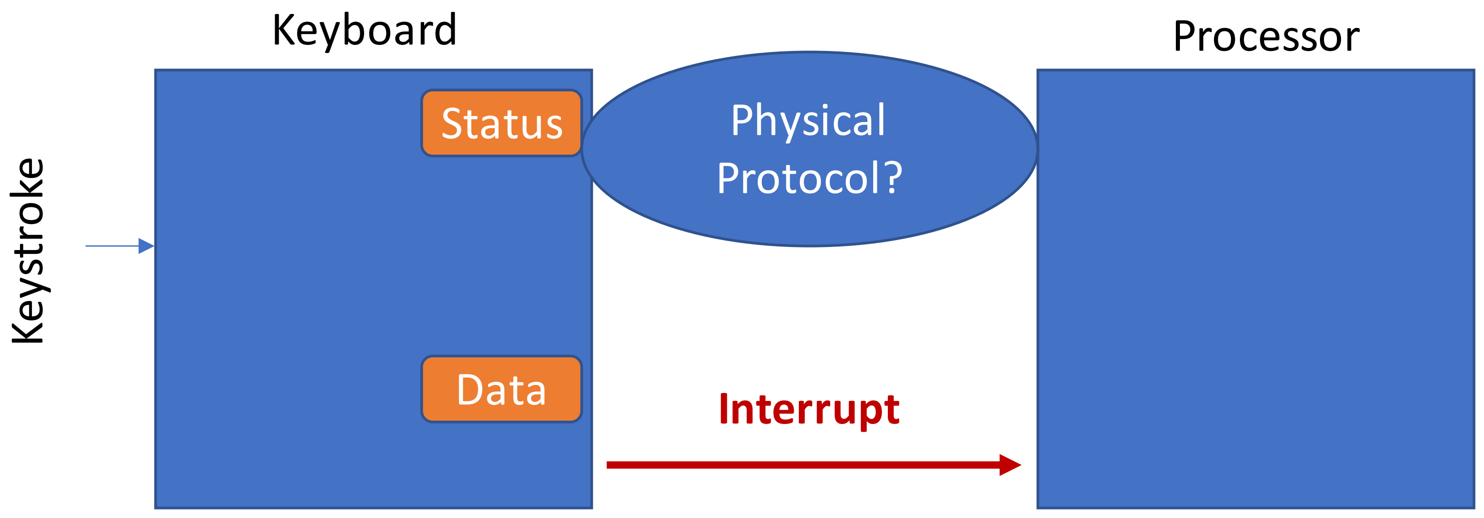
KU
THE UNIVERSITY OF
KANSAS

I/O Interfaces

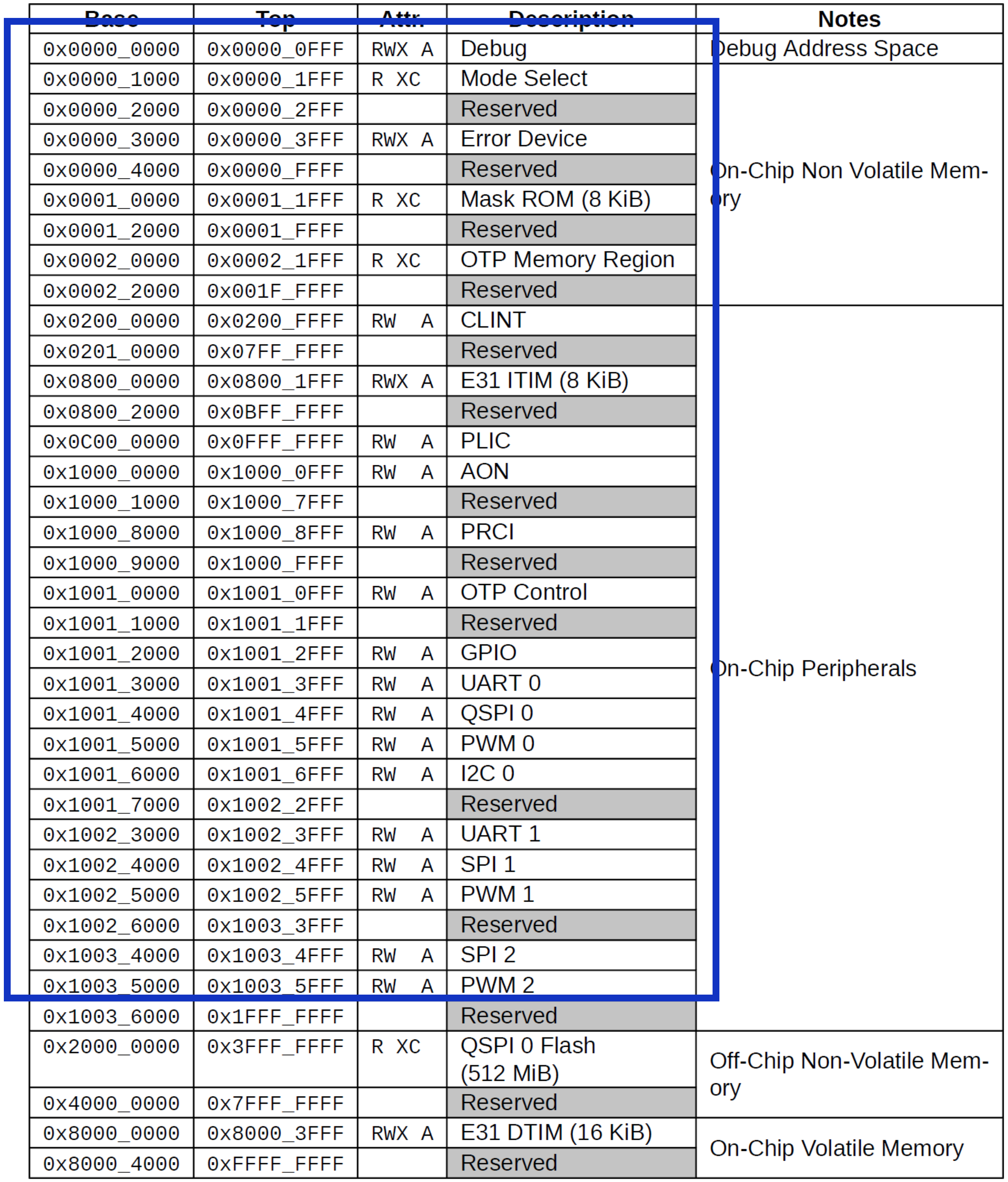
EECS388 Fall 2022

Context

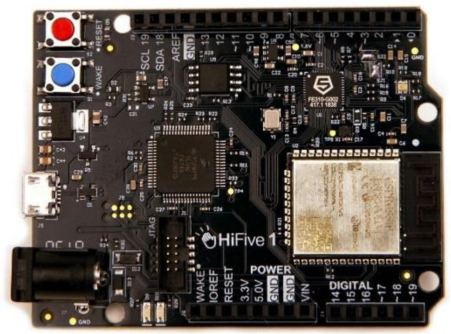
- Recommended reading:
  - Si-five Programmer's Manual
  - Chapter 11 of "AVR Microcontroller and Embedded systems"
  - Wikipedia



