Lecture 05
Digital Circuits

CE346 – Microprocessor System Design

Branden Ghena – Spring 2021

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

Northwestern

Today's Goals

- Understand the basics of digital circuitry
  - Enough to be able to interact with the Microbit

- Explore how the Microbit controls digital inputs and outputs

Outline

- **Digital circuits**

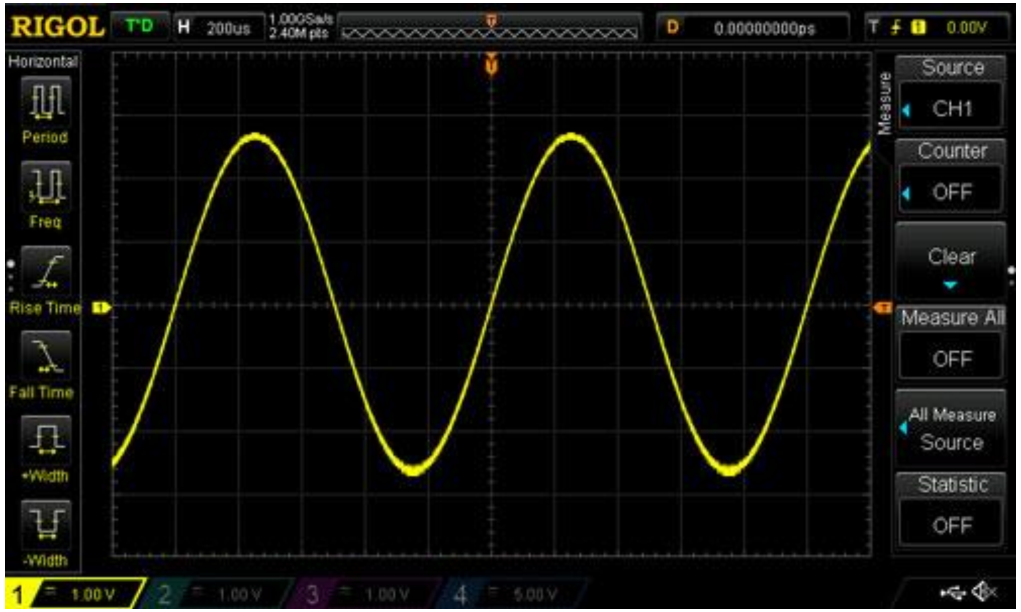
- Controlling digital signals
  - GPIO
  - GPIOTE

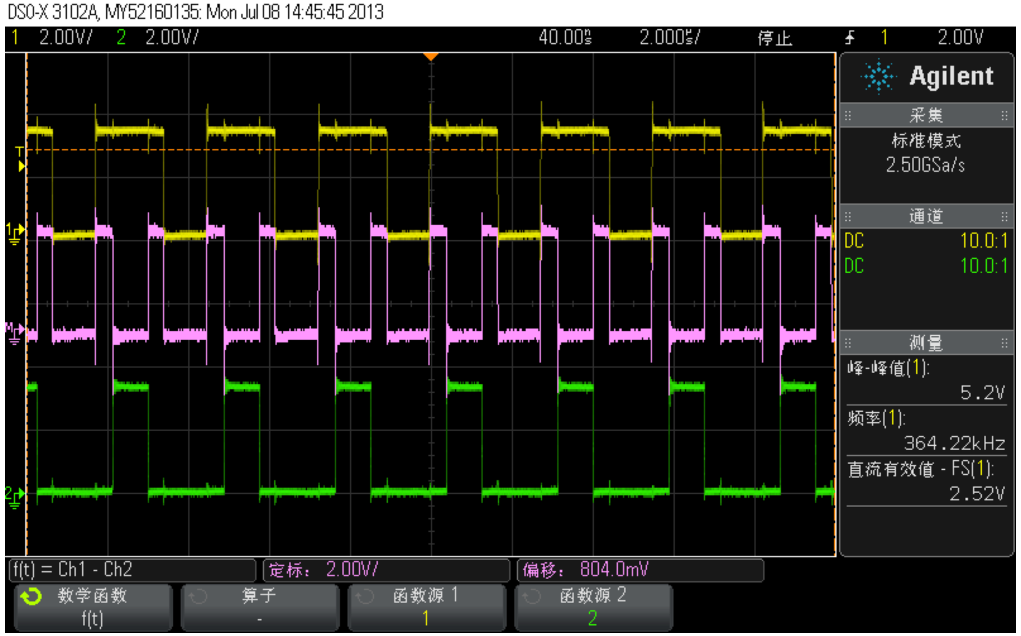
Digital signals

- 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

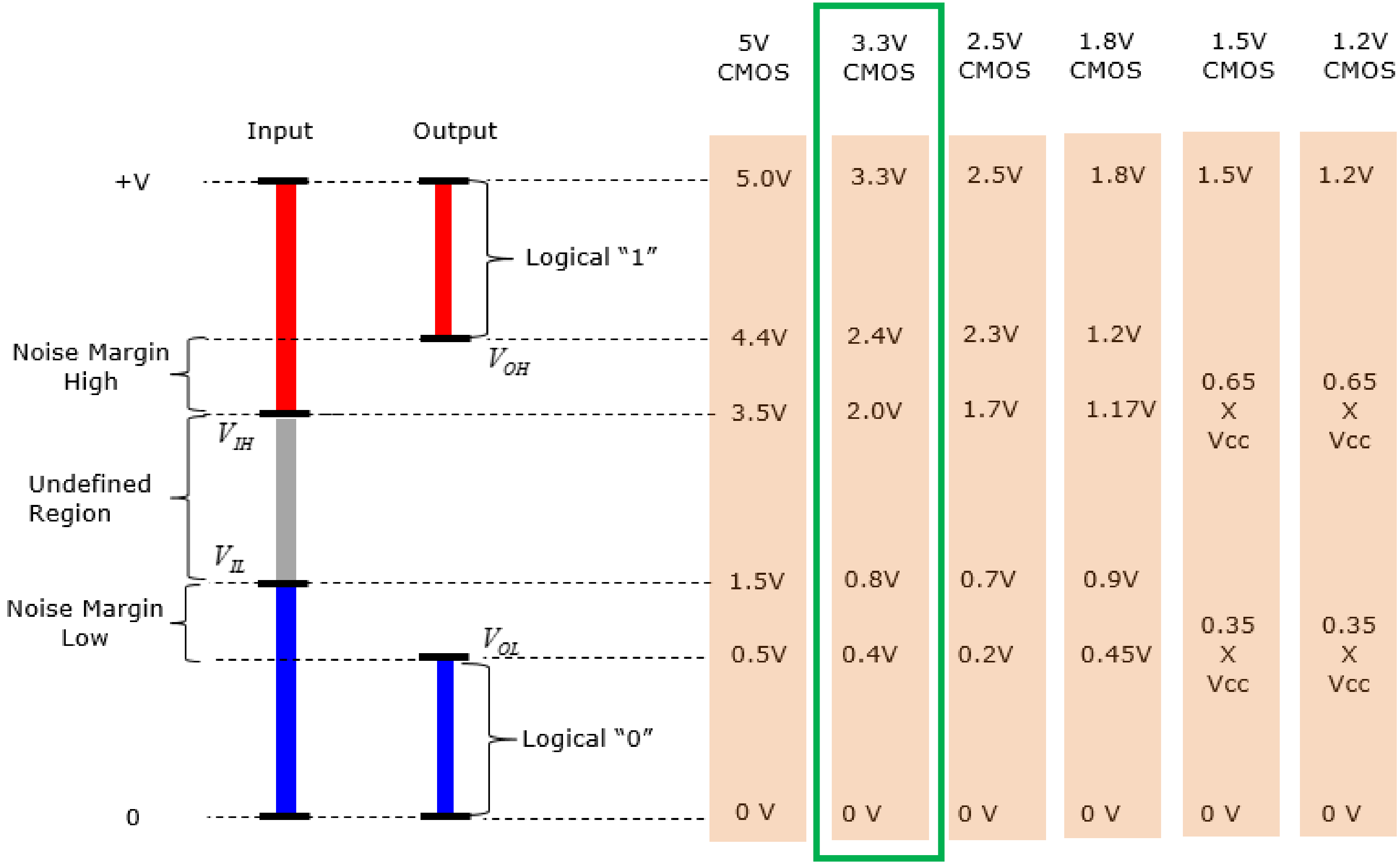
  - No math for voltage level
    - Either high or low





