GlobalLogic®

Basecamp: C/Embedded 10

Agenda

1. Homework review
2. TTL
3. GPIO
   - Input
   - Output
4. Clocks
5. Practice
   - LED blinking, Arduino style
   - LED blinking, AVR-C
   - Delay
   - Interrupt for a button

Hardware input/output

Hardware

Hardware logic

- Binary logic levels true-false should be mapped to a hardware levels
- Usually, hardware logic levels have more states, than just regular true-false:
  - High: presence of the signal (i.e. high voltage opposite to low, light is turned on)
  - Low: absence of the signal (i.e. low voltage, light is turned off)
  - Z – “high impedance” (i.e. hardware is not influencing on the signal)
  - X – transition state, aka hold state, or “don’t care”
- Binary logic have to be mapped to the hardware in order to make schematics work

GPIO

- Usually MCUs and SOCs have several pins, for general input/output
- Logic/digital levels
  - High is usually mapped to 1
  - Low is usually mapped to 0
- Additional states
  - Z is used when several devices connected to the same pin, and device doesn’t want to influence other chips communication

GPIO

Configuration

- Same pin can be used either for input, or for output
- Usually, there should be only one output connected to several inputs
- So in order to set the state of the port, chip should select
  - Direction: input or output
  - Default state: pull up/down (to high or low state)
  - Polarity of the pin
    - Allows to inverse the levels

Examples

Example 1:	Direction = 1 (output), port = 1 (high)
	Result: logical 1 (3v) on the output
Example 2:	Direction = 0 (input), pull-up = 0 (low)
	Another chip: direction 1 (output)
	Result: defined by the port of another chip
Example 3:	Direction = 0 (input), pull-up = 1 (high)
	Another chip: direction = 0 (input), pull-up = 0 (low)
	Result: undefined state
Example 4:	Direction = 0 (input), pull-up = 1 (high)
	Another chip: Hi-Z
	Result: 1 (according to the pull-up)

Useful links

- http://easyelectronics.ru/avr-uchebnyj-kurs-ustrojstvo-i-rabota-portov-vvoda-vyvoda.html
- https://habrahabr.ru/post/253213/
- https://habrahabr.ru/post/253961/
- https://habrahabr.ru/post/255715/
- https://habrahabr.ru/post/256269/
- http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

Practice on led/buttons

Practice agenda

Preparations

- Get Arduino with Mega 2560
- Fetch datasheet for Mega 2560
- Install Arduino IDE
- “Hello world”, Arduino style
    - Write Arduino-style hello-world https://www.arduino.cc/en/Serial/Print
    - Write Arduino-style led blinking https://www.arduino.cc/en/Tutorial/blink
    - Write Arduino-style button reading https://www.arduino.cc/en/Tutorial/Button

