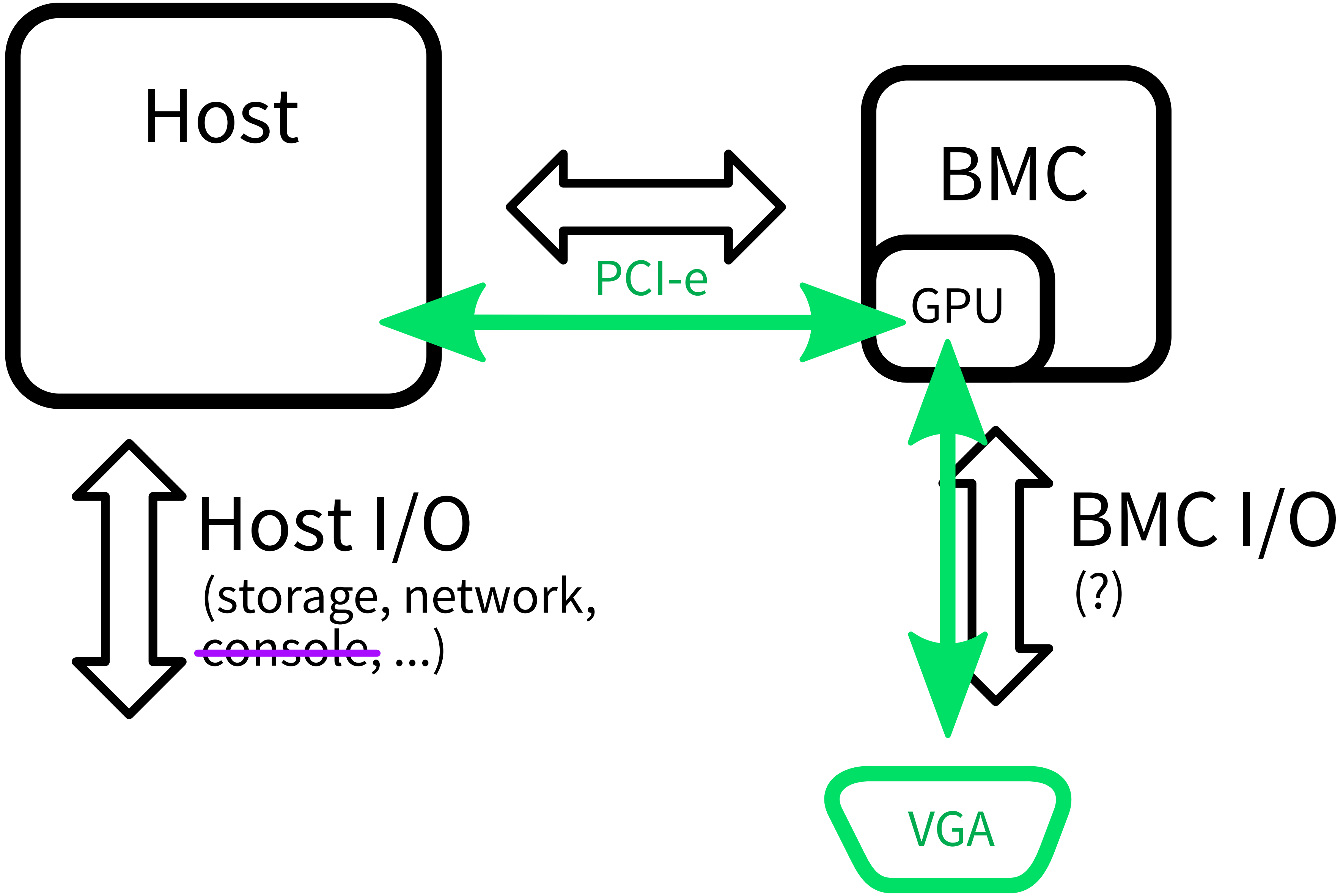
host serial


Host
LPC/eSPI bus
BMC
Host I/O
(storage, network,
~~console,~~ ...)
BMC I/O
(?)
RS232
serial

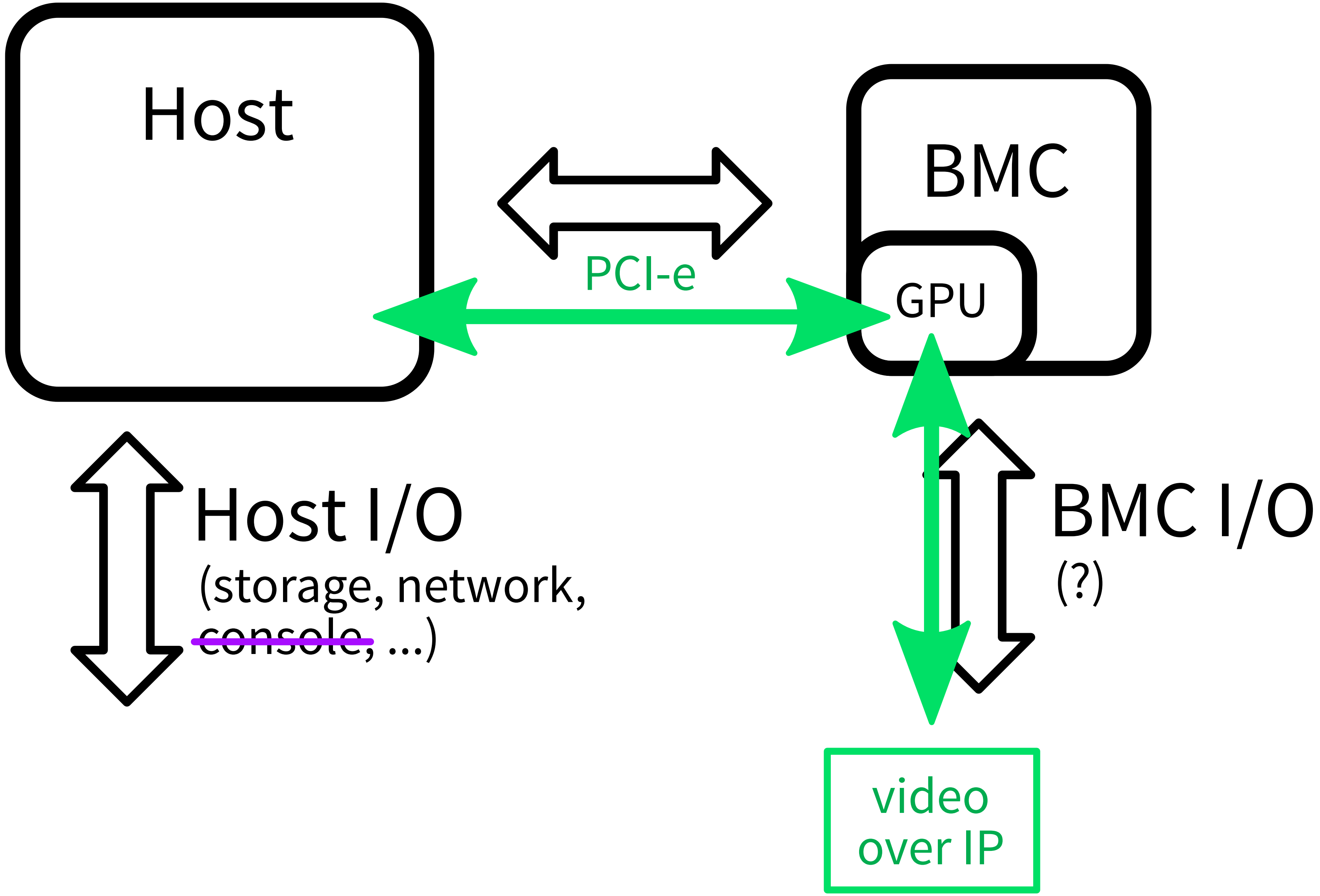
host serial


Host
LPC/eSPI bus
BMC
Host I/O
(storage, network, ~~console,~~ ...)
Ethernet Interface
BMC I/O
(?)
Serial-over-LAN (SoL)

