

Lecture 09

Analog Input

CE346 – Microcontroller System Design

Branden Ghena – Fall 2025

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

Administrivia

- Labs continue as normal
 - This week: Breadboarding
- Next week: Design Presentations
 - Details will be posted to Piazza tomorrow
 - Proposal feedback is coming, but I'm pretty happy with projects
 - Schedule will be up in the next day or two
- Quiz today
 - Tell your friends

Today's Goals

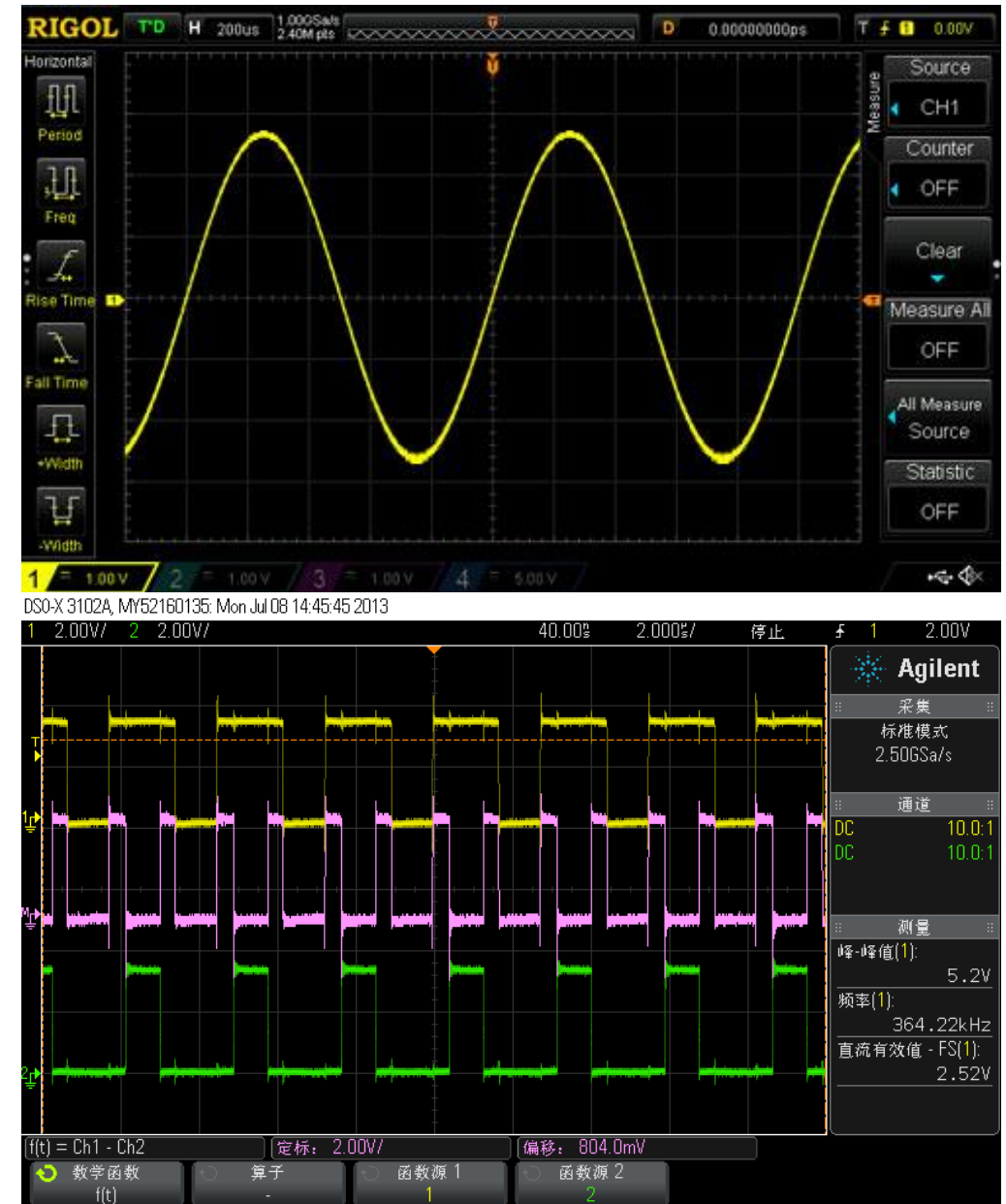
- Explore methods for sensing analog signals
 - Comparators
 - Analog-to-Digital Converters
- Discuss nRF implementation of these peripherals
- Apply analog sensing to a task: heartbeat detection

Outline

- **Comparators (and nRF implementations)**
- General ADC Design
- nRF ADC Implementation
- Heartbeat Sampling

Analog signals

- Exist in infinite states
 - From a maximum to a minimum
- Often used for interactions with the real world
 - Sensors usually generate analog signals
- Microbit example: microphone



Interacting with analog signals

- Microcontrollers are inherently digital
- Need a method for translating analog signal into a digital one
- Options:
 1. Determine if signal is higher or lower than some amount (Boolean)
 2. Determine voltage value of signal (N-bit number)

Interacting with analog signals

- Microcontrollers are inherently digital
- Need a method for translating analog signal into a digital one
- Options:
 - 1. Determine if signal is higher or lower than some amount (Boolean)**
 2. Determine voltage value of signal (N-bit number)

Determination is done by a Comparator

General comparator design

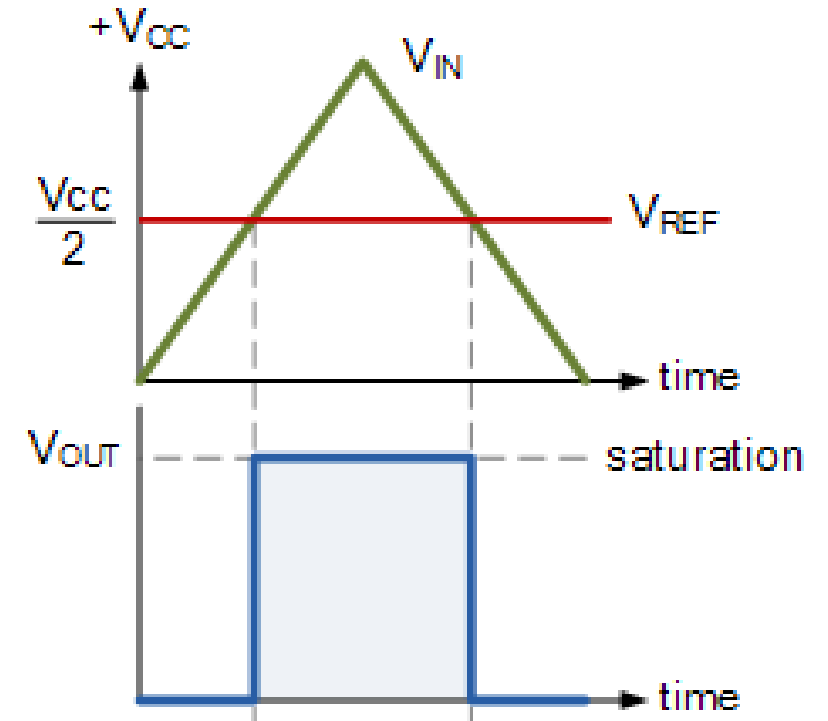
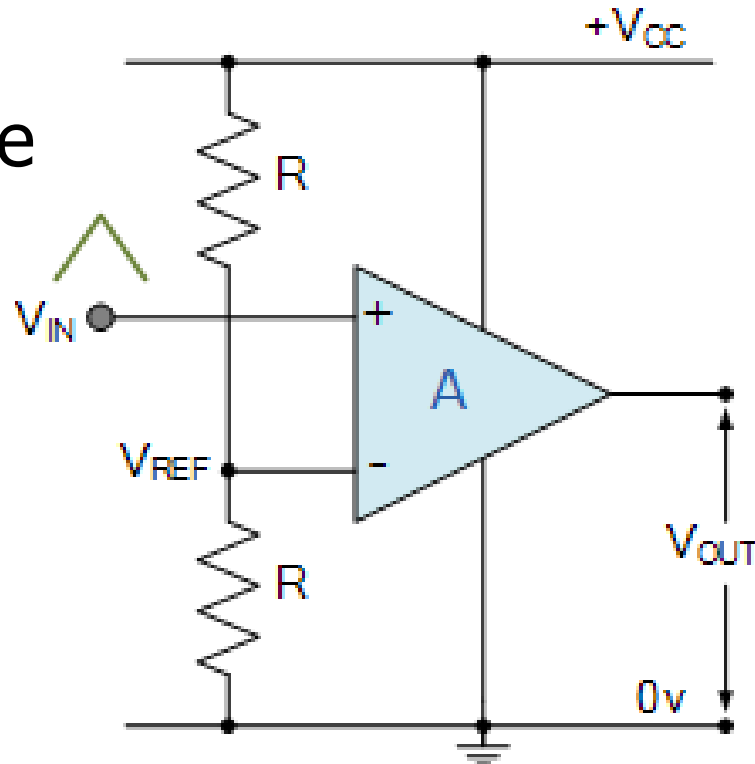
- Compares an analog input signal to a reference voltage

- V_{OUT} digital signal

- High: $V_{IN} > V_{REF}$
- Low: $V_{IN} < V_{REF}$

- Advantages:

- Simple (handful of transistors)
- Low power

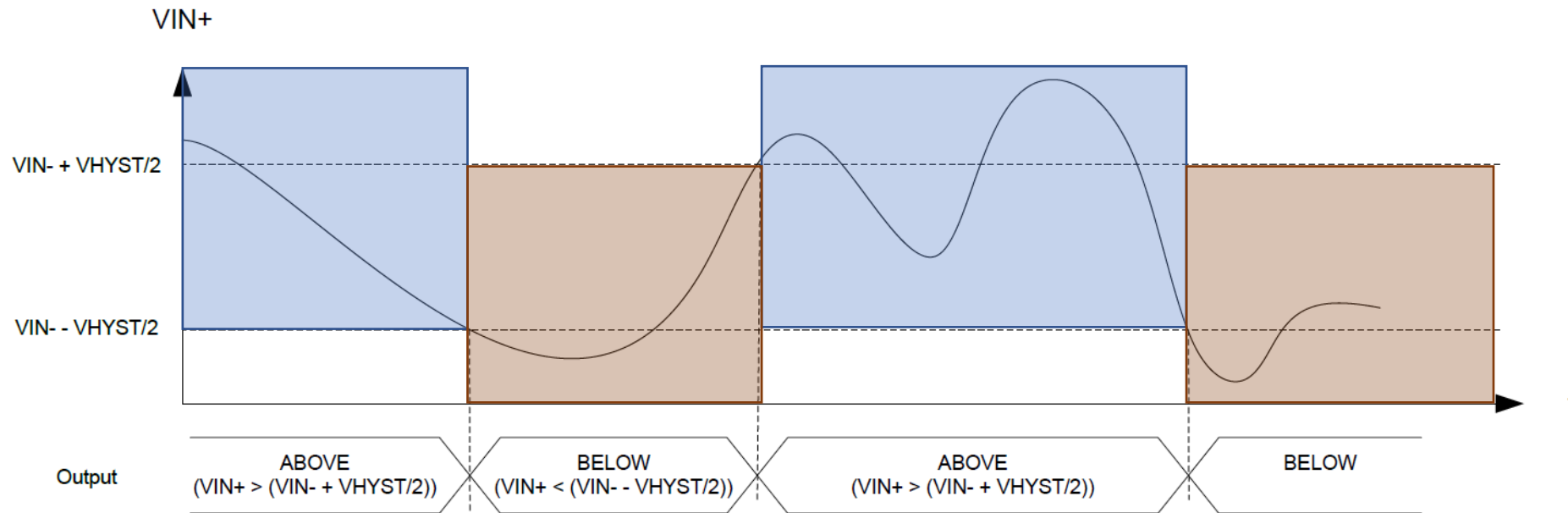
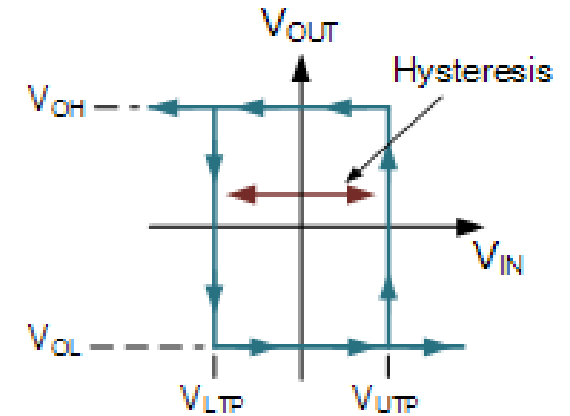


