

12.24196

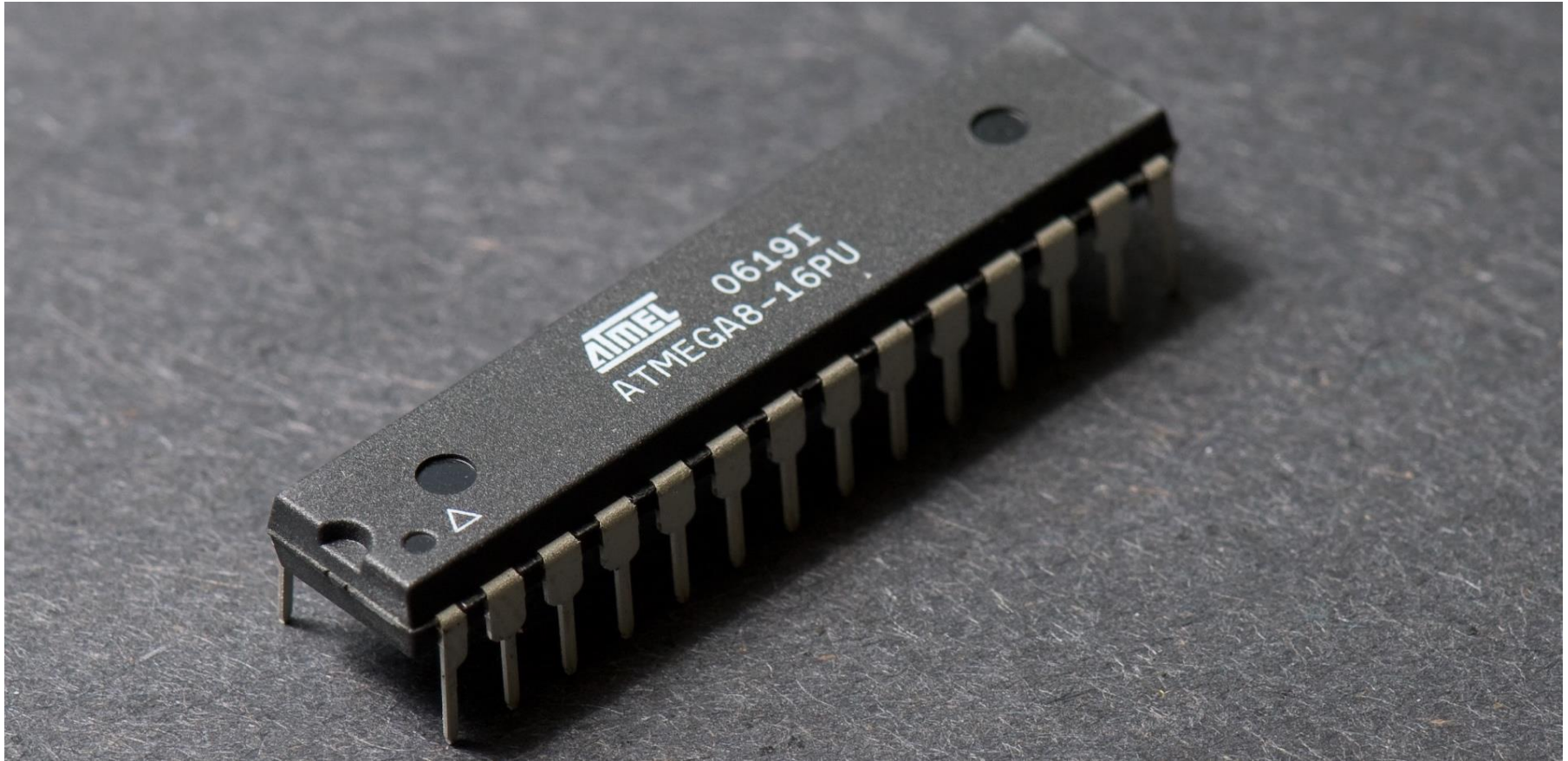
# Introduction to Embedded Systems

Prof. Dr.-Ing. Stefan Kowalewski | Julius Kahle, M. Sc.  
Summer Semester 2025

Part 1

## Microcontrollers

# Microcontrollers



ATMega 8

© Peter Halasz (Creative Commons License)

# Content

## Microcontrollers

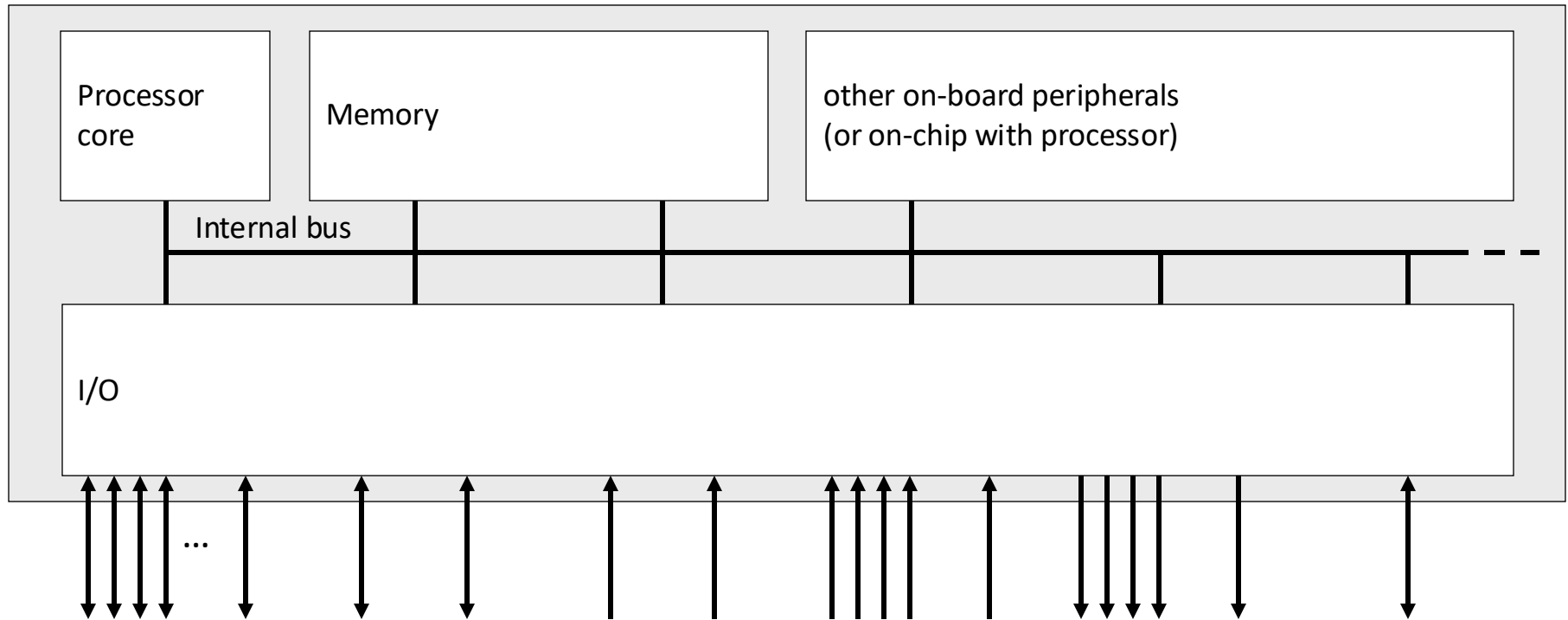
1. Basics
2. Structure/elements
3. Digital I/O
4. Interrupts
5. Timers/Counters
6. Analog I/O

# Microprocessors vs. microcontrollers?

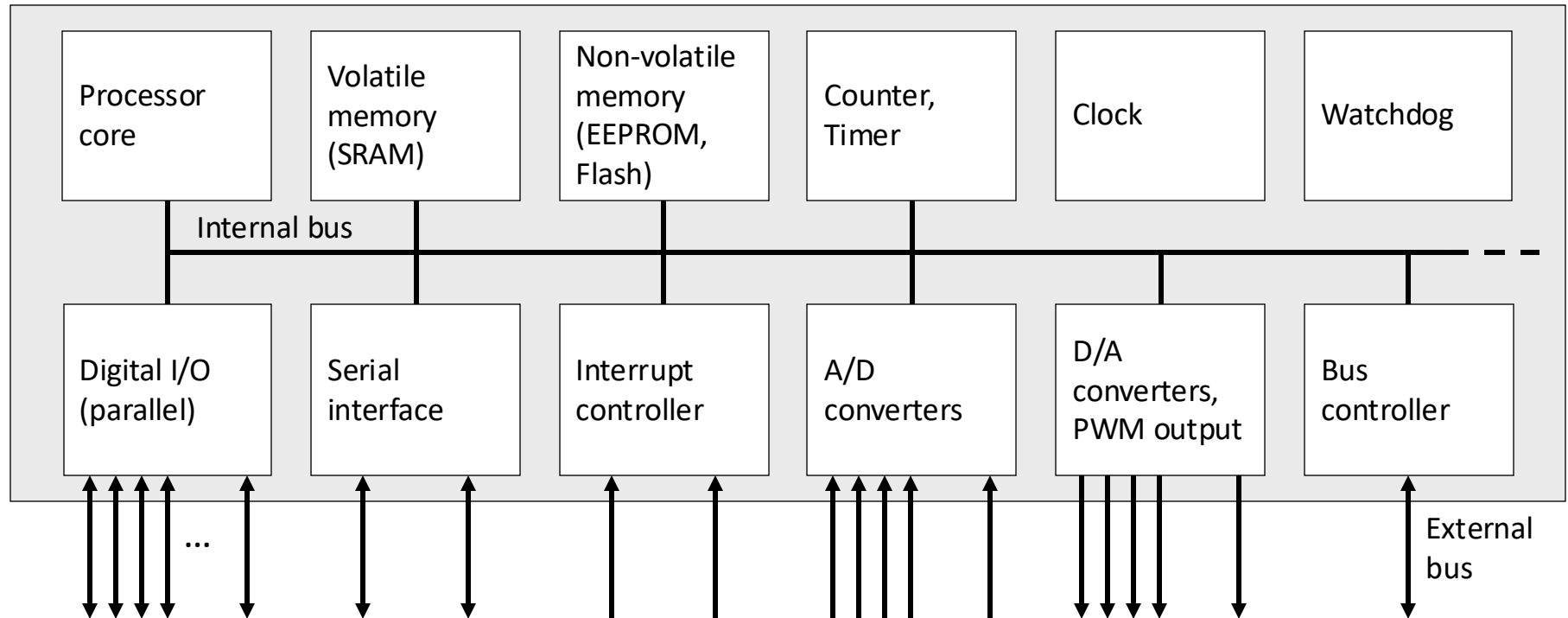
---

- ▶ Microcontroller (often abbrev. “ $\mu$ C”):
  - stand-alone device for embedded applications
  - $\approx$  low-end microprocessor + memory + I/O + additional peripherals
  - not a general-purpose device
  - cost-optimized control unit for particular application area
  - (but more general than Application Specific Instruction Set Processors (ASIPs) and Systems-on-Chip (SoCs))
  
- ▶ Microcontroller family:
  - Same microprocessor
  - Scalability w.r.t. memory, I/O capabilities, on-chip peripherals, etc.

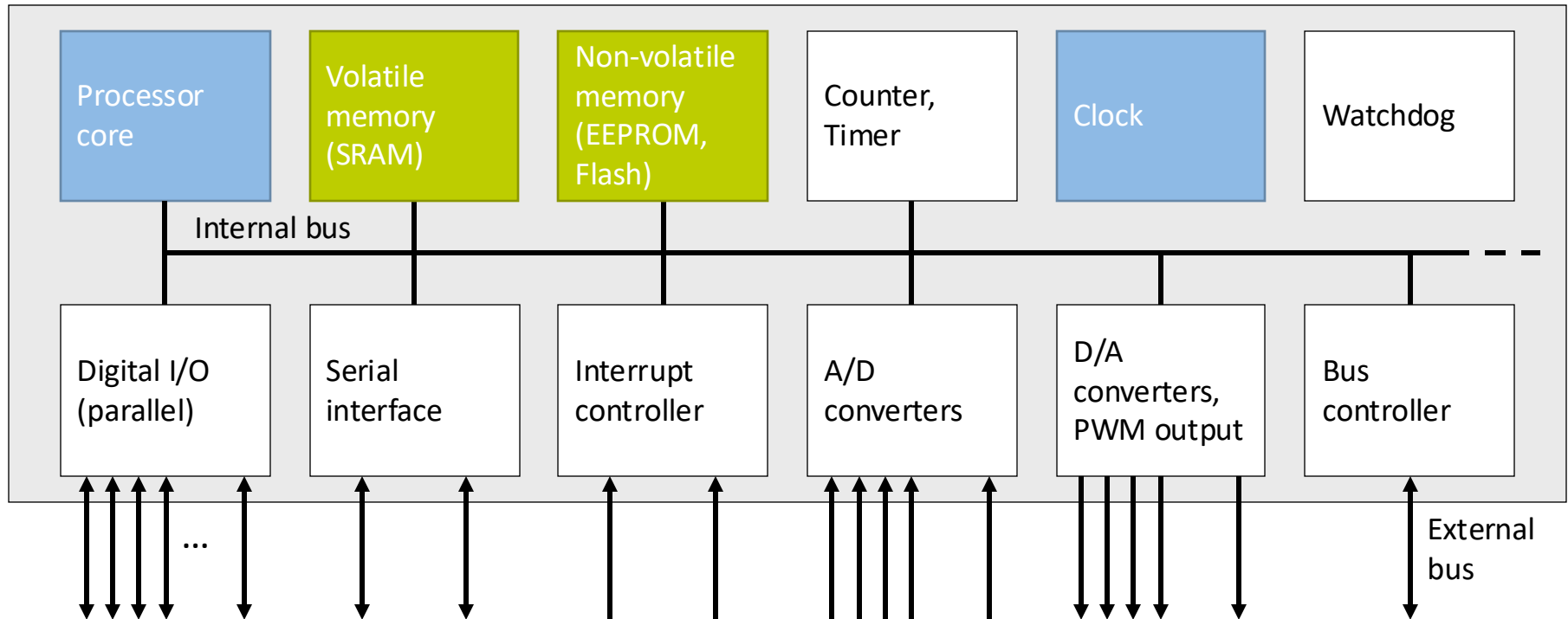
# Basic structure of a microcontroller



# Basic structure of a microcontroller - refined



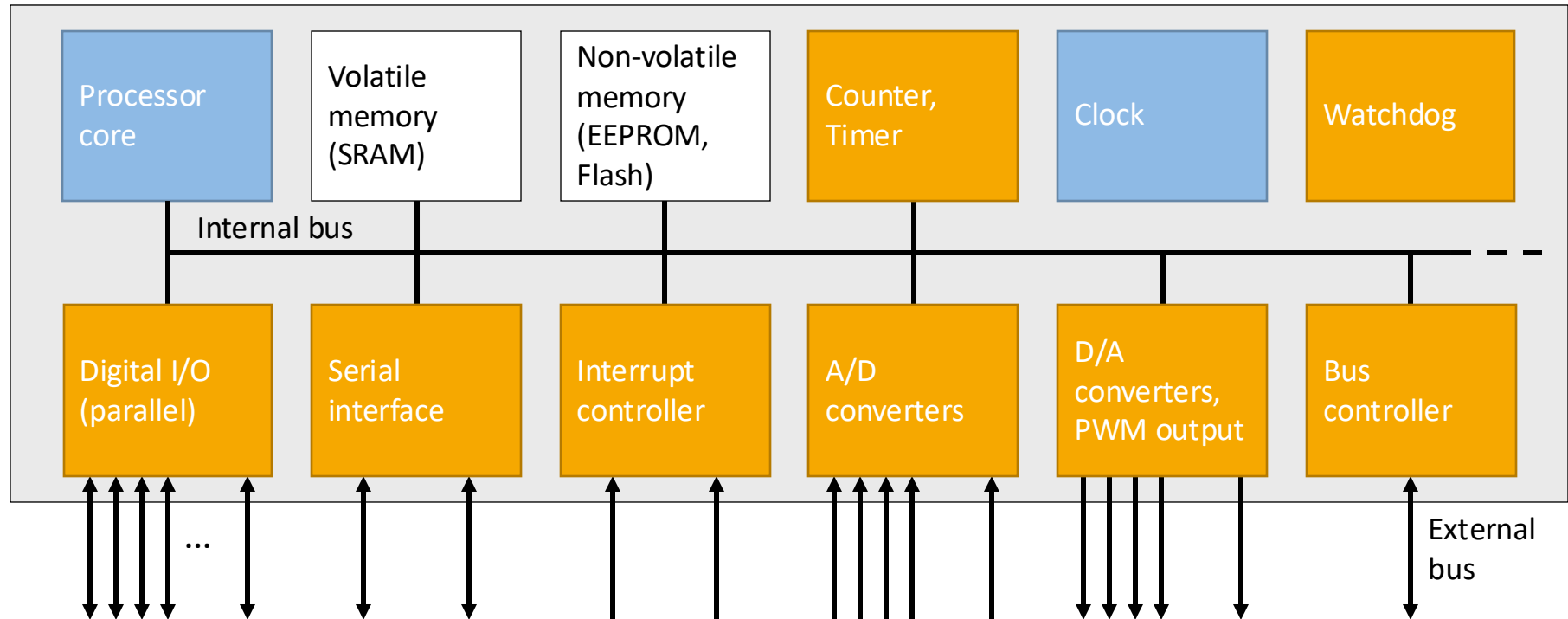
# How to access internal blocks? - memory



## Memory:

- all memory types share a common address range or
- different memory types are mapped into one address range (if you use C the compiler handles most of it)

# How to access internal blocks? - digital I/O & on-chip peripherals



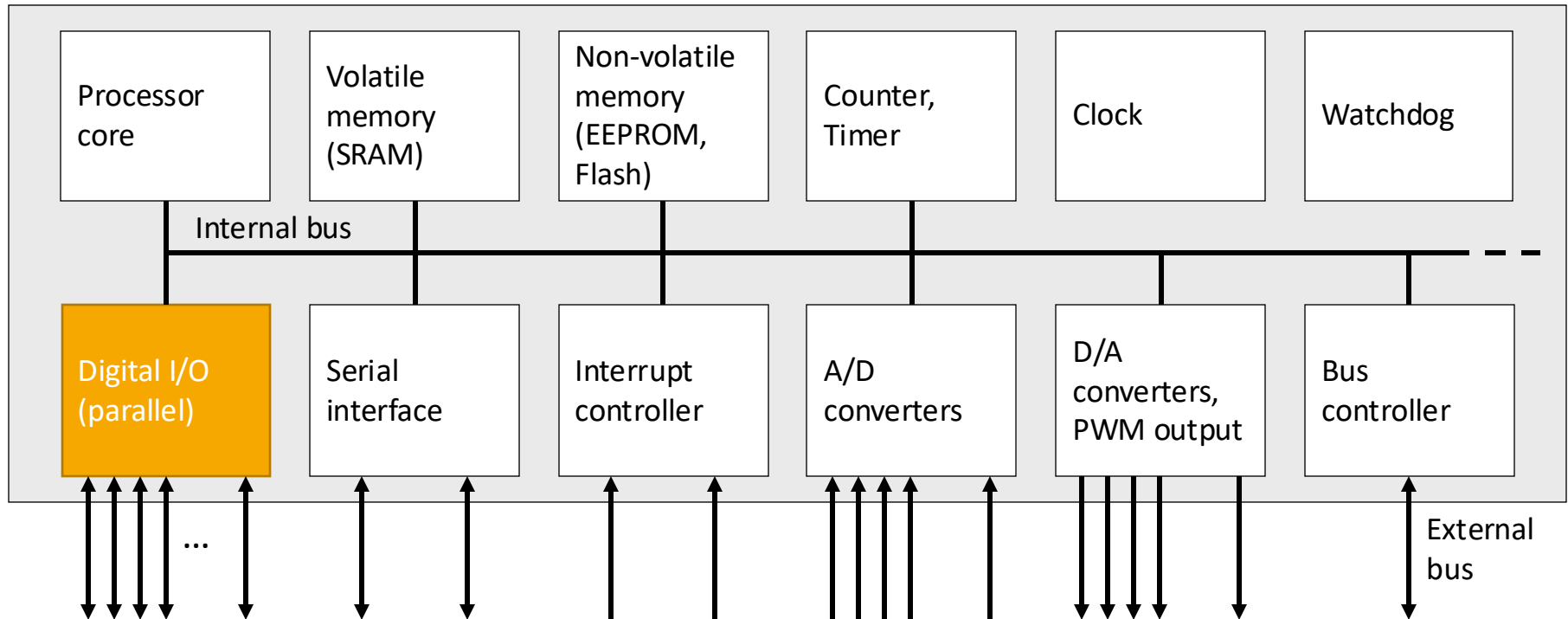
Digital I/O and on-chip peripherals are accessed by dedicated registers.