host video



host video - remote



Host ↔ BMC interfaces

LPC/eSPI	basic IO
PCIe	video, DMA engine
USB	BMC-endpoint devices
I²C	On-chip peripherals
GPIO	Arbitrary signalling
PECI	Processor control (x86)
FSI	Processor control (OpenPOWER)

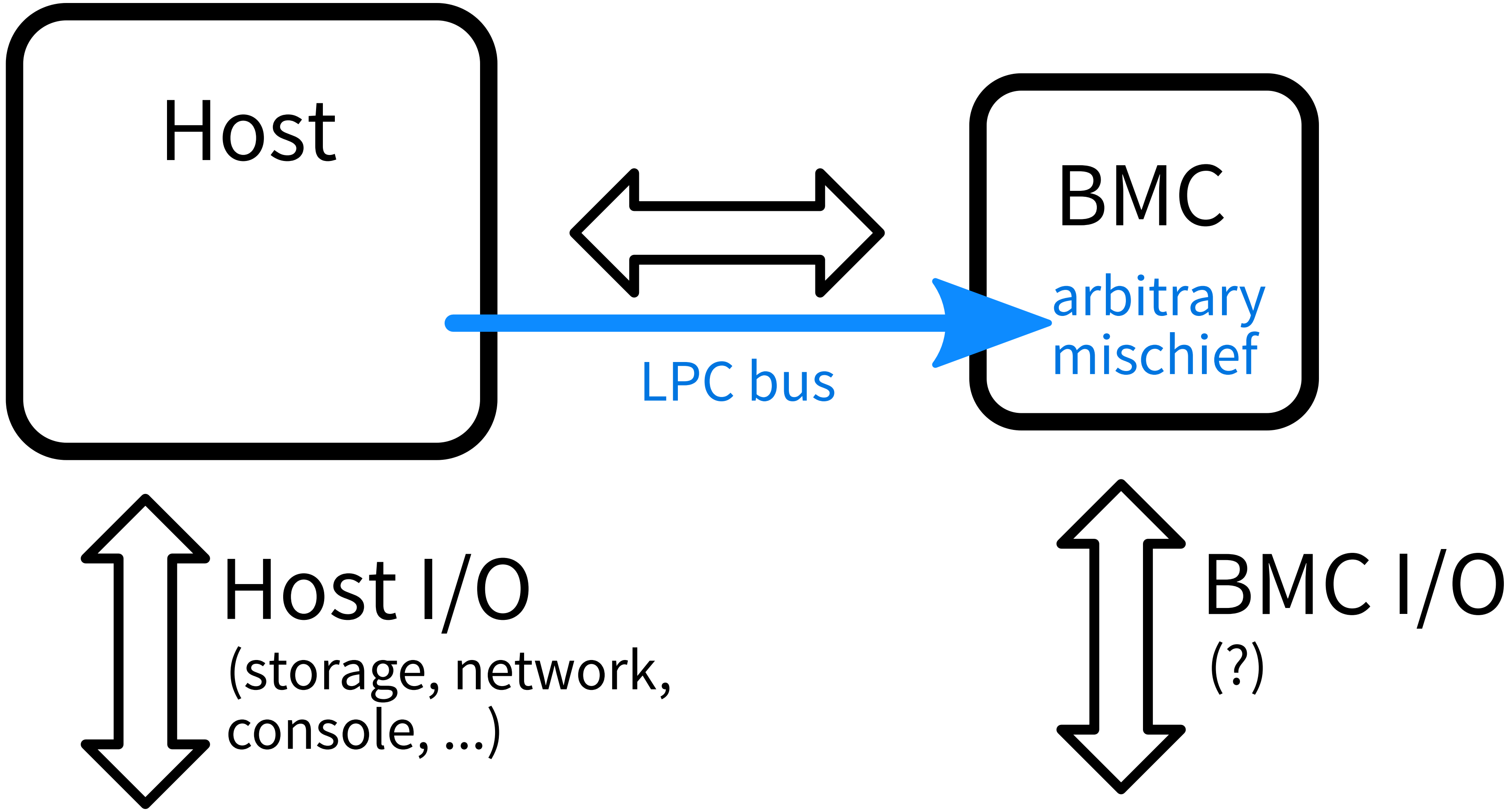
LPC devices

`0x3f8`	Serial #1
`0x2f8`	Serial #2
`0x80`	POST codes
`0xe4`	BT: in-band IPMI
configurable	LPC2AHB bridge

wait, what?

LPC	host ↔ BMC bus
2	to
AHB	*internal* BMC bus

LPC2AHB



Host

BMC

arbitrary mischief

LPC bus

Host I/O

(storage, network, console, ...)