Comparator design questions

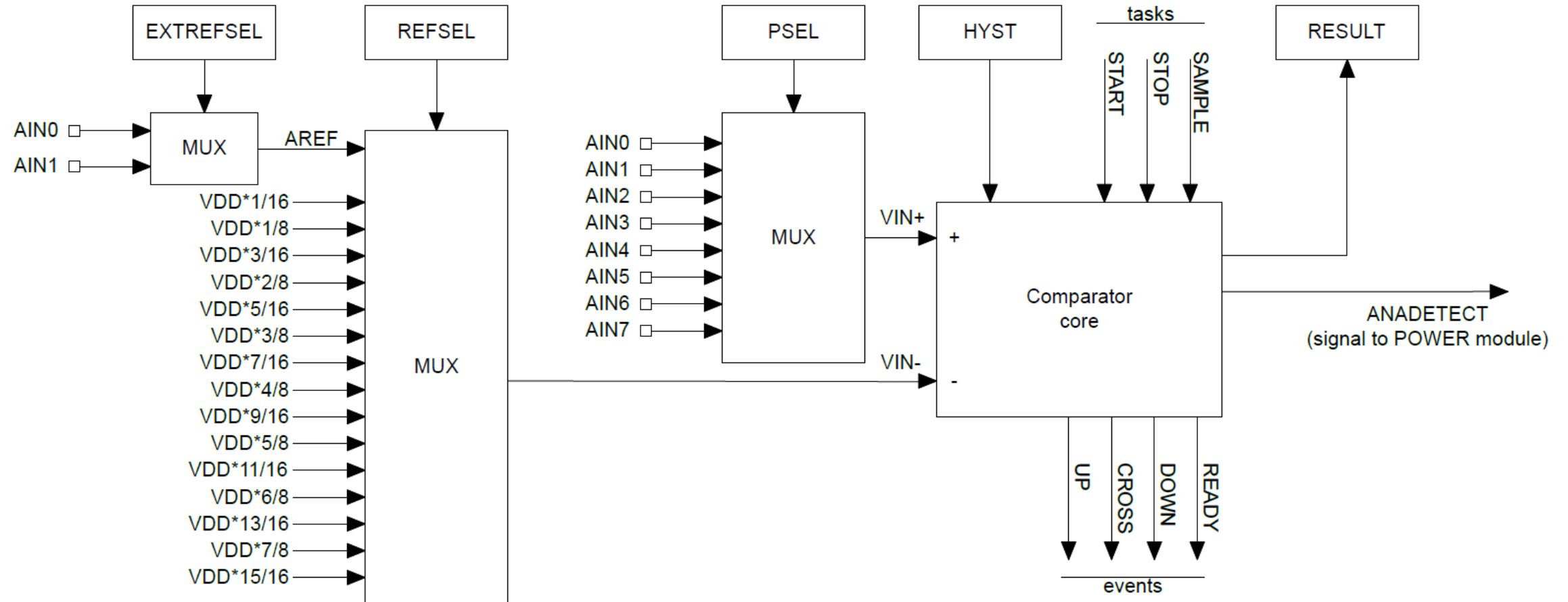
- What reference voltages are available?
 - A few internal voltages
 - Usually also allows external references from input pins
- When is an output generated?
 - Usually when status changes
 - Low-to-high, High-to-low, Both (like GPIO interrupts)

Hysteresis

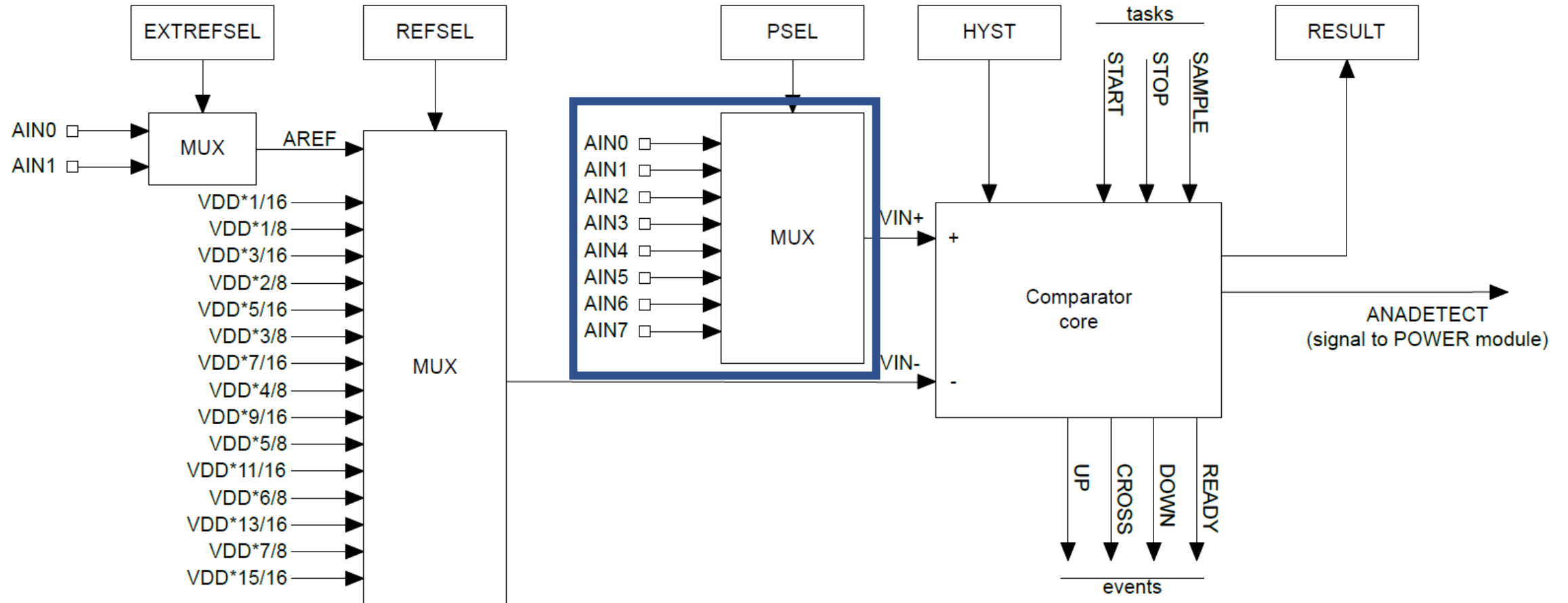
- A window added around signal state changes to prevent small amounts of noise from changing the output



nRF low-power comparator (LPCOMP)

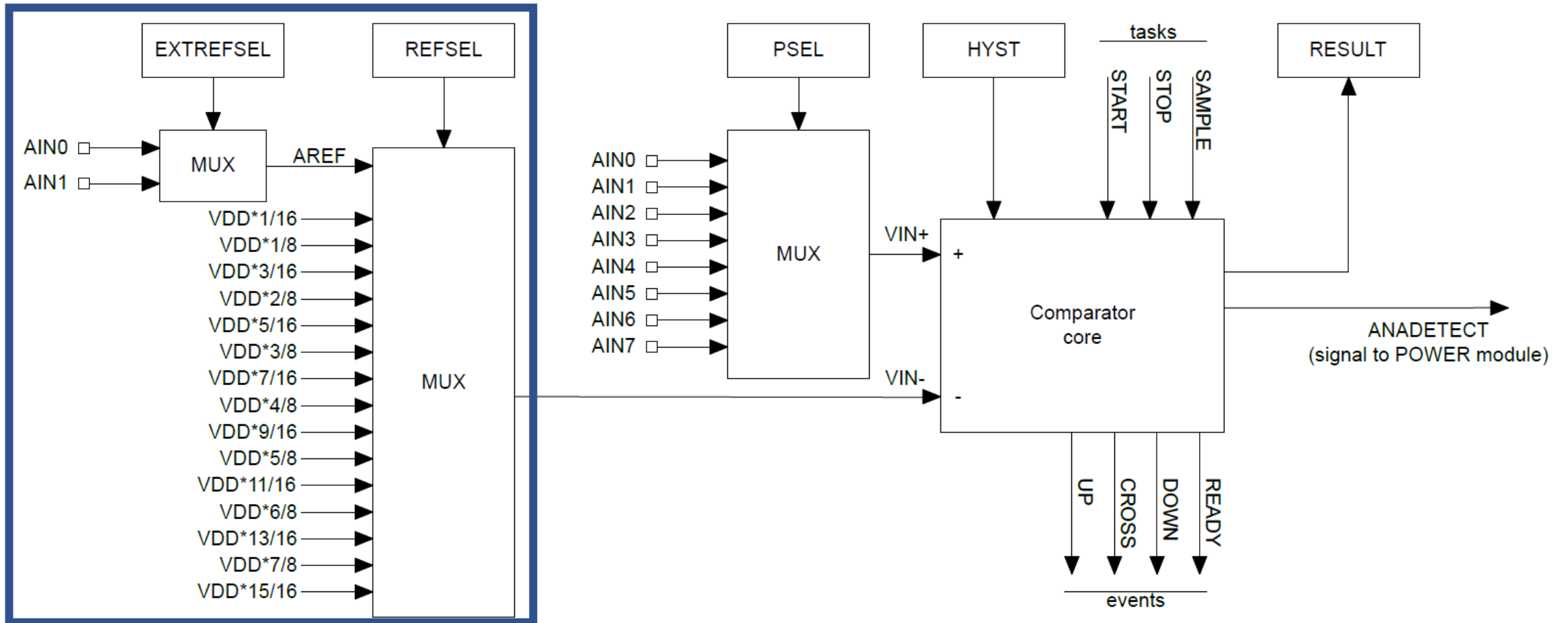


nRF low-power comparator (LPCOMP)



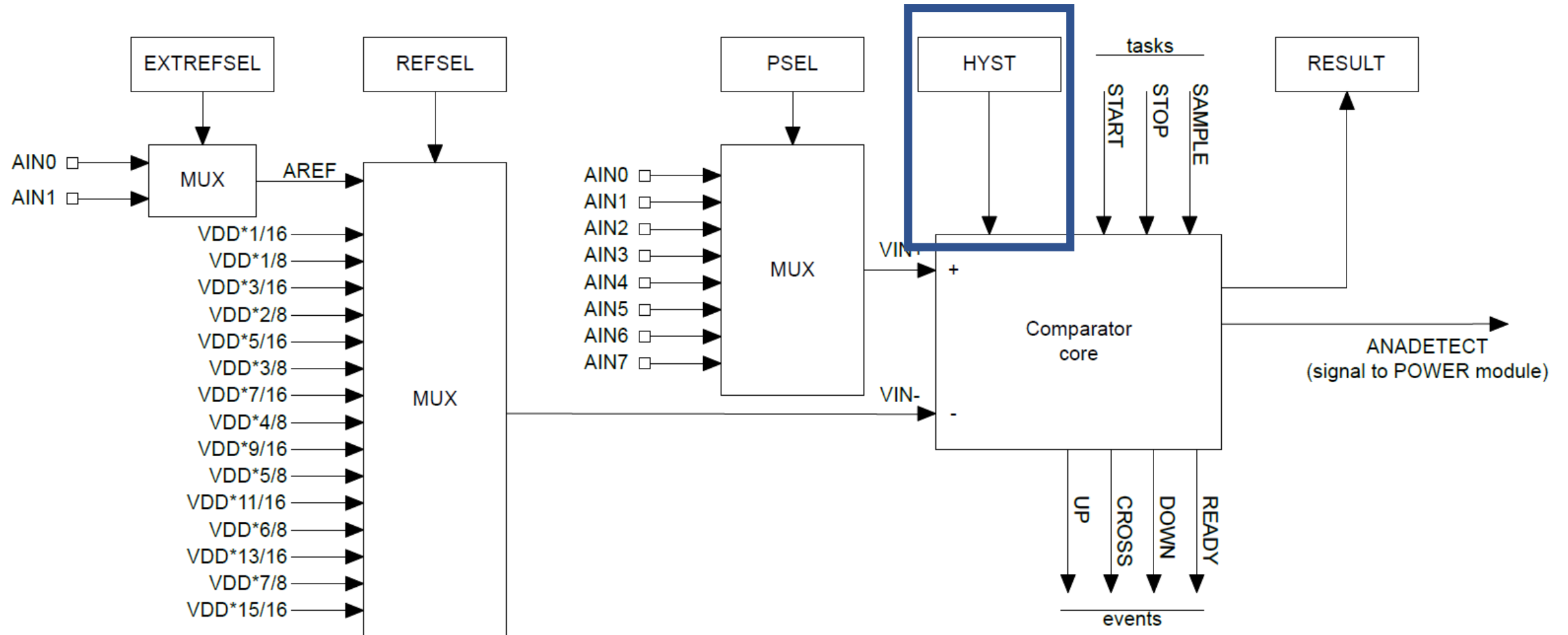
- Input: one of eight analog input pins

nRF low-power comparator (LPCOMP)



