# Lecture 08
# Analog Input

CE346 – Microprocessor System Design

Branden Ghena – Fall 2022

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

# Administrivia

- Project proposals are due tonight!
  - Plan is to get you project proposal feedback over the next several days

- Remember to answer the post-lab questions on Canvas!
  - You can do them late if you forget, but don’t forget

# Today’s Goals

- Explore methods for sensing analog signals
  - Comparators
  - Analog-to-Digital Converters

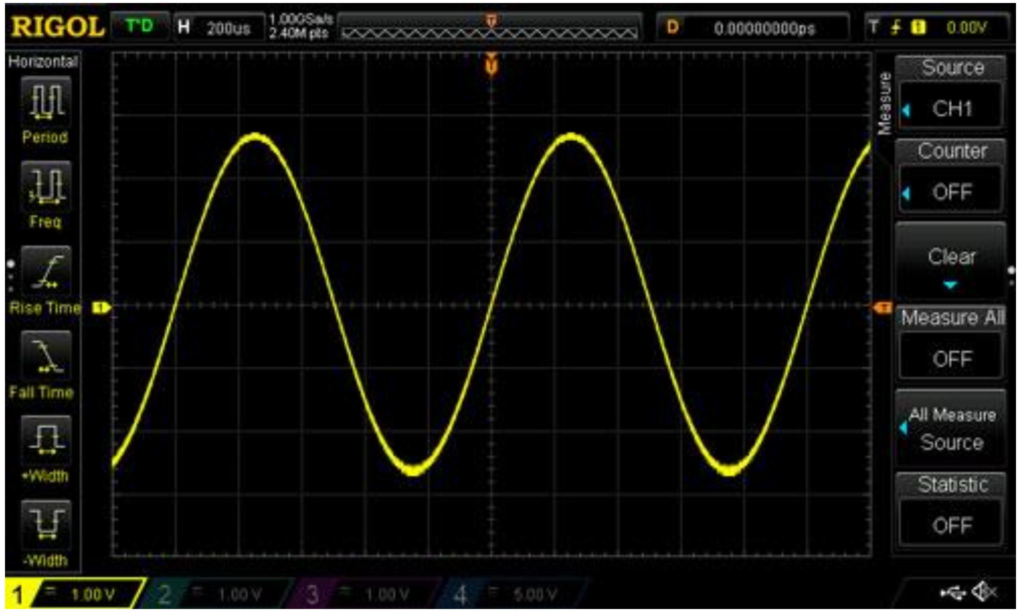
- Discuss nRF implementation of these peripherals

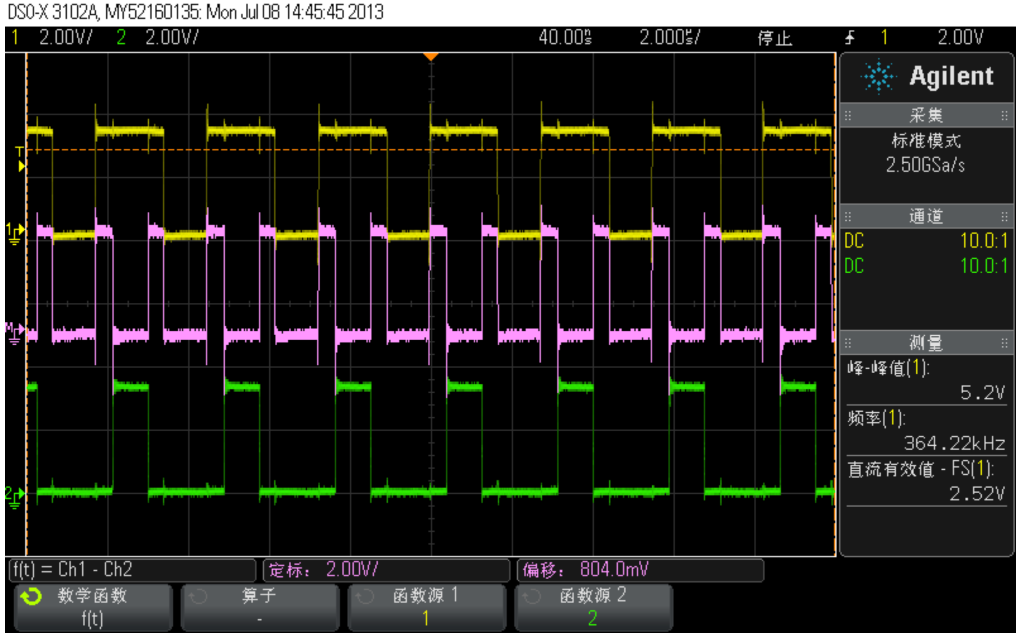
# Outline

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

# 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)

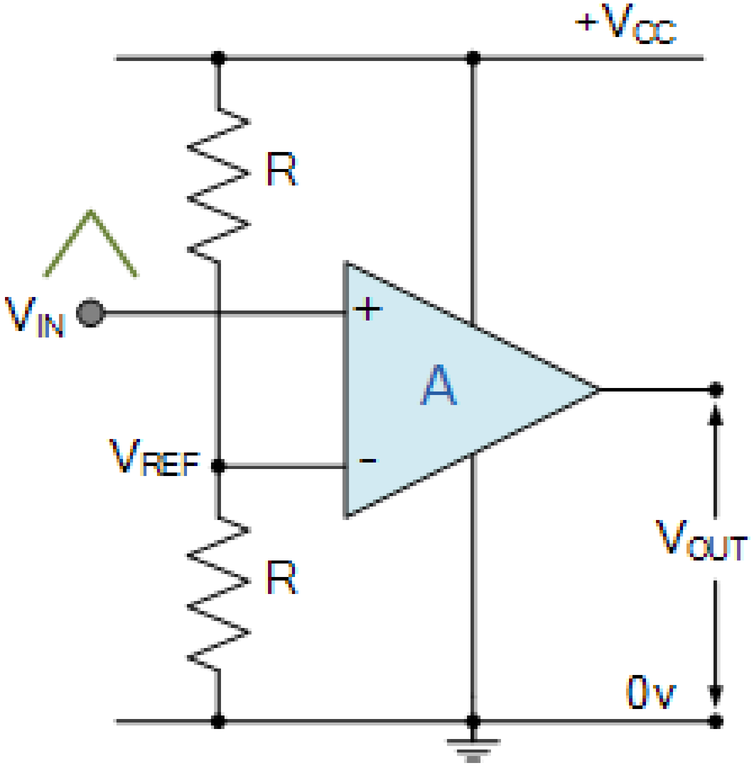
  **Determination is done by a Comparator**

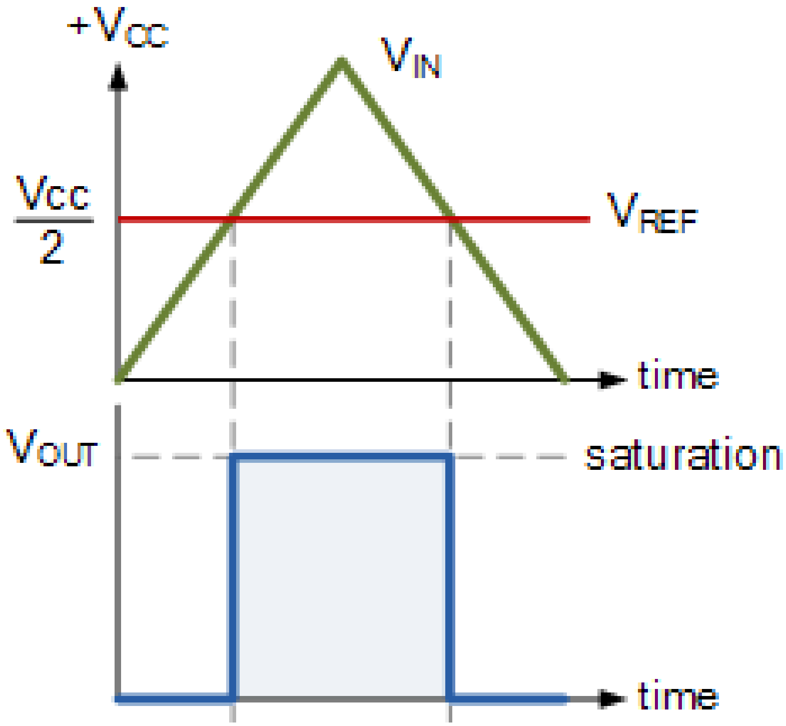
# General comparator design

- 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
  - Low power



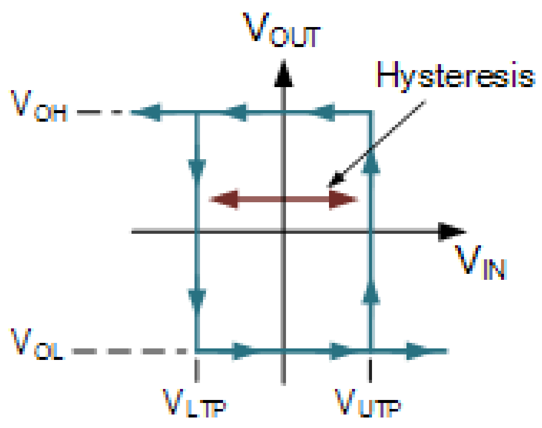
# Comparator design questions

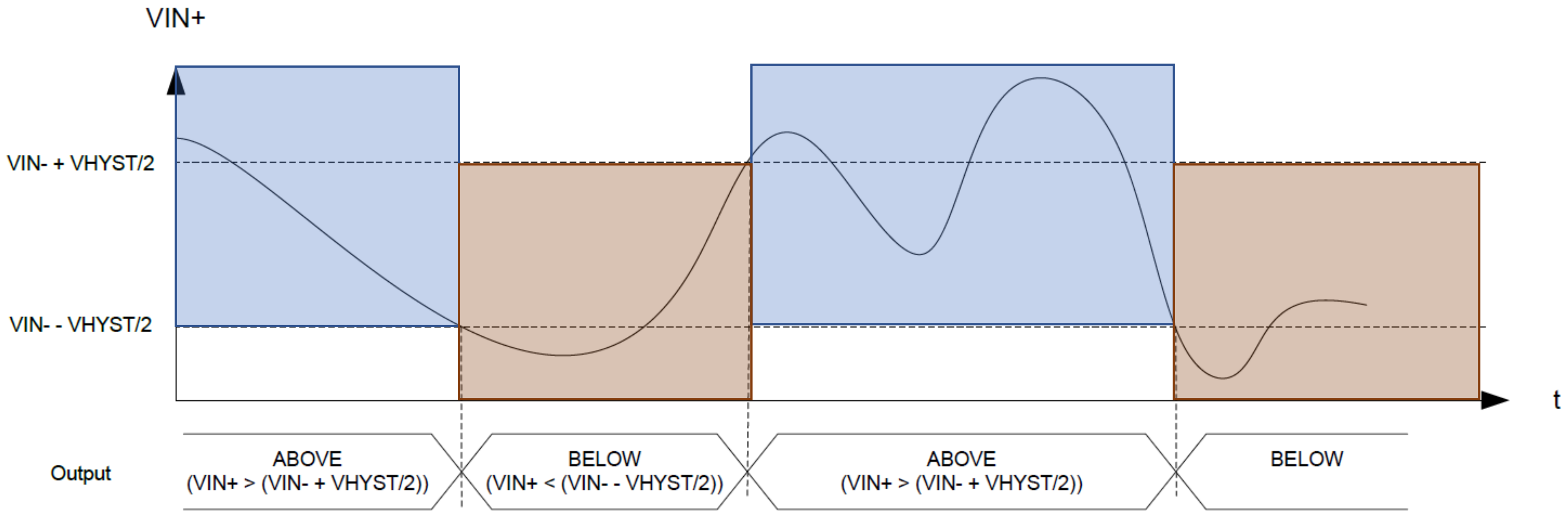
- What reference voltages are available?
  - A few internal voltages
  - Usually also allows external references from input pins

- When is an output generated?
  - Usually when status changes
    - Low-to-high, High-to-low, Both (like GPIO interrupts)

