Technische Hochschule Rosenheim
Technical University of Applied Sciences

Prof. Dr. Florian Künzner
Technical University of Applied Sciences Rosenheim, Computer Science

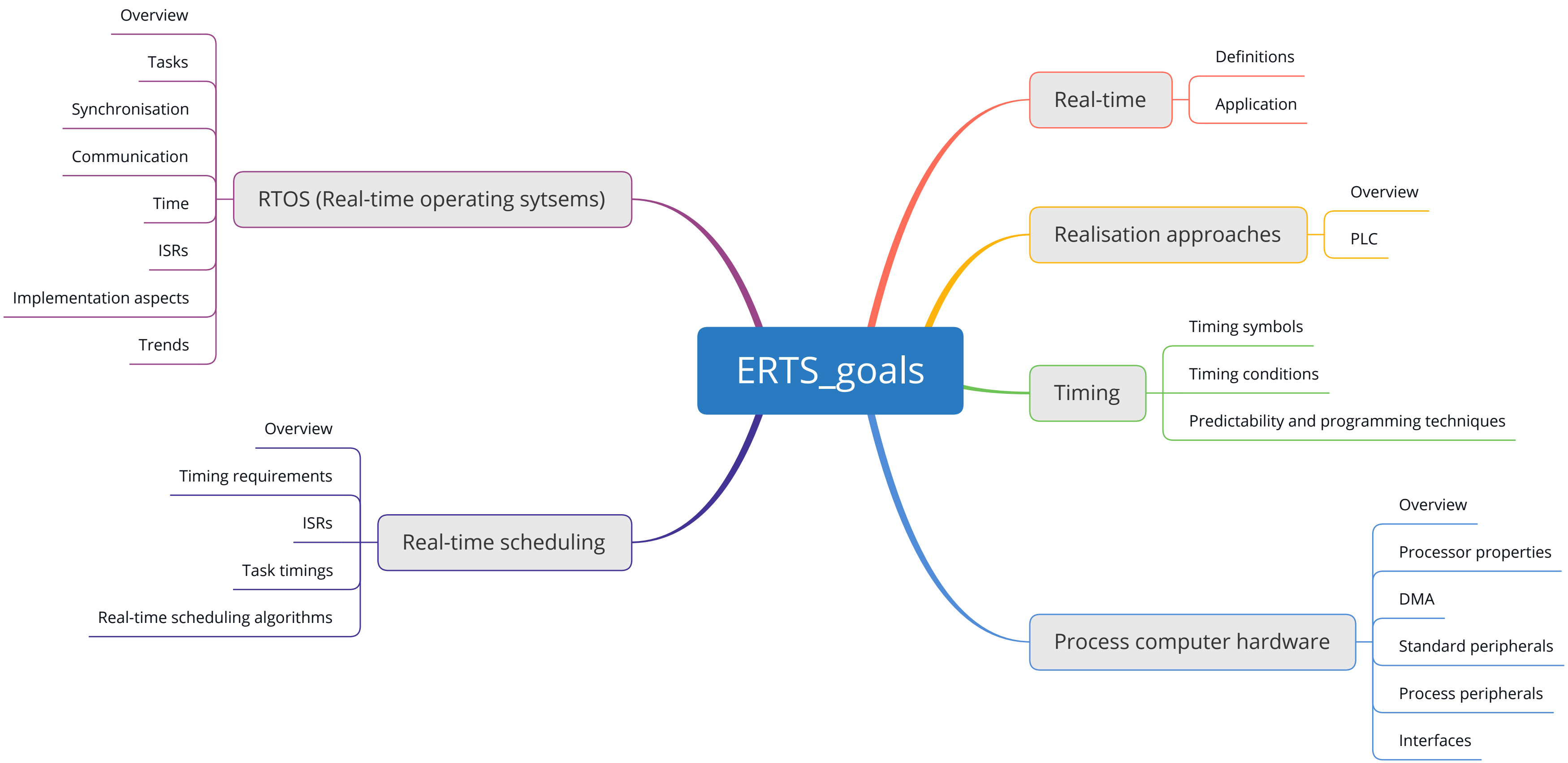
ERTS - Embedded real-time systems

ERTS 5 – Process computer HW 2

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal



CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Goal

ERTS::Process computer HW 2

- Digital I/O
- Analog I/O
- Real-time clocks
- RTC examples
- Pulse-width modulation

