Exercise session 3

Thomas Braipson

17 October 2025

Plan

- Today: programming with interrupts
- Next week: lab 2

Exercise 1: Two blinking LEDs

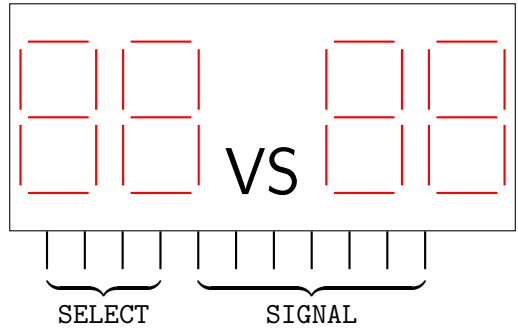
A circuit is composed of two LEDs and a microcontroller equipped with a timer with tunable frequency. The frequency of this timer can be set to

- 24 kHz,
- 8 kHz,
- 6 kHz, or
- 1 kHz.

LEDs should respectively blink at 8 kHz and 3 kHz. How can we program a solution for this, without any busywaiting, if we assume that the instruction clock frequency of the MCU is 1 MHz and that the timer sends an interrupt request at each tick?

Exercise 2: Football score display

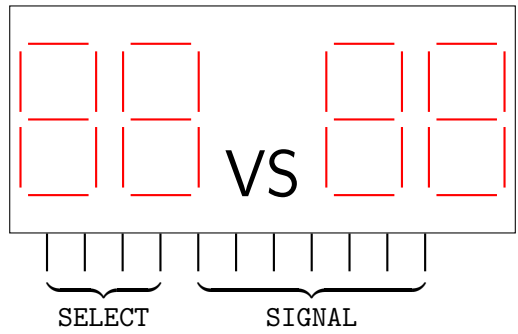
A screen equipped with four seven-segment displays should show the score of the game. It works as follows: each display is lit at once by selecting the appropriate `SELECT` pin and turing on the desired segments via the `SIGNAL` pins. This should be done often enough so that the naked eye does not see that only one display is lit at once.



Exercise 2a: Converting numbers to segments

For the following exercises, we will assume that an 8-bit MCU performs computations.

The score of each team is a number between 0 and 99. Therefore, this number can be internally stored as one byte. Yet, the display takes as input the signal to be applied to each segment. How can we convert in constant time the 8-bit value into the 2 times 7 bits used to display it ?



Exercise 2b: A first program

A team may score a goal at any time (we assume that a delay under 500 ms between two goals cannot happen). Per team, one MCU pin is connected to an external signal responsible for telling that one extra goal has just been scored by the corresponding team. This signal will trigger an interrupt request. The delay between information reception and display has to be as short as possible. It is not permitted to display a number that is neither the current nor the previous score for each team.

The following code shows an implementation. What is wrong with it (two things)? How can we circumvent this?

```
char score_t_1, score_t_2 = 0;
void interrupt update_scores(void)
{
  if(team_1_has_scored){
    team_1_has_scored = 0;
    score_t_1 ++;
  }
  if(team_2_has_scored){
    team_2_has_scored = 0;
    score_t_2 ++;
  }
}
int main() {
  while(1){
    display(1, score_1);
    display(2, score_1);
    display(3, score_2);
    display(4, score_2);
  }
}
```

Exercise 2c: A second program

Here is another implementation. What are the issues now ?

```
int main() {
  volatile char score_t_1, score_t_2 = 0;
  while(1){
    if(team_1_has_scored){
      team_1_has_scored = 0;
      score_t_1 ++;
    }
    if(team_2_has_scored){
      team_2_has_scored = 0;
      score_t_2 ++;
    }
    display(1, score_1);
    display(2, score_1);
    display(3, score_2);
    display(4, score_2);
  }
}
```

Exercise 2d: A third program[1]

The MCU is equipped with a timer able to send an interrupt request. How can we use this to have a good program?

With this solution, is it possible to perform other computations? If yes, what are the conditions on them?

[1] I let you think about this and will not send a correction. You are welcome to email me with your solution to get some feedback.