Lecture 05
Digital Circuits

CE346 – Microprocessor System Design

Branden Ghena – Fall 2021

Some slides borrowed from:
Josiah Hester (Northwestern), Prabal Dutta (UC Berkeley)

Administrivia

- Labs
  - Debrief: How did that go?
    - Can use personal computers if preferred
  - See schedule of Lab hours available on Canvas
- Quiz
  - Today at end of class

Project Proposals

- It is time to start forming teams and working on Proposals
  - Due next week Tuesday!

  - Project details are posted to Campuswire
  - Specific proposal details are on the Canvas assignment

Today’s Goals

- Explore a final interaction pattern: DMA
- Understand the basics of digital circuitry
  - Enough to be able to interact with the Microbit
- Finish our exploration of the GPIO peripherals on the Microbit
- One step deeper in to EE-land: energy use

Outline

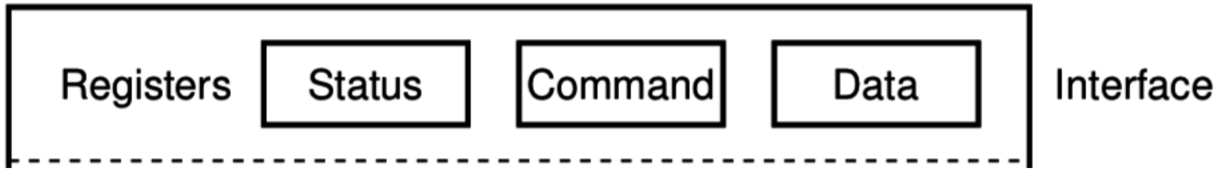
- **DMA**

- Digital Circuits

- GPIOTE

- Energy

Reminder: Polling I/O

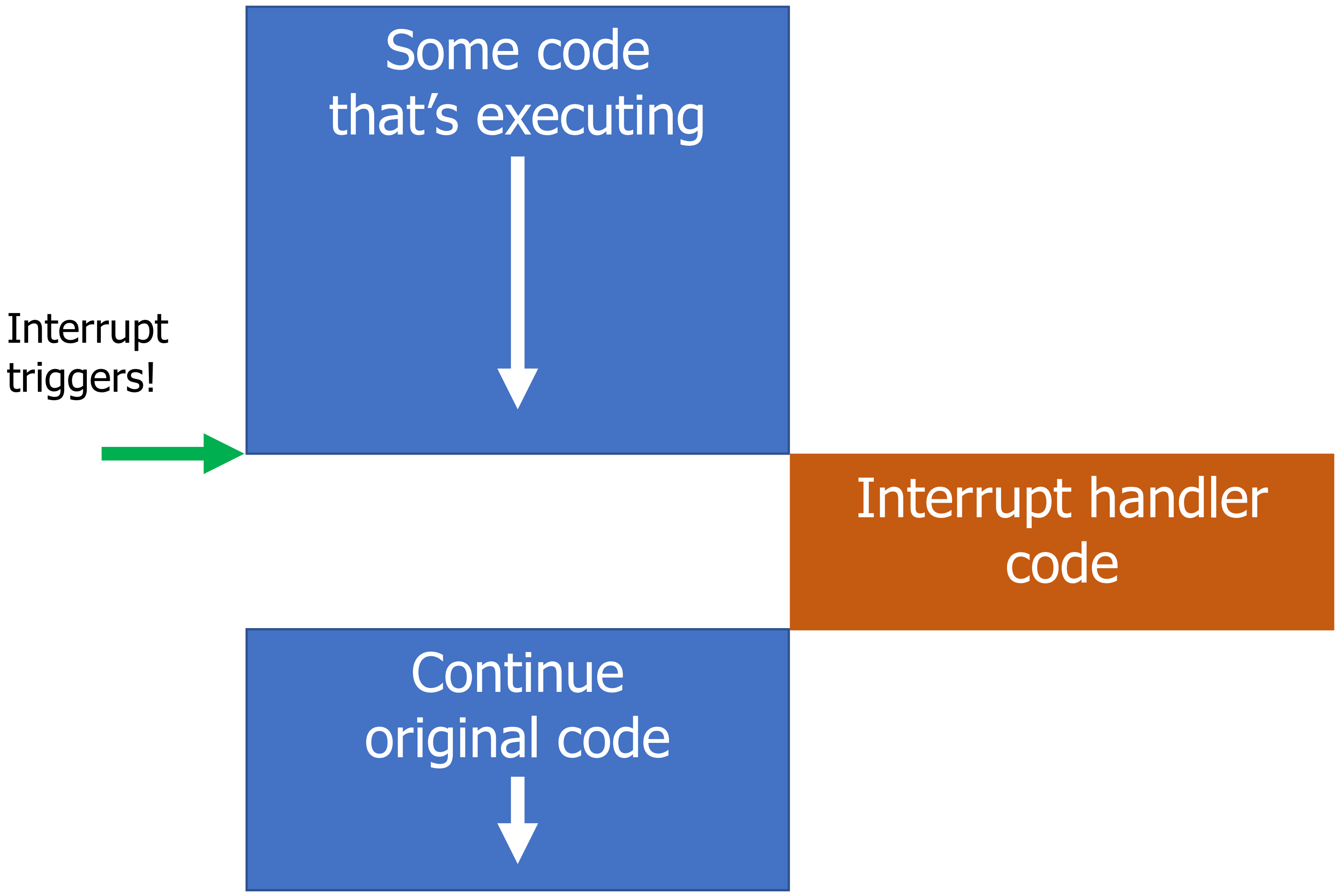


1. while STATUS==BUSY; Wait
   - (Need to make sure device is ready for a command)
2. Write value(s) to DATA
3. Write command(s) to COMMAND
4. while STATUS==BUSY; Wait
   - (Need to make sure device has completed the request)
5. Read value(s) from Data

This is the “polling” model of I/O.

“Poll” the peripheral in software repeatedly to see if it’s ready yet.

