EECS 388: Embedded Systems

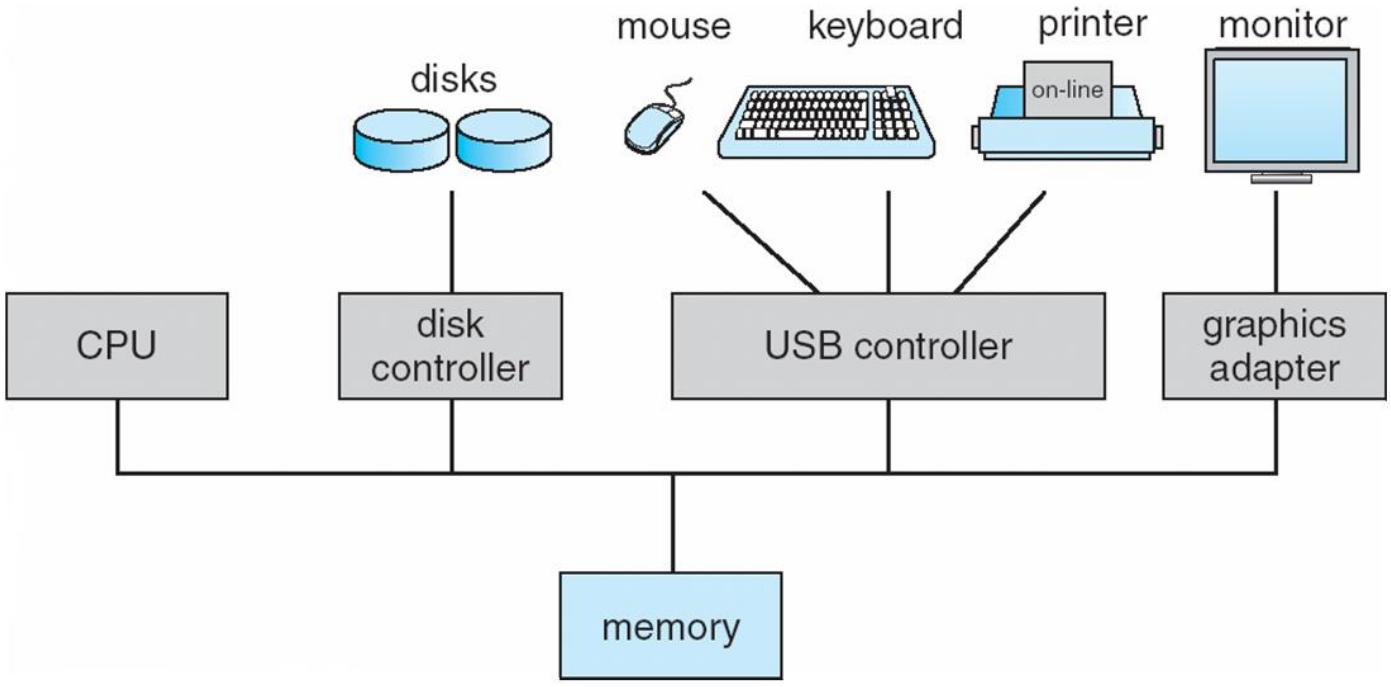
4. I/O Interface

Heechul Yun

Agenda

- Memory mapped I/O
- I/O interfaces

Input/output (I/O) Devices



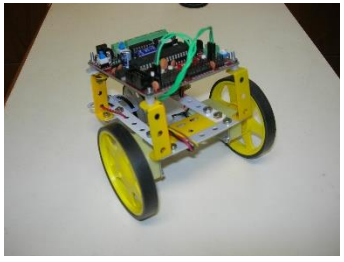
Input/output (I/O) Devices

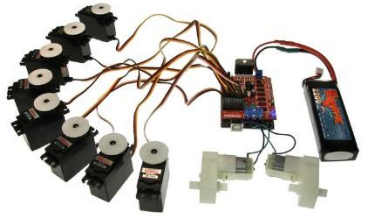




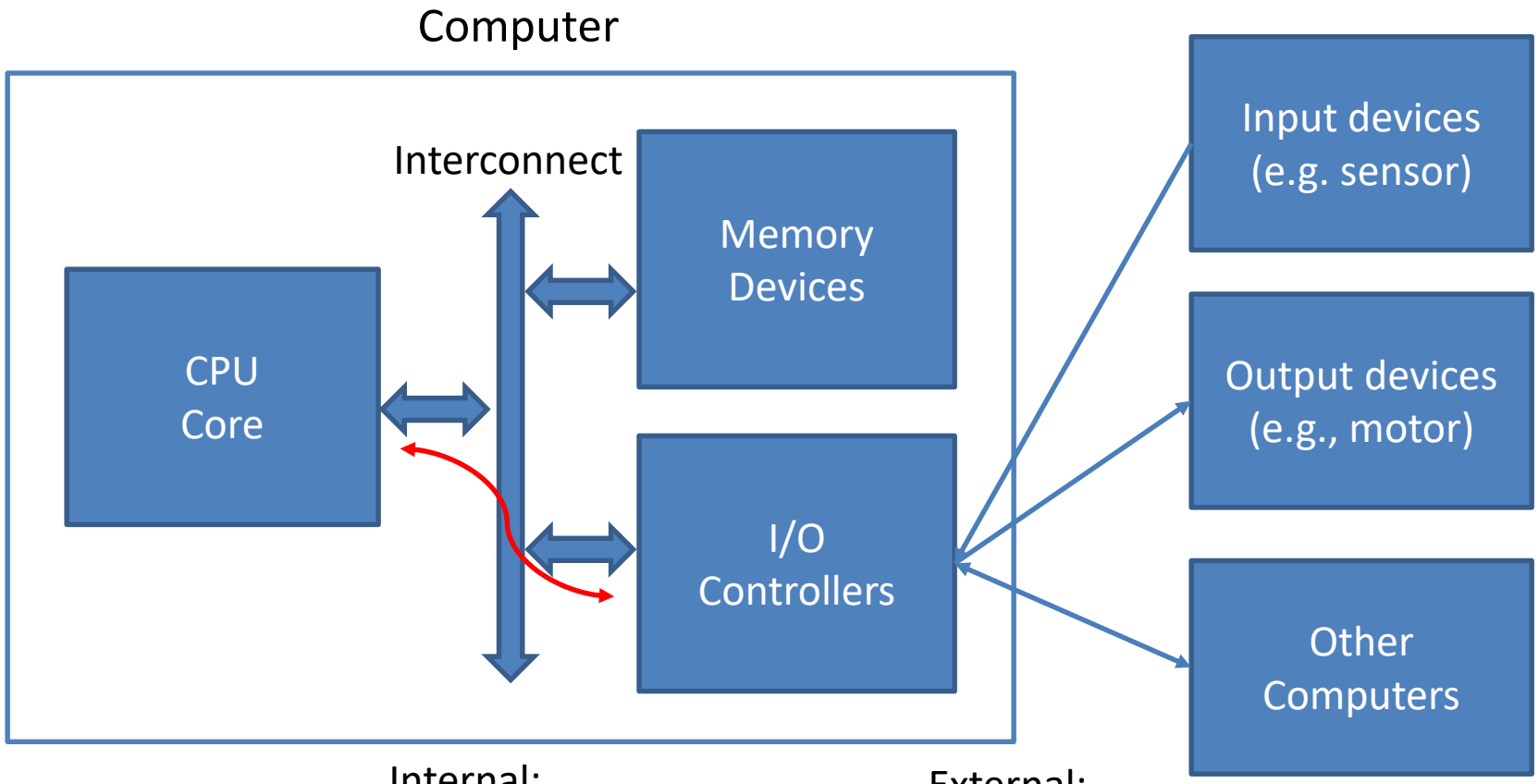






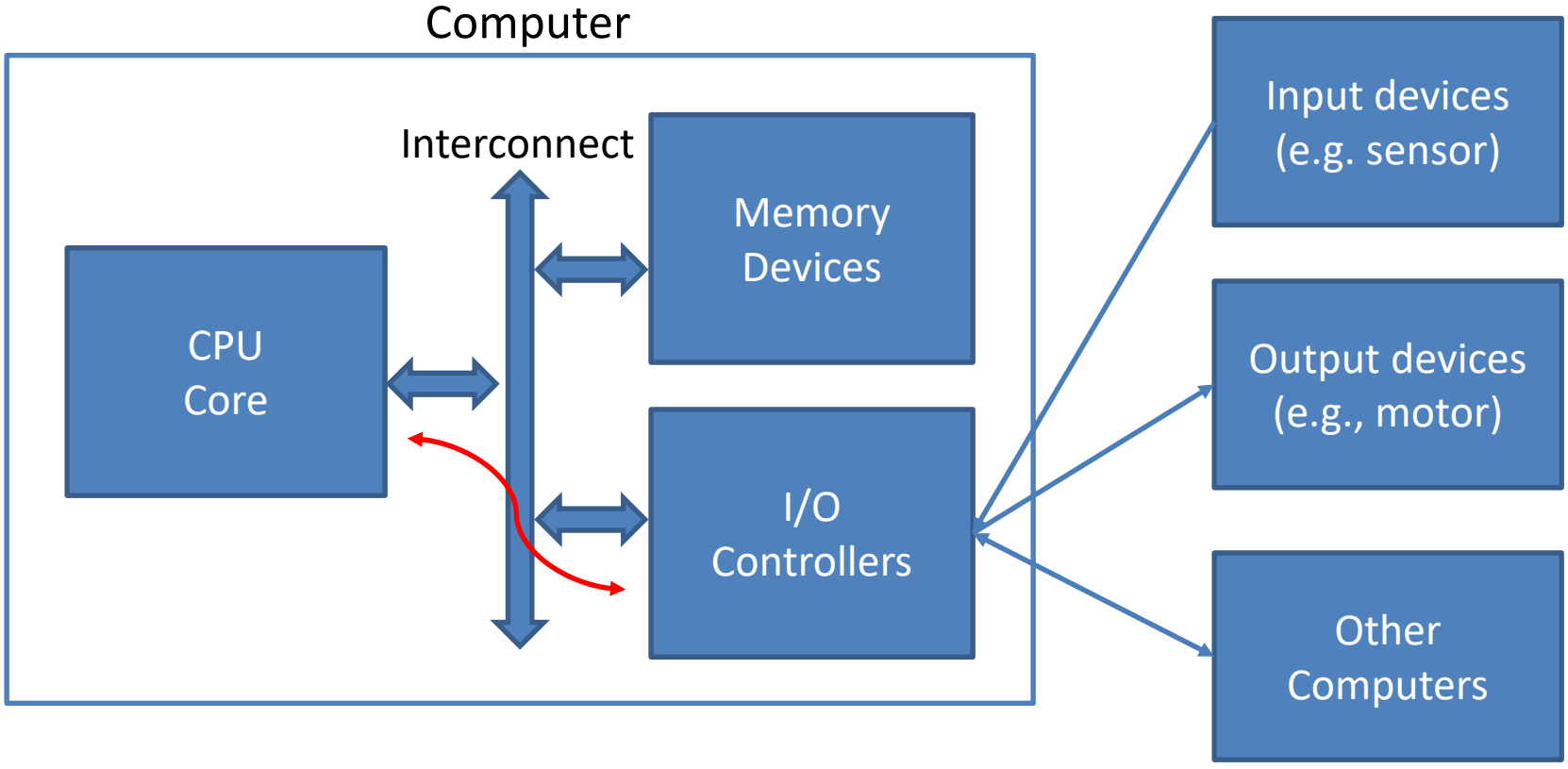


Input and Output (I/O) Interfaces



How Does CPU Talk to Devices?