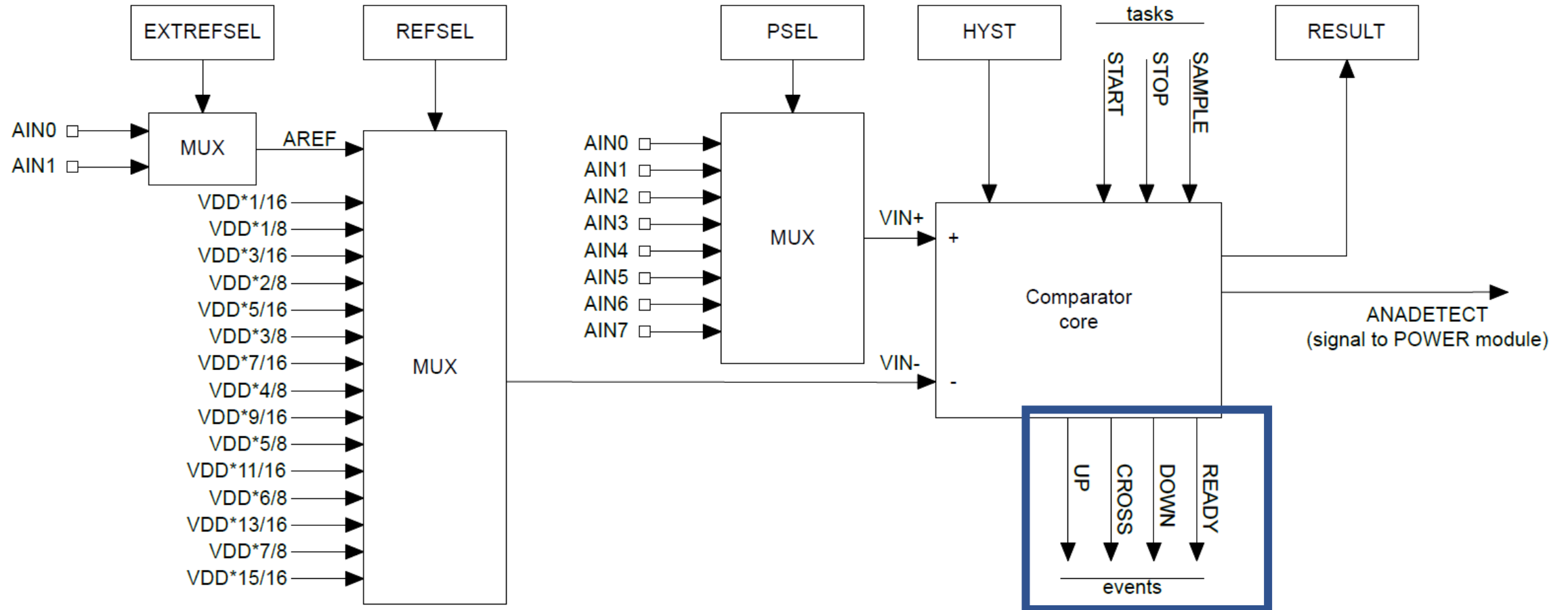
- Reference: one of two analog inputs or selection of $VDD * N/16$
 - VDD : Input voltage of the system

nRF low-power comparator (LPCOMP)



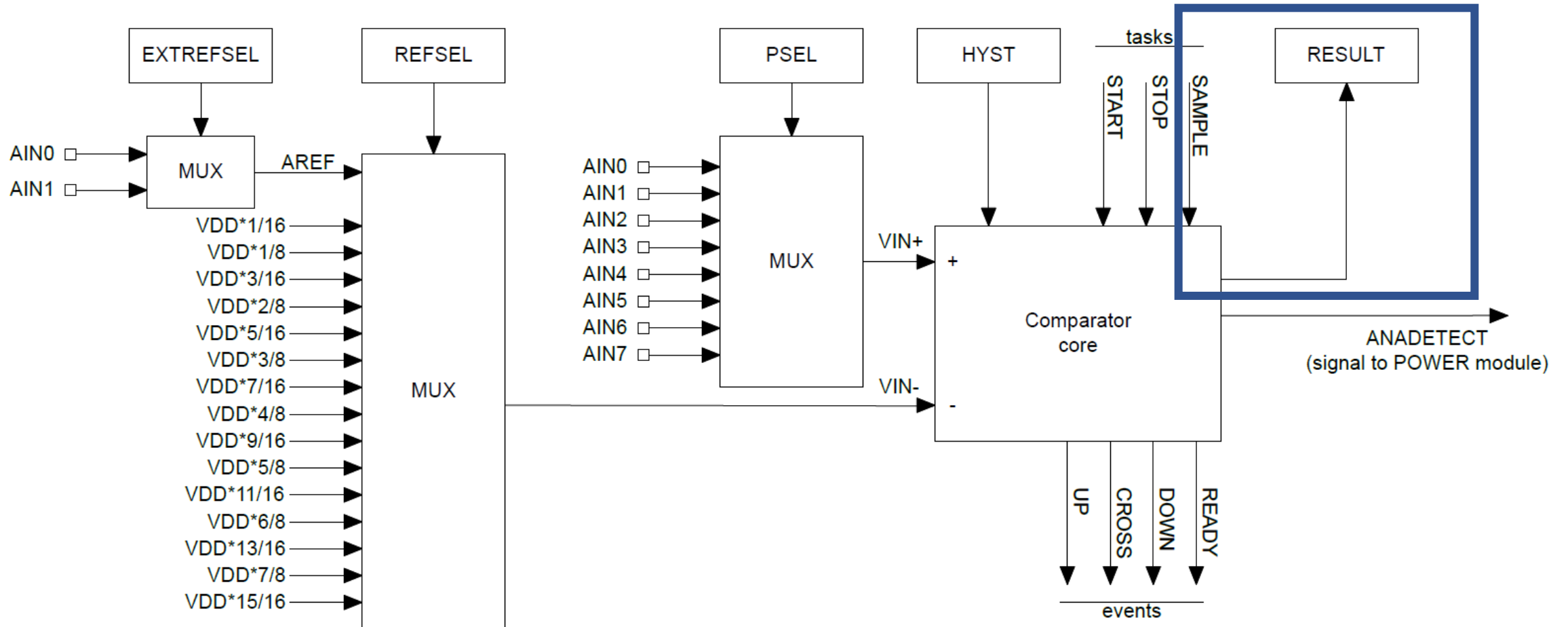
- Hysteresis: +/- 50 mV range around VIN- when enabled

nRF low-power comparator (LPCOMP)



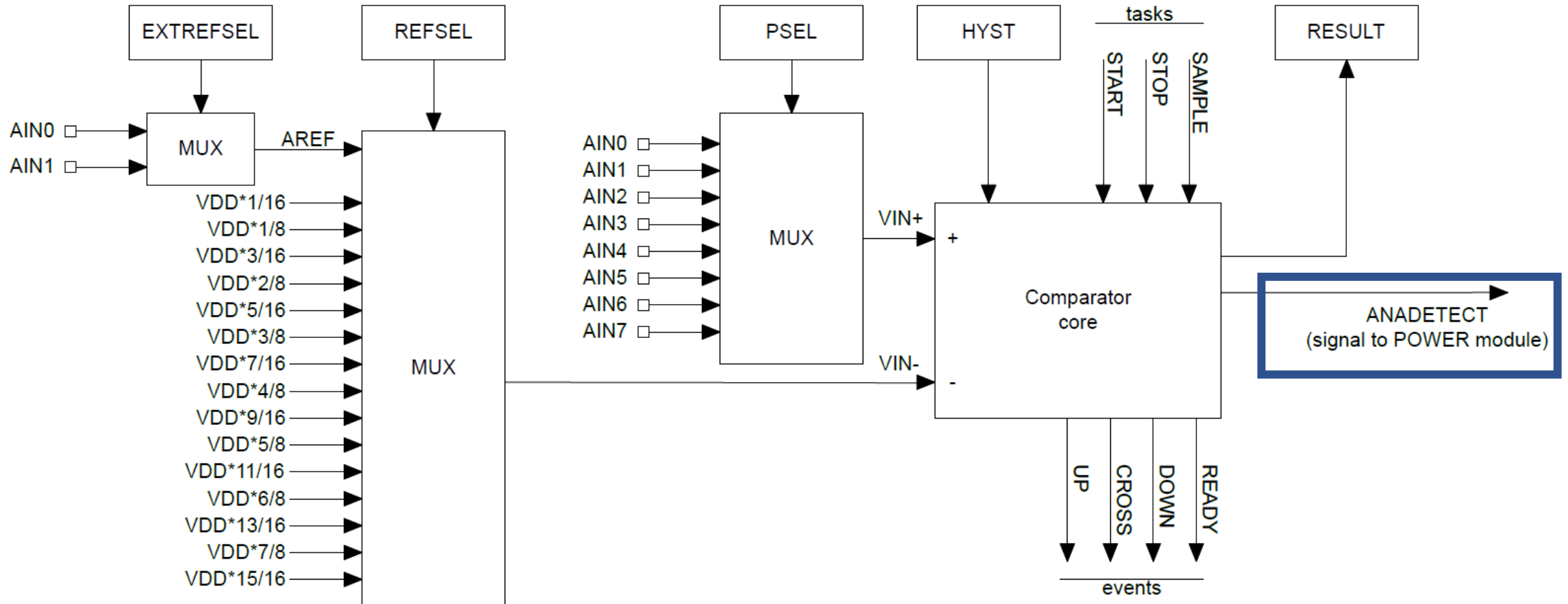
- Events: transition signals + ready ($\sim 150 \mu s$ startup time)

nRF low-power comparator (LPCOMP)



- Can also request what the current comparison state is (high/low)

nRF low-power comparator (LPCOMP)



- Can be used for low-power wakeup of microcontroller

nRF COMP peripheral

- Analog Comparator (not low power)
 - More advanced version of a comparator (otherwise similar)
- What advantages would a more capable comparator have?
 - Configurable hysteresis
 - LPCOMP: +/- 50 mV COMP: any of the N/64 voltage levels
 - Faster detection
 - LPCOMP: 5 μ s COMP: 0.1-0.6 μ s (depends on power mode)
 - More possible reference voltages
 - LPCOMP: VDD or input COMP: VDD, 1.2v, 1.8v, 2.4v, or input
 - LPCOMP: 16 levels COMP: 64 levels

Break + Question: Internal reference voltages

- **Why have internal voltage references other than the system voltage (VDD)? (i.e, 1.2v or 2.4v)**

Break + Question: Internal reference voltages

- **Why have internal voltage references other than the system voltage (VDD)? (i.e, 1.2v or 2.4v)**
 - What if what you want to measure *is* VDD?
 - Battery voltage
 - Did someone just unplug me?
 - etc.
 - What if VDD isn't stable?
 - Battery voltage
 - Energy-harvesting system
 - Hard to know what any particular value means...

Outline

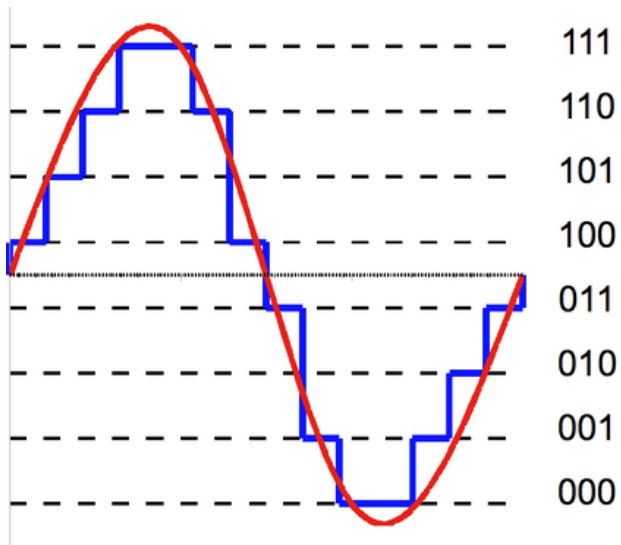
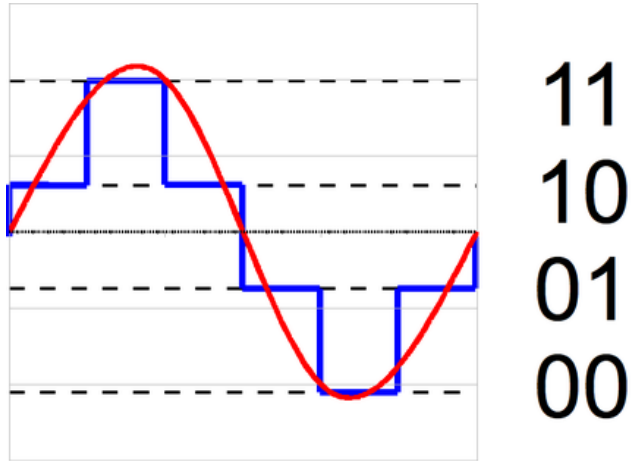
- Comparators (and nRF implementations)
- **General ADC Design**
- nRF ADC Implementation
- Heartbeat Sampling

Interacting with analog signals

- Microcontrollers are inherently digital
- Need a method for translating analog signal into a digital one
- Options:
 1. Determine if signal is higher or lower than some amount (Boolean)
 - 2. Determine voltage value of signal (N-bit number)**