# Content

1. Basics
2. Structure/elements
3. Digital I/O
4. Interrupts
5. Timers/Counters
6. Analog I/O

# Basic structure of a microcontroller - refined



# Digital I/O pins

- ▶ Basic means to monitor and control external hardware.
- ▶ Usually, digital I/O pins PORTA
  - are grouped into ports of 8 pins (on 8-bit architecture).
  - are bidirectional (i.e., can be used as input or output pins)
  - can have alternate functions (i.e., can be used for purposes different than digital I/O, e.g., as analog I/O pins)
- ▶ Monitoring, access and control of digital I/O pins is done via three special registers for each port:
  - Data Direction Register (DDR)
  - Port Register (PORT)
  - Port Input Register (PIN)

# Control of digital I/O pins via registers

## ► Data Direction Register (DDR):

- read/write
- specifies for each bit of the corresponding port whether it is an input or an output bit

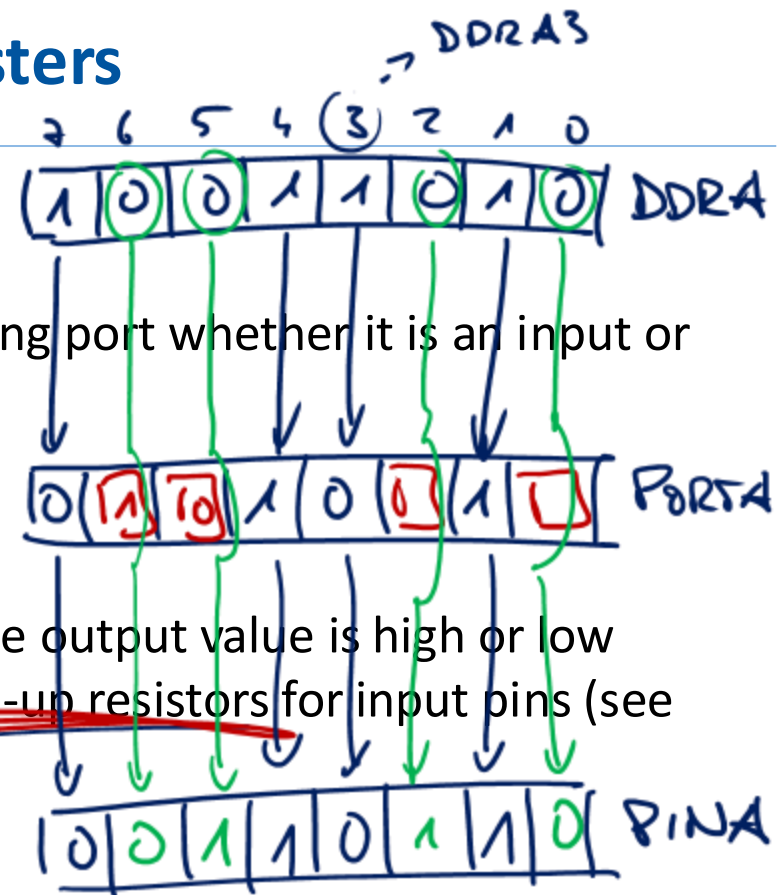
output = 1  
input = 0

## ► Port Register (PORT):

- read/write
- specifies for the output pins whether the output value is high or low
- ATmega16: also used for controlling ~~pull-up~~ resistors for input pins (see next slides)

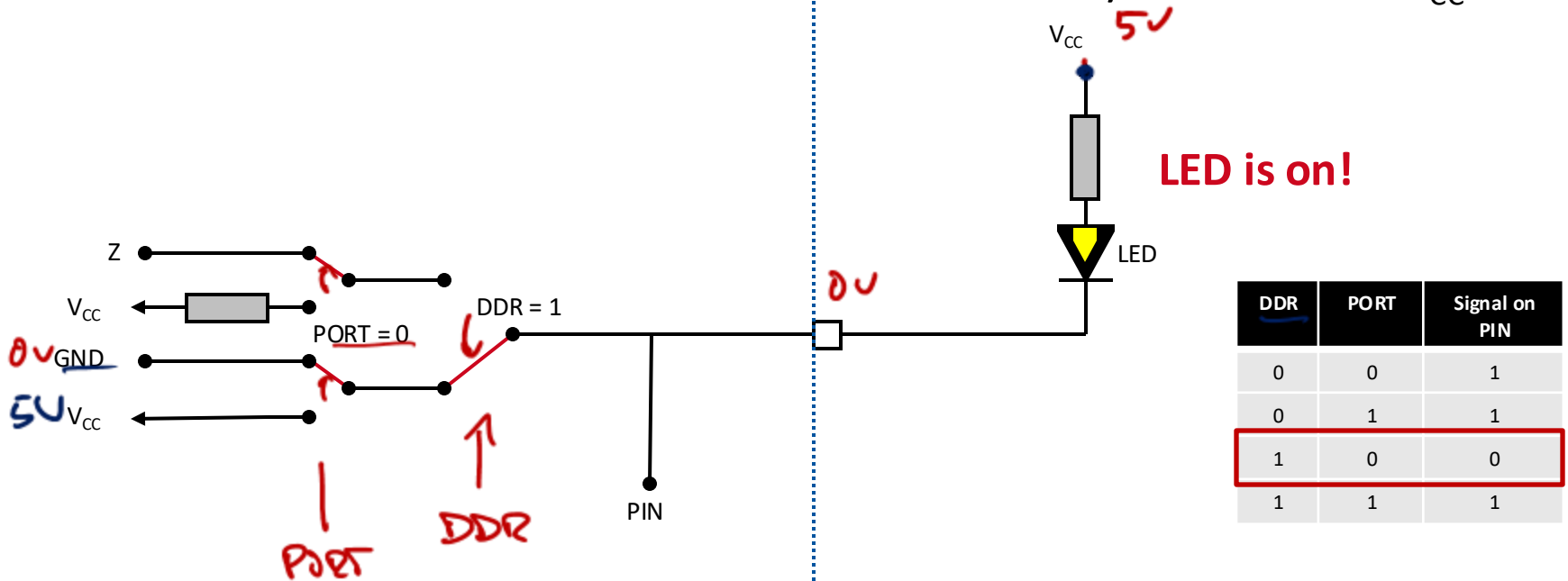
## ► Port Input Register (PIN):

- read only (writing has no effect or unintuitive semantics)
- contains the current value (high or low) of all pins (input and output)
- usual purpose: reading values of input pins



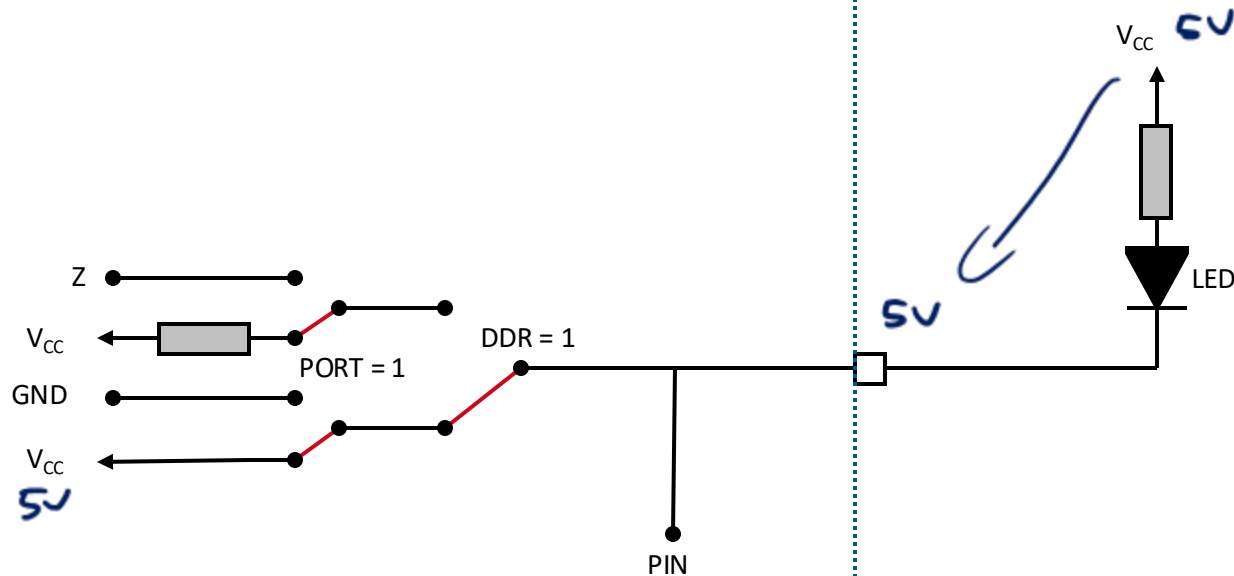
# Example: LED control

$\mu C$



# Example: LED control

$\mu C$



LED is off!

DDR	PORT	Signal on PIN
0	0	1
0	1	1
1	0	0
1	1	1

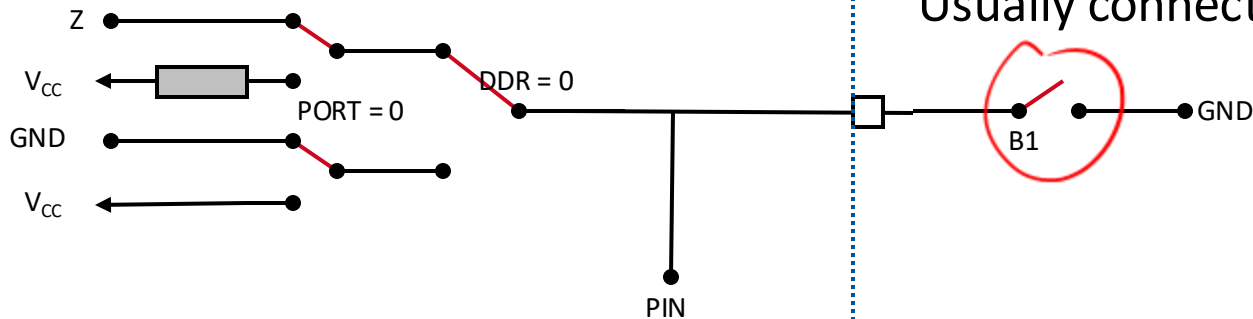
# Digital Output

---

- ▶ When a pin is configured as output, the controller drives the pin according to the PORT value of that pin.
- ▶ Usually: logic 1 → VCC and logic 0 → GND
- ▶ The electric current depends on the connected circuits:
  - Short circuit fault is possible
  - External current limiter might be needed

# Example: Reading a button

$\mu C$



**PIN is undefined  
when B1 is open**

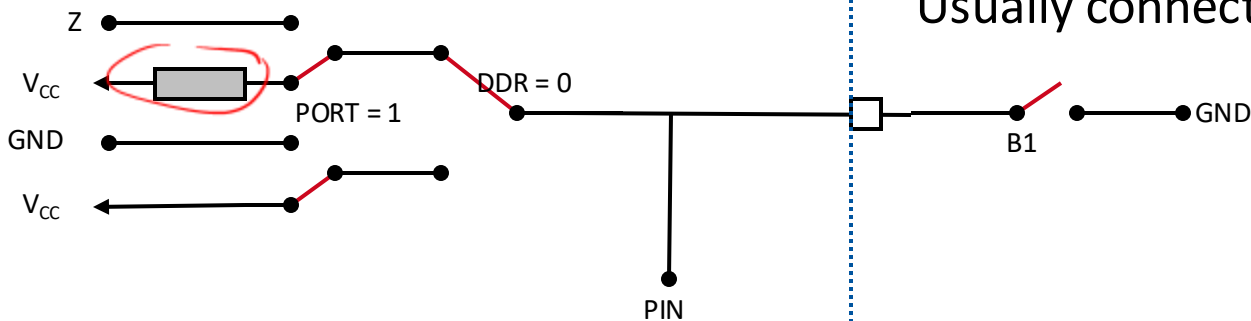
Usually connected to GND!

DDR	PORT	Signal on PIN
0	0	?
0	1	1
1	0	0
1	1	1



$\mu C$

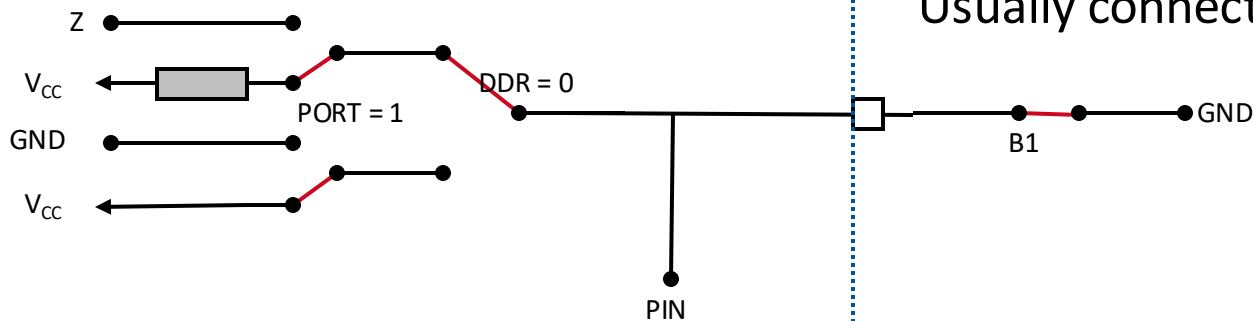
Usually connected to GND!



DDR	PORT	Signal on PIN
0	0	?
0	1	1
1	0	0
1	1	1

# Example: Reading a button

$\mu C$





Pull-up resistors allow  
the voltage to collapse

Usually connected to GND!

DDR	PORT	Signal on PIN
0	0	0
0	1	0
1	0	0
1	1	⚡

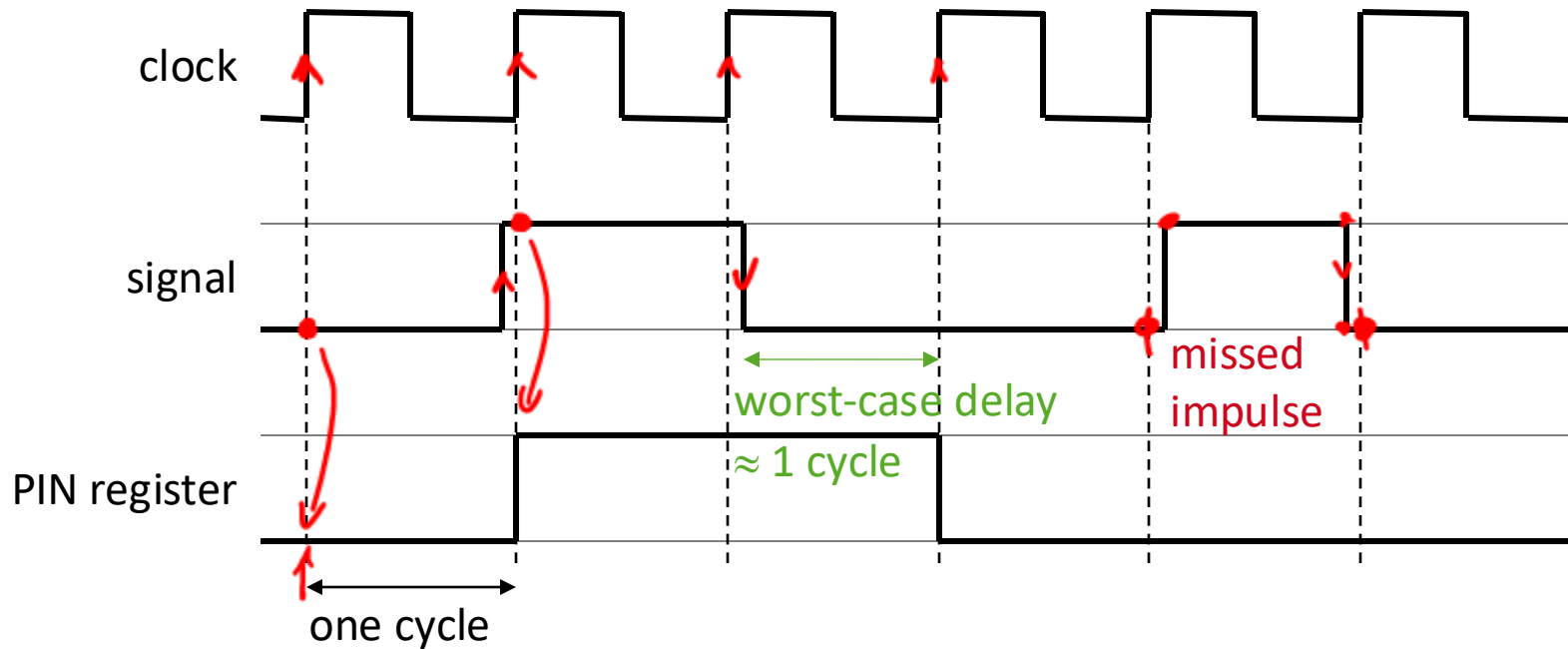
# Digital I/O Summary

DDR	PORT	I	II	III	IV	V
0	0	0	1	?	0	1
0	1	0	1	1	0	1
1	0	0		0	0	0
1	1		1	1	1	1

- ▶ I: direct connection to GND
- ▶ II: direct connection to VCC
- ▶ III: open switch / button
- ▶ IV: resistor connected to GND
- ▶ V: resistor connected to VCC

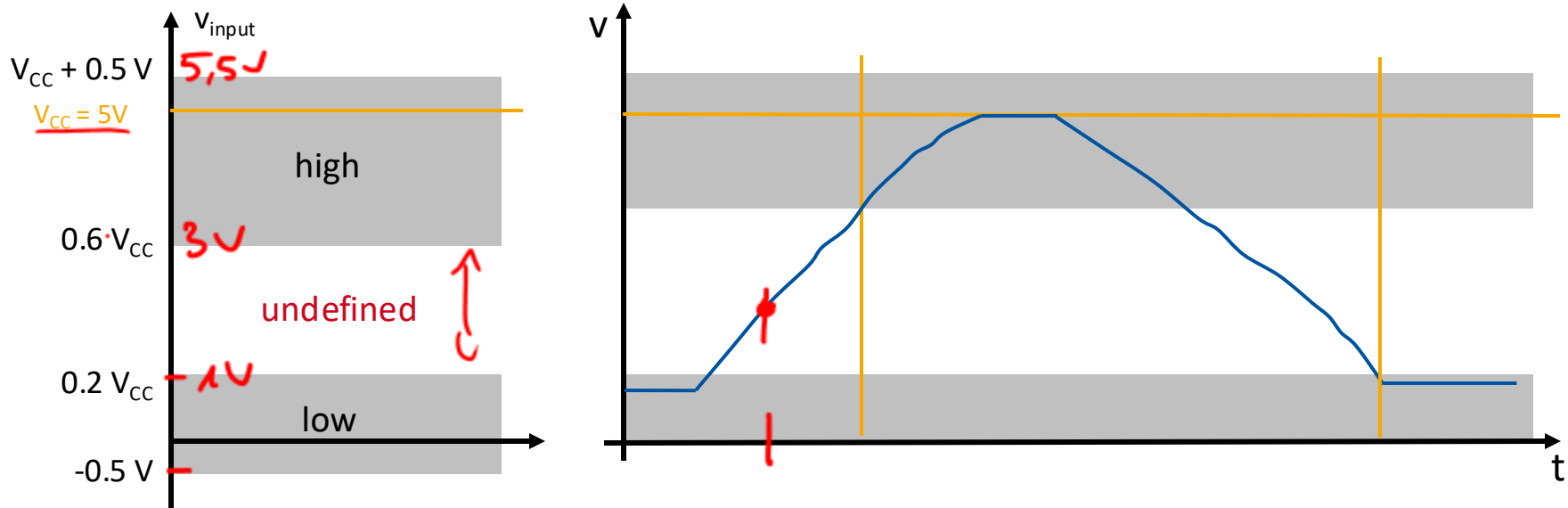
# Digital input: Sampling

- ▶ Sampling with every clock cycle causes a worst-case delay of  $\sim 1$  clock cycle.
- ▶ Impulses shorter than a clock cycle may be undetected.



# Digital input: Sampling (2)

- Problem: Signal does not always have a well-defined level.

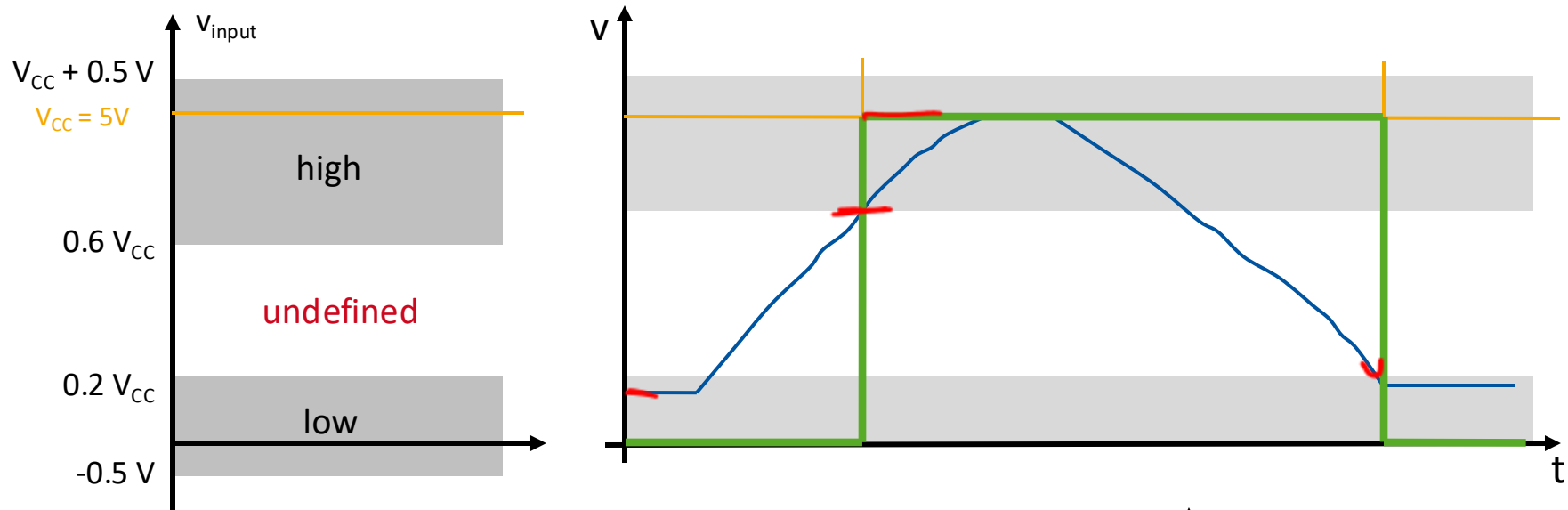


- Remedy: Schmitt trigger

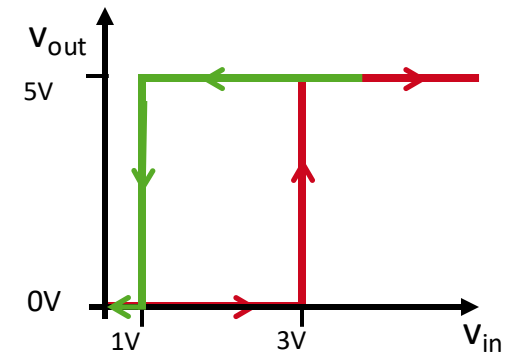
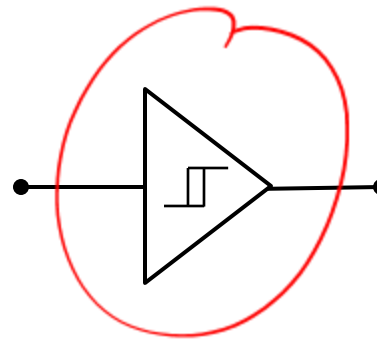


# Digital input: Sampling (2) Schmitt Trigger

- Problem: Signal does not always have a well-defined level.

