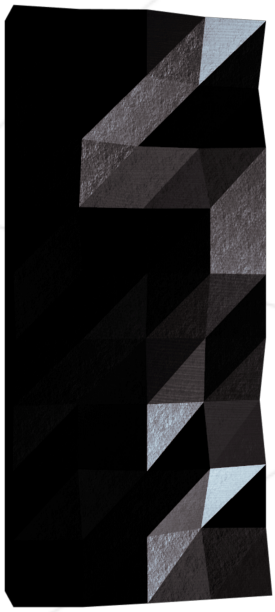

Embedded pool

Module04 : Interruptions

contact@42chips.fr

Summary: OBJECTION !

Version: 1

Chapter I

Introduction

An American study conducted on high-ranking executives has revealed that, on average, an executive is interrupted every 8 minutes, and the interruption lasts 3 minutes.
This may be due to an unexpected visitor, a phone call, an email, or something that poses a problem...

Adding to this, it takes a few minutes to regain one's concentration and immerse oneself back into work, and this is when the real situation starts to become clear.

Over a period of 13 minutes, the employee spends 5 minutes on an unexpected and often unprofitable activity, which represents 40% of their time.

We understand better why some people move forward so quickly, while others suffer from these mishaps.

Simply by managing our time and limiting interruptions, we can save up to 40% of our productivity.

Based on a 40-hour workweek, this amounts to 752 hours per year, or 94 days of work!

Chapter II

General instructions

Unless explicitly stated otherwise, the following instructions will be valid for all assignments.

- The language used for this project is C.
- It is not necessary to code according to the 42 norm.
- The exercises are ordered very precisely from the simplest to the most complex. Under no circumstances will we consider or evaluate a complex exercise if a simpler one is not perfectly successful.
- You must not leave any files other than those explicitly specified by the exercise instructions in your directory during peer evaluation.
- All technical answers to your questions can be found in the `datasheets` or on the Internet. It is up to you to use and abuse these resources to understand how to complete your exercise.
- You must use the datasheet of the microcontroller provided to you and comment on the important parts of your program by indicating where you found the clues in the document, and if necessary, explaining your approach. Don't write long blocks of text, keep it clear.
- Do you have a question? Ask your neighbor to the right or left. You can ask in the dedicated channel on the Piscine's Discord, or as a last resort, ask a staff member.

Chapter III

ISR

	Exercise 00
External Interrupt	
Turn-in directory : $ex00/$	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- You are required to write a program that changes the state of LED D1 when the button SW1 is pressed.
- You must use interrupts to read the state of the button. Reading the PINx registers is not allowed.

	Exercise 01
Timer0 interrupt and PWM	
Turn-in directory : $ex01/$	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- You must configure `Timer0` to trigger a periodic interrupt that varies the duty cycle of the LED PB1 controlled by `Timer1`.
- The frequency of `Timer1` must be high enough to no longer see the LED blinking.
- Do not hesitate to use multiple Timer registers to complete this exercise.
- The duty cycle should vary in a loop from 0% to 100% and then from 100% to 0% in 1 second.

Chapter IV

Bonus: Multiplex!

	Exercise 02
Binary counter	
Turn-in directory : *ex*02/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- You must write a program that:
  - each time you press the SW1 button increments a value
  - each time you press the SW2 button decrements a value
  - and displays this value permanently on LEDs D1 D2 D3 and D4 in binary.
- You must use interrupts and have nothing in your main loop.

`If all 4 LEDs were on GPIO PB0, PB1, PB2, PB3, this exercise would be easier.`
`Unfortunately LED D4 is on PB4 instead of PB3 because PB3 is used for something else.`
`You will have to manipulate bits.`