- CPU talks to device controllers
  - Via **memory mapped I/O**

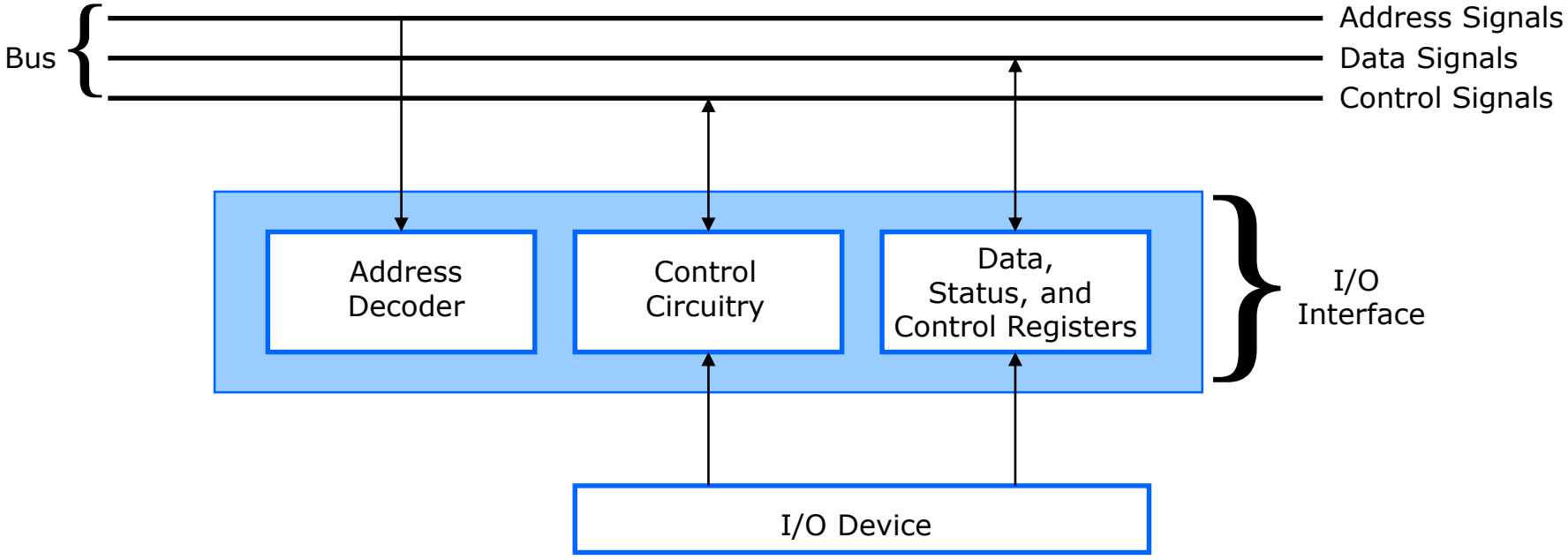


Memory Mapped I/O

- Parts of physical memory space are mapped to hardware controllers (peripherals)
  - Mapped to control registers and buffers
- Reading/writing from/to the memory mapped regions in peripheral specific ways

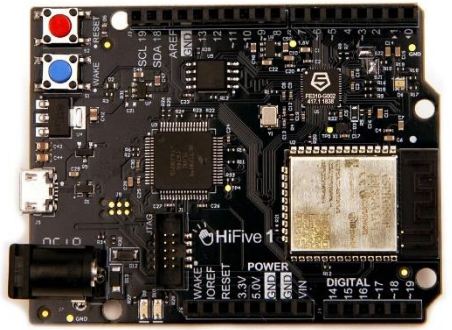
Memory Mapped I/O

- Via system bus (CPU – memory or I/O device)



