Embedded Systems

Lecture 4
Hardware-Software Interfaces – Timer, PWM and ADC

© Michele Magno

D-ITET Center for Project-Based Learning

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Where We are

Hardware-Software

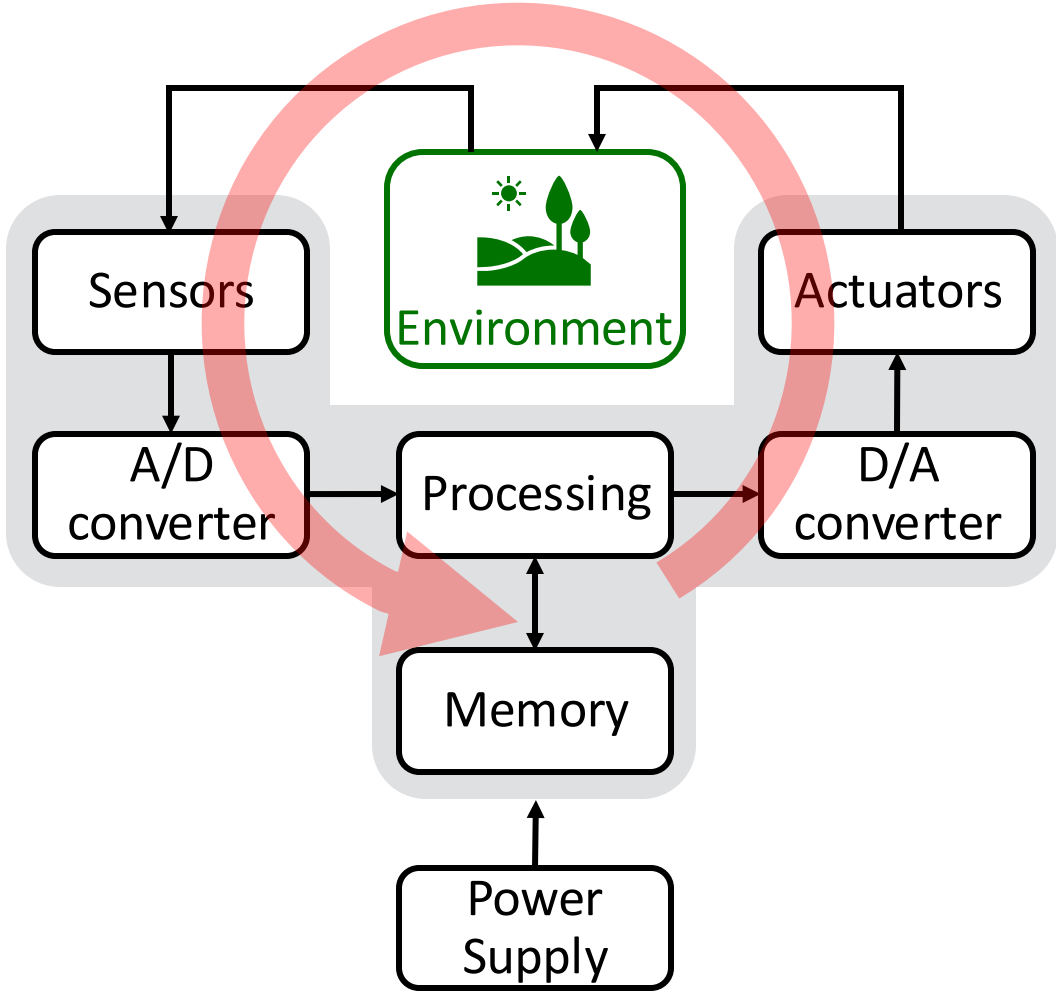
Real-Time

Special

0. Introduction into Embedded Systems
1. Hardware-Software Architecture and Software Development
2. Hardware-Software Interfaces – (GPIO), Interrupt, and Clock
3. Hardware-Software Interfaces - Serial Interfaces
4. No Lecture
5. Hardware-Software Interfaces - Timer, PWM and ADC
6. Real-Time Systems
7. Dynamic Scheduling and Real-Time Operating Systems
8. Deterministic Scheduling
9. Low Power Design
10. Computational Units
11. Implementation Strategies & Project Kick-off
12. Project Q&A

What is Missing?

- Processing Unit
- Storage
  - SRAM / DRAM / Flash
  - Memory Map
- Input and Output
  - General Purpose Input/Output (GPIO)
  - Interrupt
  - Memory Mapped Device Access
  - **Analog Digital Conversion**
- Interfaces
  - UART, SPI, I2C
- Clocks and Timers
  - Clock Tree
  - **Timer and PWM**

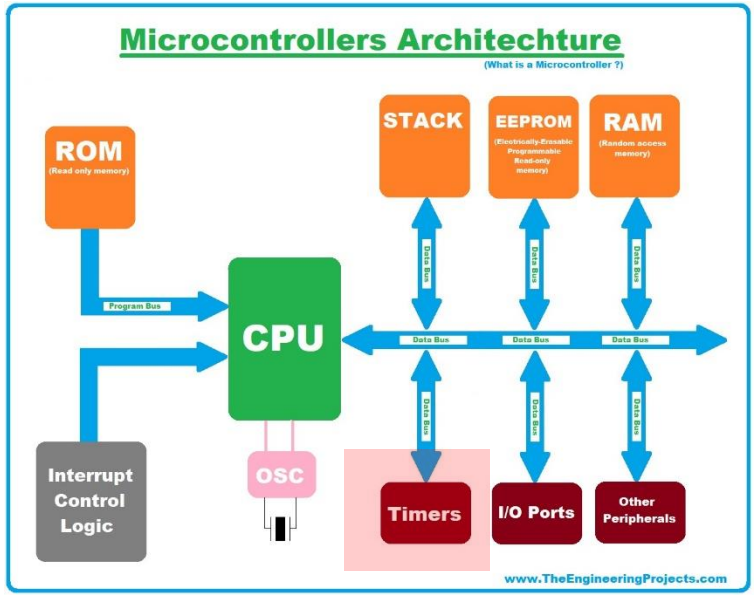


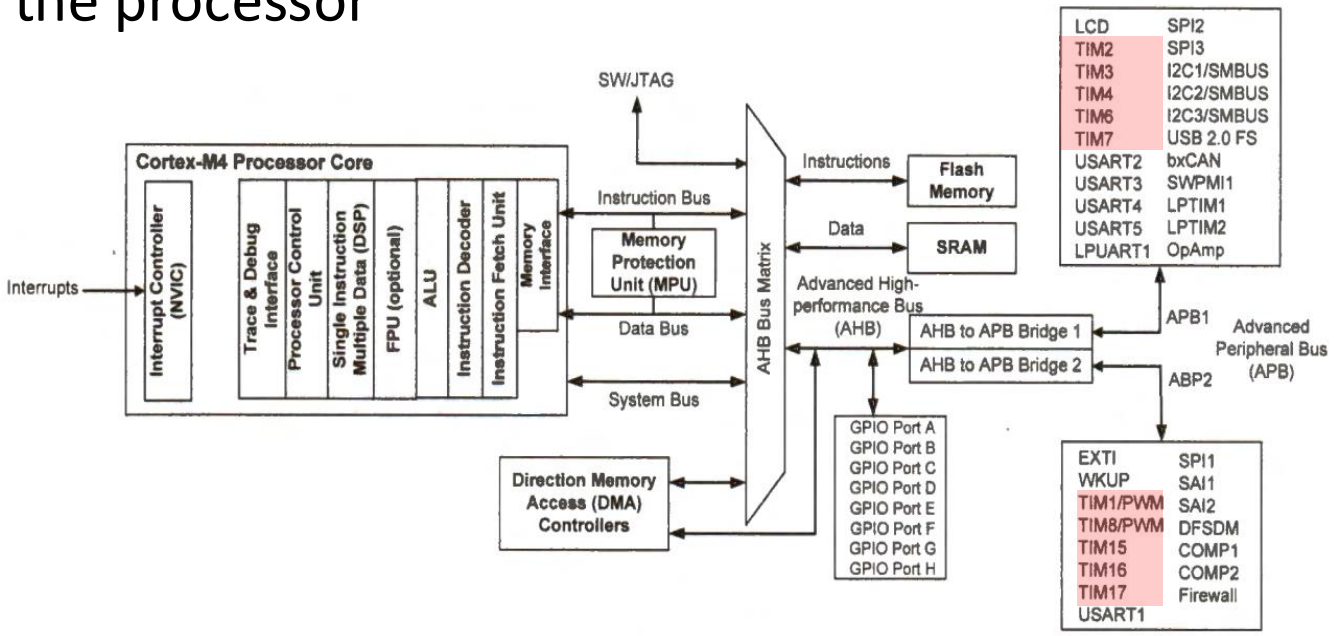
Timers

Timers

Timers are hardware peripherals which can be used in tasks which involves counting events or performing time-related operations:

- Accurate Delay function
- Sampling at a fixed rate
- Counting Events without using the processor





[1] Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, page 55

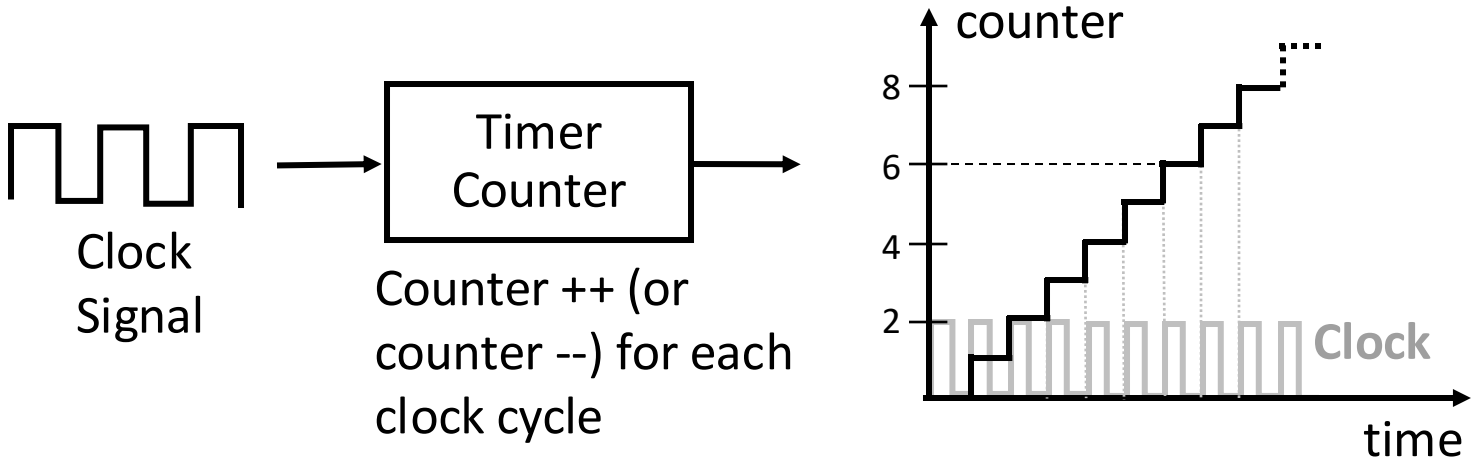
Timers are Counters

A *timer is a counter* that counts upwards or downwards by incrementing or decrementing the counter for every rising or falling clock edge of a *constant input clock.*

- The counting speed depends on the clock frequency.
- Timers run independently of processor core allowing the core to perform other tasks or entering the sleep mode to save power.

$$Max\ value = 2^n - 1\,.$$

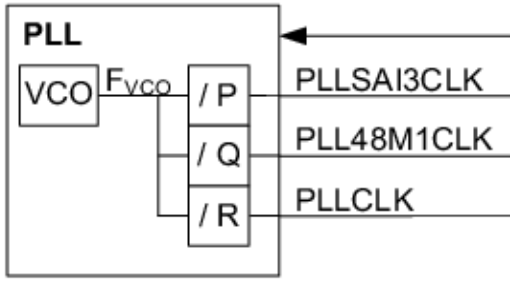
i.e. $n = 8. Max\ value = 255.$

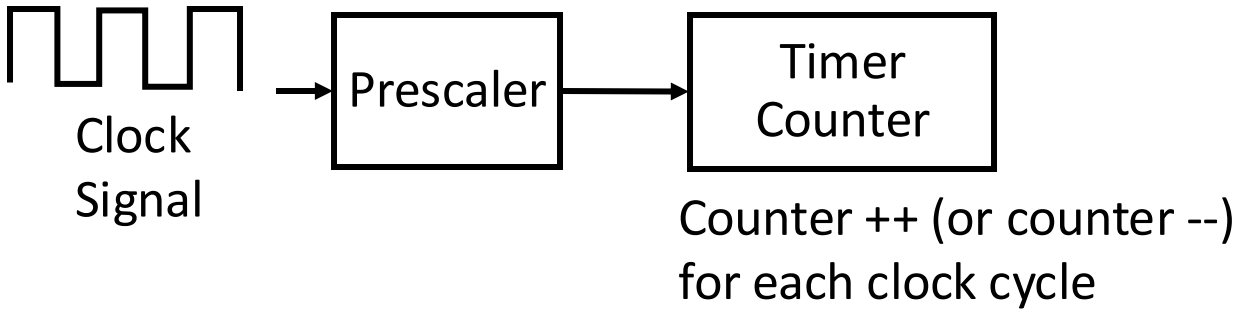


Timers – Input Clock

The frequency of constant input clock can be created using PLLs and/or scaled down by the timer's internal prescaler to generate the requested timer resolution.

- *Prescaler*: internal counter to divide the frequency by an integer.





Timers – Resolution

Depending on the application, an appropriate input clock must be defined. This is often a trade-off between the input clock frequency and max value of the counter.