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



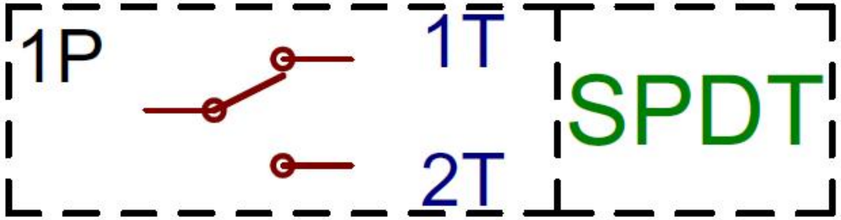
http://www.sharetechnote.com/html/Electronics_CMOS.html

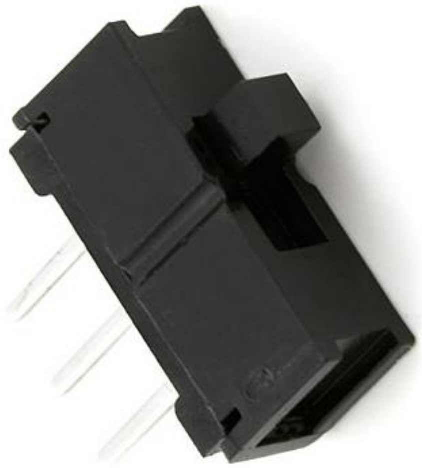
Digital circuits

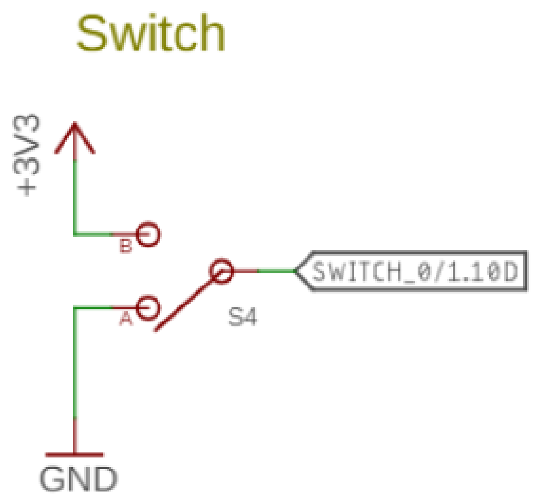
- Connecting components together with digital signals
  - Mostly ICs
  - Also buttons/switches and LEDs
- Way simpler than analog circuits
  - Mostly connecting boxes with wires
  - Plus a few resistors here and there
- An abstraction
  - Not sufficient for fully understanding electronics behavior, but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

Switches

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





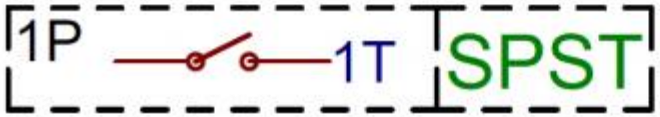


- For controlling microcontrollers
  - Often connect outer pins to VCC and Ground respectively
  - Input then goes High or Low depending on switch state

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

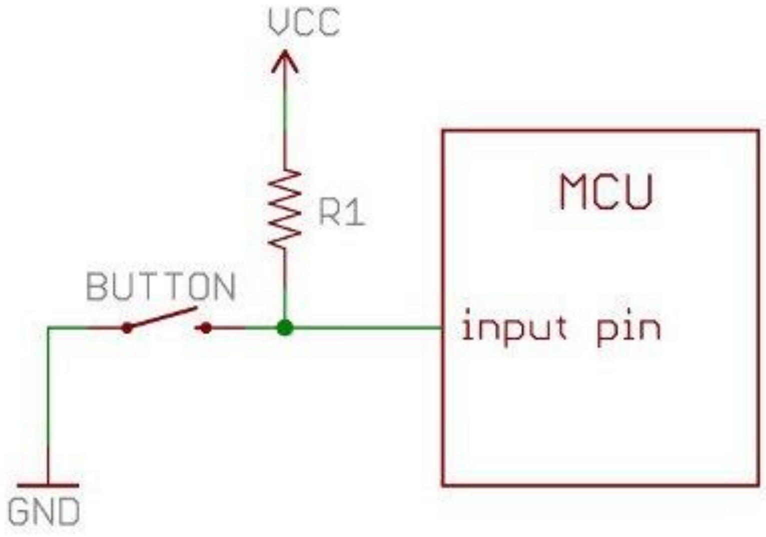
- When button is pushed, input signal is low
- **What is the value of the input when the button is unpressed?**
  - Floating! Could be any voltage

  - Solution: connect weakly to either high or low voltage

Current flows through the "path of least resistance"

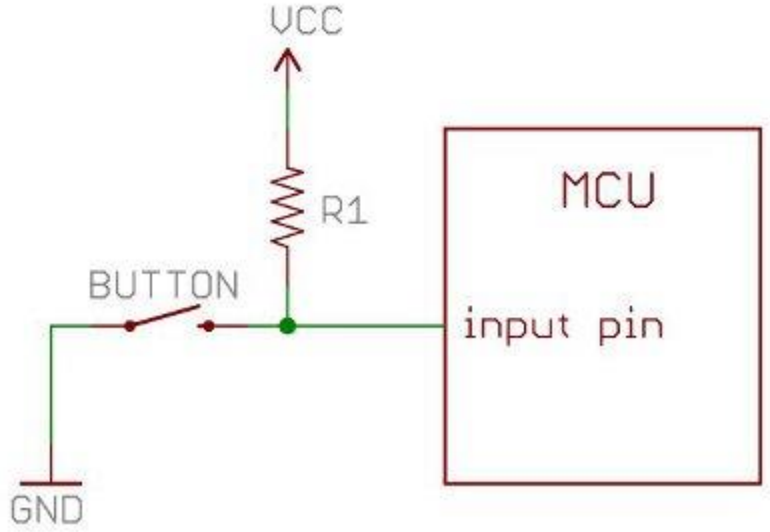
- Simplification
  - Works well for the types of circuits we use

- 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

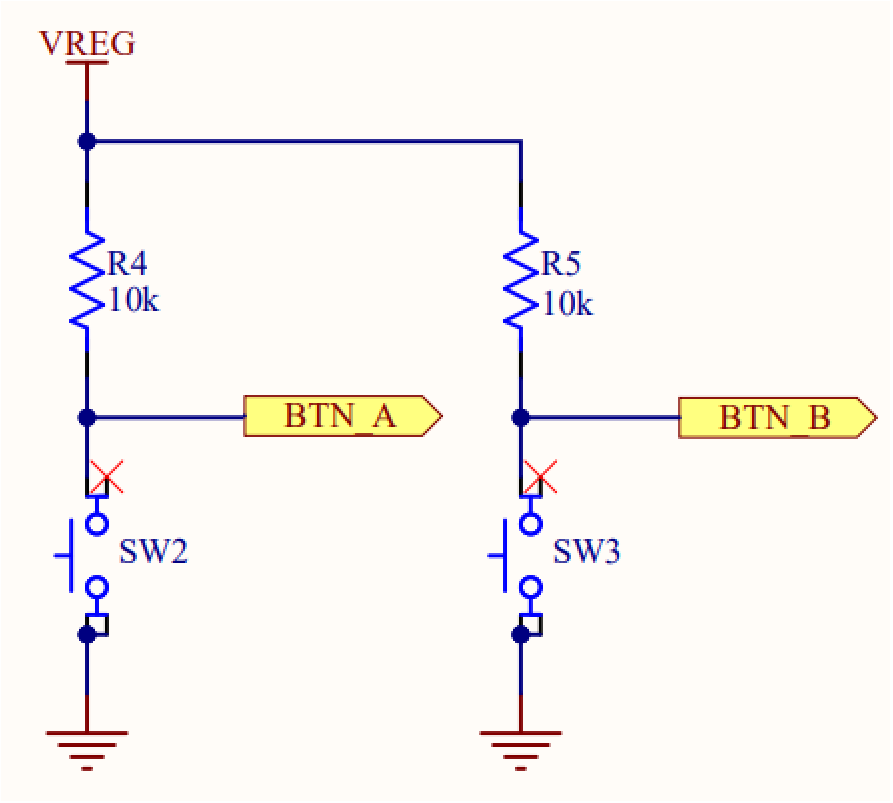


- When button is open (disconnected)
  - Connection through the resistor sets signal