Memory Map of SiFive FE310

Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	Mask ROM (8 KiB)	
0x0001_2000	0x0001_FFFF		Reserved	
0x0002_0000	0x0002_1FFF	R XC	OTP Memory Region	
0x0002_2000	0x001F_FFFF		Reserved	
0x0200_0000	0x0200_FFFF	RW A	CLINT	On-Chip Peripherals
0x0201_0000	0x07FF_FFFF		Reserved	
0x0800_0000	0x0800_1FFF	RWX A	E31 ITIM (8 KiB)	
0x0800_2000	0x0BFF_FFFF		Reserved	
0x0C00_0000	0x0FFF_FFFF	RW A	PLIC	
0x1000_0000	0x1000_0FFF	RW A	AON	
0x1000_1000	0x1000_7FFF		Reserved	
0x1000_8000	0x1000_8FFF	RW A	PRCI	
0x1000_9000	0x1000_FFFF		Reserved	
0x1001_0000	0x1001_0FFF	RW A	OTP Control	
0x1001_1000	0x1001_1FFF		Reserved	
0x1001_2000	0x1001_2FFF	RW A	GPIO	
0x1001_3000	0x1001_3FFF	RW A	UART 0	
0x1001_4000	0x1001_4FFF	RW A	QSPI 0	
0x1001_5000	0x1001_5FFF	RW A	PWM 0	
0x1001_6000	0x1001_6FFF	RW A	I2C 0	
0x1001_7000	0x1002_2FFF		Reserved	
0x1002_3000	0x1002_3FFF	RW A	UART 1	
0x1002_4000	0x1002_4FFF	RW A	SPI 1	
0x1002_5000	0x1002_5FFF	RW A	PWM 1	
0x1002_6000	0x1003_3FFF		Reserved	
0x1003_4000	0x1003_4FFF	RW A	SPI 2	
0x1003_5000	0x1003_5FFF	RW A	PWM 2	
0x1003_6000	0x1FFF_FFFF		Reserved	
0x2000_0000	0x3FFF_FFFF	R XC	QSPI 0 Flash (512 MiB)	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF		Reserved	
0x8000_0000	0x8000_3FFF	RWX A	E31 DTIM (16 KiB)	On-Chip Volatile Memory
0x8000_4000	0xFFFF_FFFF		Reserved	



CPU: 32 bit RISC-V
Clock: 320 MHz
SRAM: 16 KB (D)
Flash: 4MB

Memory mapped I/O regions

Memory Map of SiFive FE310

Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	Mask ROM (8 KiB)	
0x0001_2000	0x0001_FFFF		Reserved	
0x0002_0000	0x0002_1FFF	R XC	OTP Memory Region	
0x0002_2000	0x001F_FFFF		Reserved	
0x0200_0000	0x0200_FFFF	RW A	CLINT	On-Chip Peripherals
0x0201_0000	0x07FF_FFFF		Reserved	
0x0800_0000	0x0800_1FFF	RWX A	E31 ITIM (8 KiB)	
0x0800_2000	0x0BFF_FFFF		Reserved	
0x0C00_0000	0x0FFF_FFFF	RW A	PLIC	
0x1000_0000	0x1000_0FFF	RW A	AON	
0x1000_1000	0x1000_7FFF		Reserved	
0x1000_8000	0x1000_8FFF	RW A	PRCI	
0x1000_9000	0x1000_FFFF		Reserved	
0x1001_0000	0x1001_0FFF	RW A	OTP Control	
0x1001_1000	0x1001_1FFF		Reserved	
0x1001_2000	0x1001_2FFF	RW A	GPIO	
0x1001_3000	0x1001_3FFF	RW A	UART 0	
0x1001_4000	0x1001_4FFF	RW A	QSPI 0	
0x1001_5000	0x1001_5FFF	RW A	PWM 0	
0x1001_6000	0x1001_6FFF	RW A	I2C 0	
0x1001_7000	0x1002_2FFF		Reserved	
0x1002_3000	0x1002_3FFF	RW A	UART 1	
0x1002_4000	0x1002_4FFF	RW A	SPI 1	
0x1002_5000	0x1002_5FFF	RW A	PWM 1	
0x1002_6000	0x1003_3FFF		Reserved	
0x1003_4000	0x1003_4FFF	RW A	SPI 2	
0x1003_5000	0x1003_5FFF	RW A	PWM 2	
0x1003_6000	0x1FFF_FFFF		Reserved	
0x2000_0000	0x3FFF_FFFF	R XC	QSPI 0 Flash (512 MiB)	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF		Reserved	
0x8000_0000	0x8000_3FFF	RWX A	E31 DTIM (16 KiB)	On-Chip Volatile Memory
0x8000_4000	0xFFFF_FFFF		Reserved	

GPIO registers are mapped at 0x10012000 – 0x10012FFF

Offset	Name	Description
0x00	input_val	Pin value
0x04	input_en	Pin input enable*
0x08	output_en	Pin output enable*
0x0C	output_val	Output value
0x10	pue	Internal pull-up enable*
0x14	ds	Pin drive strength
0x18	rise_ie	Rise interrupt enable
0x1C	rise_ip	Rise interrupt pending
0x20	fall_ie	Fall interrupt enable
0x24	fall_ip	Fall interrupt pending
0x28	high_ie	High interrupt enable
0x2C	high_ip	High interrupt pending
0x30	low_ie	Low interrupt enable
0x34	low_ip	Low interrupt pending
0x40	out_xor	Output XOR (invert)

Memory Mapped I/O: Example

```
/*****************************************************************************
 *    memory map
 *****************************************************************************/
#define GPIO_CTRL_ADDR      0x10012000  // GPIO controller base address
#define GPIO_INPUT_VAL      0x00        // input val
#define GPIO_INPUT_EN       0x04        // input enable
#define GPIO_OUTPUT_EN      0x08        // output enable
#define GPIO_OUTPUT_VAL     0x0C        // output_val
#define GPIO_OUTPUT_XOR     0x40        // output XOR (invert)
```

```
void gpio_write(int gpio, int state)
{
 uint32_t val = *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_VAL);
 if (state == ON)
   val |= (1<<gpio);
 else
   val &= (~(1<<gpio));
 *(volatile uint32_t *) (GPIO_CTRL_ADDR + GPIO_OUTPUT_VAL) = val;
 return;
}
```

Memory Map of SiFive FE310

Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	Mask ROM (8 KiB)	
0x0001_2000	0x0001_FFFF		Reserved	
0x0002_0000	0x0002_1FFF	R XC	OTP Memory Region	
0x0002_2000	0x001F_FFFF		Reserved	
0x0200_0000	0x0200_FFFF	RW A	CLINT	On-Chip Peripherals
0x0201_0000	0x07FF_FFFF		Reserved	
0x0800_0000	0x0800_1FFF	RWX A	E31 ITIM (8 KiB)	
0x0800_2000	0x0BFF_FFFF		Reserved	
0x0C00_0000	0x0FFF_FFFF	RW A	PLIC	
0x1000_0000	0x1000_0FFF	RW A	AON	
0x1000_1000	0x1000_7FFF		Reserved	
0x1000_8000	0x1000_8FFF	RW A	PRCI	
0x1000_9000	0x1000_FFFF		Reserved	
0x1001_0000	0x1001_0FFF	RW A	OTP Control	
0x1001_1000	0x1001_1FFF		Reserved	
0x1001_2000	0x1001_2FFF	RW A	GPIO	
0x1001_3000	0x1001_3FFF	RW A	UART 0	
0x1001_4000	0x1001_4FFF	RW A	QSPI 0	
0x1001_5000	0x1001_5FFF	RW A	PWM 0	
0x1001_6000	0x1001_6FFF	RW A	I2C 0	
0x1001_7000	0x1002_2FFF		Reserved	
0x1002_3000	0x1002_3FFF	RW A	UART 1	
0x1002_4000	0x1002_4FFF	RW A	SPI 1	
0x1002_5000	0x1002_5FFF	RW A	PWM 1	
0x1002_6000	0x1003_3FFF		Reserved	
0x1003_4000	0x1003_4FFF	RW A	SPI 2	
0x1003_5000	0x1003_5FFF	RW A	PWM 2	
0x1003_6000	0x1FFF_FFFF		Reserved	
0x2000_0000	0x3FFF_FFFF	R XC	QSPI 0 Flash (512 MiB)	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF		Reserved	
0x8000_0000	0x8000_3FFF	RWX A	E31 DTIM (16 KiB)	On-Chip Volatile Memory
0x8000_4000	0xFFFF_FFFF		Reserved	