Reminder: Interrupts, visually

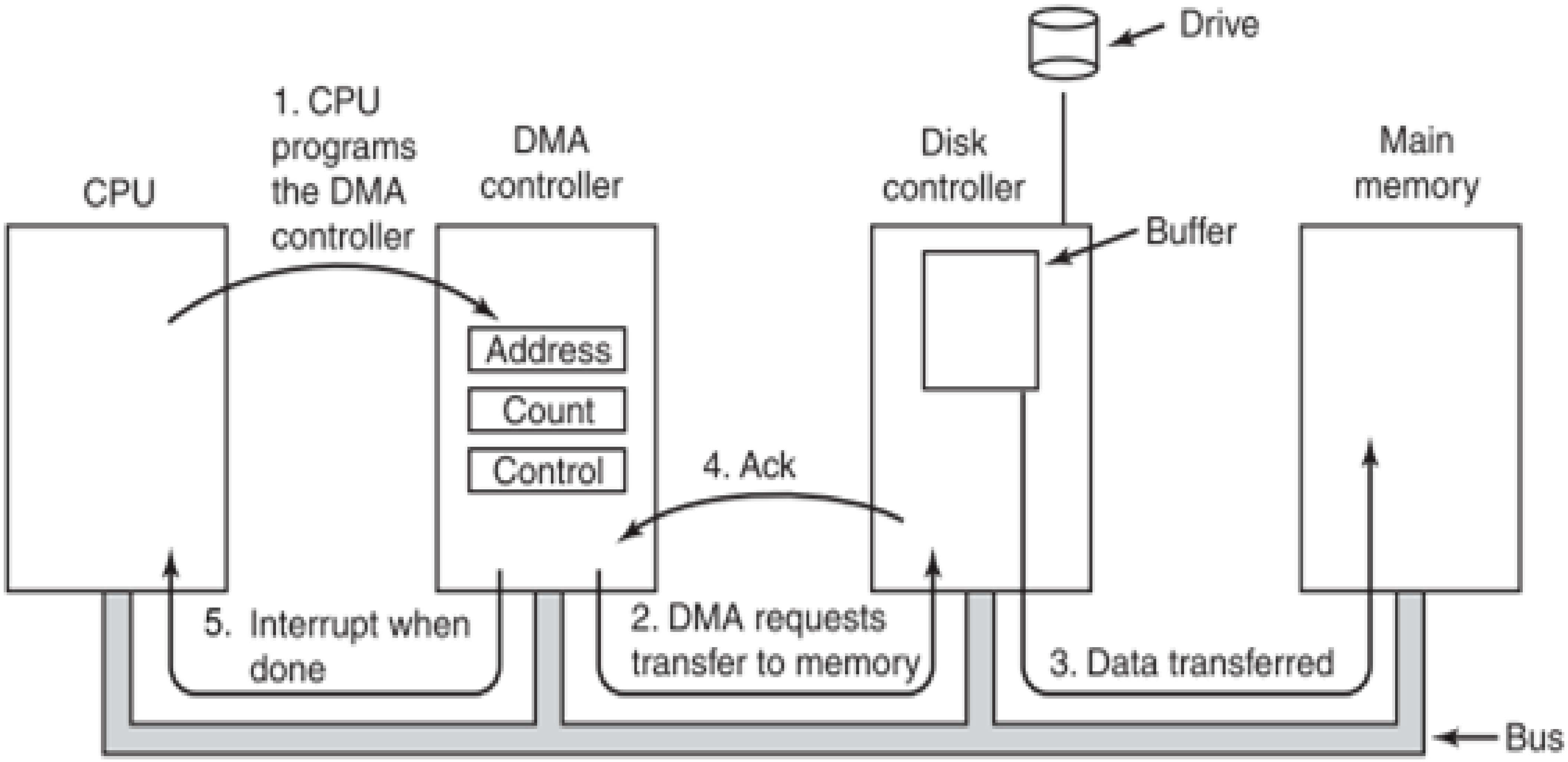


Direct Memory Access (DMA)

- Even with interrupts, providing data to the peripheral is time consuming
  - Need to be interrupted every byte, to copy the next byte over

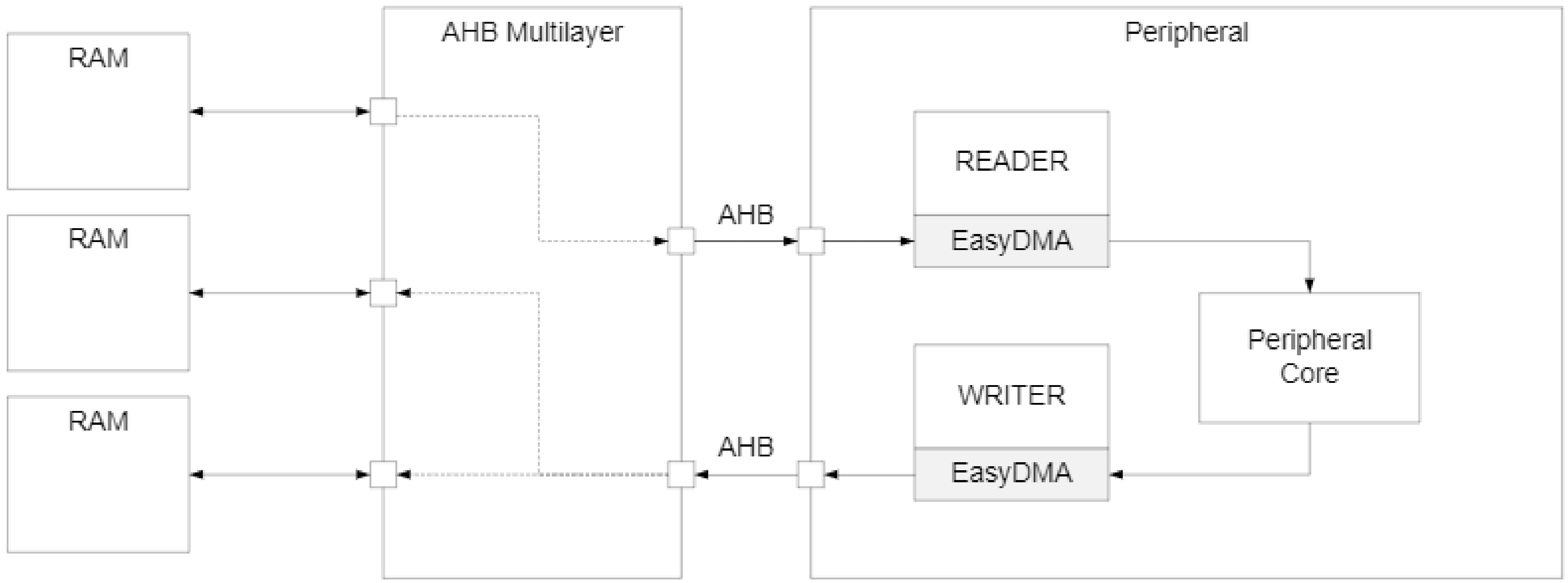
- DMA is an alternative method that uses hardware to do the memory transfers for the processor
  - Software writes address of the data and the size to the peripheral
  - Peripheral reads data directly from memory
  - Processor can go do other things while read/write is occurring

General-purpose DMA



Special-purpose DMA

- nRF52 uses “EasyDMA”, which is built into individual peripherals
  - Only capable of transferring data in/out of that peripheral
  - Easier to set up and use in practice
  - Only available on some peripherals though (no DMA for TEMP)



Warning: addresses for DMA MUST be in RAM!

Full peripheral interaction pattern

1. Configure the peripheral
2. Enable peripheral interrupts
3. Set up peripheral DMA transfer
4. Start peripheral

Continue on to other code

5. Interrupt occurs, signaling DMA transfer complete
6. Set up next DMA transfer

Continue on to other code, and repeat

Outline

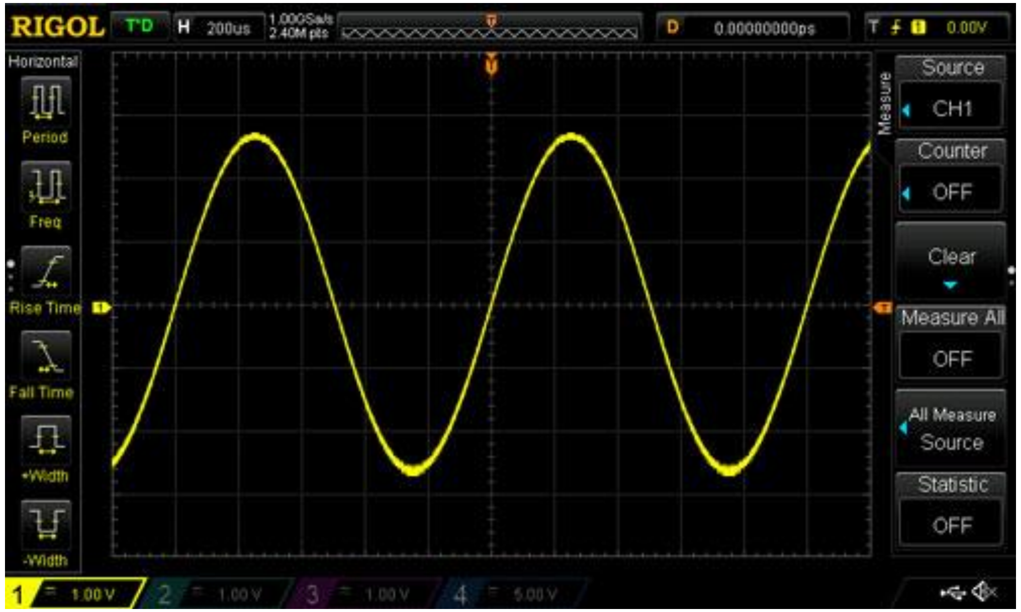
- DMA
- **Digital Circuits**
- GPIOTE
- Energy