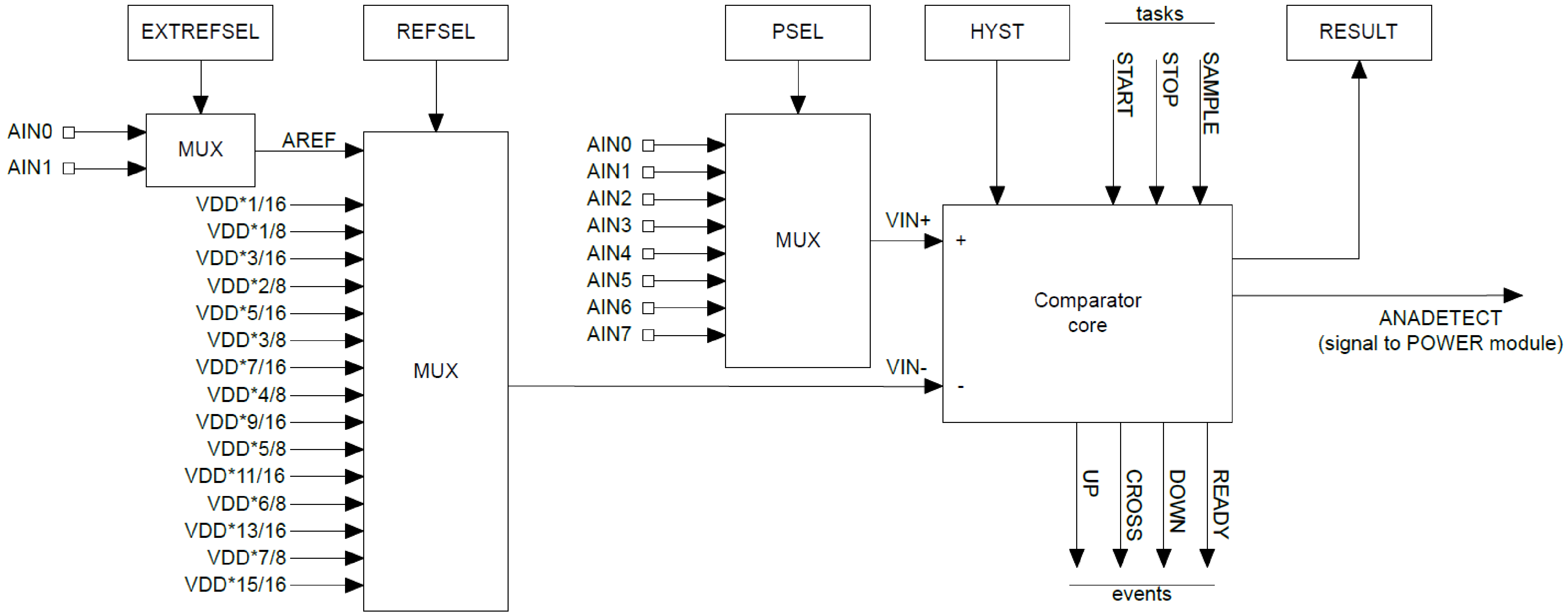
# Hysteresis

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

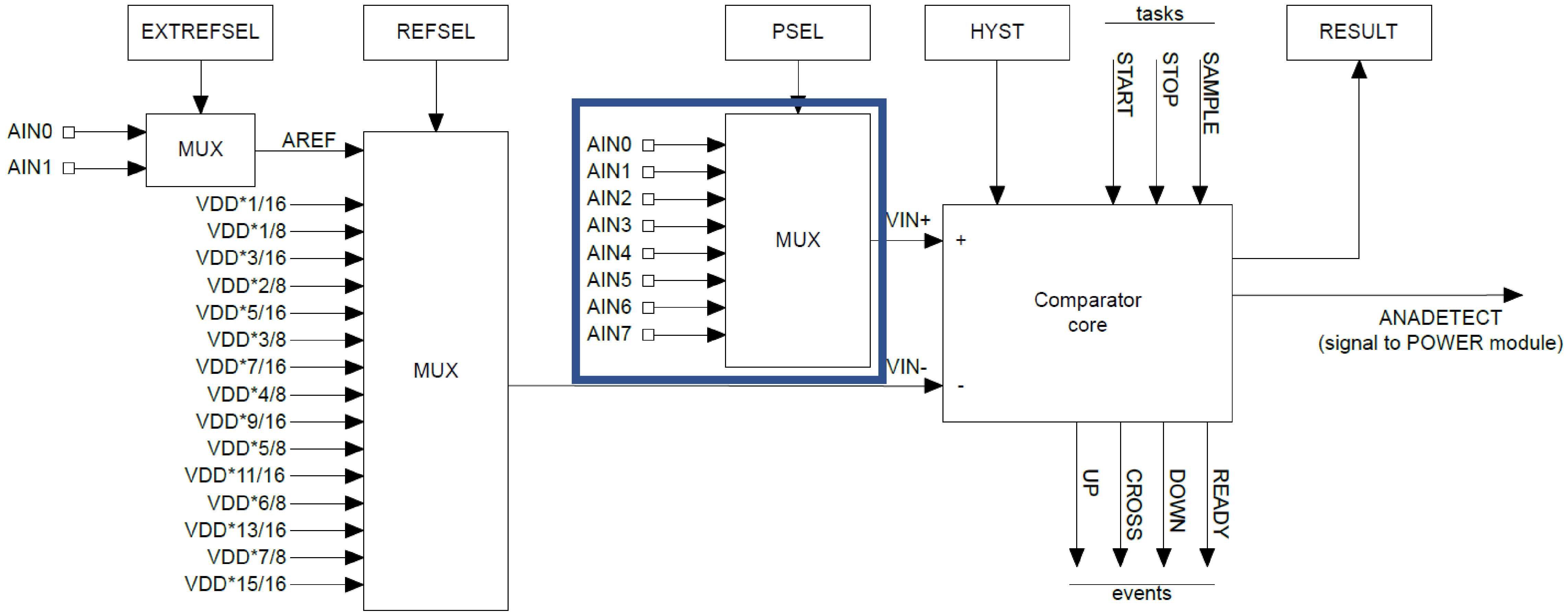


# nRF low-power comparator (LPCOMP)

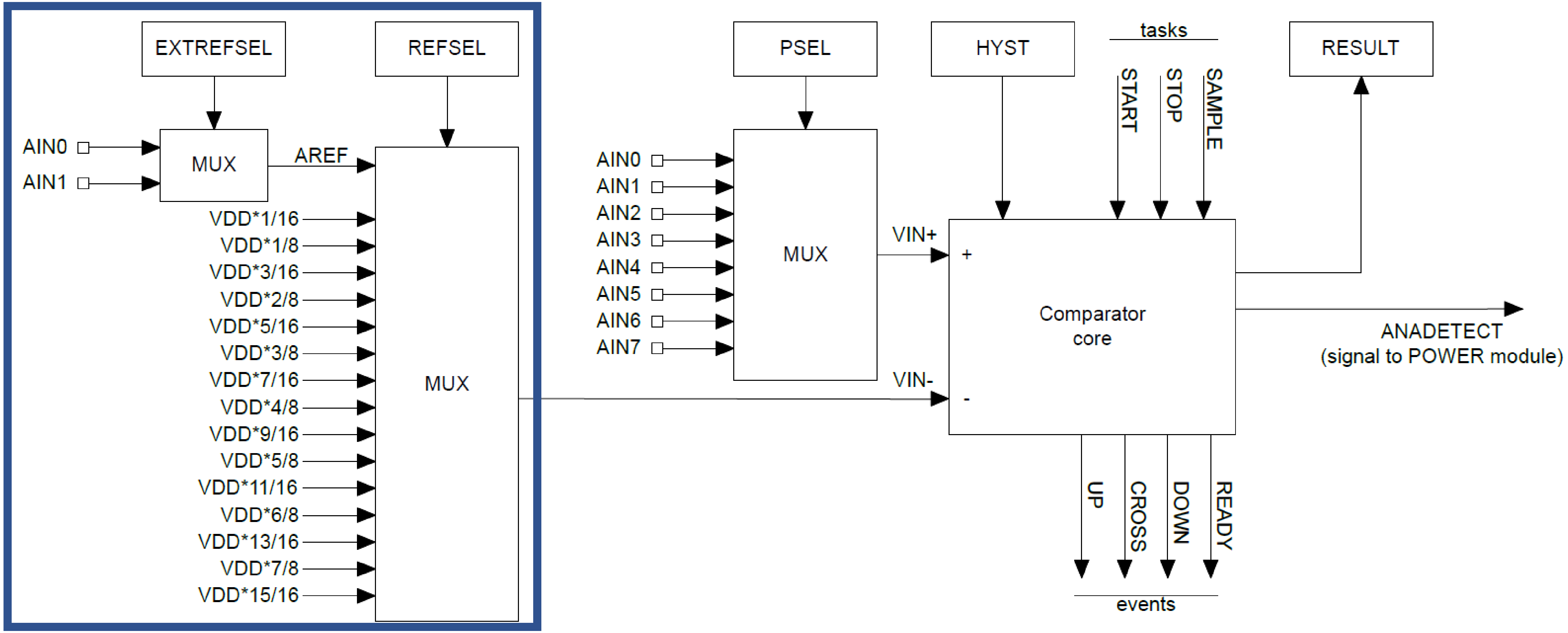


# nRF low-power comparator (LPCOMP)



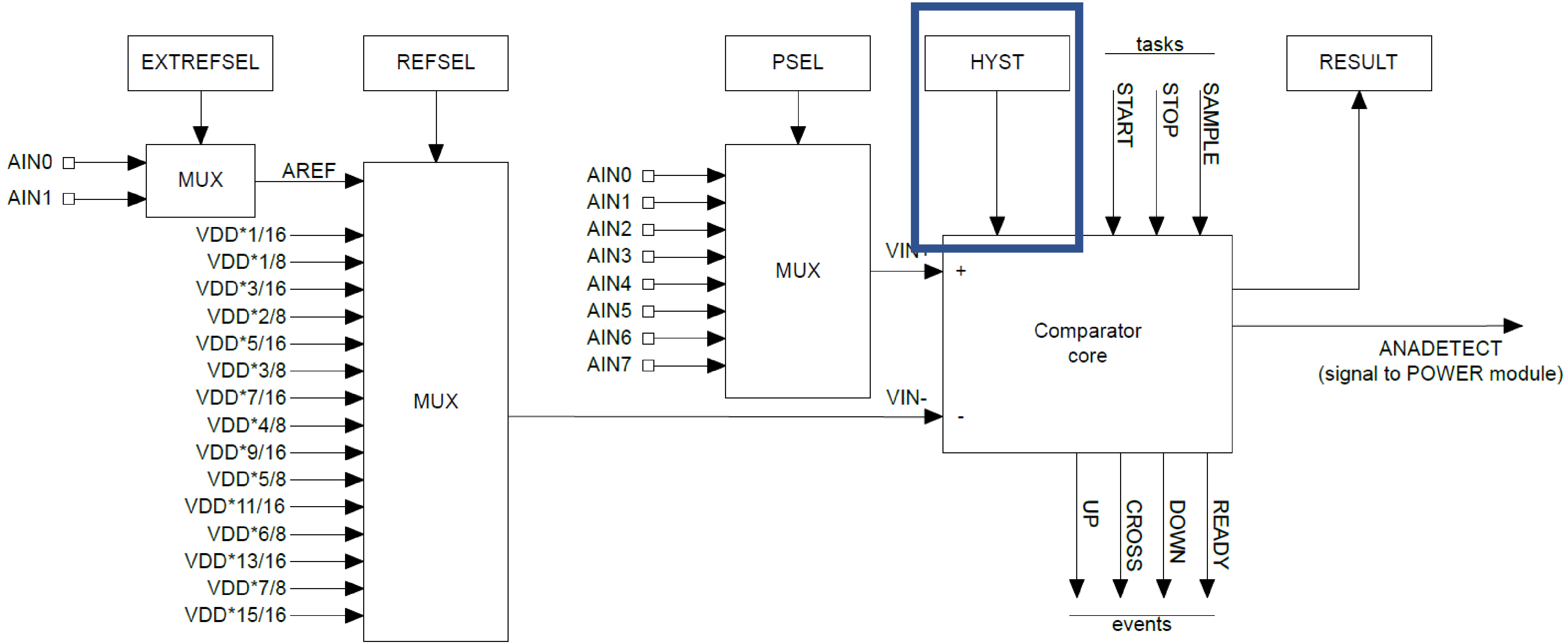

- Input: one of eight analog input pins

# nRF low-power comparator (LPCOMP)



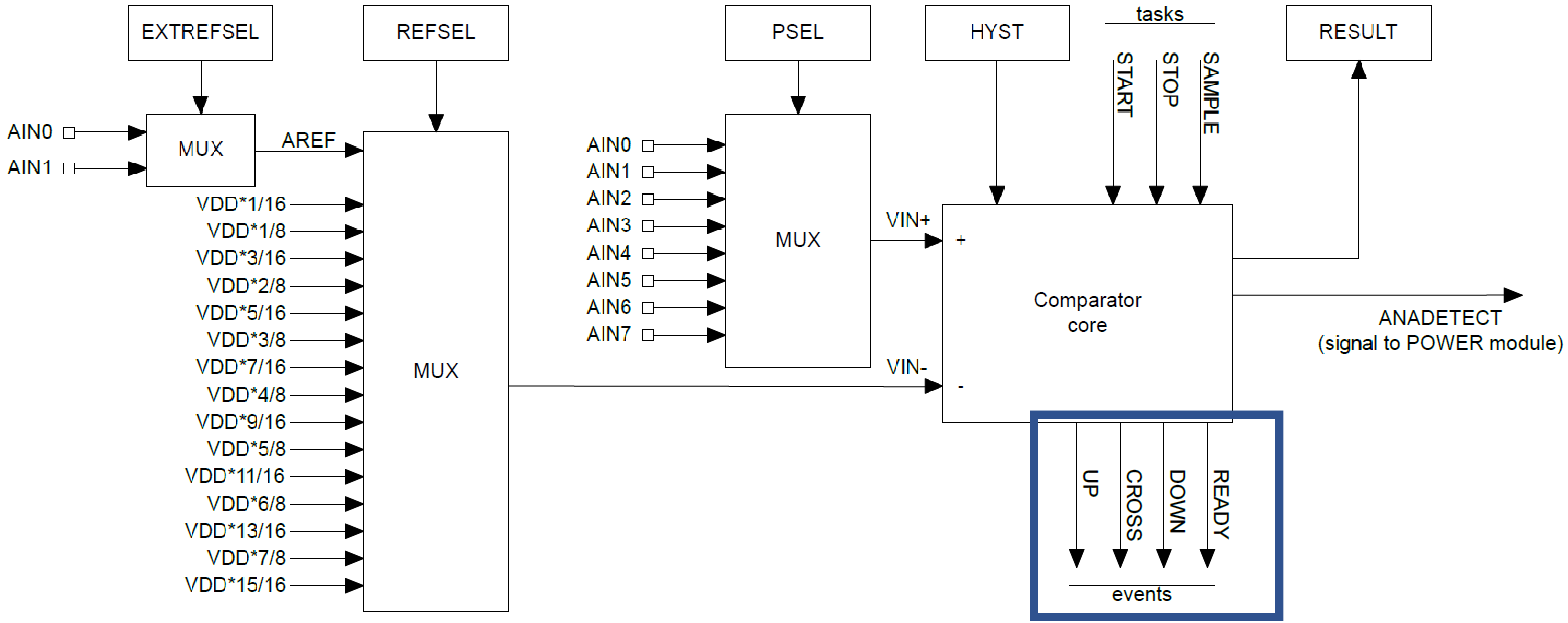

- Reference: one of two analog inputs or selection of VDD * N/16

# nRF low-power comparator (LPCOMP)



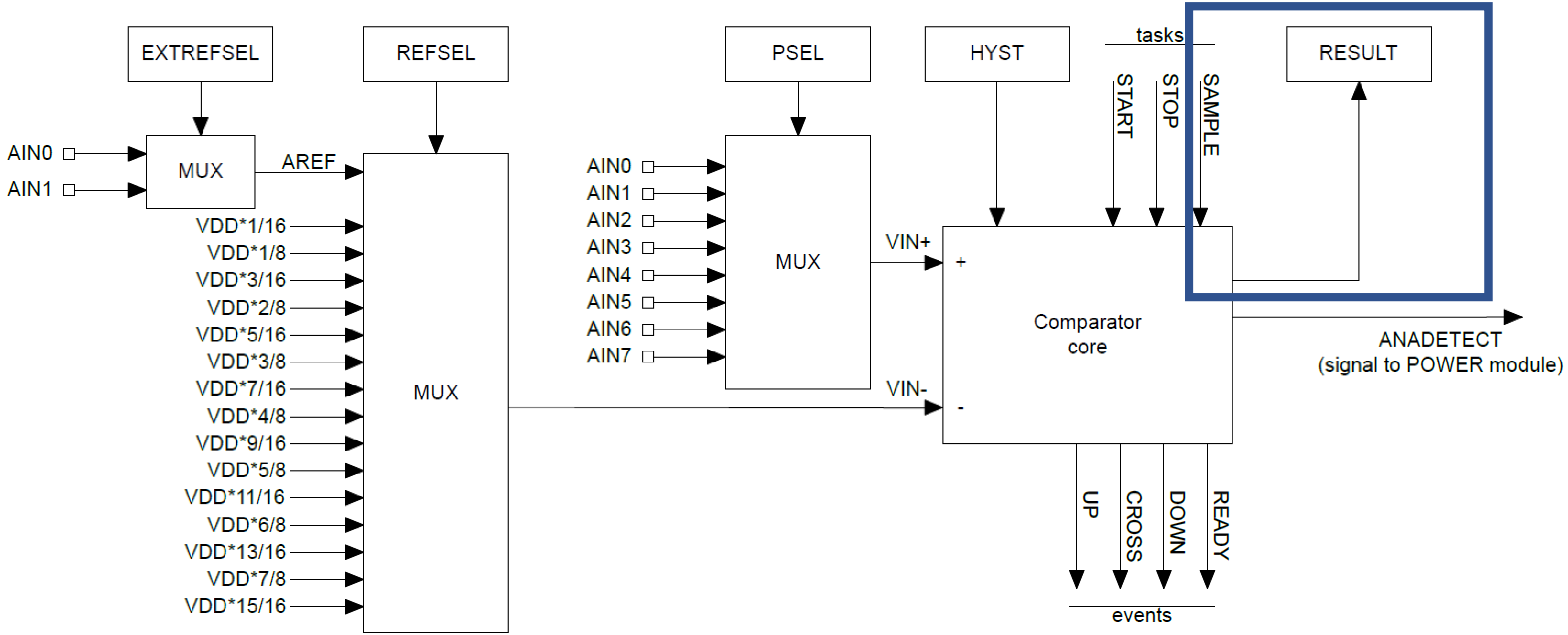

- Hysteresis: +/- 50 mV range around VIN- when enabled

# nRF low-power comparator (LPCOMP)



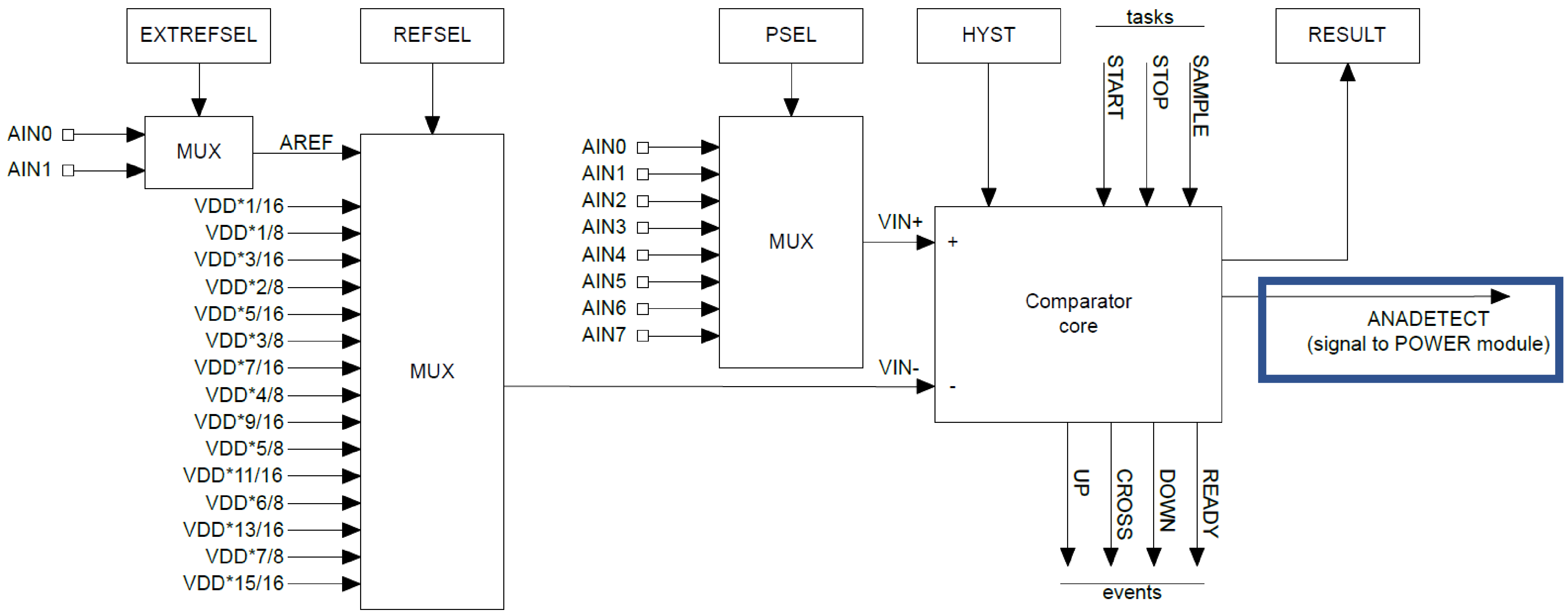

- Events: transition signals + ready (~150 μs)