Memory Map of SiFive FE310

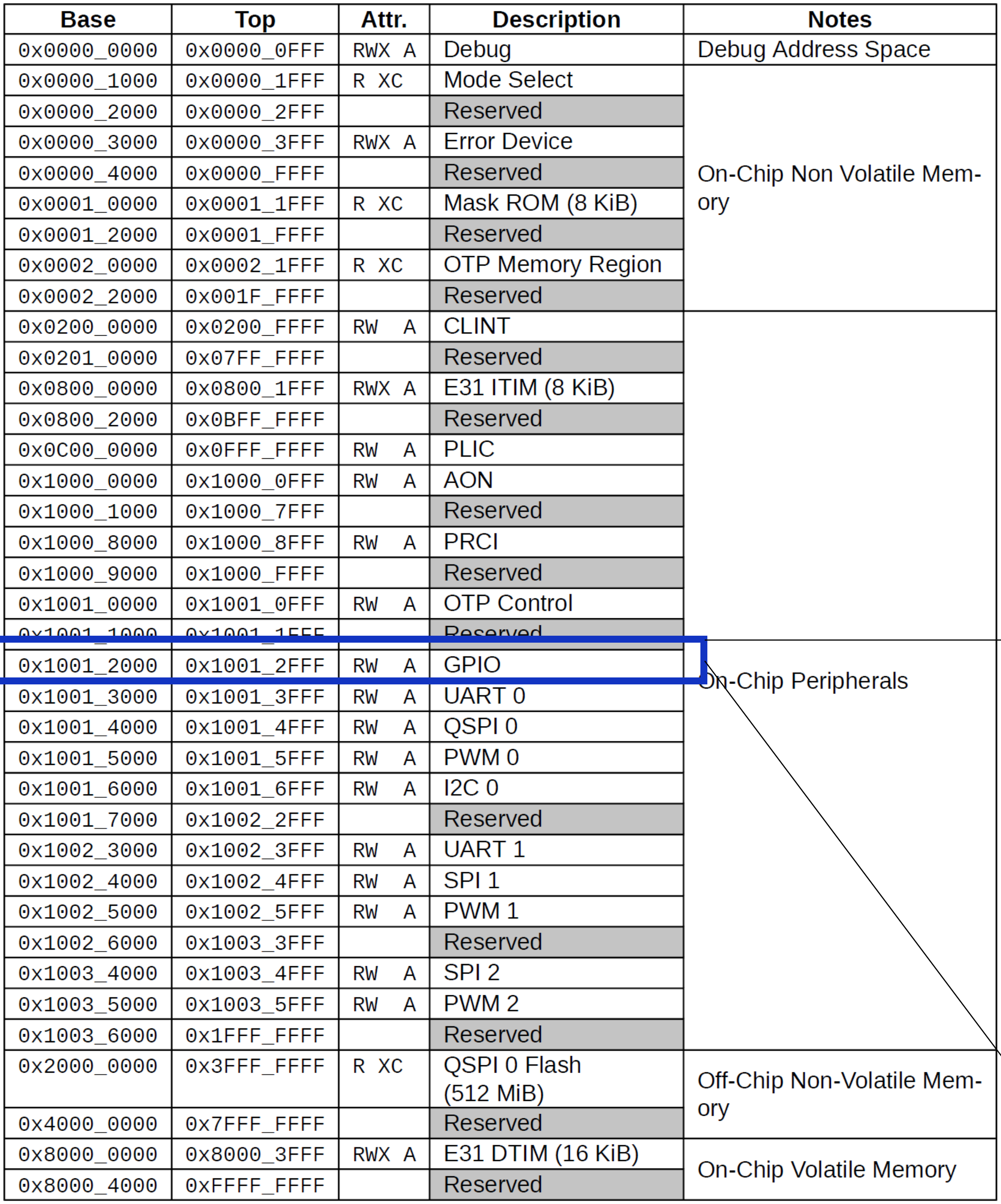


Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_0FFF	RWX A	Debug	Debug Address Space
0x0000_1000	0x0000_1FFF	R XC	Mode Select	On-Chip Non Volatile Memory
0x0000_2000	0x0000_2FFF		Reserved	
0x0000_3000	0x0000_3FFF	RWX A	Error Device	
0x0000_4000	0x0000_FFFF		Reserved	
0x0001_0000	0x0001_1FFF	R XC	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	



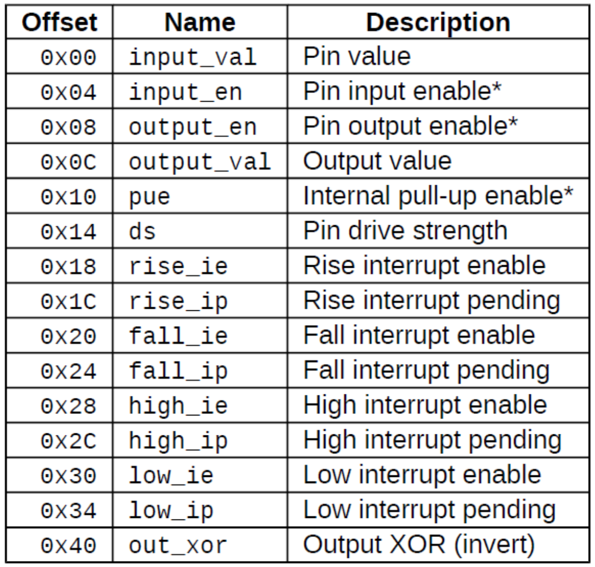
Memory mapped I/O regions

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	

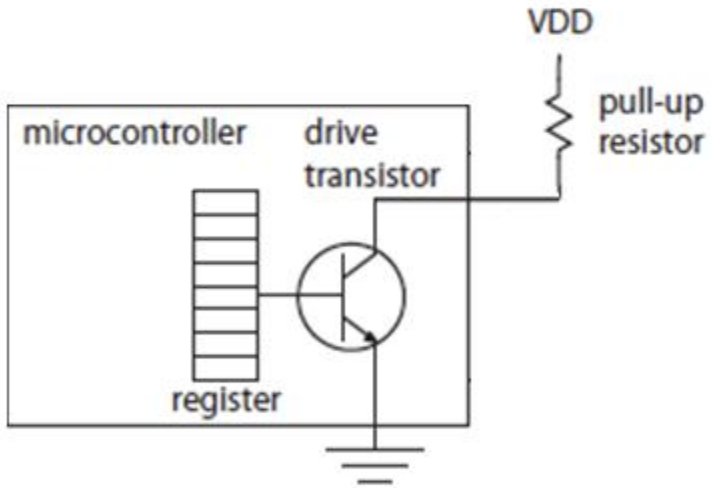
GPIO registers are mapped at 0x10012000 – 0x10012FFF



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)

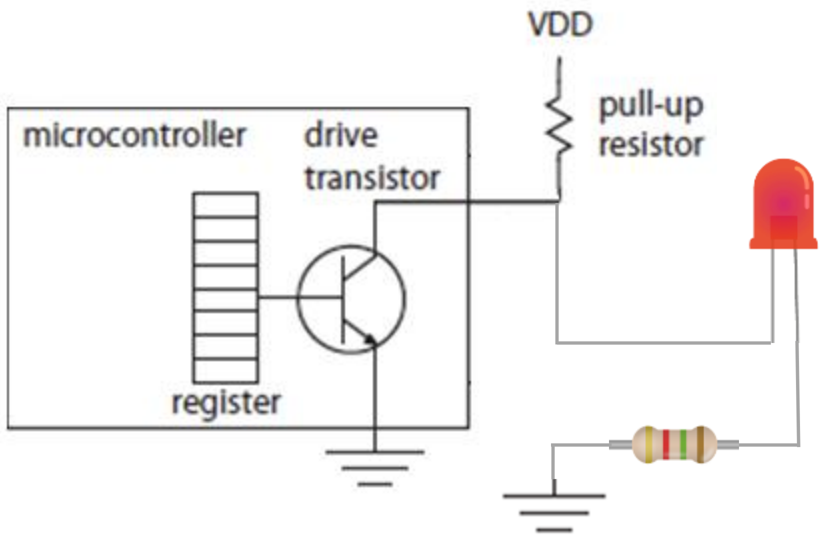
General Purpose I/O (GPIO)

- Programmable digital input/output pins
- Use voltage levels to represent digital signals
  - 3.3V = logic 1 (for 3.3V microcontrollers)
  - 0V = logic 0
- Can be configured as
  - Input or output
- Useful to interact with external devices



Example: Turn on an LED

- Assume VDD = 3.3, GPIO pin can draw up to 18mA, the LED's nominal voltage is 2.1V
- Ohm's law: $I \times R = V$
- What resistor is needed?



Example: Turn on an LED

- Assume VDD = 3.3, GPIO pin can draw up to 18mA, the LED's nominal voltage is 2.1V
- Ohm's law: $I \times R = V$
- What resistor is needed?