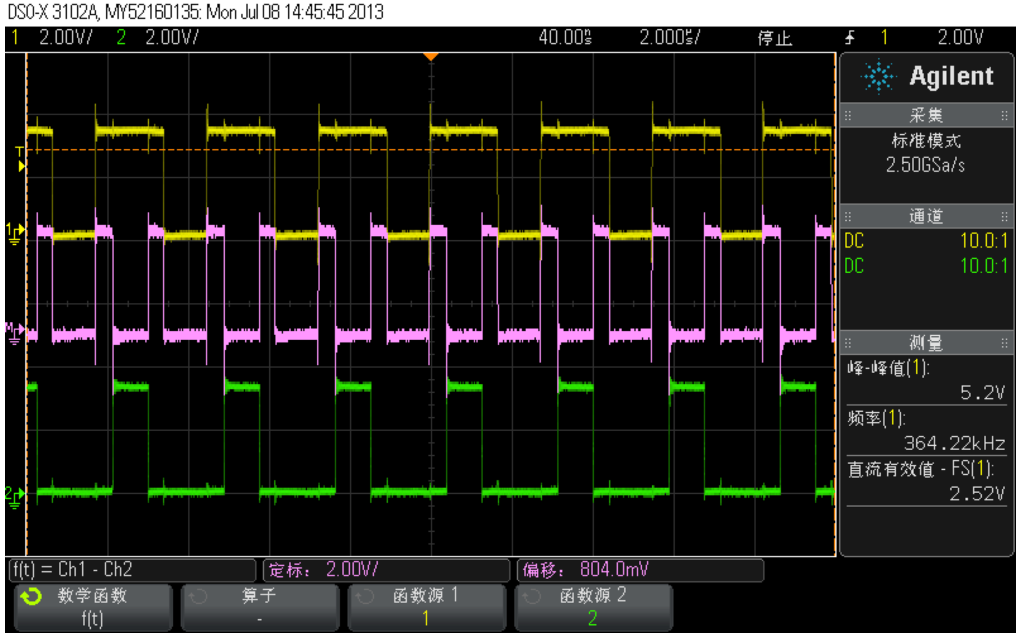
Digital signals

- Exist in two states:
  - High (a.k.a. Set, a.k.a. 1)
  - Low (a.k.a. Clear, a.k.a. 0)

- Simpler to interact with
  - Constrained to two voltages
  - With quick transitions between the two

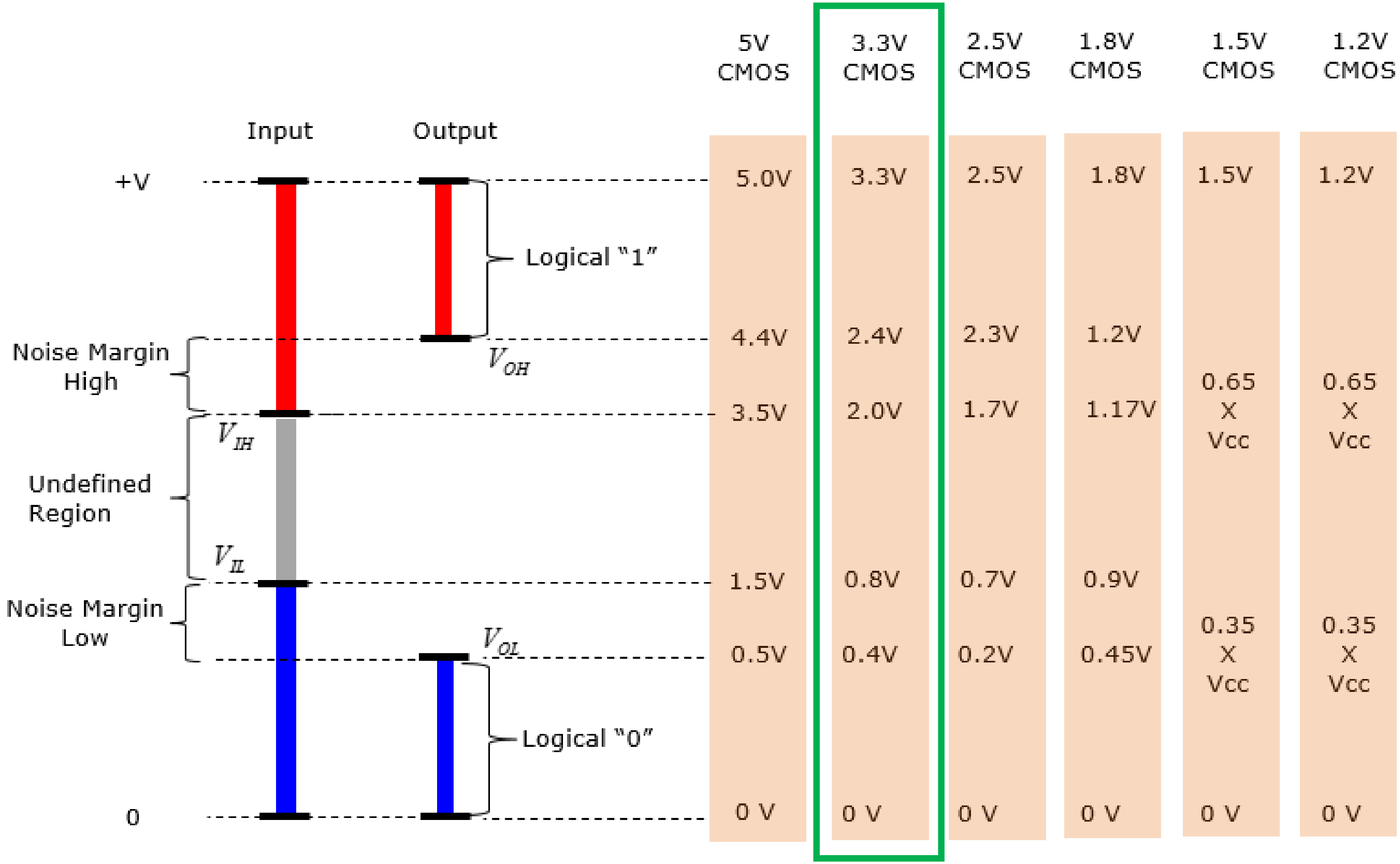
  - No math for voltage level
    - Either high or low





Digital signals map to voltage ranges

- Upper range is high signal
  - ~0.7*VDD
- Bottom range is low signal
  - ~0.3*VDD
- Middle is undefined
  - Only exists during transitions



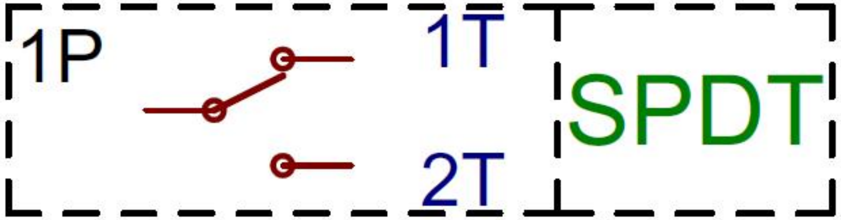
http://www.sharetechnote.com/html/Electronics_CMOS.html

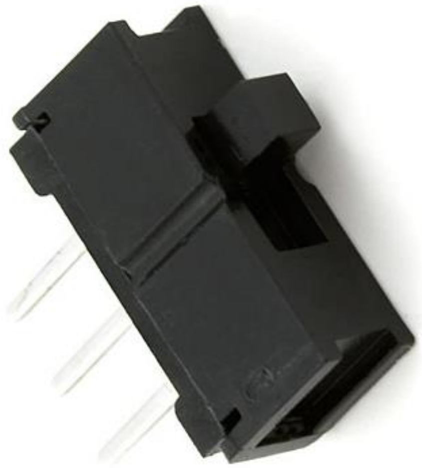
Digital circuits

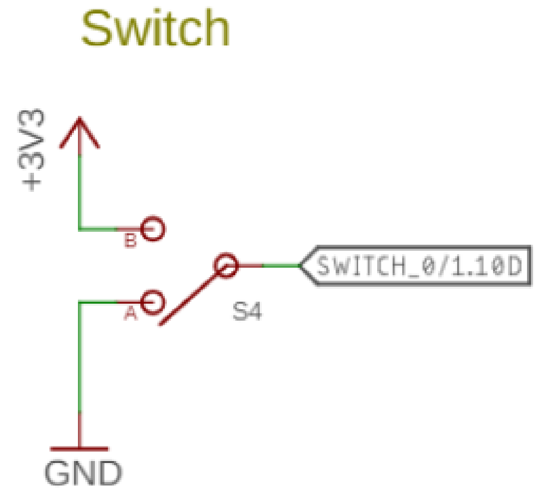
- Connecting components together with digital signals
  - Mostly ICs
  - Also buttons/switches and LEDs
- Way simpler than analog circuits
  - Mostly connecting boxes with wires
  - Plus a few resistors here and there
- An abstraction
  - Not sufficient for fully understanding electronics behavior, but close