# nRF low-power comparator (LPCOMP)



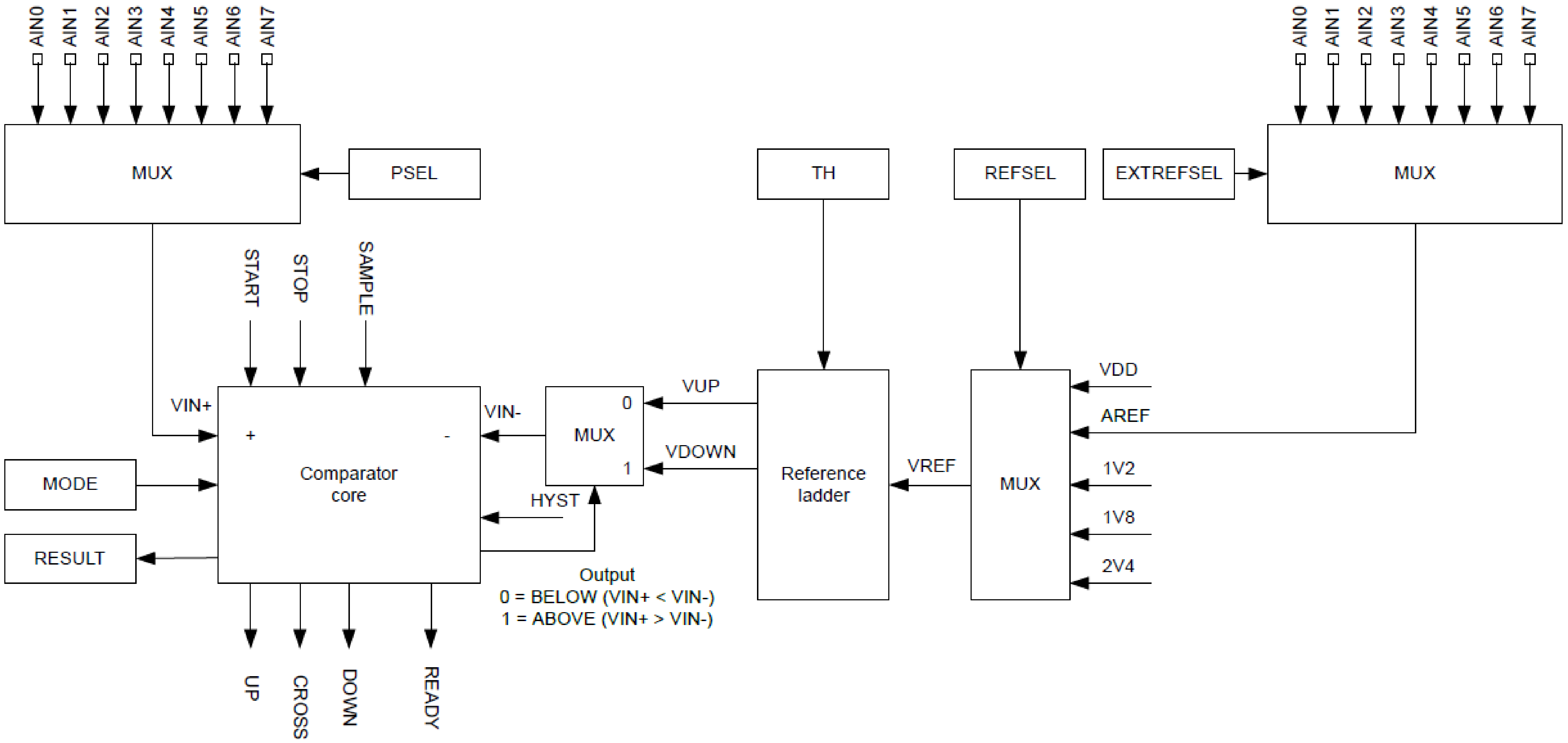

- Can also request what the current comparison state is (high/low)

# nRF low-power comparator (LPCOMP)




- Can be used for low-power wakeup of microcontroller

# nRF COMP peripheral



# 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 μs COMP: 0.1-0.6 μs (depending 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 VDD?**

# Break + Question: Internal reference voltages

- **Why have internal voltage references other than VDD?**

  - What if want 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

# Interacting with analog signals

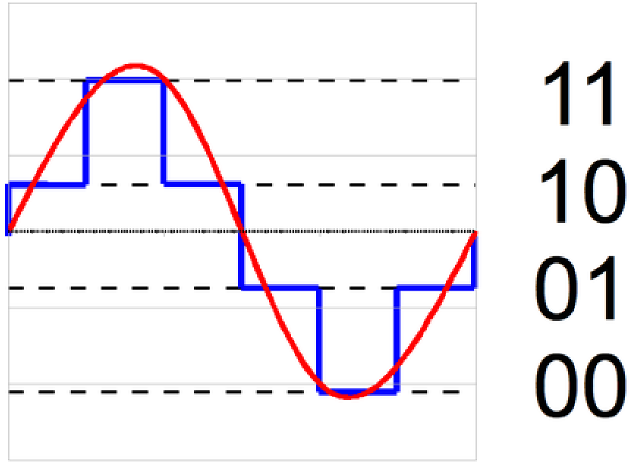
- Microcontrollers are inherently digital

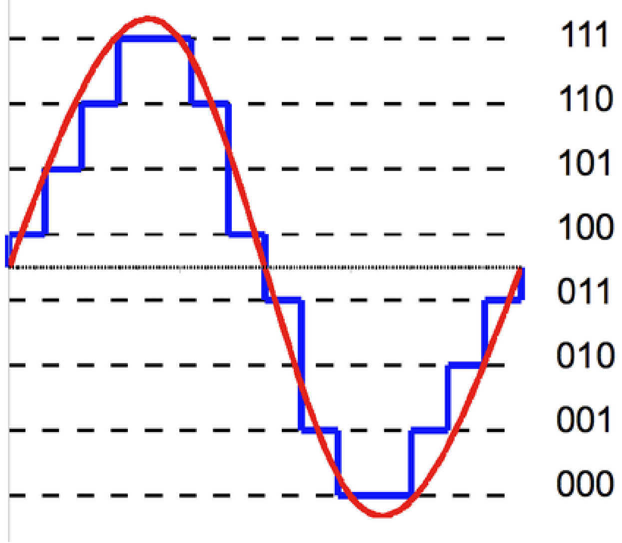
- Need a method for translating analog signal into a digital one