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Digital input

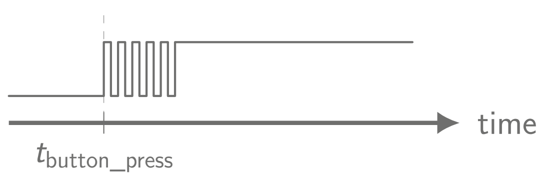
Used for reading binary signals:

- Generating interrupts: on rising or falling edge
- Timer: trigger signal
- On/off detection
- Data transfer (read): UART, SPI, I^2C, RS232, ...

A recurring problem: contact bounce

Hardware switches introduce a contact bouncing, which makes the input go high and low several times, instead of exactly once.

HW switch



Possible solutions:

- Hardware solution based on electrical parts
- Software solution

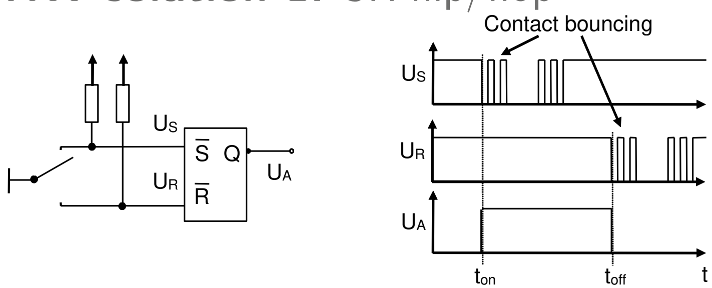
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Digital input

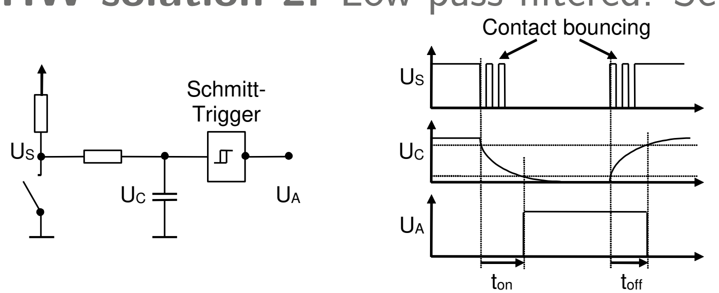
Possible hardware solutions for contact de-bounce:

- **HW solution 1:** SR flip/flop



⇒ Only possible for switches

- **HW solution 2:** Low pass filtered: Schmitt trigger



⇒ Can introduce a switch-on and switch-off delay

⇒ Also possible for push buttons

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Digital input

Possible software solutions for contact de-bounce:

- **SW solution 1:** Read input multiple times in a defined time period

$\Rightarrow$ Introduces an additional CPU load and an additional delay (read time)

- **SW solution 2:** Introduce a waiting timer after an impuls

```
//init
bool old_state = false;

//cyclic application
while(true) {
  bool io_state = read_io_port();
  if(io_state != old_state) {
    //perform some action

    delay(WAIT_TIME);
    old_state = io_state;
  }
}
```

$\Rightarrow$ Usually not suitable for real-time systems

Digital input

Possible software solutions for contact de-bounce:

- **SW solution 3:** Remember last switch time

```c
//init
const uint32_t DEBOUNCE_DELAY = /*HERE SOME APPROPRIATE TIME*/;
uint32_t last_switch_time = 0;
uint32_t current_time = 0;
bool old_state = false;

//cyclic application
while(true){
  bool io_state = read_io_port();
  current_time = get_time();
  if(io_state != old_state &&
     (current_time - last_switch_time) > DEBOUNCE_DELAY)
  {
    //perform some action

    old_state = io_state;
    last_switch_time = current_time
  }
}
```

⇒ `get_time();` may be expensive and relying on a SysTick

⇒ Additional variables (e.g. `last_switch_time`) required

Digital output

Used for writing binary signals:

- On/off: relais, engine, state information, ...
- Pulse-width modulation (PWM)
- Timer
- LCD
- Data transfer (send): UART, SPI, I^2C, RS232, ...
- ...