Switches

- Single Pole, Double Throw switch
  - Middle pin (Pole) connects to one of two outer pins (Throws)





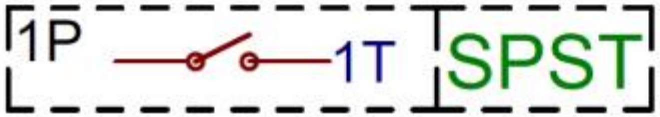


- For controlling microcontrollers
  - Often connect outer pins to VCC and Ground respectively
  - Input then goes High or Low depending on switch state

https://learn.sparkfun.com/tutorials/button-and-switch-basics/

Buttons

- Single Pole, Single Throw switch
  - Pole pin either connects to Throw pin or is disconnected
  - Come in normally-closed (connected) and normally-open (disconnected)



Disconnected circuits

- When button is pushed, input signal is low
- **What is the value of the input when the button is unpressed?**

Disconnected circuits

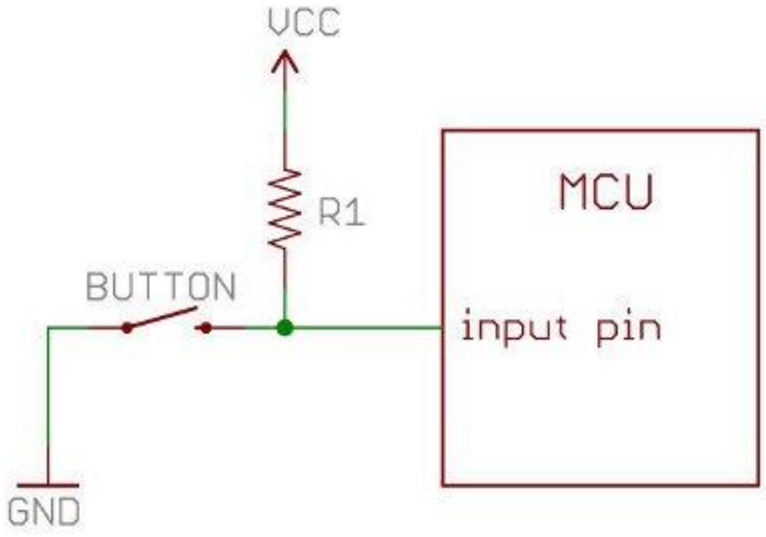
- When button is pushed, input signal is low
- **What is the value of the input when the button is unpressed?**
  - Floating! Could be any voltage
  - Solution: connect weakly to either high or low voltage

Current flows through the “path of least resistance”

- Simplification
  - Works well for the types of circuits we use

- Pull-up resistor
  - When button is open (disconnected), the only path is through the resistor

  - When button is closed (connected) the least resistance path is through the button to Ground

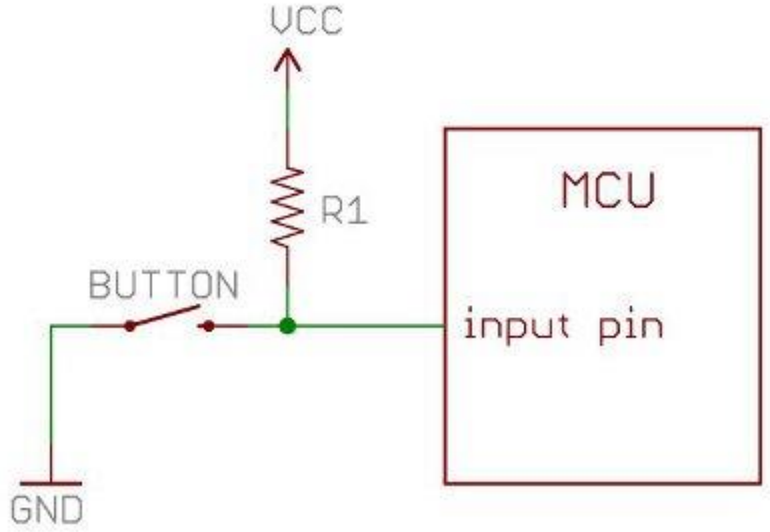


Pull-up resistors and pull-down resistors

- Resistor sets the “default” value of a wire
  - Pull-up connects to VCC
  - Pull-down connects to Ground
  - Usually 10-100 kΩ

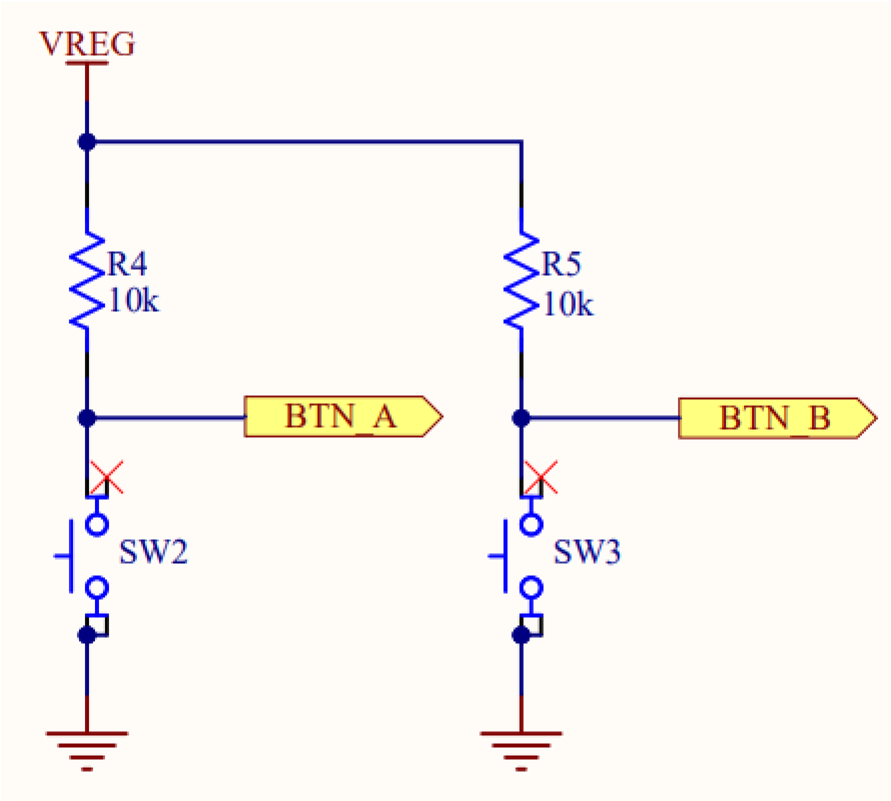
- When button is open (disconnected)
  - Connection through the resistor sets signal

- When button is closed (connected)
  - Signal is directly connected to a voltage source
  - Much lower resistance means that signal dominates



