

# Practice 1. Introduction to MPLAB X IDE

José Manuel Aguilar Zamudio, Renato Arcos Guzmán, Diego Arnoldo Azuela Rosas  
Equipo 2

## Objectives—

The student will become familiar with the MPLAB integrated development environment tools. The student will create his (her) first program using the MPLAB software and will put it into the PIC microcontroller making use of the Curiosity development/evaluation board.

*Index Terms*—Curiosity board, PIC18, microcontrollers.

## I. INTRODUCTION

MICROCONTROLLERS are all around the world. Each day, Microcontrollers, are more present in the many aspects of our lives: in our work, inside our houses, and in more. We can find them controlling small devices like cellphones, microwaves, washing machines, and televisions [1].

A microcontroller is one device or chip that is used to govern one or more processes. For example, the controller that regulates the room temperature of an air conditioner; it has a sensor that continuously measures the internal temperature and, when the preset limits are exceeded, it generates the necessary signals to adjust the temperature.

## II. STATE OF THE ART

### A. The practices and the PIC microcontroller

The main objective of this practices is to provide students the foundation to fully understand the operation of the PIC18F45K50 microcontroller [2]. This will be achieved through 11 documents that will guide the reader to create their own electronic card or Printed Circuit Board (PCB) and to be able to program it; in order to, execute different functions.

The advantages of ta PIC microcontroller to others on the market, which is why it will be used throughout this manual, are as follows:

- Easy to operate.

This paragraph of the first footnote will contain the date on which you submitted your report for review.

The next few paragraphs contains the authors' current affiliations, including current address and e-mail.

Ph. D. Raúl Peña Ortega is with Tecnológico de Monterrey, as a Professor (e-mail: raul.p.ortega@tec.mx).

M.C. Salvador Rodríguez López is with Tecnológico de Monterrey, as a Professor (e-mail: salvadorrdzlop@tec.mx).

- There is enough documentation to work with it and it's easy to obtain it.
- The price is comparatively lower than its competitors.
- It has a high operating speed.
- Development tools are cheap and easy to use.
- There are a variety of hardware that can record, erase and check the behavior of PIC.
- Once you learn to handle a PIC, it will easier handle any other models of microcontrollers.

### B. MPLAB X

MPLAB X Integrated Development Environment (IDE) is an expandable, highly configurable software program that incorporates powerful tools to help you discover, configure, develop, debug and qualify embedded designs for most of Microchip's microcontrollers and digital signal controllers. MPLAB X IDE works seamlessly with the MPLAB development ecosystem of software and tools, many of which are completely free. [3].

### C. CURIOSITY development board

The Curiosity HPC Development Board is a 8-bit prototyping board. It is designed from the ground-up to take full advantage of Microchip's MPLAB® X integrated development environment. The Curiosity Development Board supports Microchip's 28- and 40-pin 8-bit PIC MCUs. Programming/debugging is accomplished through the PICKit On Board (PKOB), eliminating the need for an external programming/debugging tool.

## III. RESULTS

In this section, you must report the outcomes of the laboratory activities.

### A. Install the MPLAB IDE

1. Go to <https://www.microchip.com/mplab/mplab-x-ide>, scroll down to "Downloads" section and download the 'MPLAB X IDE' software, then install it on your computer.

### B. Create, Compile and Debug your First Program

2. Connect the Curiosity board to your PC.
3. Start the MPLABX IDE program. Go to File->New Project and select the "Microchip Embedded" and "Standalone Project" options. Then Click on "Next".

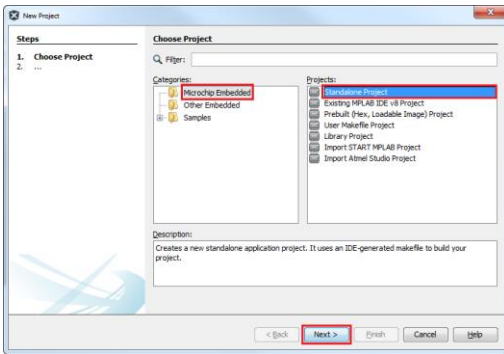


Fig. 1. Choose Project.

4. Select the PIC18F45K50 device and Click on “Next”.

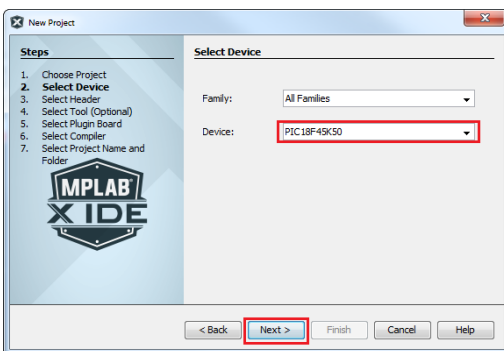


Fig. 2. Select Device.