UART0 registers are mapped at 0x10013000 – 0x10013FFF

Offset	Name	Description
0x00	txdata	Transmit data register
0x04	rxdata	Receive data register
0x08	txctrl	Transmit control register
0x0C	rxctrl	Receive control register
0x10	ie	UART interrupt enable
0x14	ip	UART interrupt pending
0x18	div	Baud rate divisor

Volatile in C

```
void ser_write(char c)
{
  uint32_t regval;
  /* busy-wait if tx FIFO is full  */
  do {
    regval = *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA);
  } while (regval & 0x80000000);

  /* write the character */
  *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA) = c;
}
```

```
ser_write:
        li      a4,268513280
.L17:
        lw      a5,0(a4)
        bltz    a5,.L17
        sw      a0,0(a4)
        ret
```

a4 = 0x10013000

a5 = *a4

Branch to .L17 if a5 < zero

Volatile in C

```
void ser_write(char c)
{
  uint32_t regval;
  /* busy-wait if tx FIFO is full  */
  do {
    regval = *(uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA);
  } while (regval & 0x80000000);

  /* write the character */
  *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA) = c;
}
```

```
ser_write:
        li      a5,268513280
        lw      a5,0(a5)
.L17:
        bltz    a5,.L17
        li      a5,268513280
        sw      a0,0(a5)
```