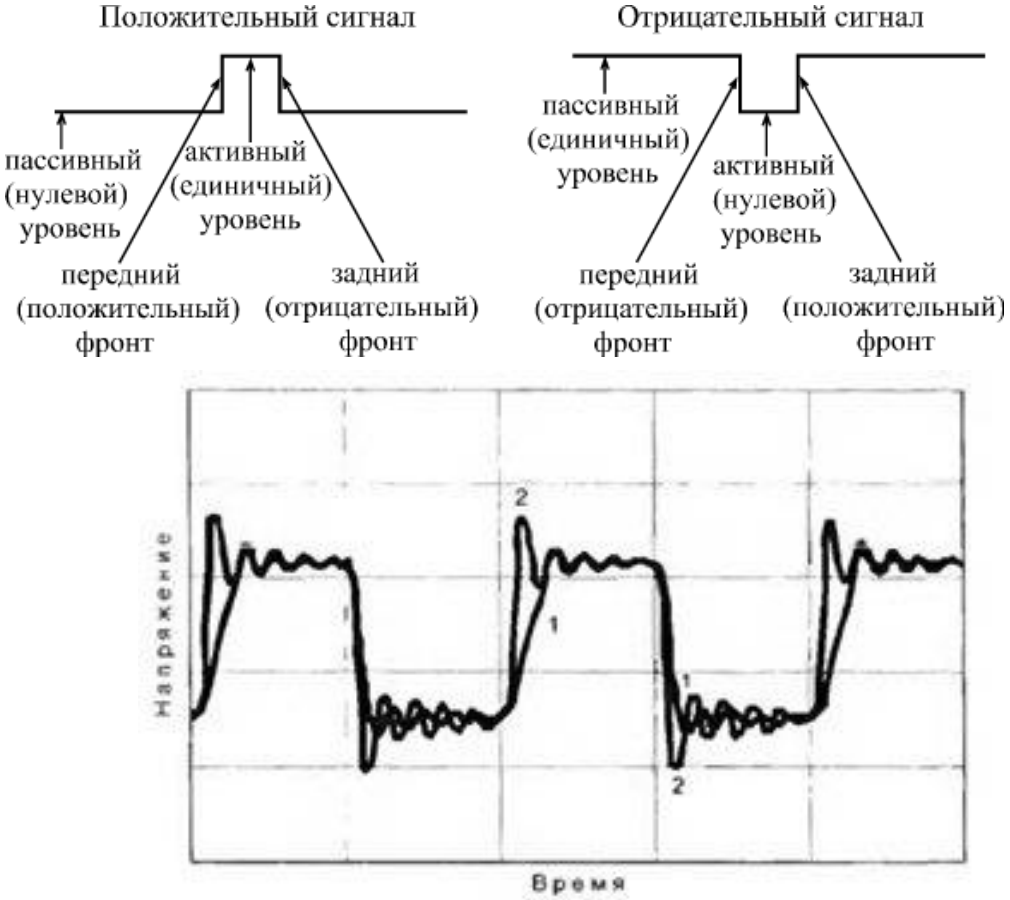
Switching to a real work

- Output, C style
    - Find the port connected to a led
    - Initialize port direction to output
    - Write 1 to the port
    - Do Arduino-style delay
    - Write 0 to the port
    - Do Arduino-style delay
- Input C style
    - Find the port connected to a led
    - Initialize port direction to output
    - Write 1 to the port
    - Do Arduino-style delay
    - Write 0 to the port
    - Do Arduino-style delay

Clocks and delays

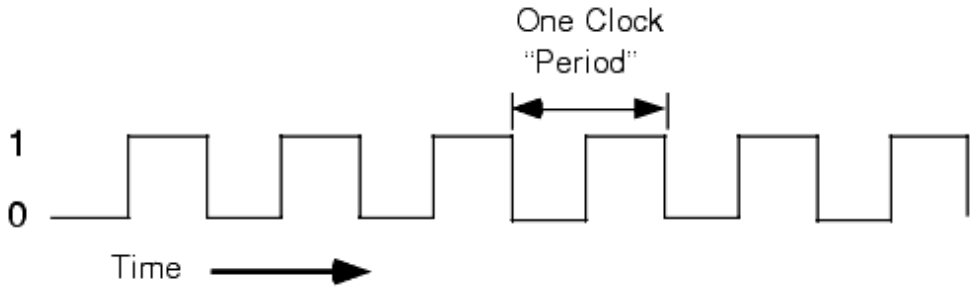
Signal shape

- Digital signals captured have shape
  - High and low level
  - Positive and negative edge
- Port reading can be done only when the signal is set to the level
  - Reading port value during transition state can show irrelevant results
  - MCU reads ports when the level is set.
- Edge can be detected with a special hardware, like interrupt controller.