Buttons on the Microbit

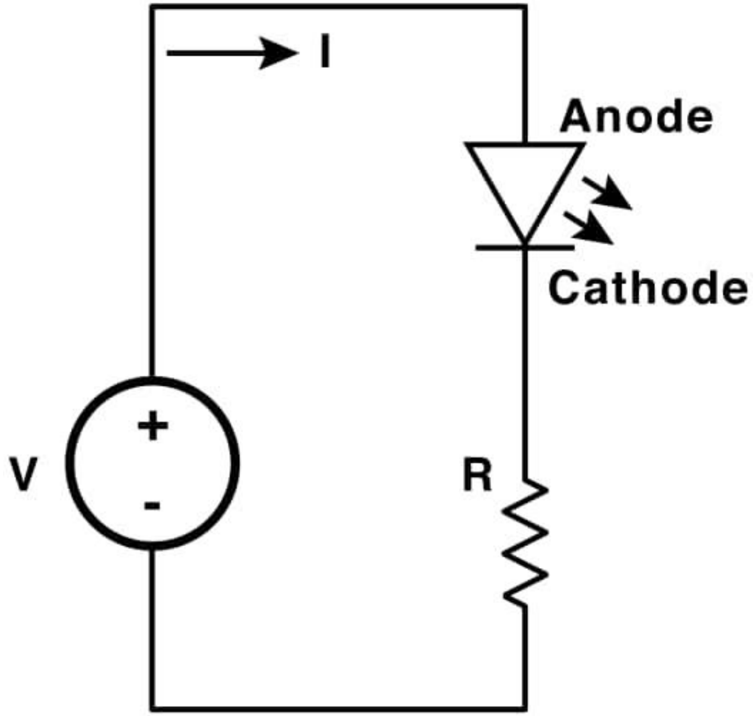
- Normally open buttons
  - Disconnected by default
- Active low signal
  - Activating (pushing) button creates a low signal
- Pull-up resistors
  - Set button signal high by default



LEDs

- Light Emitting Diodes
  - Generate light as current passes through them
  - Various colors available

- Diodes
  - Only allow current to go through one way
  - Not particularly relevant for LEDs
    - Treat as a digital component

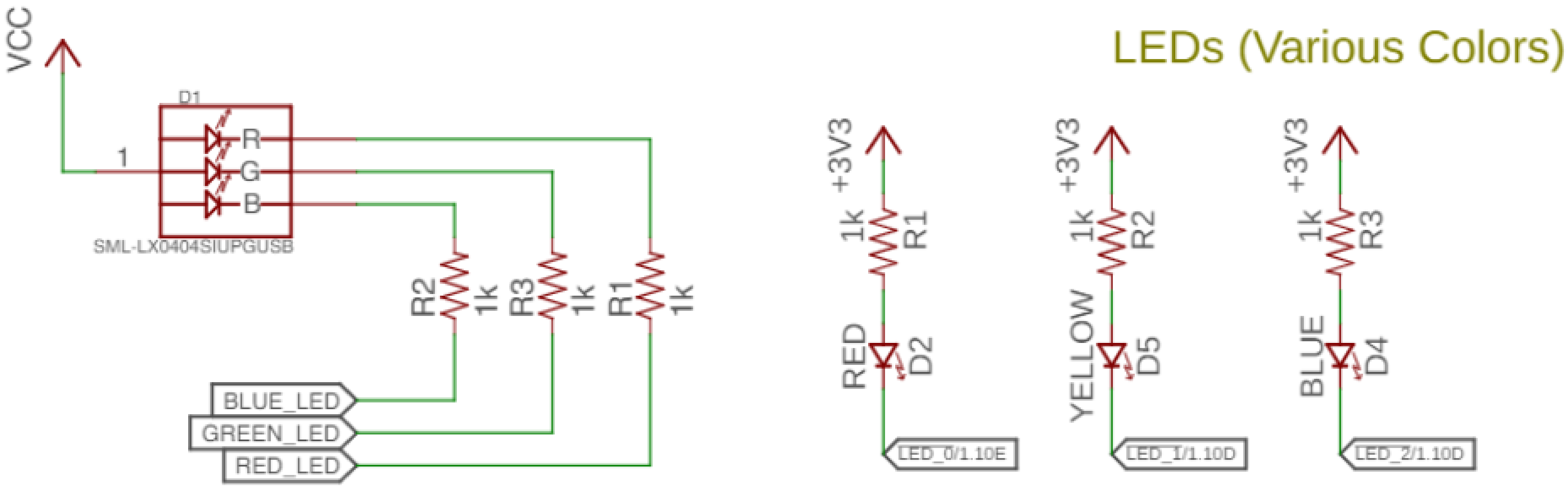


- Connect anode to high voltage and cathode to ground
  - Plus a resistor to limit the total amount of current

https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds

Active state for LEDs

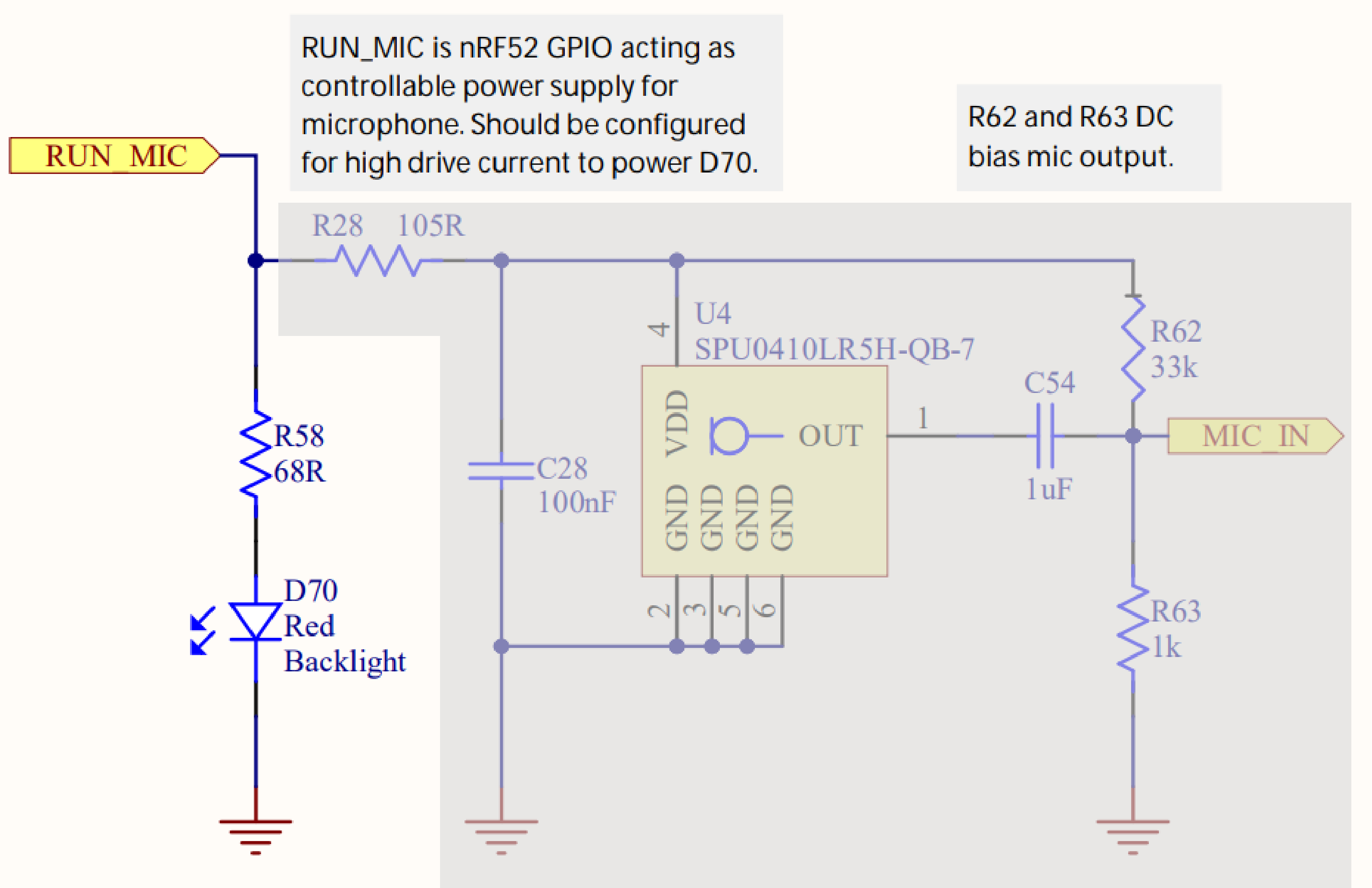
- LEDs can be active high or active low depending on configuration
  - Active high is how people assume they work
  - Active low is often used instead
    - GPIO pins can usually sink more current than they can source