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

Buttons on the Microbit

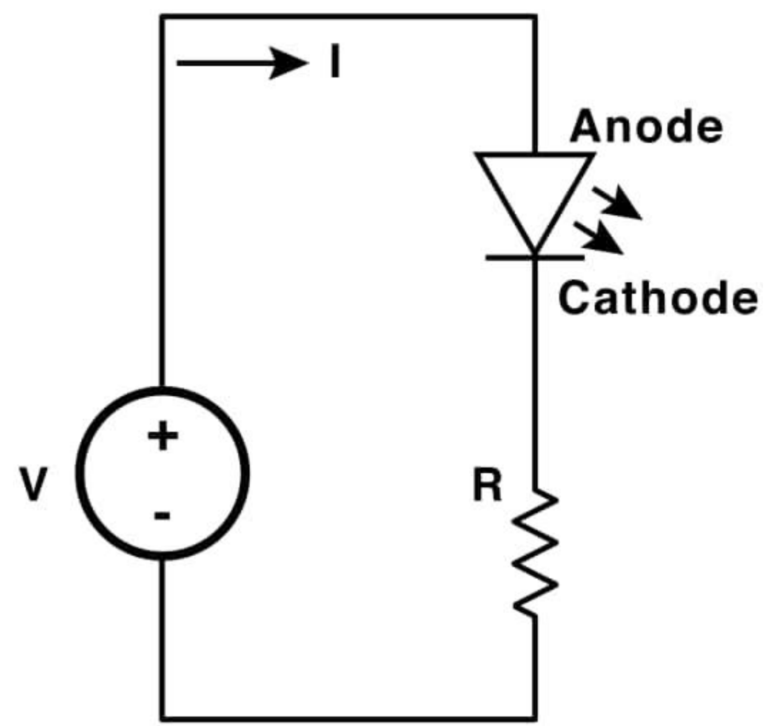
- Normally open buttons
  - Disconnected by default
- Active low signal
  - Activating (pushing) button creates a low signal
- Pull-up resistors
  - Set button signal high by default



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

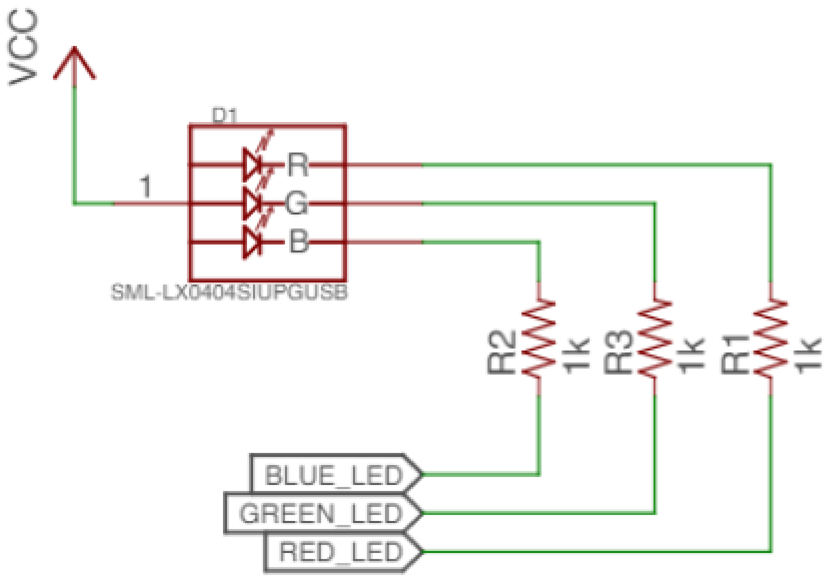


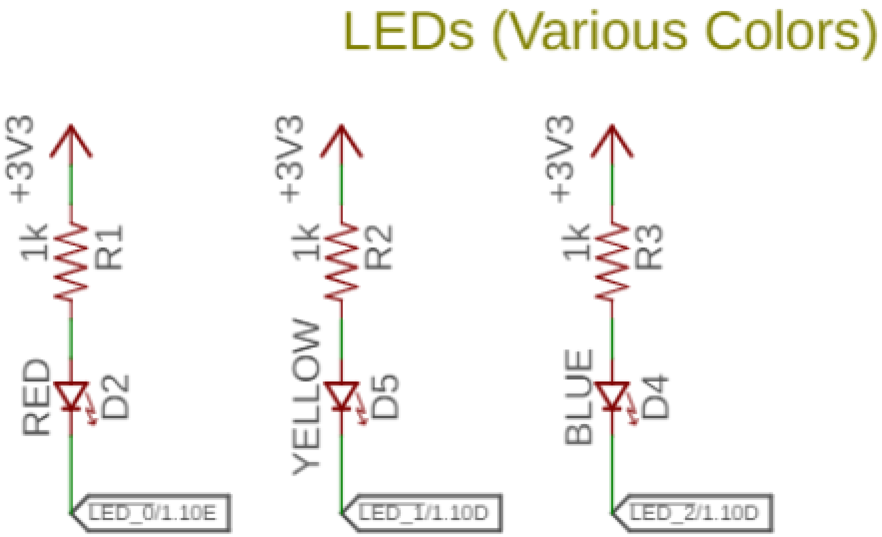
- Connect anode to high voltage and cathode to ground
  - Plus a resistor to limit the total amount of current