High Resolution

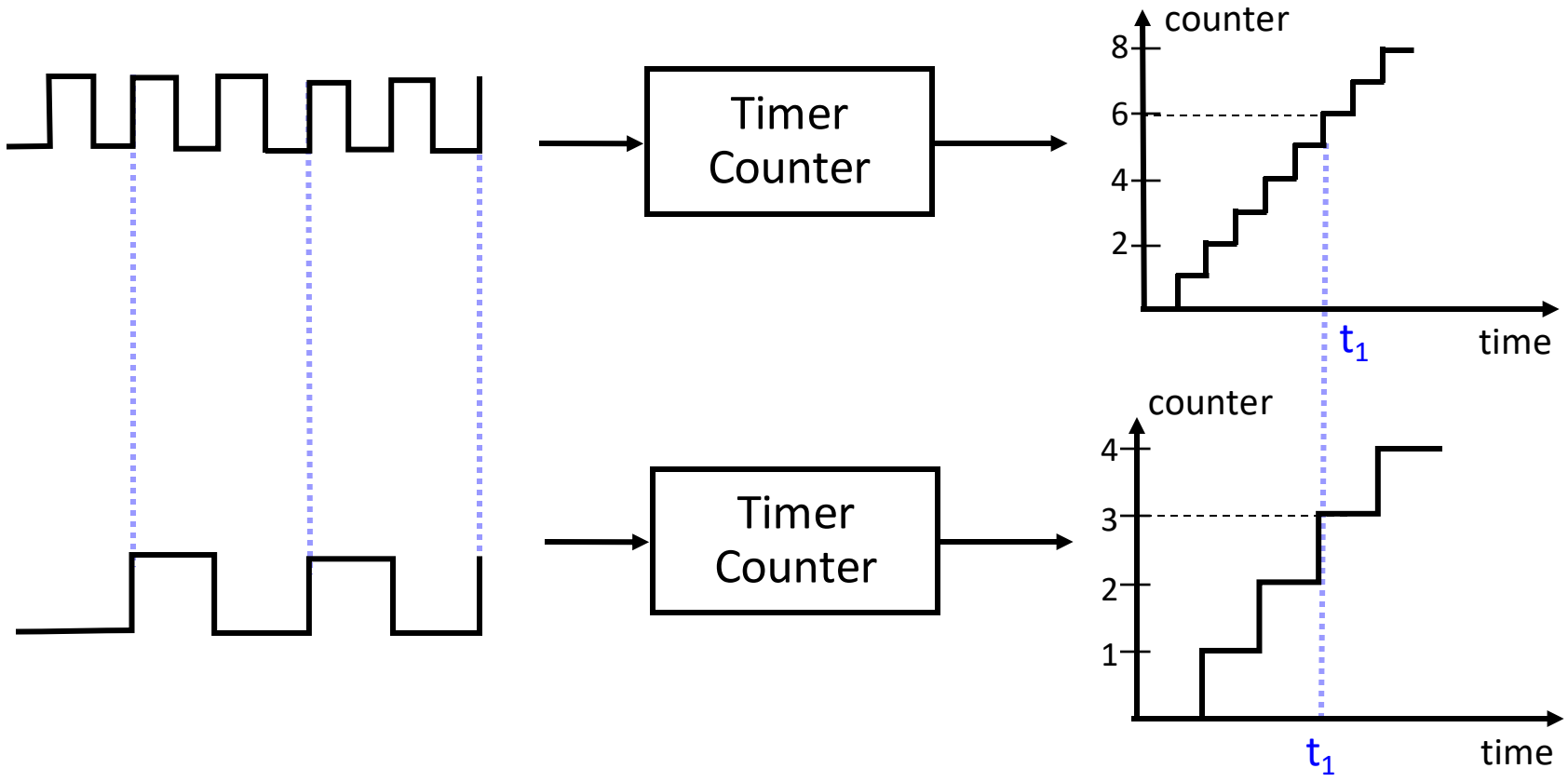
Fast clock

Higher counter

Low Resolution

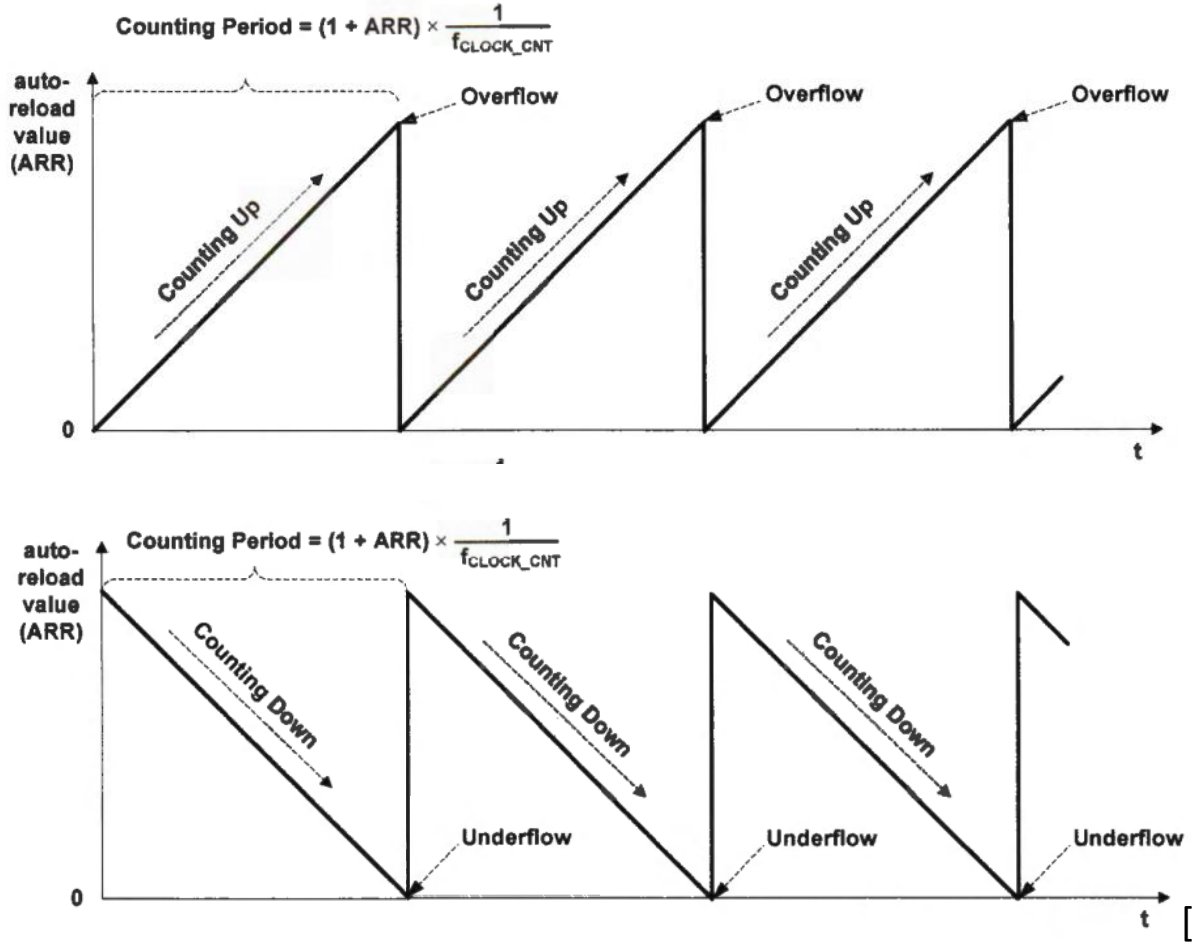
Slow clock

Smaller counter



Timers – Counting Directions

- Counter counts upwards/downwards until it reaches either a max value/zero.
- Max value can be configured using the ARR (Auto-Reload Register).
- Upwards: counting up until ARR is reached, reset to 0.
- Period: $(1 + ARR) * \frac{1}{f_{clock}}$
- Downwards: counting down until 0, reset to ARR.

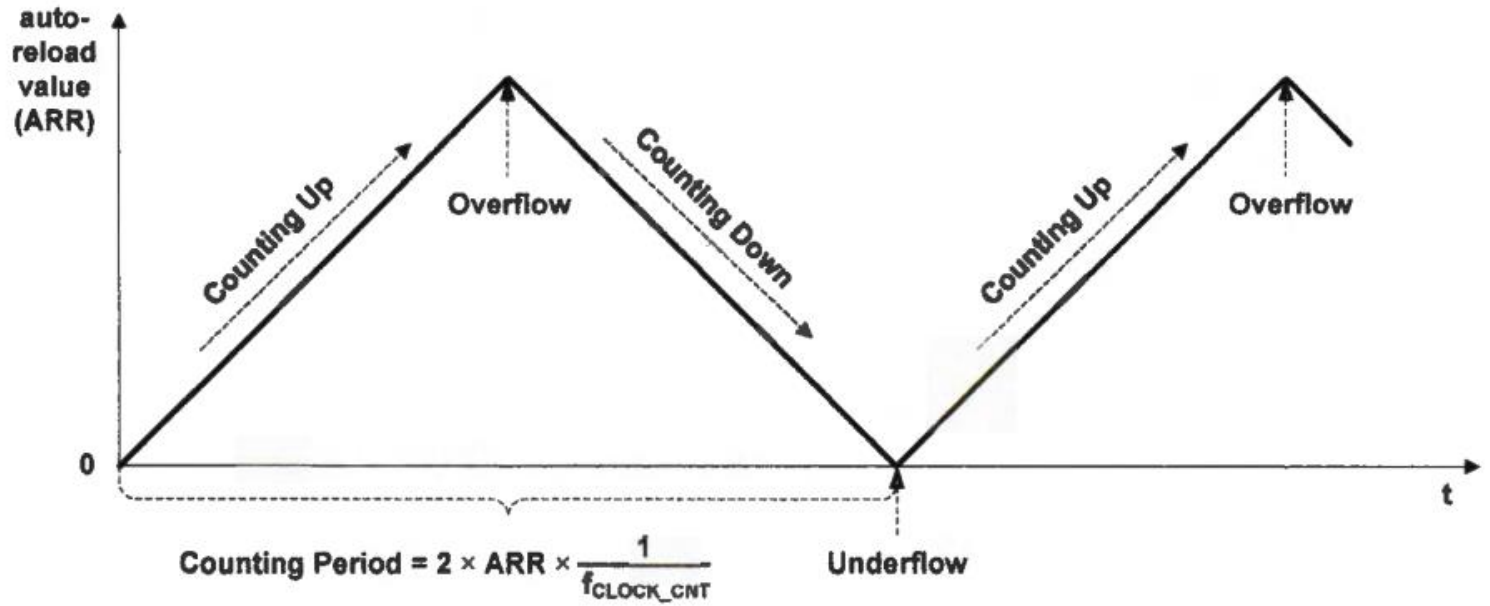


[1]

[1] Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, page 407

Timers – Counting Directions

- Upwards & Downwards :
  - counting up until ARR is reached and then down until 0.
  - Period:$(2 * ARR) * \frac{1}{f_{clock}}$



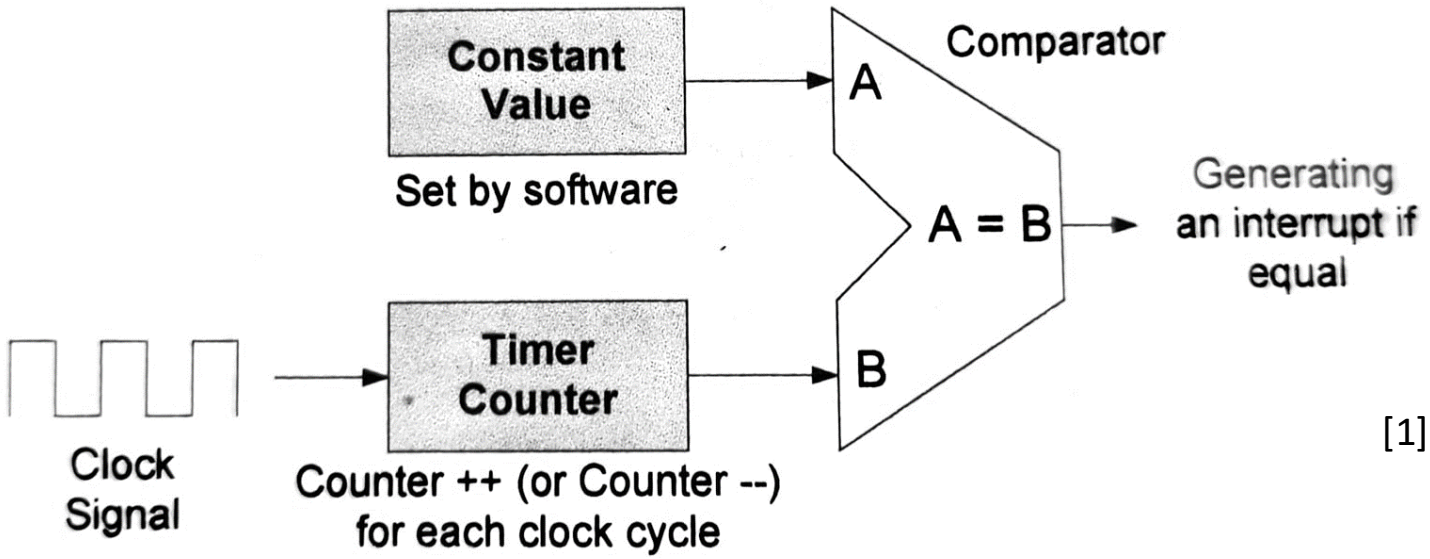
[1]

[1] Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, page 407

Timers – Output Compare Mode

A *comparator* constantly compares the counter value with a constant value set in software and stored in one of the timer's control registers.

- Constant value is stored in CCR (Capture/Compare register).
- The comparator constantly compares the current counting value with the CCR and generates an interrupt if they are equal.

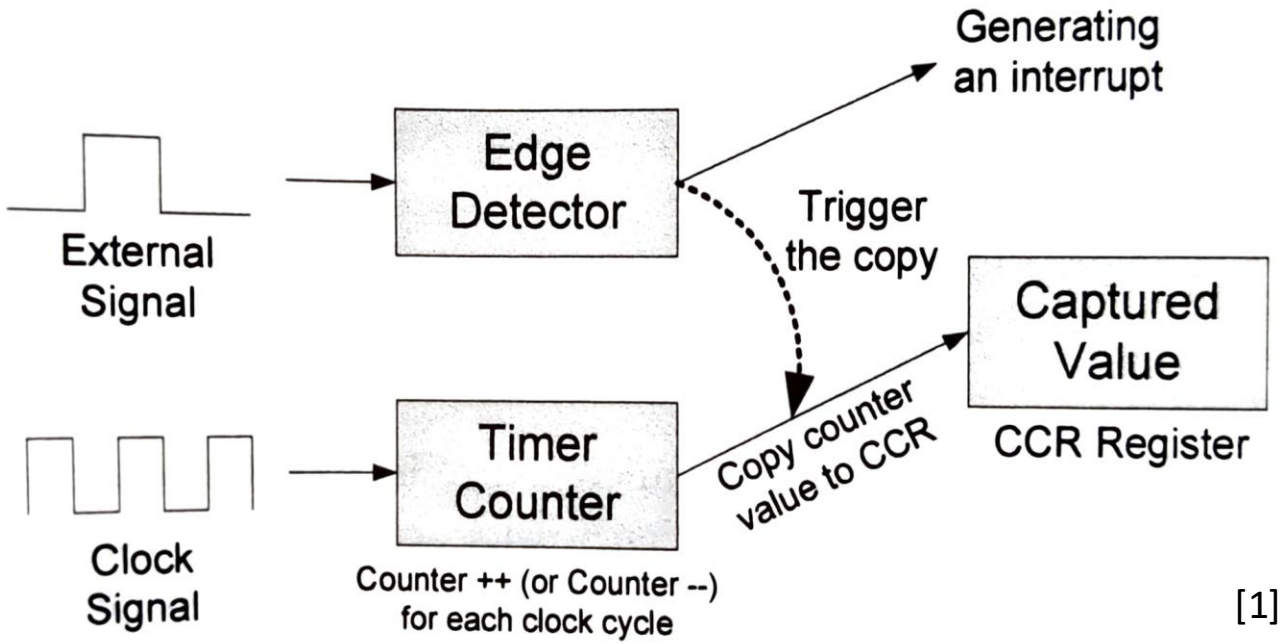


[1]

[1] Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, page 405

Timers – Input Capture Mode

- Waits for an external signal with an edge detector and stores the counter in the CCR register(Compare/Capture Register).
- Simultaneously, a normal interrupt can be triggered, too.
- Measures the time between different events.

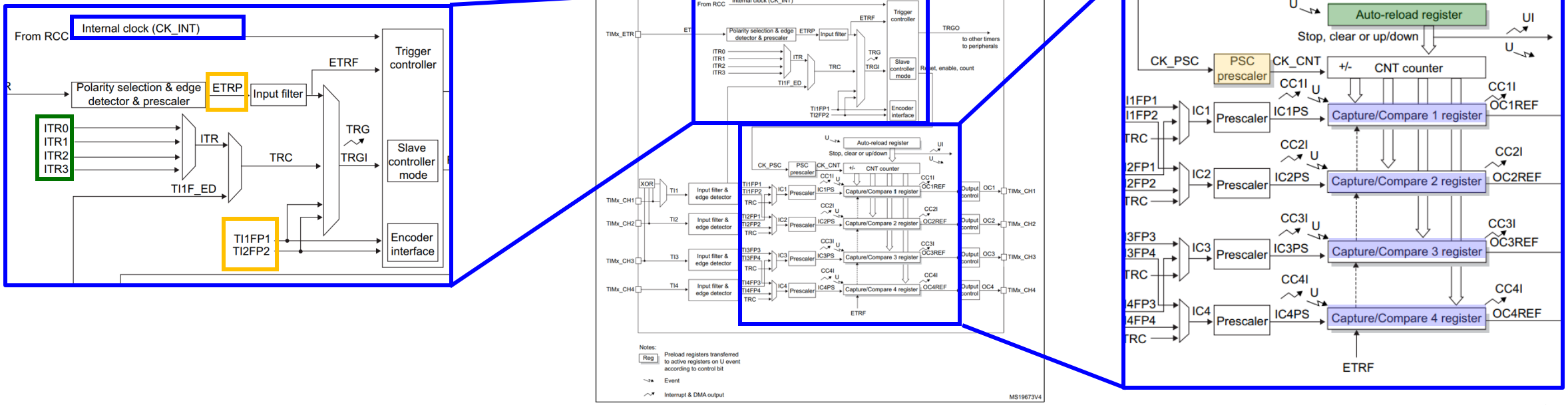


[1]

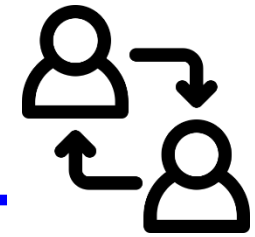
[1] Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, page 406

Timers – STM32L47xx

- Example of how a simplified timer looks.
- The counter can be triggered based on a clock, external inputs or other timers.
- One Timer can have multiple channels, each with different CCR values which can share a prescaler and ARR.
- Other prescalers are used to prescale the input in input capture mode.