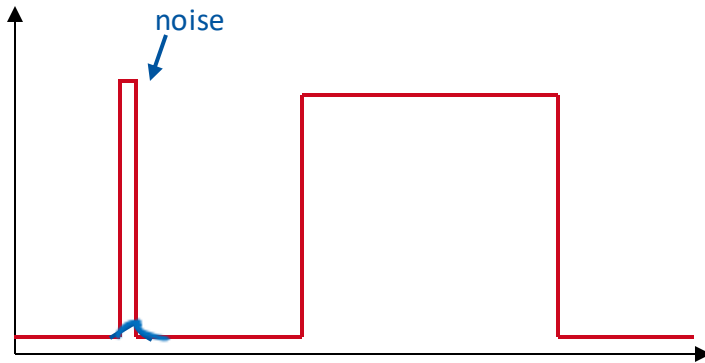


- Remedy: Schmitt trigger

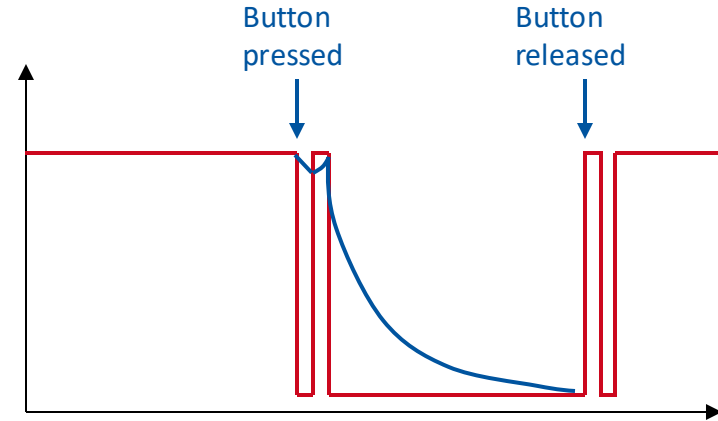


# Digital input: Sampling (3)

## Noisy signals

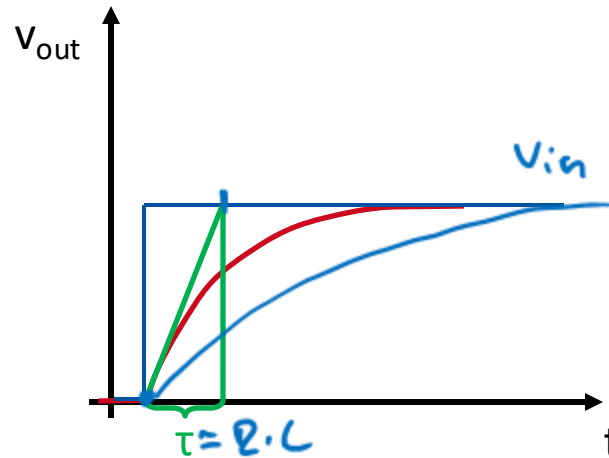
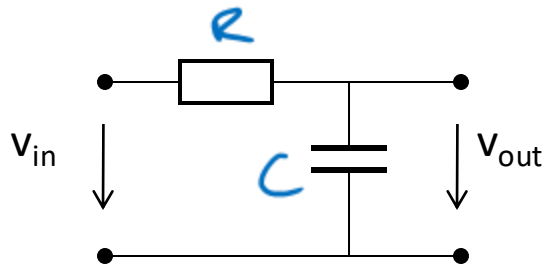


## Bouncing



## Solutions by Hardware:

- a) Low pass filter

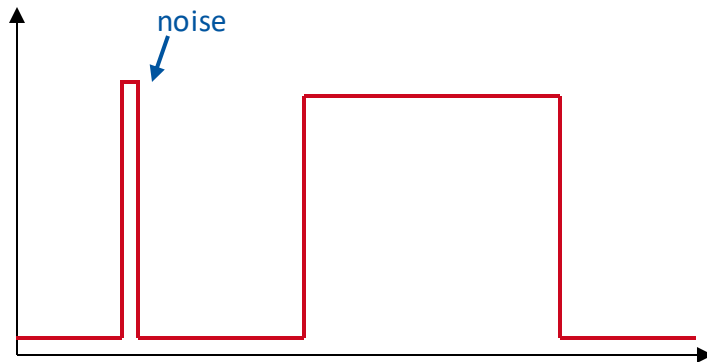


$$V_{out}(t) = K(1 - e^{-\frac{t}{\tau}})$$

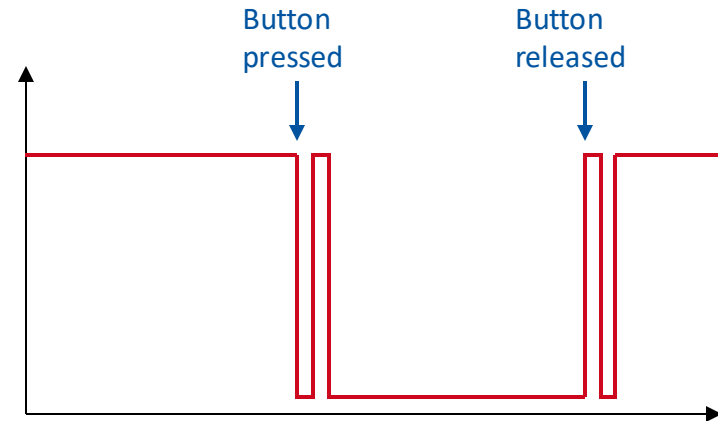
$\tau = RC$

# Digital input: Sampling (3)

## Noisy signals



## Bouncing



### ► Solutions:

### ► By Hardware:

- Low pass filter
- Built in noise cancelation

### ► By Software:

- Read Signal twice or more.



# Content

1. Basics
2. Structure/elements
3. Digital I/O
4. Interrupts
5. Timers/Counters
6. Analog I/O



# Why Interrupts?

---

- ▶ Microcontrollers have to react to events (internal ↔ external)
- ▶ How to ensure proper and timely reaction?

## 1. Polling

- Periodically check for event
- Disadvantages:
  - Waste of CPU time if the event occurs infrequently
  - Polling sequence has to fit in the rest of the code (hard to modify or extend)

## 2. Interrupts (IRs)

- MCU polls the signal and interrupts the main program if a state change is detected.
- MCU calls an interrupt service routine (ISR) which handles the event

# Interrupt Control

- ▶ To use Interrupts they have to be activated by modifying the according registers.
- ▶ Usually there is a
  - global bit for all interrupts (global interrupt enable) and
  - an individual bit for each interrupt (<name> interrupt enable).
- ▶ ATmega family: Mapping of interrupts on the according ISR is done by an Interrupt Vector Table. Example:

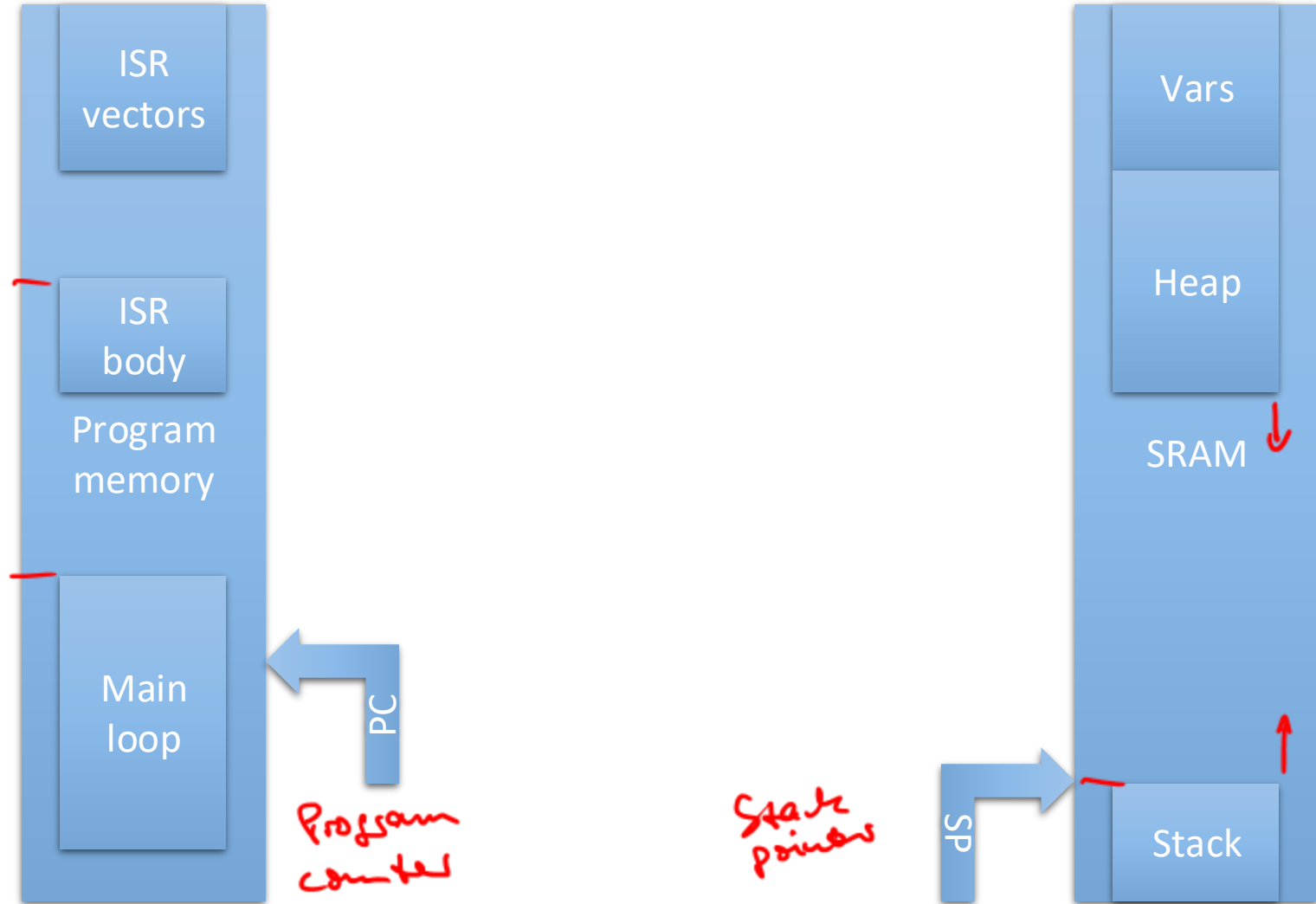
Vector No.	Source	Prg. Addr.	MNEMONICS
1	Reset	\$0000	JMP \$0312
2	→ External Interrupt 0	\$0002	JMP \$1302
3	External Interrupt 1	\$0004	JMP \$0004
...	→ ... Ext. Int. 2	...	...

- ▶ A **jump instruction** to the according ISR body must be placed at each address
- ▶ Empty vectors should point to an **infinite loop** (trap)

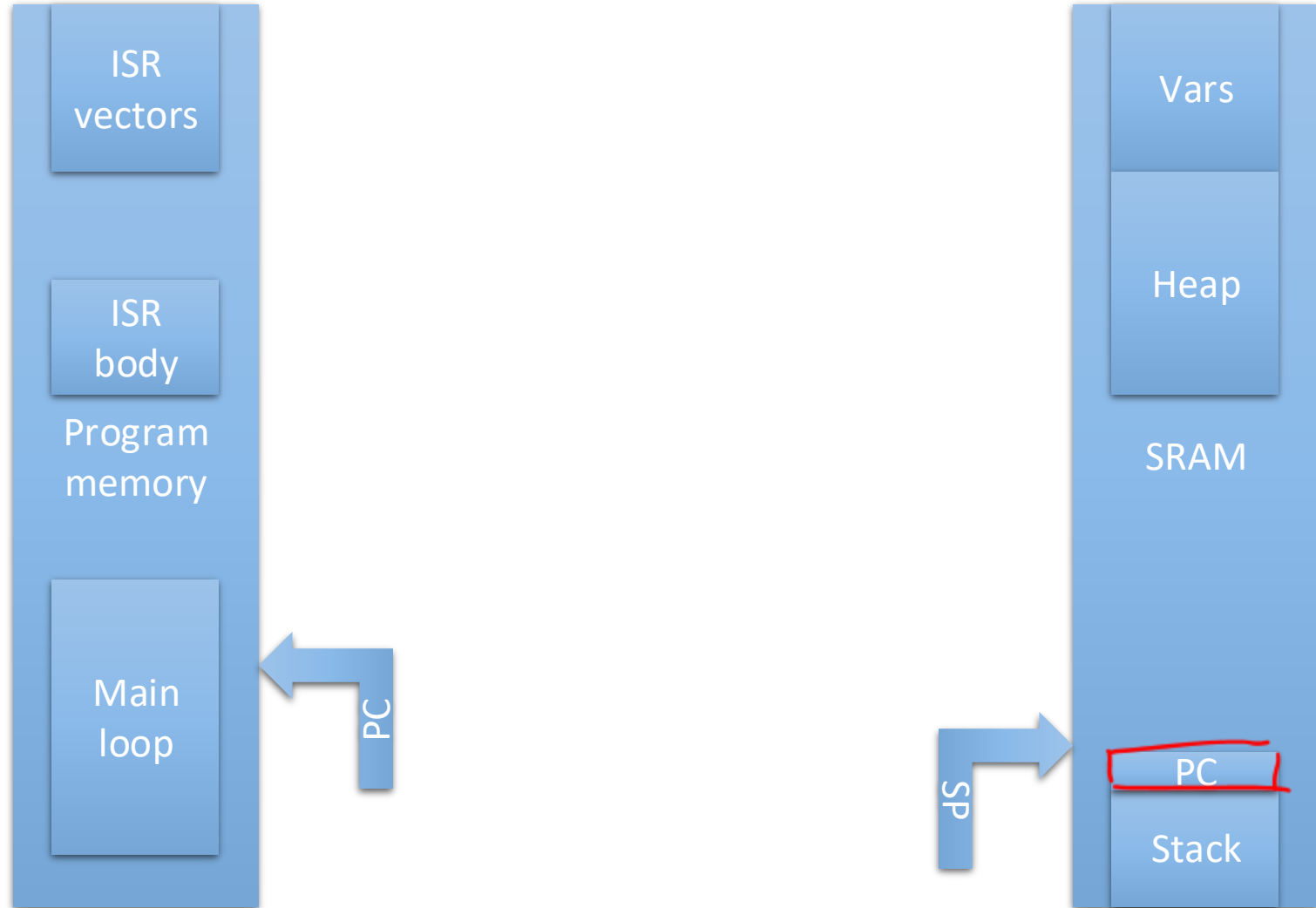
# Interrupt Handling

- ▶ MCU monitors certain events (e.g., timer overflow)
- ▶ When an event takes place, a flag is set by hardware
- ▶ MCU calls ISR, if three bits are set:
  - Global interrupt enable bit (I bit)
  - Individual interrupt enable bit (e.g., timer overflow enable bit)
  - Interrupt flag (e.g., timer overflow flag)
- ▶ Conflicts are resolved by priorities
  - Static priorities (e.g., ATMEL ATmega family)
  - ( Dynamic priorities (e.g., Renesas R8C family) )

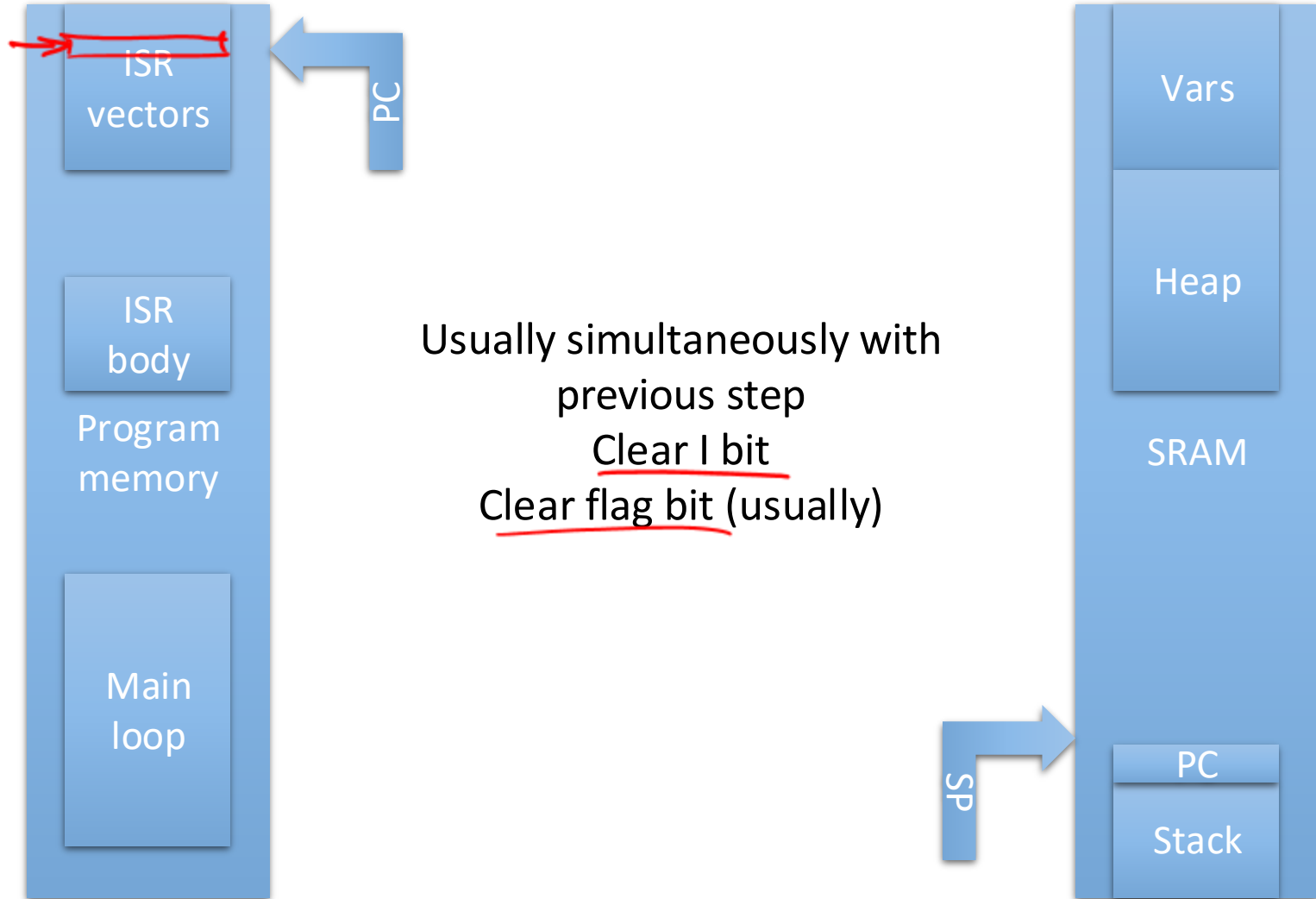
# Interrupt Service Routine - Before Call



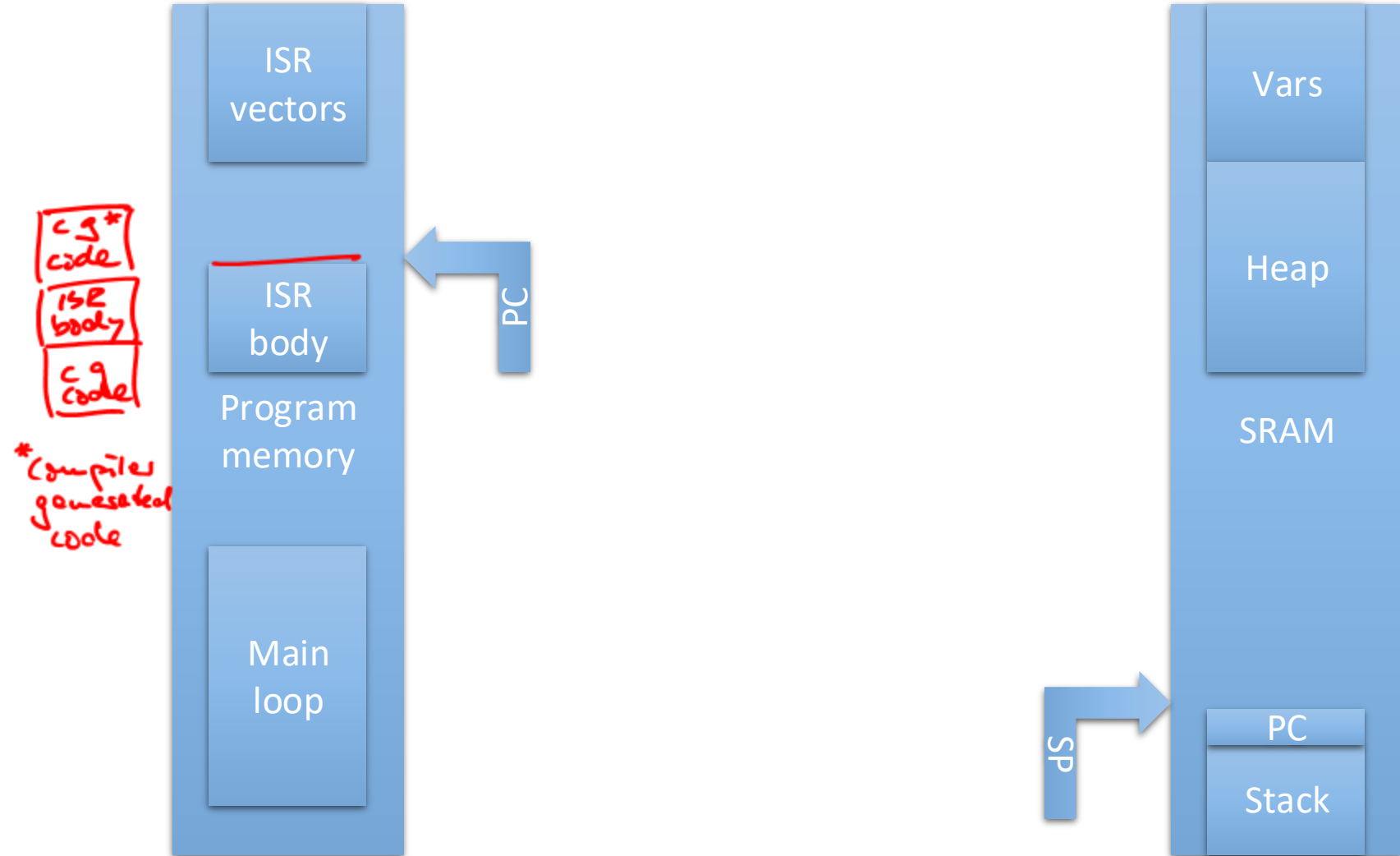
# Interrupt Service Routine - Save Return Address