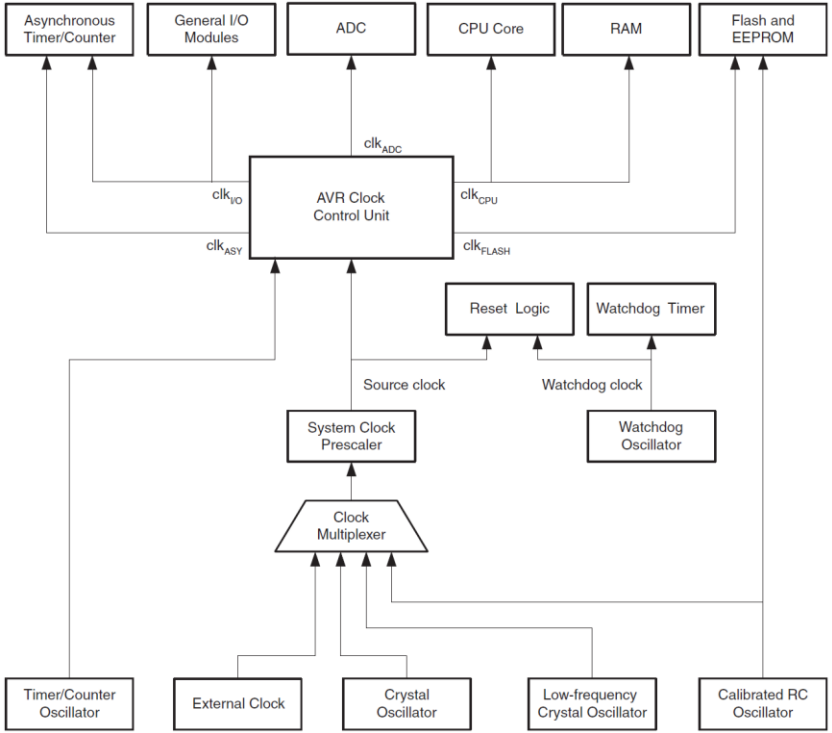
Clock introduction

- So-called clocks are widely used in digital electronics
  - Synchronize everything
    - Synchronize internal MCU structures and registers
    - Synchronize different controllers, like MCU with DDR controller, etc.
  - Time measurements
    - Both fast measurements and timers feeding
    - Timers are used for long delays, i.e. milliseconds and seconds
- Any processor should be clocked in order to change the states, read commands, execute the code, etc.
- AtMega has common selectable MCU clock source with a specified frequency



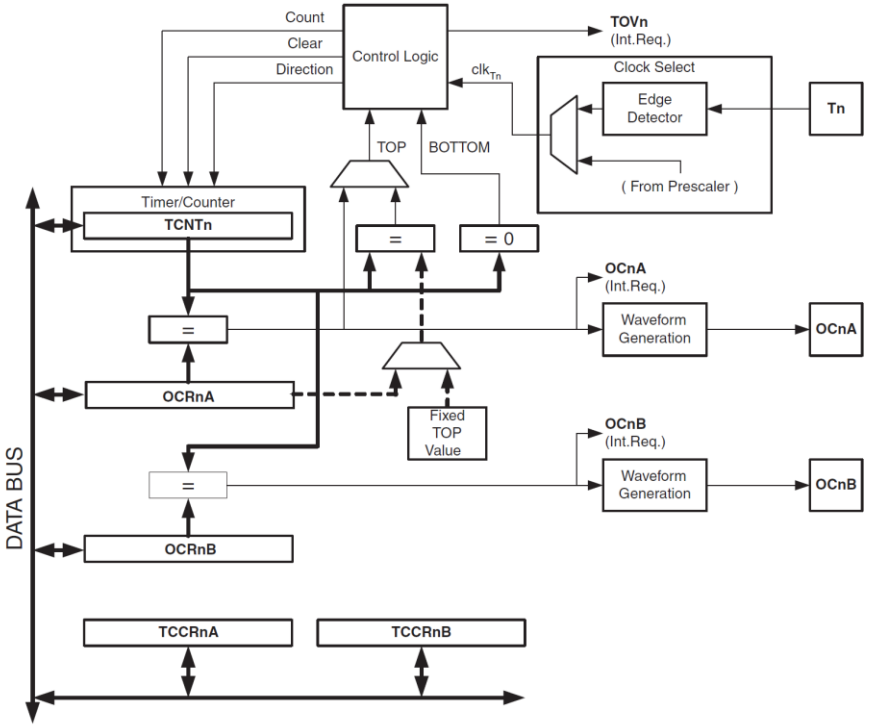
Clock sources

- Clocks can be programmably prescaled
- AVR8 has clock multiplexer allowing to select feeding from external clock, external oscillator or internal oscillator
- AVR8's clock unit has several outputs for CPU/memory, ADC and I/O and separate logic for a watchdog
- F_CPU is a macro from avr-gcc defining current CPU frequency



