Embedded Systems
Lab 1 - Introduction to embedded circuit realisation

A laptop with a working installation of MPLABX IDE is required.

1 Introduction

In this lab, you will learn how to program a microcontroller in assembly. At the end of this lab, you should be able to quickly find the information you want in a datasheet. Due to to the curent lockdown, we had to adapt our labs modalities. All the parts requiring some specific hardware have been removed. The lab sessions will therefore consists in a series of assembly code exercises on the Microchip IDE, MPLABX.

Lab sessions are mandatory. Since we cannot organize presential lab sessions, we'll ask you to do them remotly and send a report (one for each group) containing your answers to the questions we ask in this document. Please, keep your answers short and to the point. There is no need for unecessary filling.

Your report will be used to evaluate your participation to the lab. The labs will be evaluated for a total of 15% of the final note of the course.

Note 1: For practical reasons, we suggest you to keep the same groups for the labs than the one for the project.
Note 2: You will be asked to send your report before the deadline stated on the exercise sessions website.

2 Preparation

Before asking you to play with code and electrical components, we want you to take some time to document yourself on the microcontroller. For this, we ask you to answer the following questions in your report. To complete this preparation in a decent amount of time, you are encouraged to share it evenly between the different members of your group.

The questions asked hereunder refer to the sections 3, 4, 5, 6, 13, 23 and 30 of the PIC16F1789 datasheet. You are not asked to read them from top to bottom, but rather to understand the structure and the type of information given in each of them in order to be able to pick the information you are looking for efficiently.

1. On components with more than two pins, how can you identify the number assigned to each pin on the physical device?
2. Give a brief description of the three main reset sources of microcontrollers (master clear, brown-out and watchdog timer).
3. To which address of the program memory does a reset bring you back?
4. In the PIC16F1789, which special function registers (SFR) need to be modified to configure the microcontroller to use the internal clock at a given frequency?
5. In the PIC16F1789, which values would you assign to them if you want to use the internal oscillator at a speed of 4MHz? Explain how you chose these values.

6. In the PIC16F1789, which registers would you modify to configure the microcontroller to use RD0 as a digital output set to 1? In which bank are they located?
7. Which task does the assembly code shown below perform? How does it do that (more specifically, explain the role of lines which ask you to do it)?
8. In the initialisation part of the code, replace the question marks accordingly based on the SFR values.
9. At what (exact) frequency will the LED blink? Give your computation details.

```
; ************************************************************ ;
;                        BLINKY                                ;
;      make a LED plugged on the RD0 pin blink at a given ;
;      frequency using Timer1                           ;
;                                                       ;
;      INFO0064 - Embedded Systems - Lab 1              ;
;                                                       ;
; ************************************************************ ;

    processor 16f1789
    #include  "config.inc"
    #define value_counter 0x03

    PSECT text, abs, class=CODE, delta=2

; That is where the MCU will start executing the program (0x00)
   org   00h
   goto    start        ; jump to the beginning of the code

;BEGINNING OF THE PROGRAM
start:
   call  initialisation    ; initialisation routine configuring the MCU
   goto  main_loop         ; main loop

;INITIALISATION
initialisation:
    ; configure clock
    ;configuration of the GPIO
   movlb   01h
   clrf    TRISD              ; All pins of PORTD are output
   movlb   02h
   movlw   00000001B
   movwf   LATD               ; RD0 = 1 while RD1..7 = 0;

    ;configuration of clock - ?(a) frequency - ?(b) source
   movlb 01h
   movlw 01101110B
   movwf OSCCON        ; configure oscillator (cf datasheet SFR OSCCON)
   movlw 00000000B
   movwf OSCTUNE       ; configure oscillator (cf datasheet SFR OSCTUNE)
```

```
    ; Timer1 ON - ?(c)x prescaling - ?(d) clock source
   movlb 00h
   movlw 00010001B
   movwf T1CON            ; configure Timer1 (cf. datasheet SFR T1CON)

    ; Declare counter variable at the first GPR address of a bank
   counter EQU 20h

    ;Store initial counter value in first GPR address of bank 2
   movlb 02h
   movlw value_counter
   movwf counter

   return

;MAIN LOOP
main_loop:

    movlb  00h          ; select bank 0
    btfss  TMR1H, 7   ; explain what is performed here (a)
    goto main_loop

    bcf     T1CON, 0
    clrf   TMR1H
    clrf   TMR1L
    bsf     T1CON, 0

    movlb  02h        ; select bank 2
    decfsz counter, 1   ; explain what is performed here (b)
    goto main_loop

    movlw  value_counter
    movwf  counter    ; reset the value of the counter

    movlw  01h
    xorwf  LATD, 1    ; RD0 = !RD0

    goto   main_loop
```

3 Circuit Assembly

In this part, we will focus on circuit assembly based on an electronic schematic. Due to current conditions, you will unfortunately not be able to do this part of the lab. It indeed requires you to have access to your toolbox and the instruments of the r.100.