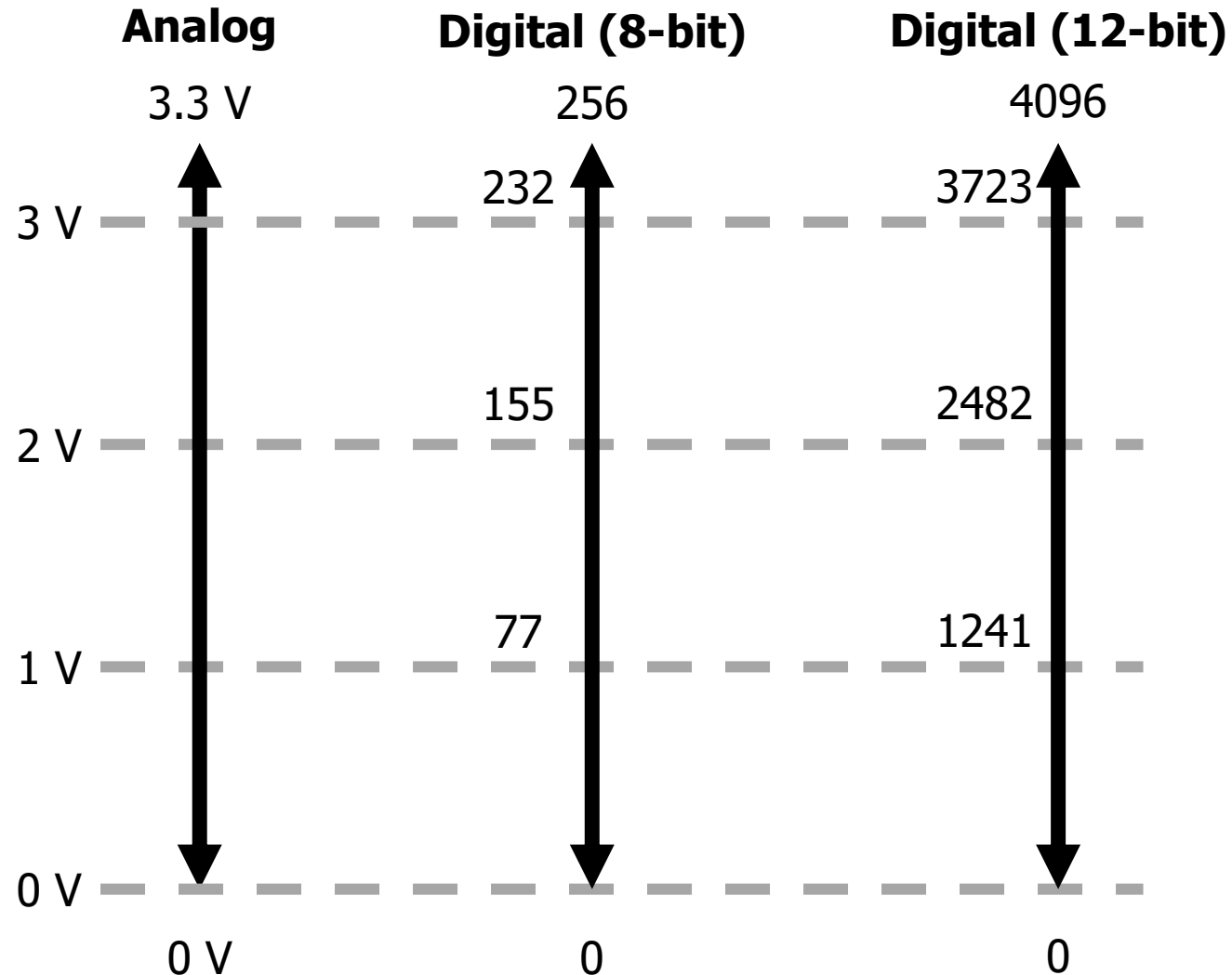
Translation is done by an Analog-to-Digital Converter (ADC)

Quantization



- Analog voltages are represented by discrete voltage levels
- Comparators are 1-bit ADCs
 - Split into two regions
 - Good ADCs split into 4000-16000 regions
- More levels gives a more accurate representation of the signal

Translating voltage and ADC counts



$$Value = \frac{V_{IN}}{V_{REF}} * (2^{Resolution})$$

$$V_{IN} = \frac{Value}{2^{Resolution}} * V_{REF}$$

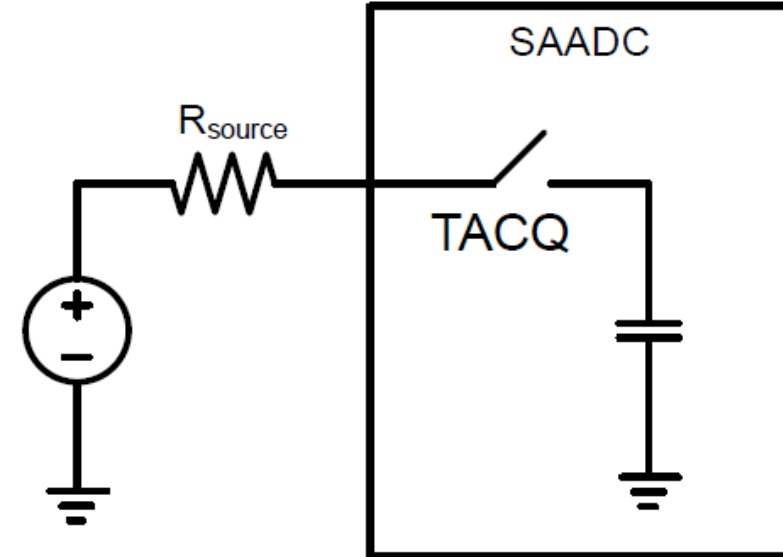
- V_{REF} selects maximum range
- Ground is usually minimum range
- Resolution depends on hardware
 - Either hardcoded or a selection

Analog to digital translation process

- Two steps:

1. Acquisition

- Read in signal for some amount of time
- Signal connected to a capacitor
- Fills capacitor up to voltage level
- Speed depends on input resistance
 - 1-100 μs is common

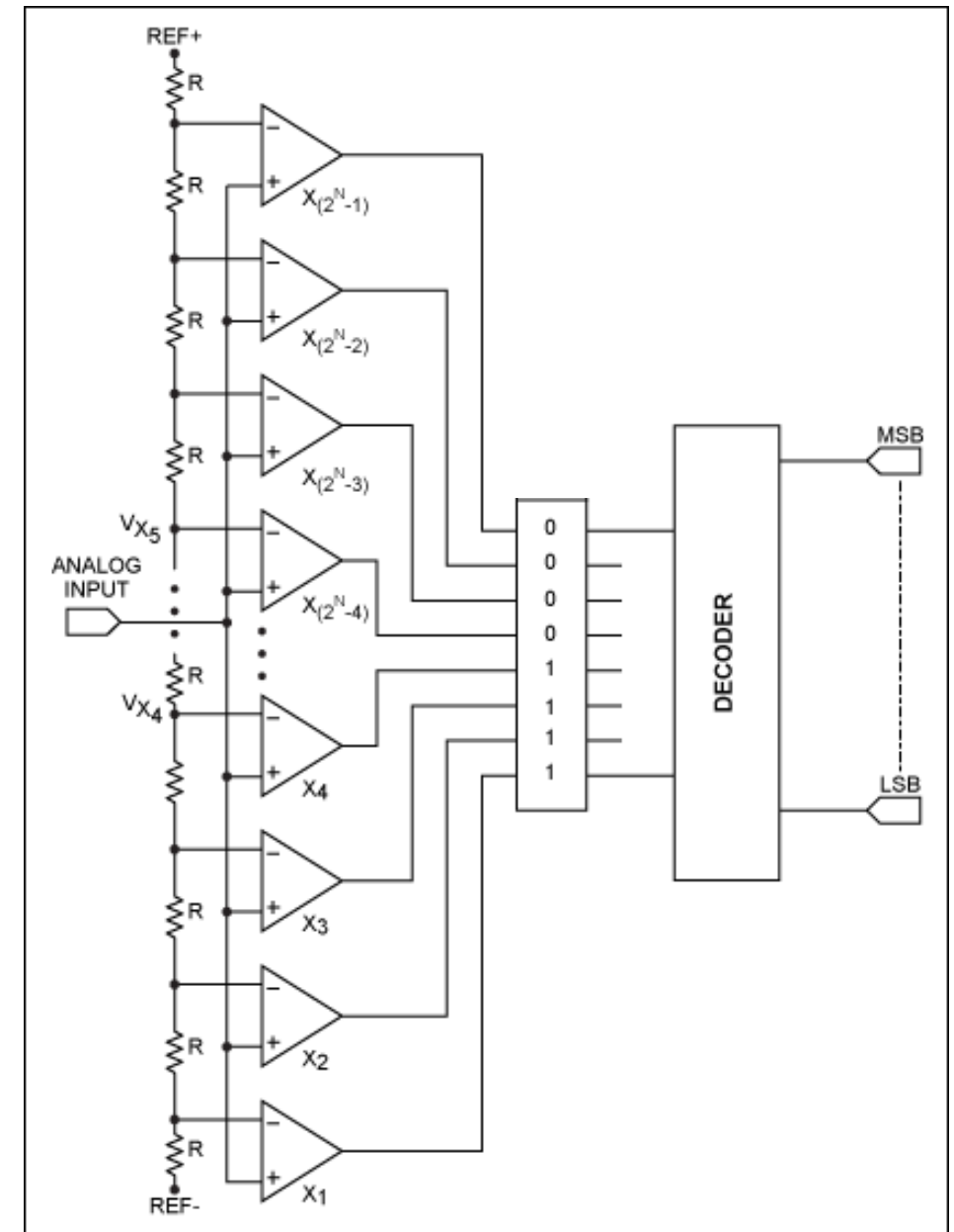


2. Conversion

- Determine which digital value the read signal corresponds to

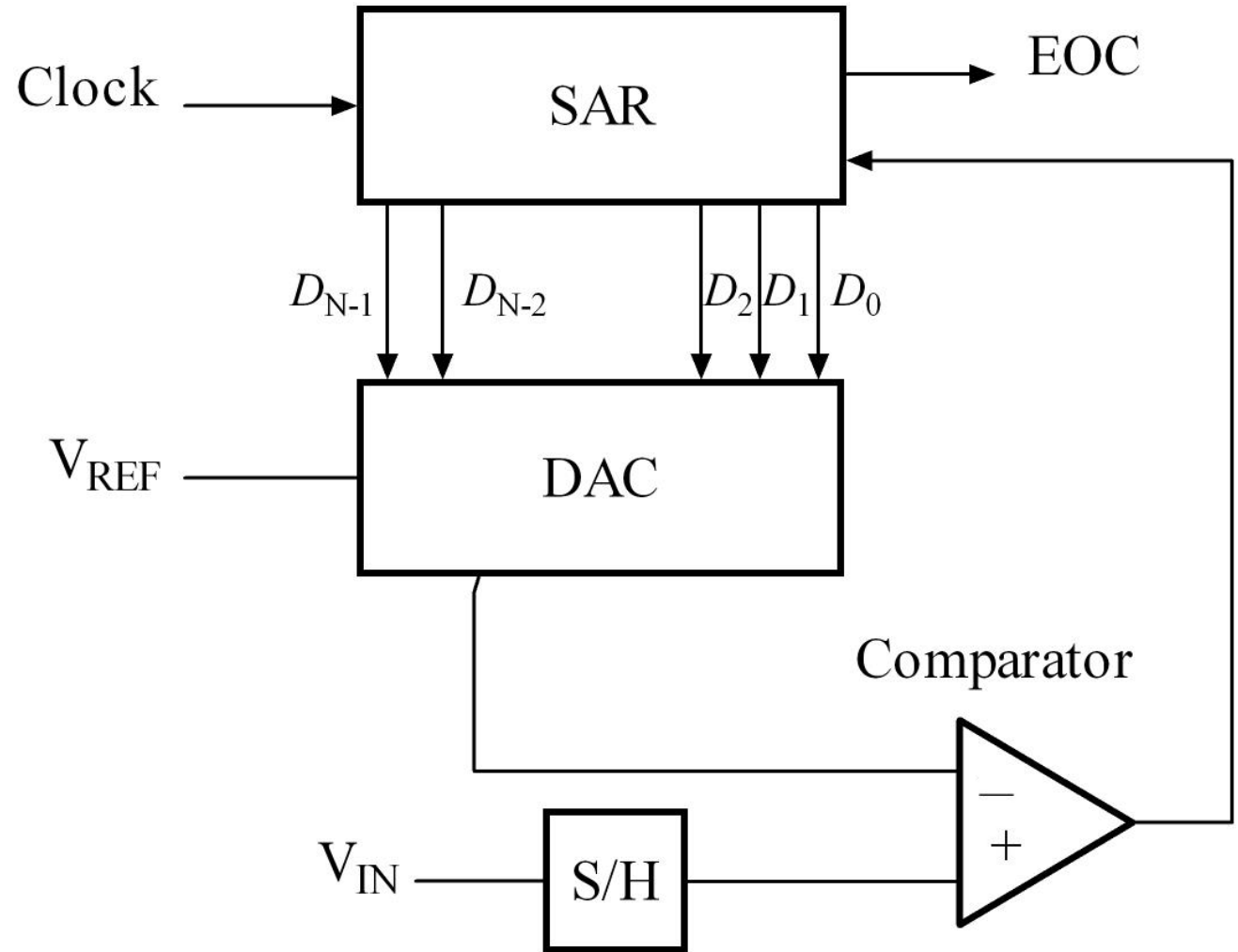
Direct-conversion ADC

- Chain comparators together
 - Each with a separate reference voltage
- Digital value determined immediately
 - Also known as "Flash" ADCs
- Downside: needs $2^n - 1$ comparators
 - Reserved for expensive applications