# Interrupt Service Routine - Jump to Interrupt Vector

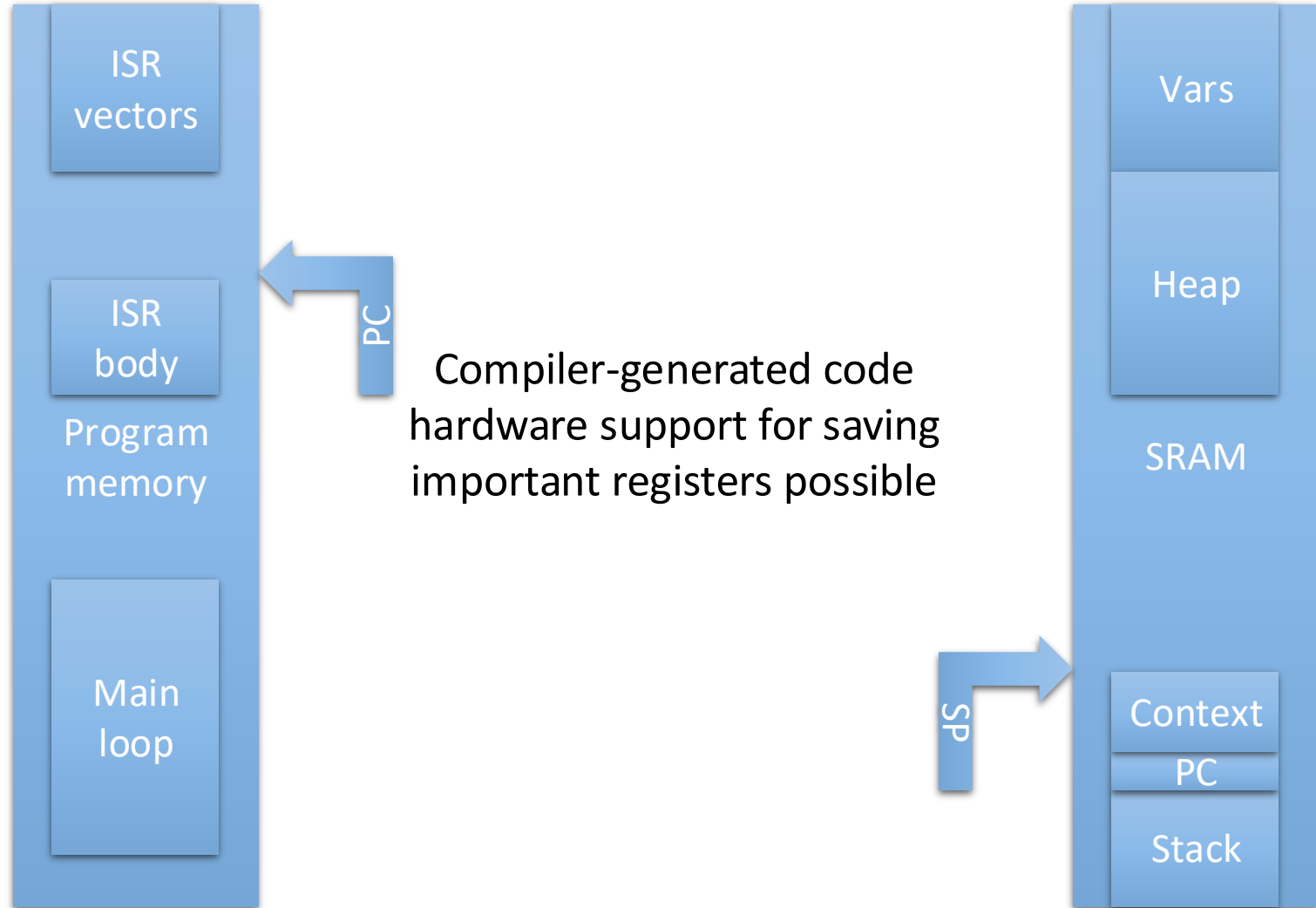


# Interrupt Service Routine - Jump to ISR body

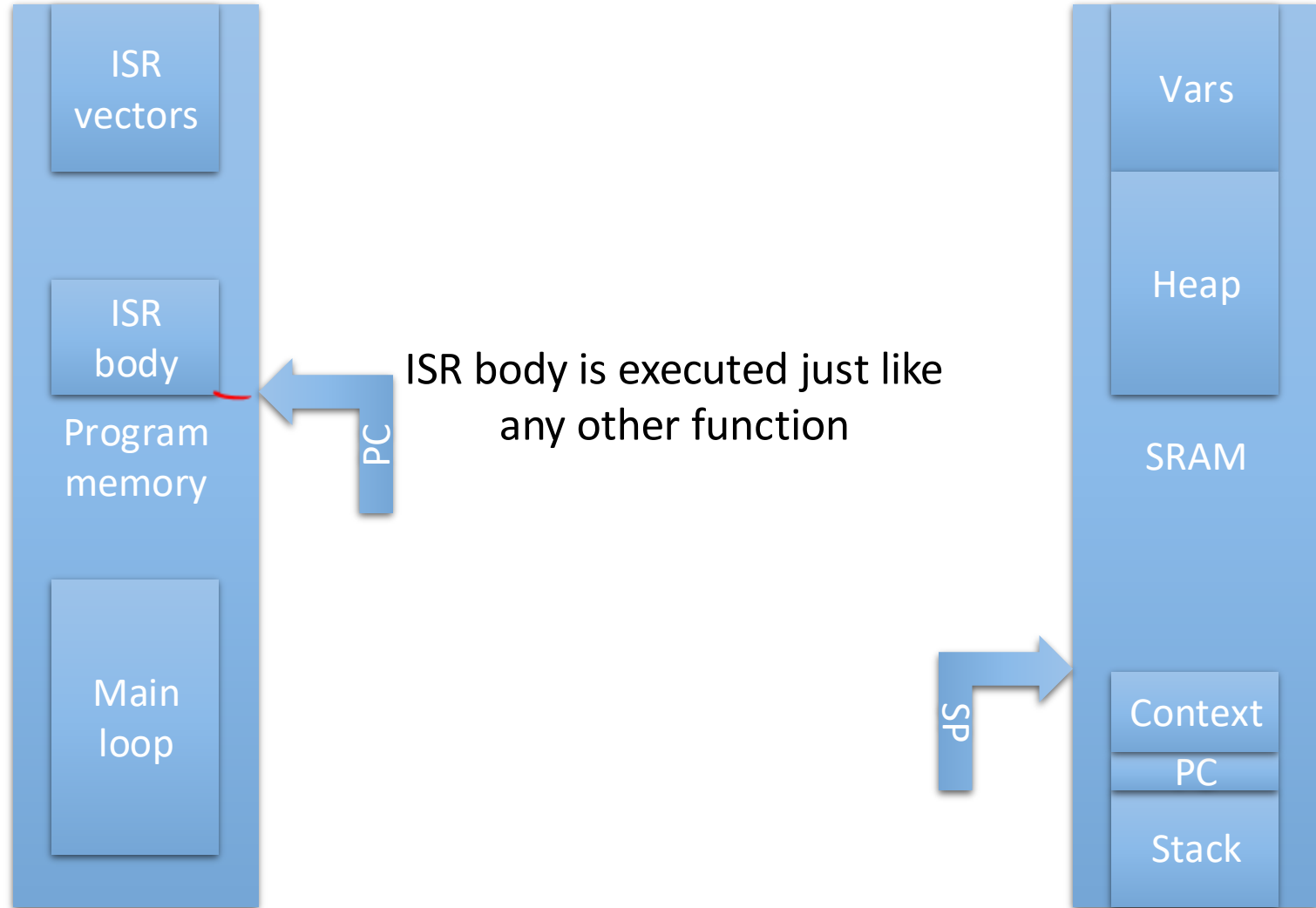




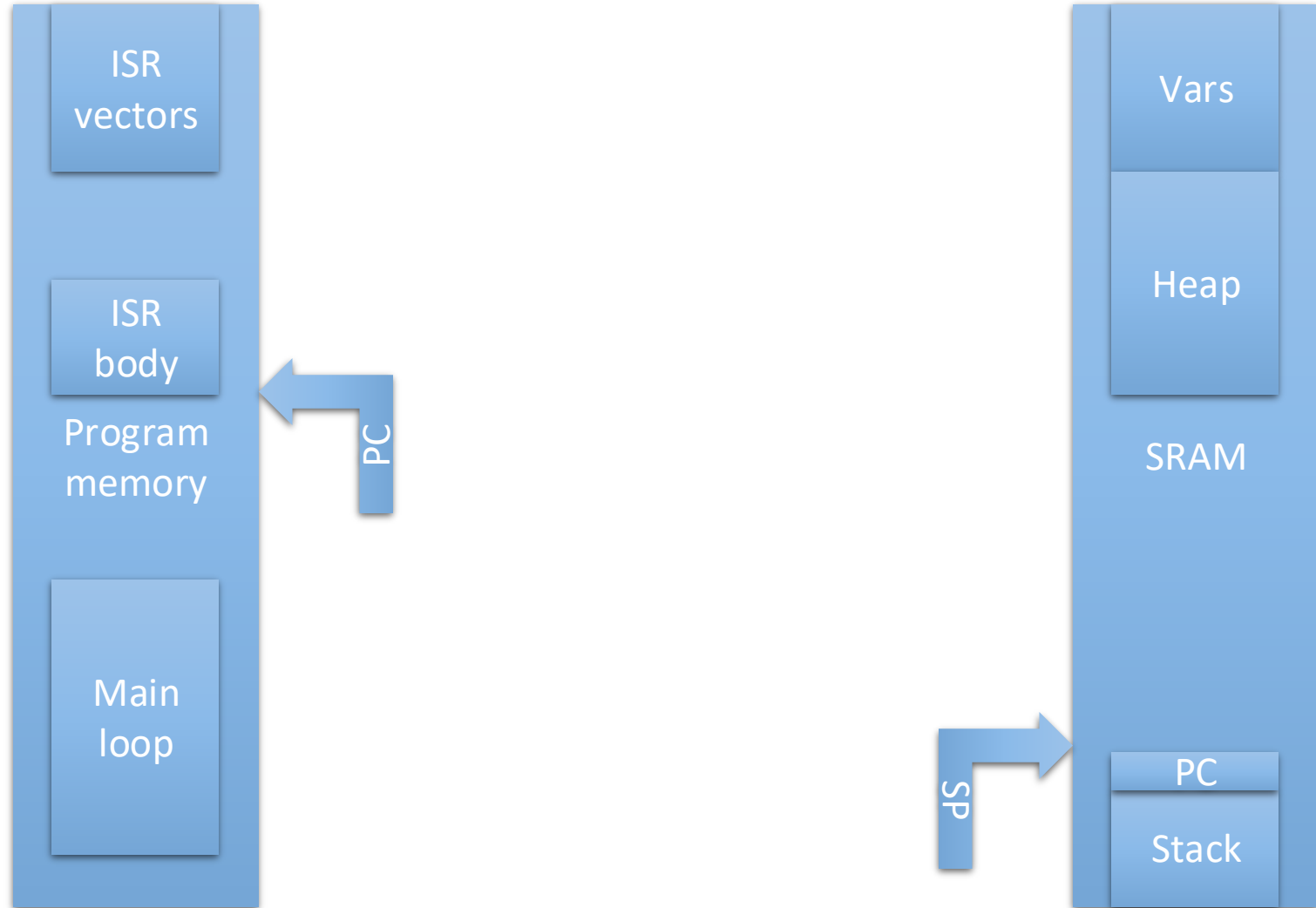
# Interrupt Service Routine - Save context



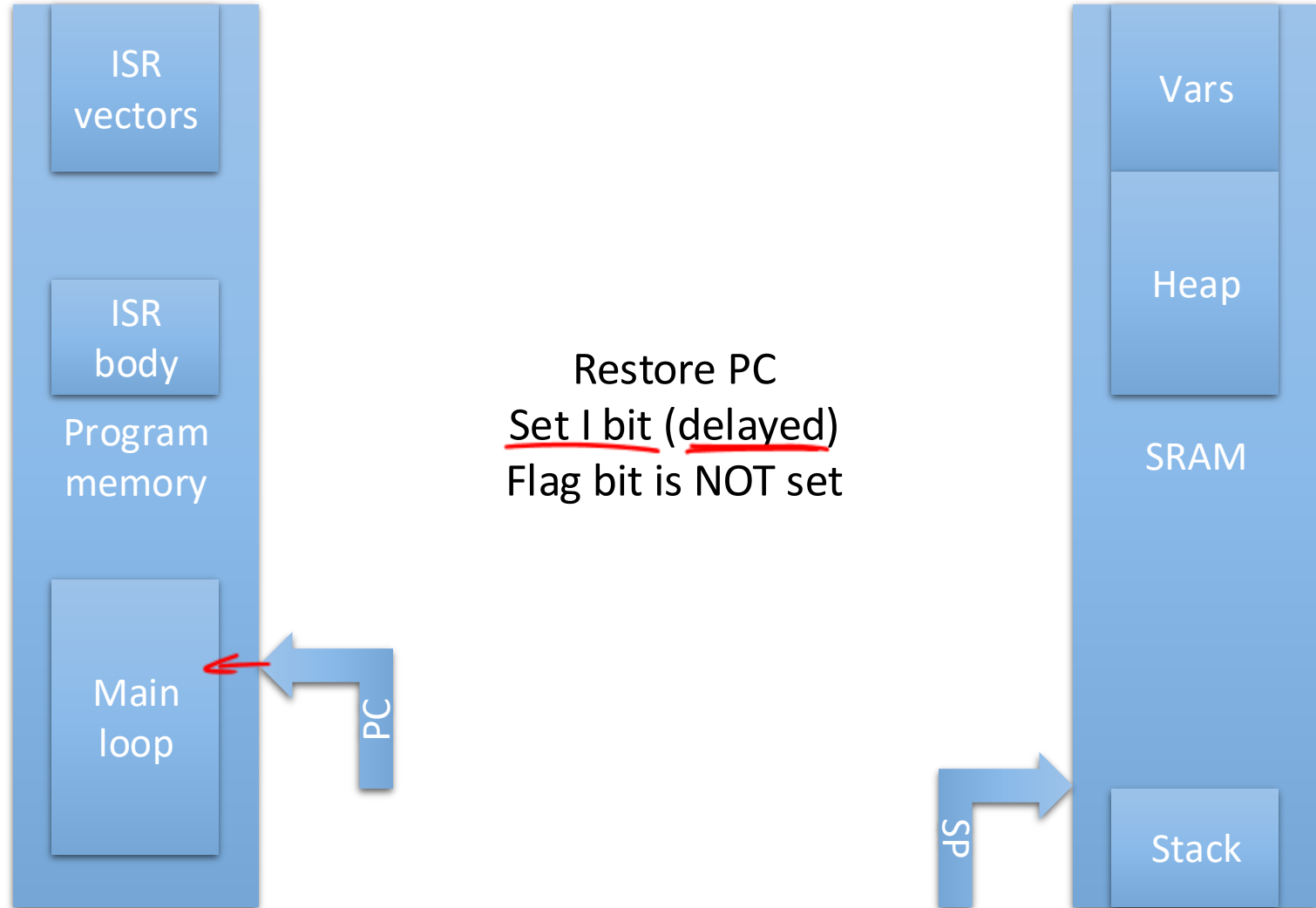
# Interrupt Service Routine - Execute ISR body



# Interrupt Service Routine - Restore context



# Interrupt Service Routine - Return to main program



# Interrupt Service Routine - Summary

- ▶ ISR is triggered by event
  - Save return address (PC) to stack
  - Clear global interrupt enable bit (I bit)
  - Clear interrupt flag bit (usually)
  - Jump to corresponding interrupt vector table entry (interrupt vector)
- ▶ Execute jump instruction at interrupt vector
- ▶ Save additional context (anything not automatically saved by hardware)
- ▶ Execute ISR body
- ▶ Restore context
- ▶ Leave ISR by assembly instruction RETI (*return from interrupt*)
  - Return to PC popped from stack
  - Set global interrupt enable bit (maybe delayed)

# Interrupt vs. Polling

▶ Hard to decide, but the following should give some hints:

▶ **Interrupts** should be favored if

- Event occurs infrequently
- Long intervals between two events
- The exact time of the state change is important
- Short impulses, polling might miss them

➔ Nothing else to do in main, could enter sleep mode

▶ **Polling** might be a better choice if

- No precise timing is necessary
- The state is important
- Impulses are long
- The signal is noisy (Interrupts would be triggered very often)



# Interrupts – final remarks

---

- ▶ A long ISR delays the main program for a long time!
  - Sometimes it is useful to move some of the ISR code to the main routine.
- ▶ If more than one IR is used the side effects have to be considered.
  - The execution of an ISR delays the reaction on all other IRs.
  - The order of IR events may have influence on the behavior.
  - Results in complex timing (cf. lectures on real time)
- ▶ If IRs are enabled the main program can be interrupted everywhere.
  - For some (short!) parts of the program it might be necessary to disable IRs.
  - Race conditions (e.g., increment from 255 to 256)

# Content

1. Basics
2. Structure/elements
3. Digital I/O
4. Interrupts
5. Timers/Counters
6. Analog I/O



# Timer/Counter

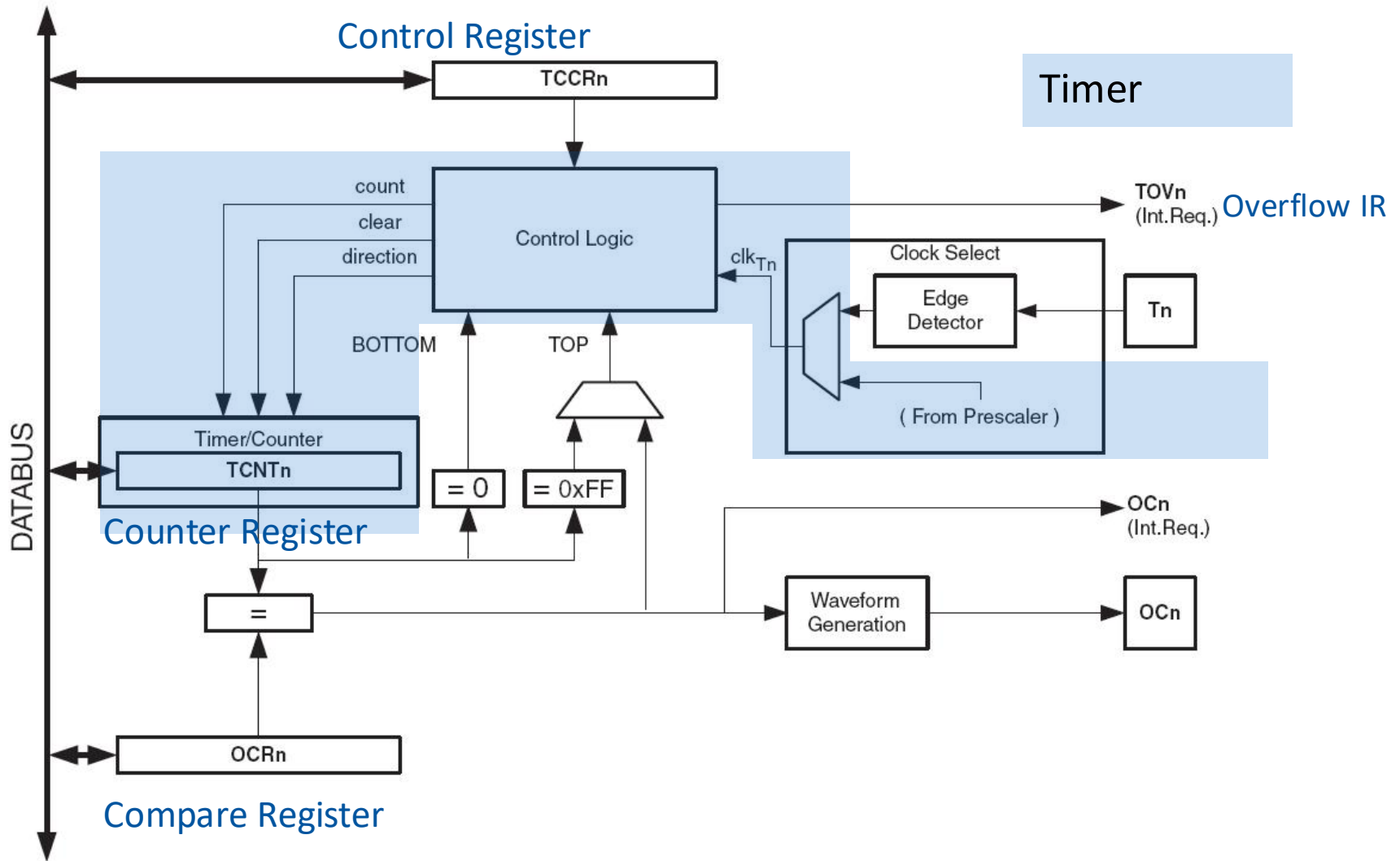
---

- ▶ On chip peripherals (dedicated hardware)
- ▶ Counter:
  - counts external events
  - e.g. number of rising edges at PINB2
- ▶ Timer:
  - counts clock cycles (with or without prescaler)
  - Each timer is basically a counter
- ▶ Most controllers provide one or more timer/counter with 8 and/or 16 bit resolution.