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

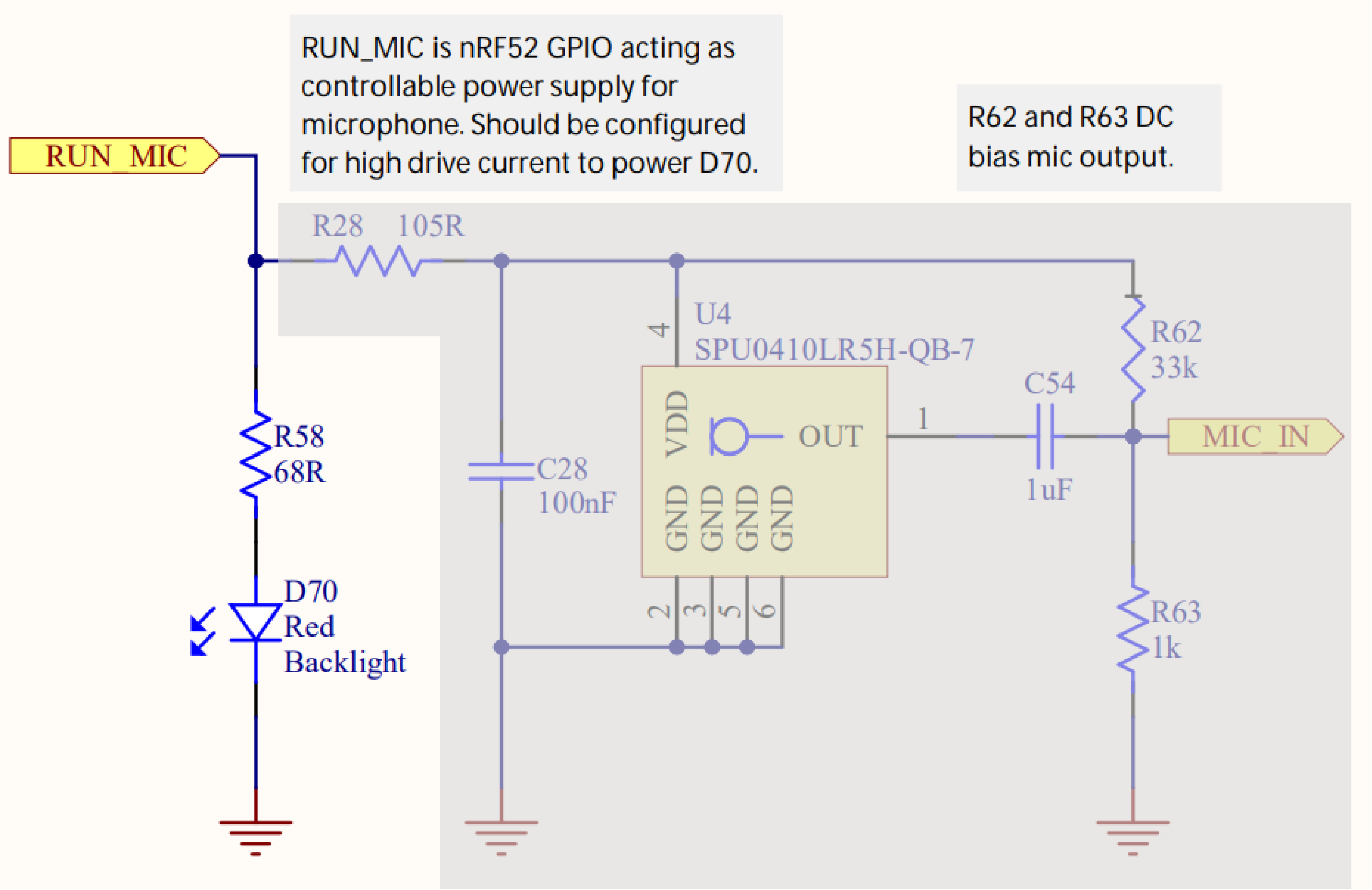




LEDs on the Microbit

- Microphone LED
  - Active high

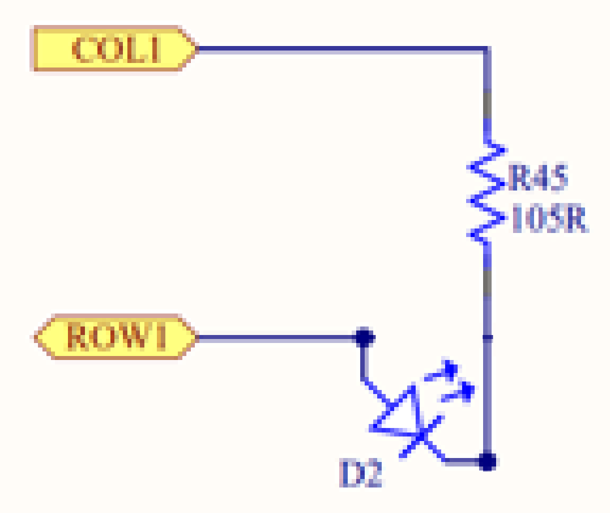
- Simple to use

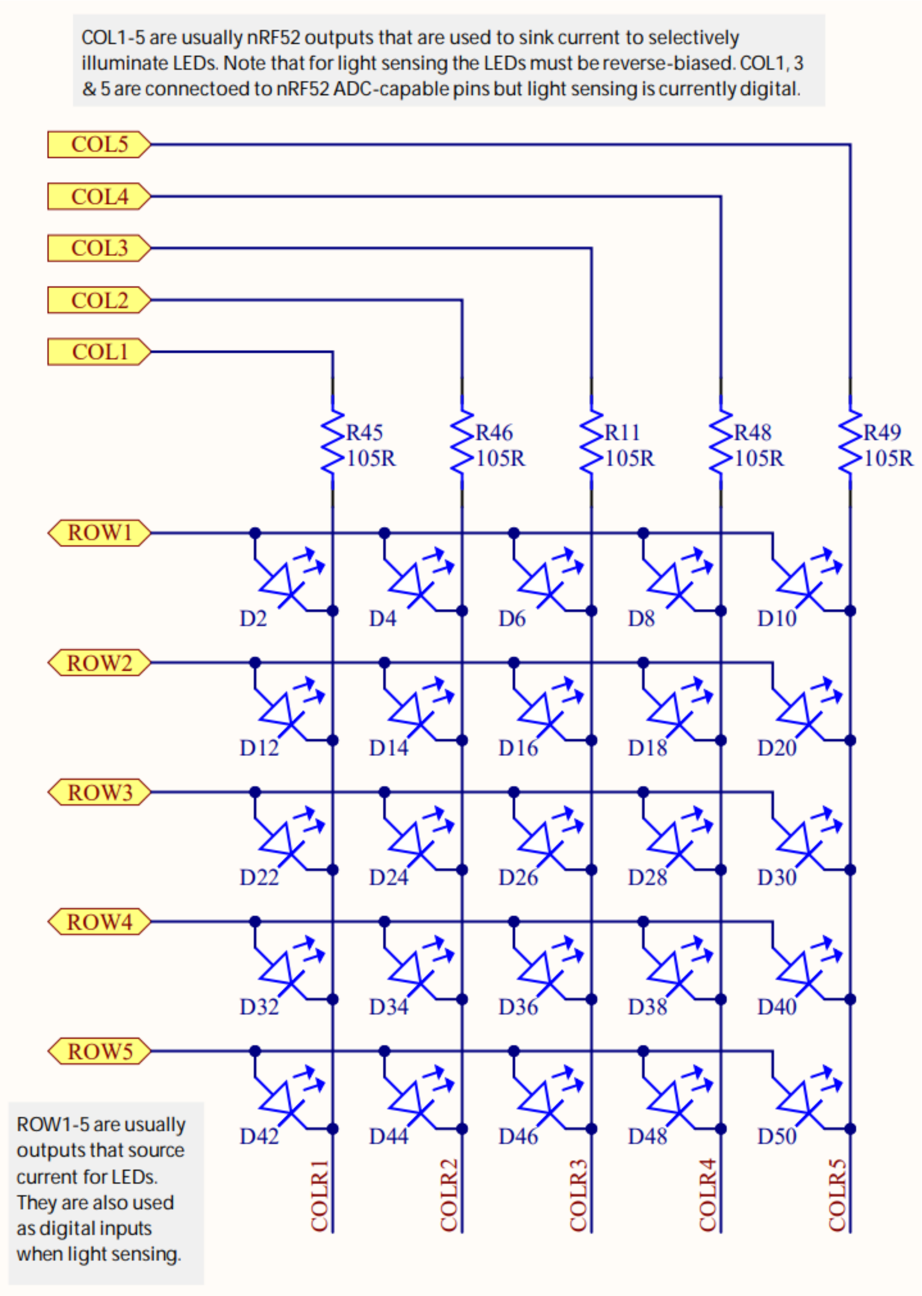


Ignore this other part for now

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

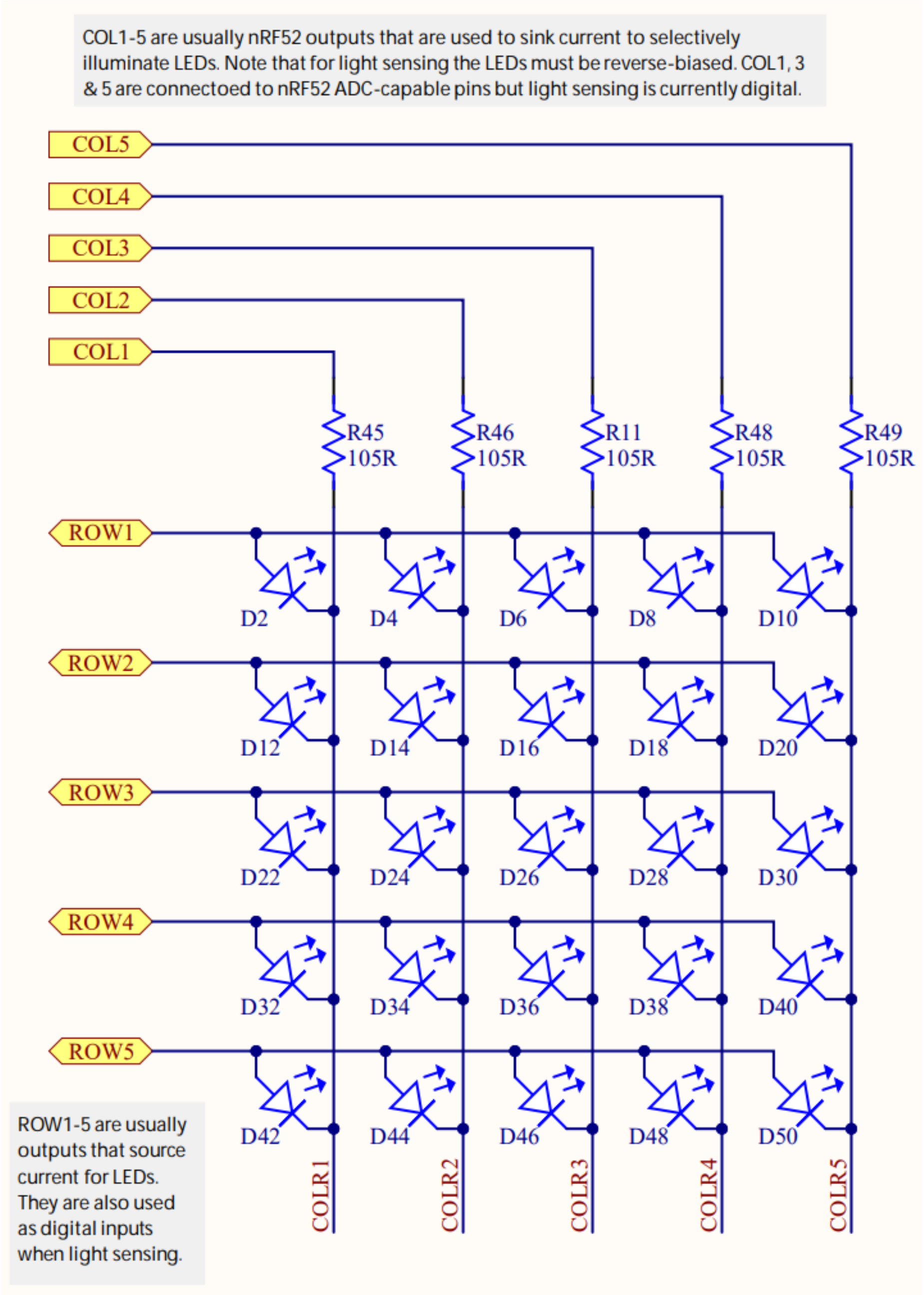




Controlling the LED matrix

- 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