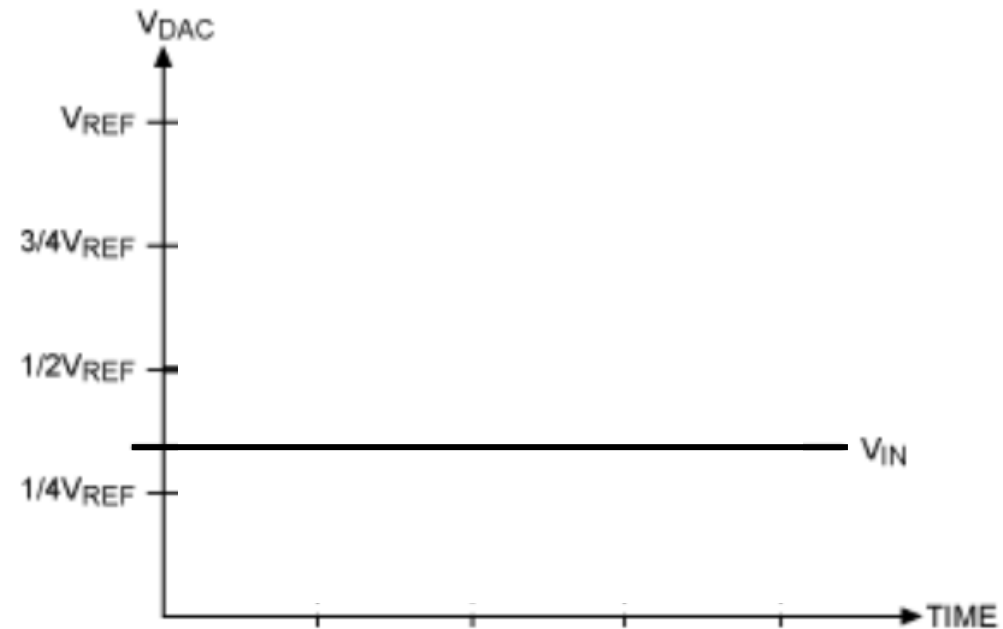
Successive-Approximation ADC

- Method: Binary Search
 - Compare signal to generated reference
 - Increase or decrease reference as needed
 - Repeat
- DAC creates reference (Digital-to-Analog Converter)
 - Final value of DAC is the ADC value



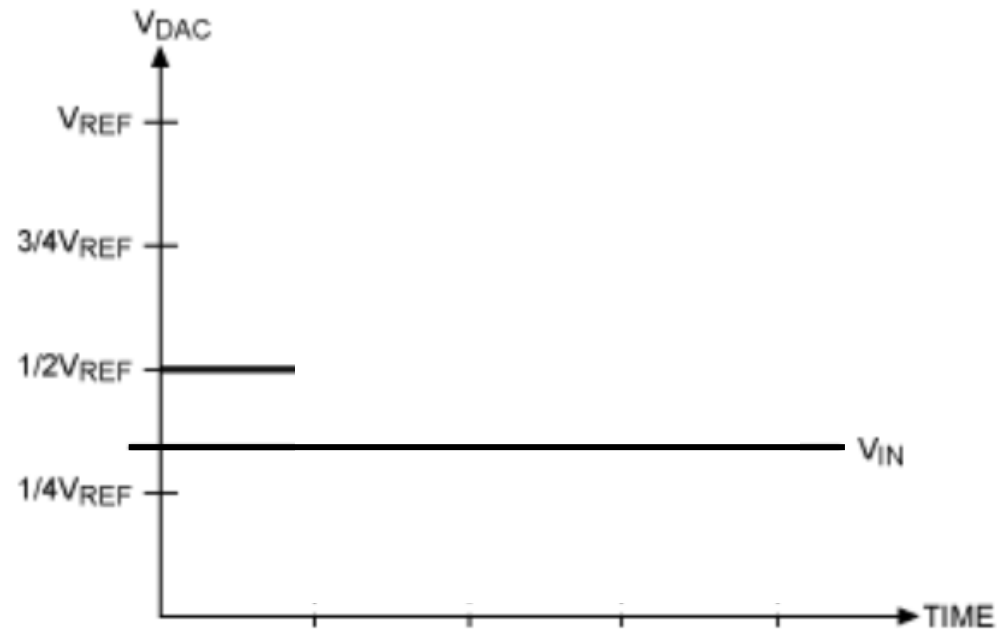
Successive Approximation Example

- 4-bit ADC with an input signal V_{IN}



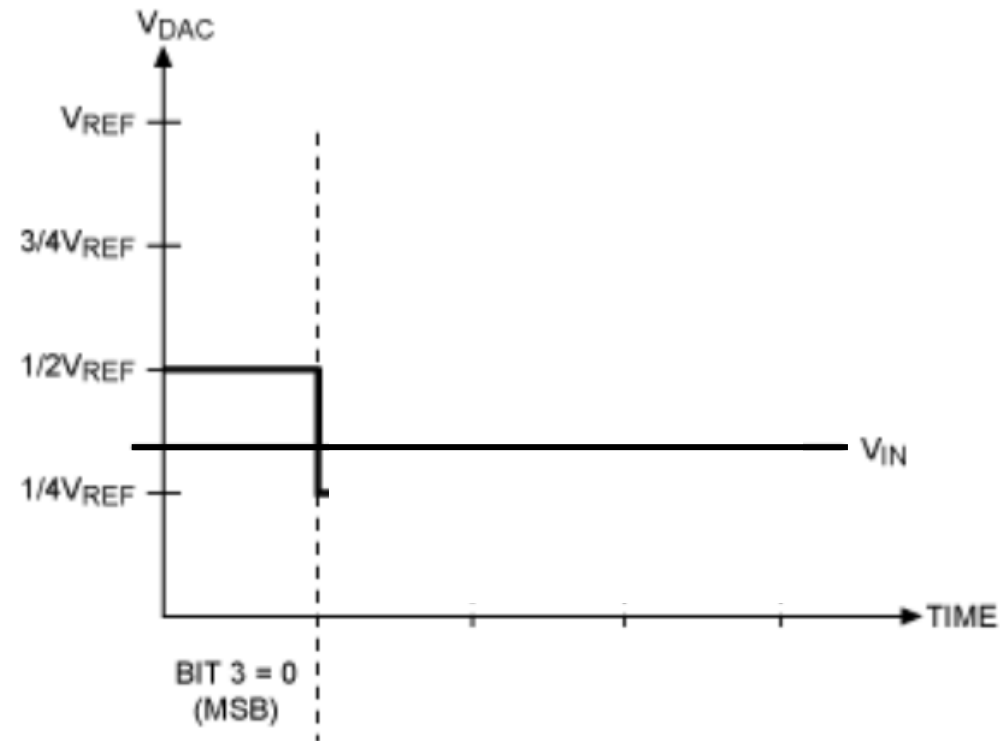
Successive Approximation Example

1. Compare $\frac{1}{2} V_{REF}$ (0b???) to V_{IN}
 - If V_{IN} is greater, bit is 1. Else 0



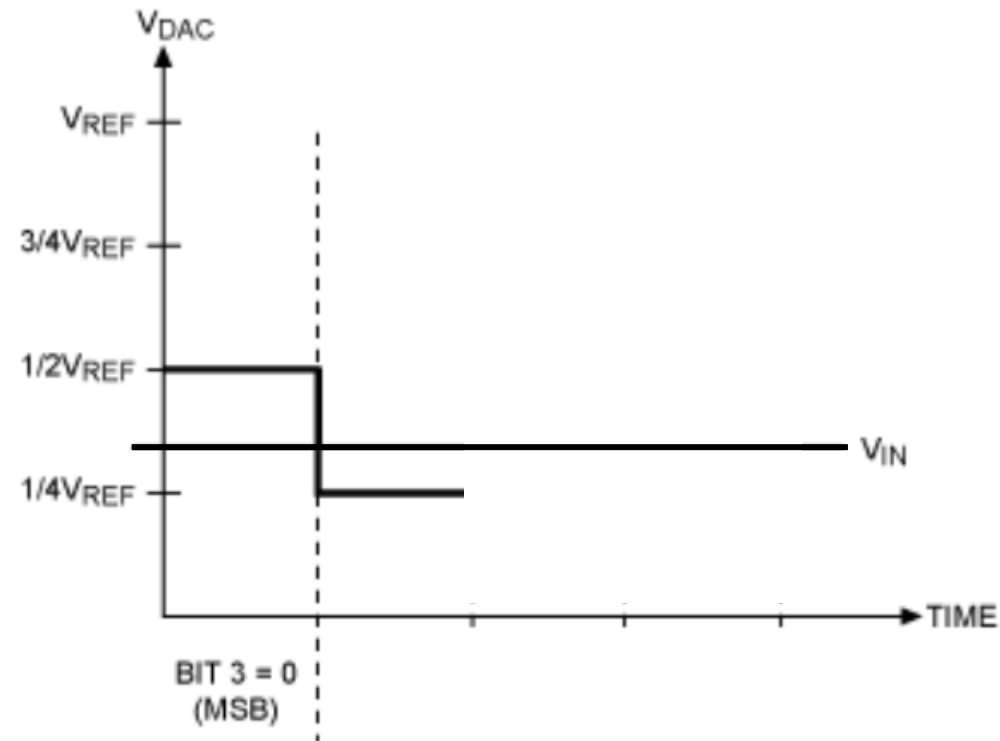
Successive Approximation Example

1. Compare $\frac{1}{2} V_{REF}$ (0b**0**???) to V_{IN}
 - If V_{IN} is greater, bit is 1. Else 0
 - V_{IN} is less. So set that bit to zero



Successive Approximation Example

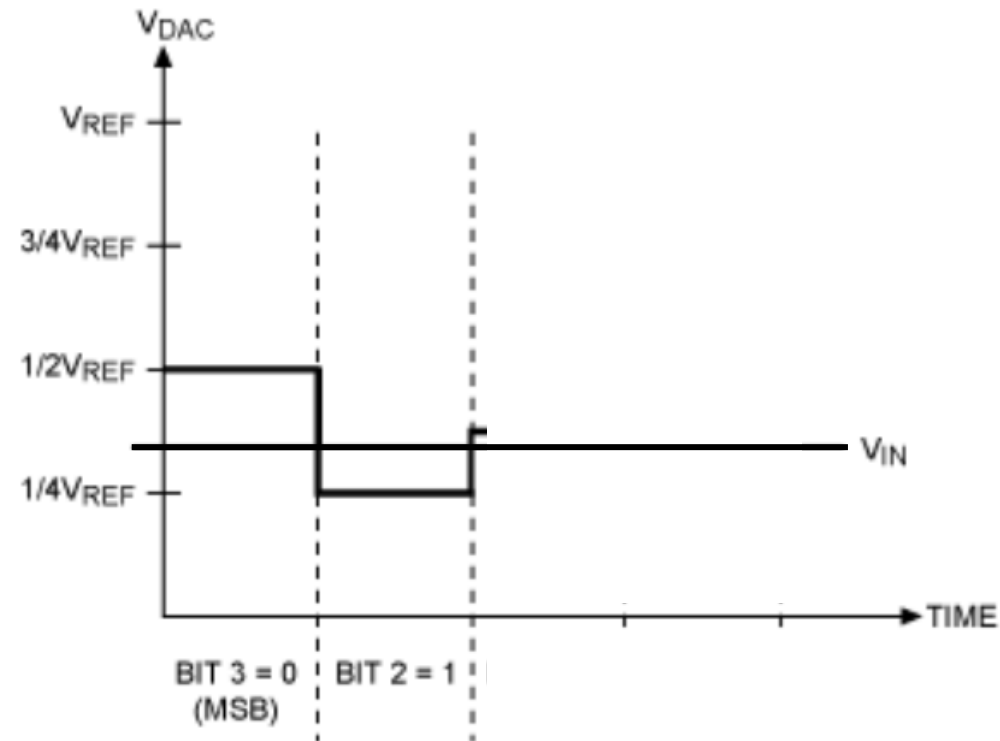
2. Compare $1/4 V_{REF}$ ($0b\mathbf{0}\underline{1}??$) to V_{IN}
- If V_{IN} is greater, bit is 1. Else 0



Successive Approximation Example

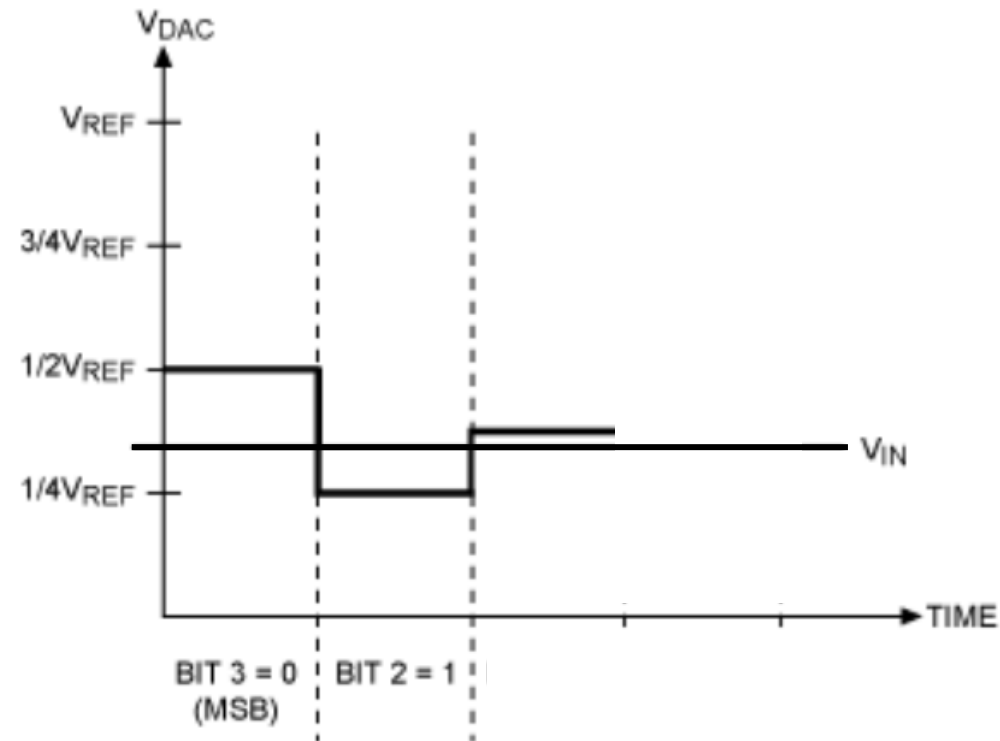
2. Compare $1/4 V_{REF}$ (0b**01**??) to V_{IN}