AVR8 timers

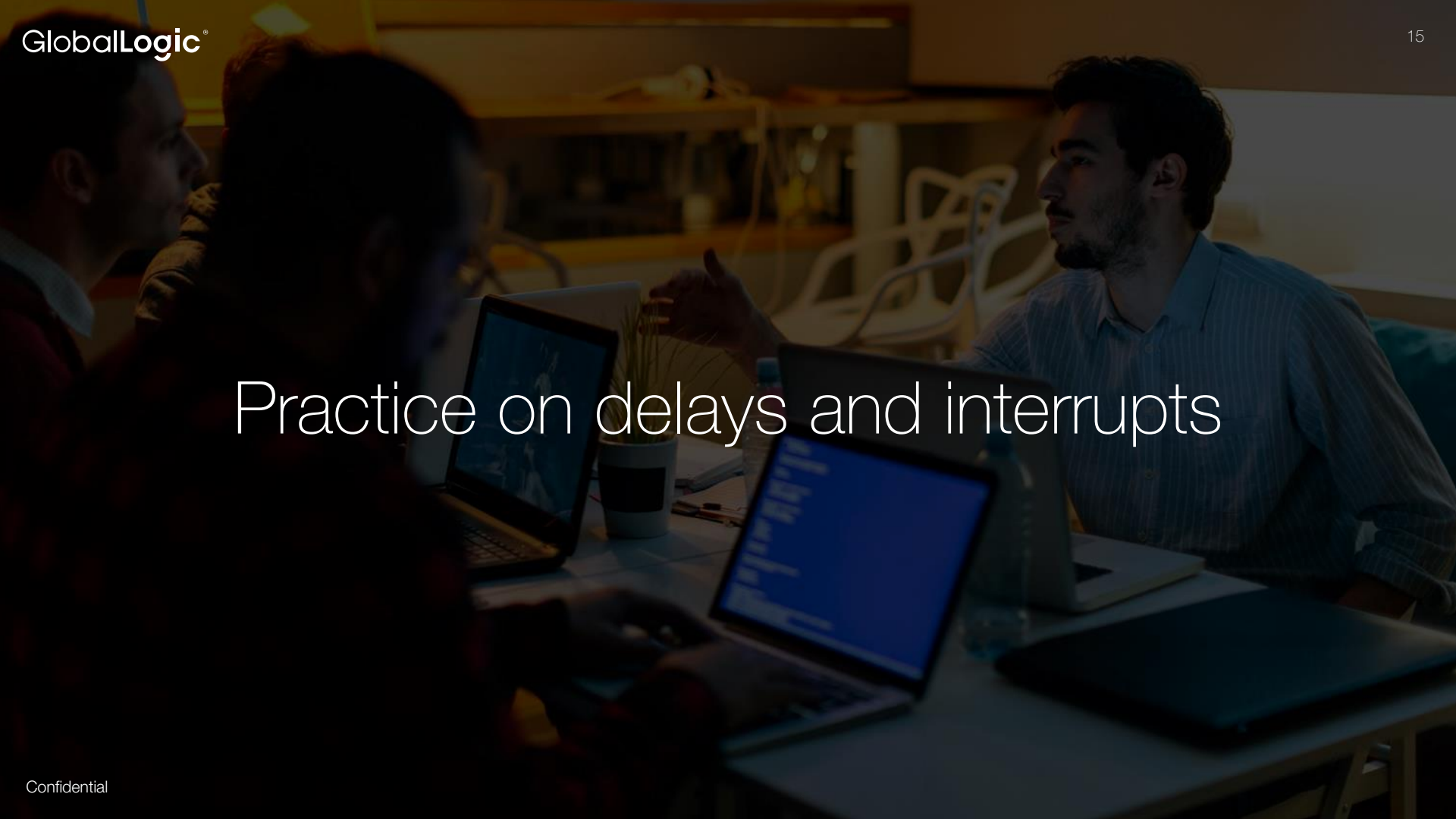
- Atmega 2560 has 6 timers, some of them are 8-bits timers, some of them are 16-bit timers
- There are timers with a comparator
- Timers with comparators and waveform generator can work as PWM
- Timers can be set for overflow interrupts, comparator interrupts and PWM output



Interrupt introduction

- Interrupts are used in order to break the regular MCU flow
- Interrupt routines are regular function pointers loaded in the interrupts table
- In order to control interrupts, developer can disable (mask) and enable (unmask) interrupts detection, if necessary.
- Atmega 2560 has several external interrupts
- Interrupts availabled at your mega: http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html

```
void setup() {
  cli()

  /* External interrupt enabling */
  PCICR |= 1;
  PCMSK0 |= INT_BIT;

  /* Timer 1 comparator interrupt enabling */
  TIMSK1 = 1 << OCIE1A;

  /* Timer 3 interrupt enabling */
  TIMSK3 = 1 << TOIE3;

  sei();
}

ISR(PCINT0_vect) {
}

ISR(TIMER1_COMPA_vect) {
}

ISR(TIMER3_OVF_vect) {
}
```