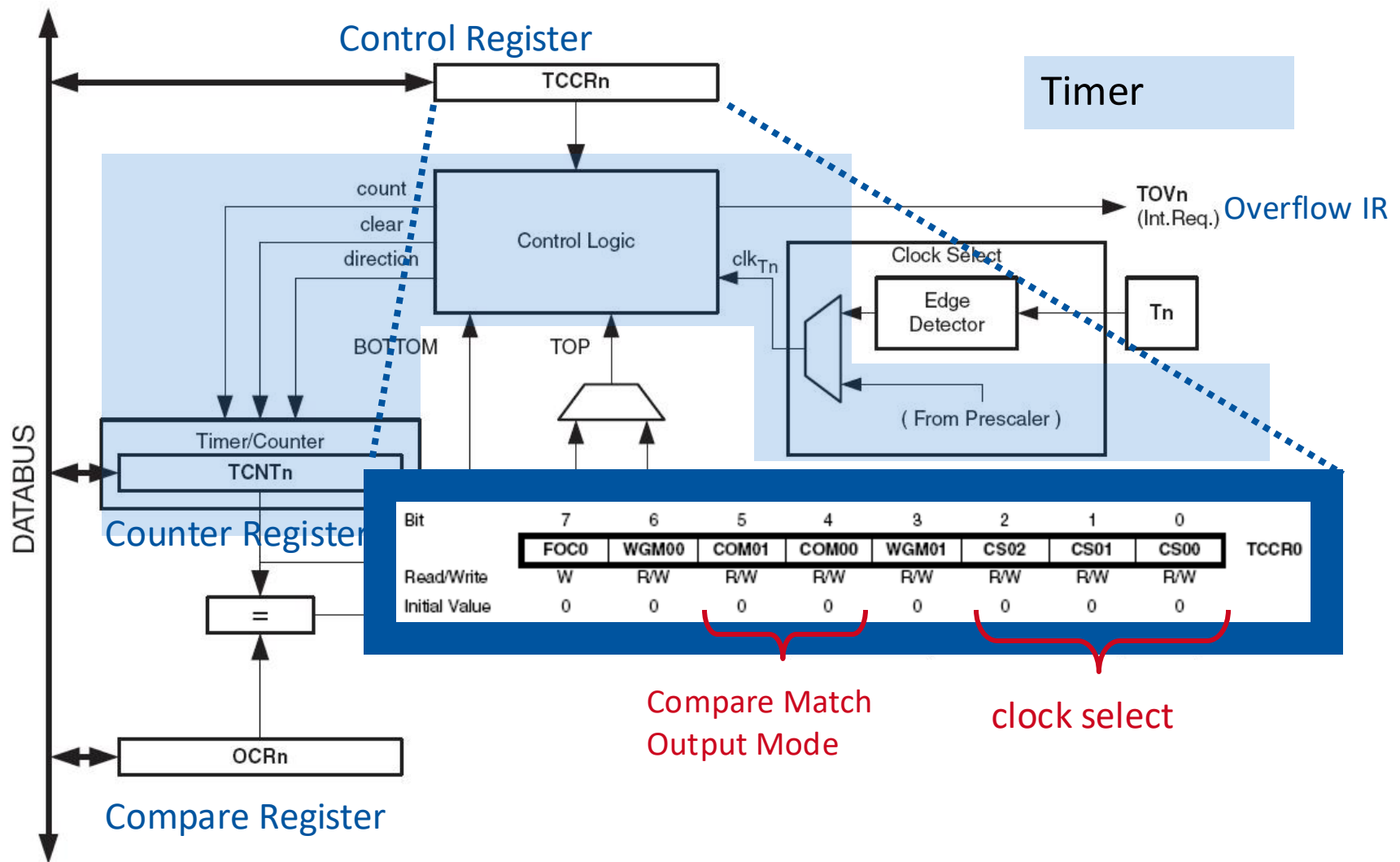
# Timer/Counter

- ▶ Each Timer/Counter unit is based on a **counter register**, which can be incremented or decremented.
- ▶ Important for the behavior of the Timer/Counter is (are) the according **control register(s)**
  - mode of operation
  - which prescaler to use (timer)
  - start & stop counting
  - enable/disable interrupts
- ▶ Often there is also a **compare register**, which can be used to generate an interrupt if its content is equal to the counter register.

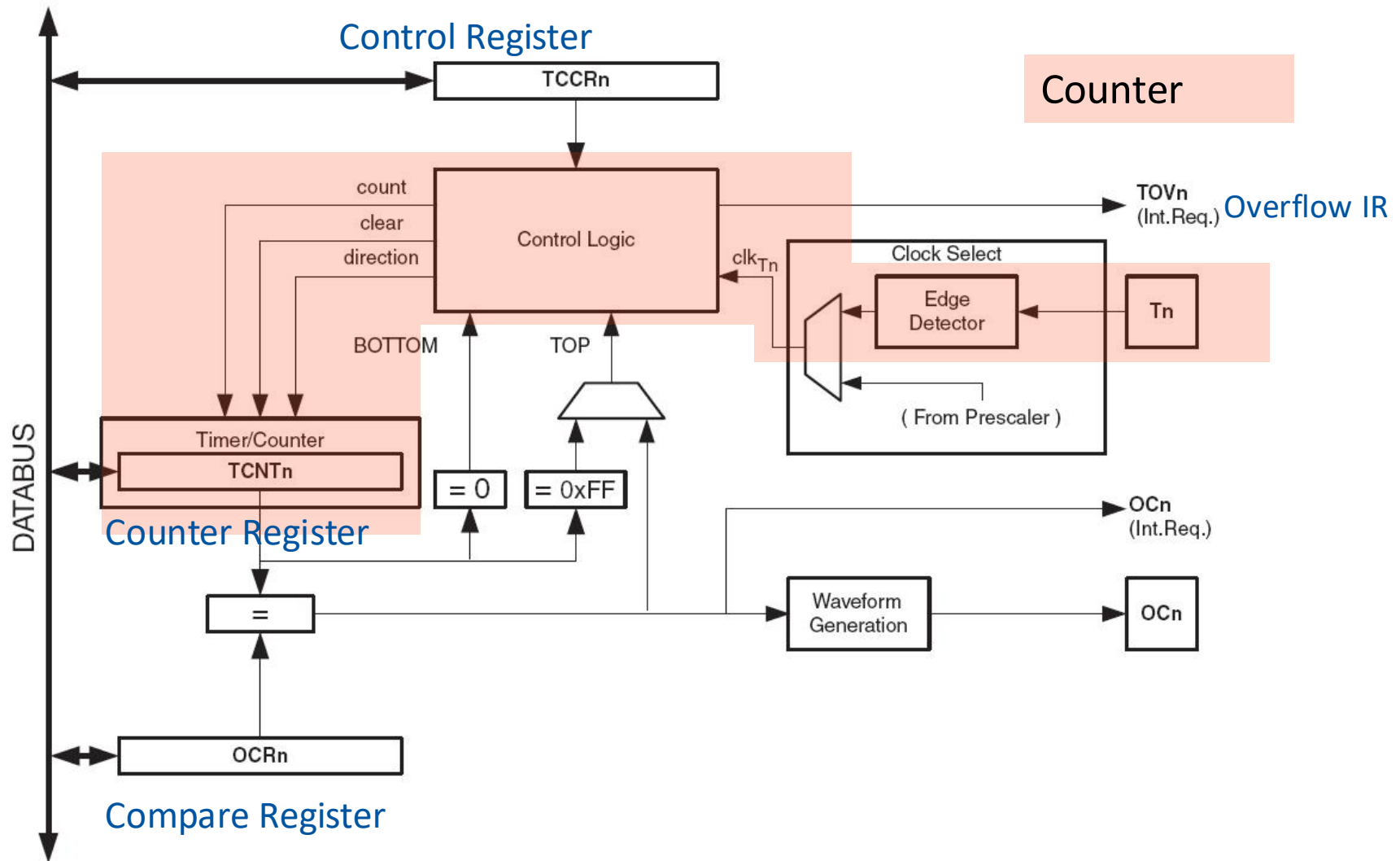
# Timer/Counter (ATmega16)



# Timer/Counter (ATmega16)



# Timer/Counter (ATmega16)



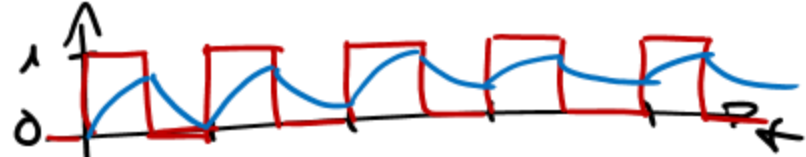
# Timer/Counter



- ▶ More than just counting events and measuring time.

- ▶ Other features

- Input capture
  - Used to timestamp (mostly external) events
  - Whenever the event occurs, the timer automatically copies its current count value to an input capture register

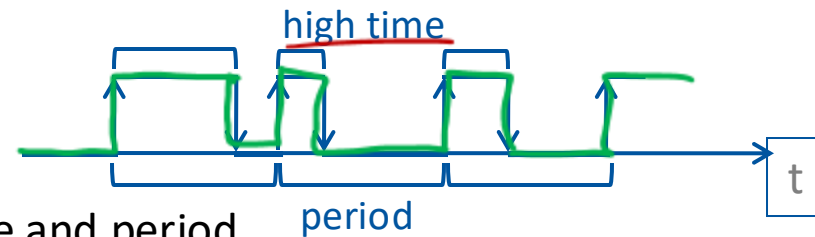


- Output compare
  - Used to generate signals
  - Whenever a certain timer value is reached, the output compare event is triggered (can automatically set or clear an output line).

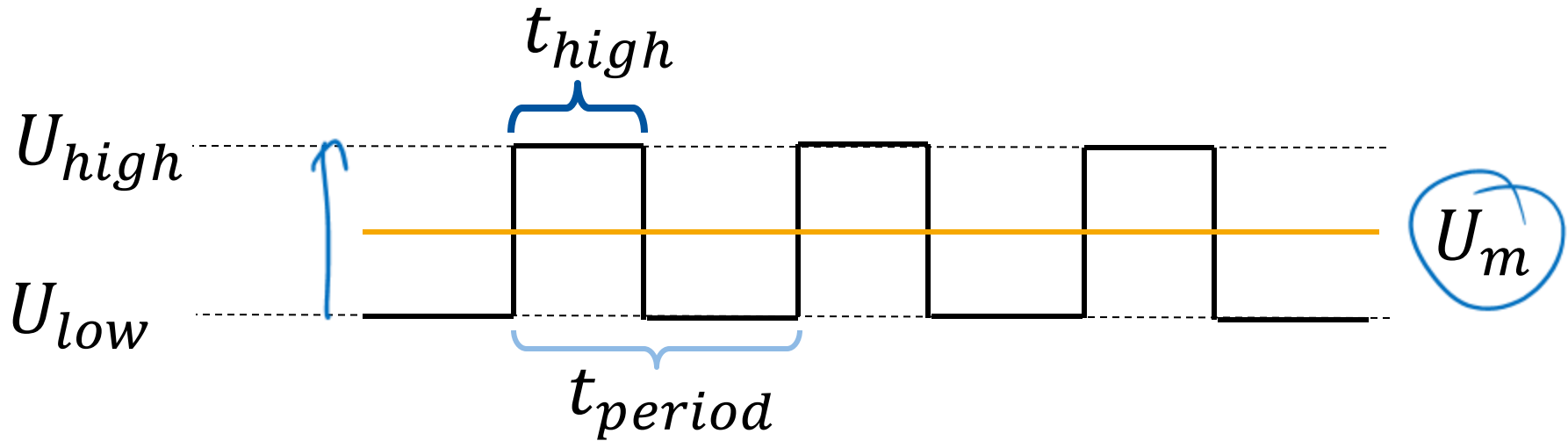


*Pulsweitenmoduliertes Signal*

- Pulse Width Modulation (PWM)
  - Special case of output compare
  - Timer generates a periodic digital output signal with configurable high-time and period.



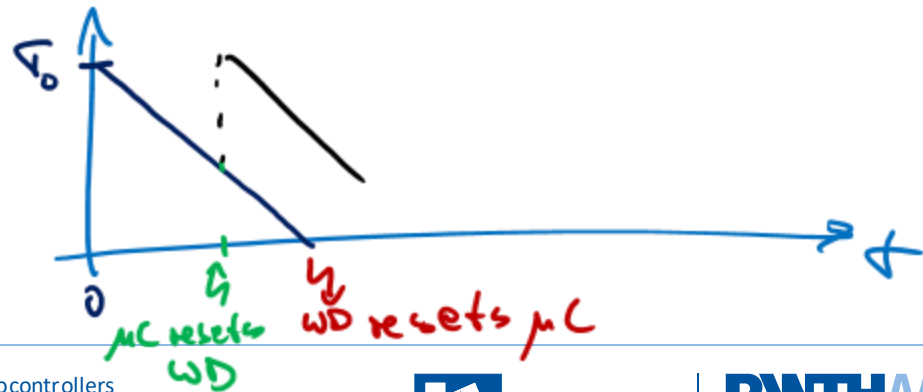
# Pulse Width Modulation (PWM)



$$U_m = U_{low} + (U_{high} - U_{low}) \cdot \frac{t_{high}}{t_{period}}$$

# Watchdog Timer (WD)

- ▶ Special Timer; used to monitor software execution
- ▶ If enabled it counts down and resets the controller as soon as the count value zero is reached.
- ▶ During SW execution the WD has to be reset to its initial value before it reaches zero. If this fails the WD resets the controller.
- ▶ Useful if the program execution hangs and a restart solves the problem.
- ▶ However the WD can also be the source of problems (see Pathfinder problem)





# More information...

---

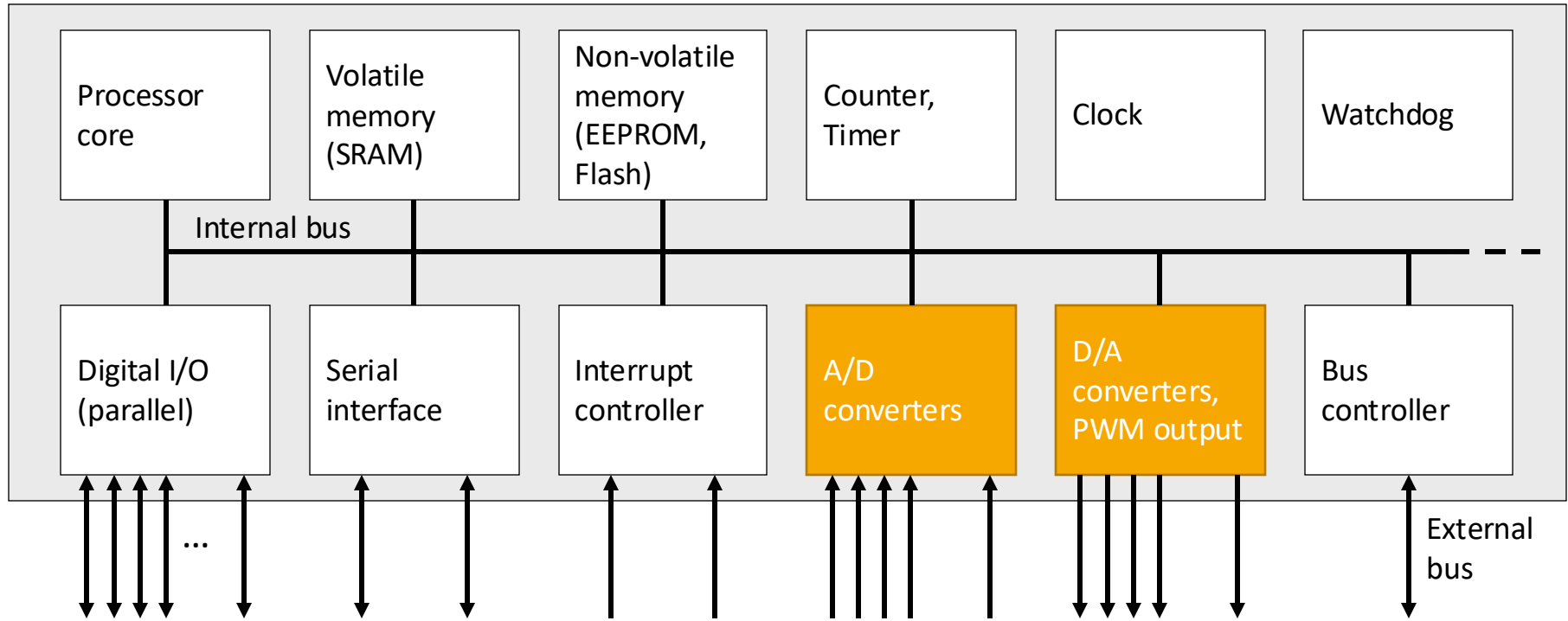
- ▶ ...about Interrupts and Timer / Counter can be found in any microcontroller data sheet like the one used in the exercise.

# Content

1. Basics
2. Structure/elements
3. Digital I/O
4. Interrupts
5. Timers/Counters
6. Analog I/O



# Reminder: Basic structure of a microcontroller - refined

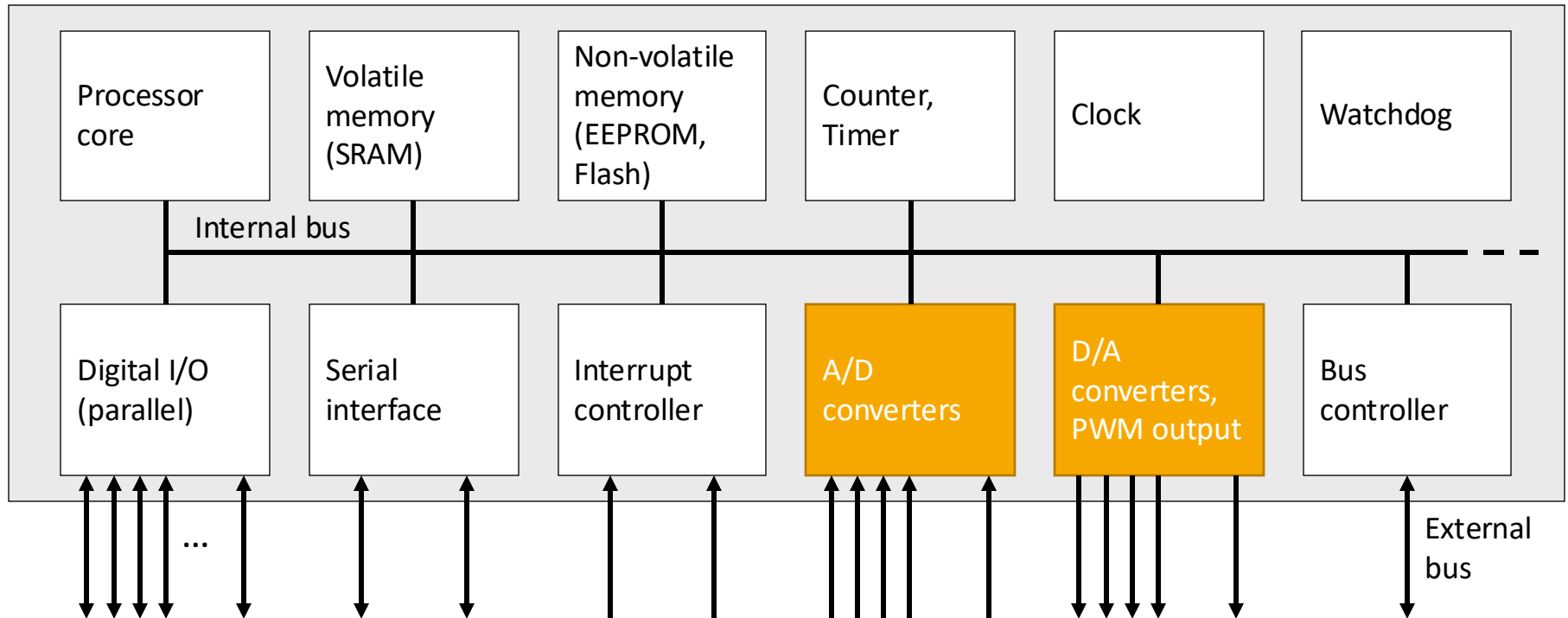


# Analog I/O Overview

---

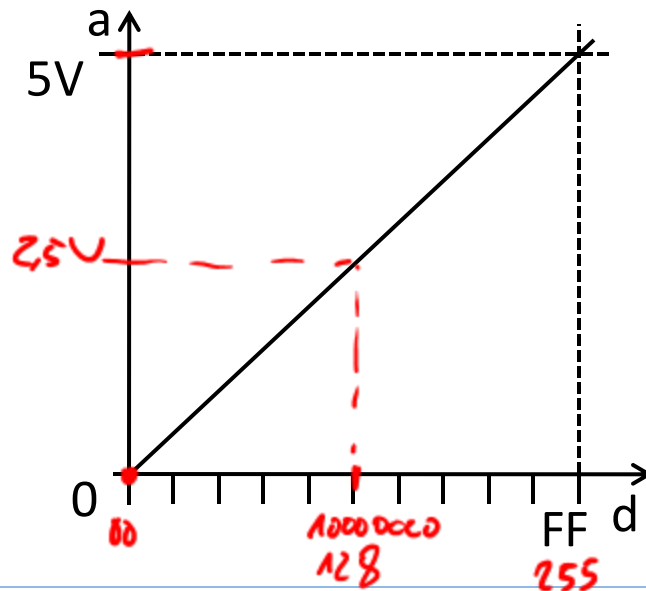
- ▶ Two directions: DAC and ADC
- ▶ DAC:
  - RC low-pass filter
  - Binary weighted resistor circuit
  - R-2R ladder
- ▶ ADC:
  - Simple: Analog Comparator
  - Flash Converter
  - Tracking Converter
  - Successive Approximation Converter
- ▶ Errors
- ▶ ATmega16 ADC