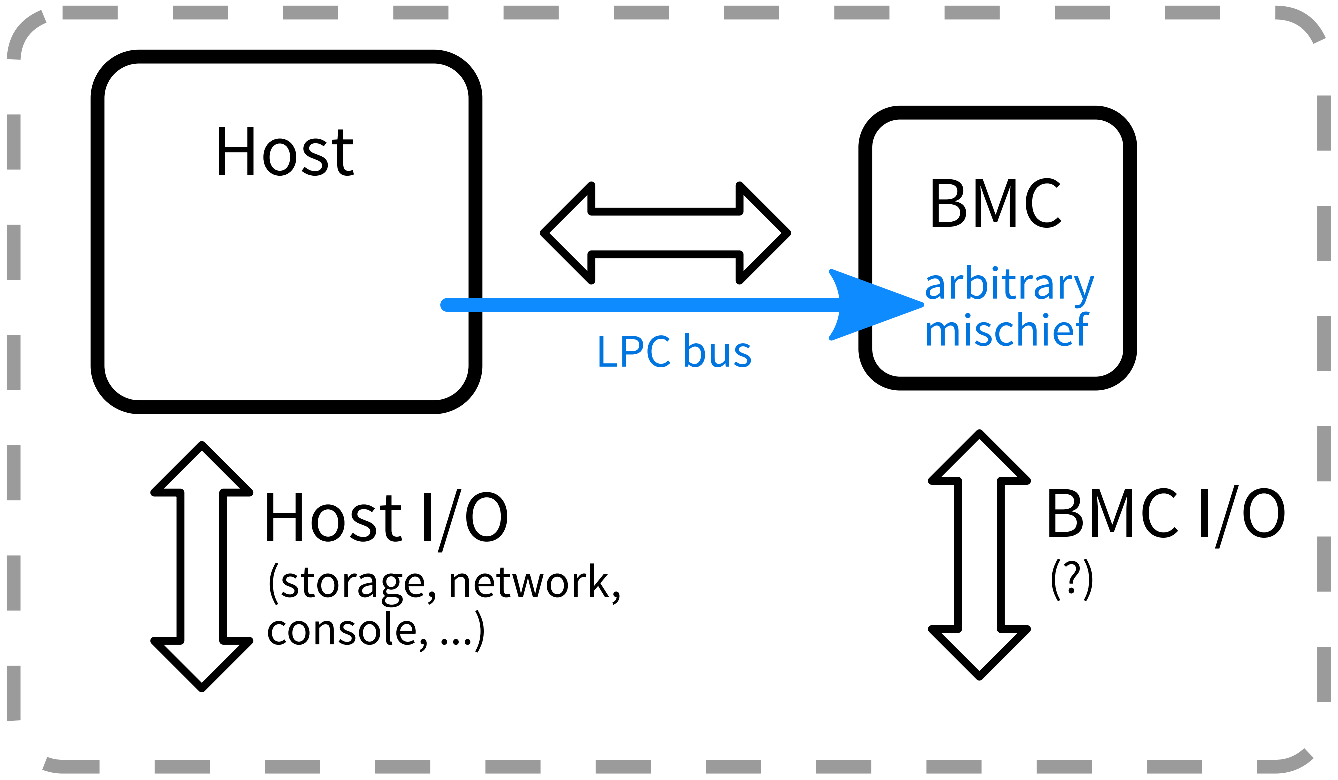
BMC I/O

(?)

CVE-2019-6260

“Gaining control of BMC from the host processor”

Owner/admin-controlled



Host

BMC

arbitrary mischief

LPC bus

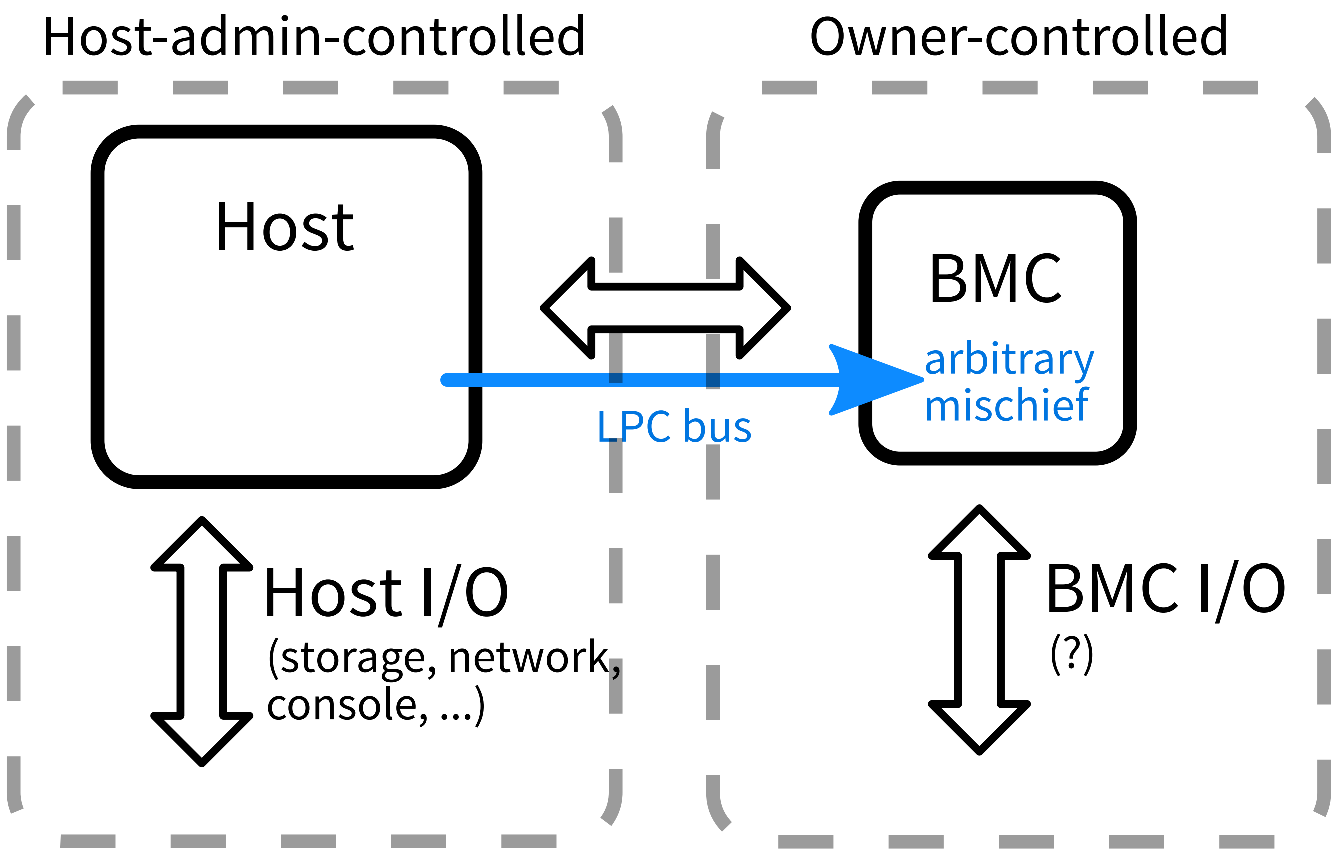
Host I/O

(storage, network, console, ...)

BMC I/O

(?)

