Embedded system: lab 2

Thomas Braipson

October 2025

1 Measuring the frequency of the clock

In this first step, we will measure the frequency at which instructions are executed on the microcontroller. For this, we have a very simple program that repeatedly sets a pin to a high voltage, then immediately sets it to low and then starts again. Therefore, the time during which the pin has a high voltage is the period at which instructions are executed. For this, download `lab2-1.asm`, compile it, and then flash the code on the MCU as for lab 1. With an oscilloscope, measure the signal on RA0. Remeber that an oscillocsope is a device that measures voltage as a function of time. It has two channels that simultaneously measure the voltage difference with respect to a **common** reference. This reference is the mass of the device (*i.e.* the voltage of aff metallic parts that can be touched from the outside). This mass is connected to the earth through the wall socket's earth pin. Therefore:

- In your circuit, make sure that the two black terminals of the probes are connected to the same voltage;
- If your circuit is connected to earth, make sure that the two black terminals of the probes are connected to that voltage.

If these precautions are not taken, a short circuit will occur inside the oscilloscope and may permanently dammage it.

From middle to middle, what is the time duration of a high pulse ? Knowing that the value of the `OSCCON` register is set to `0xF8`, does this make sense ? (hint check the datasheet).

2 Blinking an LED without busy waiting

Download `lab2-2.asm`. This program uses `TIMER0` to trigger an interrupt responsible for changing the state of pin RA0 at a fixed pace. Observe the signal with an oscilloscope. Is the frequency consistent with the fact that the prescaler of `TIMER0` is set to 256?

3 Measuring the delay of an interrupt request

Now, we will measure the minimum delay between an interrupt request and its service. For this we use a program (`lab2-3.asm`) that never disables interrupts (except during the execution of the interrupt itself). This program reacts to a low-to-high transition on the RA1 pin by setting RA0 pin to high and then immediately to low.

To generate the low-to-high transition on RA1, we use a waveform generator. Before connecting it to the circuit, it has to be configured to output a signal that will not dammage the MCU. For this, connect the output (MAIN OUT 50 Ω) to the oscilloscope with a coax-coax cable. Select a frequency range of 3 MHz and a square waveform, then tune amplitude and offset to obtain a signal ranging from 0V to 5V. Once this is done, connect this signal to the circuit and measure the delay between rising edges on input and ouptut signals.

What delay do you measure ? Is it consistent with the datasheet ?

4 Using a button

4.1 Measuring the signal

Now we will use a button. The first step of this lab is to measure the raw signal produced by such a button. For this, connect a 10 kΩ resistor from +5V to a pin of the button (this is a pull-up resistor) and a wire from the

diagonally opposed pin to ground. Connect an oscilloscope across ground and the resistor leg close to the button. With this setup, when the button is not pressed, the signal read is high and when it is pressed the signal is low. Once your circuit is done, measure the signal (hint, use single mode with negative edge on the trigger). Is this signal clean ?

A first solution to obtain a nicer signal is to build a hardware low-pass filter. This filter is shown in Figure 1. What do you measure now ?

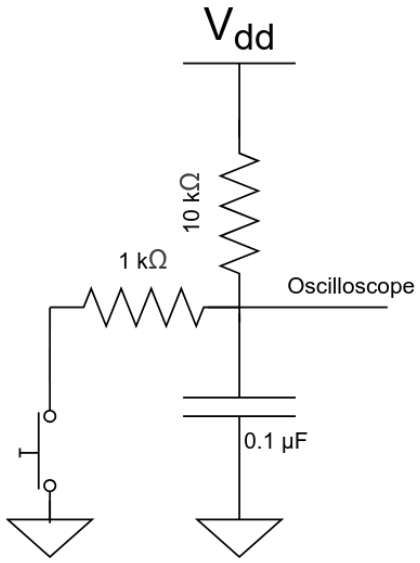

Figure 1: Hardware low-pass filter

4.2 Your turn!

Since the main message of this course is to move problems from hardware to software, you will now program a solution that only uses a button and a few wires and that is robust to bouncing signals.

A few tips:

- Internal pull-ups are available on digital input pins
- Timers can serve a slow clock without busy-waiting
- Be creative!