Embedded Systems

Lab 1 - Introduction to embedded circuit realisation

You are asked to prepare the first part before the lab. Lab duration: 45min

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

1 Introduction

In this lab, you will learn how to build an electronic circuit based on a schematic and how to program a microcontroller in assembly. At the end of this lab, you should be able to correctly identify and use each electronic component based on its symbol and to quickly find the information you want in a datasheet.

Note 1: Lab sessions are mandatory. Your presence, preparation and participation to the labs will be evaluated for a total of 15% of the final note of the course.

Note 2: For practical reasons, we suggest you to keep the same groups for the labs than the one for the project.

Note 3: You will be asked to give a copy of your preparation at the beginning of the lab.

2 Preparation

The time available to complete the lab being quite limited, your are asked to have prepared this section before attending it. 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 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 of 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?
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 which (exact) frequency will the LED blink?

```
; ********************************************************** ;
;                        BLINKY                              ;
;      make a LED plugged on the RD0 pin blink at a given    ;
;      frequency using Timer1                                ;
;                                                            ;
;      INF00064 - Embedded Systems - Lab 1                   ;
;                                                            ;
; ********************************************************** ;

    processor 16f1789
    #include  "config.inc"
    #DEFINE   value_counter 0x04

; That is where the MCU will start executing the program (0x00)
   org   0x00
   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   0x01
   clrf    TRISD             ; All pins of PORTD are output
   movlb   0x02
   movlw   b'00000001'
   movwf   LATD              ; RD0 = 1 while RD1..7 = 0;

    ;configuration of clock - ?(a) frequency - ?(b) source
   movlb 0x01
   movlw b'01101110'
   movwf OSCCON        ; configure oscillator (cf datasheet SFR OSCCON)
   movlw b'00000000'
   movwf OSCTUNE       ; configure oscillator (cf datasheet SFR OSCTUNE)

    ; Timer1 ON - ?(c)x prescaling - ?(d) clock source
   movlb 0x00
   movlw b'00110001'
   movwf T1CON             ; configure Timer1 (cf. datasheet SFR T1CON)

    ; Map the first GPR address of a bank in ram (address 20h) to the name 'counter'
   cblock  0x20
   counter
   endc

    ;set the counter value in bank2 (at address 20h)
   movlb 0x02
```

```
   movlw value_counter
   movwf counter

   return

;MAIN LOOP
main_loop:

    movlb  0x00    ; 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  0x02    ; 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  0x01
    xorwf  LATD, 1    ; RD0 = !RD0

    goto   main_loop
    END
```

3 Circuit Assembly

In this part, we will focus on circuit assembly based on an electronic schematic. For this, you will need to use your toolbox and the instruments of the lab.

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. Later in the lab, you will be asked 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.

3. 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 which 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.

4. 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.

5. **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 current drawn by 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).

6. 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 create a MPLABX project and how to program your microcontroller. For this part, you will need your laptop with a ready-to-use MPLABX IDE installation.

1. Start the MPLABX IDE. Then click on File>New Project. Choose standalone project in the new window and click next. Now select the PIC16F1789 in the list and then PICkit3 in the next form. After that, choose the mpasm compiler toolchain. Finalize your project creation by completing the next form (project name and location) as you want.