# Reminder: Basic structure of a microcontroller - refined



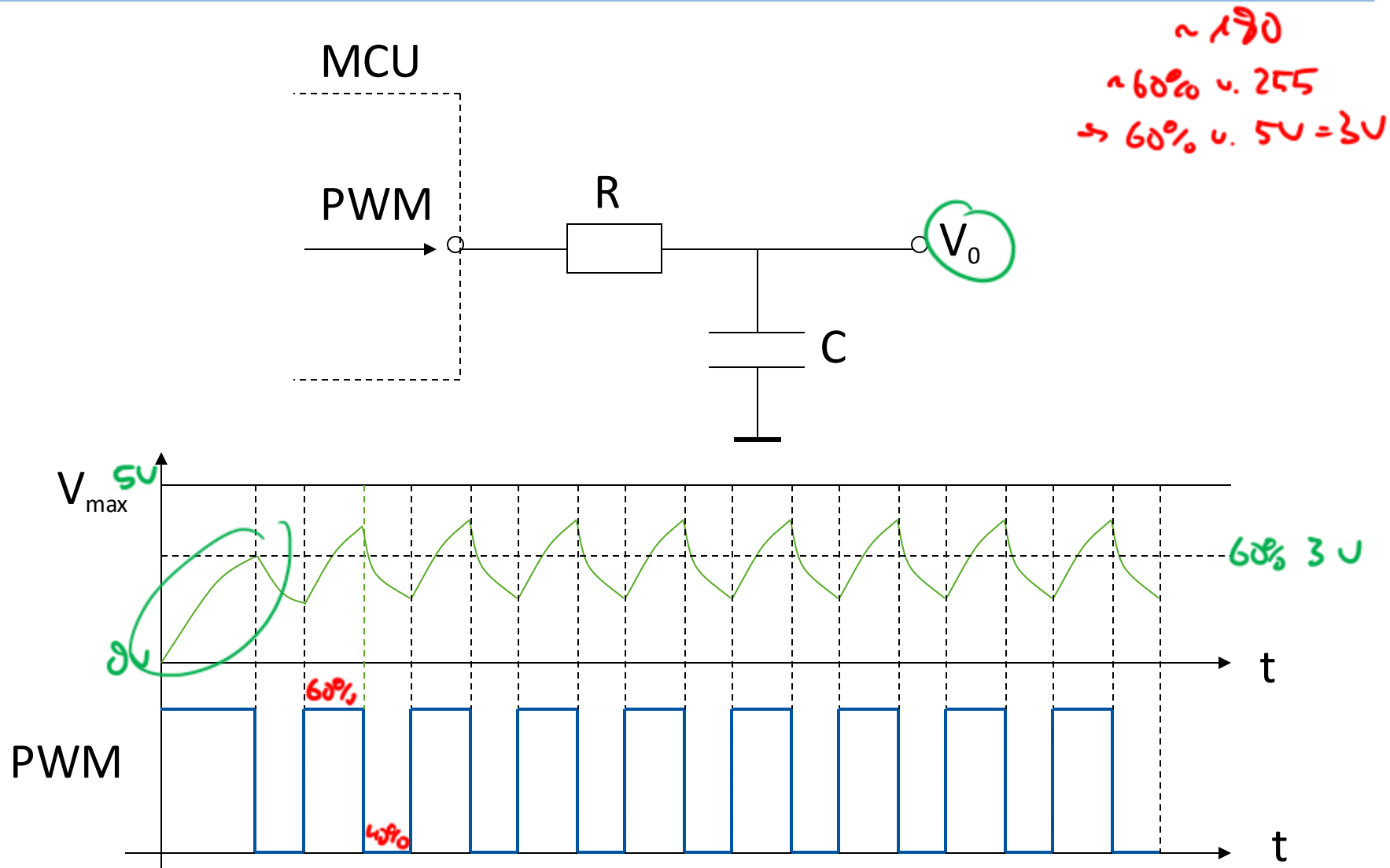
# Digital to Analog Conversion

- ▶ Transform a digital value  $B=(b_{r-1} \dots b_0)$
- ▶ Range of values:  $[0, 2^r-1]$
- ▶ Aim: proportional analog value  $V_0$



$$a = \text{DAC}(d)$$

# RC low-pass Filter Function



# RC low-pass Filter Properties

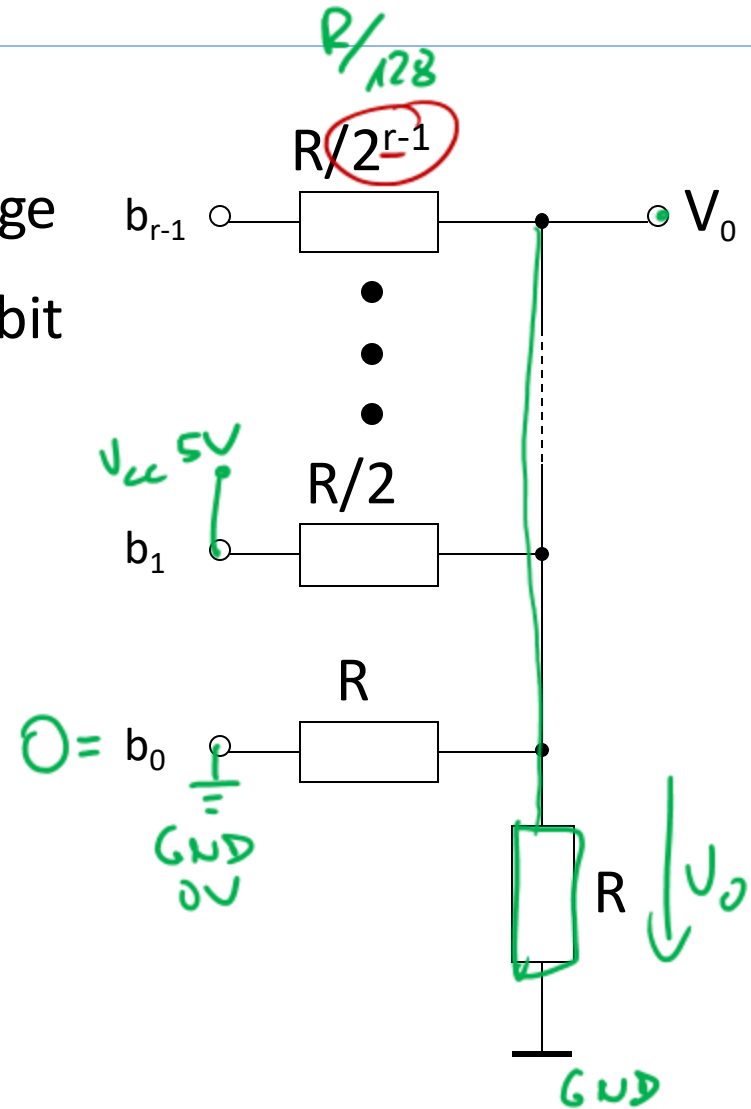
---

- ▶ Simple and cheap
- ▶ 1-PIN
- ▶ Uses PWM
- ▶ Proportional to PWM (high-time/period)
- ▶ Low quality
- ▶ Initially delayed (by charging time)



# Binary Weighted Resistor Circuit

- ▶ r-bit input
- ▶ Each bit adds to analog output voltage
- ▶ Voltage added based on position of bit
- ▶ Problems: high precision resistors

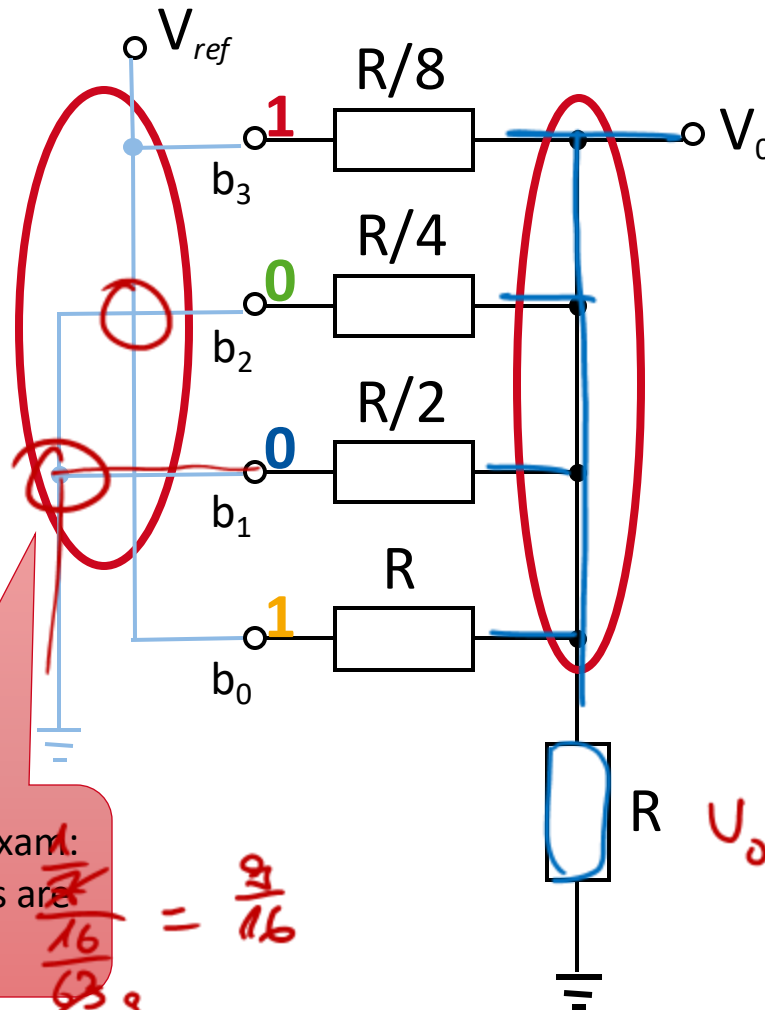


$$V_0 = V_{ref} \cdot \sum_{i=1}^r \frac{1}{2^i} b_{r-i}$$

# Binary Weighted Resistor Circuit

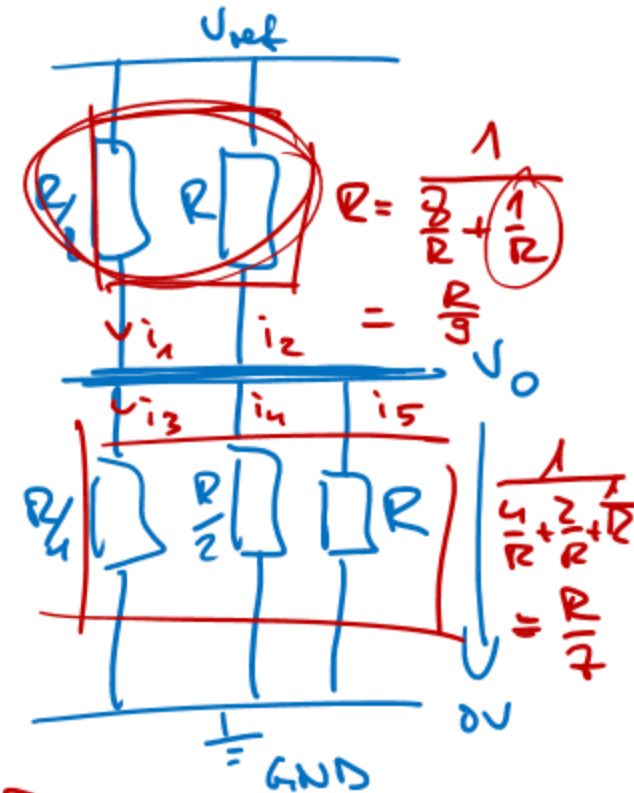
► Example:

1001  
b<sub>3</sub> b<sub>2</sub> b<sub>1</sub> b<sub>0</sub>



Remember for the exam:  
The connection dots are  
important!

$$\frac{1}{\frac{1}{16} + \frac{1}{16} + \frac{1}{16} + \frac{1}{16}} = \frac{1}{\frac{4}{16}} = \frac{16}{4} = 4$$



$$V_0 = \frac{R/8}{R/9 + 1/4R} \cdot V_{ref}$$

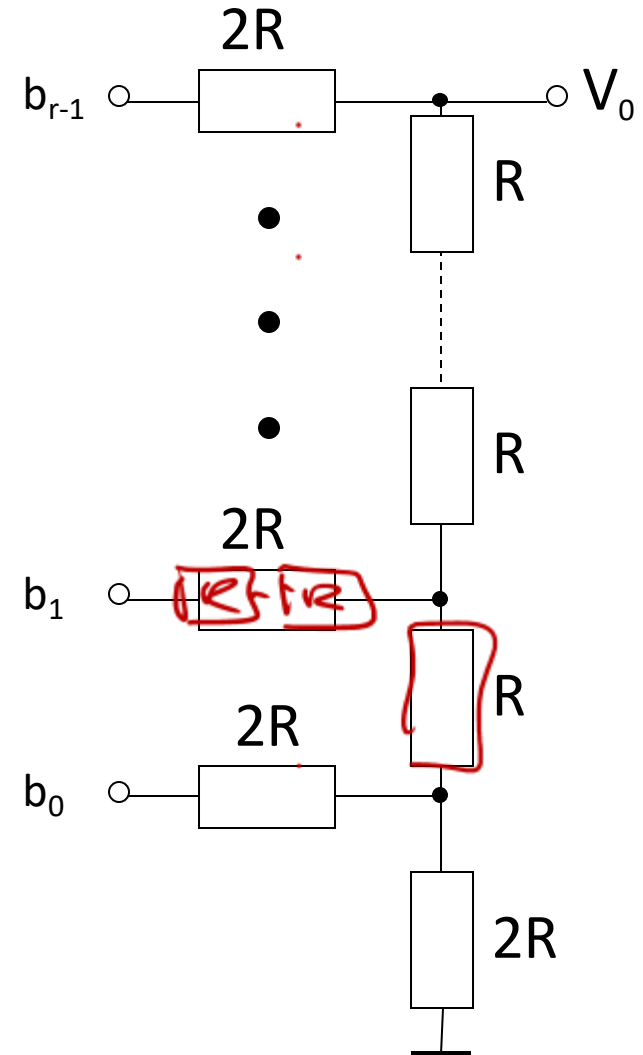
$$V_0 = 9/16 \cdot V_{ref}$$

# R-2R Ladder

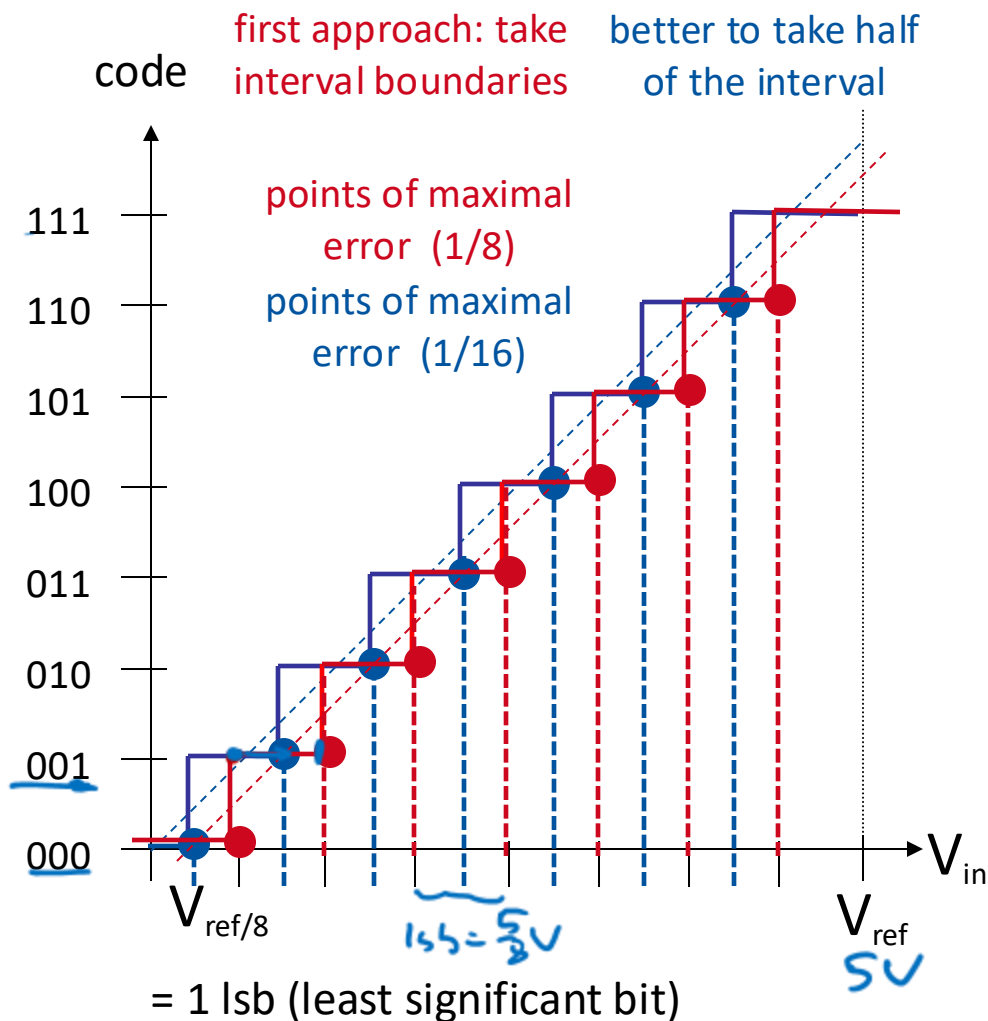
- ▶ Two types of resistors
- ▶ Can even be done with one type:

$$\text{---} \boxed{R} \text{---} \boxed{R} \text{---} = \text{---} \boxed{2R} \text{---}$$

- ▶ Simpler than solution before
  - Much more precise
  - Less cost
- ▶ Many resistors needed
- ▶ Same formula
  - Kirchhoff's circuit laws



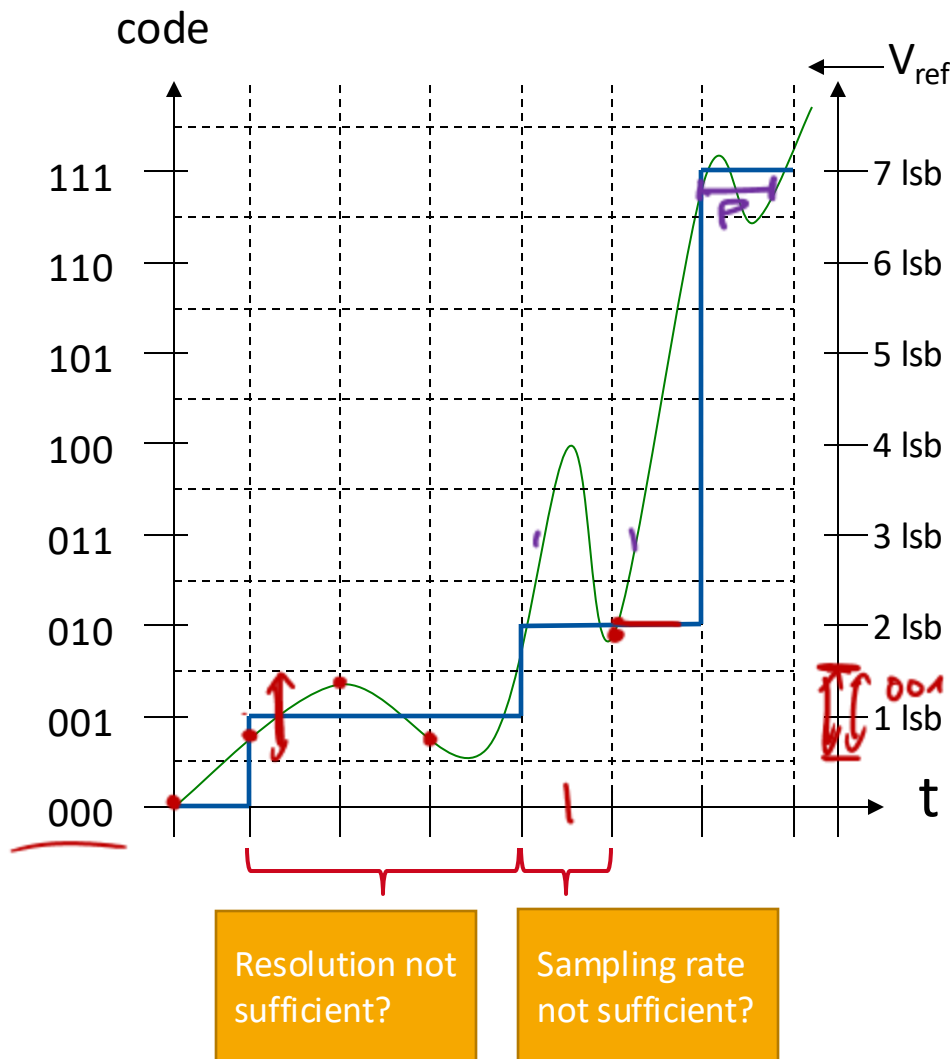
# Analog to Digital Conversion



- ▶ Transfer function
- ▶ Resolution:  $r$  bits
- ▶  $\rightarrow 2^r$  classes
- ▶  $lsb$  is smallest voltage difference  $V_{ref}/2^r$
- ▶  $lsb$  is used as unit
- ▶ Digitization error of  $0.5 \text{ lsb}$
- ▶ Class width asymmetry