Usually, transistors are used for powering/switching relais, engines, ..., to not overload the digital output pin, connected to the processor.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Analog input

Analog do digital converter (ADC)

The analog value (voltage) is converted into a number of bits in a register.

Details:

- Sample- and hold amplifier (remembers value)
- Converting into bits
- For multiple channels: each has its own sample- and hold amplifier

Types:

Class	Principle	Converting time	Resolution	Example
Counting method	1 Value/step	0.1 . . . 1000 ms	8 - 20 Bit	Dual-slope (integrating) ADC
Successive method	1 Bit/step	0.1 . . . 100 μs	8 - 16 Bit	Successive approximation ADC using a binary search
Parallel method	1 Conversion/step	1 . . . 100 ns	4 - 10 Bit	Time-interleaved ADC with m parallel ADCs

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

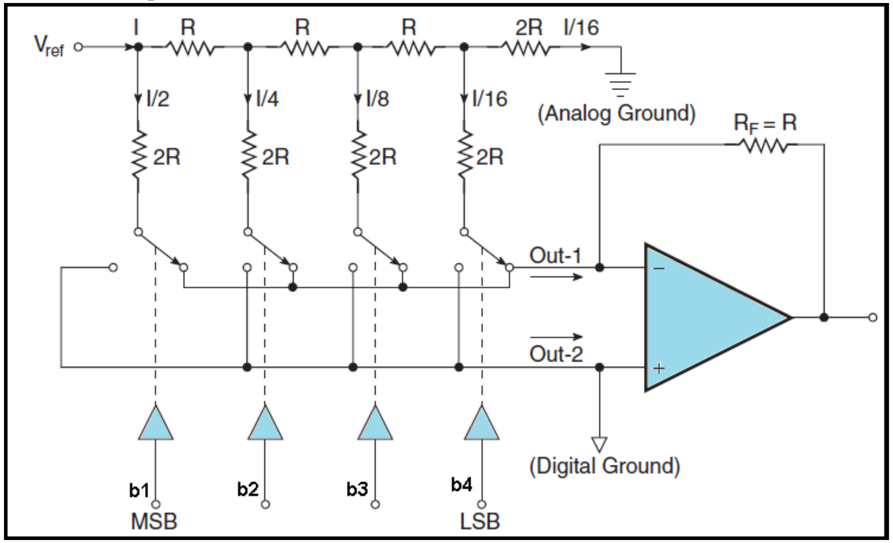
Analog output

Digital to analog converter (DAC)

The bits of a register are converted into an analog value (voltage).

- Number of bits ⇒ resolution of accuracy
- Modes
  - unipolar: 0 to $+x$ V
  - bipolar: $-x$ to $+x$ V
- Current range: e.g. 0 mA to 20 mA

Example: 4-bit R-2R ladder DAC:



[source: electronicsengineering.nbcafe.in]

Resulting voltage:

$$V_0 = V_{ref} * (R_F/R)[b_1/2^1 + b_2/2^2 + b_3/2^3 + b_4/2^4]$$

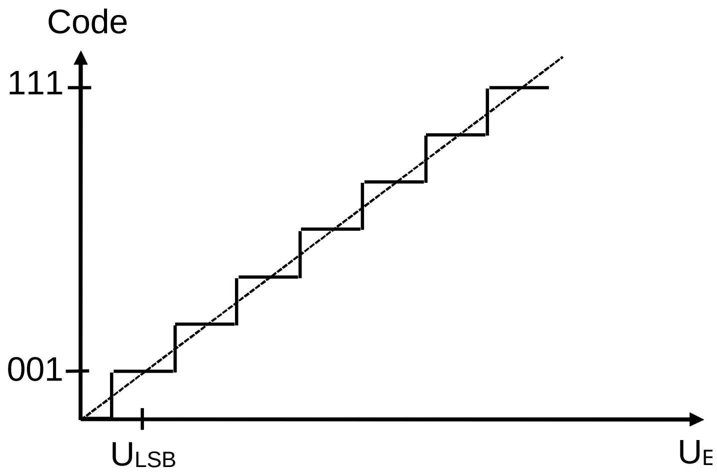
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