We nonetheless decided to keep this section in the statement of the lab so that you can come back to it when (and if) you need it for your project. Feel free to read it and ask questions about it whenever you want during the academic year.

1. Plug the components on the breadboard following the schematic given at the end of this document. You are free to organize the layout as you want. Be sure to respect the polarity of the components which have one and to not make any mistake with the various pin assignments. Don't plug the Pickit3 to your circuit at this stage.

   Note 1: The components which have a polarity are the diodes and the electrolytic capacitors. All those components have a mark which indicates the pin which should be located on the low voltage side.

   Note 2: The pin numbering of the LM7805 is given in its datasheet (link to it on the exercise session website).

   Note 3: The value of a resistor can be known by reading its color code or by measuring it with a multimeter.

   Note 4: C2 and C4 are decoupling capacitors. Their role is to stabilize the voltage locally, at the input of the microcontroller. Those capacitors should therefore be plugged as close as possible to each power input pin of the microcontroller on the breadboard.

2. Verify your circuit assembly and plug the power supply once you are sure everything is correctly wired.

3. Later in the lab, you will have the opportunity to measure the LED signal. Plug one of your coaxial cables on the the channel 1 of the oscilloscope. Then connect the red probe on the pin 19 of the microcontroller and set the black one to the ground.

4. When the circuit building is finished, turn on the DC power generator without connecting it to the circuit. Then set the output voltage to $9V$ and the current limit (right knob) to the minimum. The current limit setting is a safety feature that will protect your circuit in case of bad connections. It caps the current supplied to the circuit to the chosen limit. This is done by decreasing the supplied voltage when the current exceeds this limit. With the knob set to the minimum, the generator won't provide any power to your circuit.

5. Connect the power generator to the input of your circuit and verify your circuit assembly. If you have any doubt about some of your connections, ask a teaching assistant for help.

6. **Warning**, for this lab, your circuit should never draw more than 0.1A. If you reach this value while executing instructions hereunder, turn immediately the power supply off, you probably have a bad connection in your circuit and face some component damage if you provide more input power. Try to find your mistake and repeat the process explained at this point.

At this step, you will power your circuit. For that, first set the power supply to measure continuous currents (cc) by pressing the measure button. Then, slowly increase the current limit. The display will show you the amount of current drawn by your circuit. At some point, the current limit will exceed the current drawn by your circuit and the voltage delivered to your circuit will stabilize. Stop increasing the current limit at this point. This will protect your circuit in case of bad manipulations later in the lab.

Note 5: You will know that the current limit is reached when the supplied voltage drops (press on the measure button again to read the supplied voltage).

7. By using the multimeter, check that the voltage at important points of your circuit (microcontroller power pins, header for the Pickit3, ...) is the desired one. Once the verification is completed, you can proceed to the next part of the lab.

   Note 6: The procedure explained in the last 4 steps is important for safely powering your circuit. From now on, you are supposed to know it and are expected to apply it every time you power a new or modified circuit.

4 Microcontroller Programming

In this final part, we will show you how to program your microcontroller, and how to use the MPLABX build-in simulator. For this part, you will need your laptop with MPLABX IDE and xc8 compiler installed. You can download them here:

- MPLABX IDE: `https://www.microchip.com/mplab/mplab-x-ide`
- XC8 Compiler: `https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers`

Start the MPLABX IDE. Then open the MPLAB project given in the archive (Lab1.X). Normally, the project should be ready for use out of the box. We however advise you to check two things in the project configuration ("Production>Set Project Configuration>Customize"):

- In the Conf:[default] category, make sure that the pic-as compiler toolchain is selected;
- In the Simultator category, check that the instruction frequency is set to 1MHz.

To program your microcontroller, the procedure is the following:

1. Make sure that your circuit is powered.
2. Once this is done, you are ready to program the microcontroller. Plug the PICkit3 on the header of your circuit (be careful, the location of pin 1 is marked by on arrow on the device), connect it to your computer and click on the big green arrow "Run Main Project". If your are asked to choose your programming tool, select the PICkit3 appearing in the list.
3. If the programming was successful, you should now have a blinking LED.

If you had access to the r.100 lab, you could measure the frequency at which the LED blinks with the oscilloscope and confront your measurement to the computations you made. However this won't be possible this year. Fortunately, there is another way to analyze what is happening on your microcontroller and that is the MPALBX simulator.