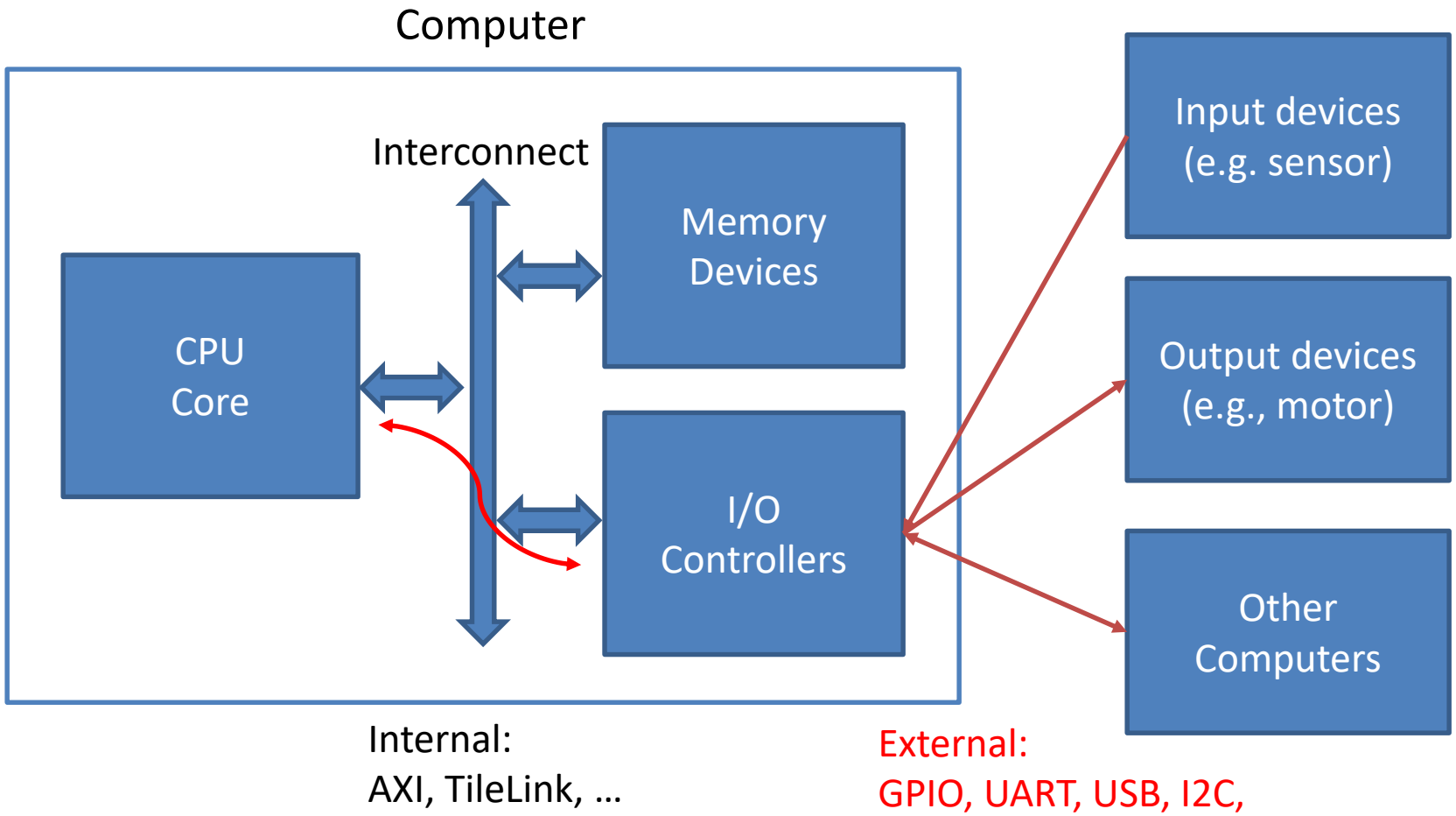
a4 = 0x10013000

a5 = *a4

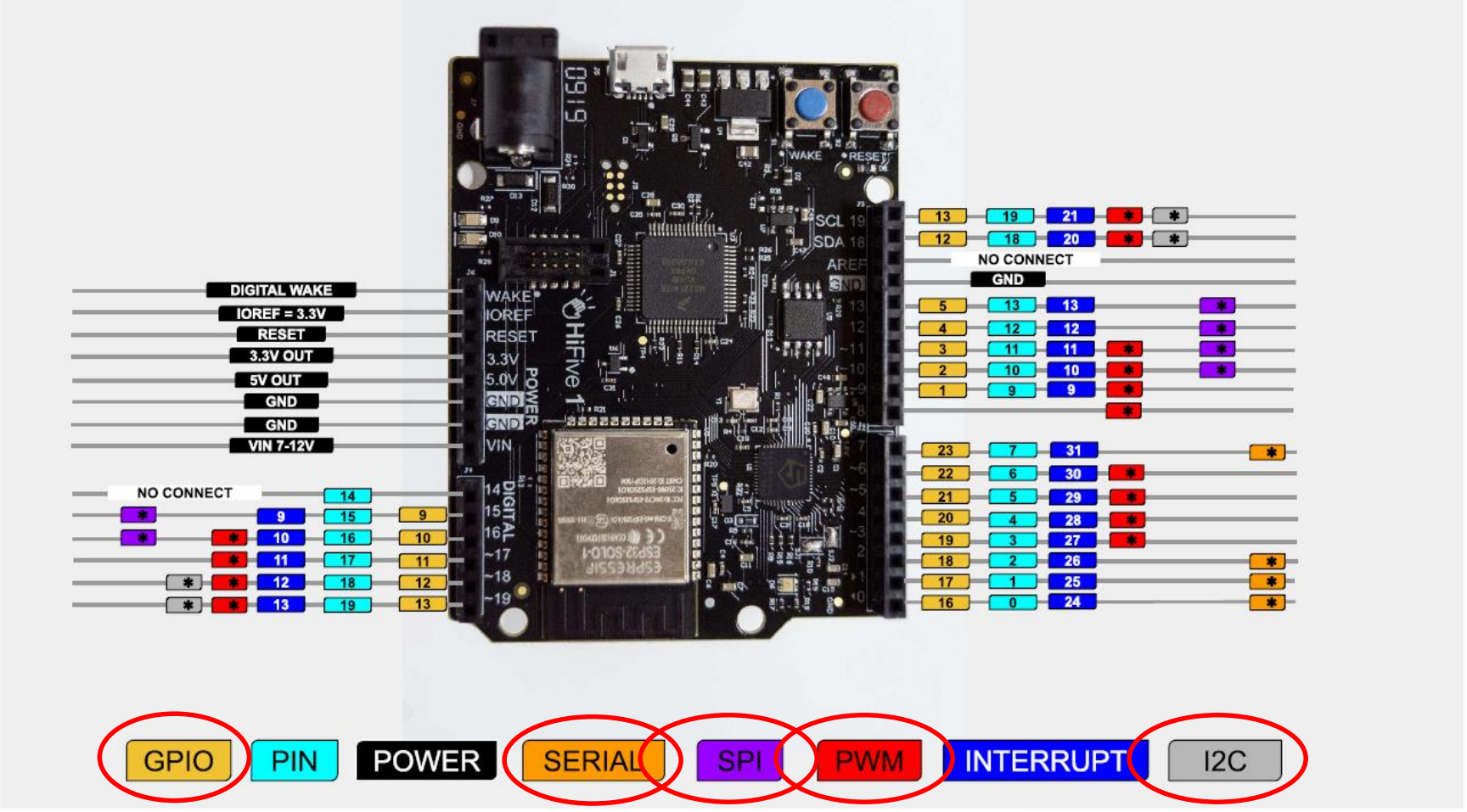
Branch to .L17 if a5 < zero

Is this correct?

Input and Output (I/O) Interfaces



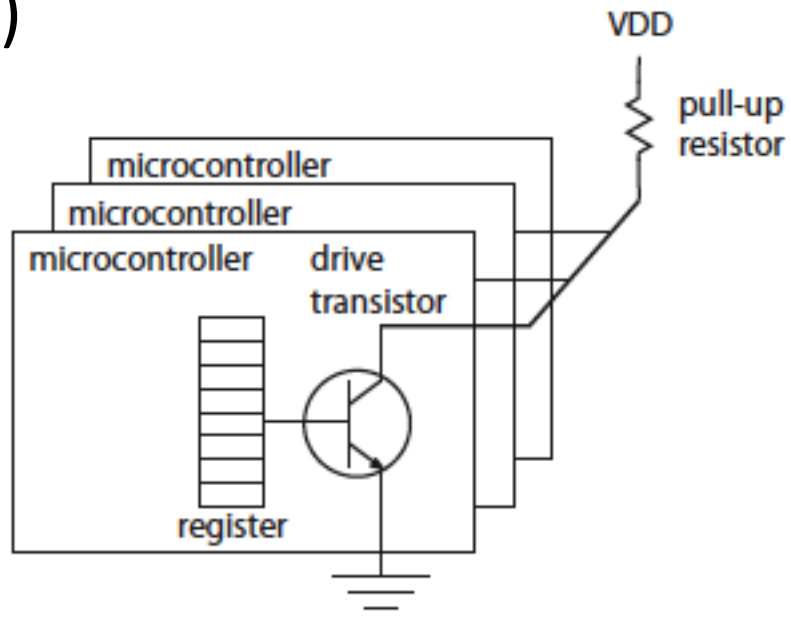
External I/O Interfaces



HiFive1 Rev B Pinout

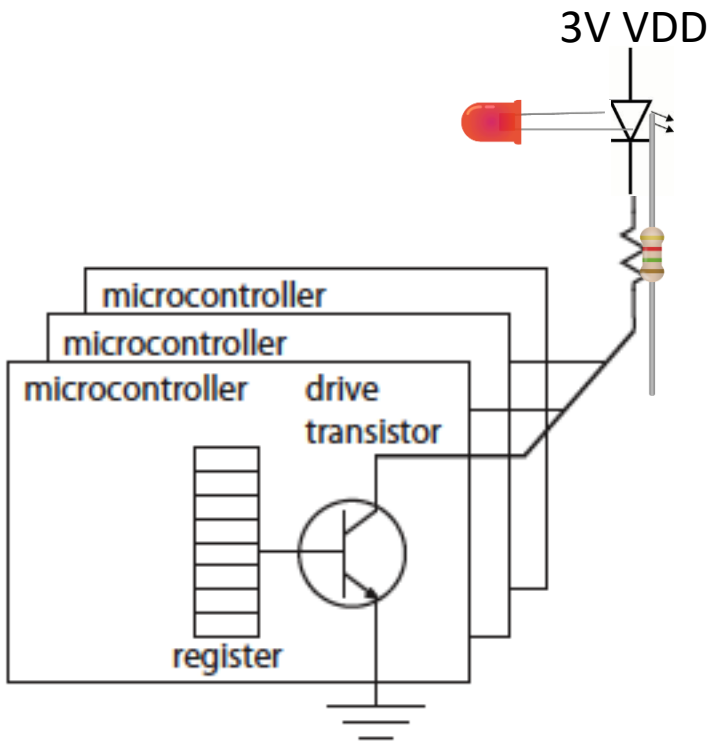
General Purpose I/O (GPIO)

- Programmable digital input/output pins
- Use voltage levels to represent digital signals
  - 5V = logic 1 (for 5V CPUs)
  - 0V = logic 0
- Can be configured as
  - Input or output
- Useful to interact with external devices



Example: Turn on an LED

- Assume a GPIO pin can draw up to 18mA, and when the LED is on, it has a voltage drop of 2V
- Ohm's law: $I \times R = V$
- What registor is needed?



Example: Turn on an LED

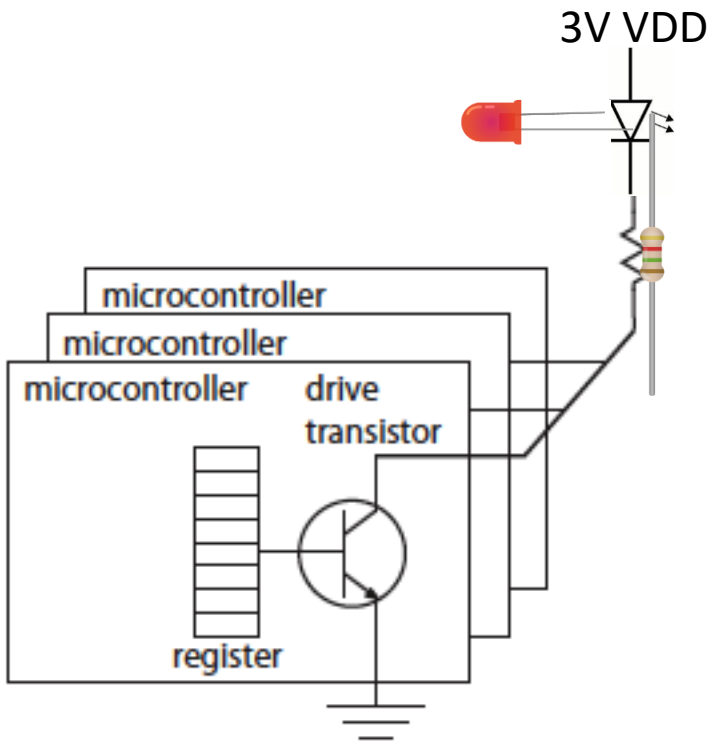
- Assume a GPIO pin can draw up to 18mA, and when the LED is on, it has a voltage drop of 2V
- Ohm's law: $I \times R = V$
- What registor is needed?