Practice on delays and interrupts

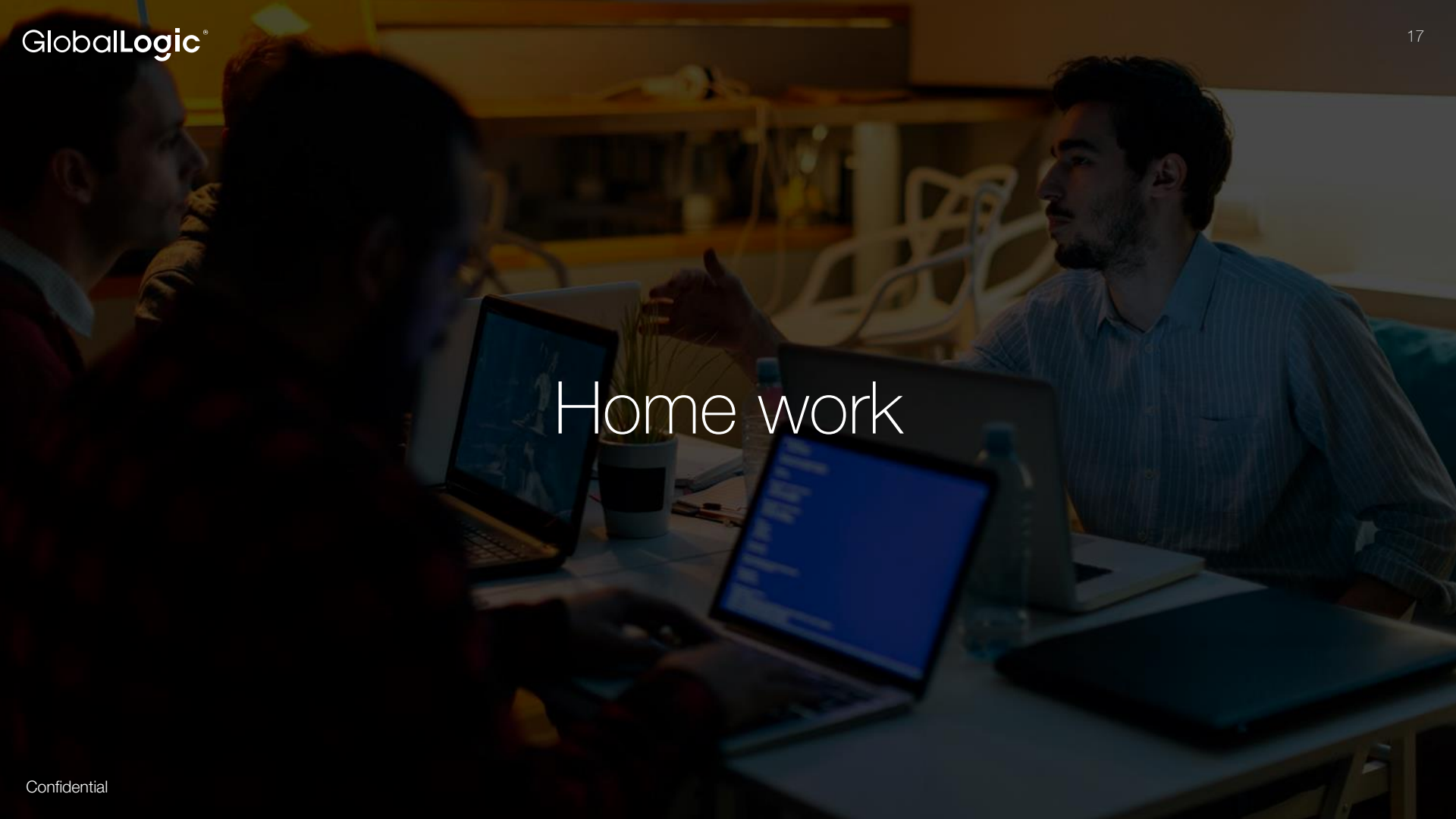
Practice

Clocks

- Write a delay function usable up to 1s
- Rewrite output using the function
- Register Timer overflow interrupt
- Register Timer comparing interrupt

Interrupts

- Register interupt on X pin, add serial output (Arduino-style) in order to observe interrupts.



Home work

Quiz and homework

1. Use PWM in order to dim the LED
2. Use ADC to get the value of encoder