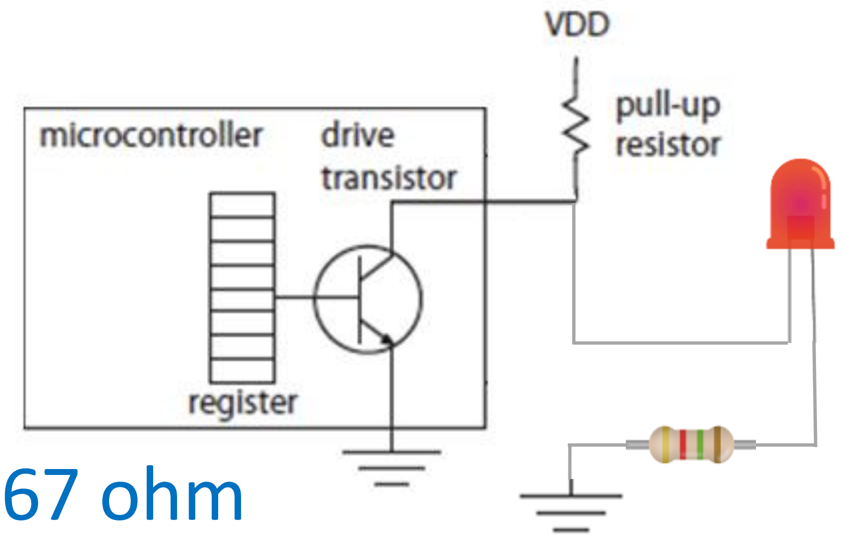
I = V / R = 1.2 / R < 18mA

R > 1.2 V/18mA

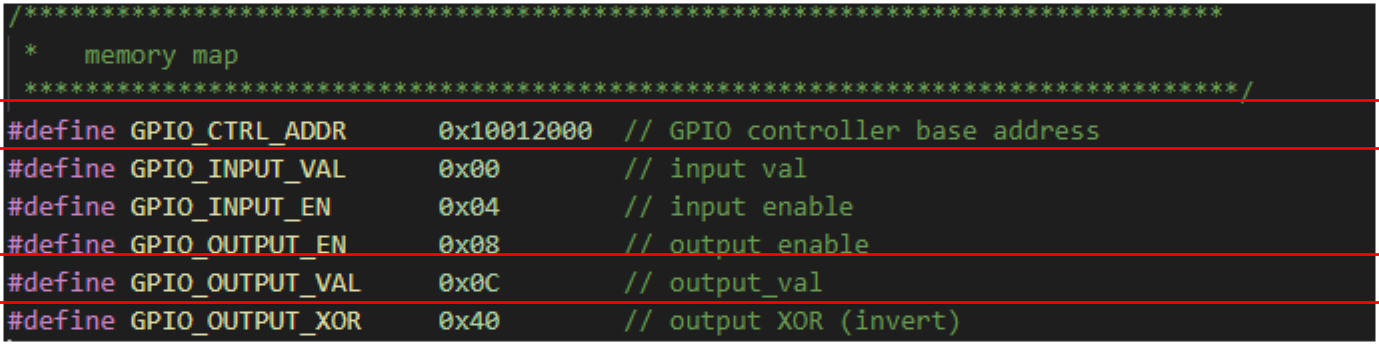
= 1200mV/18mA

= 67 ohm

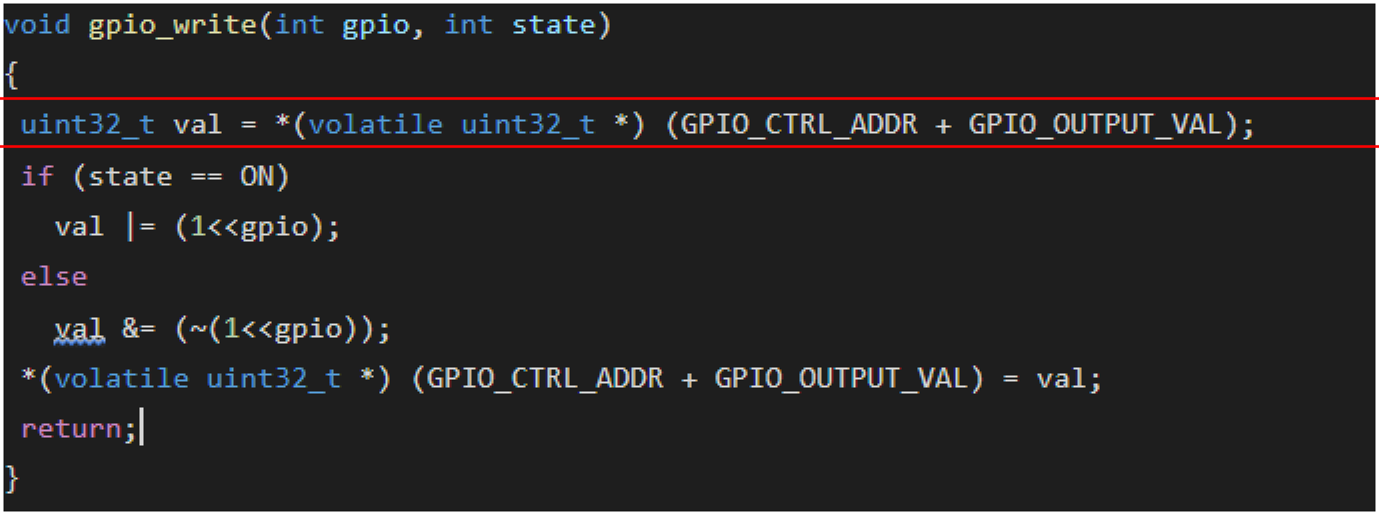
R should be greater than 67 ohm



Blinky LED on HiFive Board

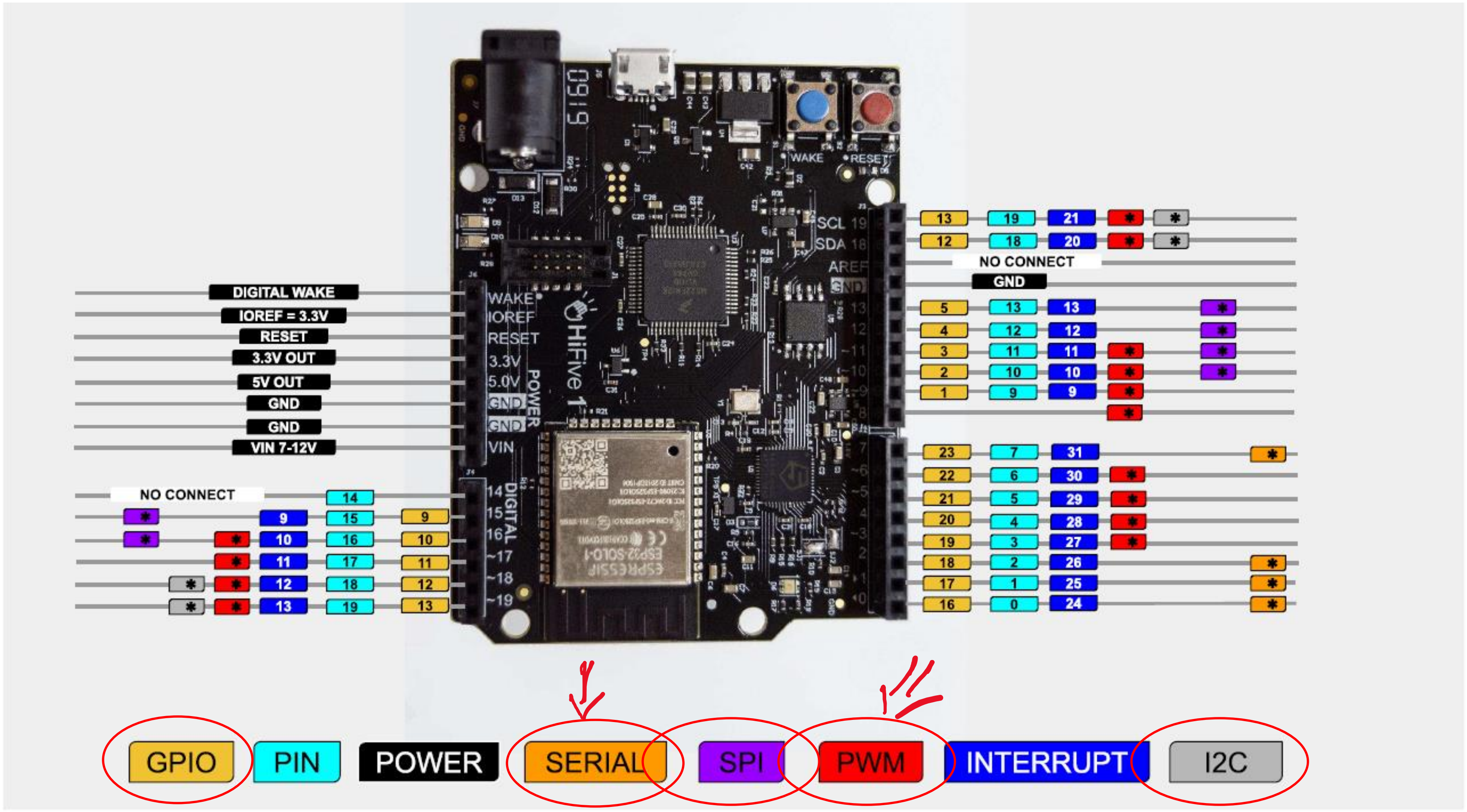


```
/****************************************************************************
 *   memory map
 ****************************************************************************/
#define GPIO_CTRL_ADDR      0x10012000  // GPIO controller base address
#define GPIO_INPUT_VAL      0x00        // input val
#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;
}
```