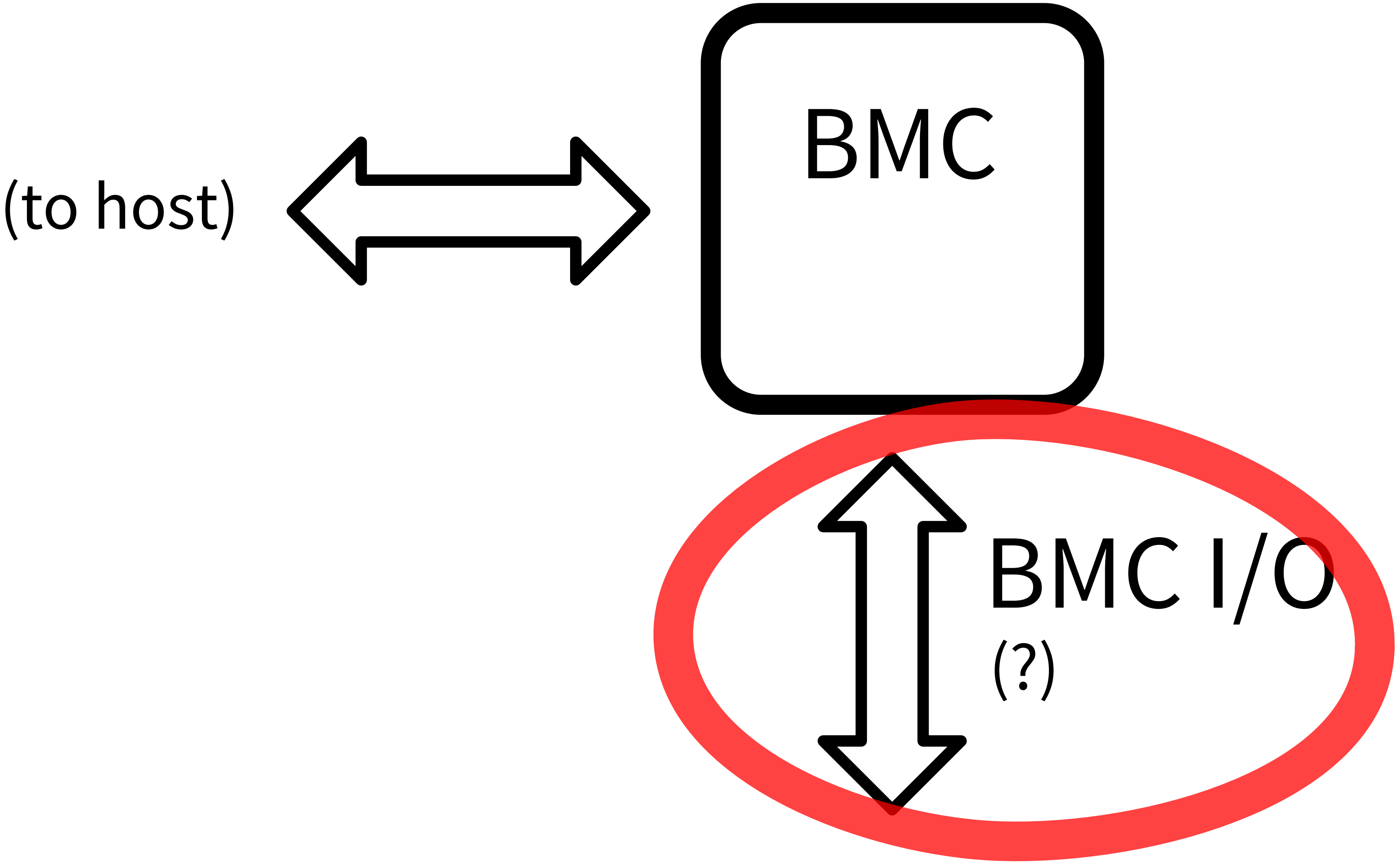
Single trust domain

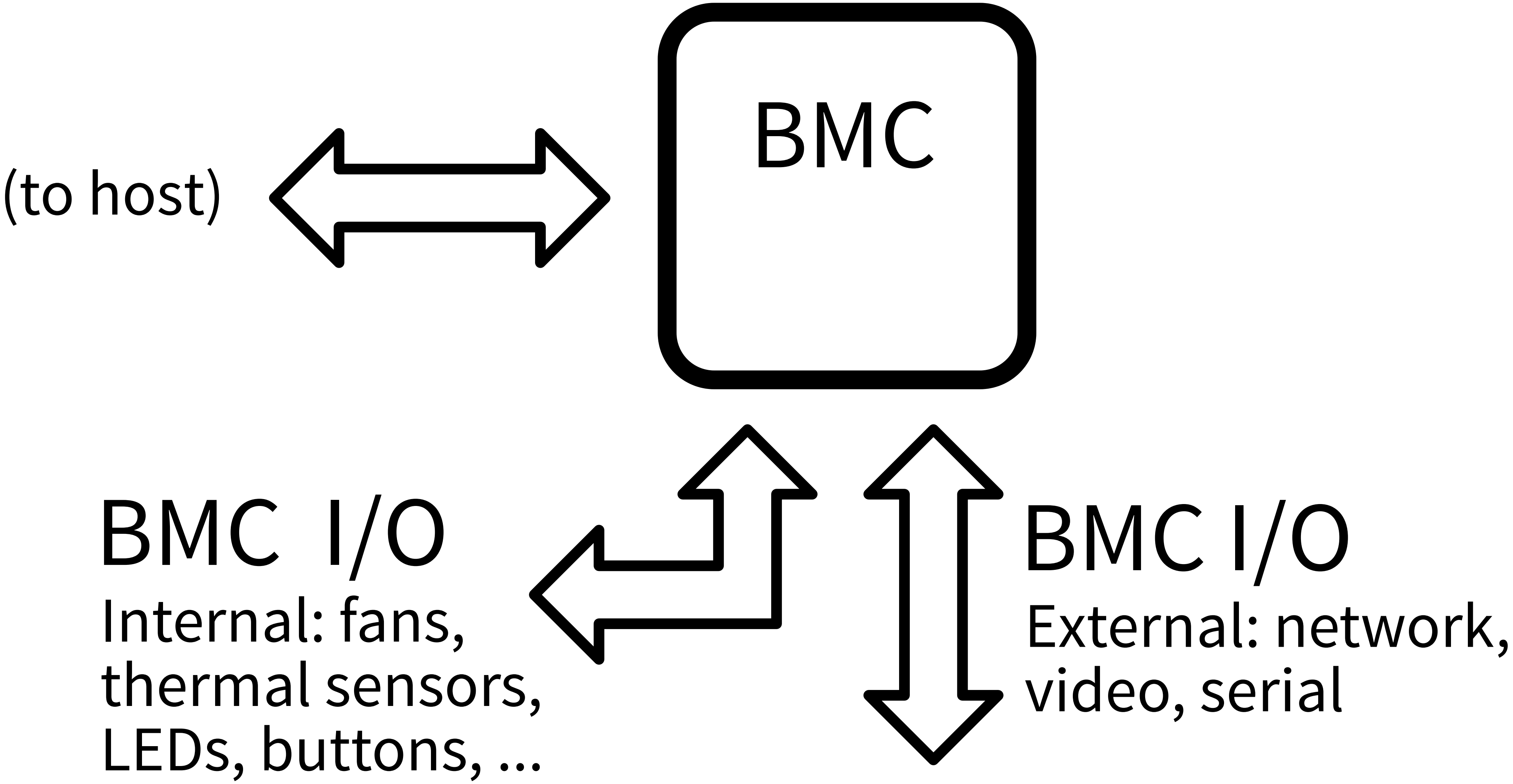


Split trust domain

⚠️

be careful with hardware boundaries


(to host)
BMC
BMC I/O
(?)