  I = V / R = 1 / R < 18mA

  R > 1V/18mA

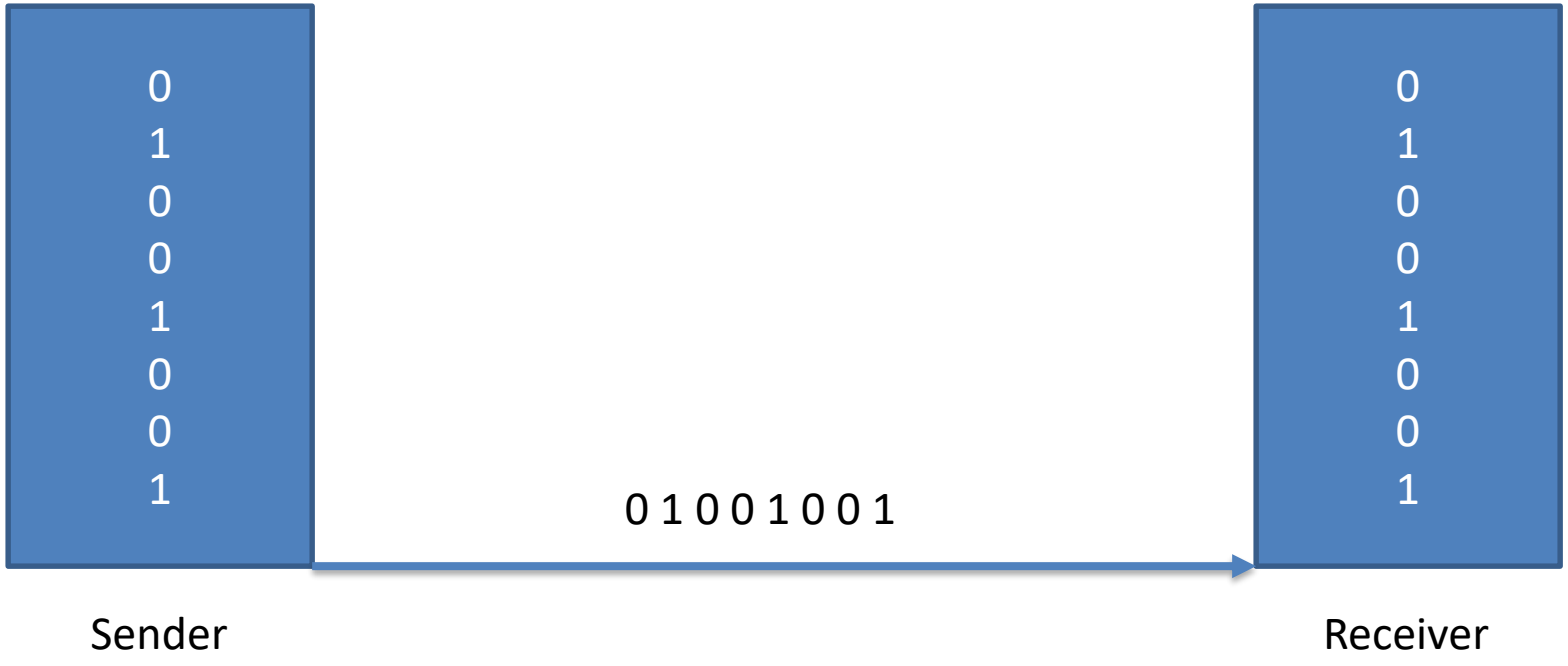
  = 1000mV/18mA

  = 56 ohm



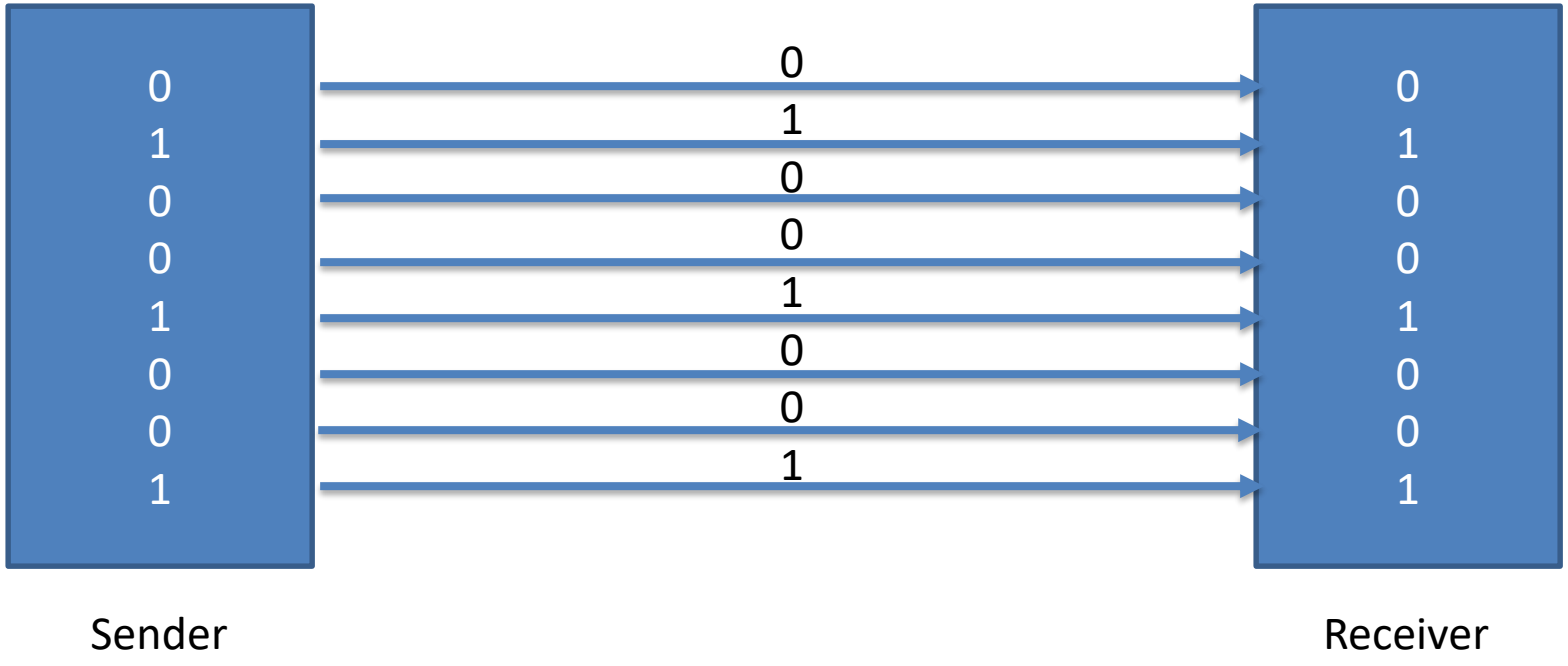
Serial vs. Parallel

- Serial communication
  - use single line



Serial vs. Parallel

- Parallel communication
  - use multiple lines



Serial vs. Parallel Interfaces

- Serial interfaces
  - RS-232: serial communication standard
  - USB: universal serial bus
  - I^2C: inter-integrated circuit
  - SPI: serial peripheral interface bus
  - SATA: serial ATA
  - …

I^2C

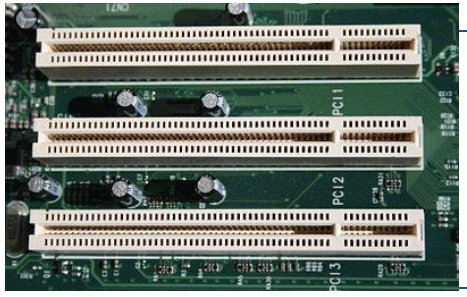
SATA

USB

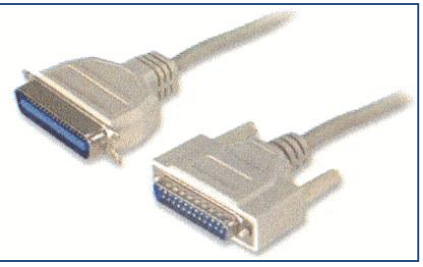

RS-232

Serial vs. Parallel Interfaces

- Parallel interfaces
  - Parallel ATA: advanced technology attachment
  - SCSI: small computer system interface
  - PCI: peripheral component interface
  - …



PCI



SCSI

PATA

Parallel port

Serial vs. Parallel Interfaces

- Are parallel interfaces better?
  - ++ better performance for the same clock speed
  - --- synchronization among the wires at high speed

- Serial interfaces are increasingly popular for external i/o (e.g., USB, SATA, PCIe, …)
- Parallel interfaces are dominant for internal interconnect within chip where clock synchronization is easier (e.g., AXI, TileLink, ..)