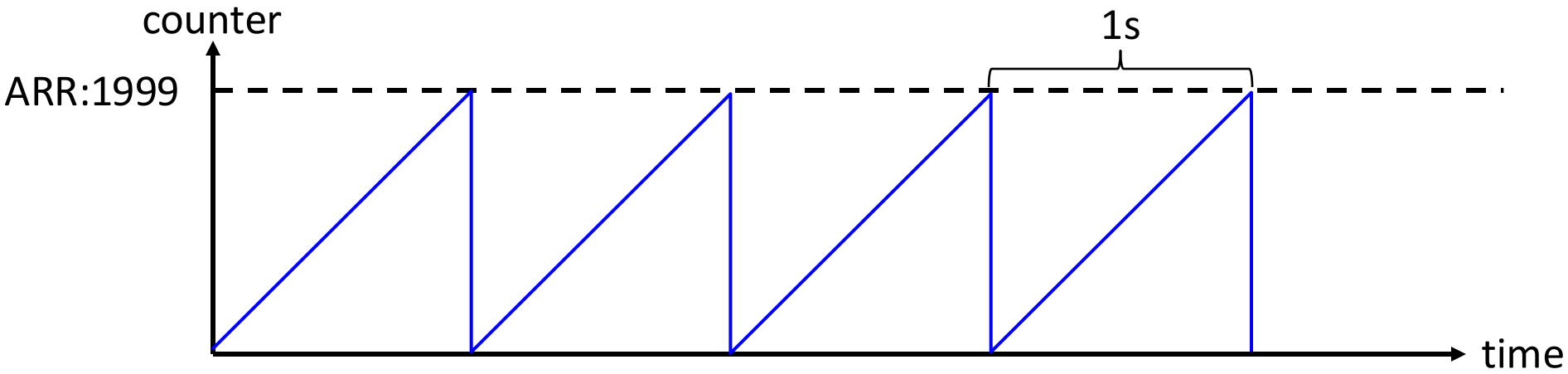


Timers – Toggle LED



Your task is to *toggle an LED every second* by generating an interrupt using a timer with the following input clock configuration: The input clock speed is set 8 MHz and gets downscaled with an internal prescaler of 4000. The LED toggles its state when an interrupt occurs.

Which values must be stored in the CCR and ARR?



Input clock: 8000000 Hz / 4000 = 2000 Hz

CCR could be set to anything between 0 and 1999 as time between two consecutive successful compares is always 1s apart.

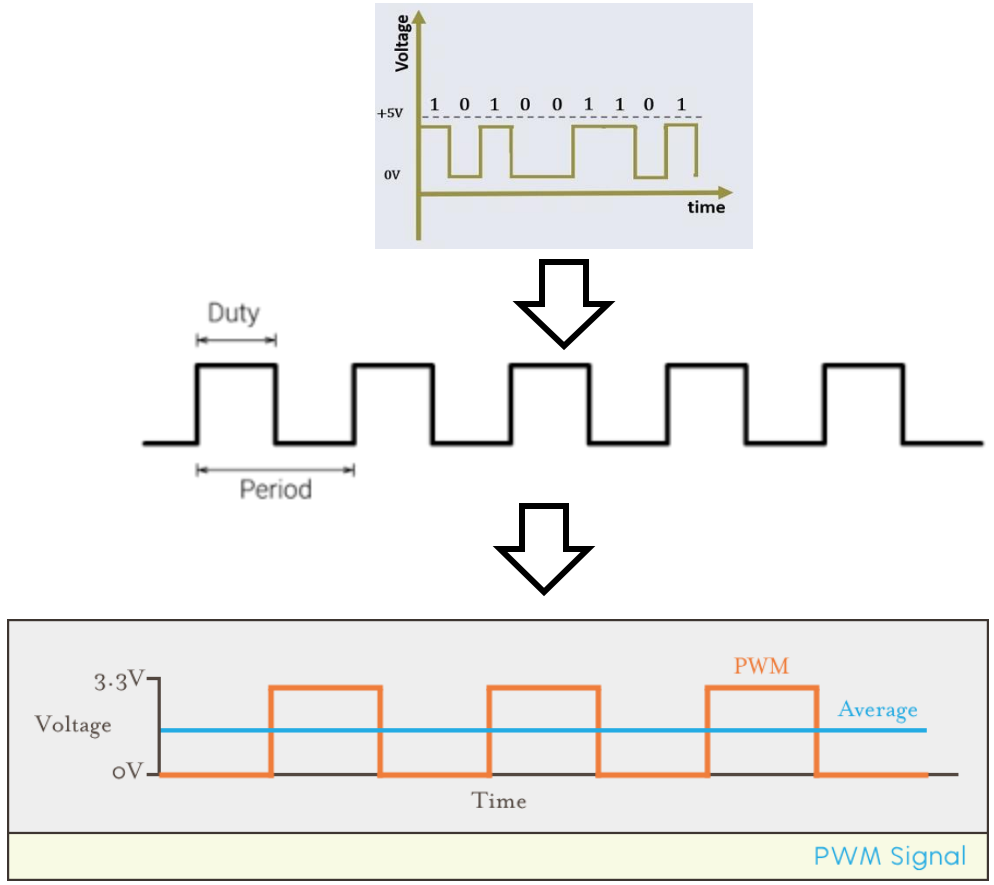
PWM – Pulse Width Modulation

PWM – Pulse Width Modulation

Pulse width modulation (PWM) is a digital technique used to create a digital signal with constant frequency but variable pulse-pause duration.

Uses rectangular waveform to quickly switch voltage source on and off to produce a desired average power on a load.

- Rectangular waveform is created using timers
- Used in:
  - Torque/speed control
  - Digital encoding in telecommunications
  - DC-to-DC power conversion
  - Controlling brightness of LED

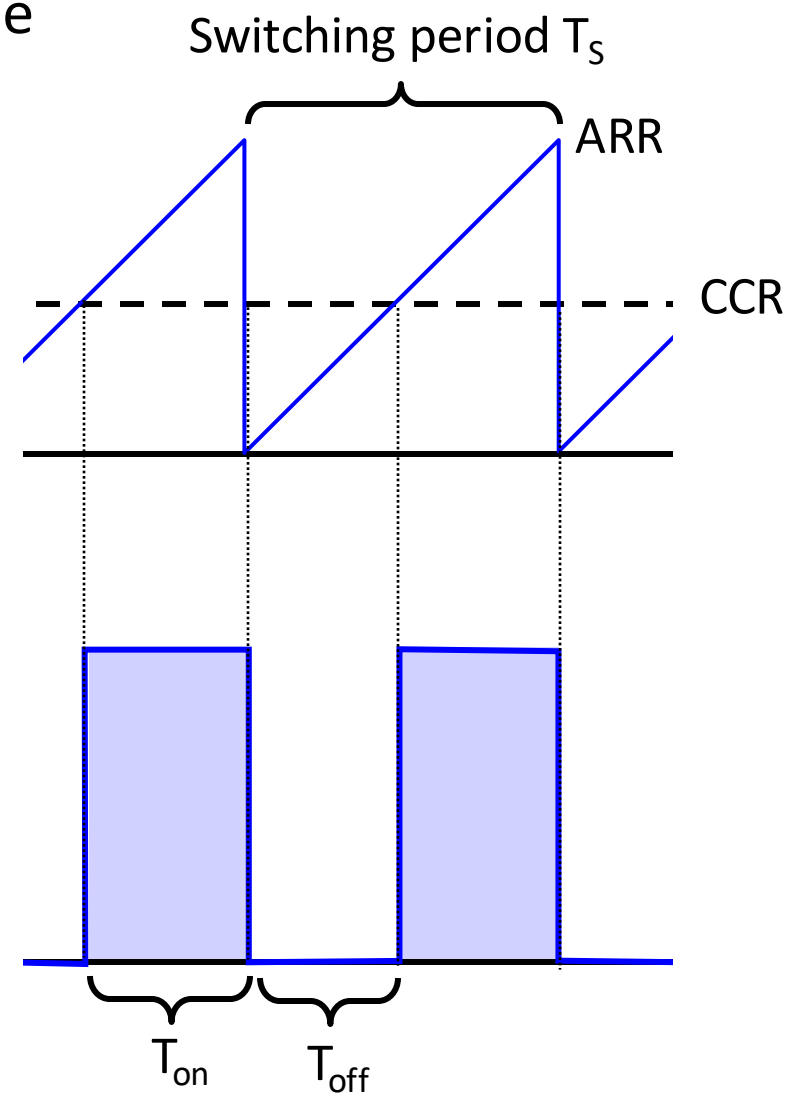


PWM – Duty Cycle

The *duty cycle* describes the proportion of the time where the signal is in a high state compared to the total signal period.

$$\textit{Duty Cycle} = \frac{pulse\ on\ time\ (T_{on})}{pulse\ switching\ period\ (T_S)} * 100\%$$

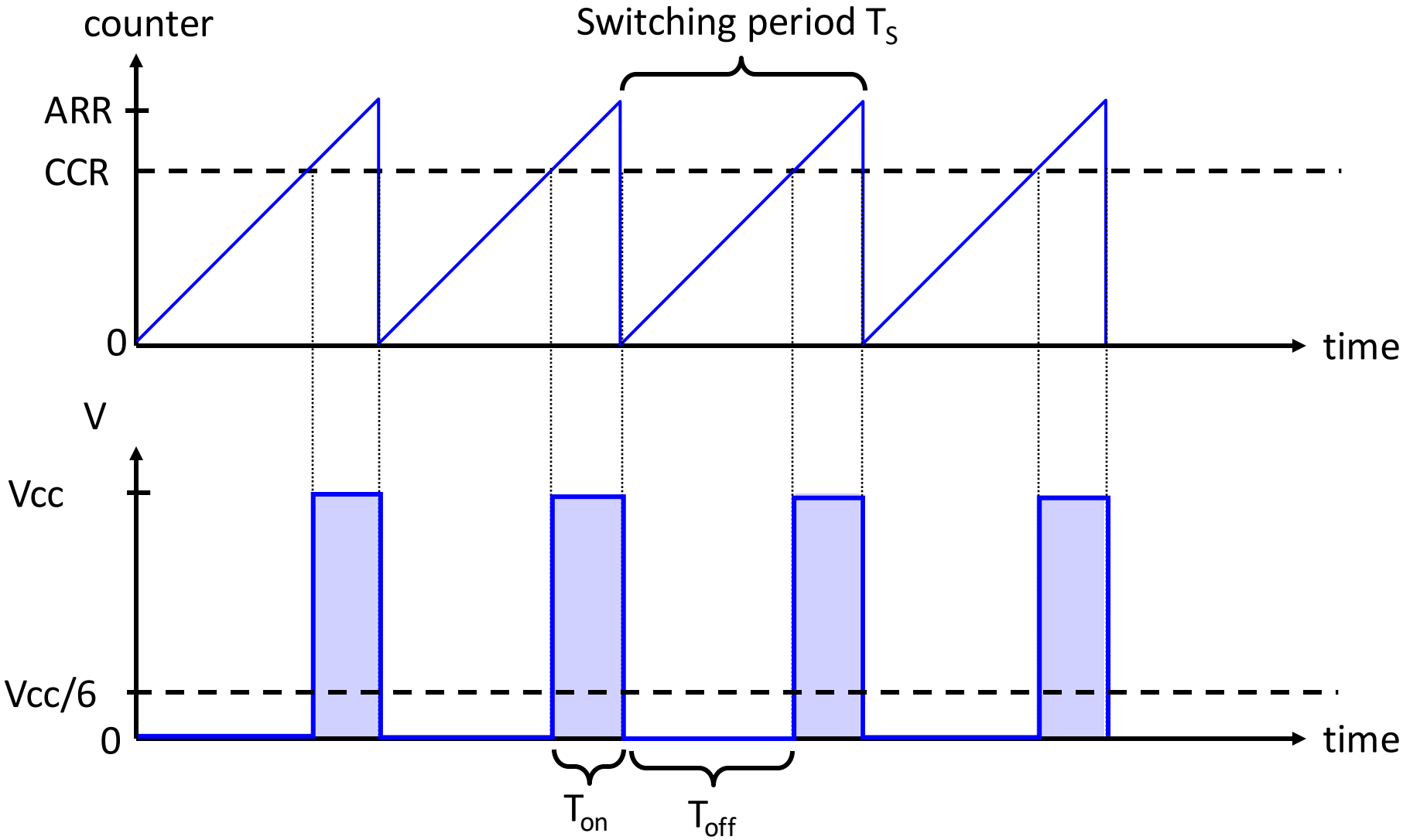
$$= \frac{T_{on}}{T_{on} + T_{off}} * 100\%$$



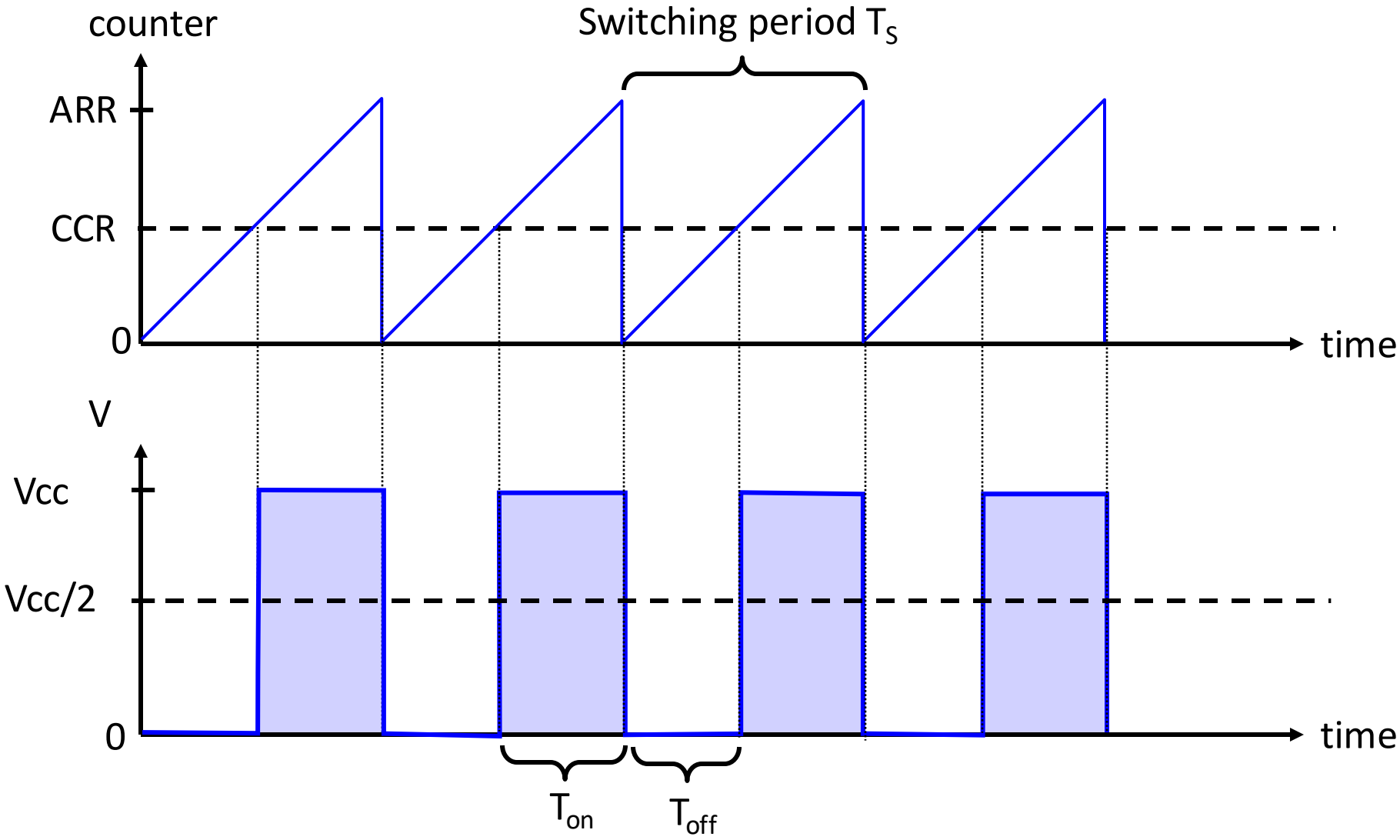
The on/off relation is controlled by the CCR and AAR:

- ARR: defines switching period
- CCR: defines T_{on} /T_{off}

PWM – Duty Cycle 1/6

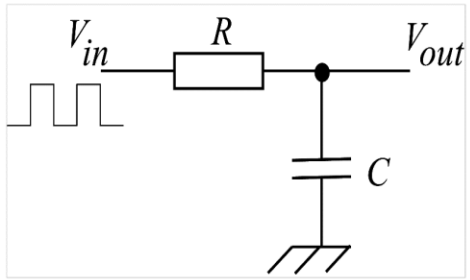


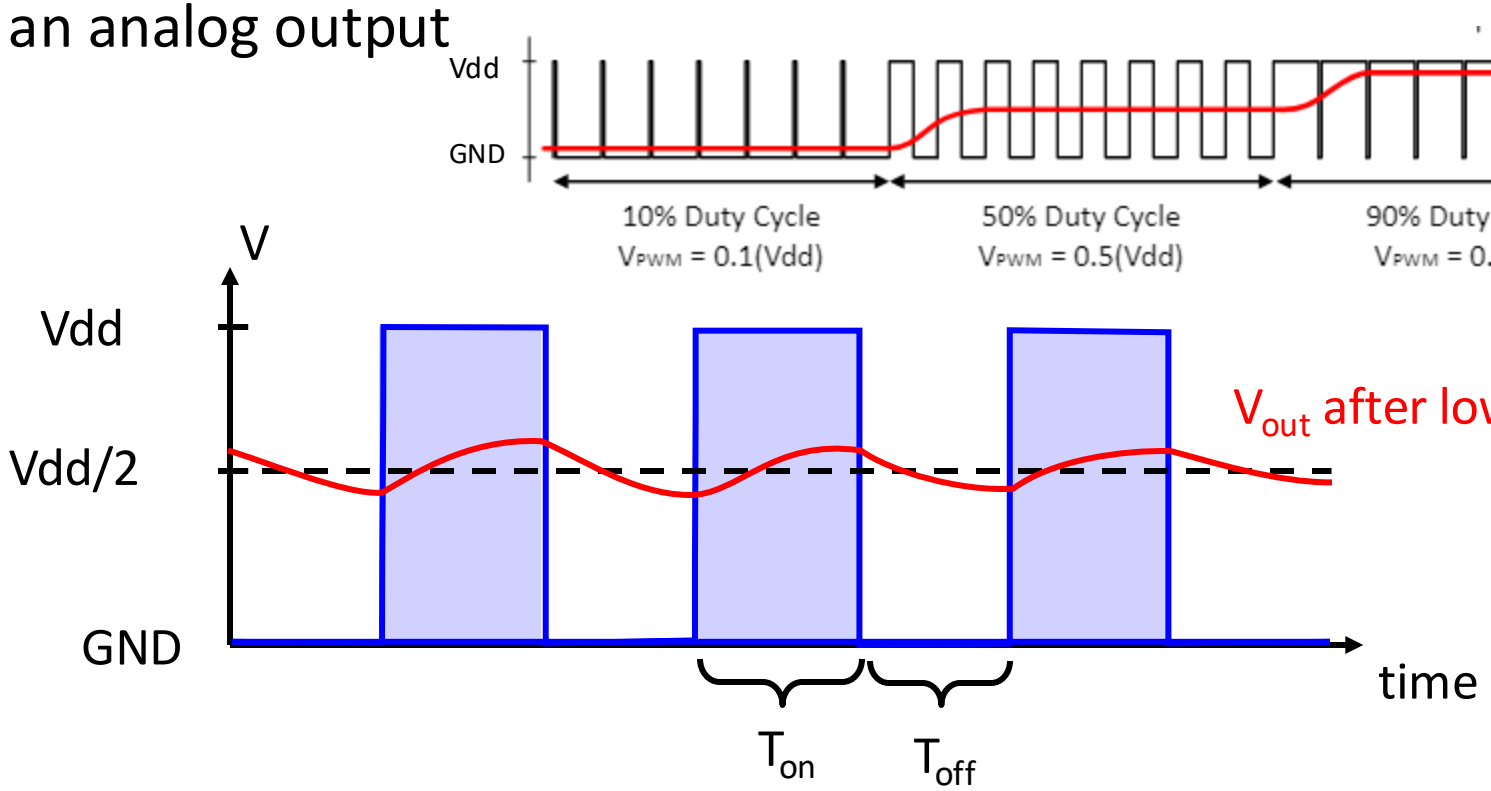
PWM – Duty cycle 1/2



PWM – Output as Digital to Analog Converter (DAC)

- Signal is sill digital
- The average value can be extracted from the PWM stream with a low-pass filter
- In this case, and if PWM frequency and values of R and C are appropriately chosen, V_{out} becomes an analog output

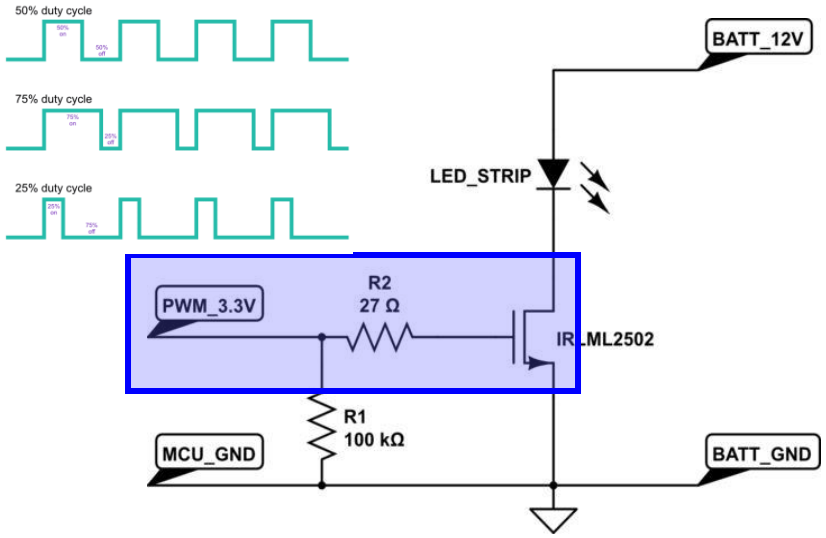




PWM – Dimming a LED

- Controlling the brightness of an LED by changing the high/low input proportion.
  - Turning it on and off very quickly.
- PWM switching frequency must be high enough not to see flickering.
- The output of an MCU can directly power a “low-current” LED.
- It uses the PWM signal as input for the transistor

Low current LED

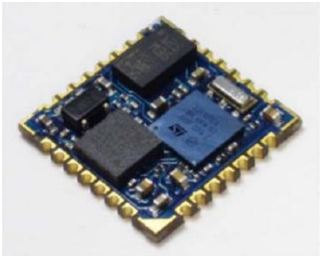


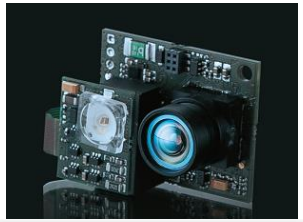
ADC – Analog-to-Digital Converter

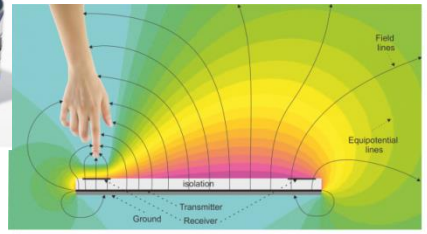
Why ADC ?

Sensors generally output an analog signal which need to be digitized

- IR sensors
- Contactless ECG
- Multi-Gas sensors
- Contactless Voltage + Current
- Acoustic sensors, analog microphones
- pH – Sensors

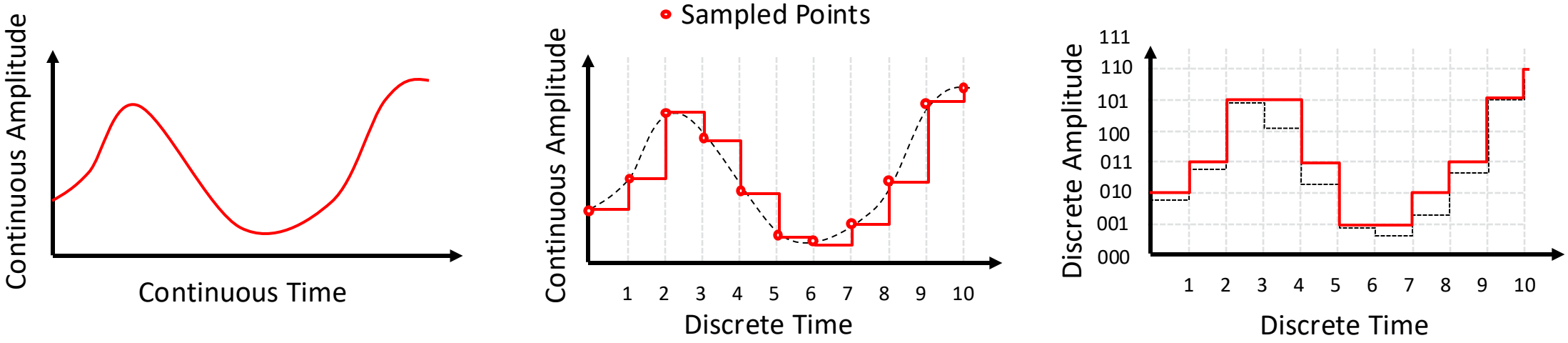


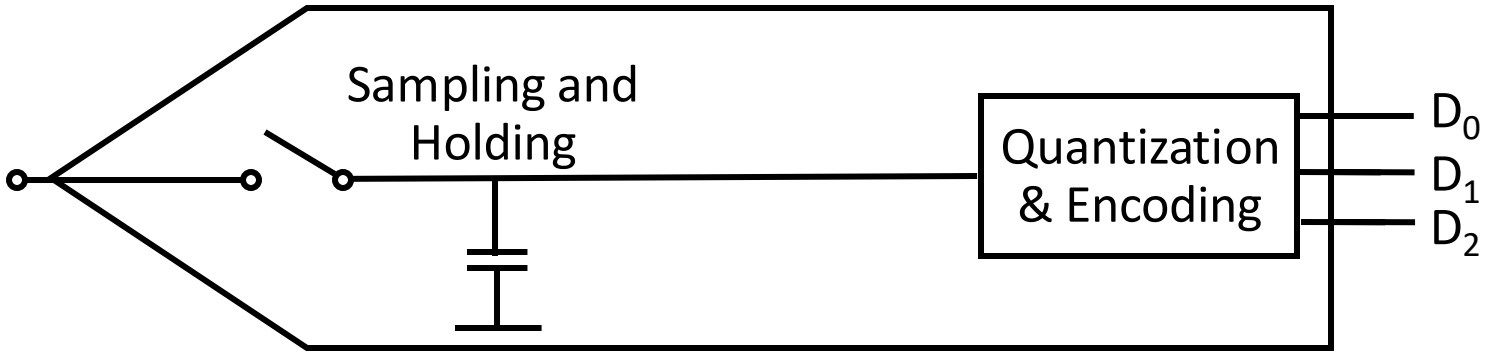




ADC – Analog-to-Digital Converter

- An analog signal is continuous in time and amplitude.
- The digital signal used by an MCU is discrete in time and amplitude





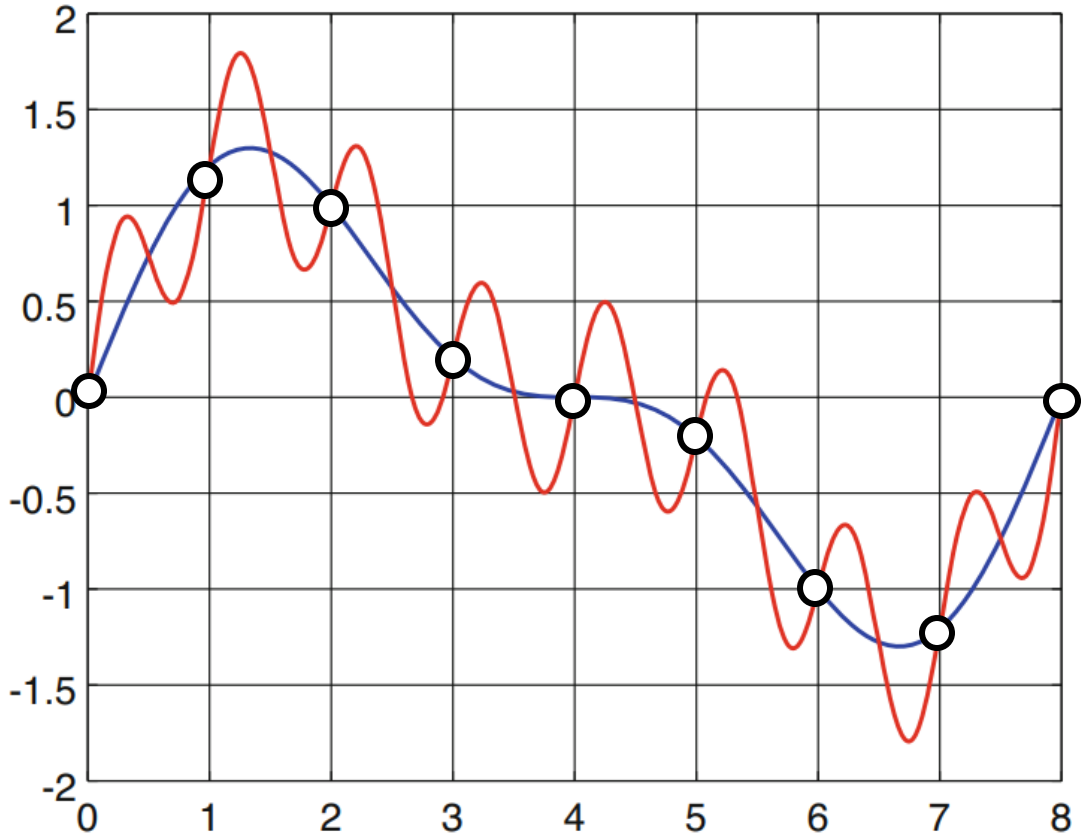
High level 3-bit SAR ADC

ADC – Metrics

- *Sampling rate*: Number of conversions an ADC can perform in a second.
- *Resolution*: The ADC resolution is defined as the smallest incremental voltage that can be recognized and thus causes a change in the digital output. This depends on the measuring range and the number of bits used.
- *Power dissipation:* Power efficiency of the ADC. Most often in mobile embedded systems, it is the power budget and not the hardware speed that limits the throughput of ADC.
- *Errors:* Offset, Gain error, Differential Non-Linearity (DNL), Integral Non-Linearity (INL)

ADC – Sampling Frequency

Visualization of functions e3(t) (blue) and e4(t) (red)



$$e_3(t) = \sin(\frac{2\pi t}{8}) + 0.5 \sin\left(\frac{2\pi t}{4}\right)$$

$$e_4(t) = \sin\left(\frac{2\pi t}{8}\right) + 0.5 \sin\left(\frac{2\pi t}{4}\right) + 0.5 \sin\left(\frac{2\pi t}{1}\right)$$

The fact that two or more unsampled signals can have the same sampled representation is **called aliasing**.

Sampling the signal n-integer times.

ADC – Avoid Aliasing

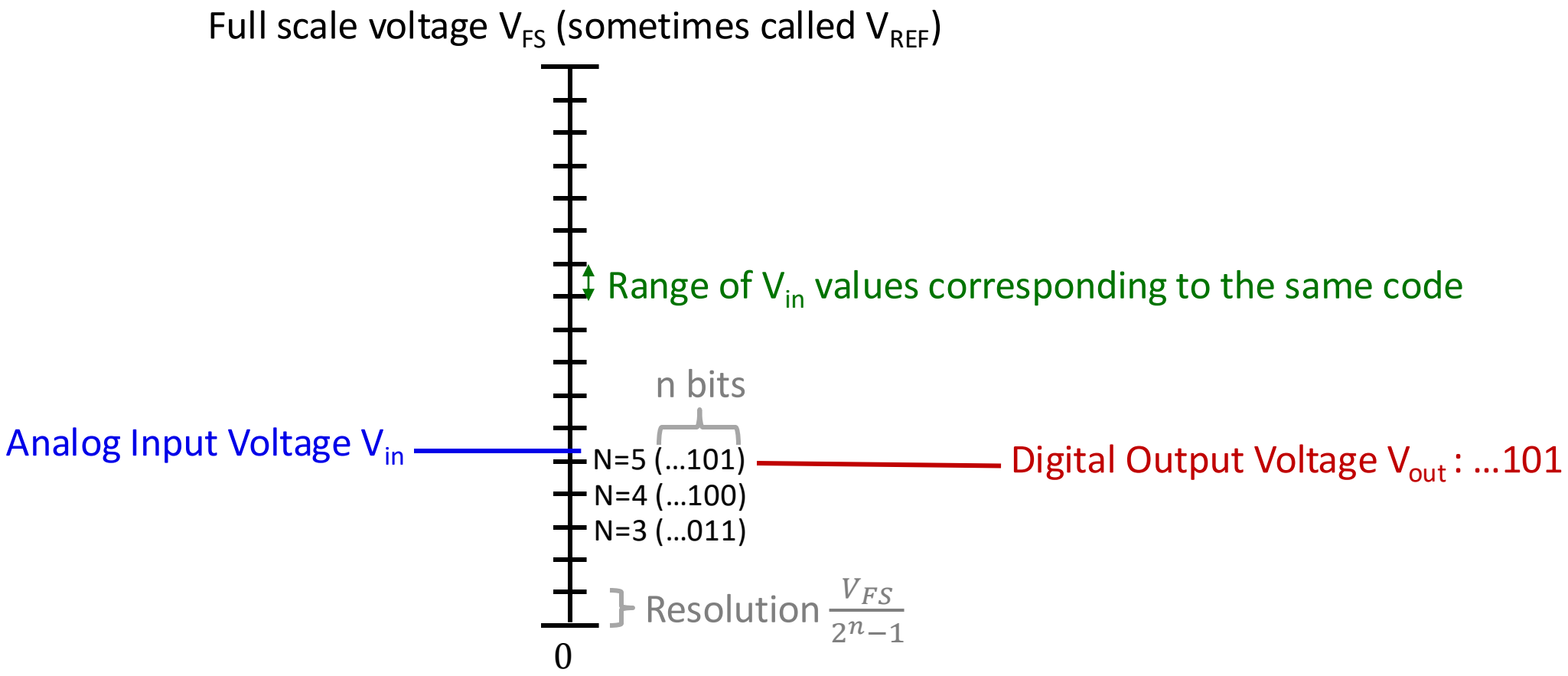
- Let us assume that we are sampling the input signal at constant time intervals, such that **Ts is the sampling period.**