(to host)

BMC

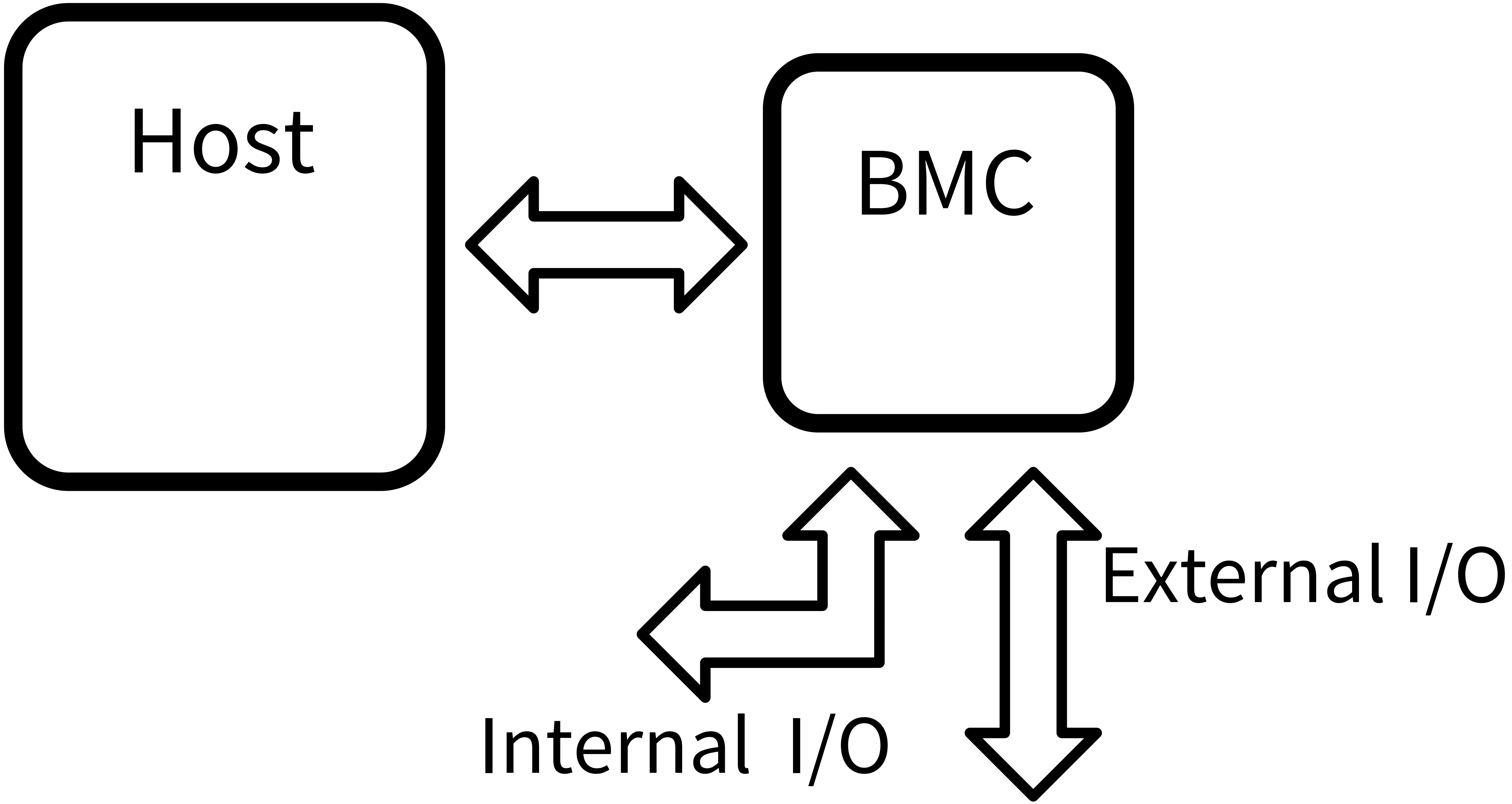
BMC I/O

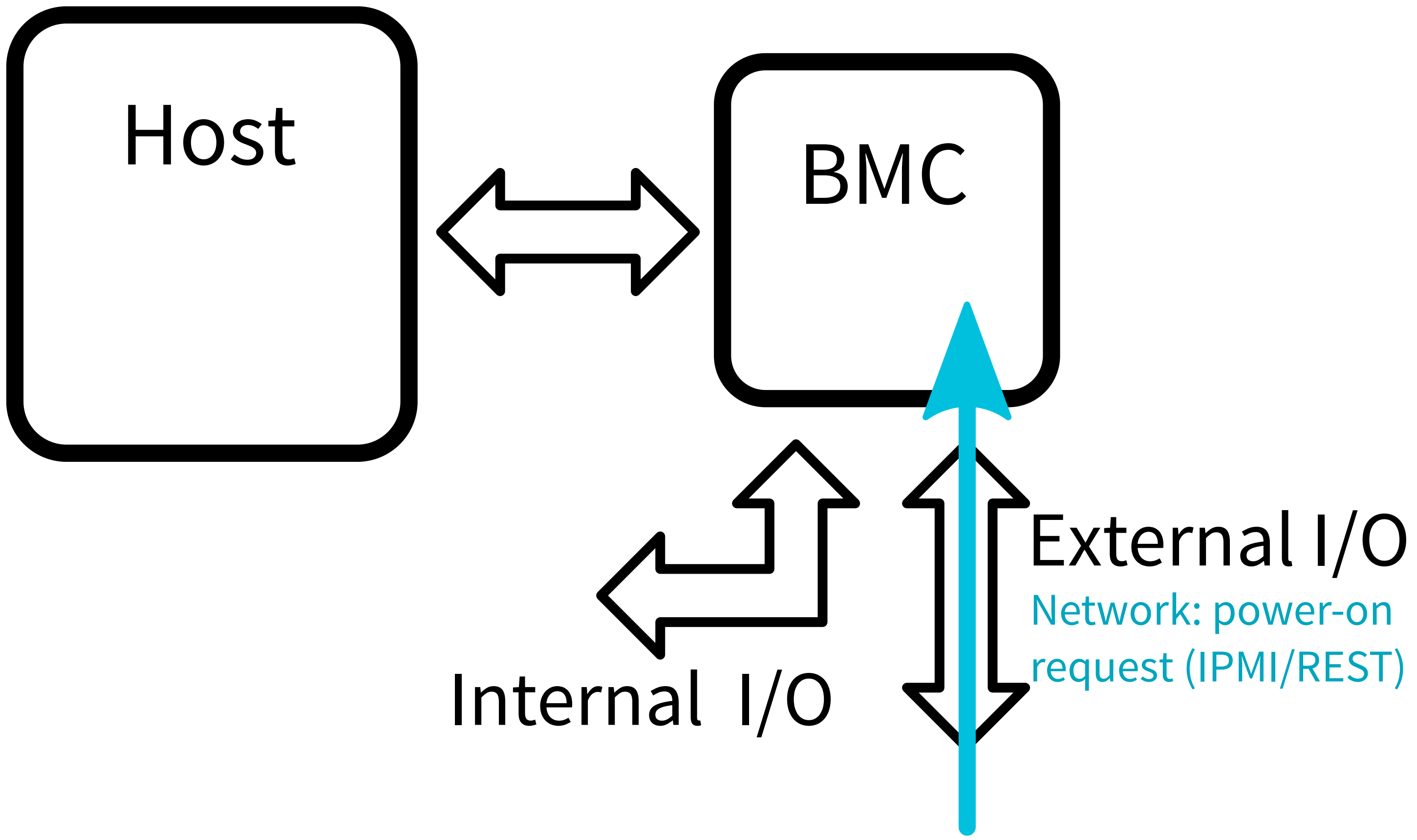
Internal: fans, thermal sensors, LEDs, buttons, ...