$$rel. error = \frac{abs. error}{curr. value}$$

# Example with Inaccuracies



## Information loss

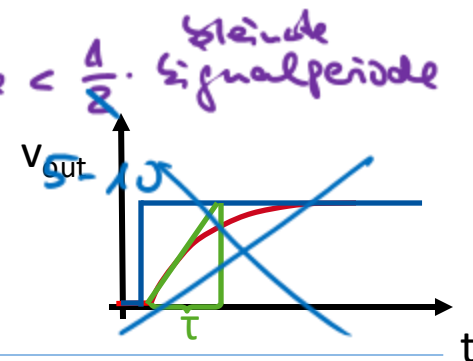
## Work around y-axis

- Improve granularity
- E.g. reduce  $V_{ref}$  or
- increase  $r$

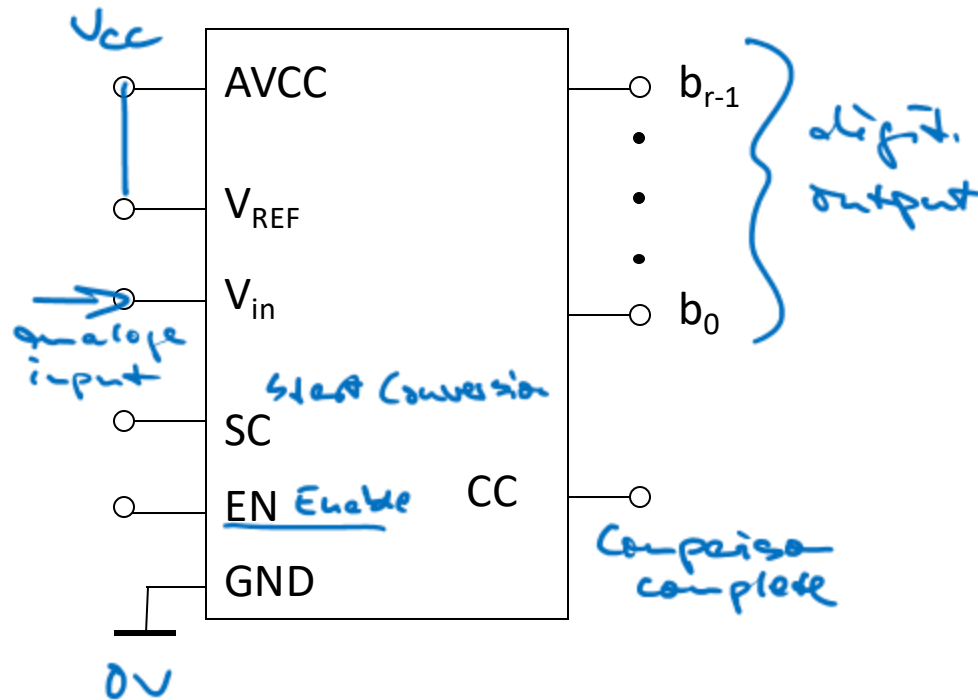
## Work around x-axis: conversion time

## Shannon's sampling theorem

Based upon the time constant of the signal



# ADC as a Black Box

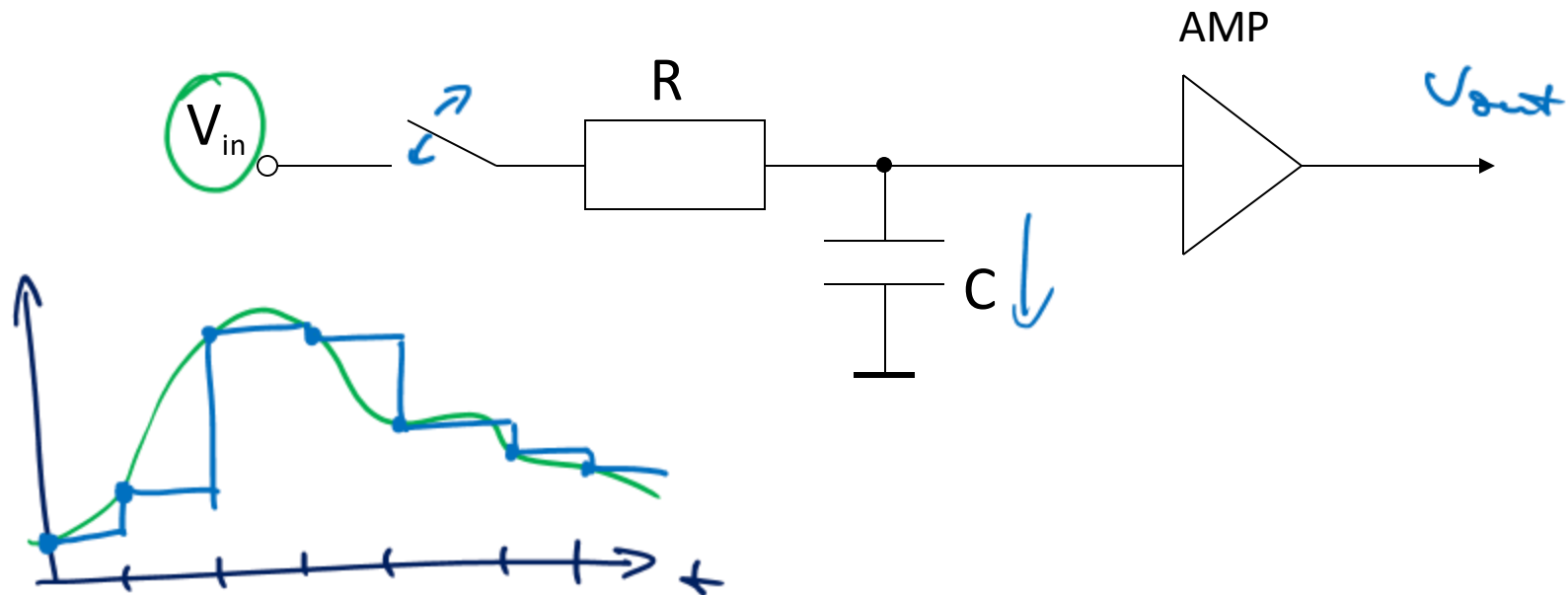


- ▶ **AVCC, GND**: Power supply
- ▶  **$V_{ref}$** : Maximum voltage to compare too
- ▶  **$V_{in}$** : Signal to measure
- ▶ **SC, EN**: Trigger input and enable input
- ▶  **$b_0 - b_{r-1}$** : Digital output pins
- ▶ **CC**: Comparison complete

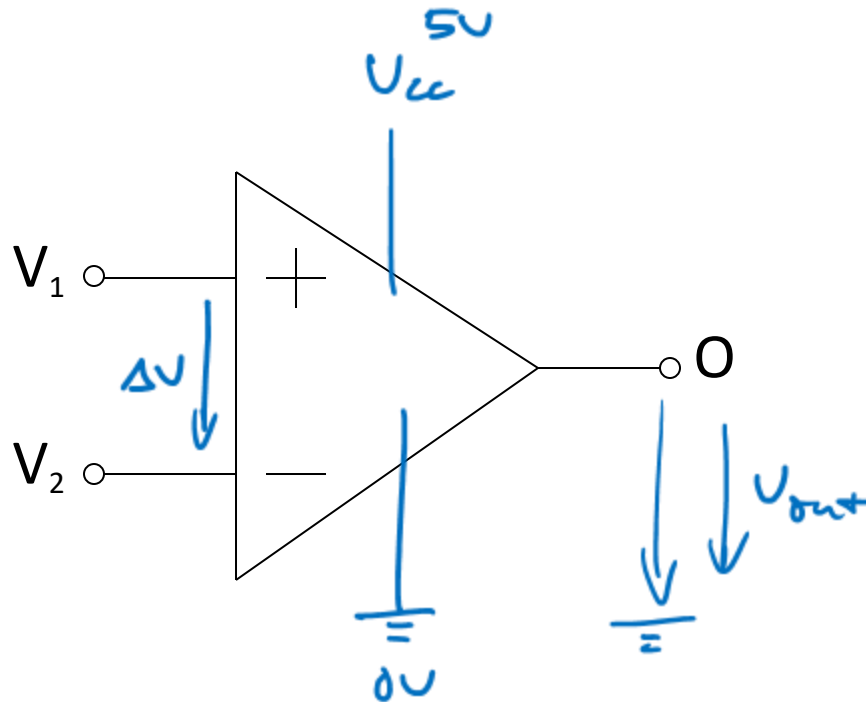
# Sample and Hold

*Abtast-Halte-Glied*

- ▶ Problem: Current may change during measurement (fluctuate)
- ▶ Solution: Create “trap” for voltage
- ▶ Capacitor is charged and disconnected



# Simple solution: Analog Comparator



$$V_{out} = \begin{cases} 5V^{\uparrow} & \text{if } V_1 > V_2 \\ 0V^{\circ} & \text{if } V_1 < V_2 \end{cases}$$

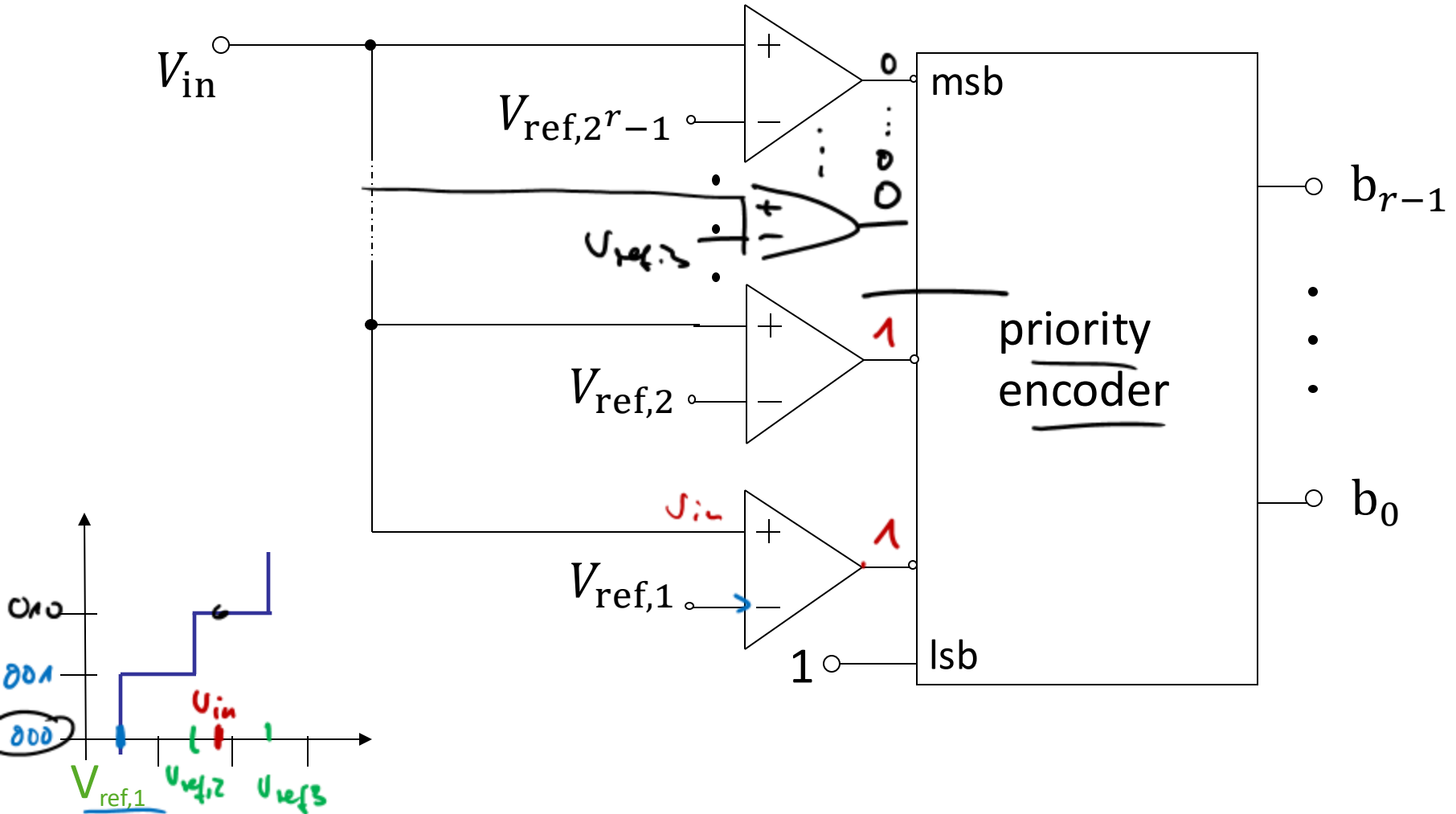
$$\cancel{V_1 = V_2}$$

- ▶ Compares simply  $V_1$  and  $V_2$
- ▶ Output 1 when  $V_1 > V_2$
- ▶ Else 0
- ▶ Suffers from meta-stability
- ▶ Obviously 1 bit
- ▶ Basis for more complex solution



# Flash Converter

00000111  $\rightarrow$  010

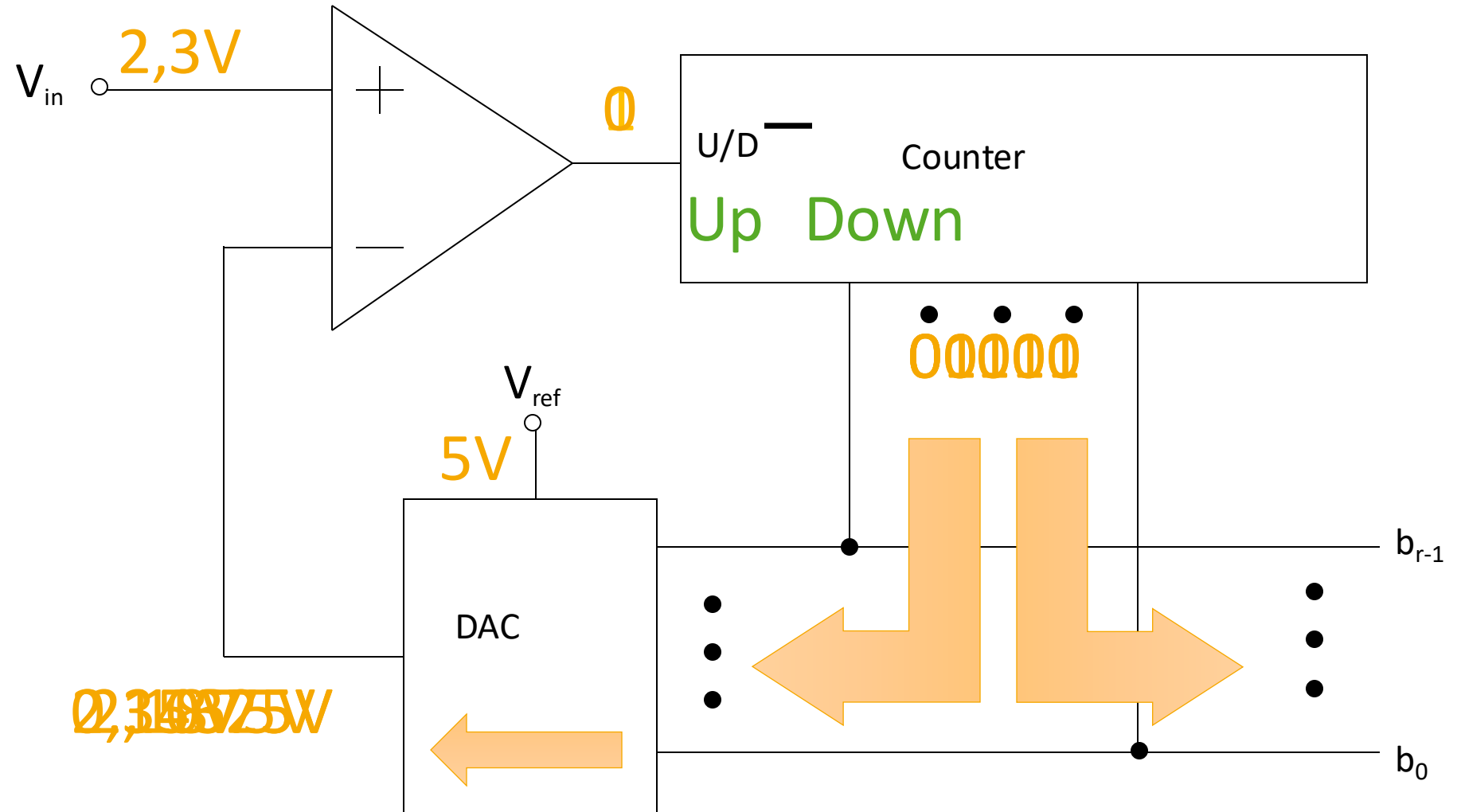


# Properties of a Flash Converter

---

- ▶ Direct application of DAC principle
- ▶ Flash means fast: Simultaneous check
- ▶ Complexity of converter:  $2^r - 1$  comparators needed for encoding
- ▶ Thus expensive

# Tracking Converter



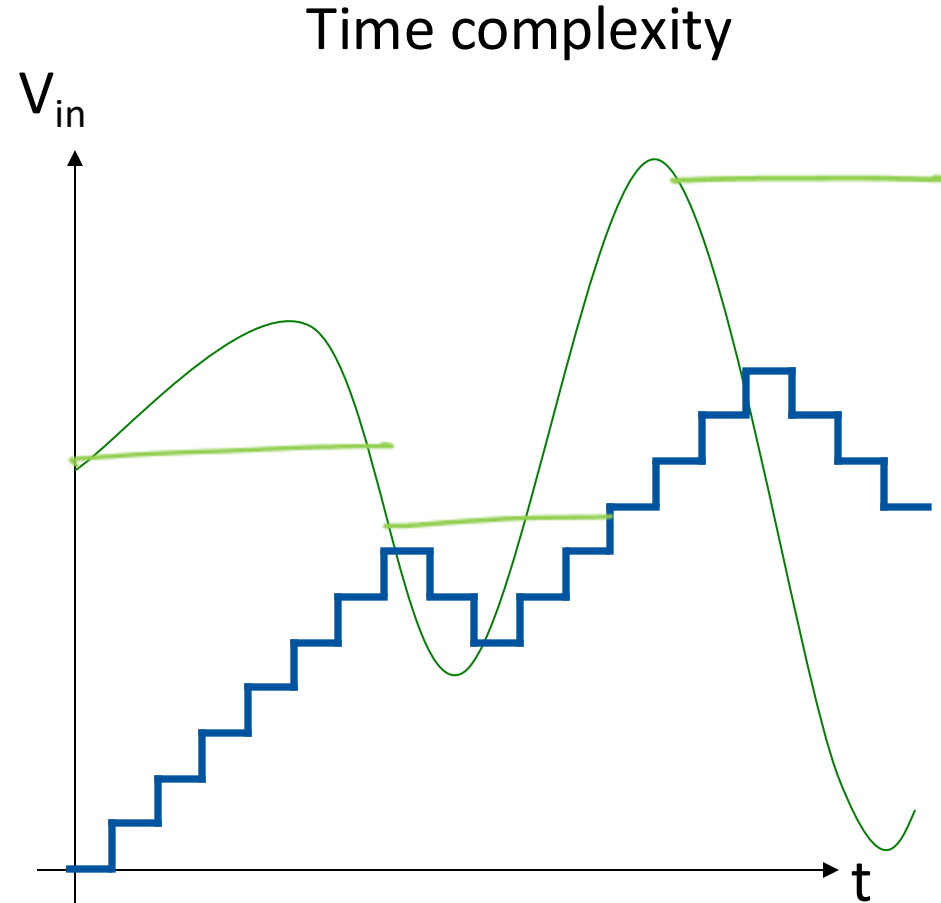
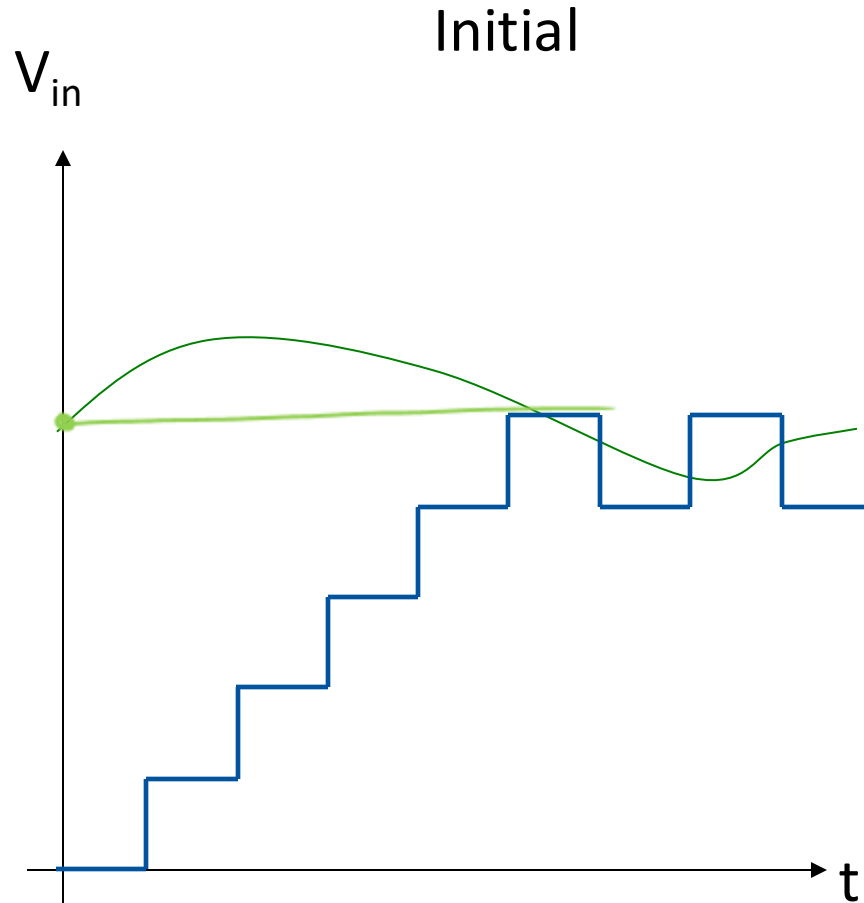
# Tracking Converter Principle

- ▶ DAC used to do ADC
- ▶ Counter holds digital estimate of value
- ▶ Counter changed linearly according to outcome of comparator
- ▶ Sample and hold
- ▶ Disadvantage: Tracking takes some time, worst case  $\mathcal{O}(2^r)$ 
  - 00000 → 00001 → 00010 → ... → 01110 → 01111 → 01110 → ...
- ▶ In the beginning not precise