- **The sampling frequency is:**

$$f_s = \frac{1}{T_s}$$

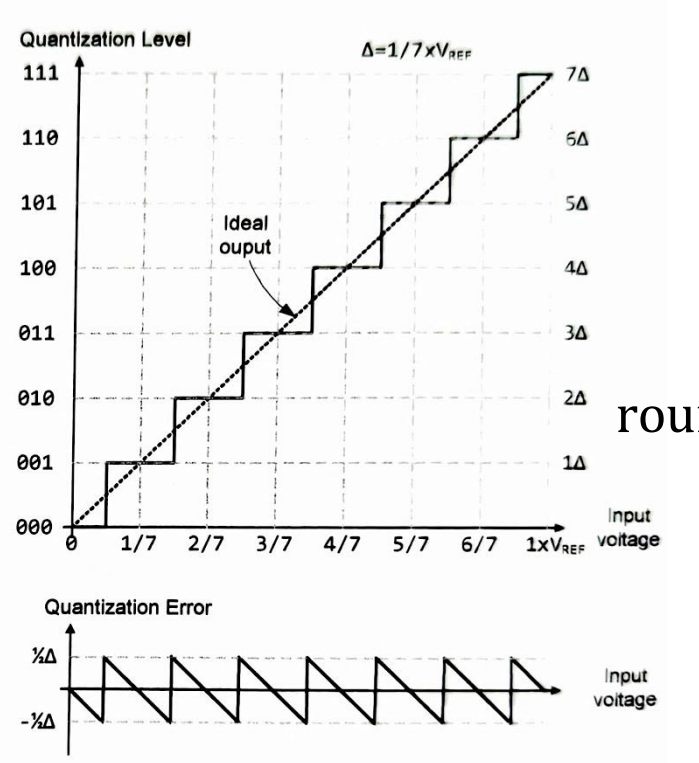
- **Sampling Theorem (Nyquist):** aliasing is avoided if we restrict the frequencies of the incoming signal to less than **half of the sampling frequency *fs*.**

ADC – Quantization



Quantizing and Encoding with n bits

ADC – Quantization



Round function

Digital Result:

$$\text{round}((2^n - 1) * \frac{V}{V_{REF}})$$

Resolution Δ:

$$\Delta = \frac{V_{FS}}{2^n - 1}$$

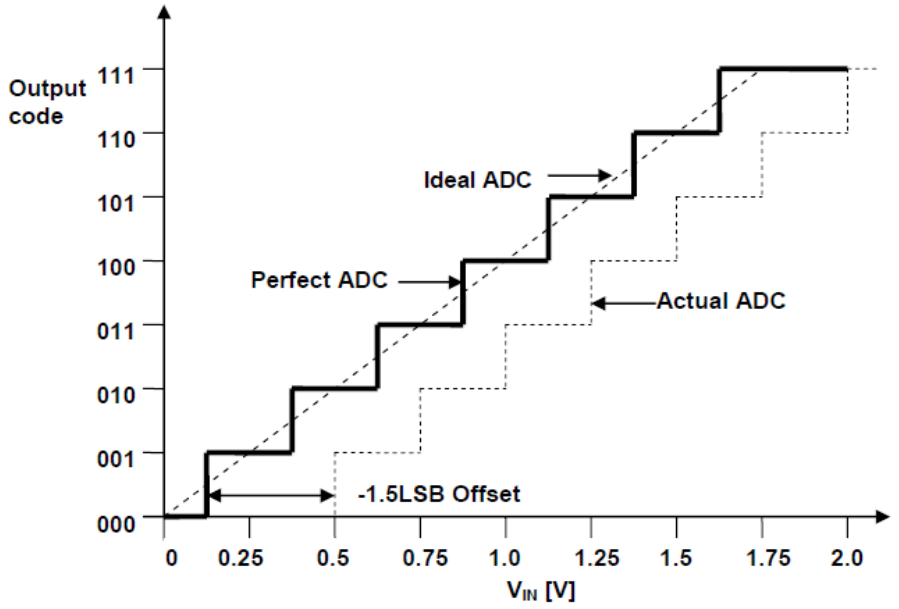
Conversion results:

$$V = \frac{\text{Digital Result}}{2^n - 1} * V_{FS}$$

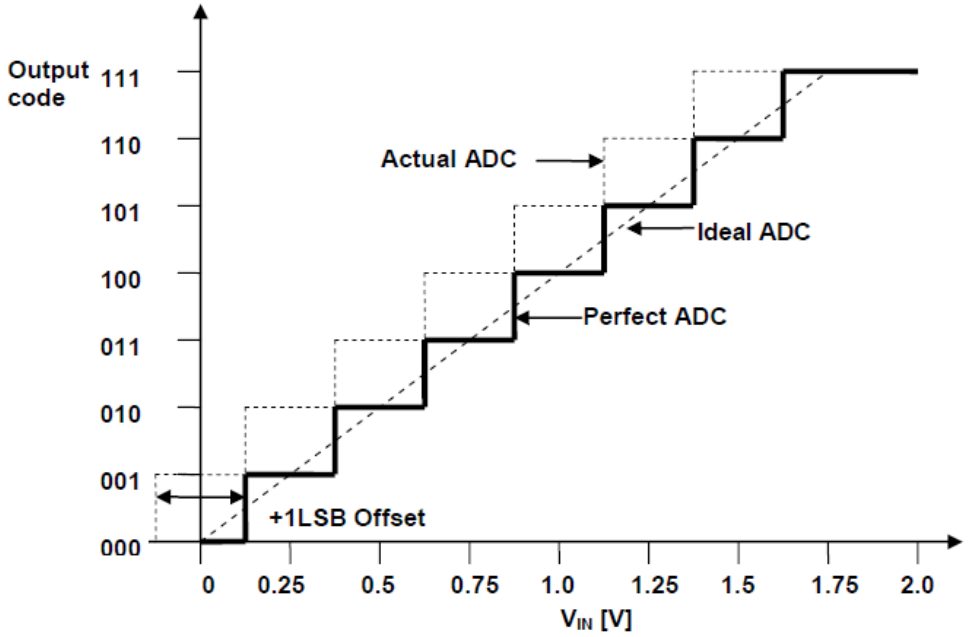
ADC – Offset Error (OE)

Definition: **OE is defined as the deviation of the actual ADC's transfer function from the perfect ADC's transfer function at the point of zero to the transition measured in the Least Significant Bit .**

- Offset error only based on error when applying 0 input voltage.
- Offset error limits the available range of ADC.
- Calculated as the number of least significant bits (LSB).



[1]



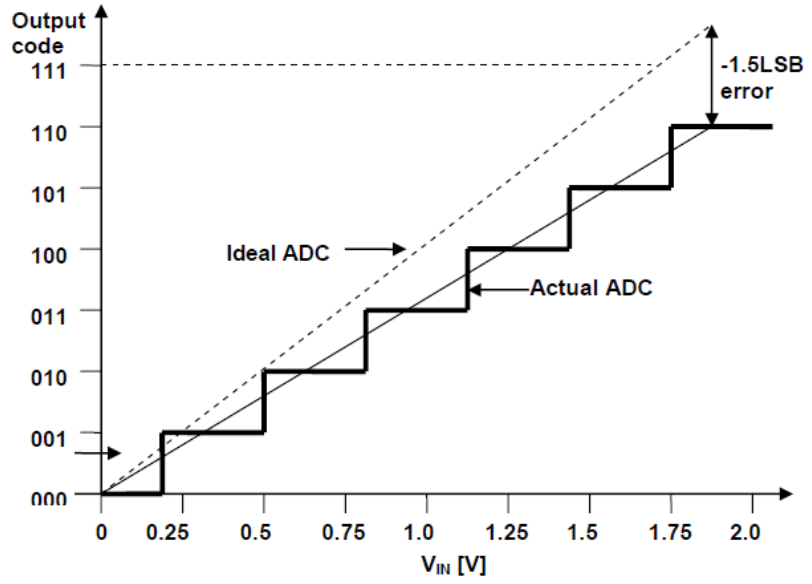
[1]

[1] https://microchipdeveloper.com/adc:adc-offset-error

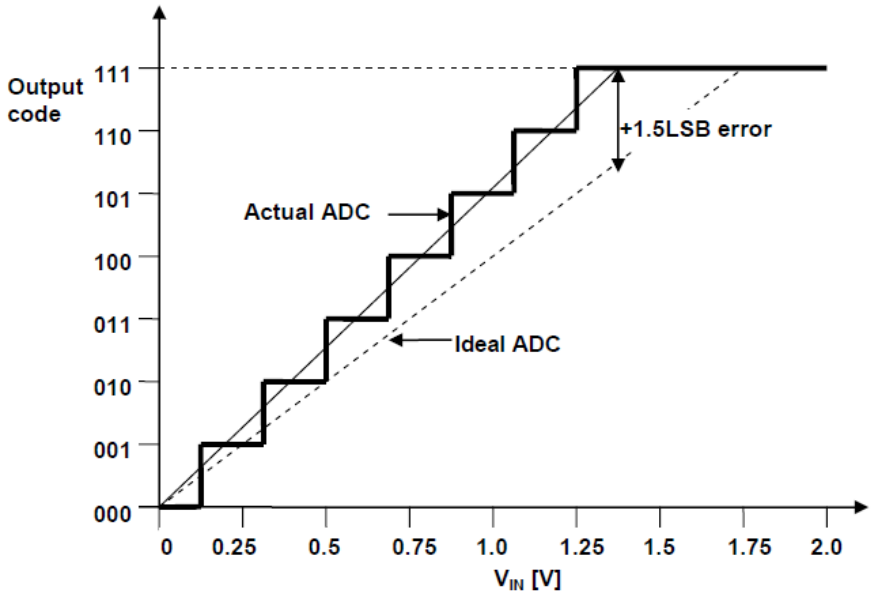
ADC – Gain Error

Definition: gain error is defined as the deviation of the last step's midpoint of the actual ADC from the last step's midpoint of the ideal ADC.

- After having corrected the offset error, there is gain error.
- Calculated as the number of least significant bits (LSB).
- A vertical line drawn between the midpoint of the last step and the ideal ADC line.



[1]

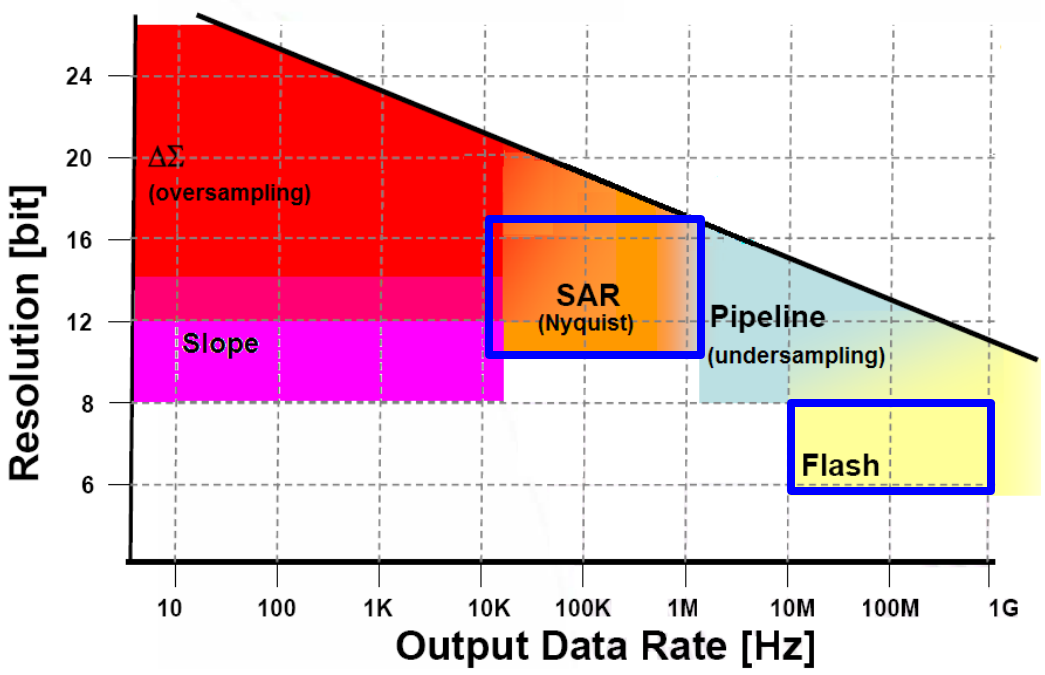


[1]

[1] https://microchipdeveloper.com/adc:adc-gain-error

ADC – Types

ADC performance



[1]

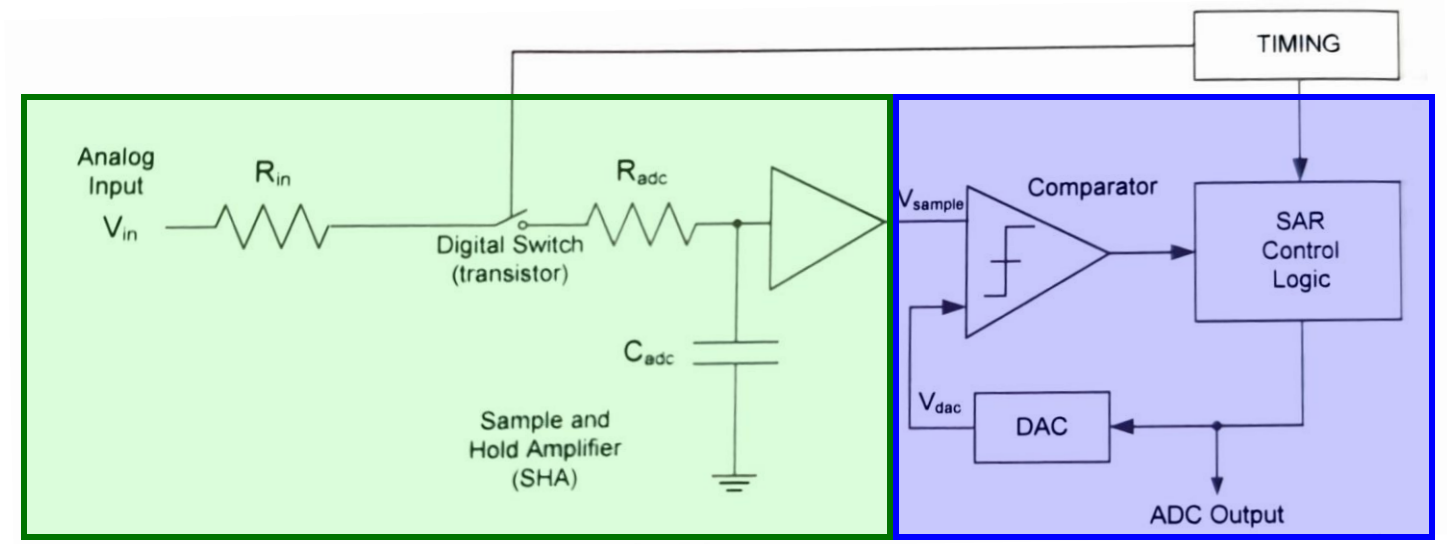
We focus at these 2 ADCs:

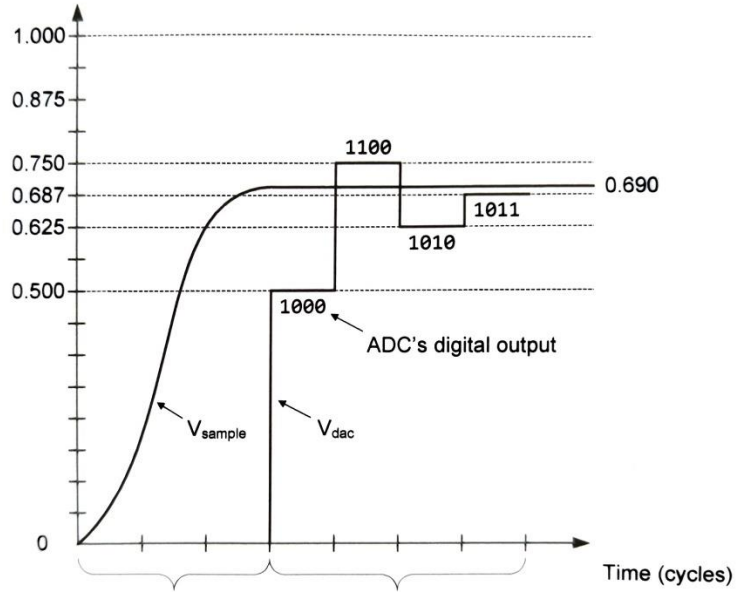
- SAR - ADC
- Flash - ADC

[1] TI MSP430 Teaching Materials

ADC – Successive-Approximation (SAR)

- Consists of Sample-and-Hold Amplifier (SHA) and SAR digital quantization
- SHA circuit does the signal time quantization
  - Convert signal into a series of value
- SAR digital quantization digitizes the value





Conversion Time = Sampling Time + Quantization Time [1]