LEDs on the Microbit

- Microphone LED
  - Active high

- Simple to use

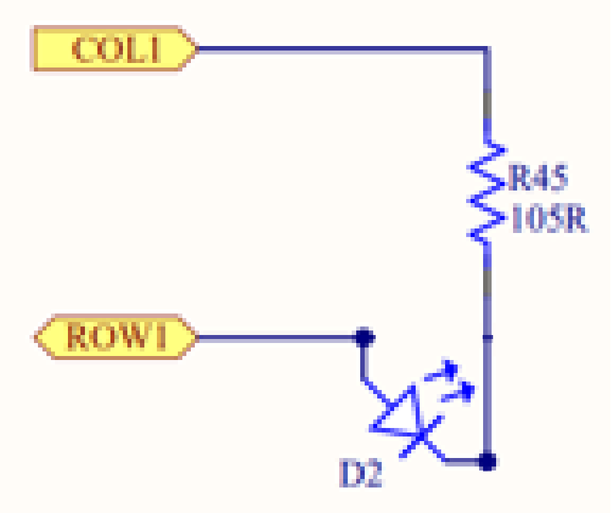


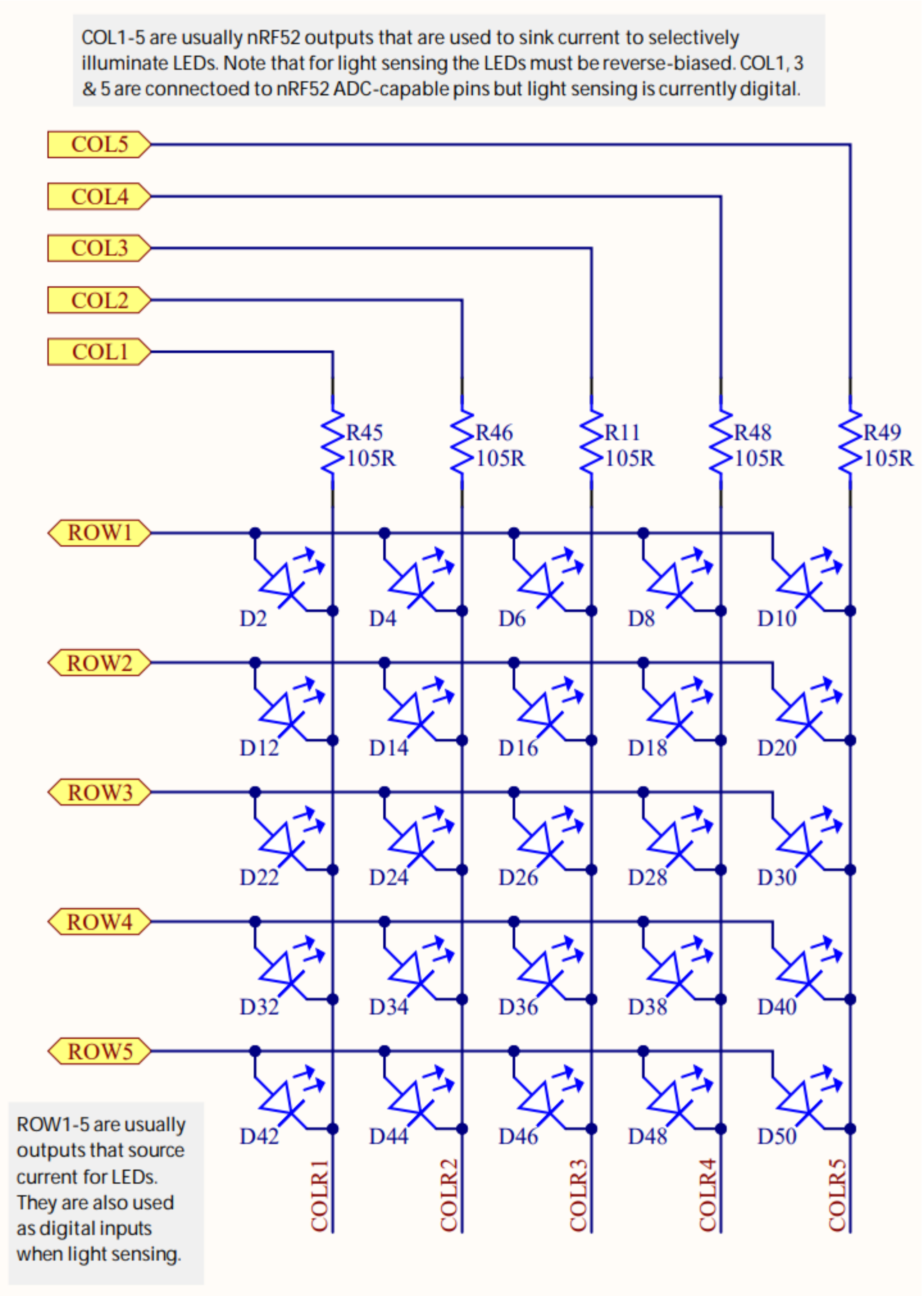
Ignore this other part for now

LEDs on the Microbit

- Use two GPIO pins to control each LED
  - Row high as VDD
  - Column low as Ground