- 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



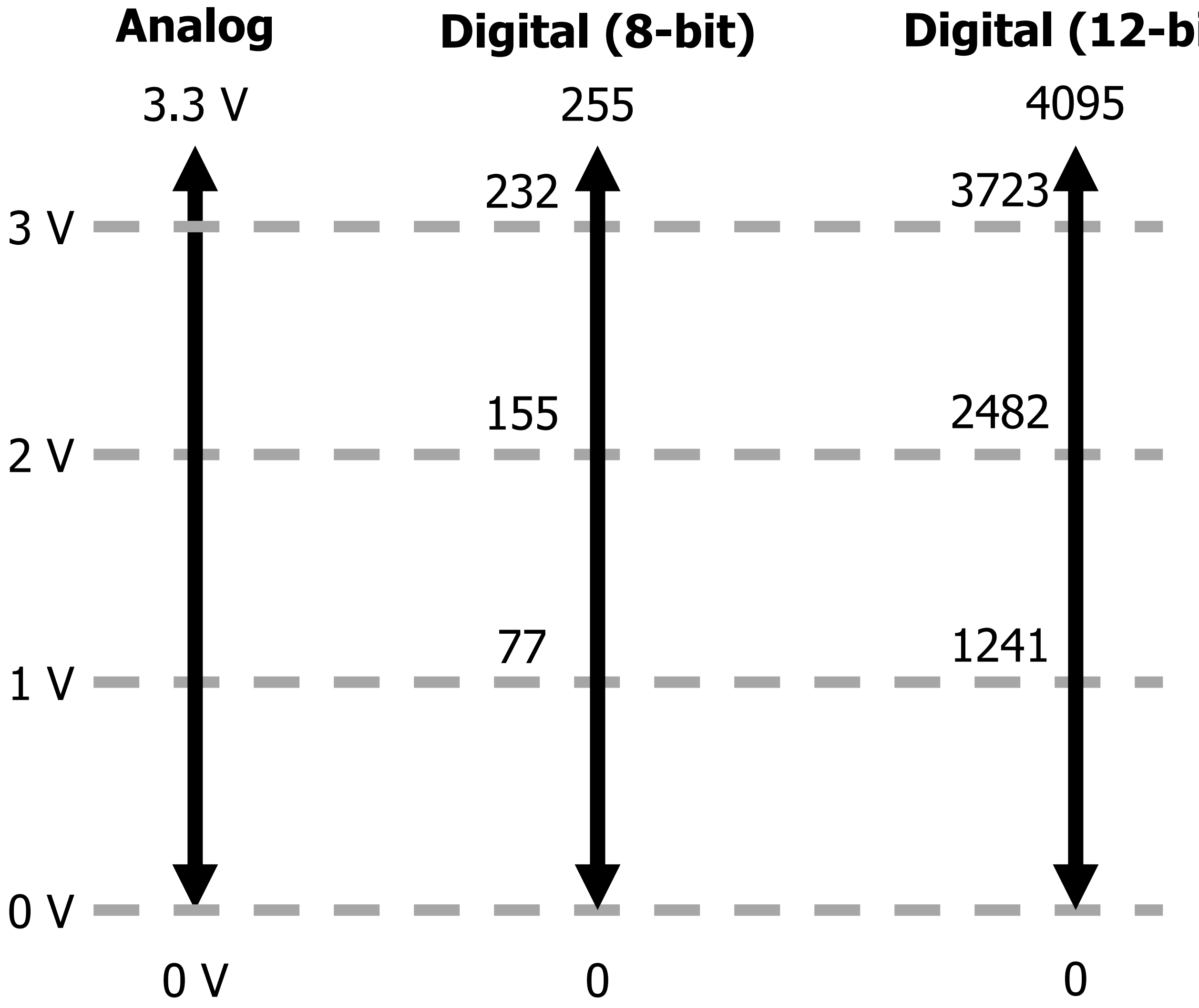




- Analog voltages are represented by discrete voltage levels

- 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

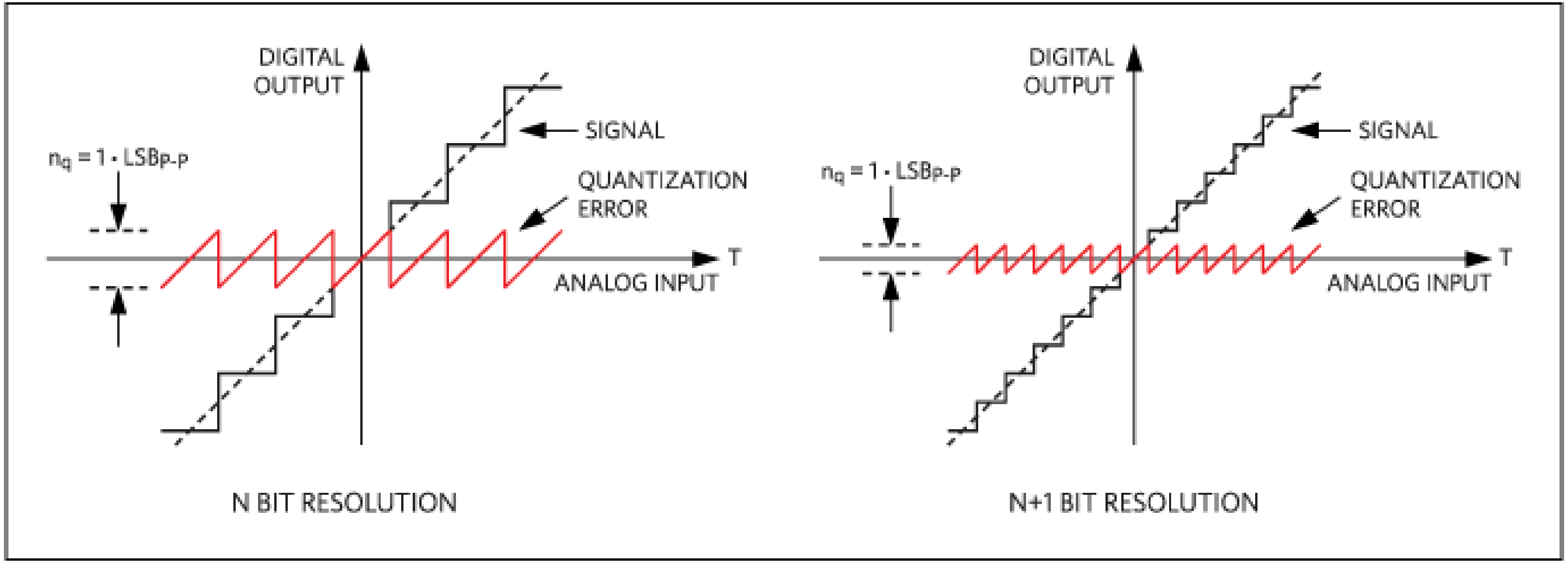
# Translating voltage and ADC counts




$$Value = \frac{V_{IN}}{V_{REF}} * \left(2^{Resolution} - 1\right)$$

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

# Quantization error



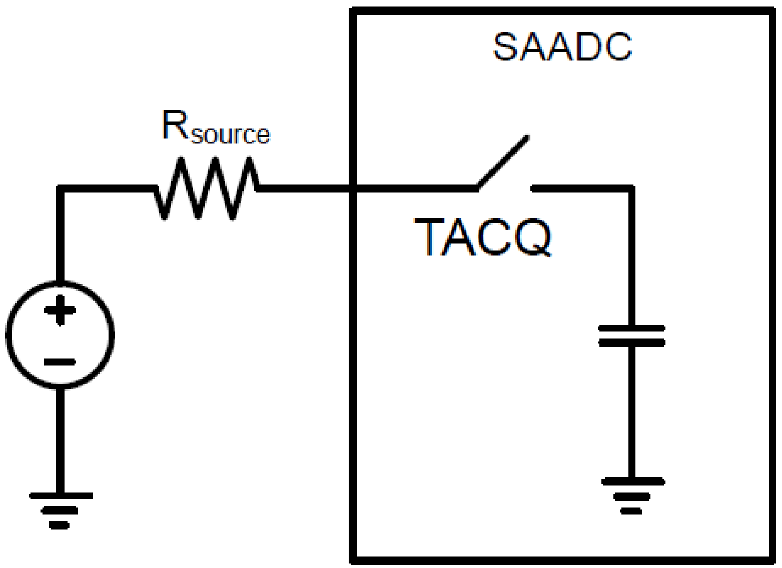

- Resolution choice determines magnitude of error
  - Each extra bit halves the magnitude of error

# 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



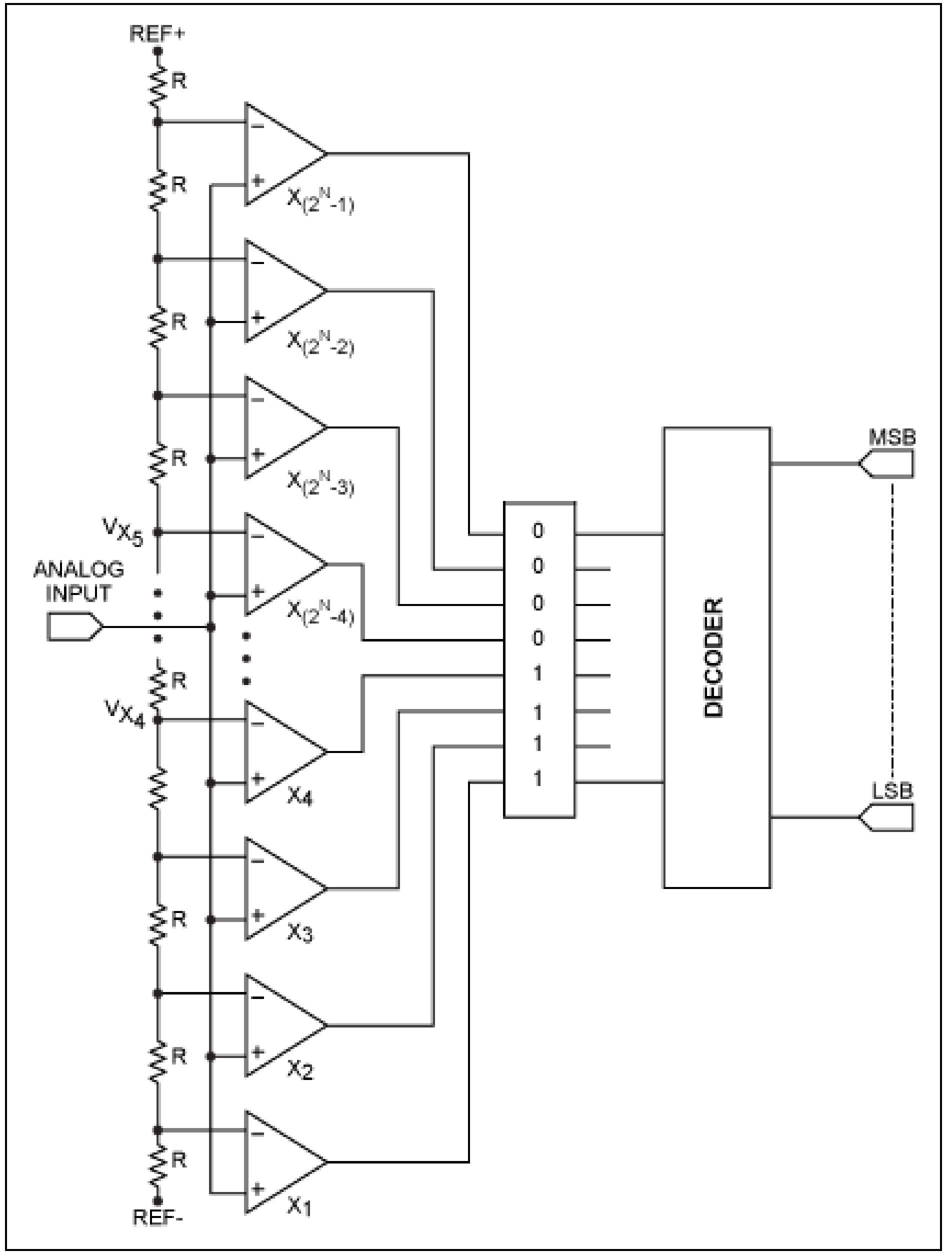

2. Conversion
    - Determine which digital value the read signal corresponds to

# Direct-conversion ADC

- Chain comparators together
  - Each with a separate reference voltage

- Digital value determined immediately
  - Also known as “Flash” ADCs

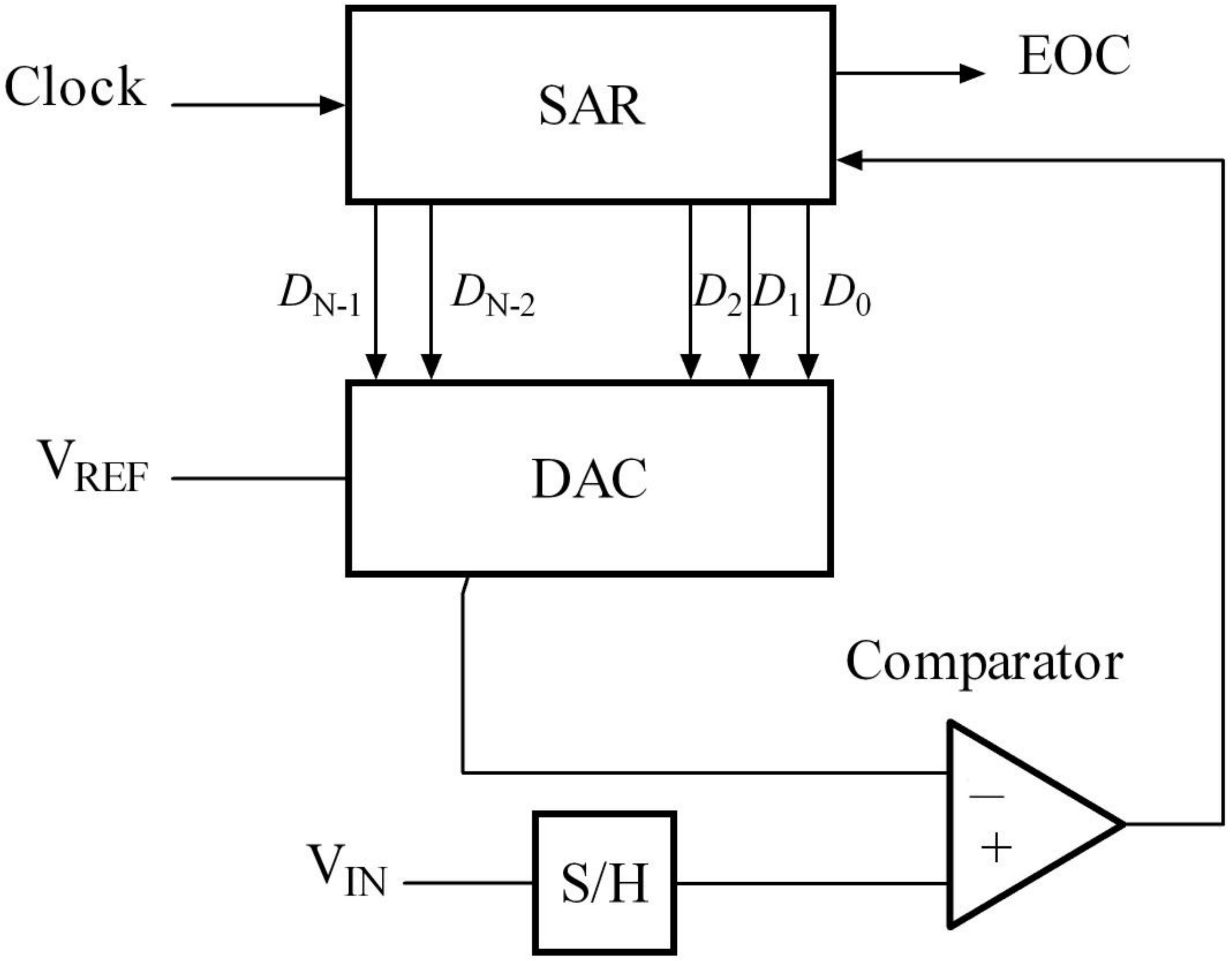
- Downside: needs $2^n$-1 comparators
  - Reserved for expensive applications



# Successive-Approximation ADC

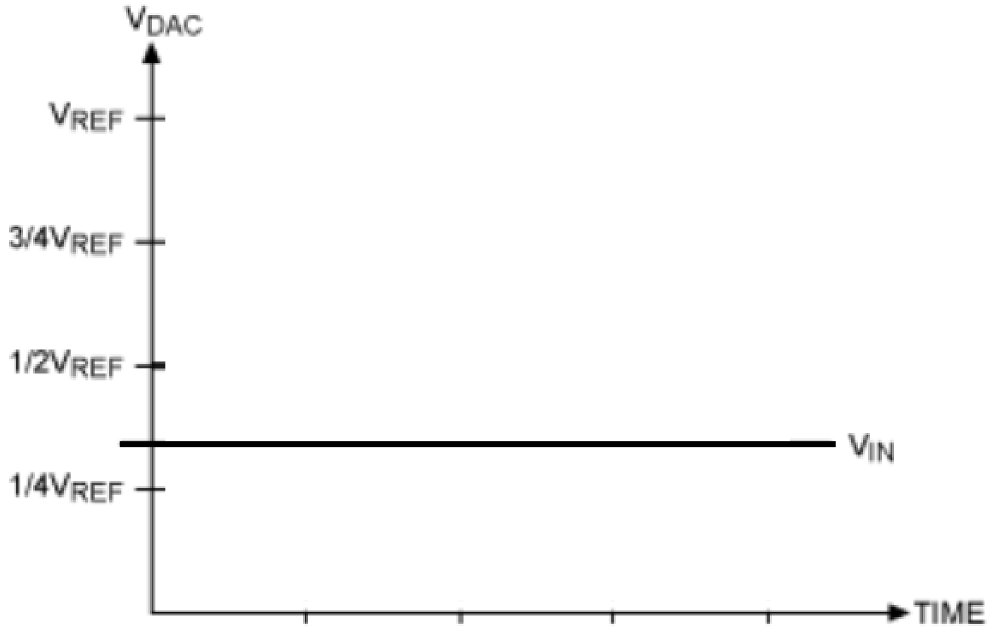
- Method: Binary Search
  - Compare signal to generated reference
  - Increase or decrease reference as needed
  - Repeat

- DAC creates reference (Digital-to-Analog Converter)
  - Final value of DAC is the ADC value