Outline

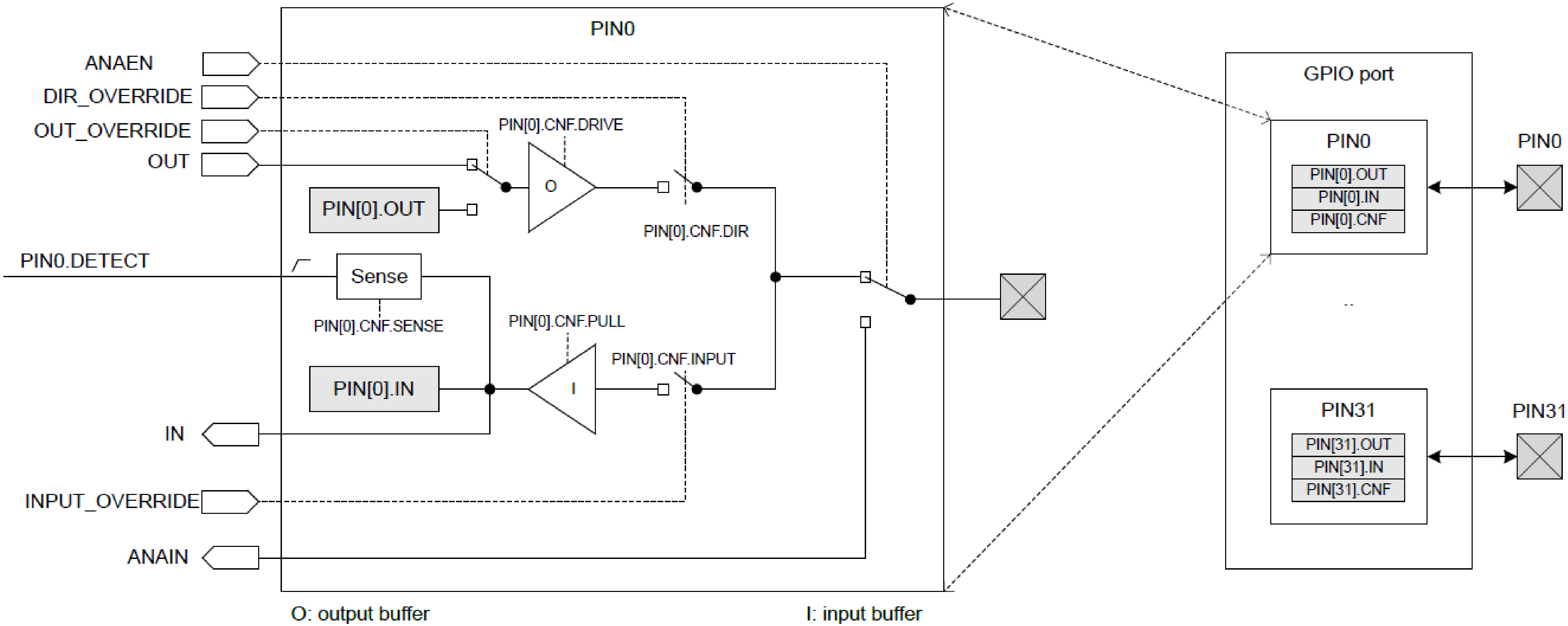
- Digital circuits
- **Controlling digital signals**
  - **GPIO**
  - GPIOTE

General Purpose Input/Output (GPIO)

- Read/write from/to external pins on the microcontroller
  - Two possible values: high (1) or low (0)

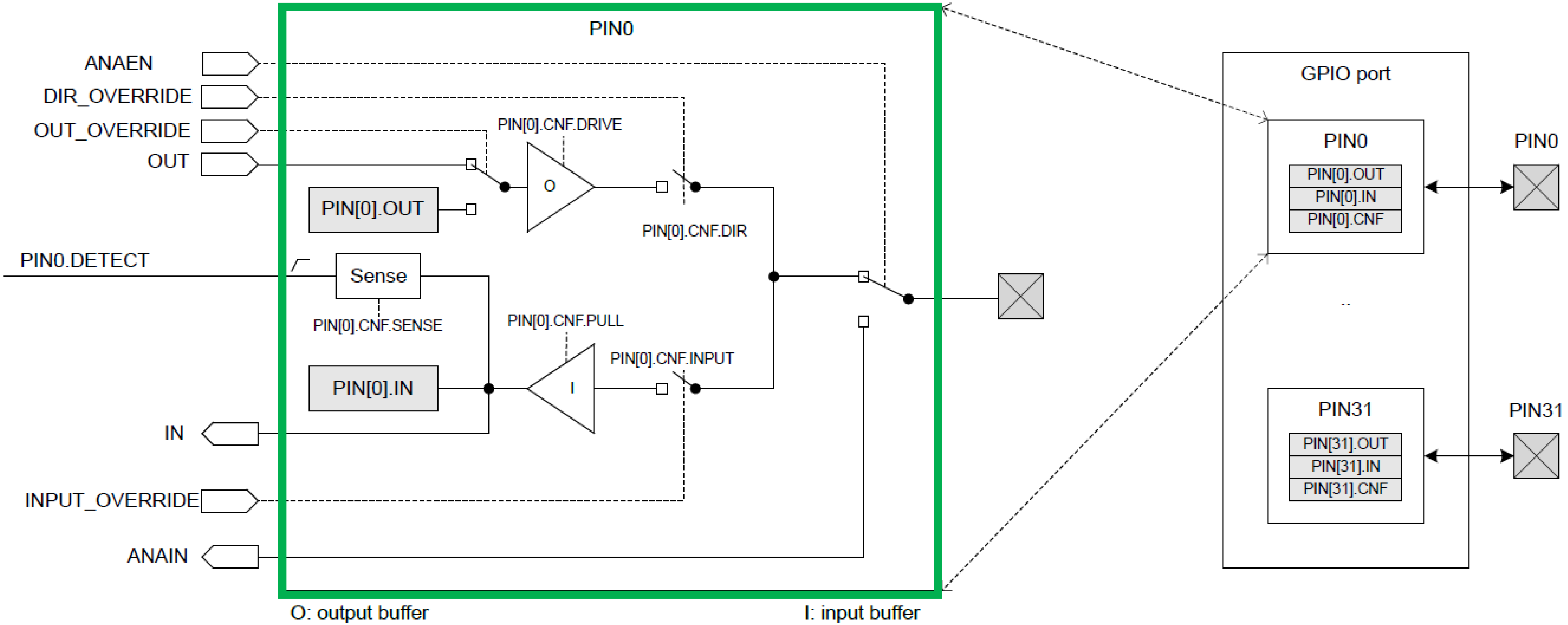
- Basic unit of operation for microcontrollers
  - Allows them to interact with buttons and LEDs
  - Every microcontroller has GPIO