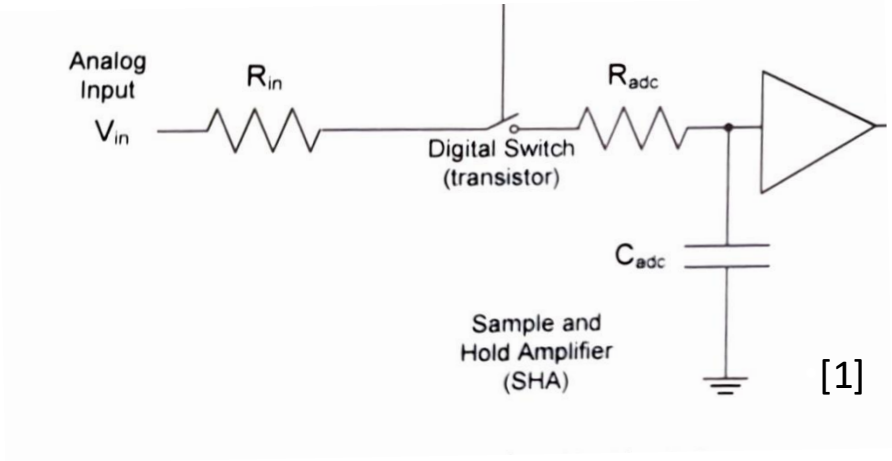
sampling Quantization

[1] Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, page 476

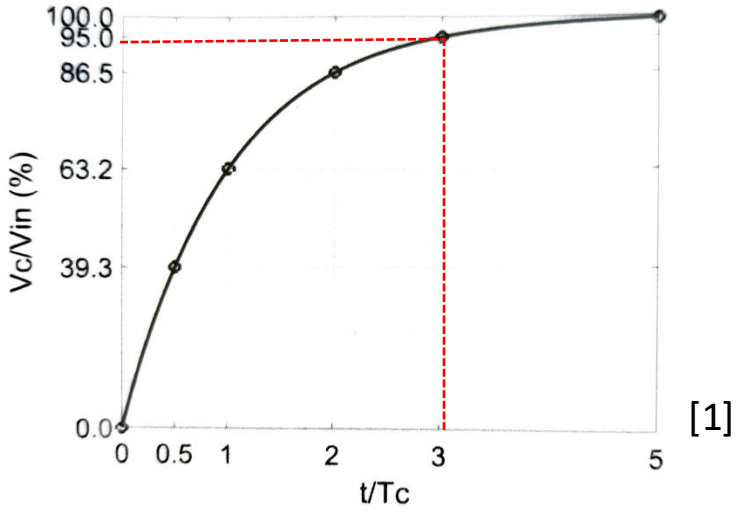
SAR-ADC – Sample-and-Hold

When the switch is closed V_{in} can not immediately be sampled because it needs to settle to a constant voltage (RC circuit).

- $V_C = V_{in}(1 - e^{-\frac{t}{RC}})$ *(see Lecture 0 - RC circuit, charging vs. discharging)*
- After 3 τ_C, V_C reaches 95% of V_{in}. Depending on the quantization error, waiting for three instead of 5 tau might be sufficient.
- Duration of switch being close is called sampling time.



[1]



[1]

[1] Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, page 476-478

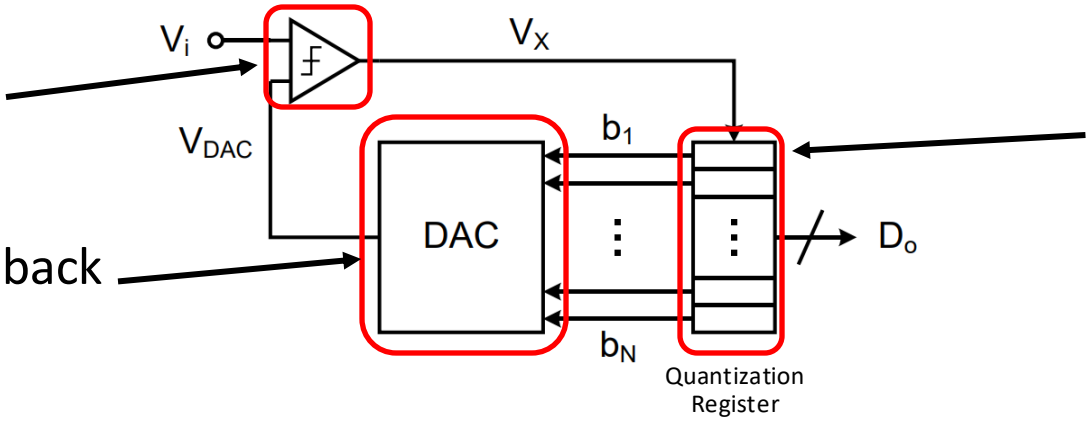
SAR-ADC – Quantization

The SAR-ADC implements a *binary search algorithm* to find the digital representation that most closely represent the analog input voltage.

- For every bit, one voltage comparison is performed:
  - Tradeoff between resolution and conversion time.
  - One comparison cycle usually takes 1 clock cycle.
- Analog voltage is successively approximated, the first comparison defines the MSB.
- The quantization circuit is built out of a comparator, a DAC (Digital-Analog-Converter) and a quantization register to store the conversion progress.

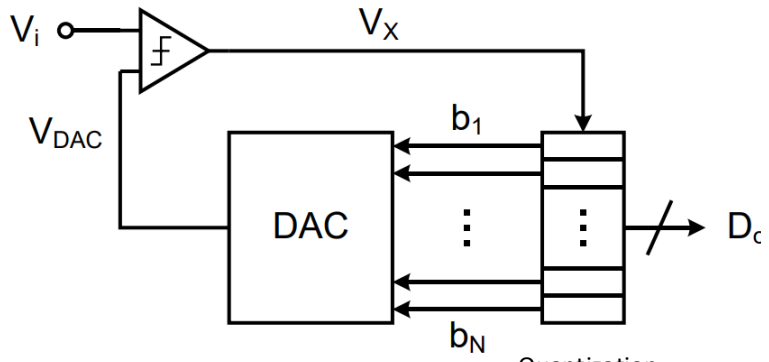
The comparator compares the input voltage with the feedback voltage V_{DAC}

The DAC generates the feedback voltage V_{DAC} based on the previous decisions



A quantization register stores all decisions of all comparisons

SAR-ADC – Quantization (0)

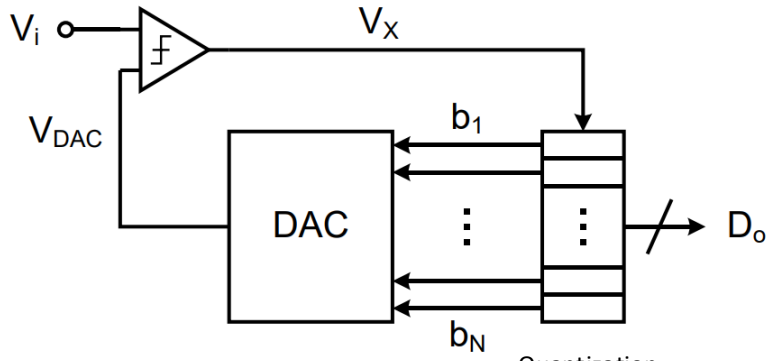


Example: 6-bit SAR-ADC

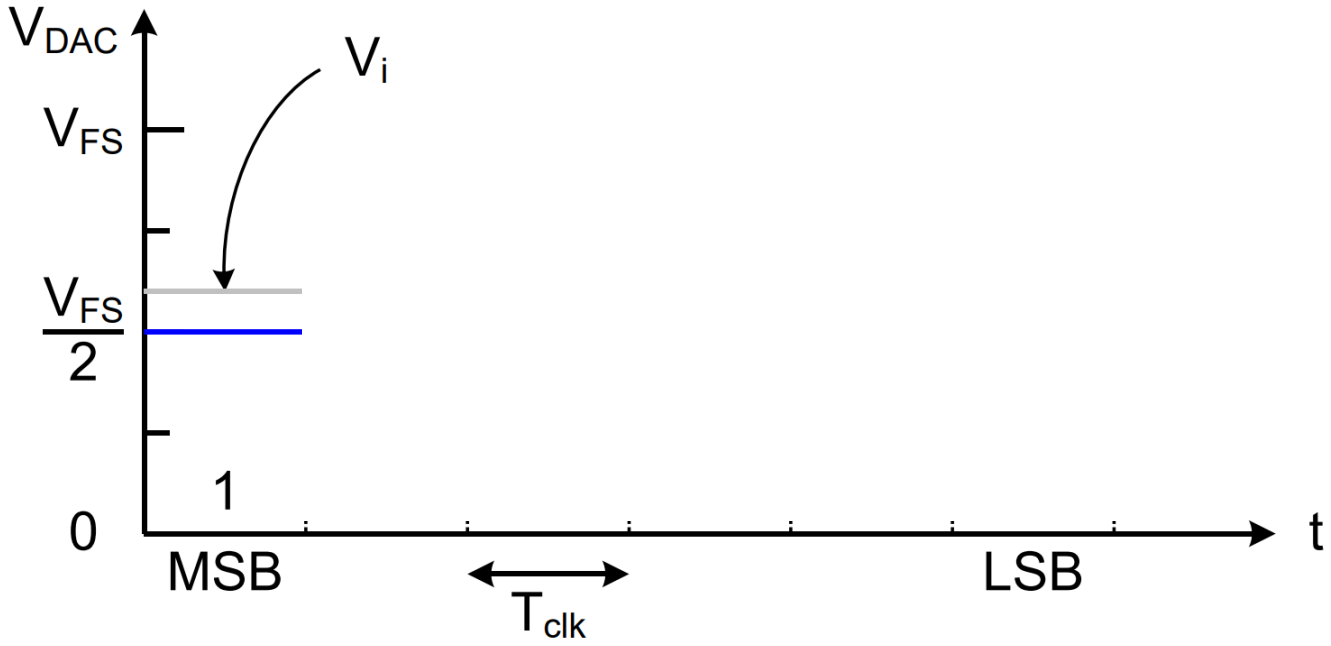
Start conditions of the SAR-ADC conversion process:

- The shift register is *all zero except b_1*
- The Feedback voltage V_{DAC} is set to $V_{FS}/2$ by the DAC's feedback circuitry.
- The comparator decision is not yet initiated and V_X is zero.
- The sample and hold capacitor has been charged with the analog input voltage V_i

SAR-ADC – Quantization (1)

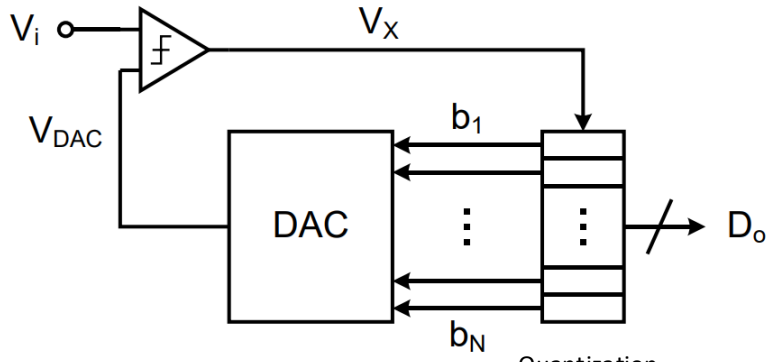


Example: 6-bit SAR-ADC

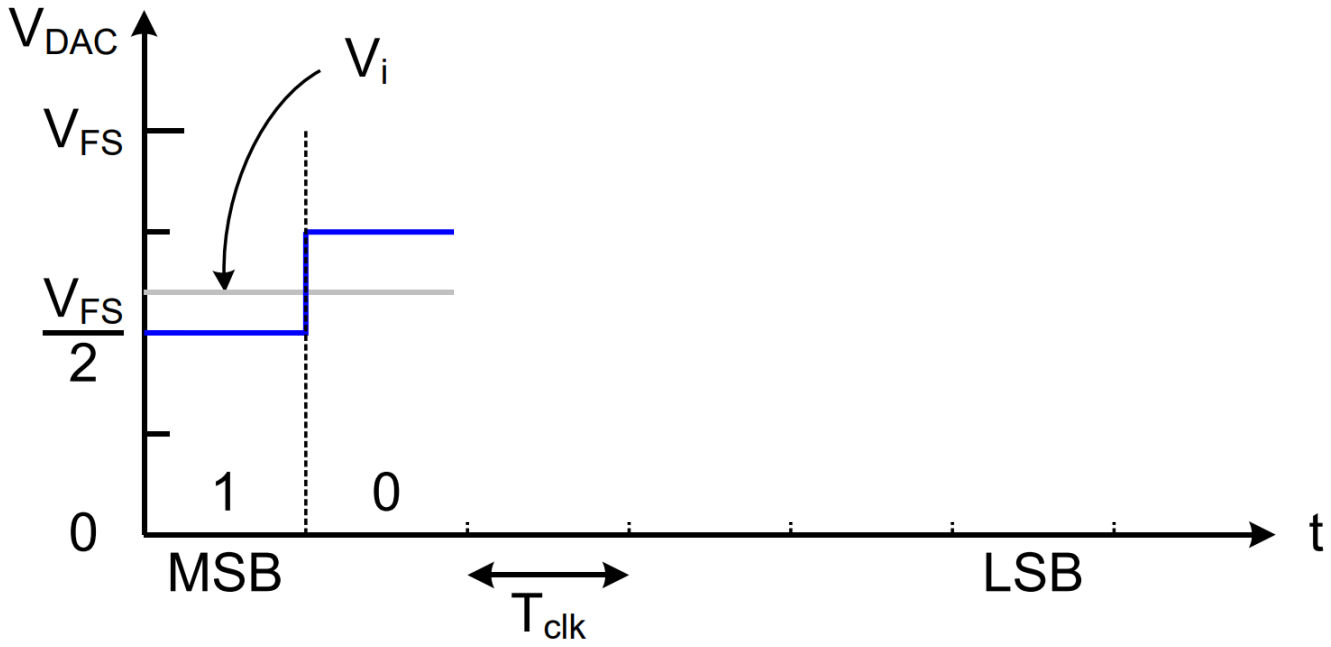


- At the beginning, V_{DAC} is set to $V_{FS}/2$ = **Quantization register value 100000**
- The comparator compares V_{DAC} with the analog input: V_i is larger than V_{DAC} and V_X is high.
- This leads to logic 1 in the quantization register: the MSB bit is thus “1”.

SAR-ADC – Quantization (2)

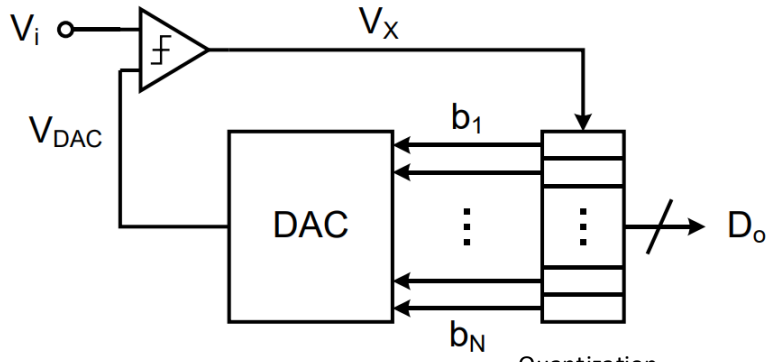


Example: 6-bit SAR-ADC

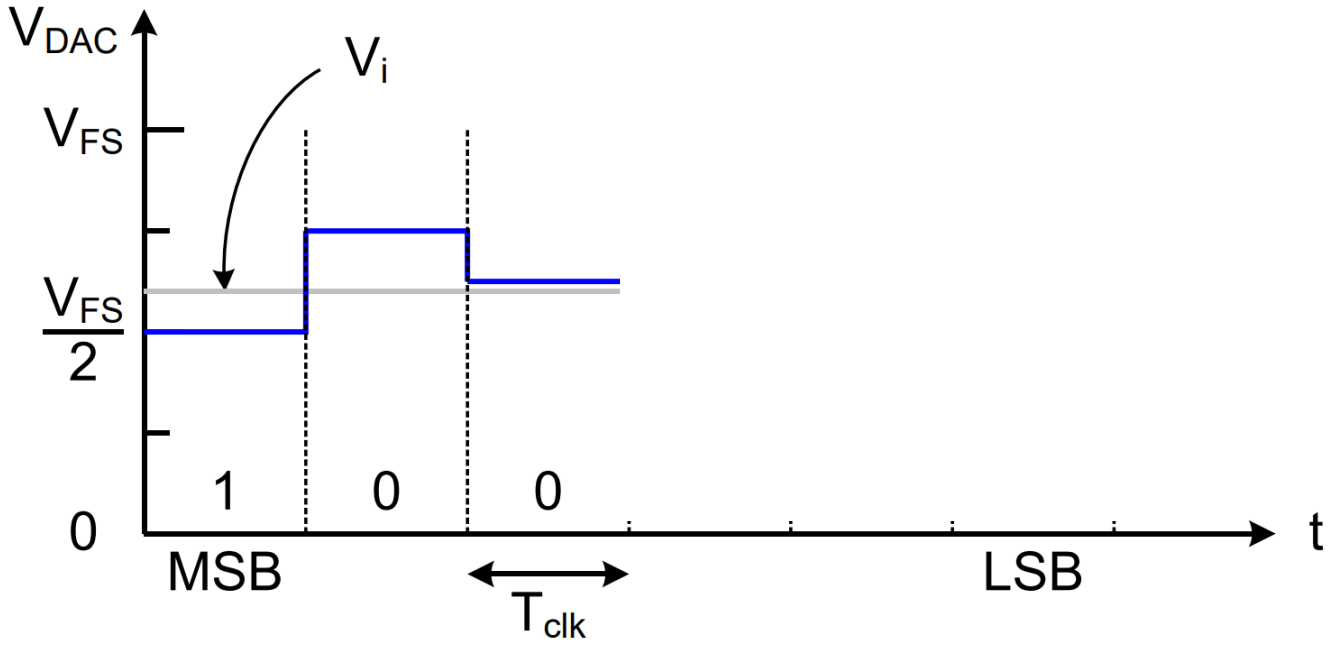


- Since the last conversion yielded logic “1”.
  - V_{DAC} is set to $V_{FS}/2 + V_{FS}/4 = 3V_{FS}/4$
- The comparator compares VDAC with the analog input: Vi is smaller than V_{DAC} and V_X is low.
- This leads to logic 0 in the quantization register: the next bit is thus “0”.