- Remember, connections only exist where there are dots

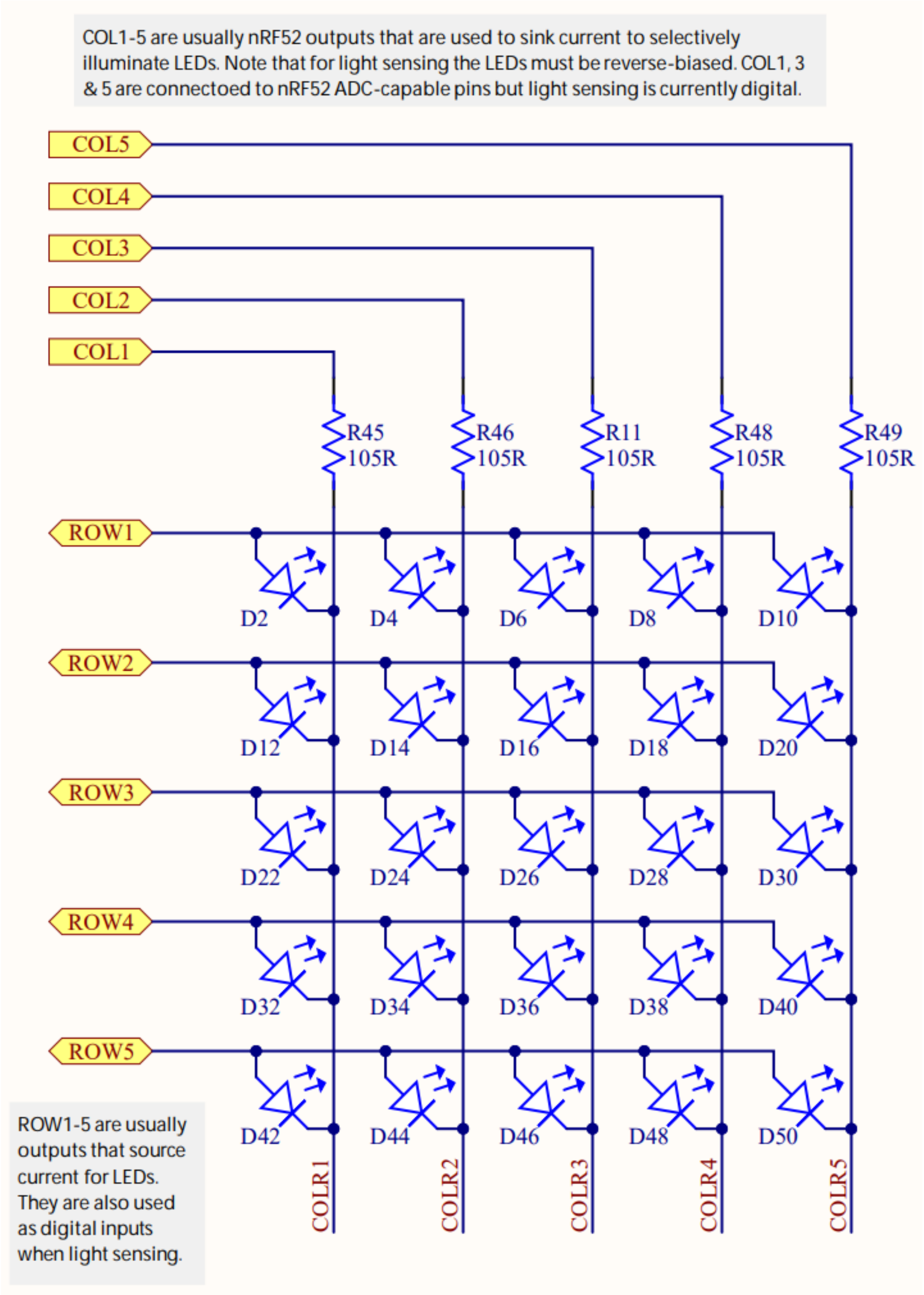




Controlling the LED matrix

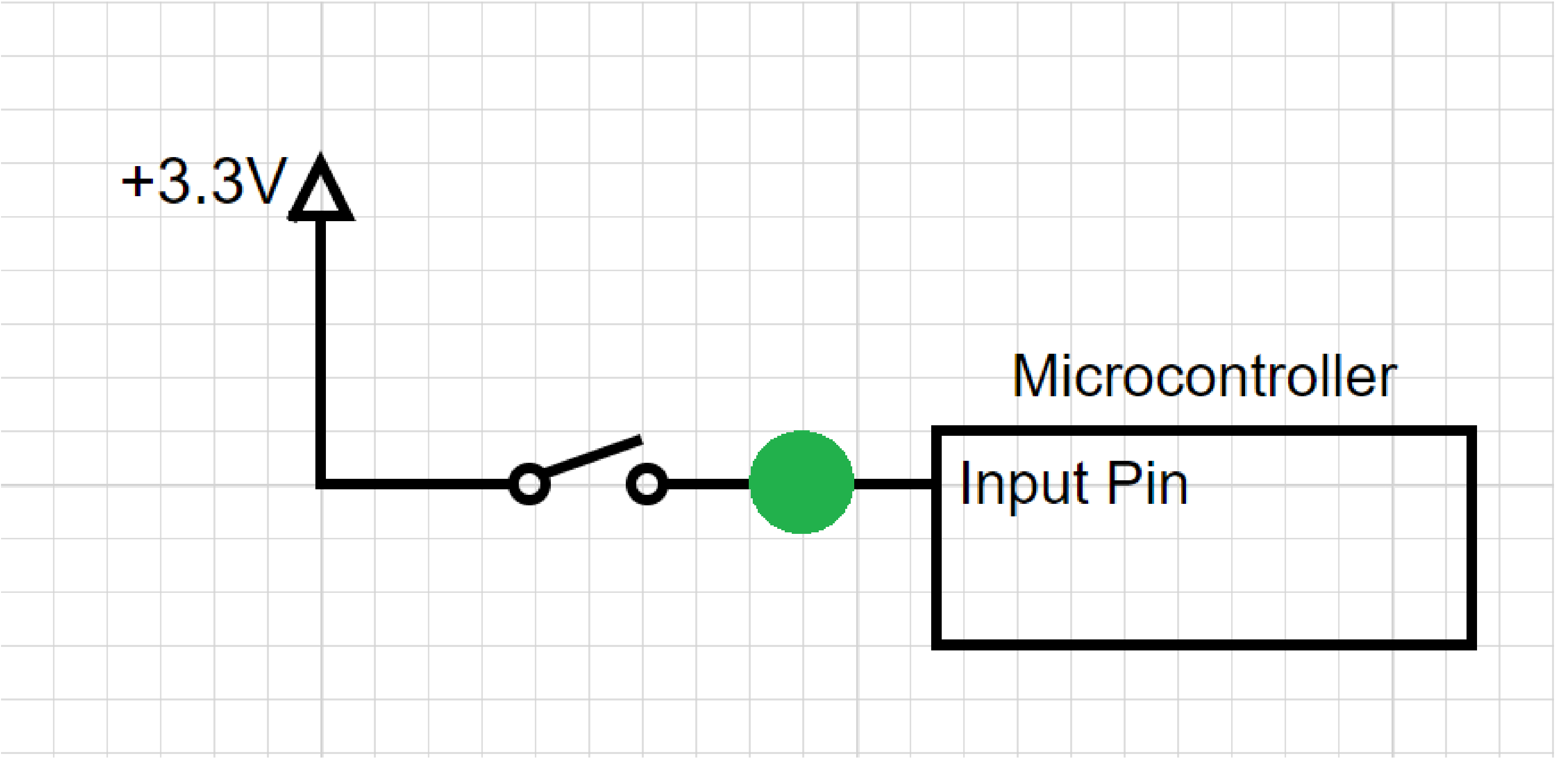
- Cannot individually control all LEDs simultaneously
  - Need to light one row at a time
  - Iterate rows quickly to make them appear on all the time

- We'll have a lab on these later
  - Combine GPIO and timers



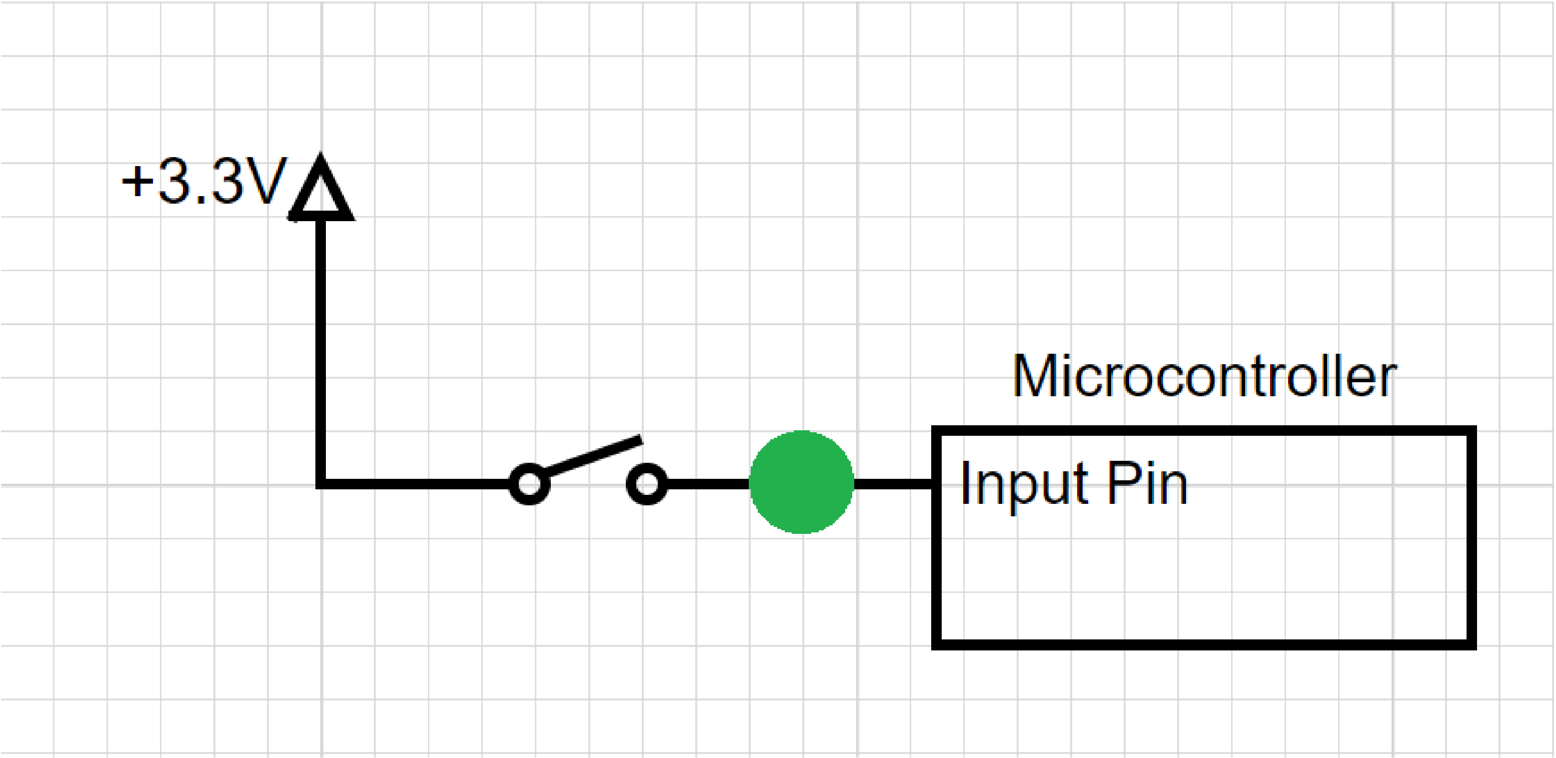
Break + Question

- Should the spot in green have?
  - Pull-up Resistor
  - Pull-down Resistor
  - Either
  - Neither