BMC I/O

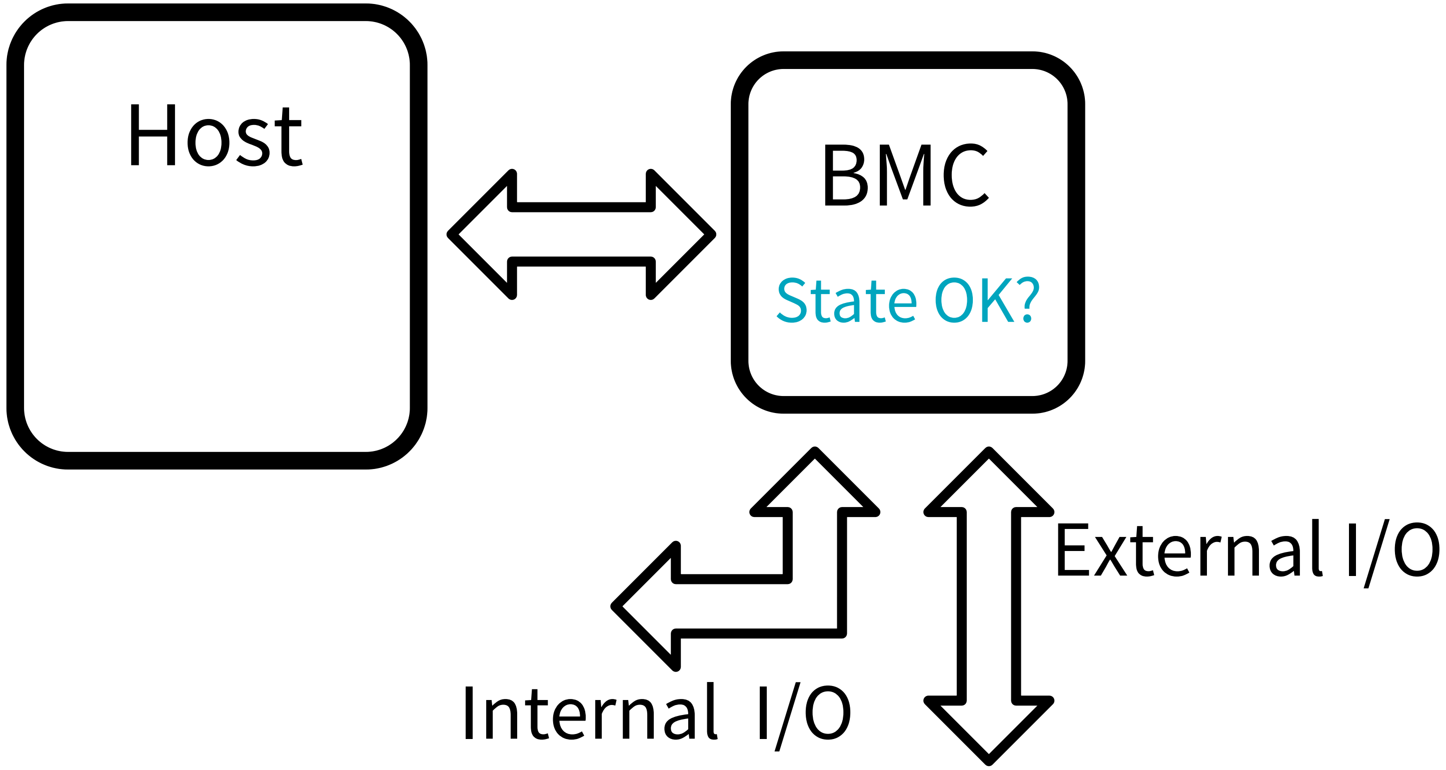
External: network, video, serial

Example: x86 power control

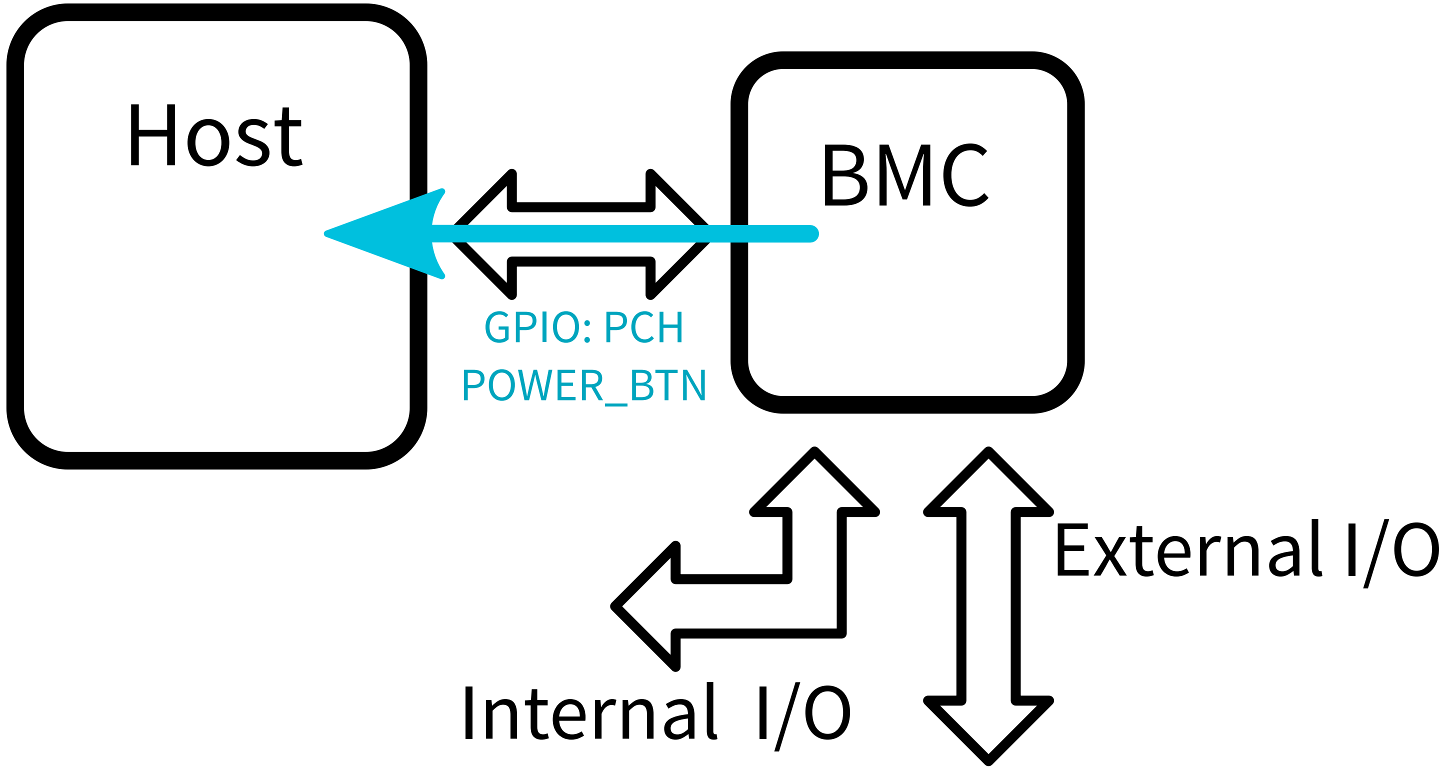

Host
BMC
External I/O
Internal I/O



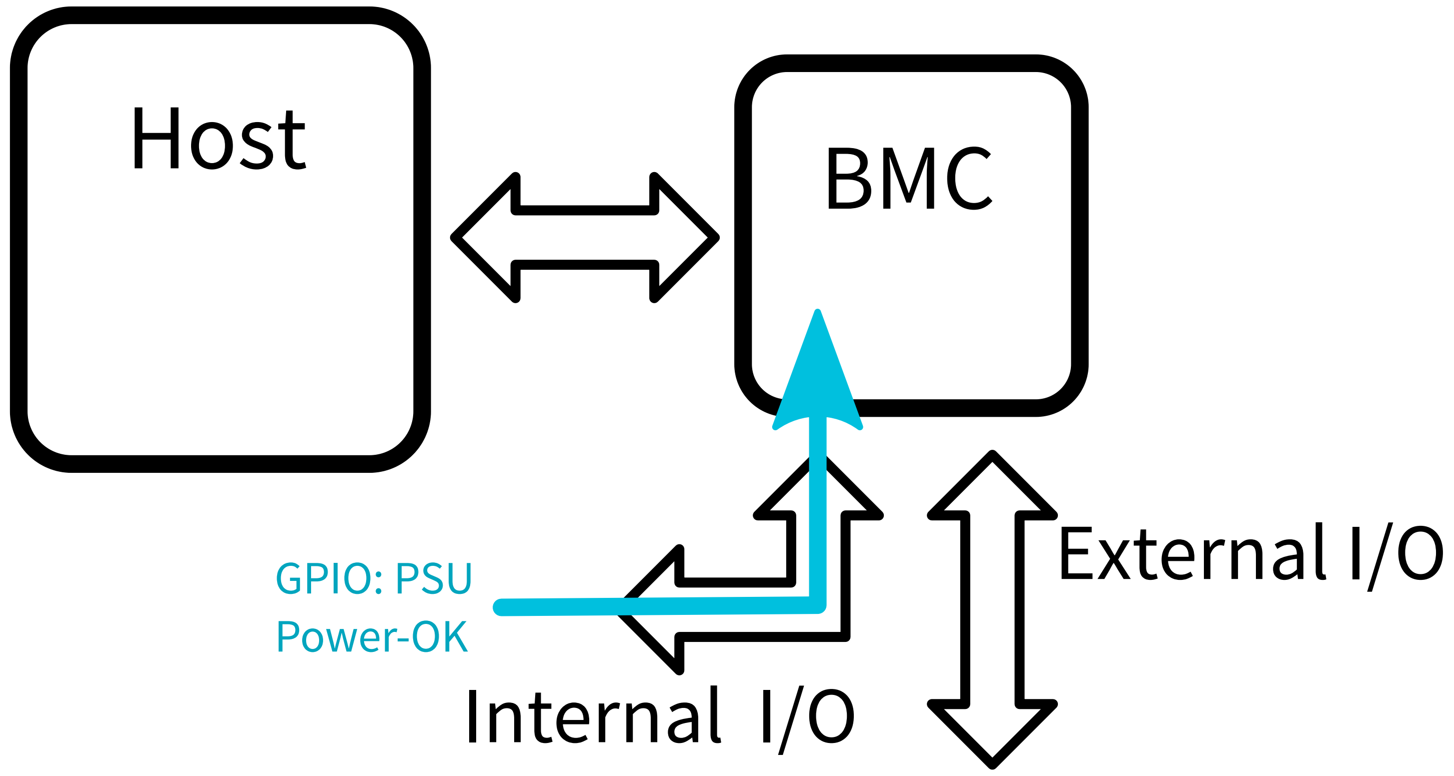