A/D converting errors

Static errors

Characteristic curve of a 3 bit A/D converter:



Problems:

- **Quantisation error**: $\pm 0.5 \times 2^{-n}$; $n =$ number of bits
- Zero-point error (horizontal misalignment) (may be solved with a potentiometer)
- Gain error (wrong gain) (may be solved with a potentiometer)
- Linearity error (may be corrected in software, tricky)

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

A/D converting errors

Dynamic errors

Nyquist–Shannon sampling theorem:

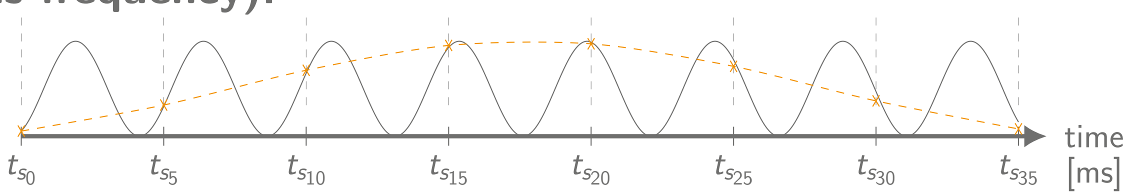
$$f_s > 2 \times f_{max}$$

- $f_s \rightarrow$ Sampling frequency
- $f_{max} \rightarrow$ Maximum frequency of signal

$\Rightarrow$ In practice, $f_s > 10 \times f_{max}$

Possible problems:

- **If f_s is to small ($\Rightarrow$ alias frequency):**
  — Signal with f_{max}
  --- Sampled signal with f_s



- If the **sampling time $\triangle t_s$ has a high jitter**, an additional source of error (unpredictability) is introduced.

$\Longrightarrow$ Mitigation of the problem possible with a sample-and-hold amplifier.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Real-time clocks: overview

In microcontrollers (and PCs), different real-time clock (RTC) mechanisms exist:

- Clock (absolute time clock)/timer
- Relative clock/relative timer
- Time-out monitoring

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Clock (absolute time clock)/timer

This kind of clocks are often called real-time clocks (RTC).

Clock mode:

- Gives the exact absolute date and time
- Usable for initialising software based time management software

Timer mode:

- Triggers an interrupt at a specified absolute time
- Wake up a device (e.g. from sleep mode)

Variants:

- Battery buffered: Allows working while the device is off
- Radio-controlled: Signal is received from a station (DCF77 or GNSS). Synchronisation can take up to several minutes.

Relative clock/relative timer

Clock mode:

- Allows relative time measurements
- With an appropriate initialisation, an absolute clock can be implemented

Timer mode:

- Triggers an interrupt at a specified relative time
- Wake up a device (e.g. from sleep mode)
- SysTick for the OS

Variants:

- Usually quartz-based with a resolution of $\approx 1\ \mu s$
- Also low-power variants possible: working in sleep mode

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Time-out monitoring

Monitor if an action has been taken place within a defined time period. If not, perform an emergency reaction.