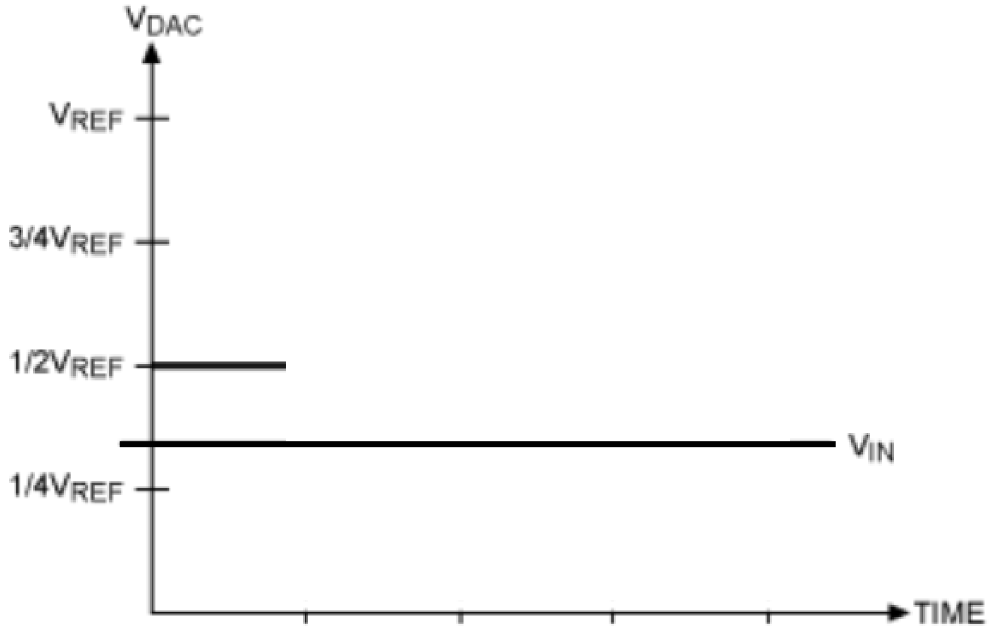
# Successive Approximation Example

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



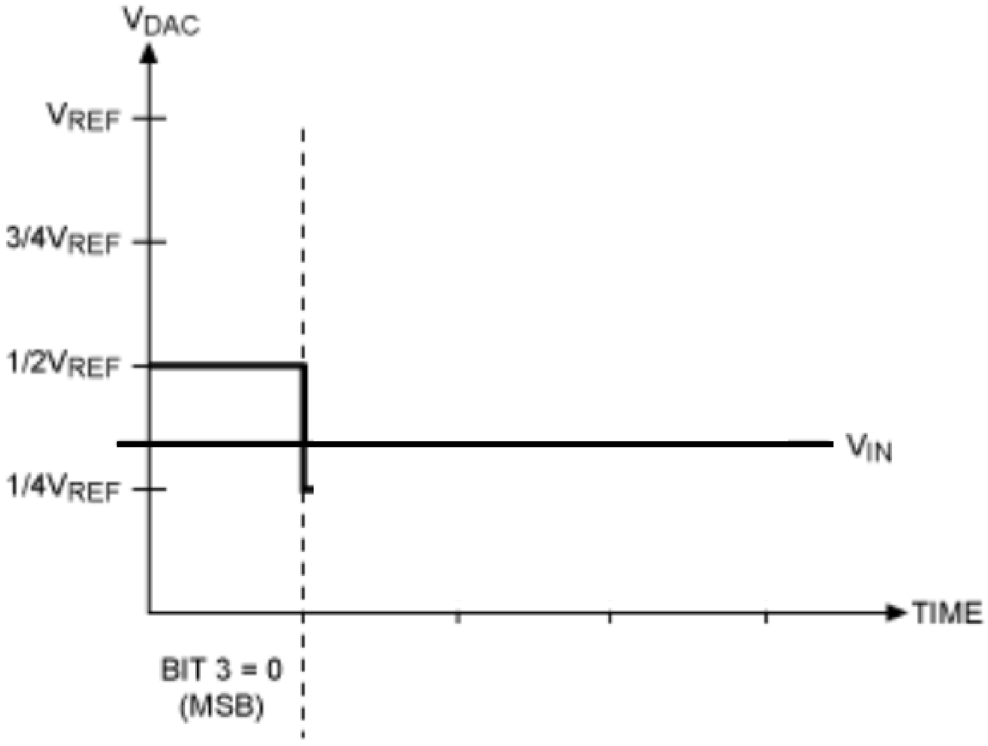
# Successive Approximation Example

1. Compare ½ $V_{REF}$ (0b<u>?</u>???) to $V_{IN}$
    - If $V_{IN}$ is greater, bit is 1. Else zero



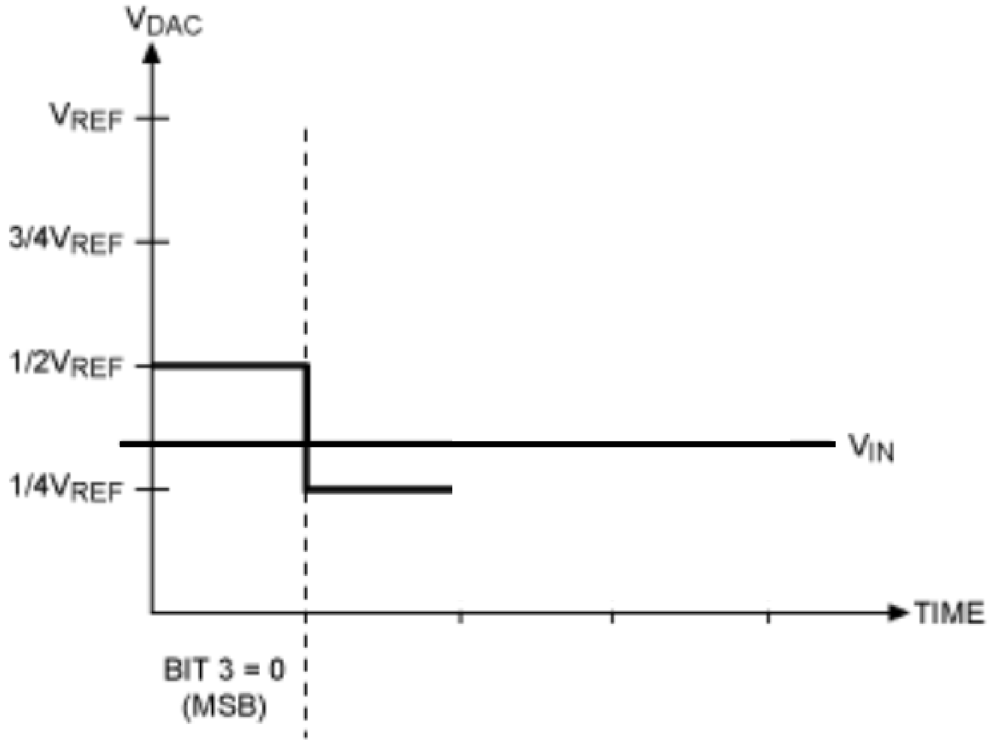
# Successive Approximation Example

1. Compare ½ $V_{REF}$ (0b**0**???) to $V_{IN}$
   - If $V_{IN}$ is greater, bit is 1. Else zero
   - $V_{IN}$ is less. So set that bit to zero



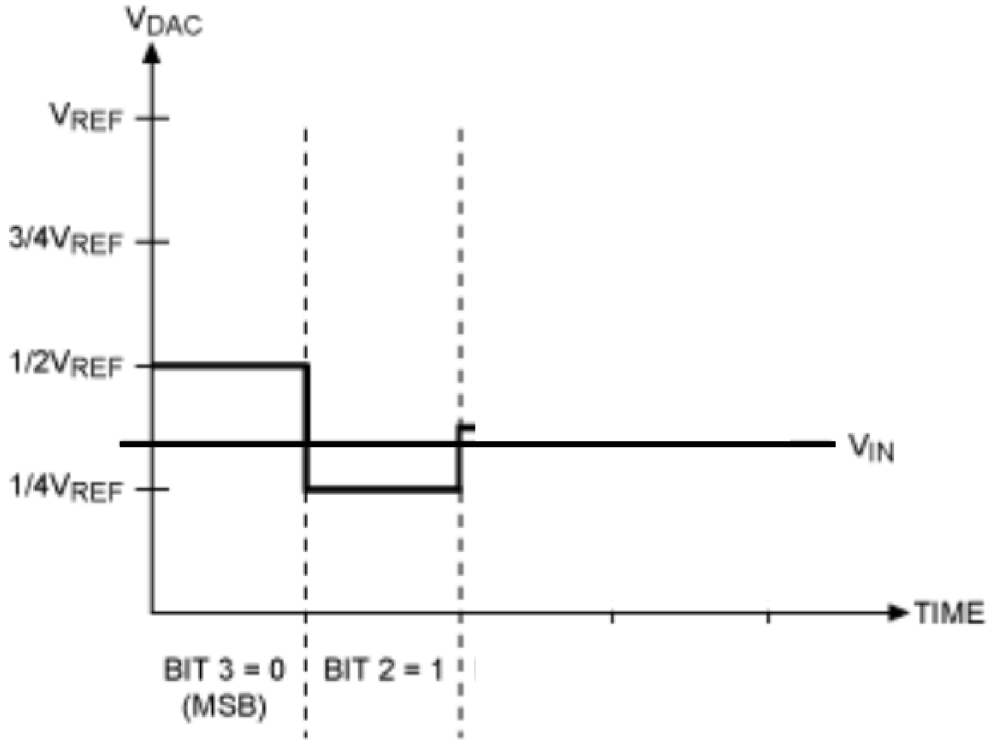
# Successive Approximation Example

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



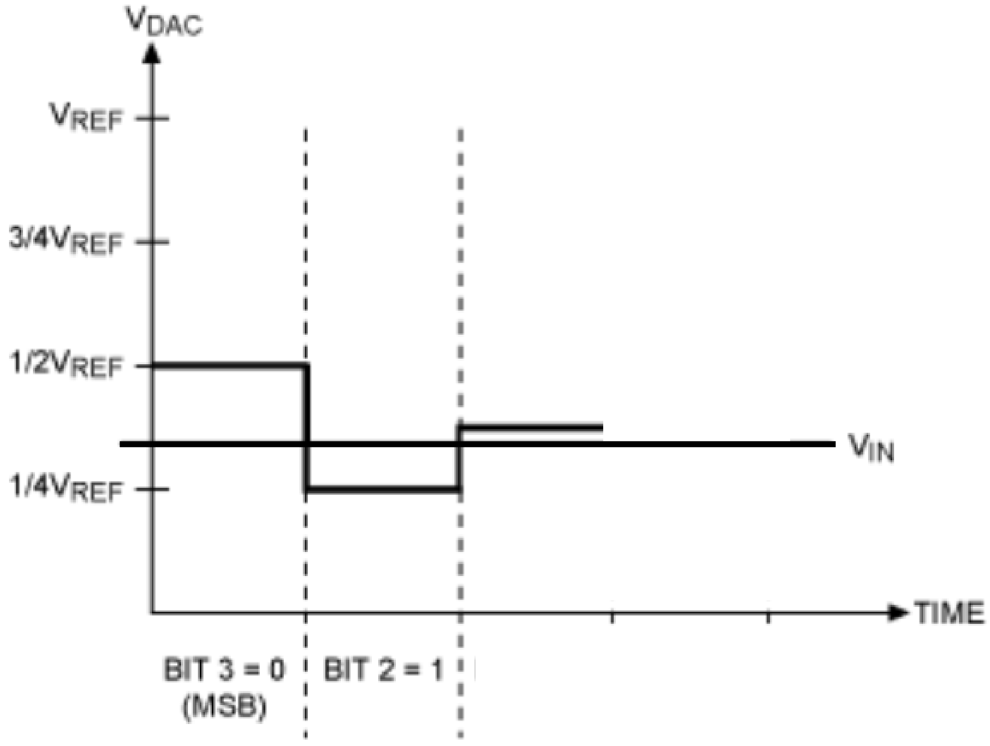
# Successive Approximation Example

2. Compare 1/4 $V_{REF}$ (0b**01**??) to $V_{IN}$
    - If $V_{IN}$ is greater, bit is 1. Else zero
    - $V_{IN}$ is greater. So set that bit to one