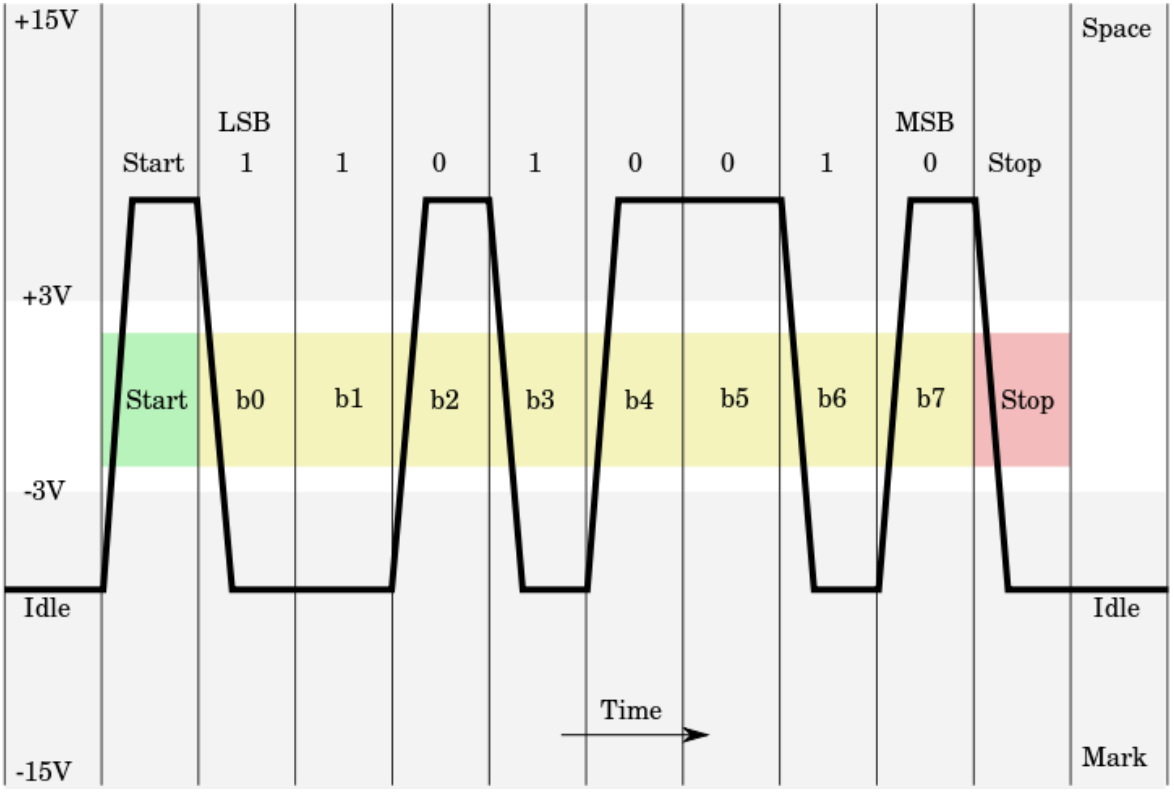
Synchronous vs. Asynchronous

- Synchronous communication
  - Requires a common shared clock
  - Higher throughput, low overhead.
  - All parallel communications are synchronous
  - Synchronization difficulties.

- Asynchronous communication
  - No shared clock.
  - Asynchronous start/stop
  - Self clocked w.r.t. agreed up on speed
  - High overhead

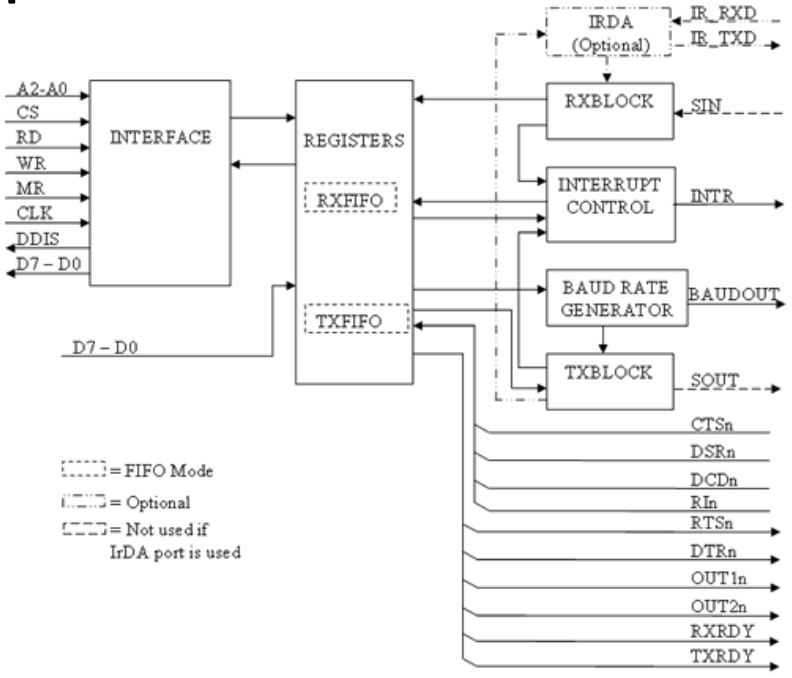
RS-232

- Asynchronous serial communication standard from 1960s, but still widely used



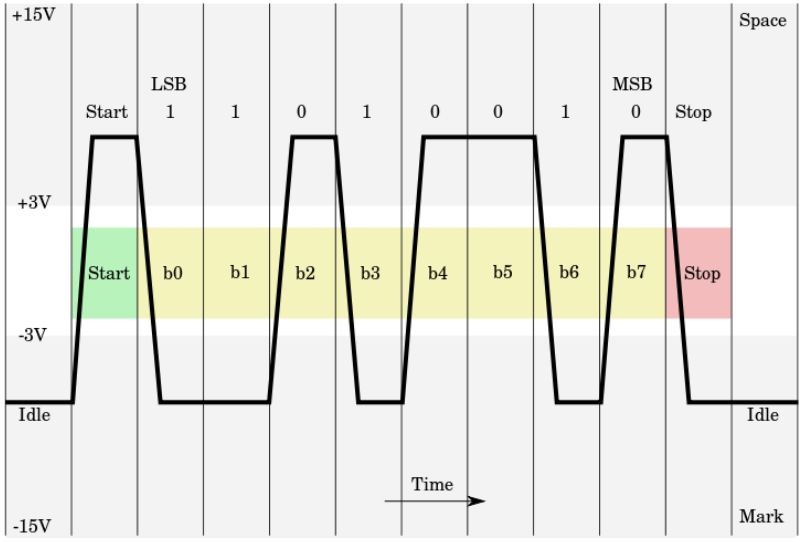
UART

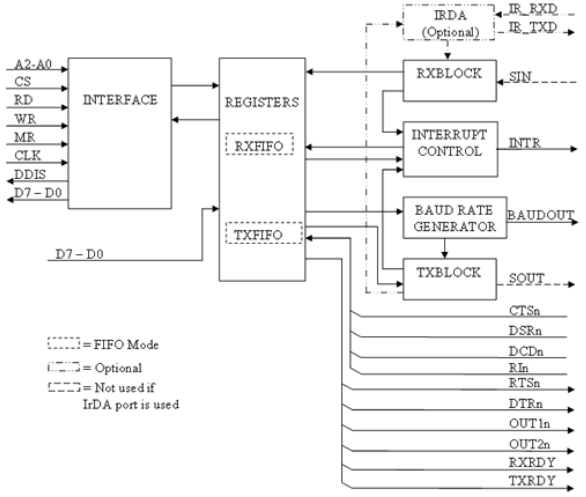
- Universal asynchronous receiver/transmitter
- Convert parallel data to serial data (vice versa)
- Can be configured to implement various serial protocol (e.g., RS-232)
- Require at least TX and RX lines to function