Break + Question

- Should the spot in green have?
  - Pull-up Resistor
  - **Pull-down Resistor** (needs to pull input low by default)
  - Either
  - Neither



Outline

- DMA
- Digital Circuits
- **GPIOTE**
- Energy

Handling interrupts from GPIO

- Separate peripheral, GPIOTE (GPIO Task/Event)
  - Manages up to 8 individual pins
    - Can read inputs and trigger interrupts
    - Can also connect outputs from events on other peripherals (PPI)
  - Can trigger interrupts for a "Port event" as well
    - Software checks which pin(s) caused the event to occur
    - Very low power operation (works with system clocks off)
- Unclear why this is a separate peripheral
  - Presumably too complicated/expensive to have 42 of them

Configuring individual input interrupts

- Pick an available GPIOTE channel (0-7)

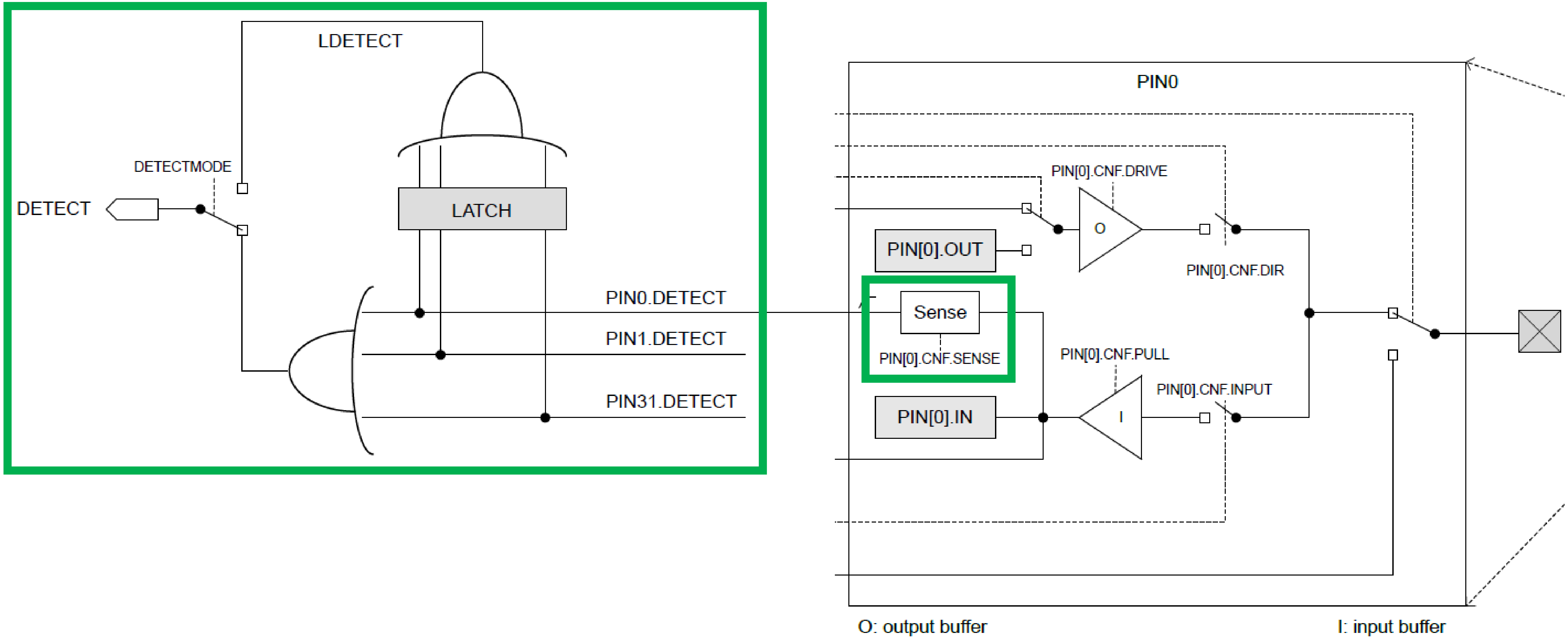
- Configure it
  - Port and Pin number
  - Task (output), Event (input), or Disabled
  - Polarity for input events
    - Low-to-high
    - High-to-low
    - Toggle (both directions)

- Enable interrupts for channel in GPIOTE (and in NVIC!)

- Clear event in interrupt handler
  - Doesn’t happen automatically

Sensing port events

- Uses the "Detect" signal. Generated from pin Sense configuration



Configuring port input interrupts

- Configure the Sense for each pin
  - High or Low
  - Allows different pins to have different “active” states

- Select detect mode
  - Direct connection to pins
  - Latched version (saved even if pin later changes back)

- Enable interrupts for port in GPIOTE (and in NVIC!)

- Clear event in interrupt handler and value in Latch register
  - Doesn’t happen automatically

Outline

- DMA
- Digital Circuits
- GPIOTE
- **Energy**

Ohm's Law

$V = I \times R$

- Volts = Current times Resistance

$P = I \times V$

- Power = Current times Voltage

Ohms Law Formulas				
Known Values	Resistance (R)	Current (I)	Voltage (V)	Power (P)
Current & Resistance	---	---	$V = I \times R$	$P = I^2 \times R$
Voltage & Current	$R = \frac{V}{I}$	---	---	$P = V \times I$
Power & Current	$R = \frac{P}{I^2}$	---	$V = \frac{P}{I}$	---
Voltage & Resistance	---	$I = \frac{V}{R}$	---	$P = \frac{V^2}{R}$
Power & Resistance	---	$I = \sqrt{\frac{P}{R}}$	$V = \sqrt{P \times R}$	---
Voltage & Power	$R = \frac{V^2}{P}$	$I = \frac{P}{V}$	---	---

- These two equations govern most of the circuit math we'll need in this course
  - Work with resistive circuits

Thinking about energy

- Batteries often list energy in mA*h (milliamp – hours)
  - Coin cell battery: 3v at 220 mAh
  - 2x AA battery: 3v at 2000 mAh
  - iPhone 11 battery: 3.7v at 3000 mAh

- nRF52833 active current: 5.6 mA (at 3v)
  - Coin cell: 40 hours -> ~2 days
  - 2x AA: 360 hours -> ~15 days
  - iPhone 11: 535 hours -> ~22 days

- So how does any of this work???

Microcontroller sleep modes

- Sleep mode
  - Processor stops running
  - Most peripherals are disabled
  - Continues until an interrupt occurs and wakes the microcontroller
    - Usually a timer or GPIO input

- nRF52833 sleep mode current: 1.8 μA (GPIO port event only)
  - Coin cell: 122222 hours -> ~5000 days -> ~14 years

- Low-power systems shoot for less than 1% duty cycle
  - Average current of ~100 μA or less
  - Warning: other stuff on the board counts!!
    - LEDs are 1-10 mA each… Power is not a concern of the Microbit

Outline

- DMA
- Digital Circuits
- GPIOTE
- Energy