GPIO on nRF52833



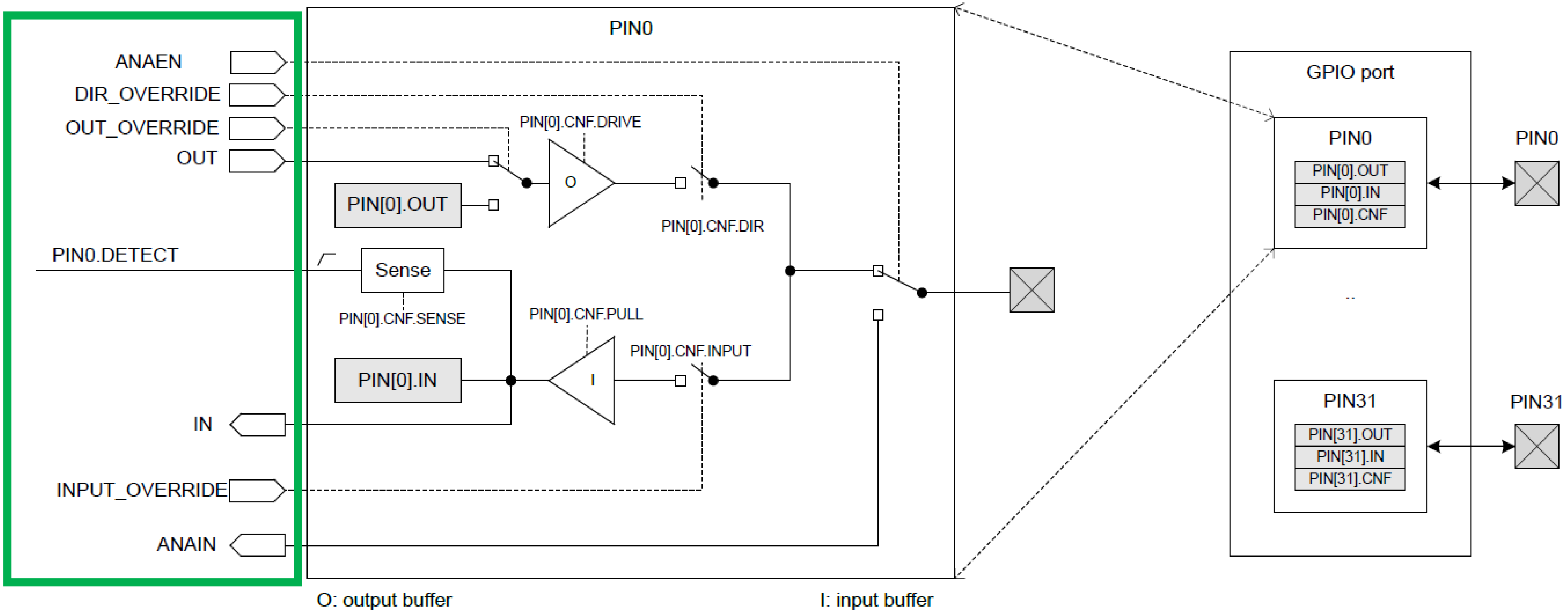
GPIO on nRF52833

Abstract model of the pin.
This isn't really how the hardware is implemented. But it's a reasonable model for users.



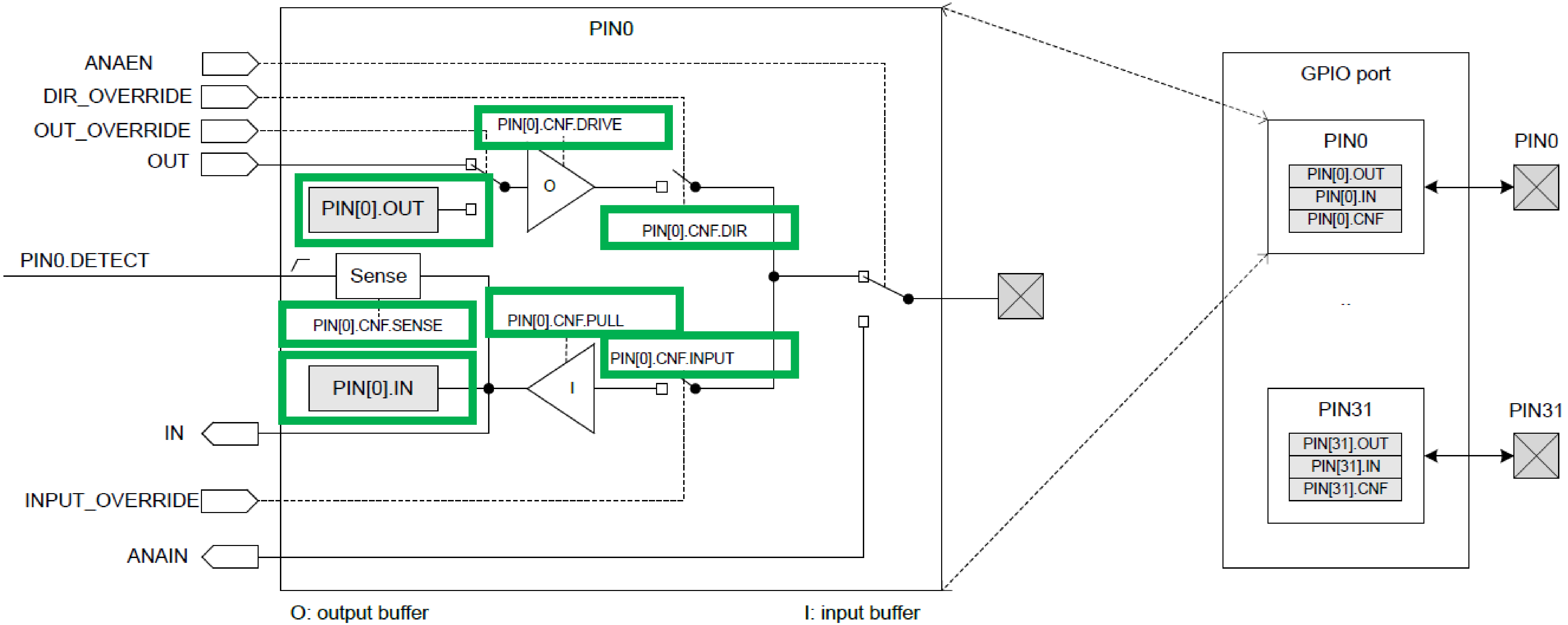
GPIO on nRF52833

Inputs and outputs to/from the peripheral.
GPIO could be controlled by other peripherals. Controlling a pin in use by other peripherals is bad.



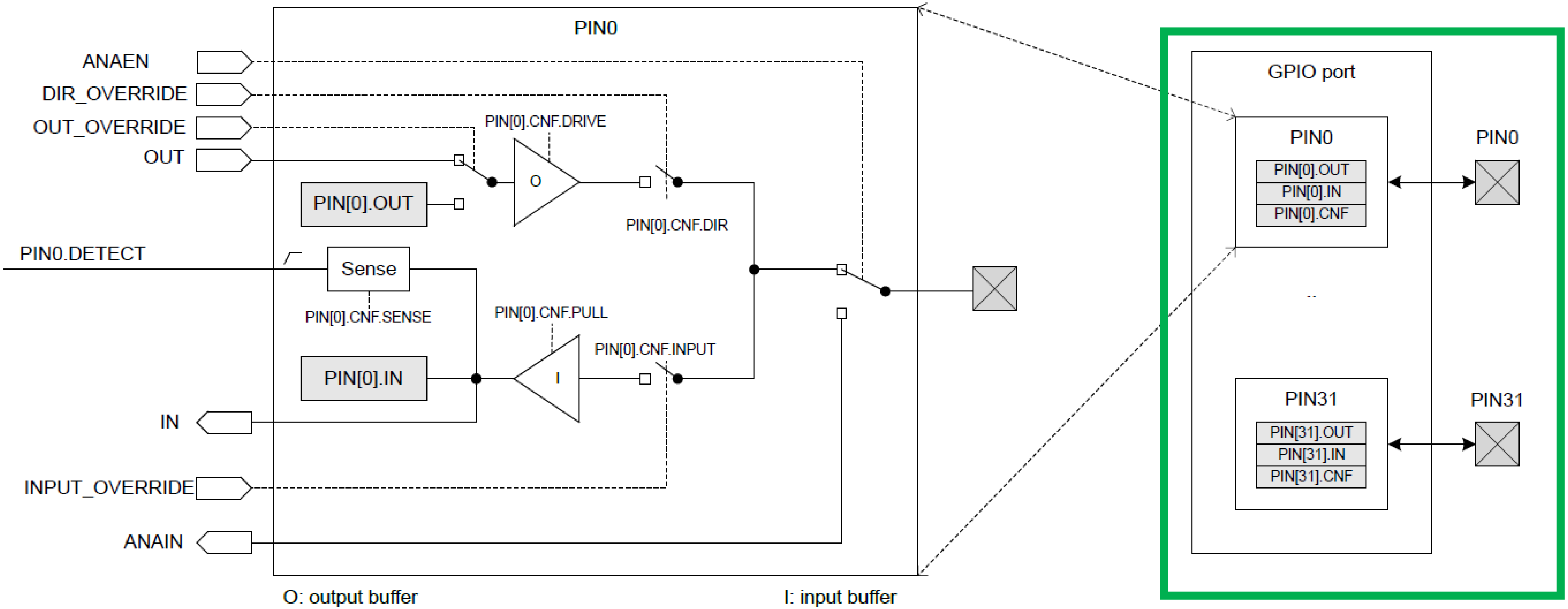
GPIO on nRF52833

Registers within the GPIO peripheral.
Configure various things about setup.