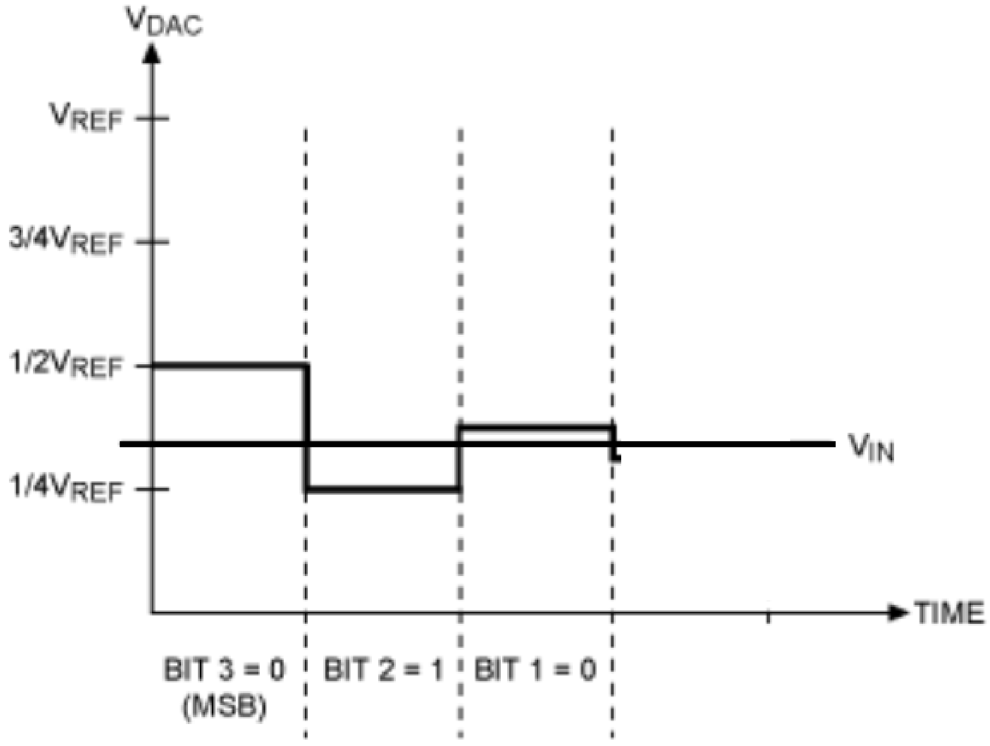
# Successive Approximation Example

3. Compare 3/8 $V_{REF}$ (0b**01**?̲?) to $V_{IN}$
   - If $V_{IN}$ is greater, bit is 1. Else zero



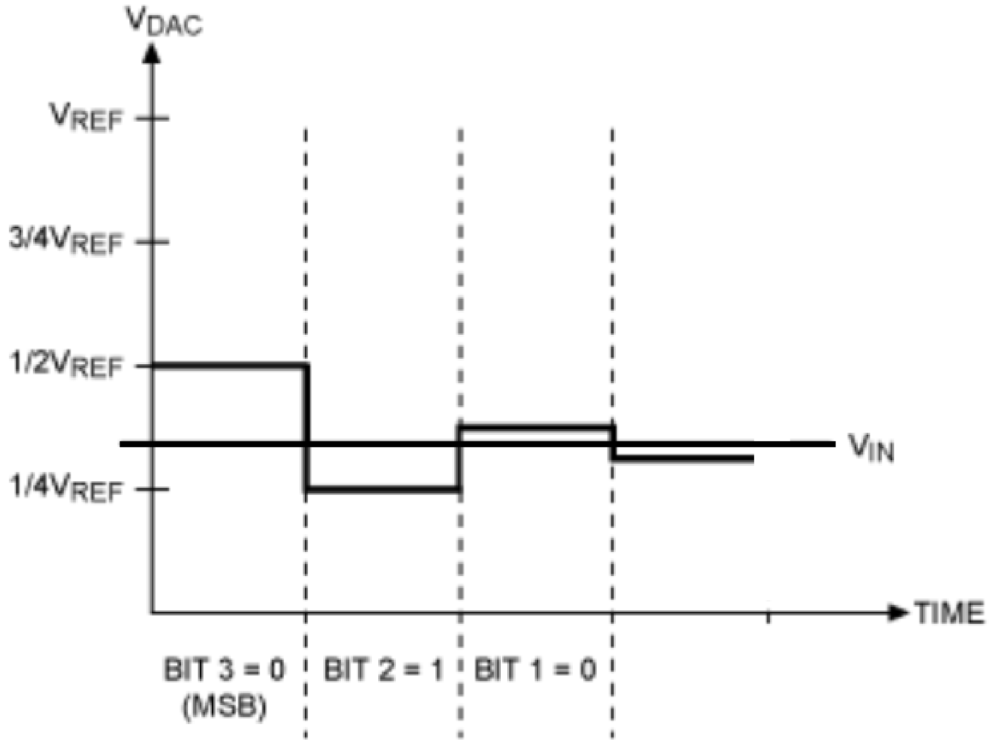
# Successive Approximation Example

3. Compare 3/8 $V_{REF}$ (0b**010**?) to $V_{IN}$
    - If $V_{IN}$ is greater, bit is 1. Else zero
    - $V_{IN}$ is less. So set that bit to zero



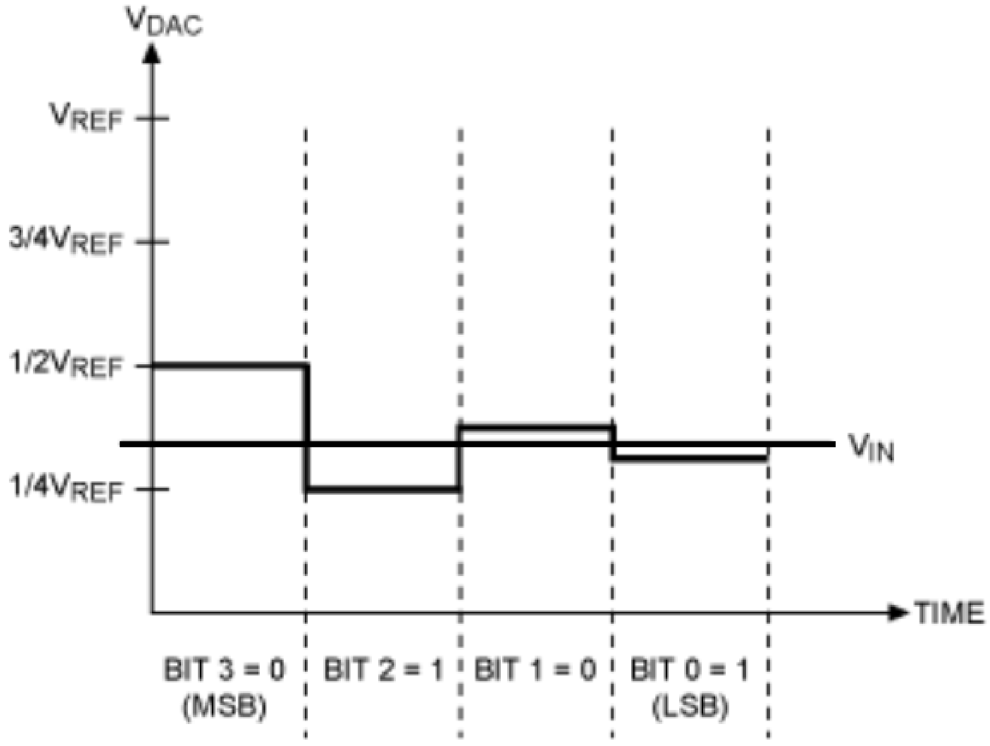
# Successive Approximation Example

4. Compare 5/16 $V_{REF}$ (0b**010**<u>?</u>) to $V_{IN}$
   - If $V_{IN}$ is greater, bit is 1. Else zero



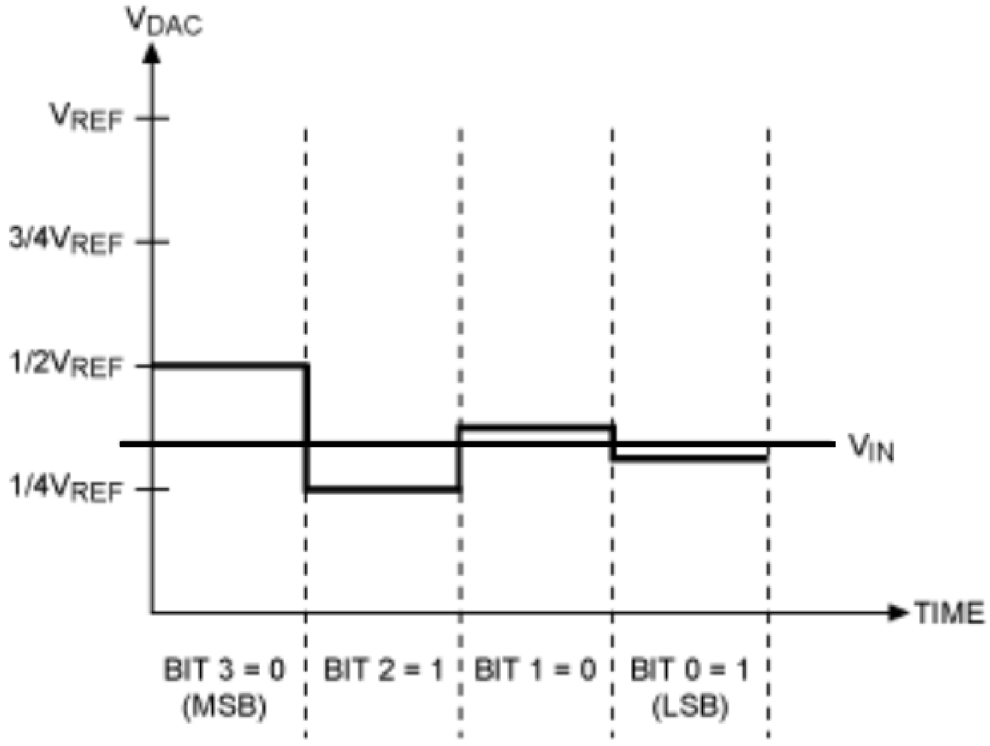
# Successive Approximation Example

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



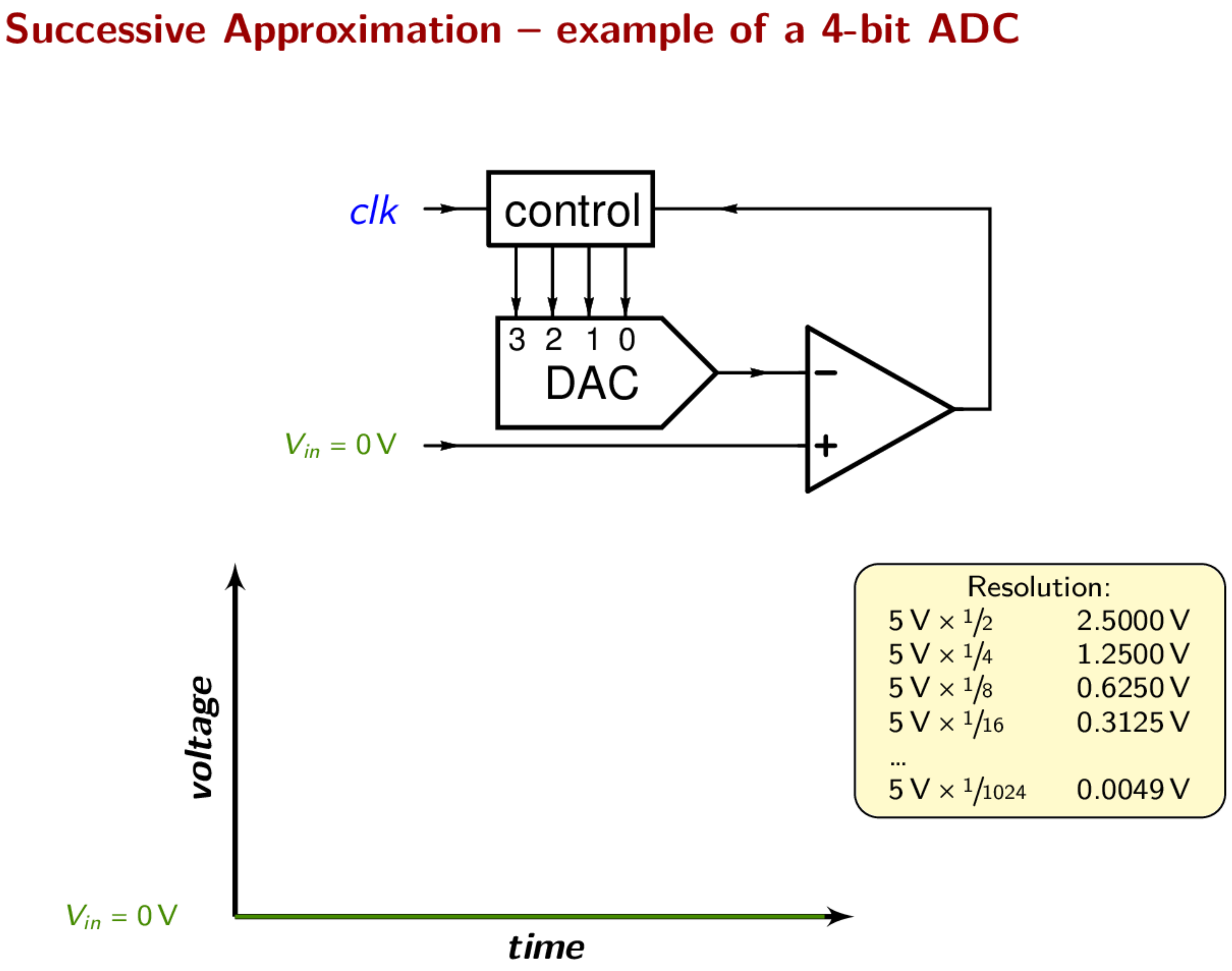
# Successive Approximation Example

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




https://en.wikipedia.org/wiki/Successive-approximation_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 probably 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

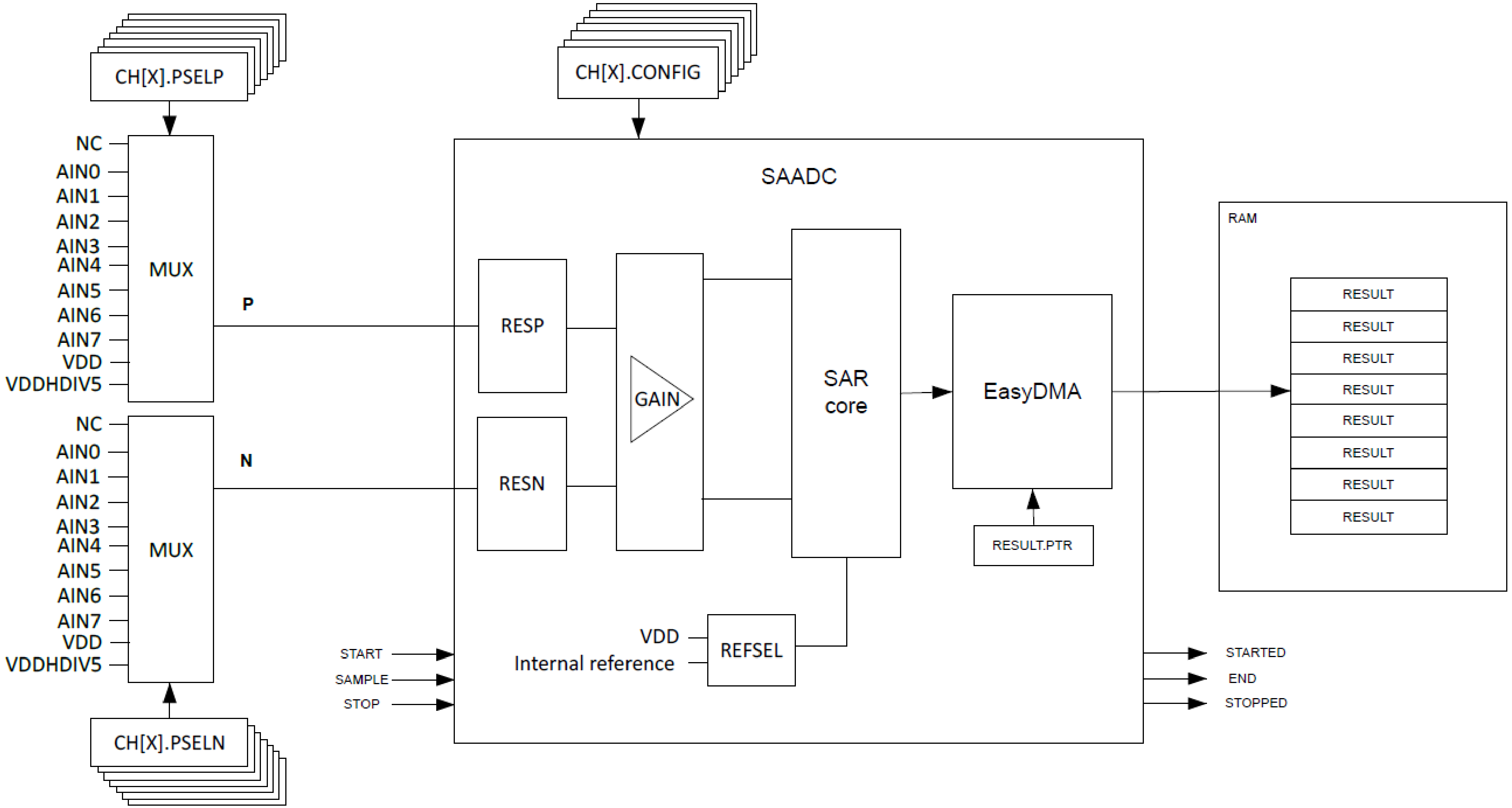
  - Temperature sensor probably doesn't need 16-bit ADC
    - Difference between 30.001°C and 30.002°C is usually irrelevant