1. BMC receives a request to power-on the host over IPMI/REST/etc



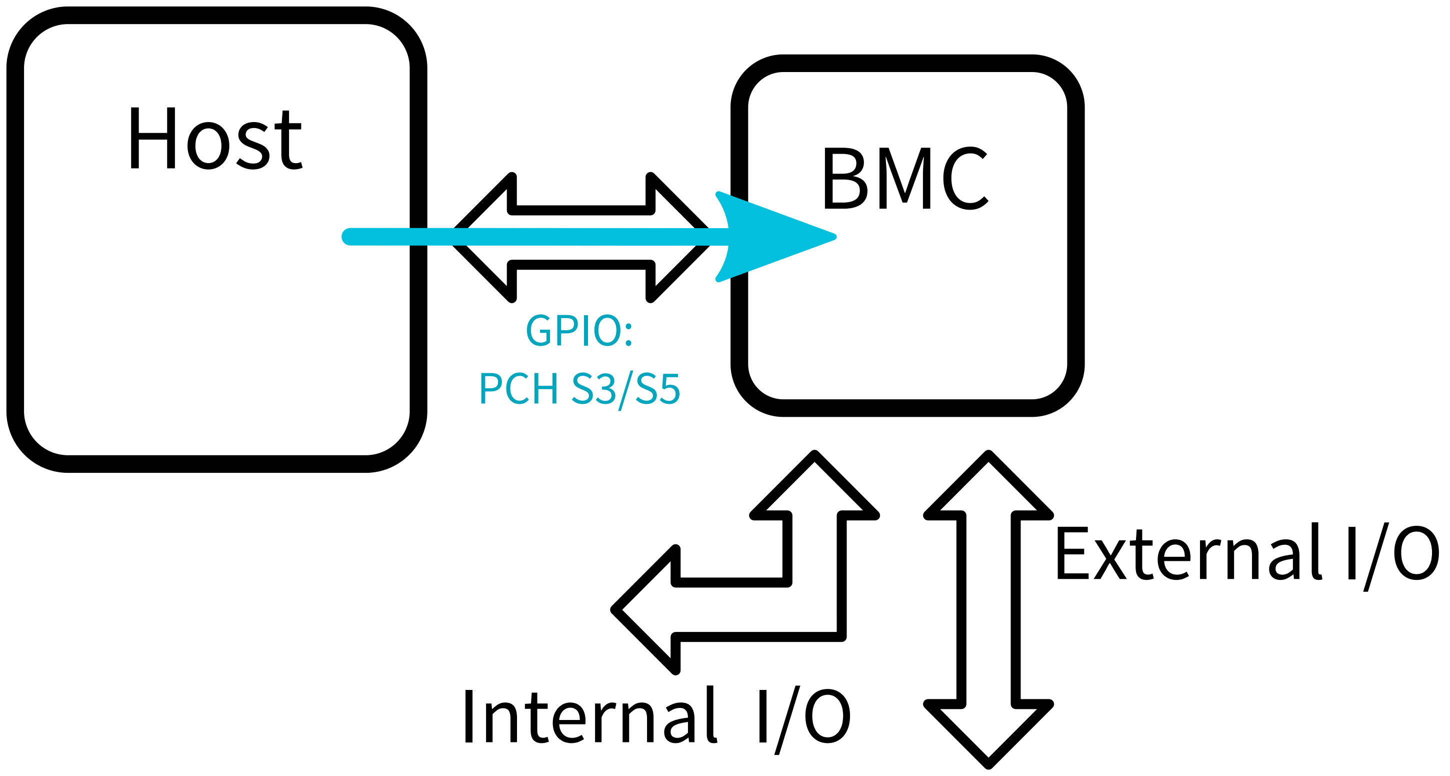
1. BMC receives a request to power-on the host over IPMI/REST/etc
2. BMC-internal state verification