GPIO on nRF52833

Peripheral contents are duplicated for each output pin.
Each pin has its own registers (or portions thereof).



Multiple ports

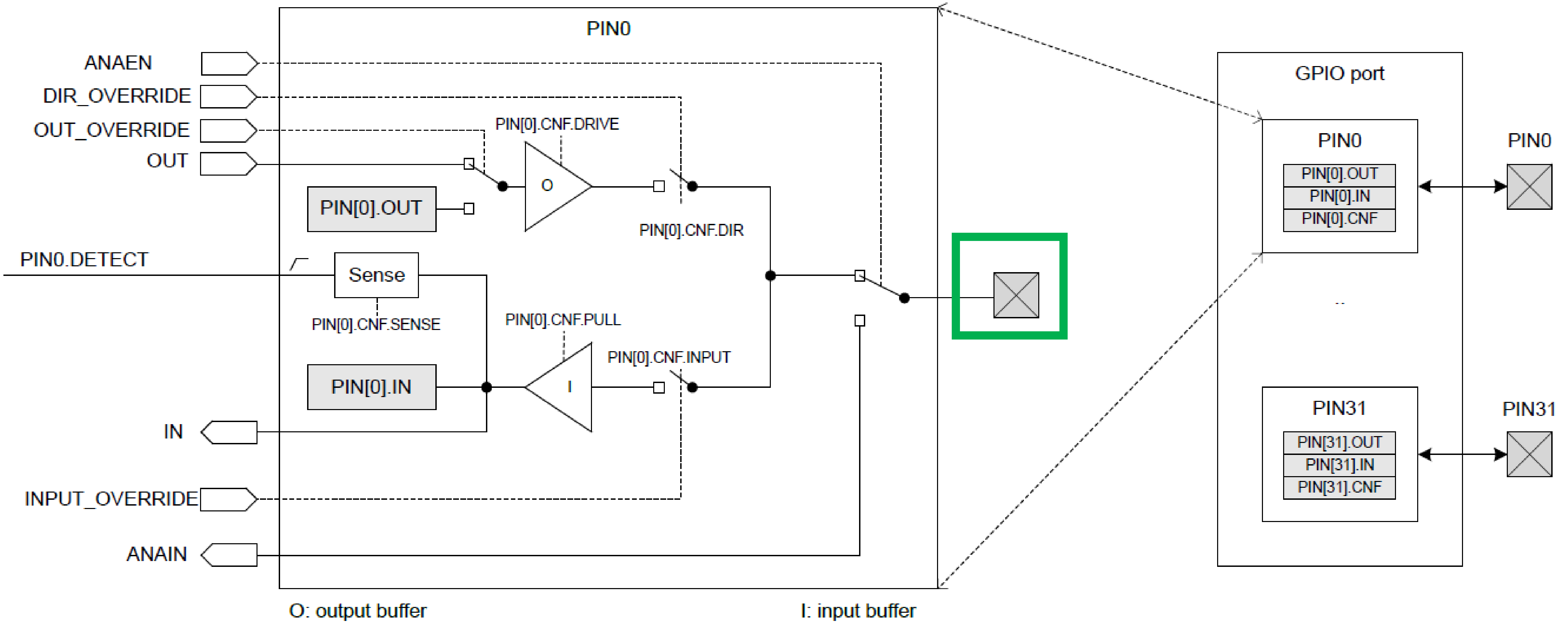
- nRF52833 has up to 42 I/O pins
  - But only 32 can fit in a single word
  - Splits them into two "ports"

Base address	Peripheral	Instance	Description	Configuration	
0x50000000	GPIO	GPIO	General purpose input and output		Deprecated
0x50000000	GPIO	P0	General purpose input and output, port 0	P0.00 to P0.31 implemented	
0x50000300	GPIO	P1	General purpose input and output, port 1	P1.00 to P1.09 implemented	

- Pins are named based on port
  - P0.14 – Button A, P0.23 – Button B
  - P1.04 – LED column 4

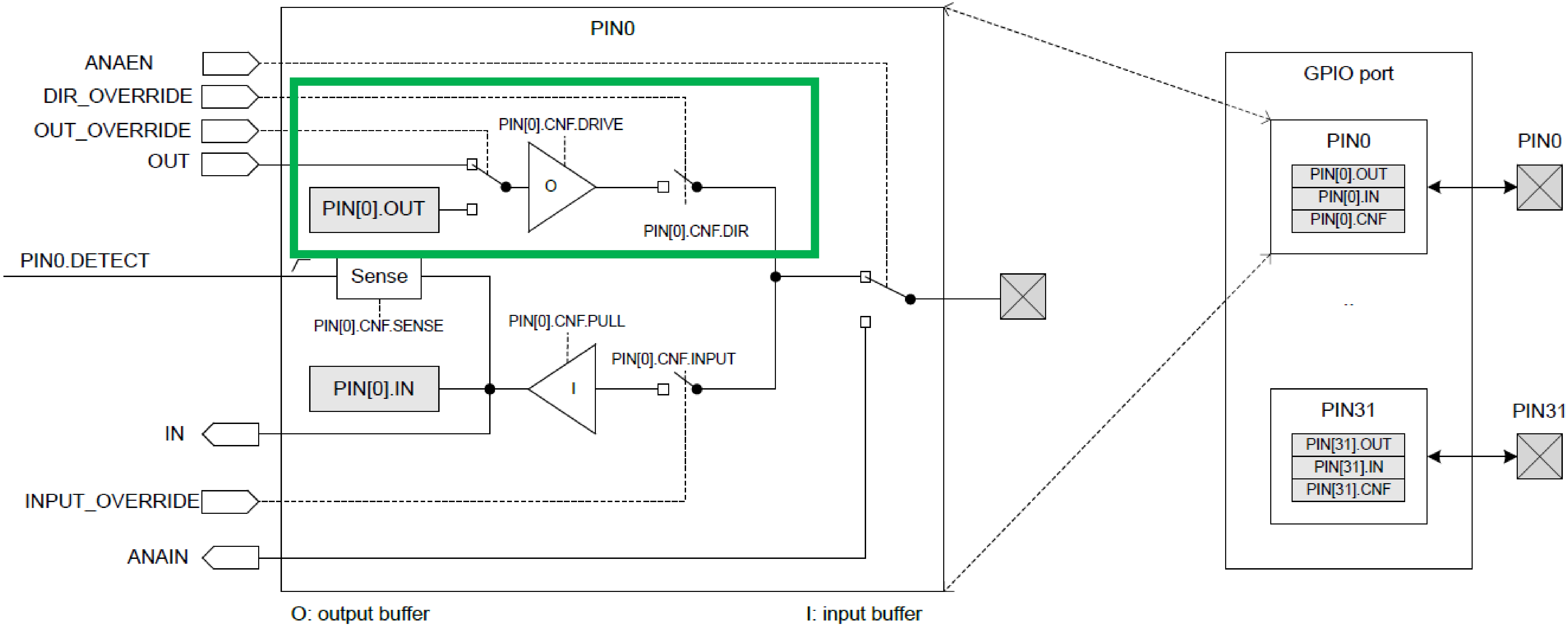
GPIO on nRF52833

External pin on the microcontroller



GPIO on nRF52833

Output chain. Signal comes from OUT register, through output buffer, to external pin.