  - Microphone might though!
    - Differences are audible until they are small enough
    - 16-bit or 24-bit tend to be common choices for audio data

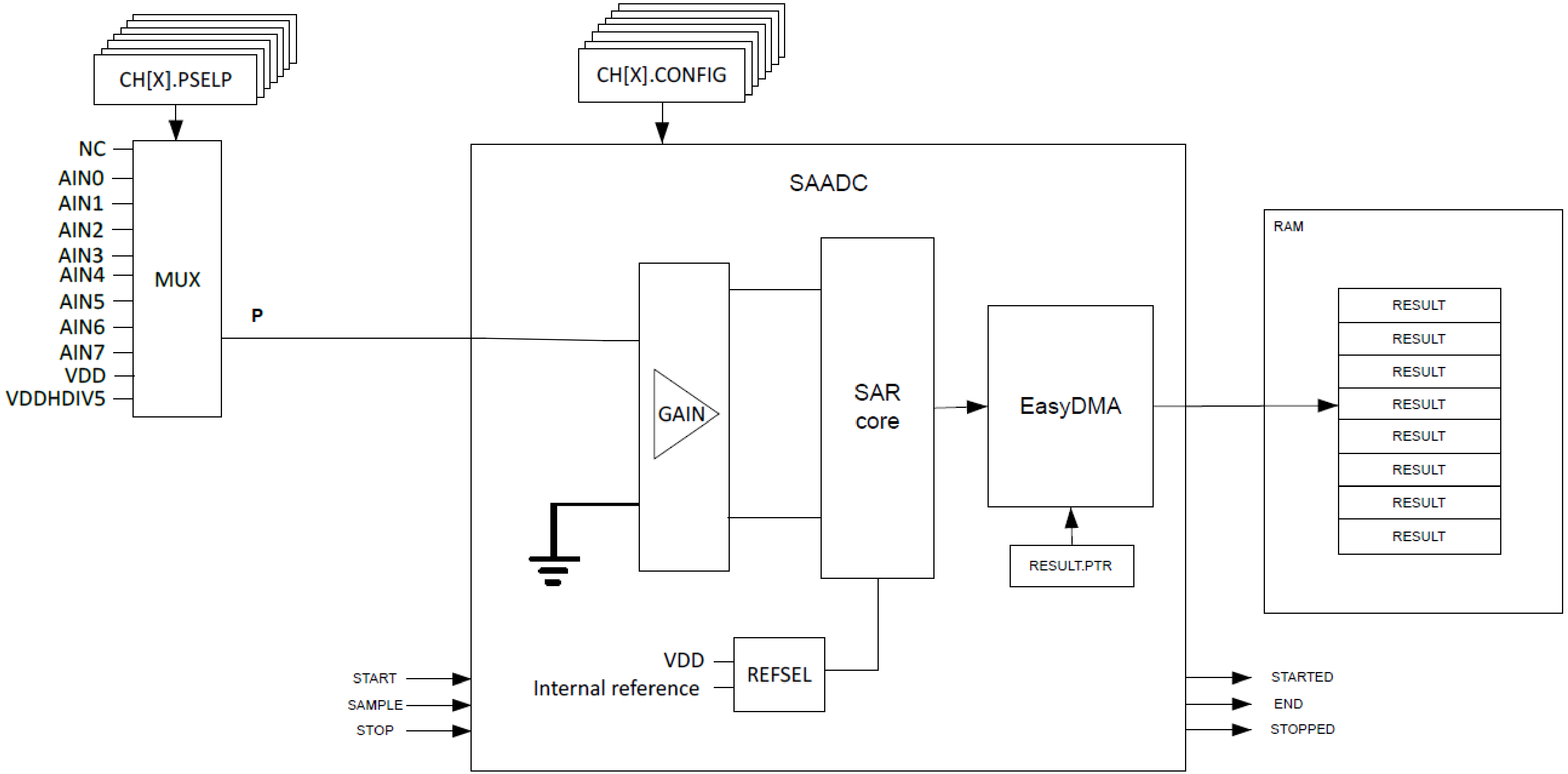
# Outline

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

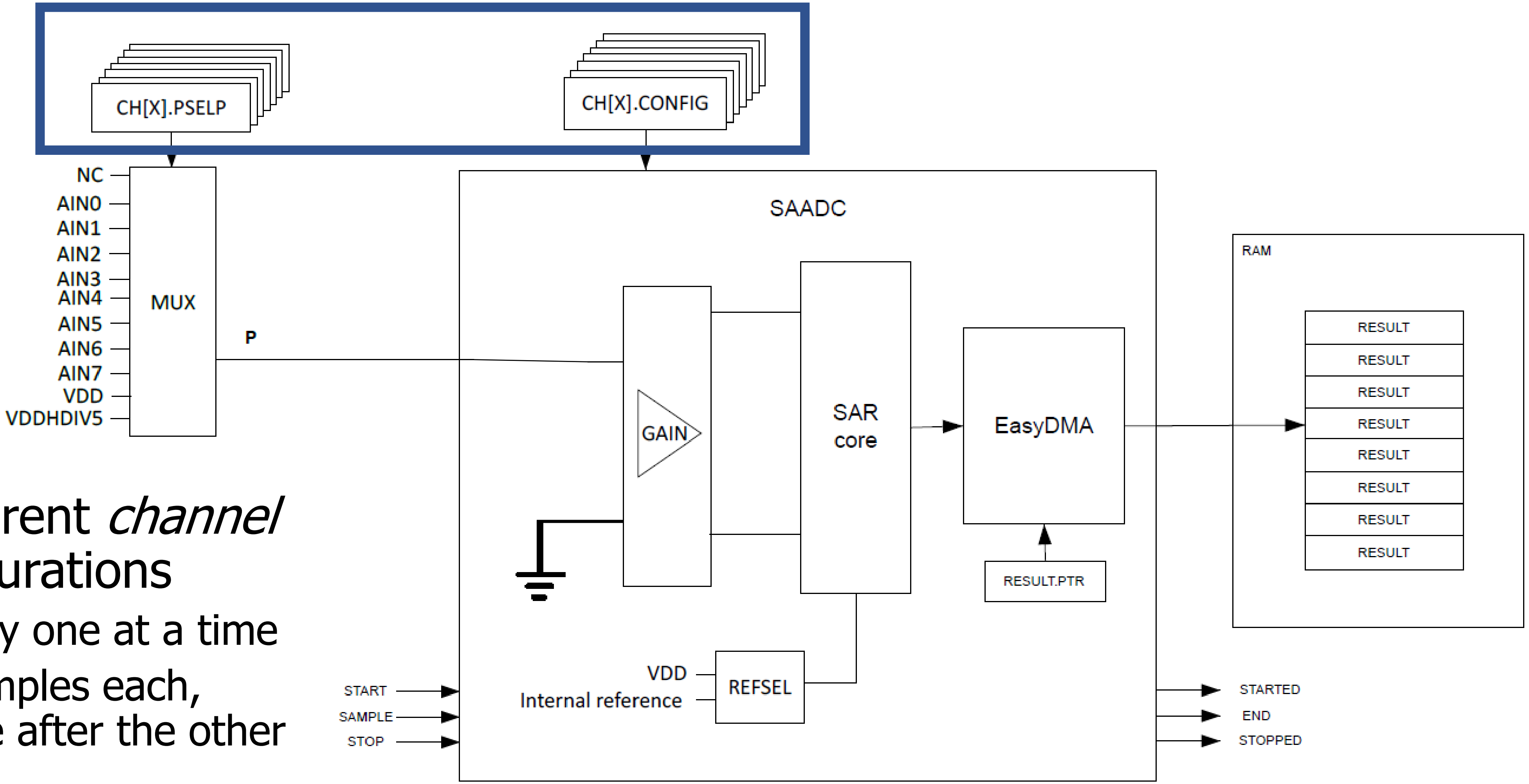
# nRF SAADC (Successive Approximation ADC)



# How most people use the SAADC



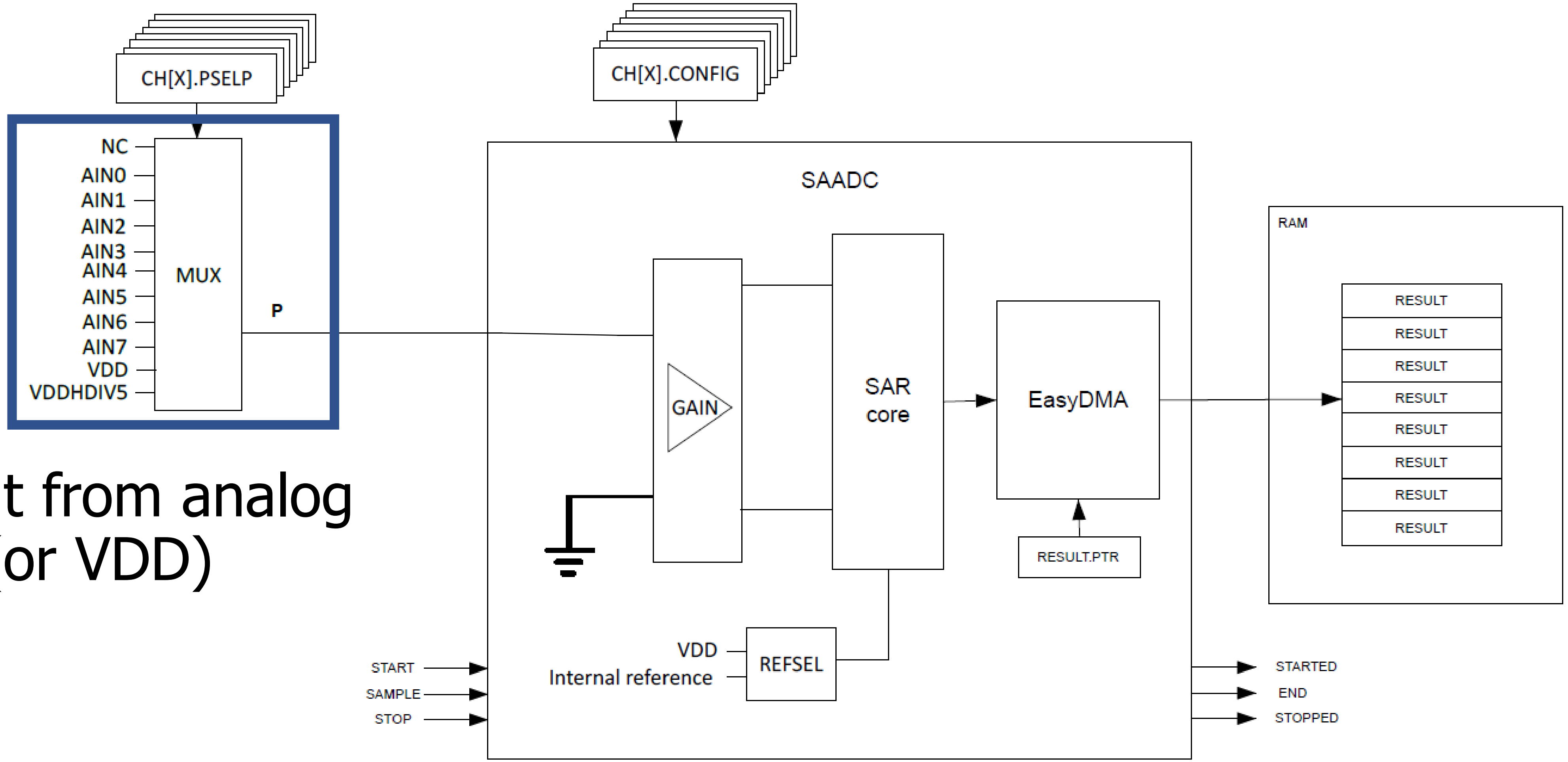
# How most people use the SAADC




- 8 different *channel* configurations
  - Only one at a time
  - Samples each, one after the other

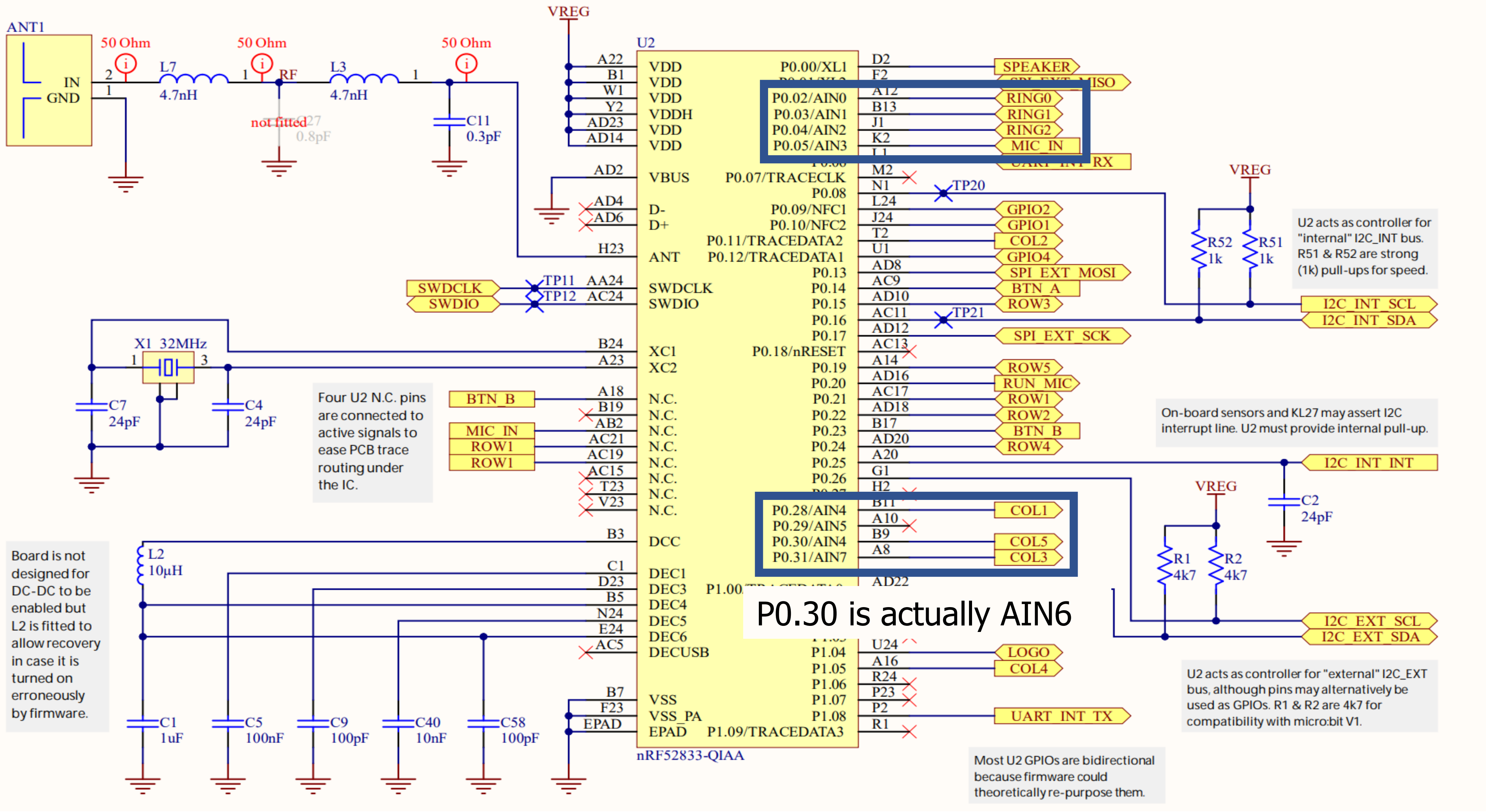
- Essentially virtualization in hardware!