SAR-ADC – Quantization (3)

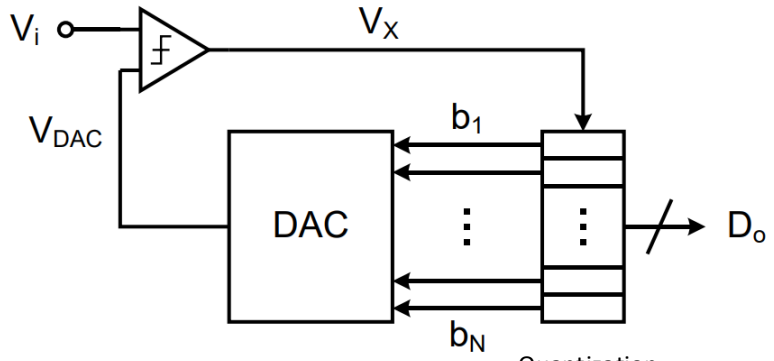


Example: 6-bit SAR-ADC

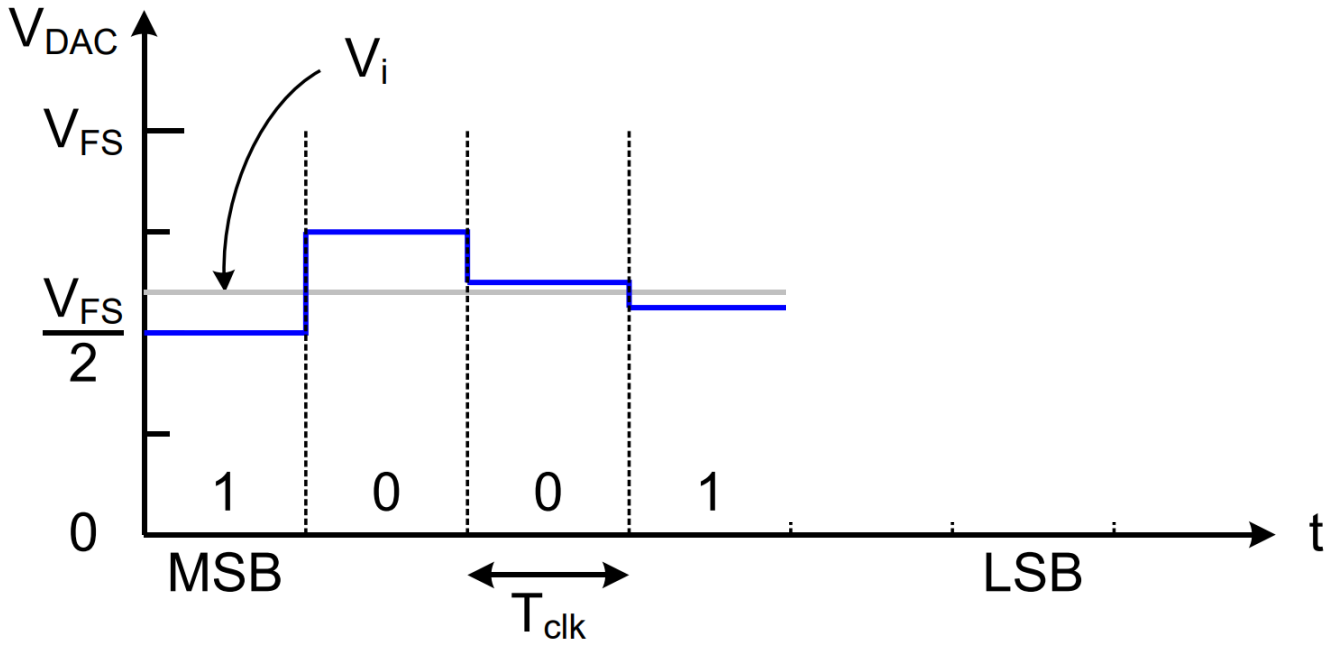


- Since the last conversion yielded logic "0".
  - V_{DAC} is set to $V_{FS}/2 + 0 + V_{FS}/8 = 5V_{FS}/8$
- The comparator compares VDAC with the analog input: V_i is smaller than V_{DAC} and V_X is low.
- This leads to logic 0 in the quantization register: the next bit is thus "0"

SAR-ADC – Quantization (4)

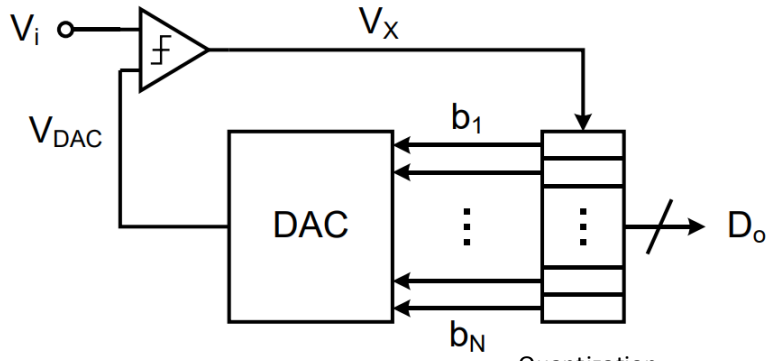


Example: 6-bit SAR-ADC

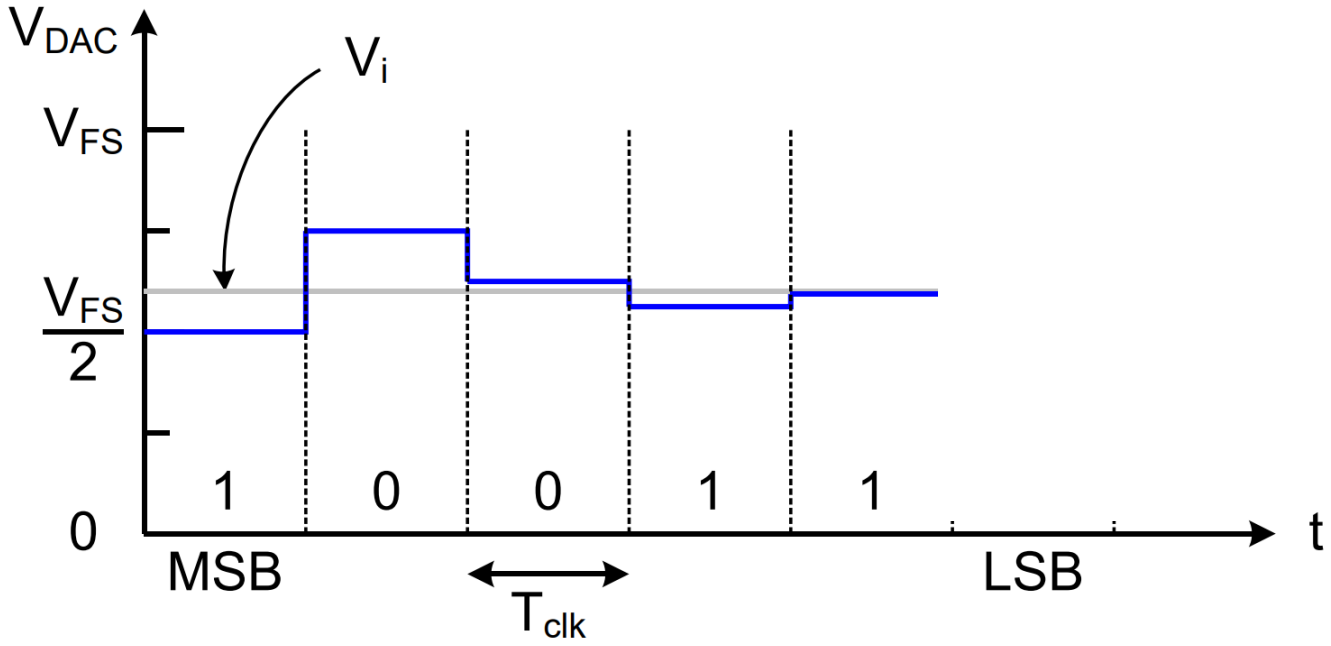


- Since the last conversion yielded logic “0”.
  - V_{DAC} is set to $V_{FS}/2 + 0 + 0 + V_{FS}/16 = 9V_{FS}/16$
- The comparator compares V_{DAC} with the analog input: V_i is larger than V_{DAC} and V_X is high.
- This leads to logic 1 in the quantization register: the next bit is thus “1”.

SAR-ADC – Quantization (5)

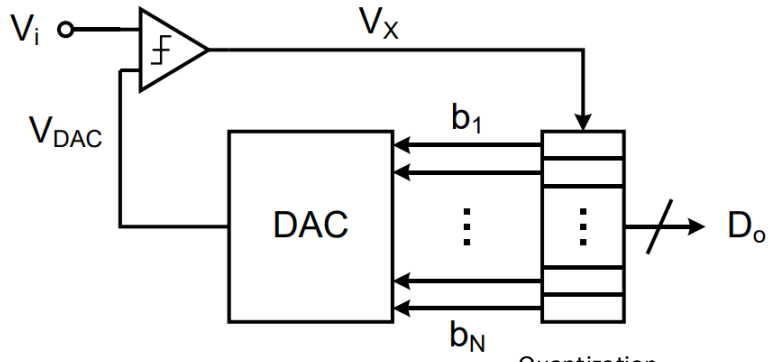


Example: 6-bit SAR-ADC

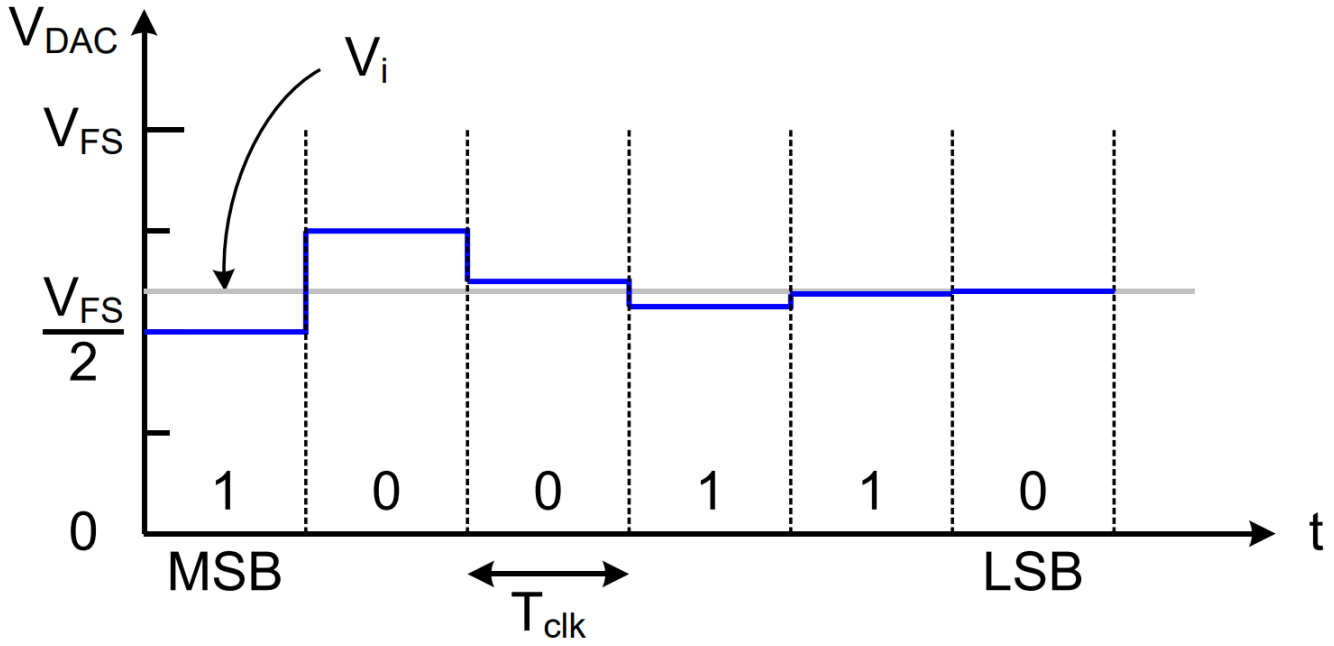


- Since the last conversion yielded logic “1”.
  - V_{DAC} is set to $V_{FS}/2 + 0 + 0 + V_{FS}/16 + V_{FS}/32 = 19V_{FS}/32$
- The comparator compares V_{DAC} with the analog input: V_i is slightly larger than V_{DAC} and V_X is high
- This leads to logic 1 in the quantization register: the next bit is thus “1”.

SAR-ADC – Quantization (6)

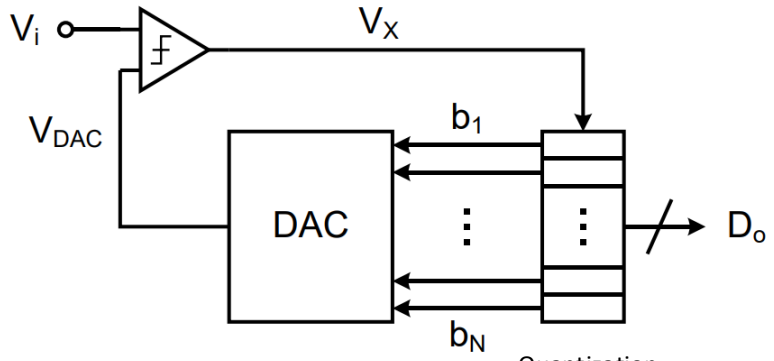


Example: 6-bit SAR-ADC

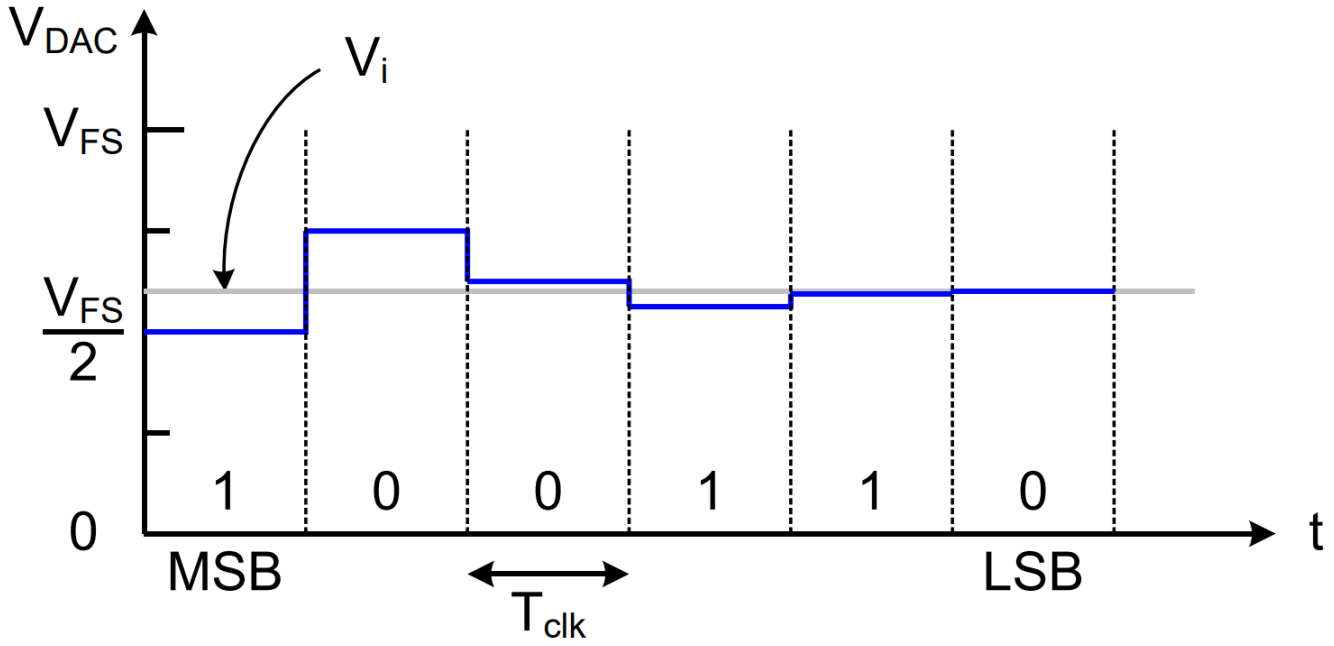


- Since the last conversion yielded logic "1".
  - V_{DAC} is set to $V_{FS}/2 + 0 + 0 + V_{FS}/16 + V_{FS}/32 + (V_{FS}/64) = 38V_{FS}/32$
- The comparator compares V_{DAC} with the analog input: V_i is slightly lower than V_{DAC} and V_X is high.
- This leads to logic 0 in the quantization register: the next bit is thus "0".