- If V_{IN} is greater, bit is 1. Else 0
- V_{IN} is greater. So set that bit to one



Successive Approximation Example

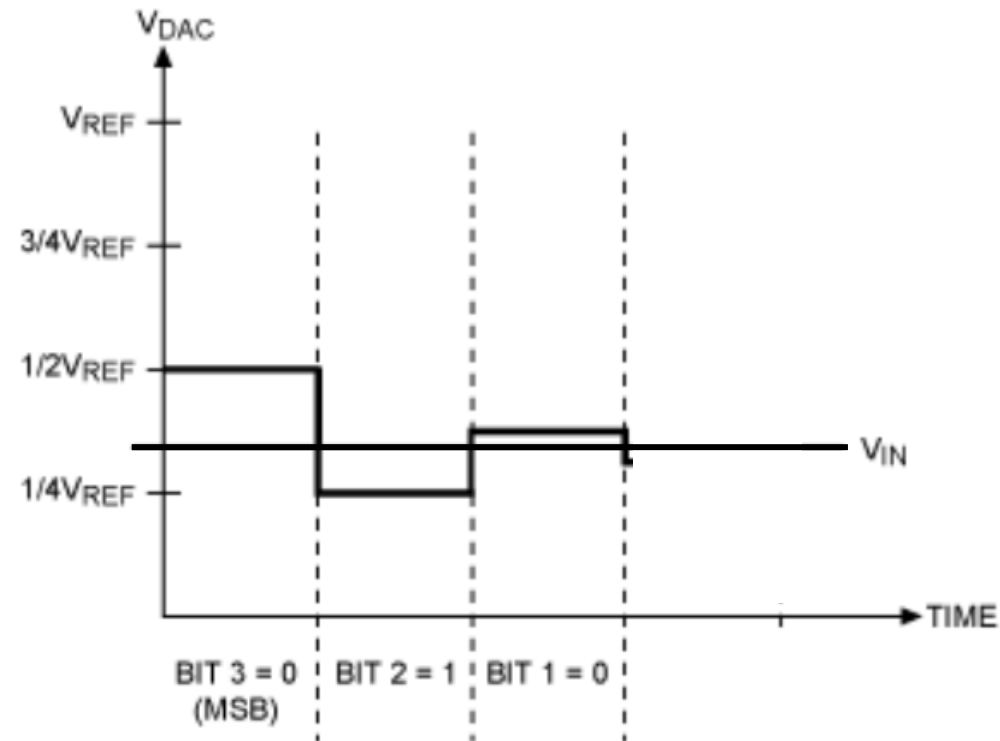
3. Compare $\frac{3}{8} V_{REF}$ (**0b01??**) to V_{IN}
- If V_{IN} is greater, bit is 1. Else 0



Successive Approximation Example

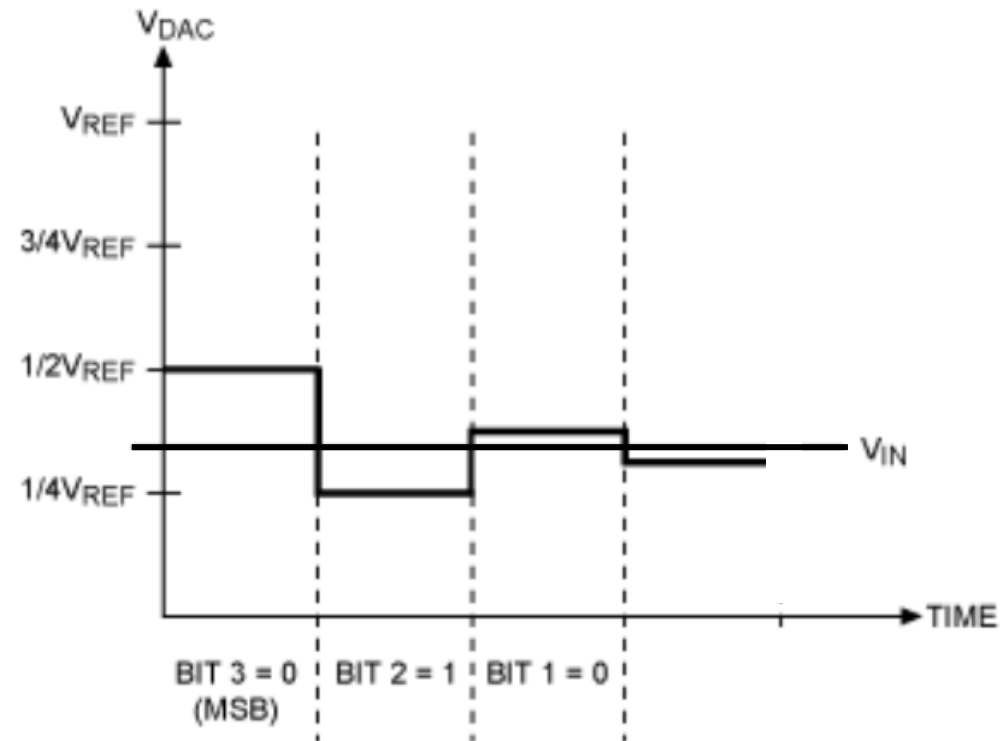
3. Compare $\frac{3}{8} V_{REF}$ (0b**010**?) to V_{IN}

- If V_{IN} is greater, bit is 1. Else 0
- V_{IN} is less. So set that bit to zero



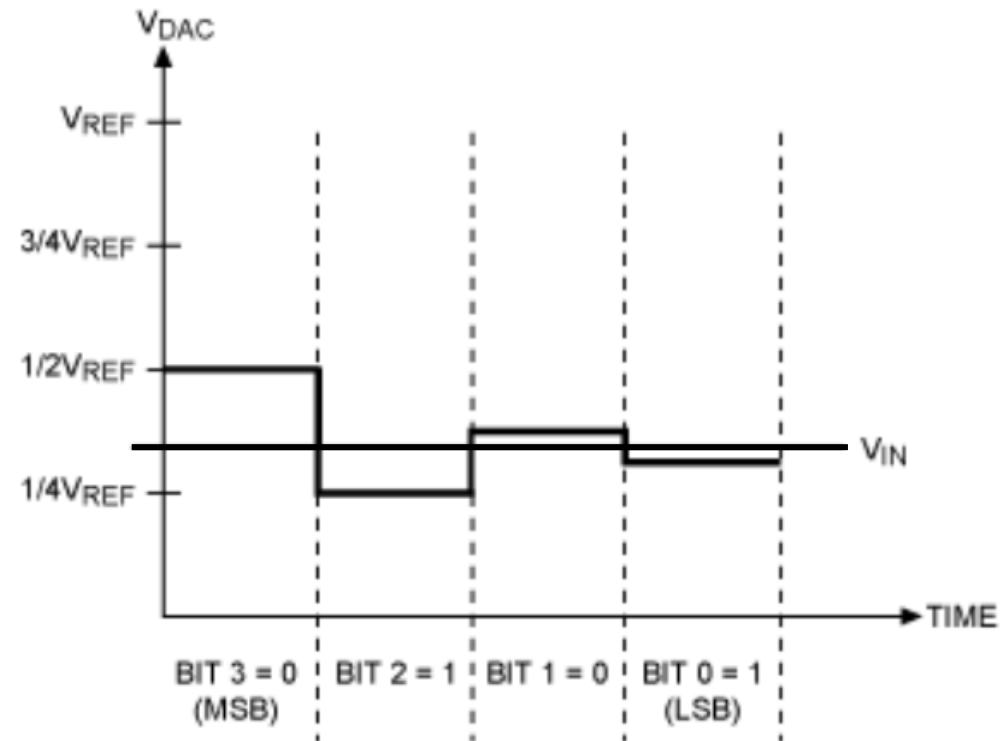
Successive Approximation Example

4. Compare $5/16 V_{REF}$ ($0b\mathbf{010}\underline{?}$) to V_{IN}
- If V_{IN} is greater, bit is 1. Else 0



Successive Approximation Example

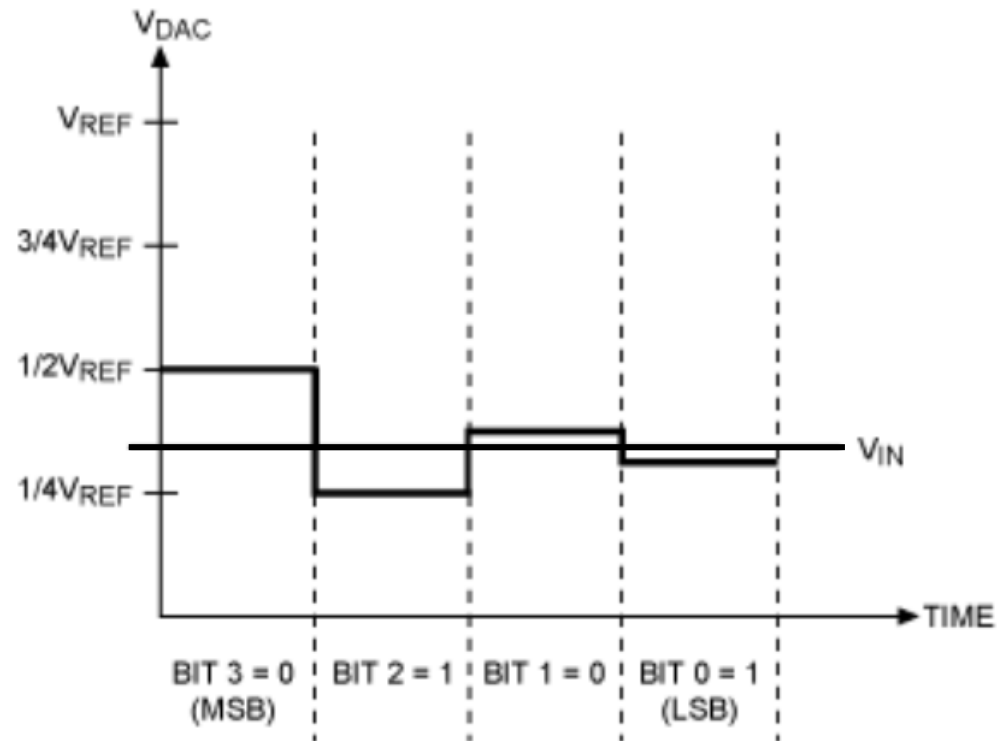
4. Compare $5/16 V_{REF}$ (0b**0101**) to V_{IN}
- If V_{IN} is greater, bit is 1. Else 0
 - V_{IN} is greater. So bit is one



Successive Approximation Example

5. Output is $5/16 V_{REF}$ (0b0101)

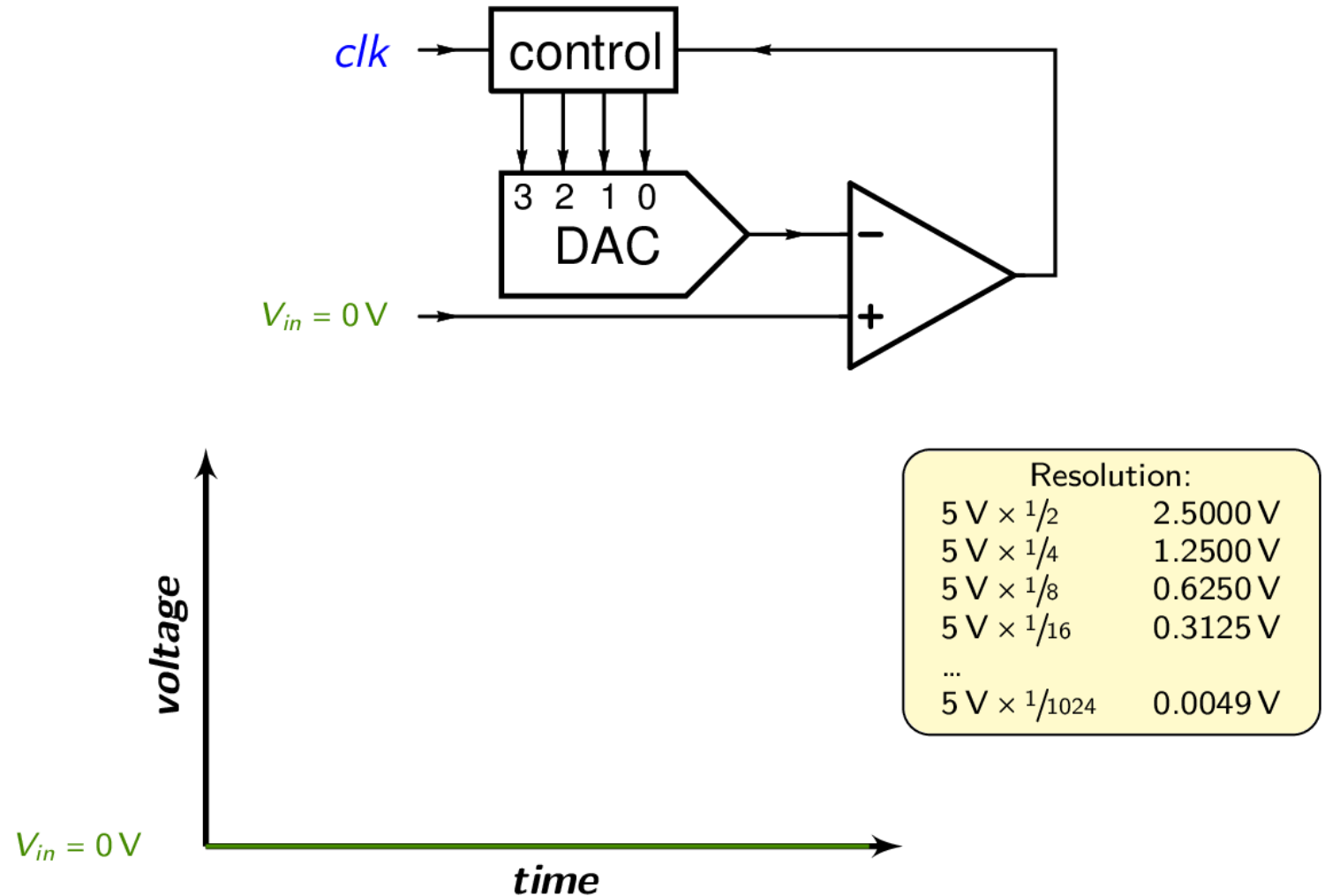
- Slight underestimate of the real value, but as close as we can get
- More bits would get us even closer



Successive Approximation Example

- Performs a binary search to determine correct reference signal value

Successive Approximation – example of a 4-bit ADC



Tradeoffs in ADC design

- Direct-Conversion: more expensive (more silicon)
- SAADC: more time consuming (more binary search time)
- Most microcontrollers land on successive-approximation designs
 - The slowdown isn't an issue for slowly changing signals
 - Quickly changing signals might need special hardware anyways

Break + Critical Thinking

- How much ADC resolution is needed?