UART Speed (Baudrate)

- Both sender and receiver must use agreed upon transmission speed (baudrate)
- Up on detecting the start bit, the receiver sample 8 more times before stop



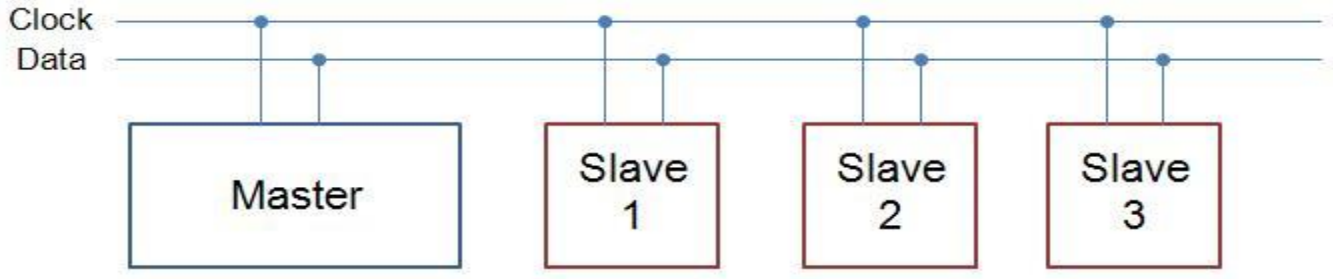


Point-to-Point vs. Bus

- Point-to-point
  - 1:1 communication
- Bus
  - Shared among multiple devices
  - Need an arbitration mechanism
  - Master
    - An entity who initiates the data transfer
  - Slave
    - An entity who cooperates with the master

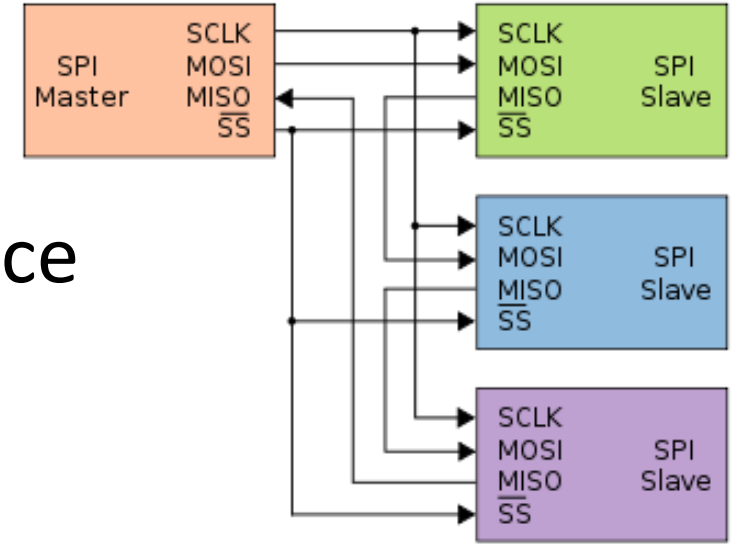
I^2C

- Inter-integrated circuit protocol (I^2C), by NXP
- Synchronous serial communication protocol
- Use two wires (SCL - clock, SDA - data)
- Multi-master, multi-slave support
- Use 7bit ID. Connect up to 127 devices.
- Half duplex, relatively slow



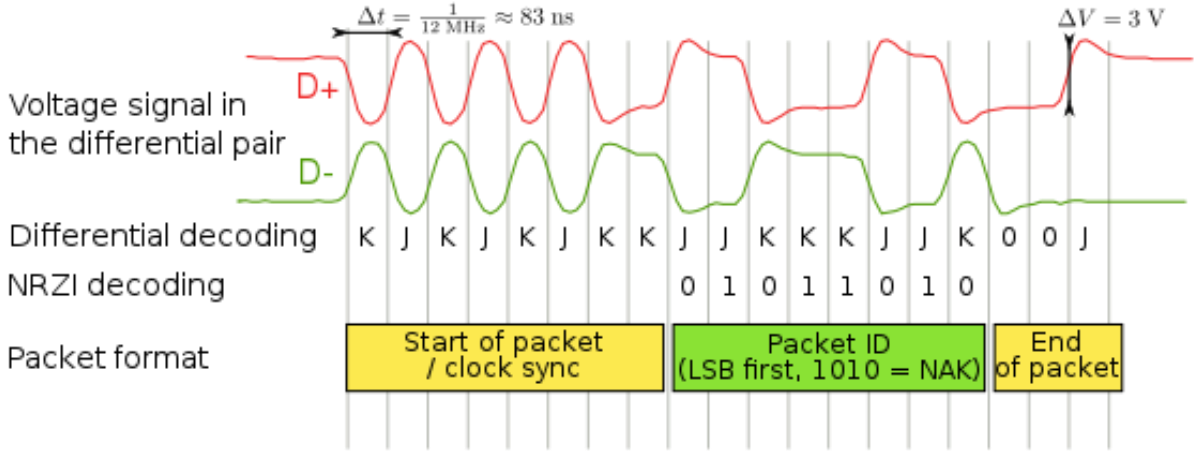
SPI

- Serial peripheral interface (SPI), by Motorola
- Synchronous serial communication protocol
- Use 4 lines, full-duplex, (relatively) fast
- Single master, multi-slave
- No start/stop bits
- Good for fast, short distance communication

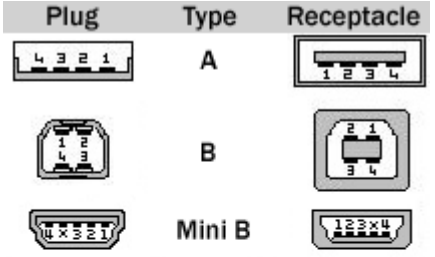


USB

- Universal serial bus
- One host, multiple devices, can form a tree
- Use two wires for single differential signal
  - Reduce noise caused by electromagnetic interference
- Device needs to process the protocol messages



https://commons.wikimedia.org/wiki/File:USB_signal_example.svg#/media/File:USB_signal_example.svg



Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

I/O Considerations

- Digital vs. analog
- Serial vs. parallel
- Wired vs. wireless
- Speed (throughput, latency)
- Real-time/QoS guarantees
- Power/Electrical requirements
- Reliability, Security

Summary

- Memory mapped I/O
  - Parts of physical memory space are mapped to hardware controllers (peripherals)
  - Reading/writing from/to the memory mapped regions in peripheral specific ways
- I/O interfaces
  - GPIO, UART, SPI, I2C, USB, …
  - Serial vs parallel
  - Asynchronous vs synchronous

Quiz

- Suppose you are sending data over a UART channel at a baud rate of 115200 bps. How long does it take to send a single 8 bit character over the channel?

Quiz

- How many *CPU cycles* would be needed to execute the busy-wait loop in the worst-case? Assume that that the CPU is running at 1 MHz and the baud rate is set at 9600 bps?

```
void ser_write(char c)
{
  uint32_t regval;
  /* busy-wait if tx FIFO is full  */
  do {
    regval = *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA);
  } while (regval & 0x80000000);

  /* write the character */
  *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA) = c;
}
```

```
ser_write:
        li      a4,268513280
.L17:
        lw      a5,0(a4)
        bltz    a5,.L17
        sw      a0,0(a4)
        ret
```

a4 = 0x10013000

a5 = *a4

Branch to .L17 if a5 < zero

Acknowledgements

- Some slides are based on the material originally developed by
  - Edward A. Lee and Prabal Dutta (UCB) for EECS149/249A
  - Rodolfo Pellizzoni (U. Waterloo) for ECE224