SAR-ADC – Quantization (7)



Example: 6-bit SAR-ADC

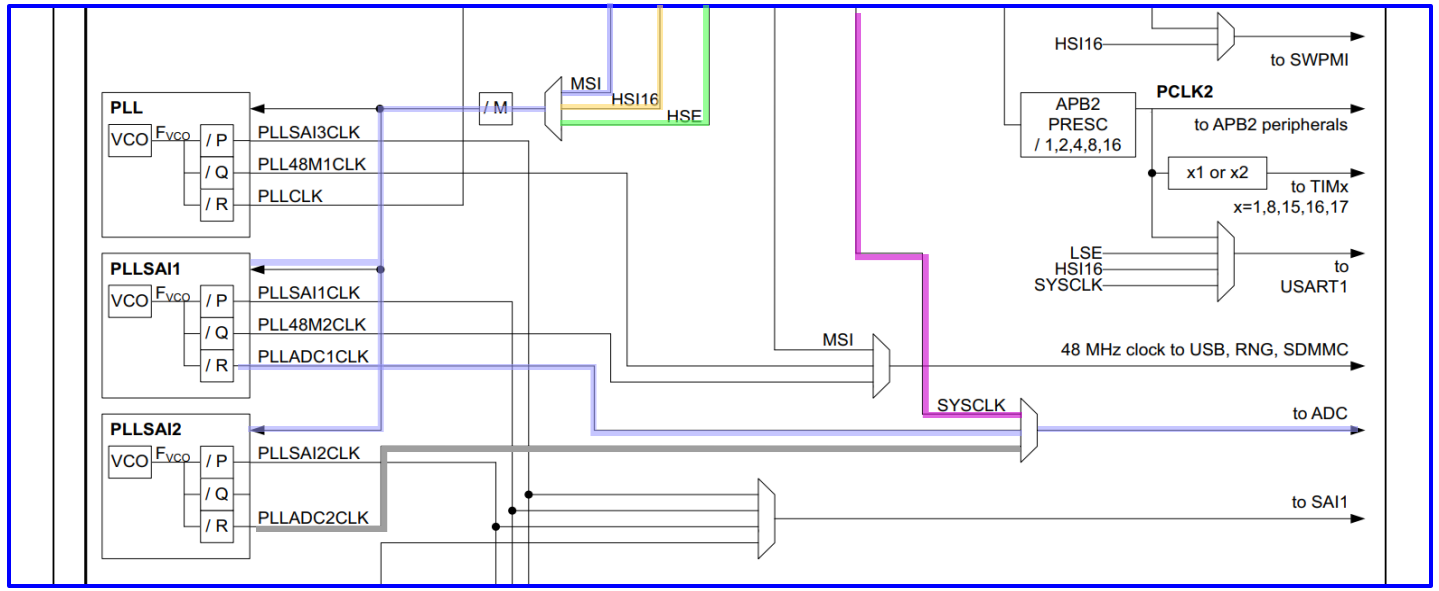


- The conversion ends when the quantization register is full.
  - Higher-order SAR-ADCs have better precision, but they need more conversion steps.

- Output: 100110: $V = \frac{\text{Digital Result}}{2^n - 1} * V_{FS} = \frac{38}{63} * V_{FS} = 0.603\ V_{FS}$

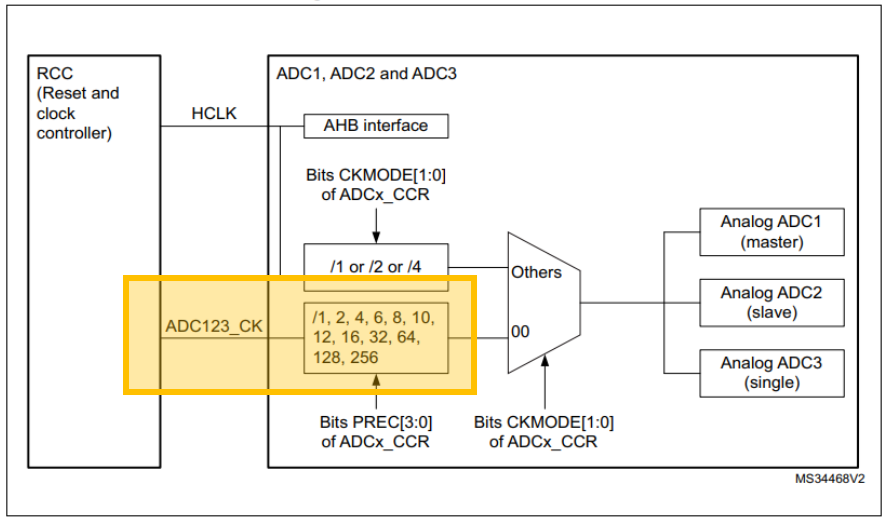
ADC STM32L476xx – Clock

- Clock Tree from STM32L476xx from Lecture 2.
- Choose a clock source depending on the requirements.
- The c lock frequency can further be divided using the ADC's internal prescaler.



[1]

Figure 67. ADC clock scheme



[2]

[1] Datasheet STM32L476xx

[2] Reference Manual RM0351

ADC STM32L476RG

- By reducing the number of ADC bits the conversion can be sped up.

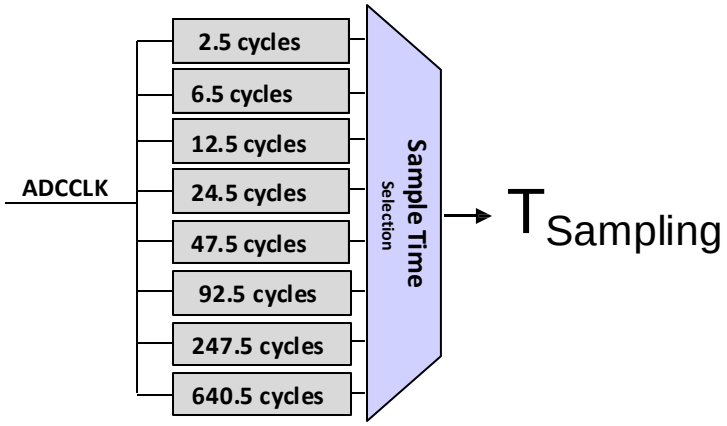


Table 110. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at F_{ADC}= 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling Time= 2.5 ADC clock cycles)	T_{CONV} (ns) at F_{ADC}= 30 MHz
12	12.5 ADC clock cycles	416.67 ns	15 ADC clock cycles	500.0 ns
10	10.5 ADC clock cycles	350.0 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	11 ADC clock cycles	366.67 ns
6	6.5 ADC clock cycles	216.67 ns	9 ADC clock cycles	300.0 ns

$T_{Quantization}$ $T_{conversion}$ [1]

$$T_{Conversion} = T_{Sampling} + T_{Quantization}$$

[1] Reference Manual RM0351

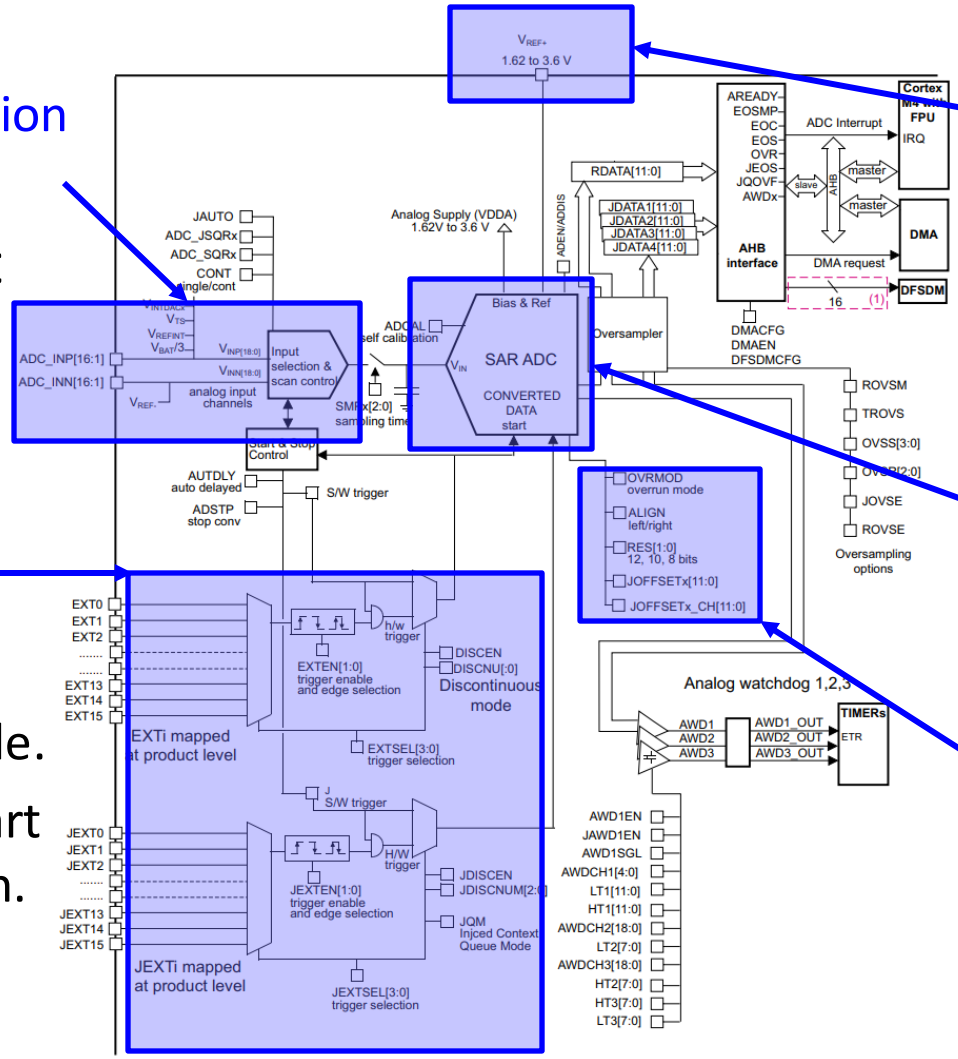
ADC – Hardware Architecture on STM32L476RG

ADC input selection over multiplexer

- Multiple ADC channels per SAR-ADC.

ADC trigger

- Software and hardware trigger available.
- Control the start of a conversion.



Reference Manual STM32L47xxx, RM0456, page 508

Analog voltage domain and reference voltage

- Vref builds the reference voltage for the ADC conversion.

Successive approximation ADC (SAR-ADC)

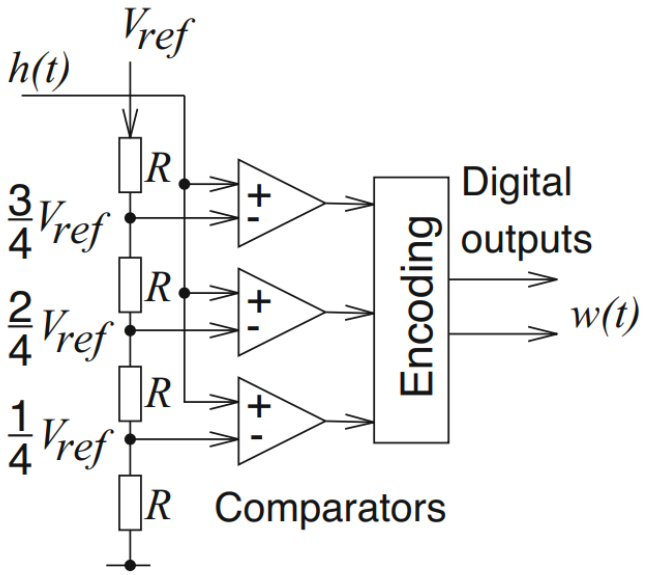
- Converts analog voltage into a digital representation (Vref dependent).

Calibration registers

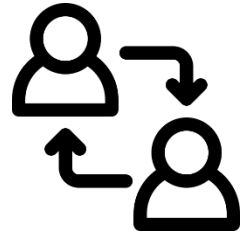
- ADC needs to be calibrated for precise measurements.

ADC – Flash ADC

- Also known as parallel ADC.
- The resistive divider provides different reference voltages to compare against.
- Fastest way to convert an analog signal to a digital signal.
- For an N-bit converter, the circuit needs 2^N-1 comparators.



Interaction: Analog or Digital?



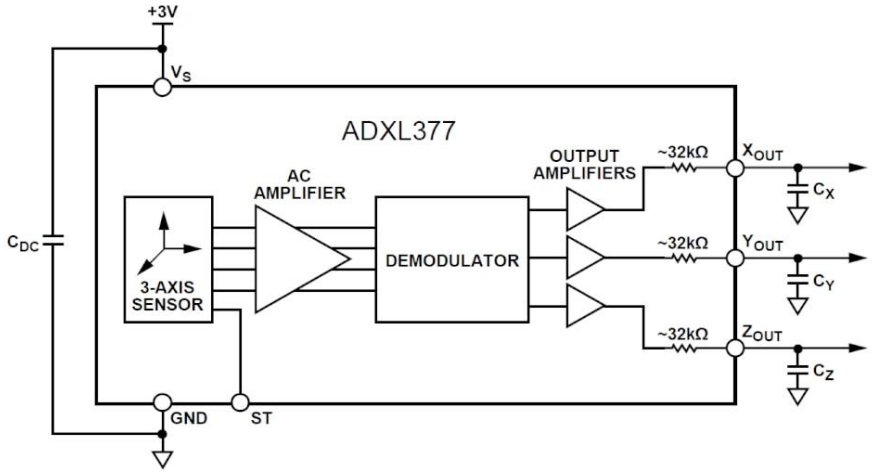
Sensors can include both transducers and ADCs. *When is a sensor to be considered as analog, and when as digital*? Are both digital? Are both Analog? Discuss with your colleagues in groups of 2-3.

Interaction: Analog or Digital?

Sensors can include both transducers and ADCs. *When is a sensor to be considered as analog, and when as digital*? Are both digital? Are both Analog? Discuss with your colleagues in groups of 2-3.

Sensor without an ADC

The sensor provides an analog output



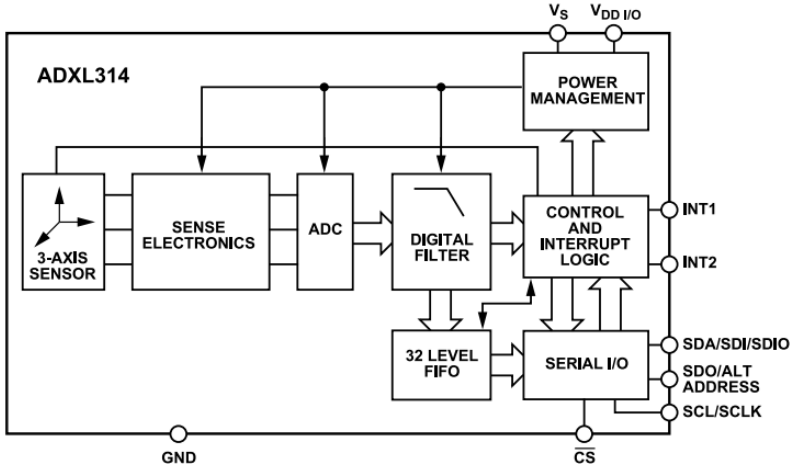
\+ Low cost and low power consumption

\- Output not always linear

\- ADC must be close to sensor

Sensor with ADC

The sensor provides a digital output



\+ Easy to integrate

\+ Already digital

\- More power

Icon by Freepik

What Did You Learn?

- ✓ Physical world is analog
- ✓ Timers are counters
  - ✓ Counting modes
  - ✓ ARR, CCR
- ✓ PWM
- ✓ ADC
  - ✓ Quantization in time and amplitude
  - ✓ ADC error types
  - ✓ ADC types
  - ✓ SAR-ADC