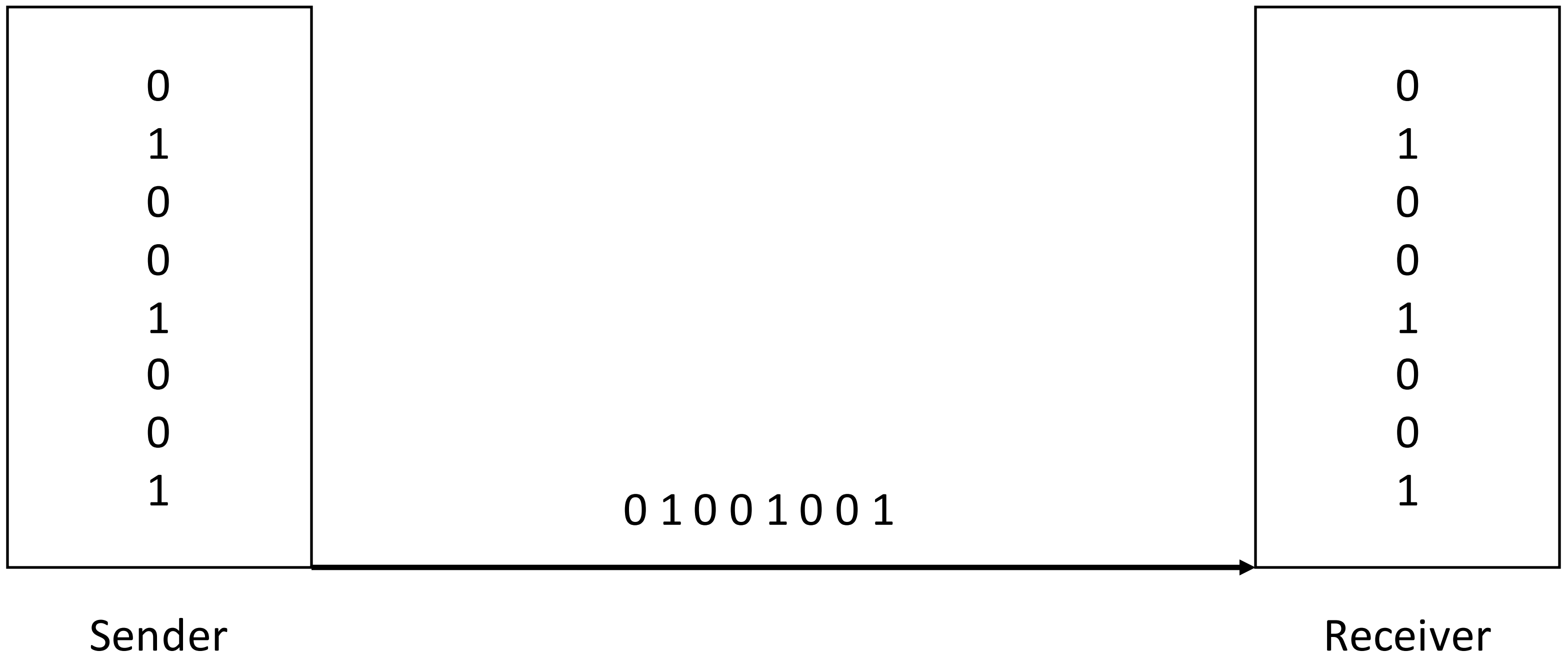
External I/O Interfaces



HiFive1 Rev B Pinout

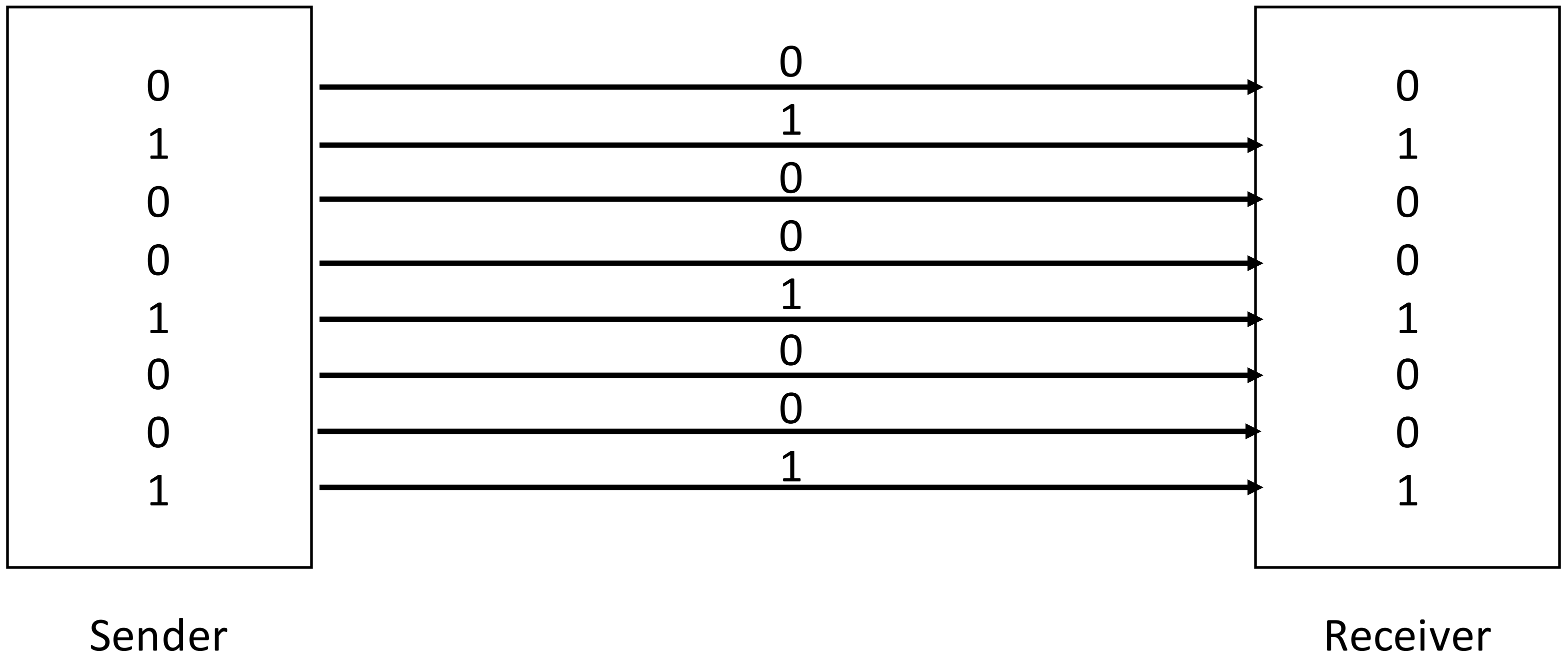
Serial vs. Parallel

Serial interfaces use a single line



Serial vs. Parallel

Parallel interfaces 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

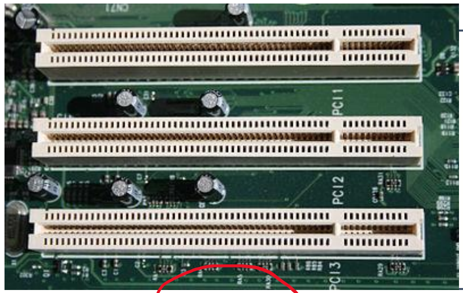
USB

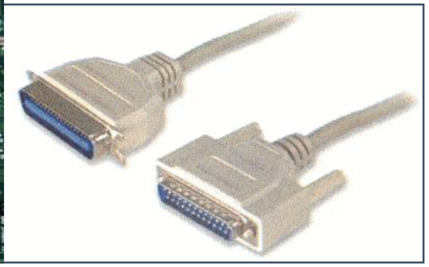
RS-232

Serial vs. Parallel Interfaces

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

PCI x1 => serial   PCI x 2   x16
x 4
x 8


PCI


SCSI

PATA

Parallel port

Serial Interfaces

- ✓Use fewer pins and wires
- ✓High scalability
- ✓Low power consumption
- ✗ Lower bandwidth for the same clock speed