Break + Critical Thinking

- How much ADC resolution is needed?
 - Resolution requirement depends on signal being sensed
 - Temperature sensor probably doesn't need 16-bit ADC
 - Difference between 30.001°C and 30.002°C is usually irrelevant
 - Microphone might though!
 - Differences are audible until they are small enough
 - 16-bit or 24-bit tend to be common choices for audio data

Outline

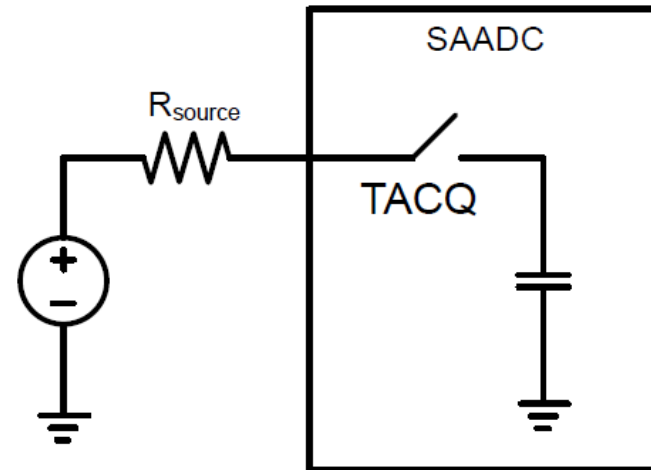
- Comparators (and nRF implementations)
- General ADC Design
- **nRF ADC Implementation**
- Heartbeat Sampling

nRF SAADC (Successive Approximation ADC)

- Only one actual ADC peripheral in the chip
 - One measurement at any given time
- However, provides 8 different “channels” with unique configurations
 - Virtualization in hardware!
 - Typical setup is to sample each channel one-after-the-other

SAADC Resolution and Sampling

- Resolution is selectable (for the whole peripheral)
 - 8, 10, 12, or 14 bits
 - Result stored as 16-bit value regardless
- Sampling time is selectable (for each channel)
 - 3-40 μs
 - Longer sampling time is important for very low-current signals

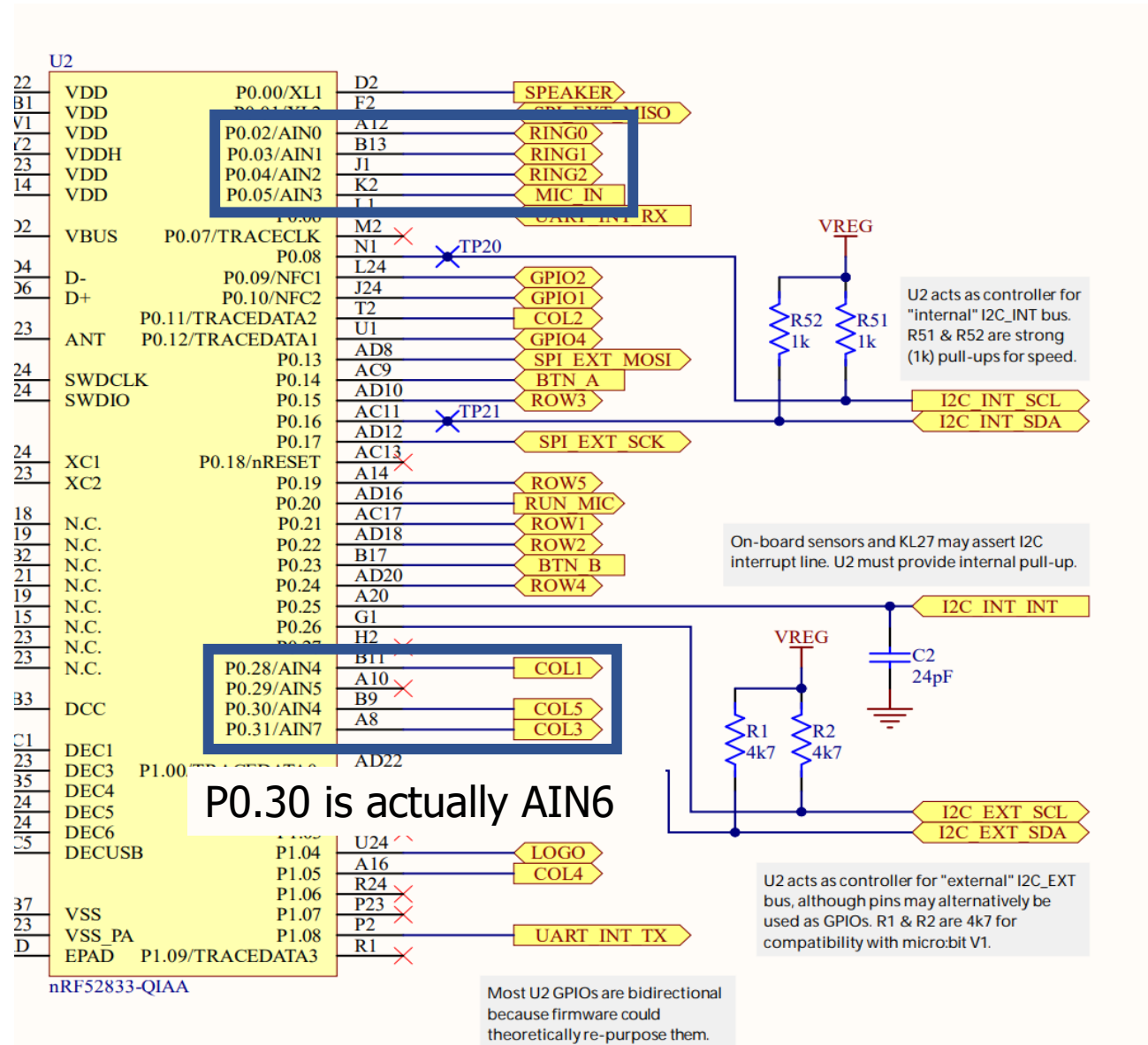


nRF SAADC Inputs

- Can read the system voltage directly
- Or read one of 8 different analog input pins
 - These pins ARE fixed and not totally reconfigurable
 - Analog signals are more susceptible to interference
- Need to make sure that you're connecting to an analog input pin

Analog inputs on the Microbit

- Of the 8 analog input pins
 - One for microphone
 - One is not connected
 - Three are repurposed for the LED matrix
 - Three are available as external inputs**
- For projects, students sometimes end up with external ADCs



Triggering sample collection

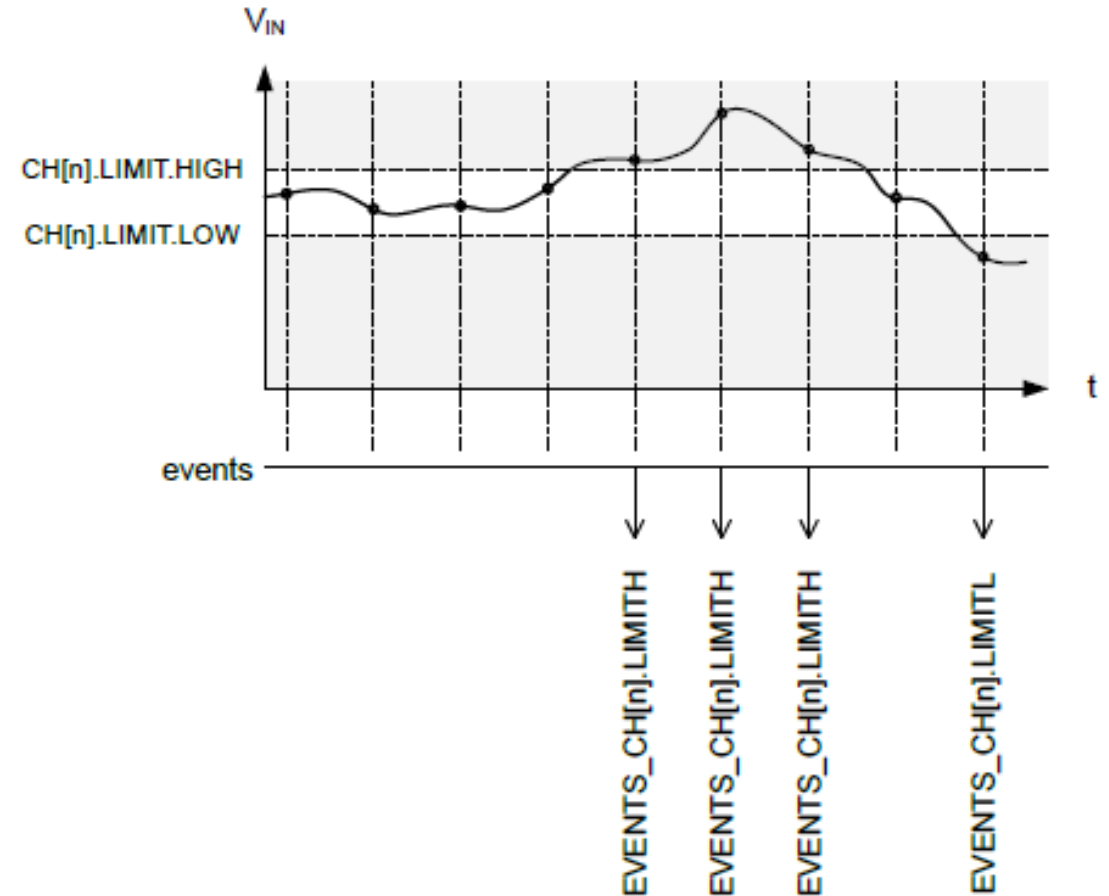
- Can be triggered with TASK_START on demand
 - Including through EVENT->TASK chaining
- Includes a timer within itself to automatically trigger sampling
 - Rate = $16 \text{ MHz} / (2^{\text{Scale}})$ where scale is 11 bits
 - Maximum rate is 7.8 kHz

EasyDMA on the SAADC

- There is no register to read ADC results from
- Instead, you must use DMA to collect samples
- At configuration time, provide:
 - Pointer to RAM
 - Must be RAM, not Flash
 - Maximum count of 16-bit samples to be written starting at address
 - Up to 32768
- When complete, a register tells you the amount of samples written to RAM

Analog signal monitoring

- Includes two comparators for each channel
 - High and Low limits
- Generates events whenever transitioning above High or below Low
 - Events can generate interrupts if desired

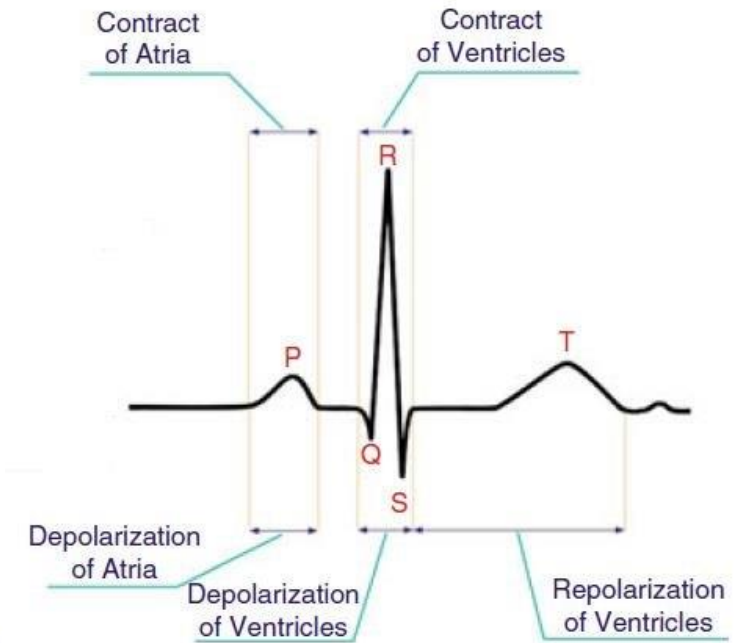
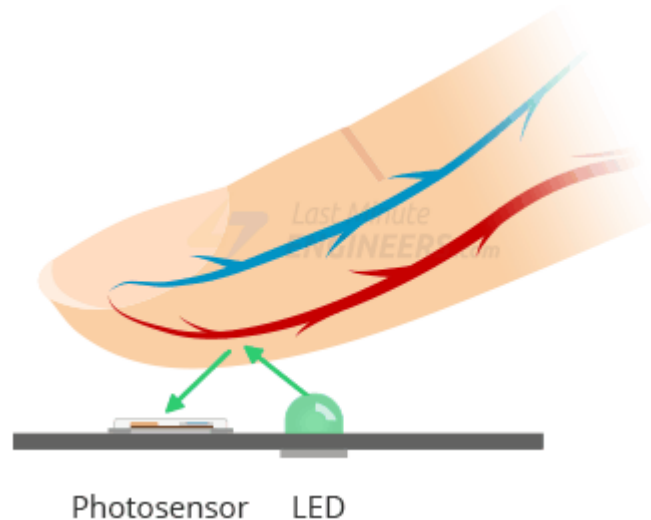