Watchdog/timeout:

```
action();

//reset counter
feed_the_dog();
```

In the hardware:

```
if(counter > max_counter){
  perform_emergency_reaction();
  //reaction: often restart
}
```

Hartbeat:

```
while(true){
  action();

  //reset counter
  feed_the_dog();
}
```

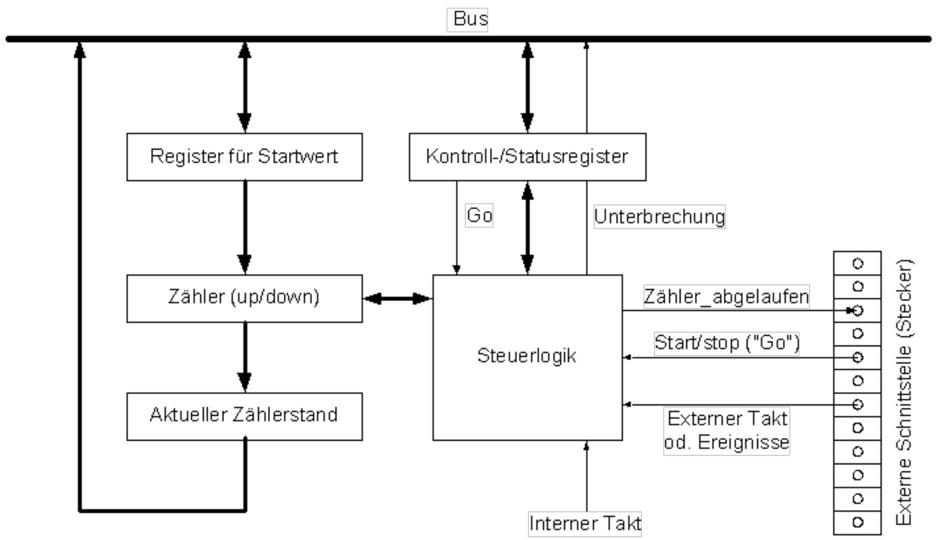
In the hardware:

```
if(counter > max_counter){
  perform_emergency_reaction();
  //reaction: often restart
}
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Abstracted real-time clock for lecture



Modes:

- Stop mode*: for time measurements
- Alarm/wake up mode*: generates interrupt after a specified time
- Counting mode: counts events (internal or external), can also generate interrupts
- Repeat mode: for stop, alarm/wake up, or counting mode

Tick:

- Internal clock
- External clock or events

* The stop and alarm/wake up modes are often called capture and compare.

Additionally possible:

- Regular pulses, with an
- Adjustable pulse length (PWM)

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

(a) RTC example: stop mode

The time of a program sequence should be measured.

Program:

```
//configure
RTC_COUNTER = 0;
RTC_STATE   = RTC_MICRO_SEC | GO;

//program sequence to measure
do_x();

uint32_t micros = RTC_COUNTER;
RTC_STATE &= ~GO; //stop
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

(b) RTC example: alarm mode

A task τ should be delayed for x μs.

Task τ:

```
//configure
RTC_IVT     = &rtc_isr;
RTC_COUNTER = x;
RTC_STATE   = RTC_IE_ENABLE | RTC_ALARM_MODE | RTC_MICRO_SEC | GO;

P(semaphore); //wait for x micro seconds
//continue after (x+ >>0) micro seconds
```

RTC ISR:

```
void rtc_isr() {
    V(semaphore);
    IRET; //return from interrupt, depends on HW
}

//This is only a principal view, the ISRs are often encapsulated by
//a HAL (hardware abstraction layer) or an OS
```

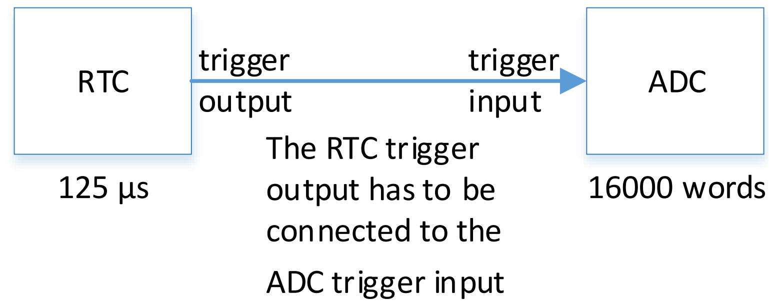
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

(c) RTC example: analog signal sampling

A signal (± 5 V) should be sampled with a time interval of $125\ \mu$s for a period of 2 s.