Parallel Interfaces

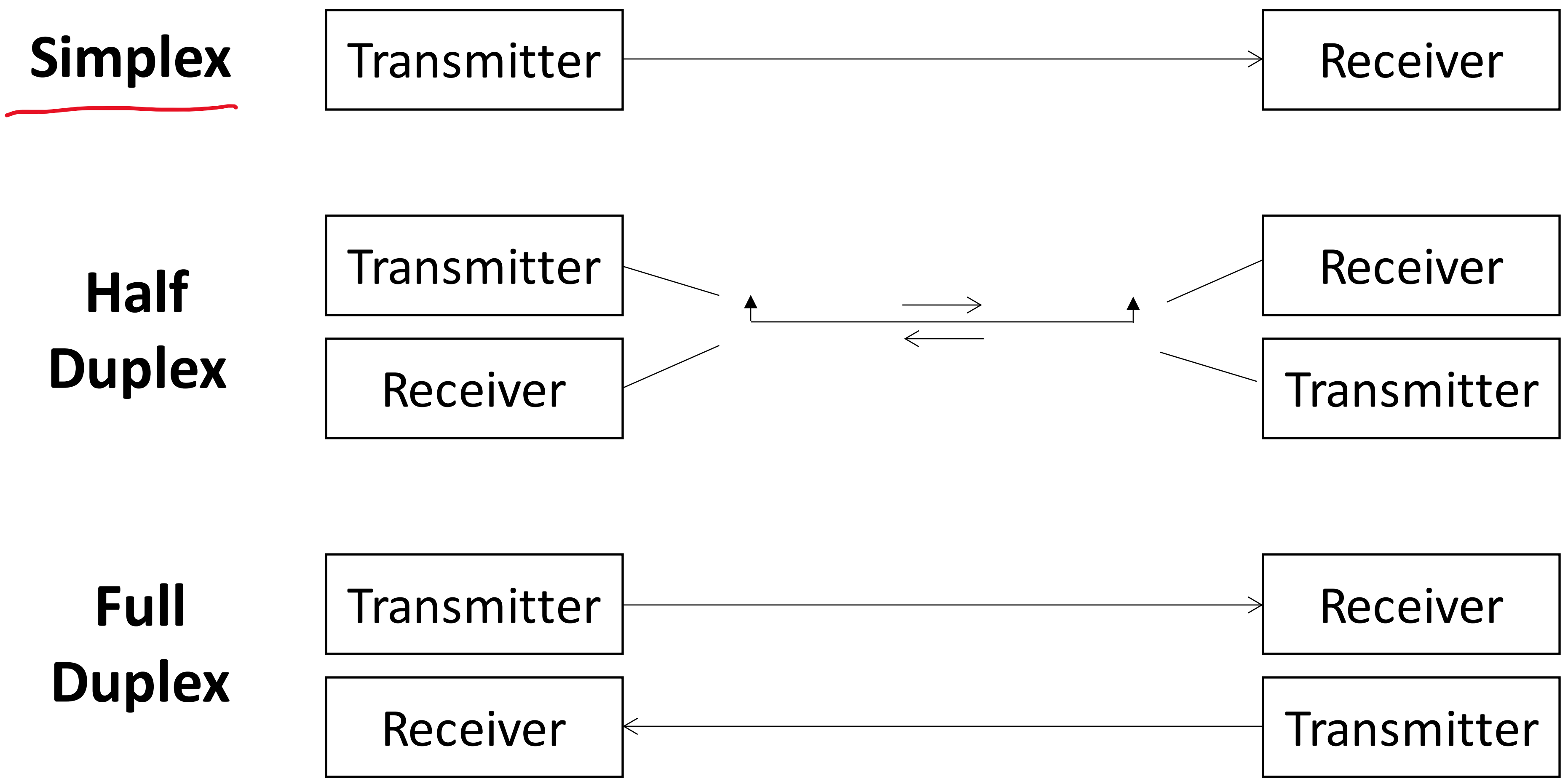
- ✗ More pins and wires
- ✗ Synchronization among lanes reduces scalability
- ✗ High power consumption
- ✓Higher bandwidth

- Serial interfaces for external I/O (e.g., USB, SATA, PCIe)
- Parallel interfaces for on-chip interconnects and memory interfaces (e.g., AXI, TileLink, DDR)

$\uparrow$

PCI

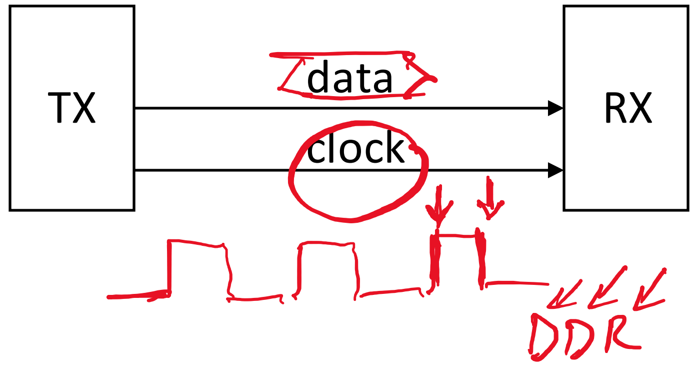
Transfer Types



Synchronous vs. Asynchronous

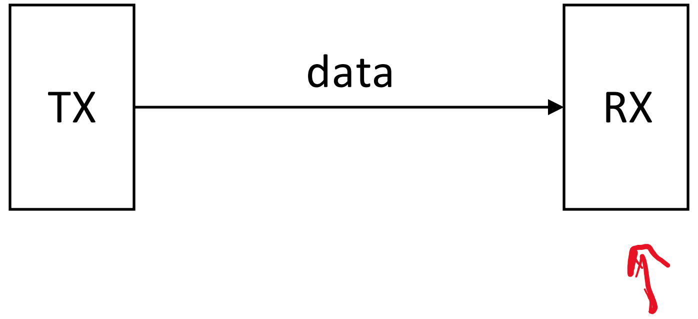
Synchronous transmission

- Requires a common shared clock
- Higher throughput communication
- Low scalability



Asynchronous transmission

- No shared clock
- Asynchronous start/stop
- Self clocked, based on an agreement between the transmitter and receiver

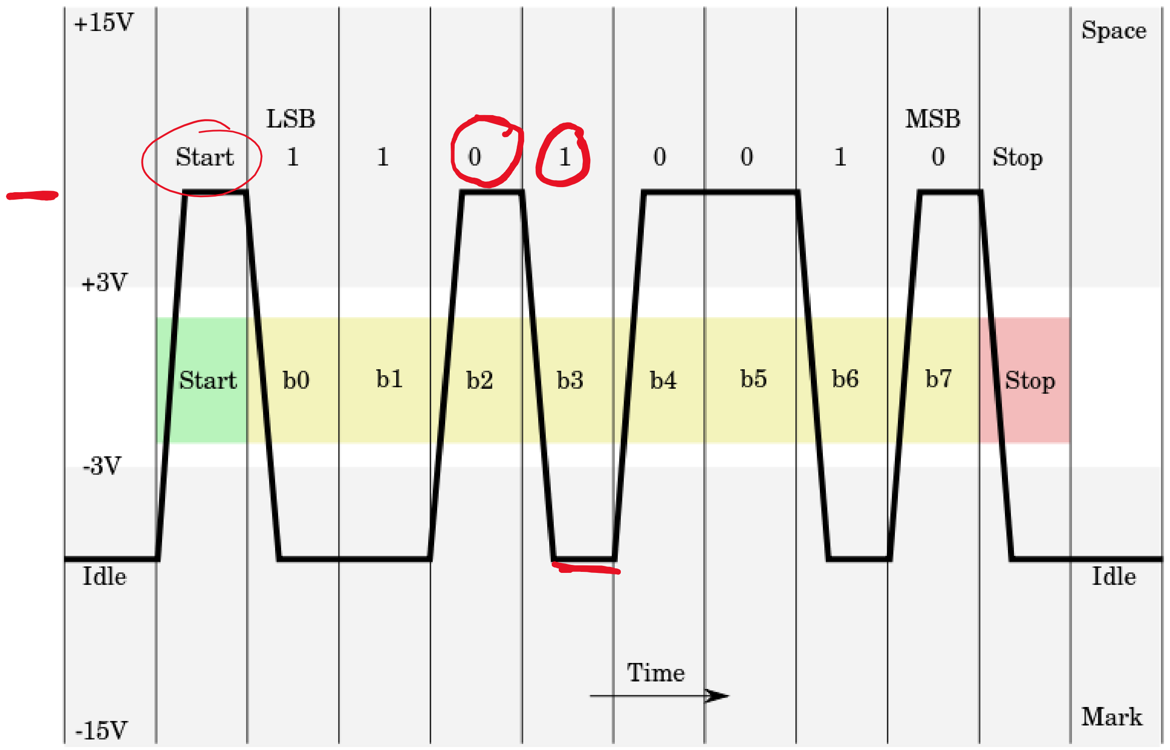


Serial Communication Standard: RS-232

- First introduced in 1962 to connect teletypes to modems
- Electrical signals and connector types standard
- Because the standard was set long before the advent of the TTL logic family, its input and output voltage levels are not $\underline{\text{TTL}}$ compatible
- A $\boxed{1}$ is represented by $\underline{-3 \text{ to } -15 \text{ V}}$
- A 0 is represented by $\underline{+3 \text{ to } +15 \text{ V}}$