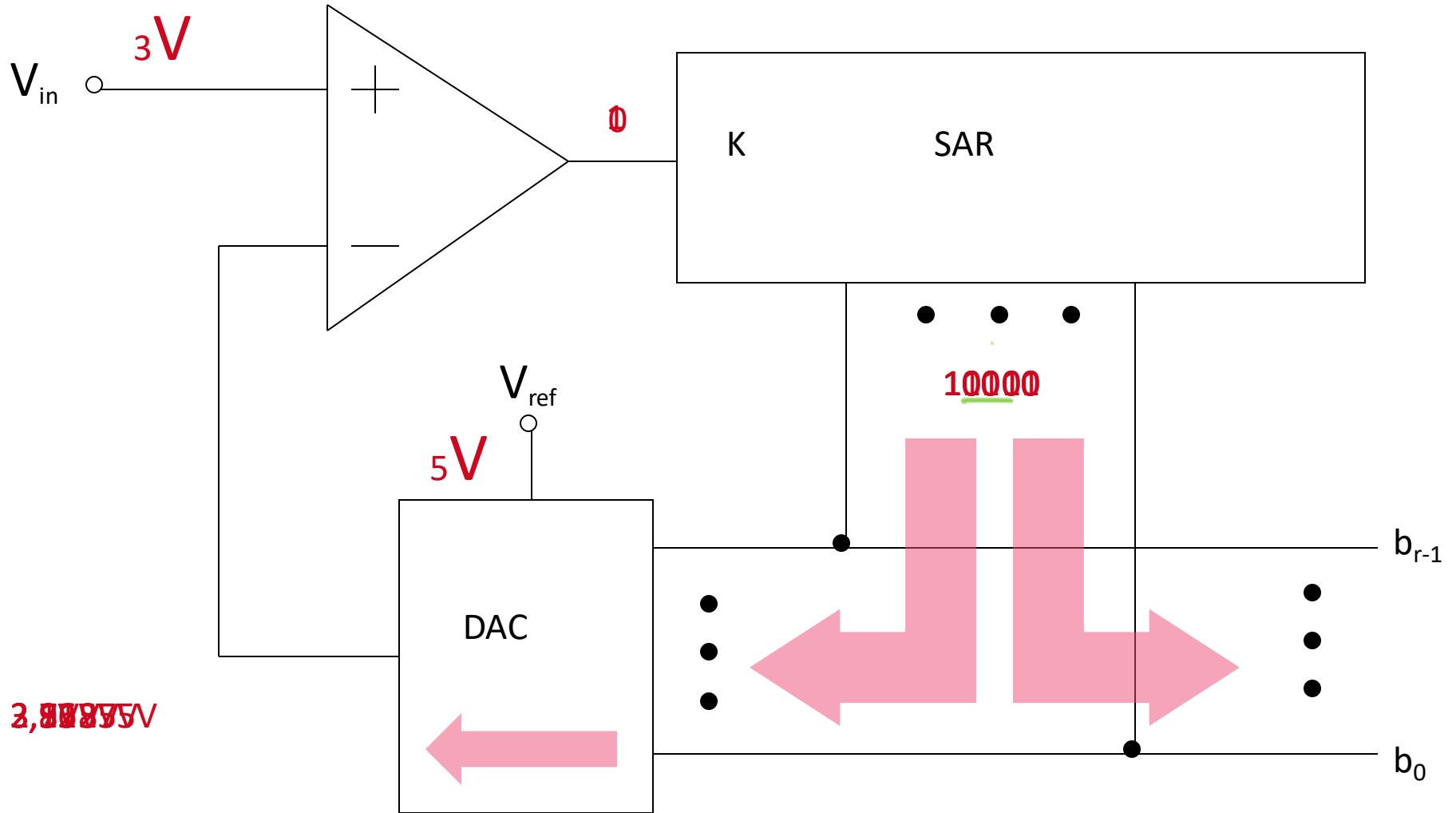
Oscillation

# Examples of Tracking Problems



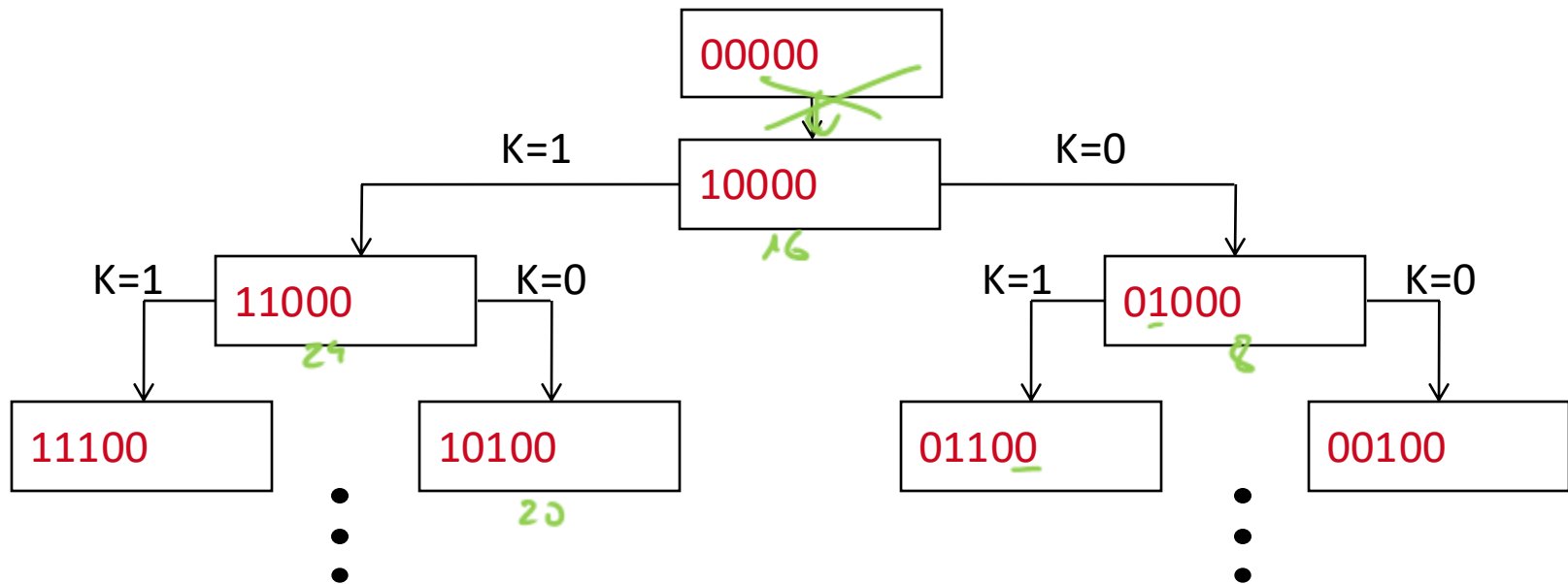
# Successive Approximation Converter

Binäre Suche



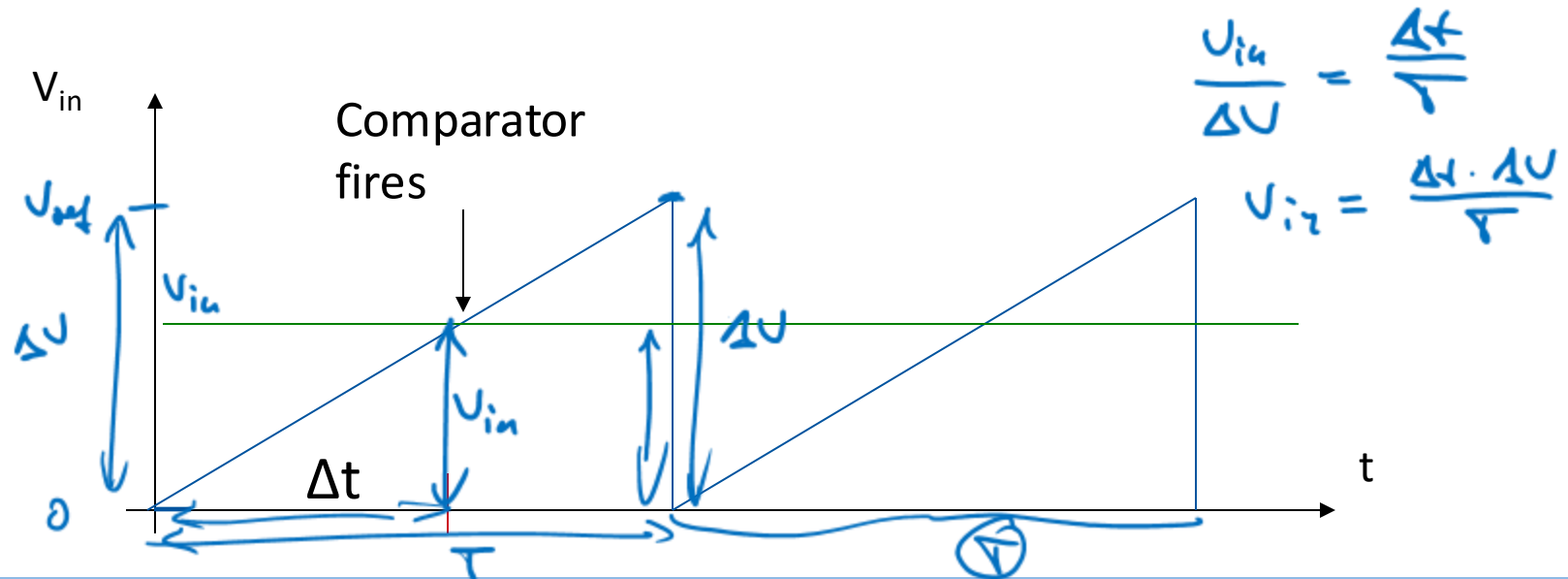
# Properties of a Successive Approximation Converter

- ▶ Has a successive approximation register (SAR)
- ▶ More sophisticated algorithm for approximation used:
  - Starts with  $b_{2^r-1}$
  - Changes are made in exponential steps
  - Only  $r$  comparisons needed,  $\mathcal{O}(r)$



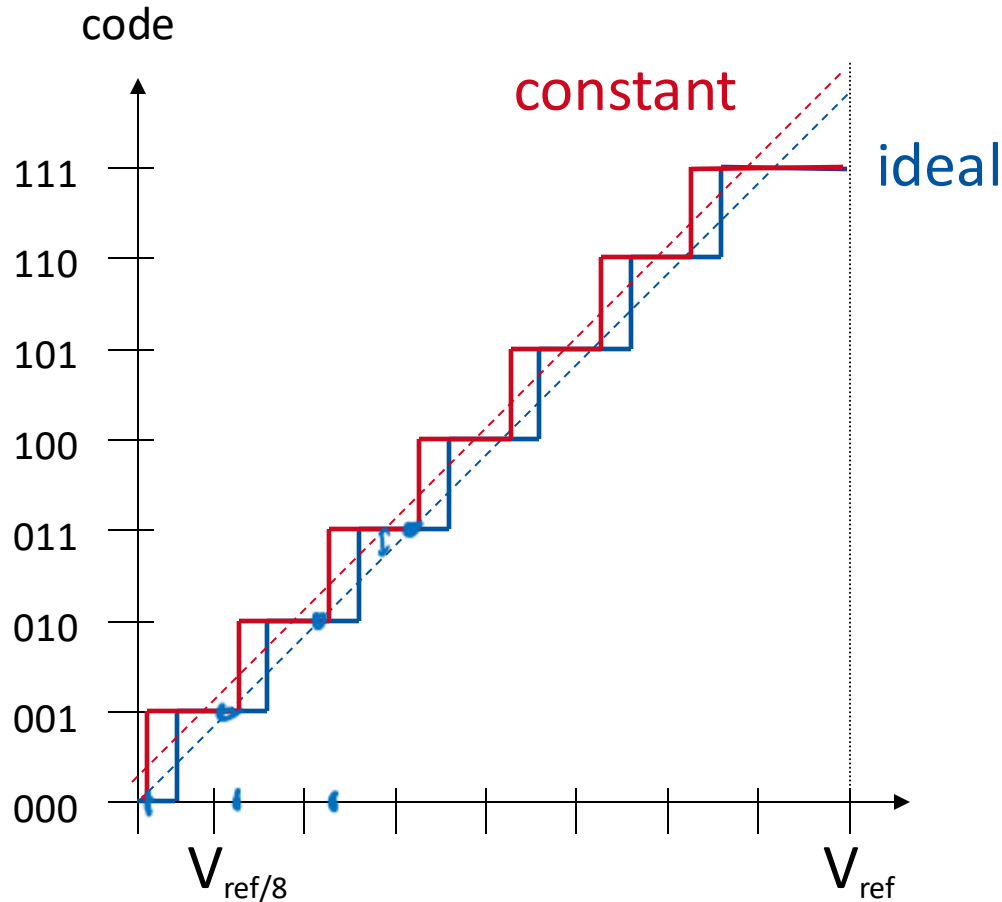
# Ramp-Compare Conversion

- ▶ Saw signal is generated
- ▶ Signal is then compared to measured signal
- ▶ When ramp voltage is reached, a comparator fires
- ▶ Value can be calculated by measuring the time until it fires
- ▶ Ramp signal may be reused for additional conversions



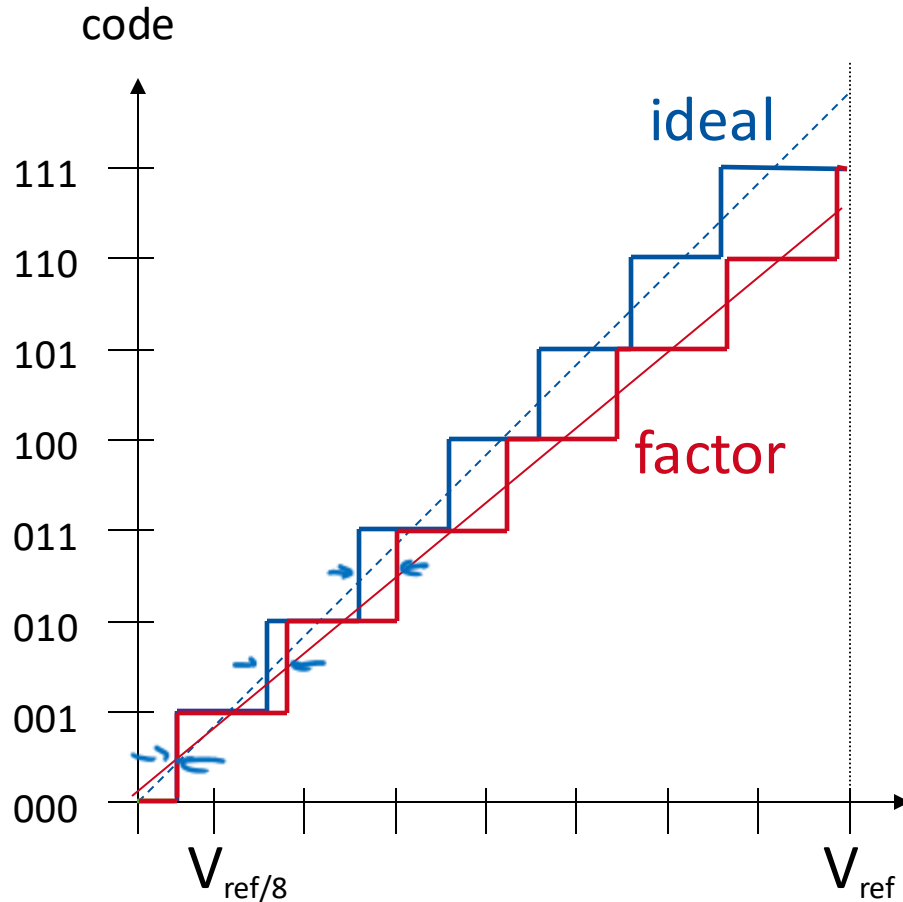


# Offset Error



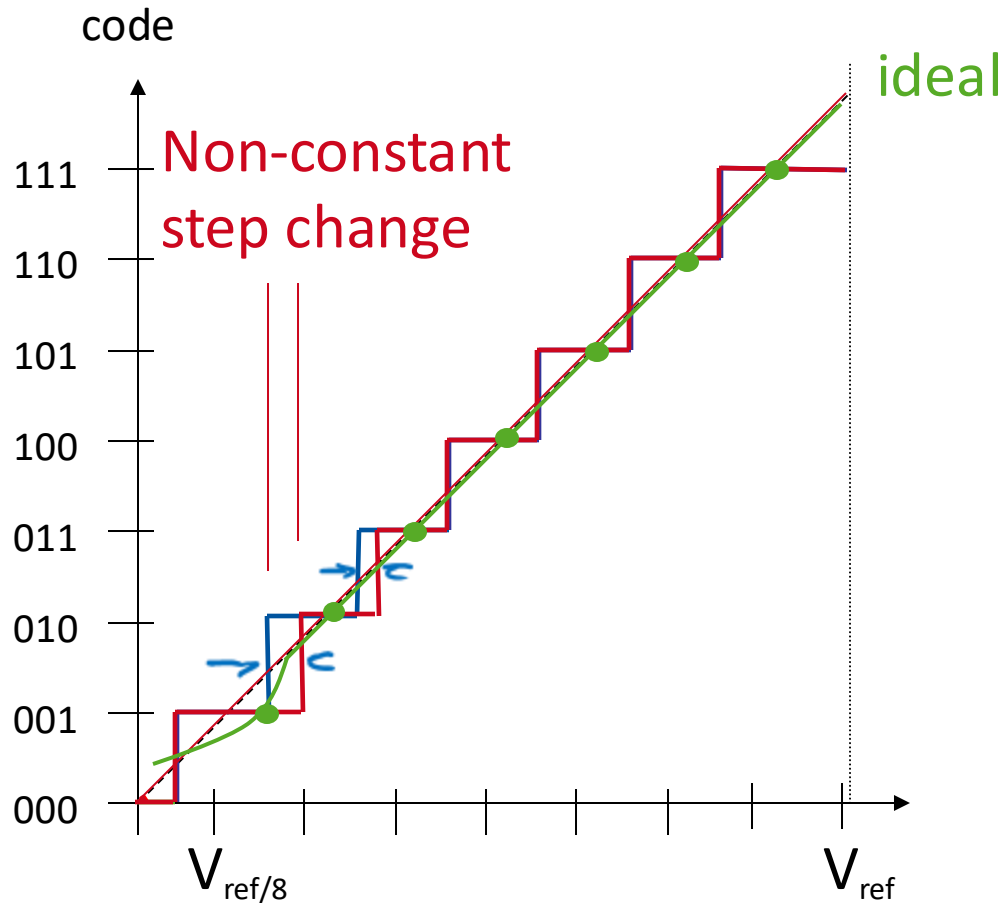
- ▶ Constant value added to function
- ▶ Step size is the same
- ▶ Simple to fix
- ▶ Often build-in offset correction

# Gain Error



- ▶ Step size differs by a constant value
- ▶ Gradient diverges
- ▶ Build-in gain adjustment

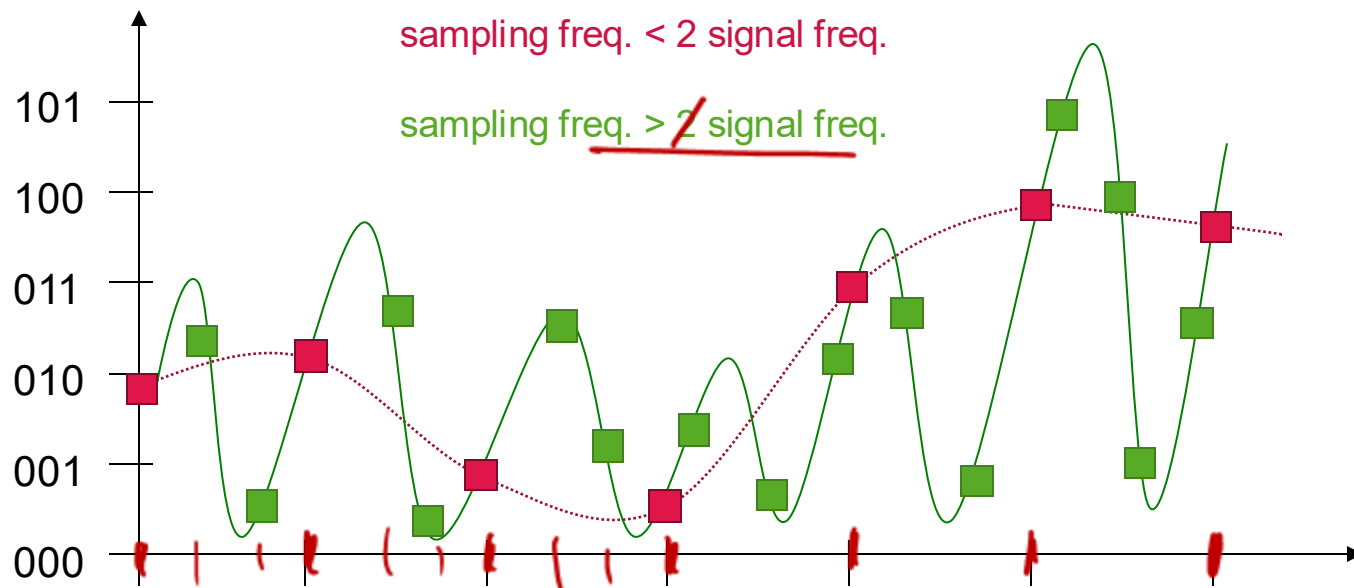
# Differential Non-Linearity



- ▶ Difference between code transition not 1 lsb
- ▶ “Step size” different
- ▶ DNL Error is worst case deviation

# Conversion Effect: Aliasing

- ▶ Signal has a higher frequency than conversion
- ▶ Nyquist criterion not met
- ▶ Converted signal is only an “alias” of original signal
- ▶ Anti-aliasing filters are low-pass filter



# ATMega16 ADC Description

- ▶ 10-bit Resolution (with atomic read)
- ▶ 0.5 LSB Integral Non-linearity
- ▶  $\pm 2$  LSB Absolute Accuracy
- ▶ 13 - 260  $\mu$ s Conversion Time
- ▶ Up to 15 kSPS (kilo samples per second) at Maximum Resolution
- ▶ 8 Multiplexed Single Ended Input Channels (single means uses GND)
- ▶ 7 Differential Input Channels (compare 2 signals; not necessarily GND)
- ▶ 2 Differential Input Channels with Optional Gain of 10x and 200x(1)

# ATMega16 ADC Description (cont.)

---

- ▶ 0 - VCC ADC Input Voltage Range
- ▶ Free Running or Single Conversion Mode (continuous vs. once)
- ▶ ADC Start Conversion by Auto Triggering on Interrupt Sources
- ▶ Interrupt on ADC Conversion Complete

# Summary

---

- ▶ Digital to analog: PWM + low-pass filter, binary weighted resistor circuit, R-2R Ladder
- ▶ ADC: a transfer function
- ▶ Analog to digital conversion: comparator, flash converter, tracking converter, successive approximation converter, ramp-compare
- ▶ Sample and hold
- ▶ 4 error types with seriousness und ways to deal with them
- ▶ ATmega16 has very sophisticated control possibilities
- ▶ Note: there are more converters like single/dual slope converter, pipeline converter...
- ▶ Main source by TU Vienna