- Number of words/s: $\frac{1}{125\ \mu s} = 8000$ words/s
- Within 2 s: $2 \times 8000 = 16000$ word samples



Main program:

```c
volatile uint32_t buffer[16000];

//configure A/D converter
ADC_WORD_COUNT = 16000;
ADC_BUS_ADR    = &buffer;
ADC_STATUS     = ADC_IE_ENABLE | ADC_RTC_TRIGGER_ENABLE
               | ADC_DMA_ENABLE | ADC_CHANNEL0 | GO;
ADC_IVT        = &adc_complete_isr;

//configure RTC
RTC_COUNTER = 125;
RTC_STATE   = RTC_ALARM_MODE | RTC_REPEAT | RTC_MICRO_SEC | GO;
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

(c) RTC example: analog signal sampling

Alternative solution with alarm mode:

Main program:

```
volatile uint32_t buffer[16000];
volatile uint32_t i = 0;

//configure
RTC_IVT     = &rtc_isr;
RTC_COUNTER = 125;
RTC_STATE   = RTC_IE_ENABLE
            | RTC_ALARM_MODE
            | RTC_REPEAT
            | RTC_MICRO_SEC
            | GO;

P(semaphore); //wait for all samples
```

RTC ISR:

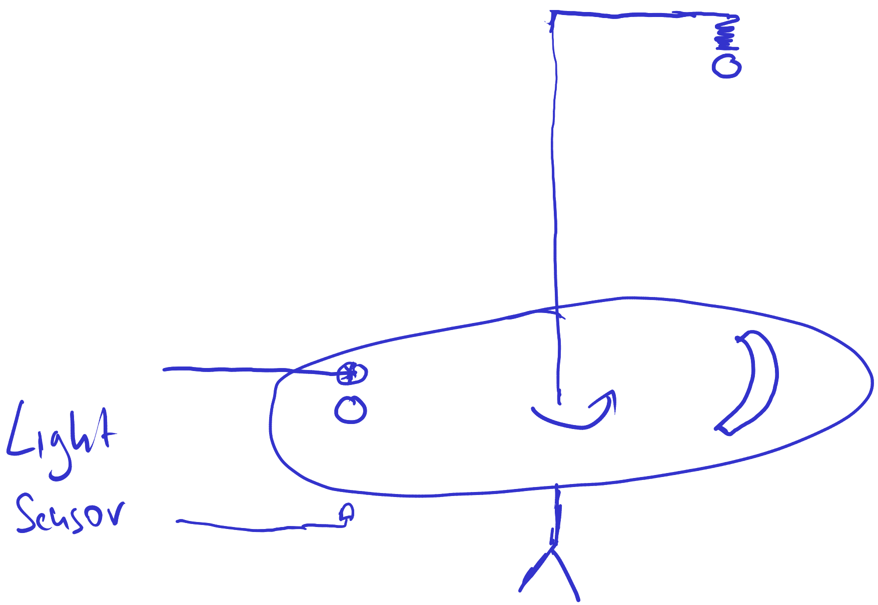
```
void rtc_isr() {
  ADC_soc();       //start of conversion
  ADC_wait_eoc(); //wait for completion
  buffer[i] = ADC_VALUE;

  ++i;
  if(i >= 16000){ //all samples done
    V(semaphore);
  }

  IRET; //return from interrupt,
        //depends on HW
}
```

Problems with this solution:

- There might be an additional timing delay until the ISR is called.
- There is an additional load on the CPU for the data transfer

(d) RTC example: ball drop example

Measure the rotation time of a disc with a light sensor and a hole in the disc.



Measure rotation time:

```
//configure
RTC_STATE    = RTC_STOP_MODE
             | RTC_MICRO_SEC;
wait_for_hole();
RTC_STATE   |= GO;   //start

wait_for_hole();
RTC_STATE  &= ~GO; //stop
rotation_time = RTC_COUNTER;
```

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

(d) RTC example: ball drop example

Discussion on measure rotation time (`wait_for_hole()`):

1. Programmed I/O: busy wait/polling on ⇒ light sensor

⇒ High CPU load, but timely fast with a low uncertainty

⇒ Program design may be hard to write/maintain with other parallel tasks.

2. Light sensor ⇒ digital input $\overset{*}{\Rightarrow}$ ISR (start/stop/get counter RTC)

* Uncertainty in timing of the interrupt reaction time $T_{IRE} = T1 + T2 + T3$

3. Light sensor ⇒ start/stop input on RTC

⇒ Ideal solution in terms of time, but only possible with a suitable RTC

4. Light sensor ⇒ digital input ⇒ ISR ⇒ Task (start/stop/get counter RTC)

⇒ A high degree of uncertainty in terms of time, because additionally multi-tasking (schedling) effects can play a role.

⇒ Program design easier to write/maintain with other parallel tasks.

CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

(e) RTC example: watchdog

Watchdog functionality:

```
//configure
RTC_COUNTER = x + 2*y; //x = init time
                       //y = cycle time
RTC_STATE   = RTC_ALARM_MODE
            | RTC_RESET_ON_ALARM
            | RTC_MICRO_SEC
            | GO;

//cyclic application
while(true) {
    //here...
    //        ...application code with cycle time y (micro sec.)
    RTC_COUNTER = 2*y; //y = cycle time
}
```

Functionality:

The alarm (watchdog) shouldn't trigger, if the application (cyclic code) is always fast enough.

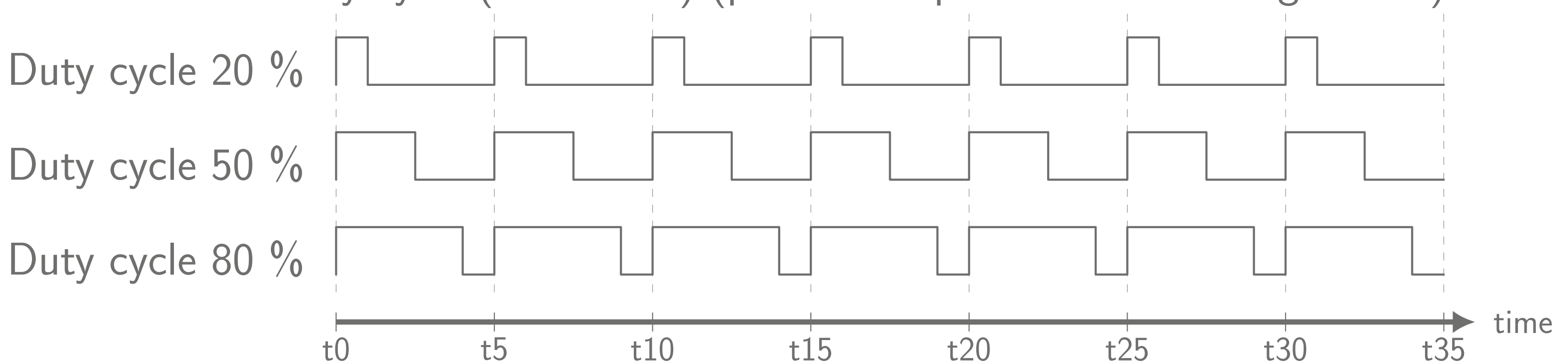
CAMPUS Rosenheim
Computer Science

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Pulse-width modulation

Pulse-width modulation (PWM) is used to control: engines, LED, ...

- Fixed time period T_P
- Square wave
- Variable duty cycle $(0 - 100\ \%)$ (part of the period where the signal is 1)



Generation possible with software or hardware

Software:

- Pro: May be more flexible than HW based PWM
- Con: CPU intense (I/O controlled by the software) + signal drift!

Hardware:

- Pro: HW based PWM signals can run independently without the CPU
- Pro: May be more accurate (means better predictability) than SW based PWM

CAMPUS Rosenheim
Computer Science

Summary and outlook

Summary

- Digital I/O
- Analog I/O
- Real-time clocks
- RTC examples
- Pulse-width modulation

Outlook

- DMA