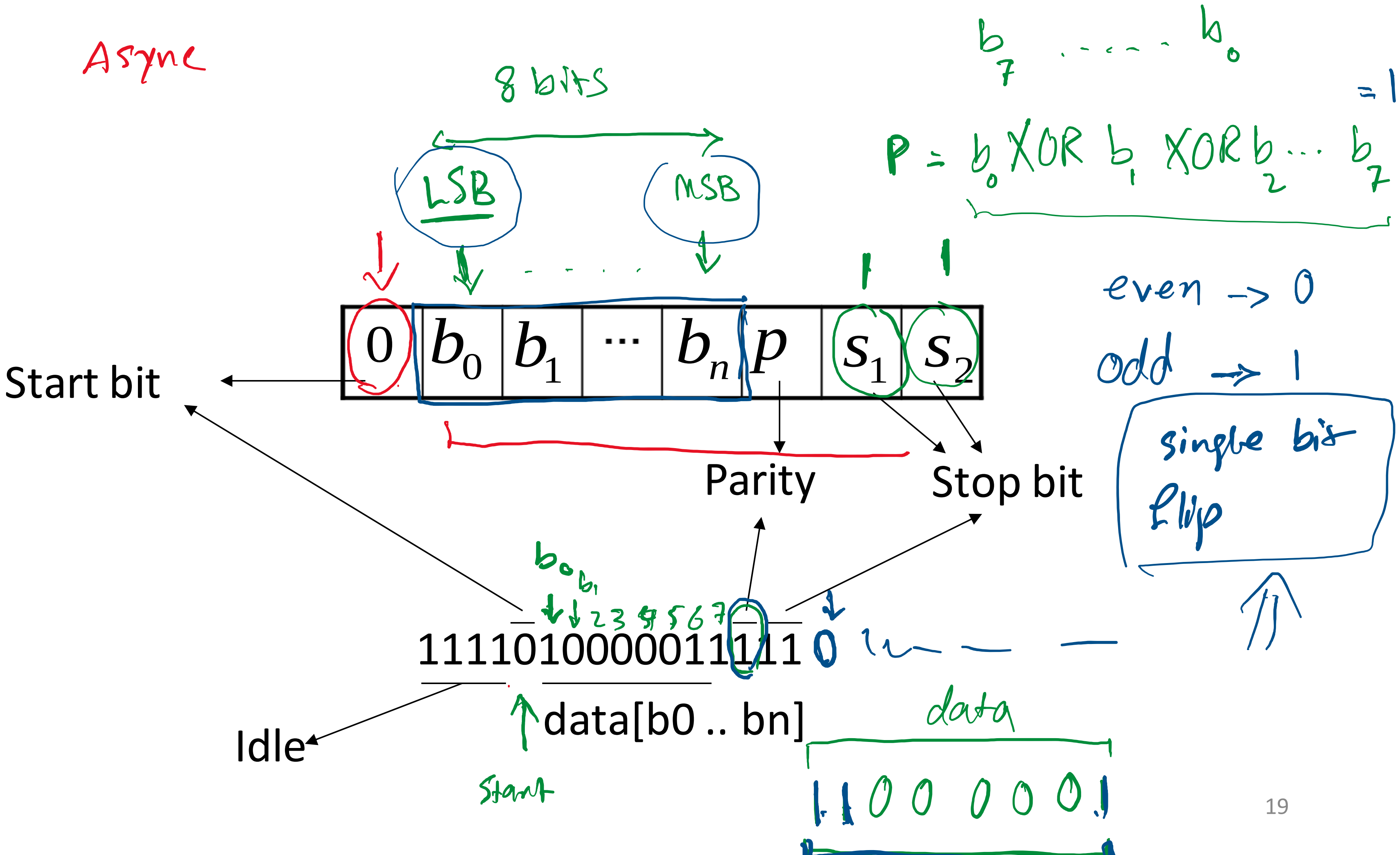
V		logic
0	$\rightarrow$	0
3.3	$\rightarrow$	1

RS-232


+15V
Space
LSB
MSB
Start
1
1
0
1
0
0
1
0
Stop
+3V
Start
b0
b1
b2
b3
b4
b5
b6
b7
Stop
-3V
Idle
Idle
Time
Mark
-15V

RS-232 Frame Format

Async



Example

- What is the data that the following RS-232 frame is transmitting? Is there any error in the frame?

0xDF

0b11011111

data   ! P   Stop   idle

idle

1111110(start)**11111011(data)**0 (parity)**11(stop)**11111

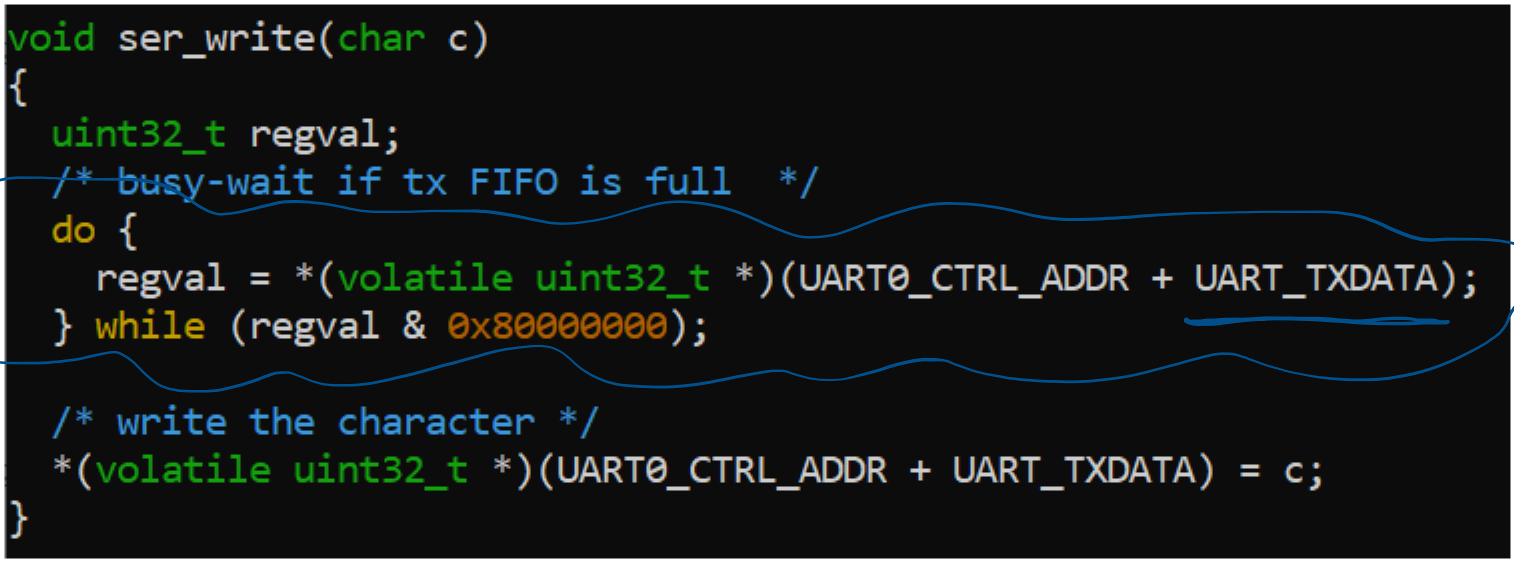
0b**11011**111 = 0xDF   There is an error

Universal Asynchronous Receiver/Transmitter (UART)

Convert parallel content of an 8-bit register to a bit sequence ready to be transmitted over a serial port e.g., RS-232