Outline

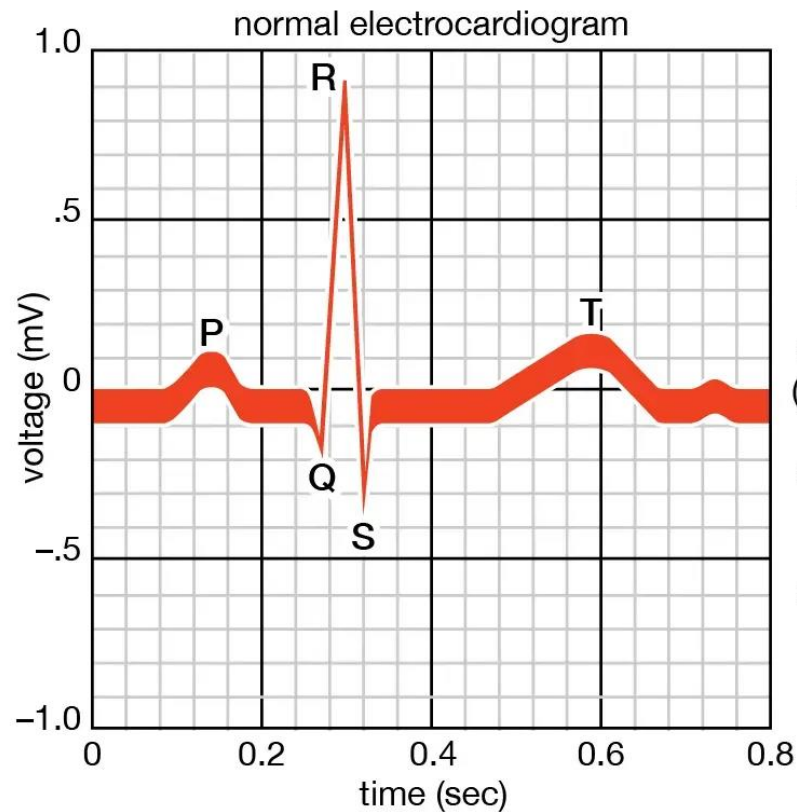
- Comparators (and nRF implementations)
- General ADC Design
- nRF ADC Implementation
- **Heartbeat Sampling**

Pulse sensor

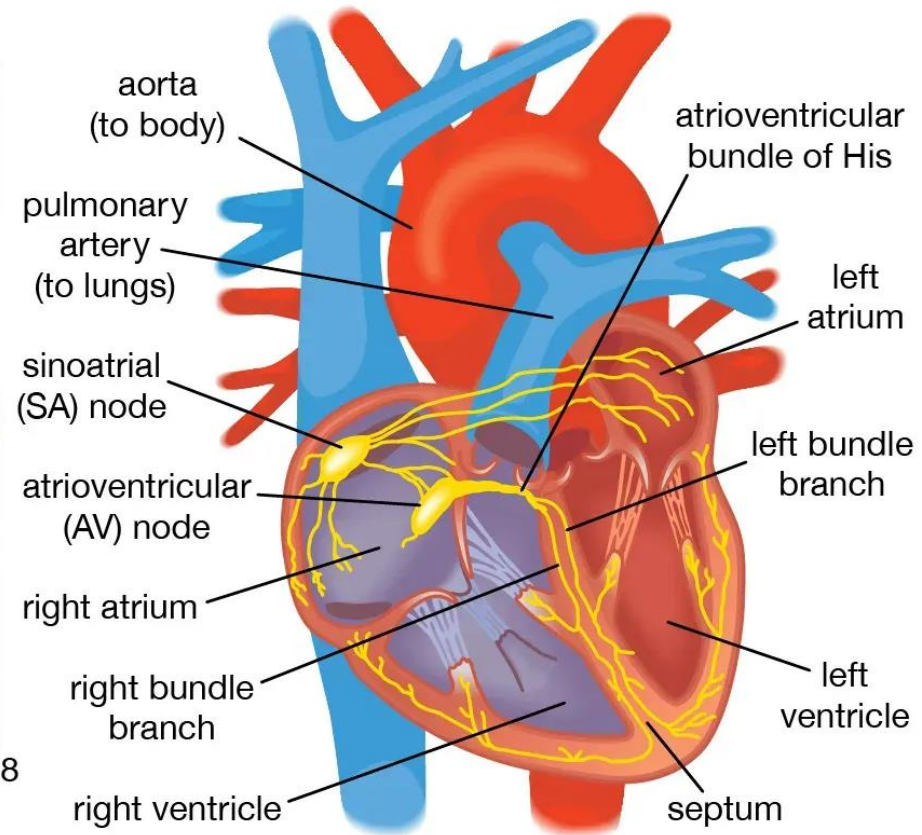
- Photo sensor that provides an analog output
- When pressed tightly against a finger, the blood flow is detectable and corresponds with your pulse
 - Counting peak rate gives heartrate



How quickly do we need to sample?



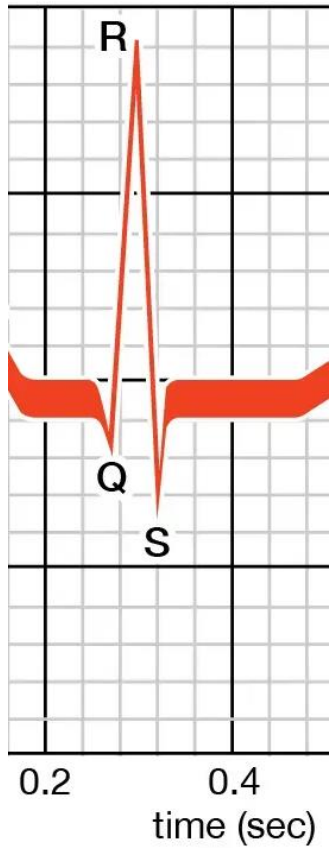
© 2013 Encyclopædia Britannica, Inc.



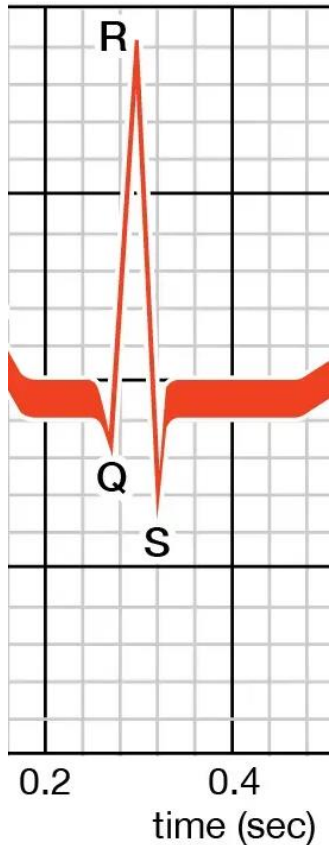
- Goal: sample fast enough to catch the “R” pulse
 - Need at *least* two samples per duration of R pulse

Determining sampling rates

- **Do the math: what sampling rate is needed?**

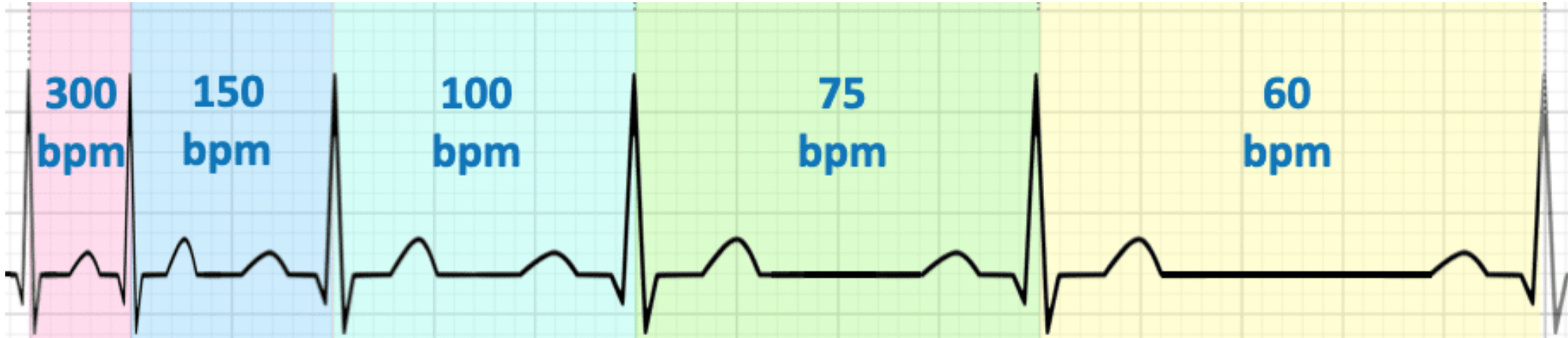


Determining sampling rates



- Do the math: what sampling rate is needed?
- Each box is about 0.04 seconds $(0.4-0.2)/5$
- So at least 2 samples / 0.04 seconds
 - At least 50 samples per second (i.e., 50 Hz)
- More is better though, especially for finding the top
 - As an engineer, I'd give a healthy margin on this
 - 250 Hz should be plenty (5 samples per R pulse)

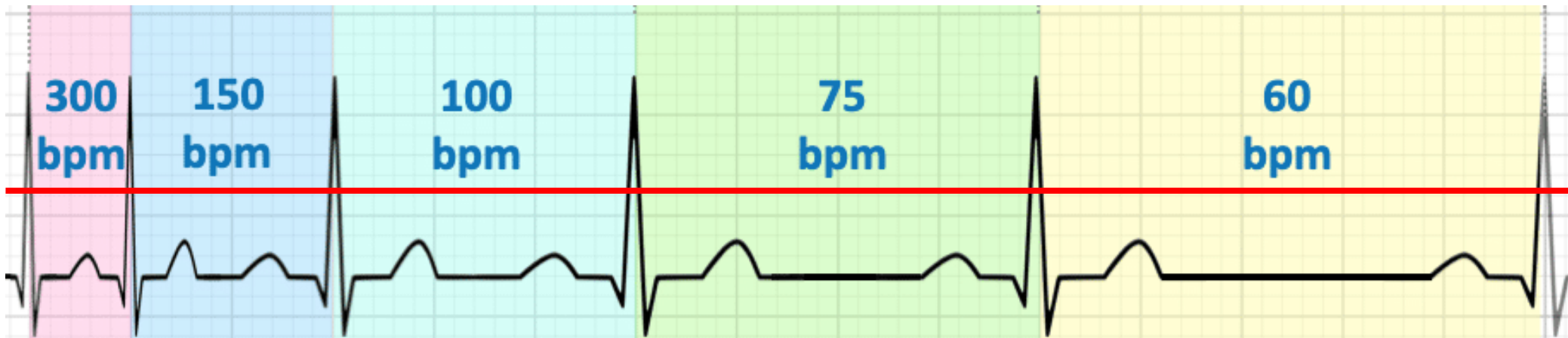
What do you do with the data?



- Measure time from peak-to-peak to determine pulse rate
 - (Hearts normally stay constant for a while, not like the diagram above)
- Mostly-equivalent alternative:
 - Count number of peaks per some unit of time
 - Measure for 10 seconds and count number of peaks in that period
- Both of these will require a timer peripheral too

Comparator would also work fine for this

- We needed to sample at 250 Hz to collect sufficient data to detect peaks
 - But we didn't really care about the *rest* of the data, just the peaks
- Instead, we could use a comparator to generate interrupts
 - When signal goes above a threshold, count that as a peak detected
 - Add a timer peripheral to get the pulse rate



Break + Design question

- How many analog samples can the Microbit hold?
 - Assumptions:
 - Only store samples in RAM (128 KB of total RAM on nRF52833)
 - 5 KB for Data (apart from sample storage)
 - 4 KB for Heap
 - 8 KB for Stack

Break + Design question

- How many analog samples can the Microbit hold?
 - Total Memory: 128 KB RAM
 - @ 2 bytes per sample = 64,000 samples
 - Available Memory: $128 \text{ KB} - (5 + 4 + 8) \text{ KB} = 111 \text{ KB RAM}$
 - @ 2 bytes per sample = 55,500 samples
 - Are they packed or padded to 16-bit?
 - Could double the samples at the trade of less resolution and increased processing
 - Note: audio sampling rate is typically 44.1 kHz -> 1.25 seconds of audio

Outline

- Comparators (and nRF implementations)
- General ADC Design
- nRF ADC Implementation
- Heartbeat Sampling