GPIO Output

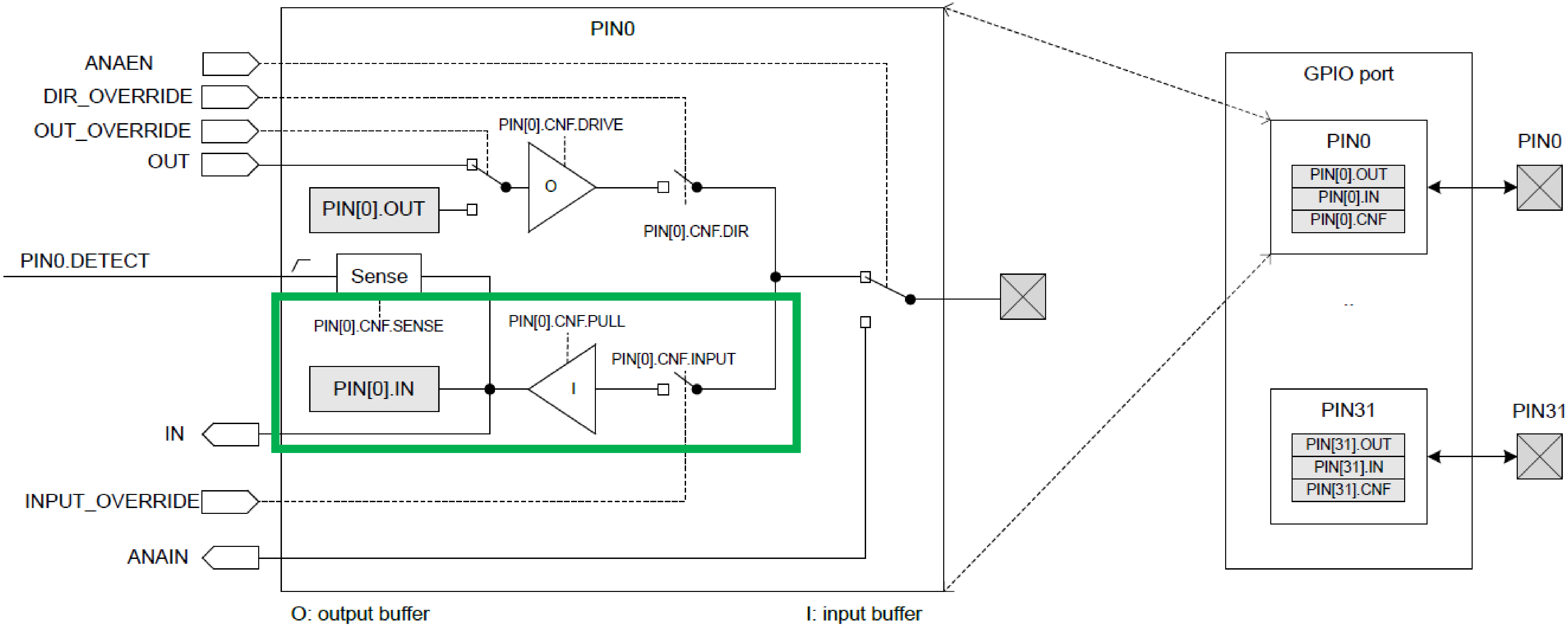
- Outputs a high or low signal

- Output configurations
  - High drive output (either for high, low, or both)
    - Sources or sinks additional current
      - For powering external devices
    - Normal drive: ~2 mA
    - High drive: ~10 mA

  - Disconnect (a.k.a. High Impedance or High-Z)
    - Wired-OR or Wired-AND scenarios

GPIO on nRF52833

Input chain. Signal goes from pin, through input buffer, to IN register.



GPIO Input

- Reads in a signal as either high or low

- Input Configurations
  - Input buffer connect/disconnect
    - Allows the pin to be disabled if not being read from

  - Pull
    - Disabled, Pulldown, Pullup
    - Connects an internal pull up/down resistor (~13 kΩ)
    - Sets default value of input

Electrical specifications

- High voltage range: 0.7*VDD to VDD (~2.3 volts)
- Low voltage range: Ground to 0.3*VDD (~1 volt)

- GPIO are extremely fast
  - Transition time is <25 ns
  - Connected directly to memory bus for faster interactions

  - This allows complicated signal patterns to be replicated in software
    - If they aren't implemented as a hardware peripheral
    - Known as bit-banging

Set/Clear registers

Register	Offset	Description
OUT	0x504	Write GPIO port
OUTSET	0x508	Set individual bits in GPIO port
OUTCLR	0x50C	Clear individual bits in GPIO port
IN	0x510	Read GPIO port
DIR	0x514	Direction of GPIO pins
DIRSET	0x518	DIR set register
DIRCLR	0x51C	DIR clear register

- OUT works traditionally: write a 1 for high, 0 for low
- OUTSET write a 1 to set that pin (high) zero has no effect
- OUTCLR write a 1 to clear that pin (low) zero has no effect
  - Lets you modify a pin without modifying the others (or reading first)

Outline

- Digital circuits
- **Controlling digital signals**
  - GPIO
  - **GPIOTE**

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)

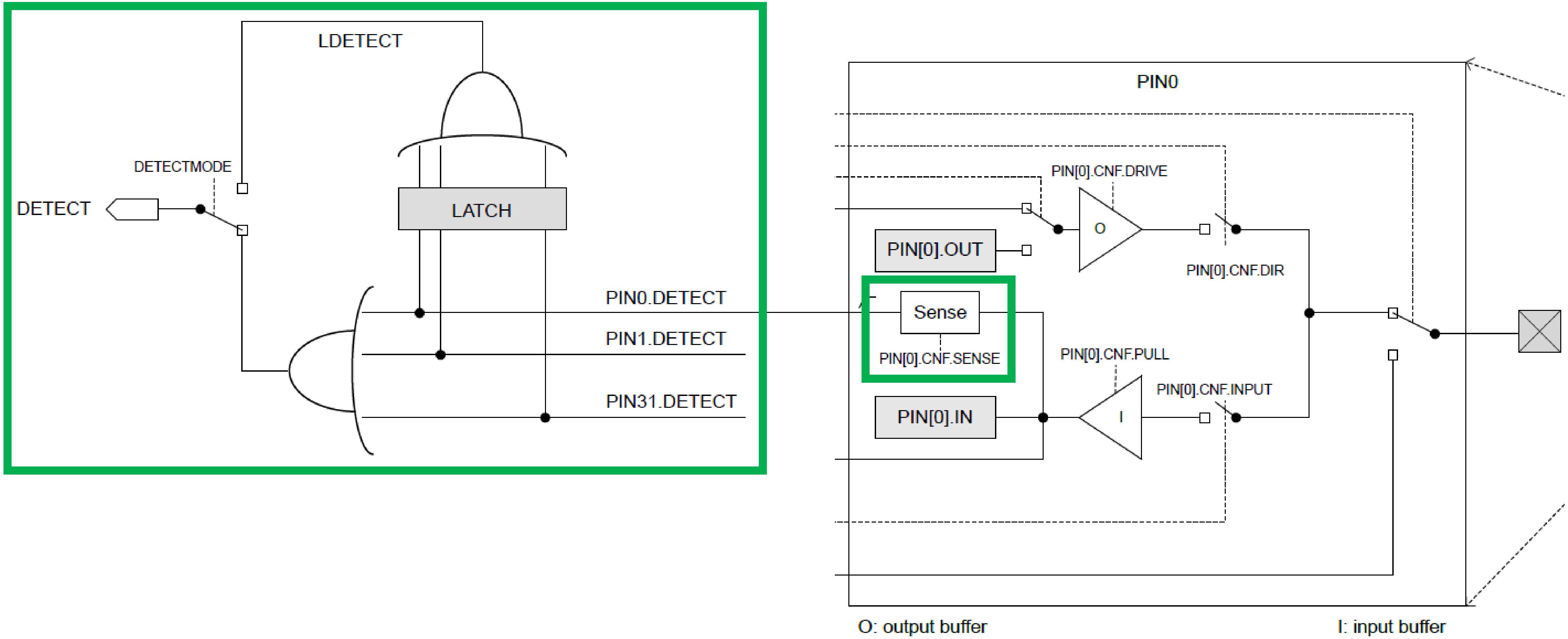
- Configure it
  - Port and Pin number
  - Task (output), Event (input), or Disabled
  - Polarity for input events
    - Low-to-high
    - High-to-low
    - Toggle (both directions)

- 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

- Digital circuits
- Controlling digital signals
  - GPIO
  - GPIOTE

Outline

- Bonus: thoughts on energy use

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