UART Asynchronous Transmission

- First check if the TX FIFO is not full then transmit a byte



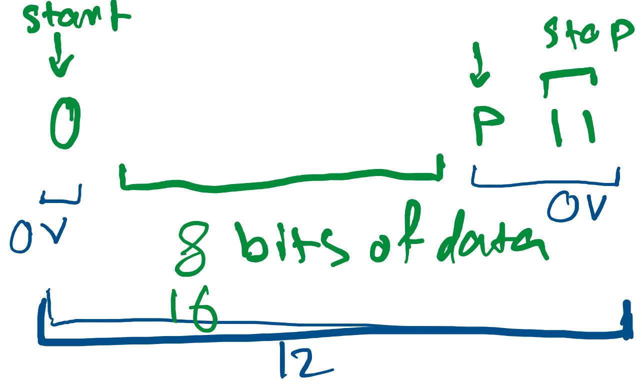
```
void ser_write(char c)
{
  uint32_t regval;
  /* busy-wait if tx FIFO is full  */
  do {
    regval = *(volatile uint32_t *)(UART0_CTRL_ADDR + UART_TXDATA);
  } while (regval & 0x80000000);

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

UART Speed (Baudrate)

Both sender and receiver must use agreed upon transmission speed (baudrate)

- When the start bit, the receiver samples 8 more bits before stop

Example

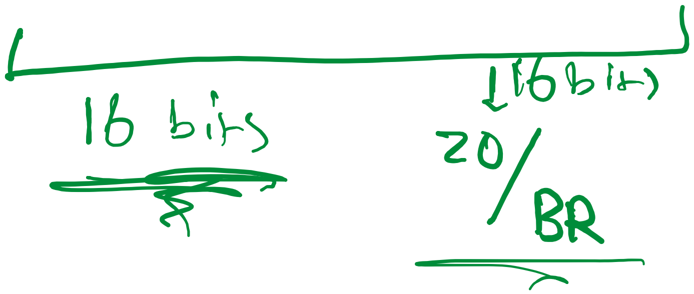


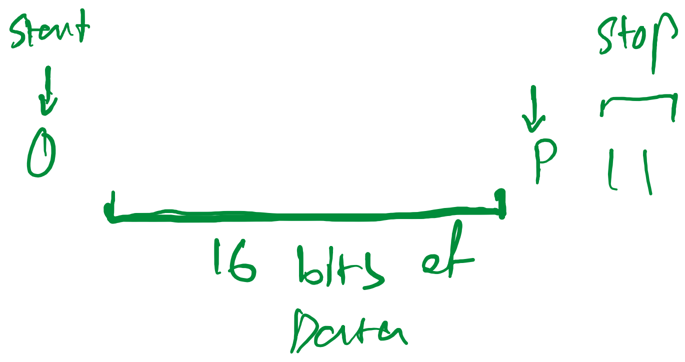
16
↓
24/BR

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

RS 232

- 12/115200 (sec)

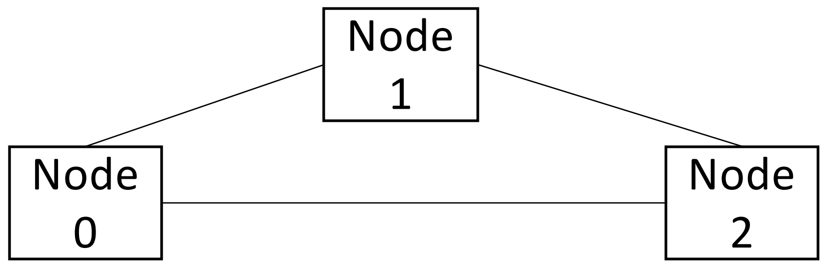




Types of Networks

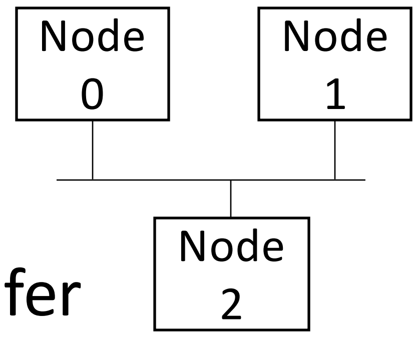
Point-to-point

- 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



Types of Networks

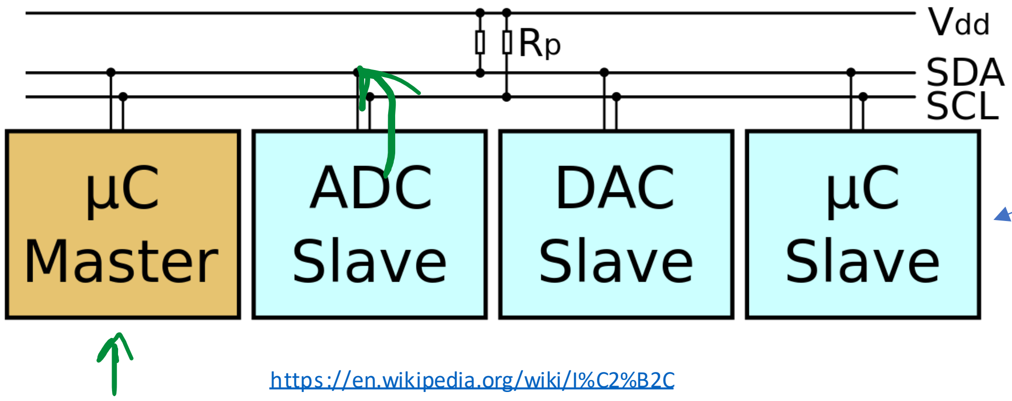
- What is the advantage and disadvantage of a bus compared with a point to point interconnect?
  - Point to point is faster when there is a direct connection
  - Bus needs arbitration
  - Bus is more efficient (higher utilization)
  - Point to point is more scalable

I^{2}C

Inter-Integrated Circuit protocol (I^{2}C), by NXP

- Wires used: 2 (SCL: clock, SDA: data)
- Speed: (standard) 100Kbps, (ultra fast) 5Mbps
- Serial, synchronous bus
- 7 or 10 bits for slave IDs: 127 ~ 1008 devices

Generates clock, initiate communication

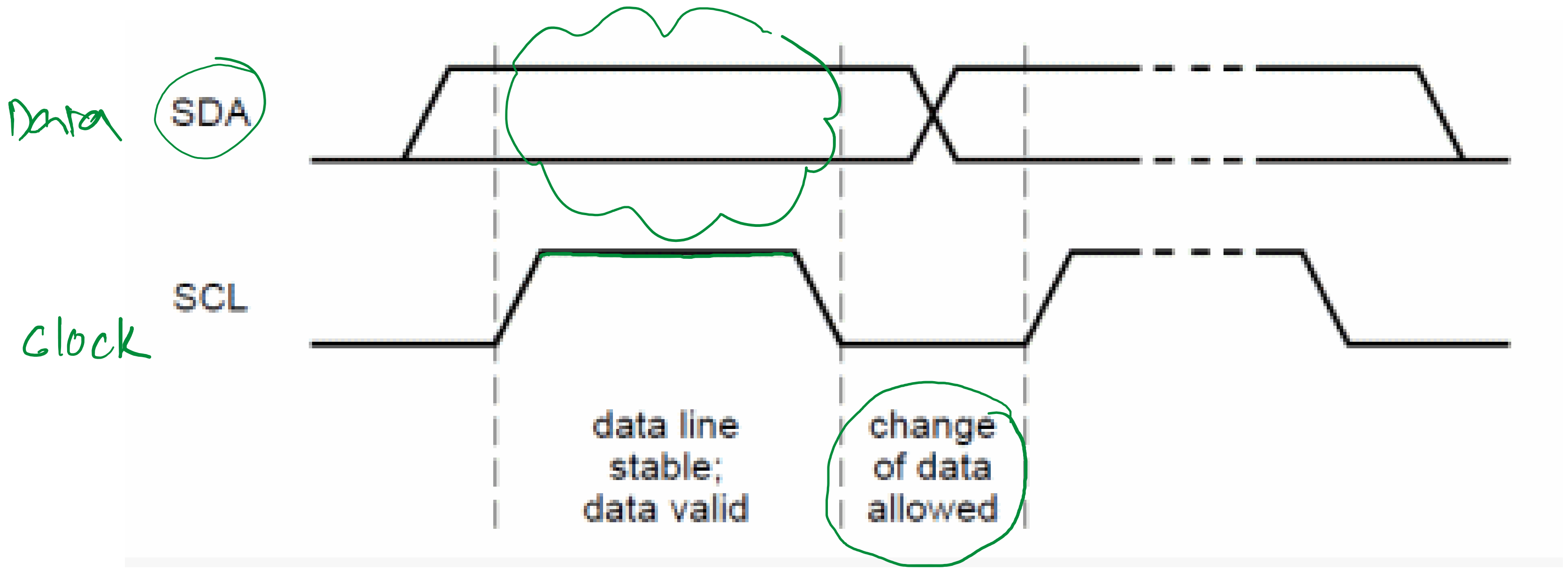


Receives clock, responds when addressed

write
read

https://en.wikipedia.org/wiki/I%C2%B2C

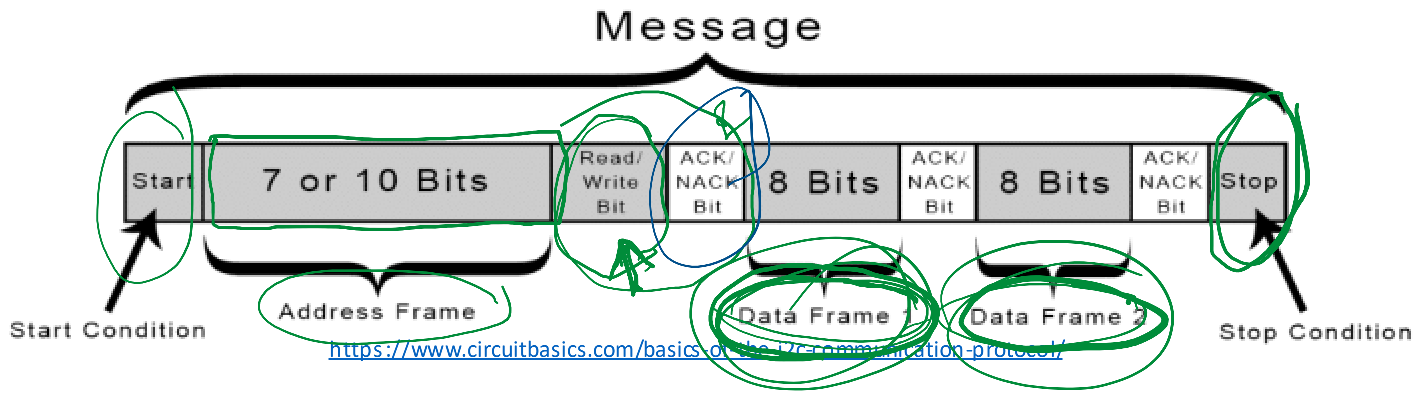
Synchronous, Serial Data Transfer in I^2C



https://i2c.info/i2c-bus-specification

I²C Protocol

16 bit

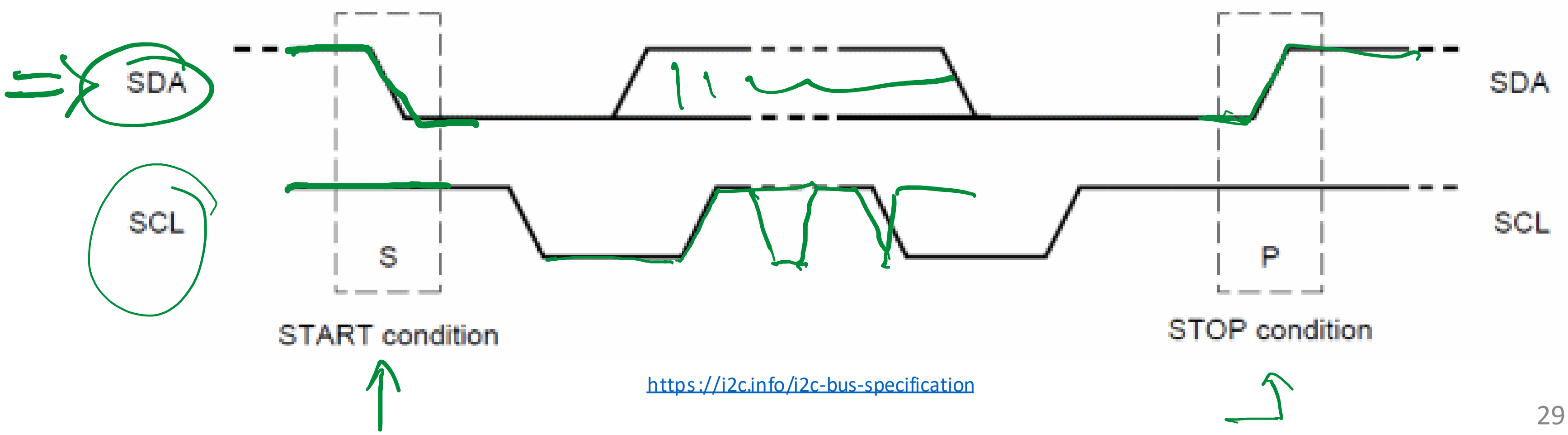