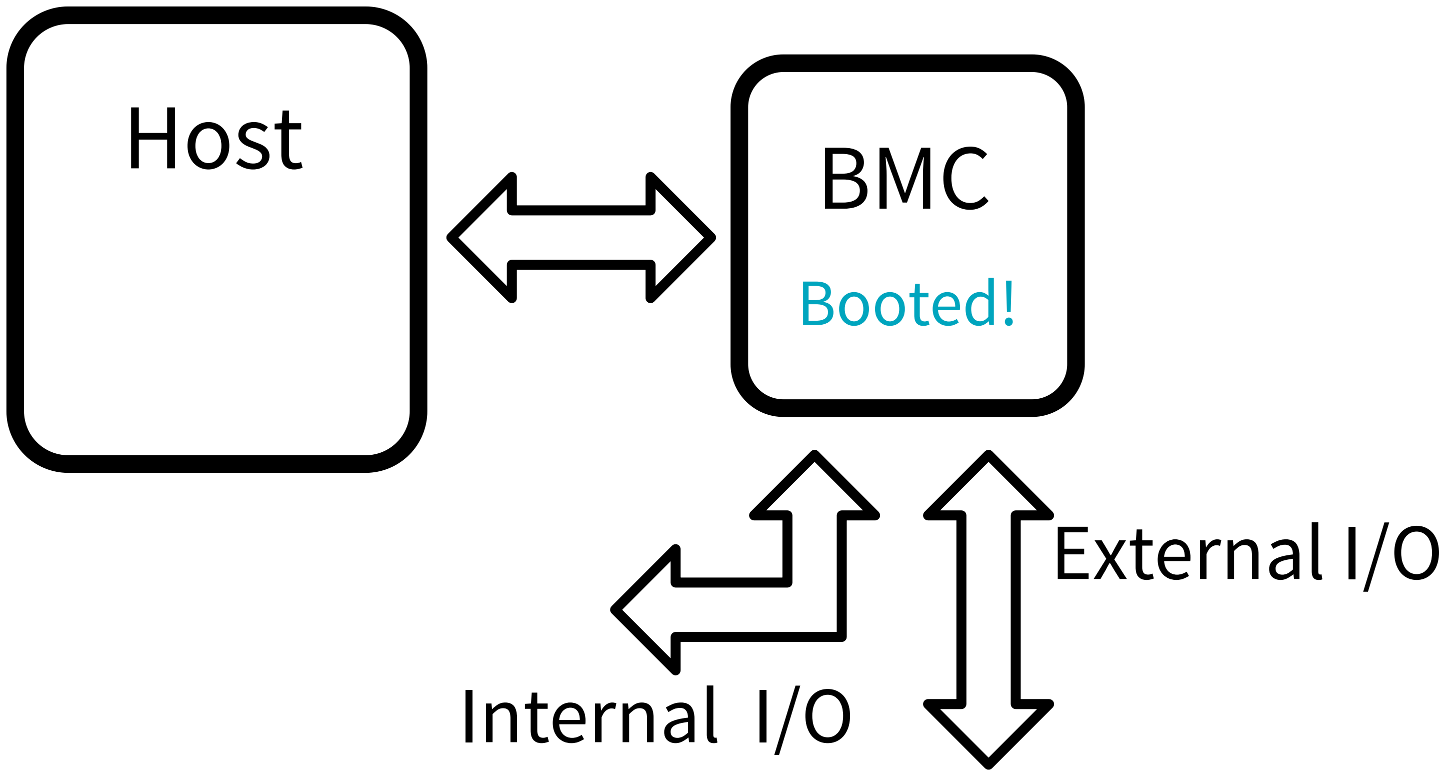
1. BMC receives a request to power-on the host over IPMI/REST/etc
2. BMC-internal state verification
3. BMC deasserts power button GPIO



1. BMC receives a request to power-on the host over IPMI/REST/etc
2. BMC-internal state verification
3. BMC deasserts power button GPIO
4. BMC checks power-OK signal



1. BMC receives a request to power-on the host over IPMI/REST/etc
2. BMC-internal state verification
3. BMC deasserts power button GPIO
4. BMC checks power-OK signal
5. BMC checks for S3/S5 exit



1. BMC receives a request to power-on the host over IPMI/REST/etc
2. BMC-internal state verification
3. BMC deasserts power button GPIO
4. BMC checks power-OK signal
5. BMC checks for S3/S5 exit

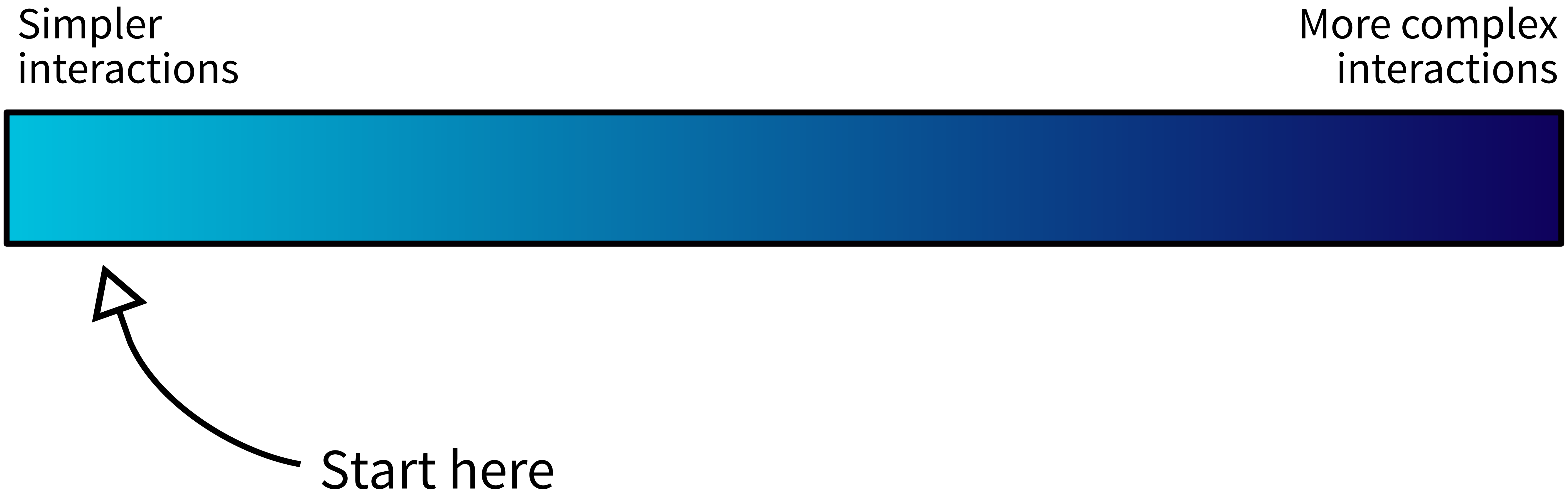
no magic required

(almost)

OpenBMC

github/openbmc

get experimentin'!



Simpler interactions

More complex interactions

Start here

Resources

- github/openbmc: OpenBMC platform
- buildroot: embedded system userspace
- Bus Pirate: i2c/SPI/UART/anything interface