# How most people use the SAADC

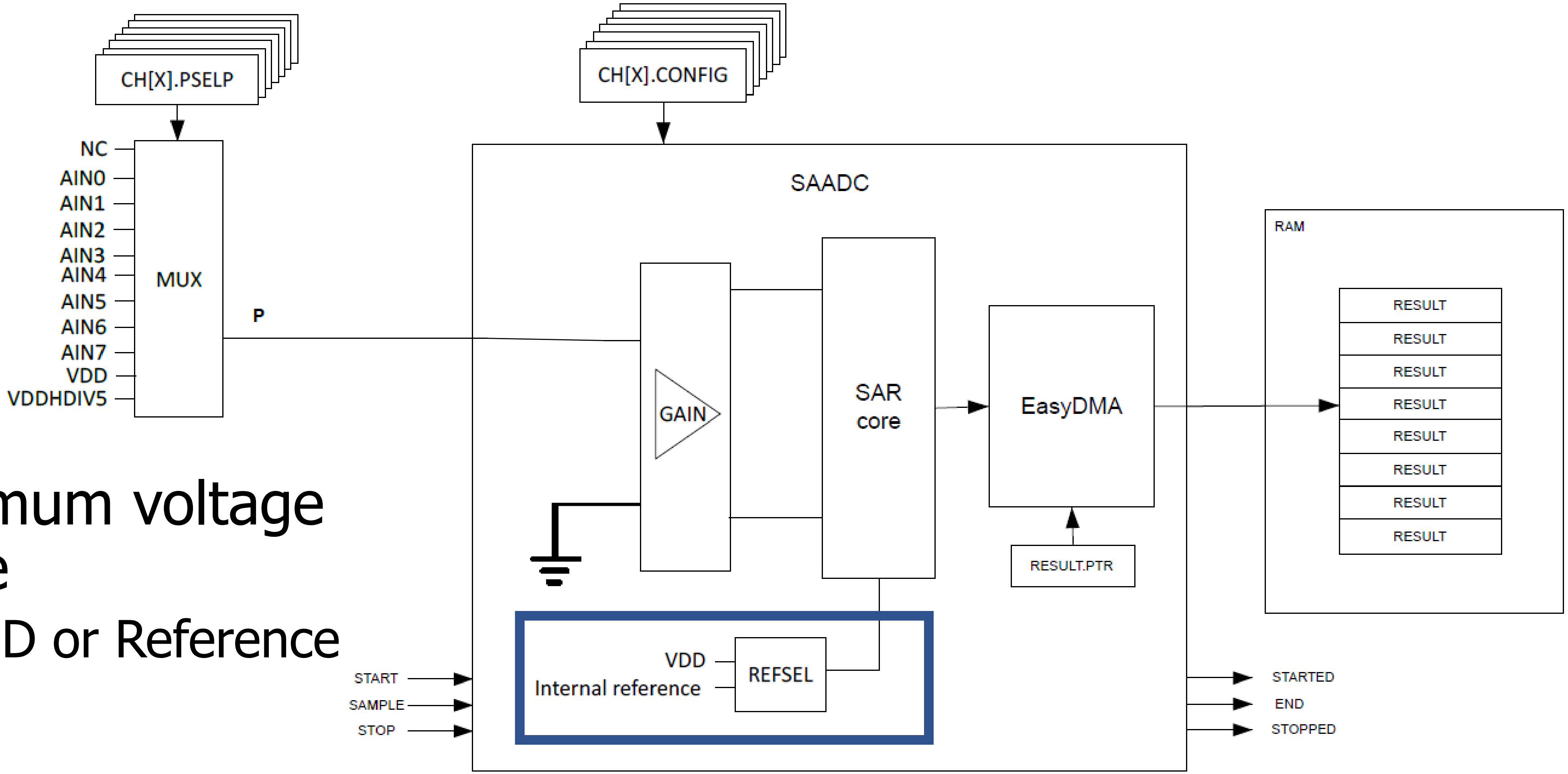



- Input from analog pin (or VDD)

# Analog inputs on the Microbit

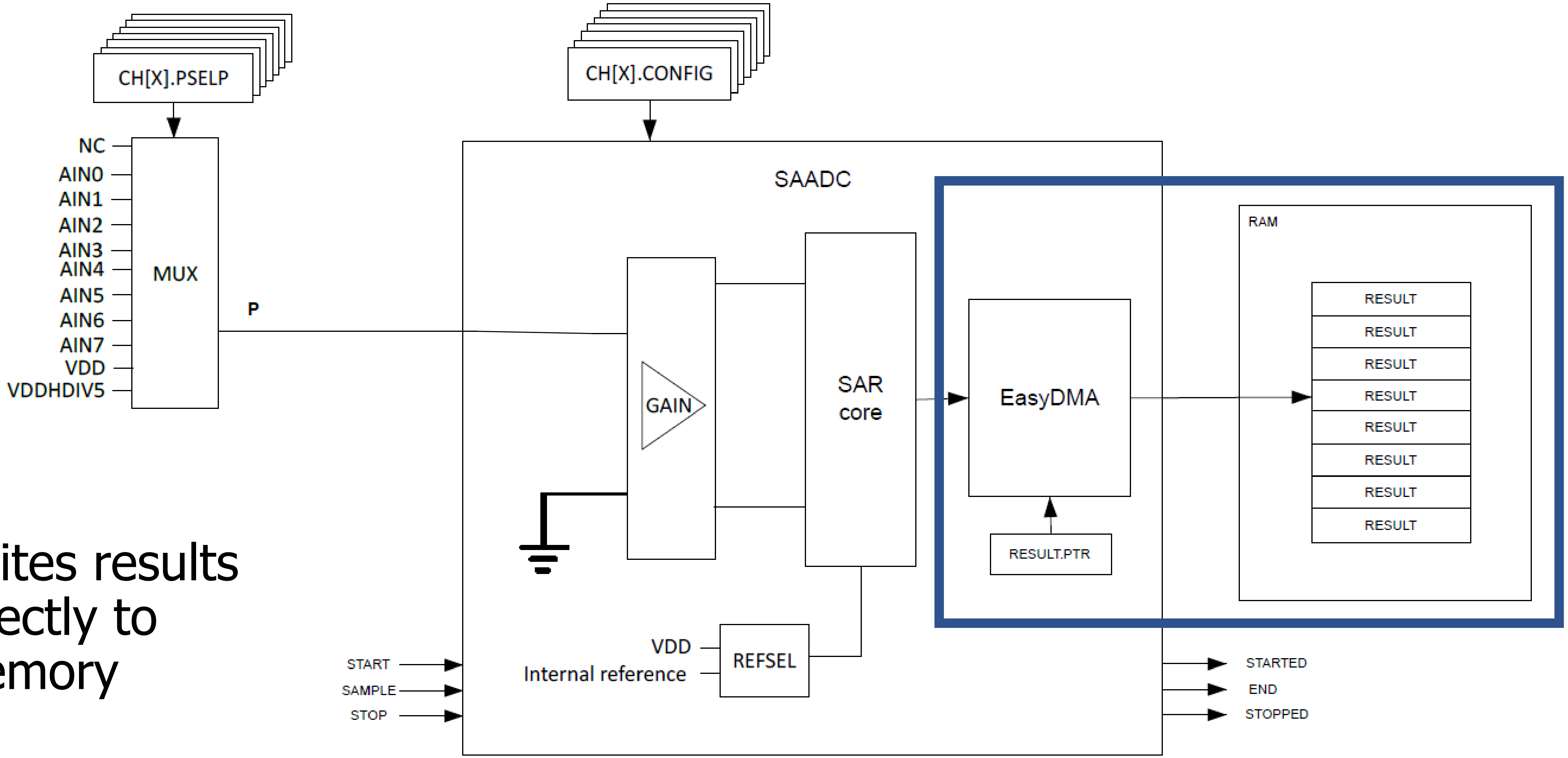


# How most people use the SAADC




- Maximum voltage range
  - VDD or Reference

# How most people use the SAADC

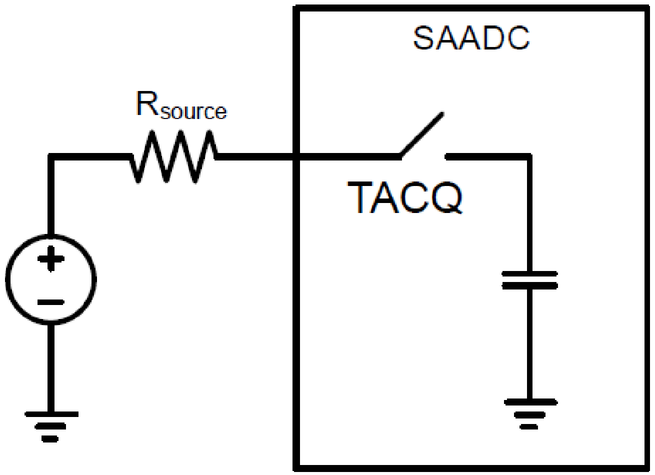



- DMA
  - Writes results directly to memory

# 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



# 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 MHz / ($2^{Scale}$)     where scale is 11 bits
  - Maximum rate is 7.8 kHz

# EasyDMA on the SAADC

- There is no register to read ADC results from

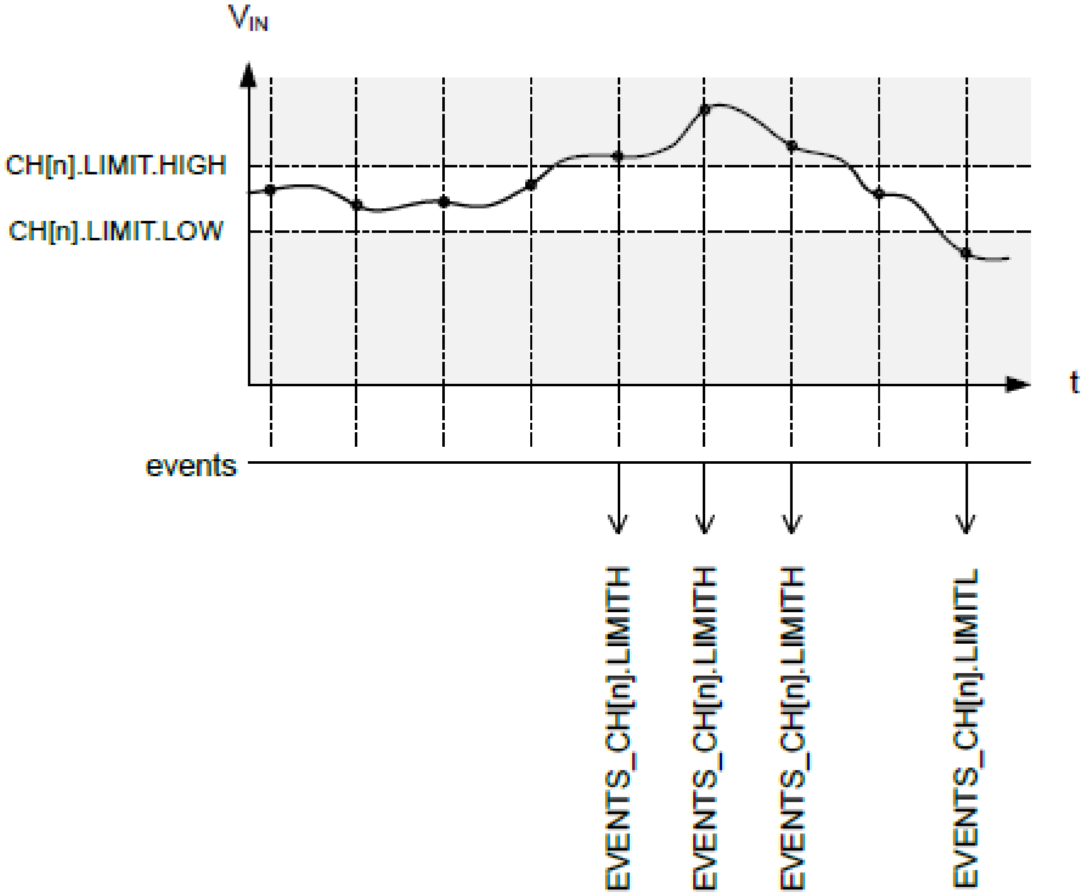
- Instead, you must use DMA to collect samples
- At configuration time, provide:
  - Pointer to RAM
    - Must be RAM, not Flash
  - Maximum count of 16-bit samples to be written starting at address
    - Up to 32768

- When complete, a register tells you the amount of samples written to RAM

# Event limit monitoring

- Includes two comparators for each channel
  - High and Low limits

- Generates events whenever transitioning to above High or below Low
  - Events can be ignored if unnecessary



# Temperature sensitivity

- ADCs are often temperature sensitive
  - nRF SAADC: 0.02% per degree C

- Recommends recalibrating every change of 10 degrees C or more
  - Automatic task for calibration
  - Real concern for deployed devices
    - Outdoors
    - Wearable

# Design question

- How many analog samples can the Microbit hold?

# Design question

- How many analog samples can the Microbit hold?

  - Available: 128 kB RAM, 512 kB Flash  (64000 samples in RAM)

  - Questions
    - Are they packed or padded to 16-bit?
    - How much memory are you using for other things?
    - Are you moving them into Flash periodically? (or external storage)

# Outline

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