https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

- Start and stop conditions

SDA high to low while SCL is high

SDA low to high while SCL is high



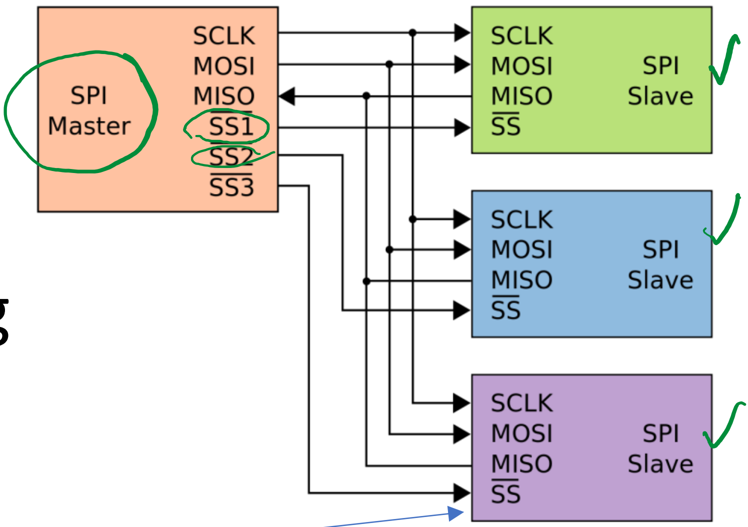
https://i2c.info/i2c-bus-specification

SPI

Serial Peripheral Interface (SPI), by Motorola

- Synchronous, serial communication protocol
- Uses 4 lines, full-duplex, over 10Mbps
- Single master, multi-slave
- No start/stop bits
- Good for fast, short distance communication, e.g., connecting

  LCD to µController

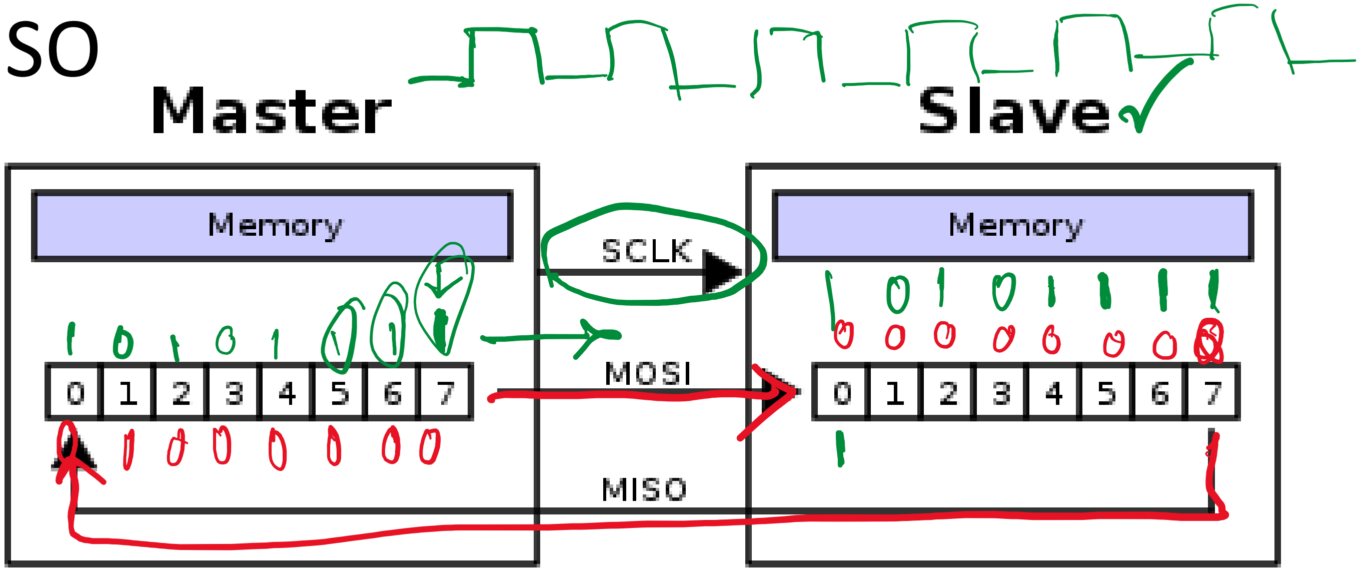


Slave Select (SS)

https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

SPI Protocol

- Master shifts out to MOSI (Master Out Slave In) and shifts in from MISO (Master In Slave Out)
- Slave shifts in from MOSI and shifts out to MISO

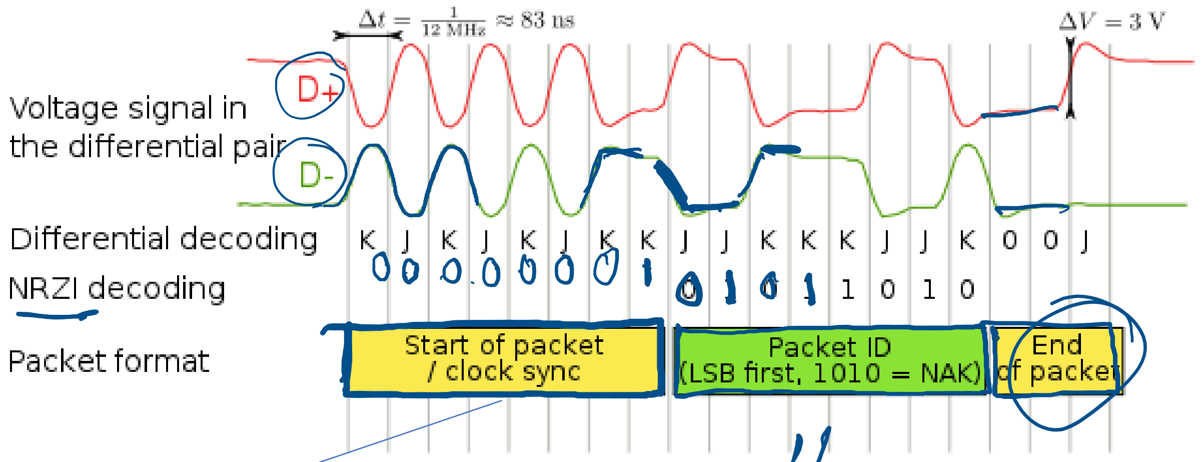


https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

Universal Serial Bus (USB)

Point to point, synchronous serial communication

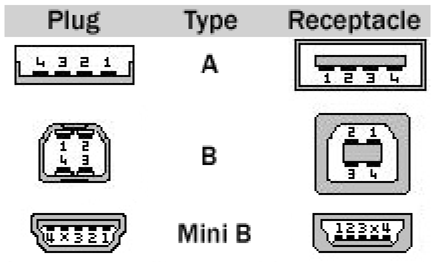
- One host, multiple devices, can form a tree
- Use two wires for single differential signal
  - Reduce noise caused by electromagnetic interference



$\Delta t = \frac{1}{12\text{ MHz}} \approx 83\text{ ns}$
$\Delta V = 3\text{ V}$
Voltage signal in the differential pair
D+
D-
Differential decoding
K J K J K J K K J J K K K J J K 0 0 J
NRZI decoding
0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0
Packet format
Start of packet / clock sync
Packet ID (LSB first, 1010 = NAK)
End of packet

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

00000001



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

I/O Considerations

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

Recap

I/O interfaces

- GPIO, UART, SPI, I2C, USB, …
- Serial vs. parallel
- Asynchronous vs. synchronous