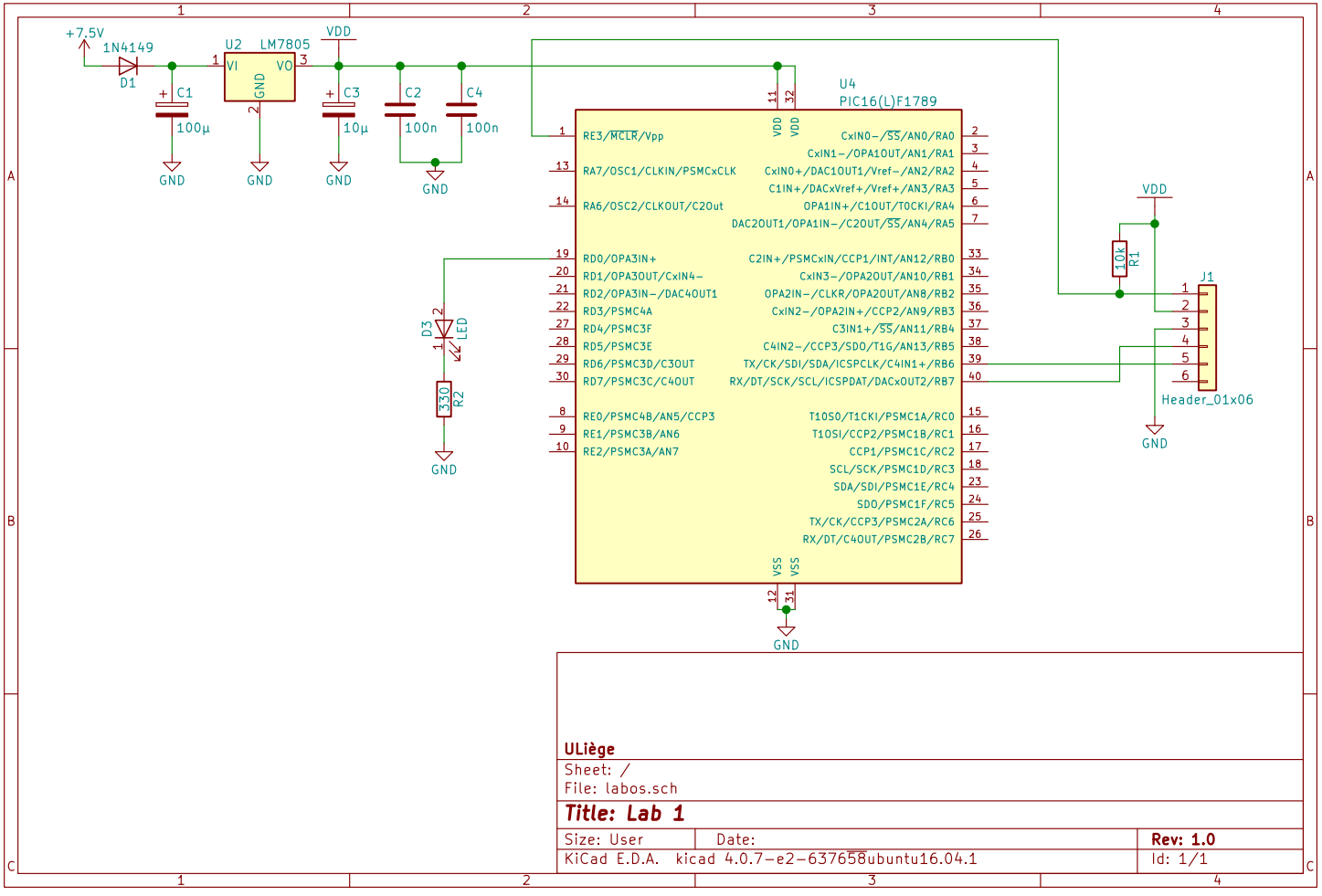
2. Add the *blinky.asm* and $config.inc$ files given in the archive to your "Source Files" folder and the $16f1789\_g.lkr$ to your "Linker Files" folder.

3. At this stage, make sure that your circuit is powered.

4. Once this is done, you are ready to program the microcontroller. Plug the PICkit3 on the header of your circuit (be careful to the location of pin 1), 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.

5. If the programming was successful, you should now have a blinking LED.

6. Measure the frequency at which the LED blinks with the oscilloscope. What do you measure? Does it correspond to what you computed during the preparation? If not, try to understand why.



ULiège

Sheet: /

File: labos.sch

Title: Lab 1

Size: User	Date:	Rev: 1.0
KiCad E.D.A. kicad 4.0.7-e2-637658ubuntu16.04.1		Id: 1/1