The purpose of this tool is to simulate the execution of your code in real-time in the IDE as it would be executed on the microcontroller. This simulator features several useful modules that help visualizing what's happening in the microcontroller. We will cover some of these features during current and following lab sessions.

As a first approach to the simulator, we will ask you to analyse the signal of the LED.

1. To run the simulator, you need to unconnect the PICkit3 from your computer (if it was plugged). Once this is done, you can launch the simulator by starting a project debugging ("debug" > "Debug Main Project"). A pop-up window will appear asking the device you want to use. Choose the simulator entry.

2. The simulator will start and you will have access to the simulation and debugging pannels ("Window> Debugging or Simulator"). In this lab, we will use the stopwatch (in the debugging tools). The stopwatch measures the time required from one breakpoint to the next one. It is especially usefull to verify timings in your code.

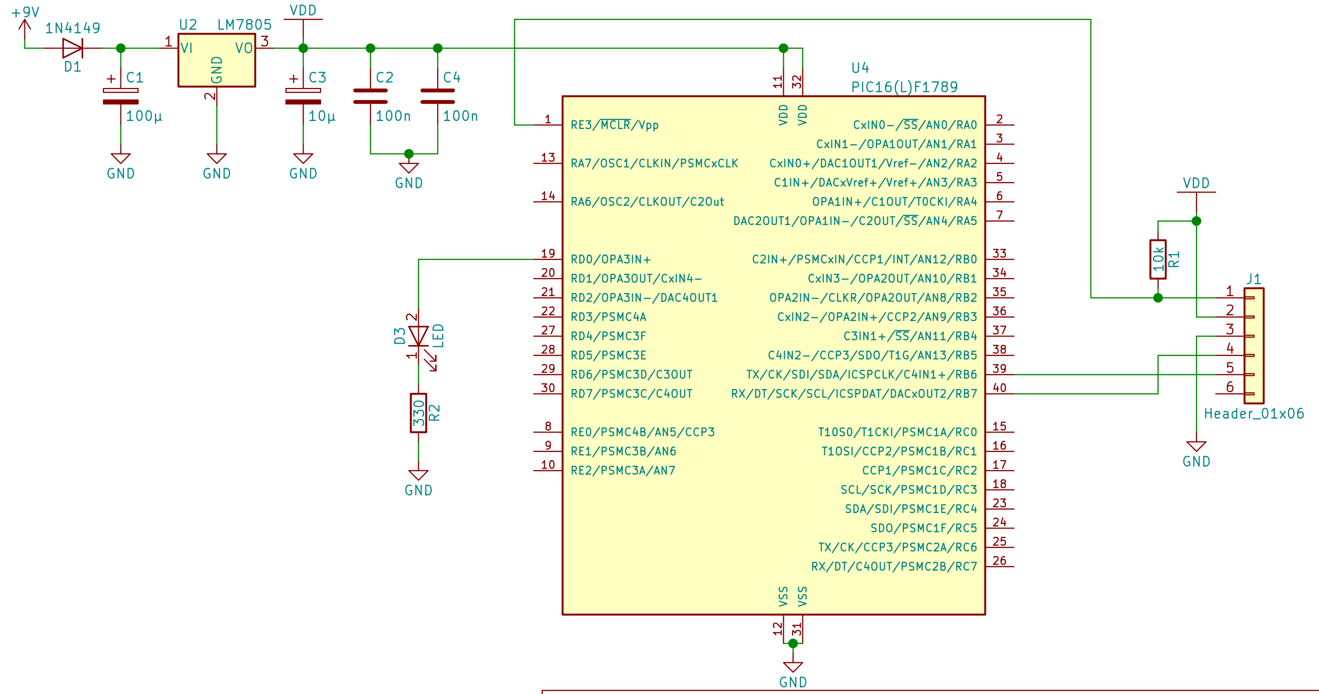
3. For the stopwatch to work, you need to define a breakpoint. Set it on line 77 of your code (xorwf LATD,1). It is the line where the state of the LED is changed.

Now, answer the following questions:

1. How much time does your code spend between two breakpoints halts? Does it match with the value you computed in the last question of section 1?

2. What would you do to divide the blinking period of the led by a factor 2?

3. Implement your modification idea in the code and measure the new delay between two executions of line 77. How much is it? Is it exactly equal to the half of your previous measurement? If not, what's your explanation for the difference you observe?



ULiège

Sheet: /

File: labos.sch

Title: Lab 1

Size: User	Date:	**Rev: 1.0**
KiCad E.D.A.	kicad 4.0.7−e2−6376~58ubuntu16.04.1	Id: 1/1