5. Choose the “Curiosity” option and Click on “Next”

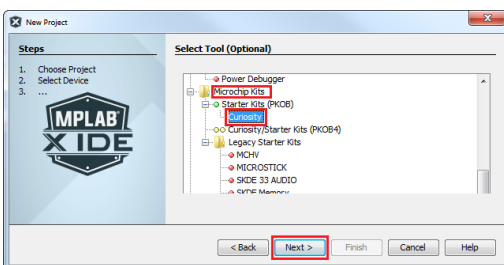


Fig. 3. Select tool.

6. Choose the XC8 Compiler Toolchain to build the program and Click on “Next”.

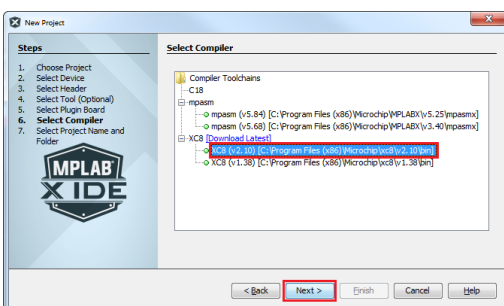


Fig. 4. Select Compiler.

7. Select the “Project Location” and “Project Name”. Make sure that the location and the name have no special characters and no spaces in the names of the directories or filenames. Then Click on “Finish”.

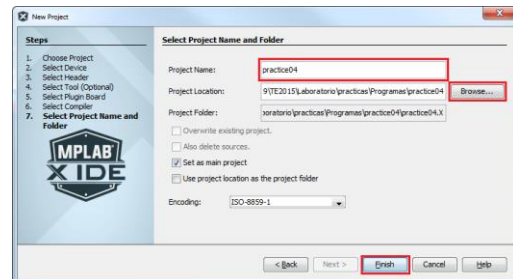


Fig. 5. Select Project Name and Folder.

8. Right-Click on “Source Files” and open a new “main.c” file. and Click on “Finish”.

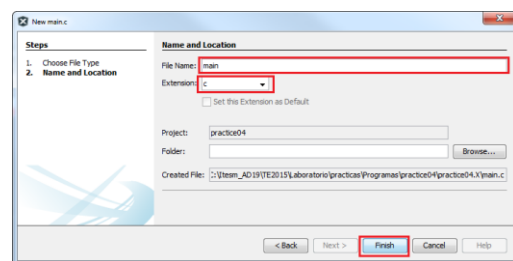
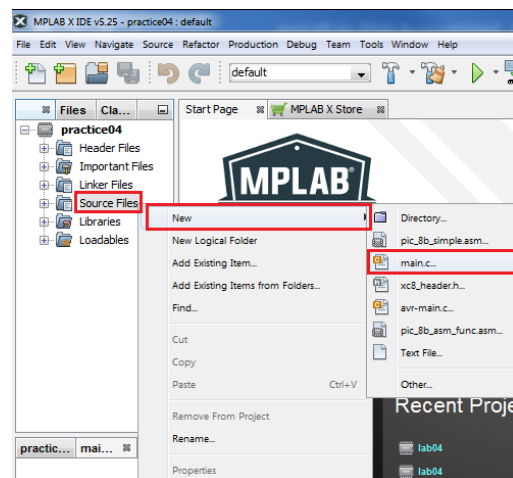


Fig. 6. Create Source file.

9. Your code must be like the one showed in the figure 16, appended at the end of the document.
10. Download ‘device\_config.h’ header file from Blackboard. Right-Click on “Header Files” and Click on “Add Existing Item”. Choose the downloaded file and Click on Select.

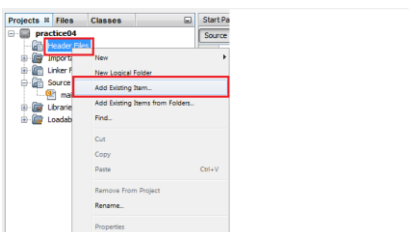


Fig. 7. Add Header file.

11. Select the Tools and click on “Options”.

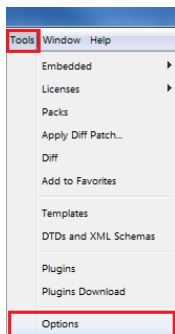


Fig. 8. Tools options.

12. Click on “Embedded ” and select following options: “Reset vector”, “Halt at Reset vector”

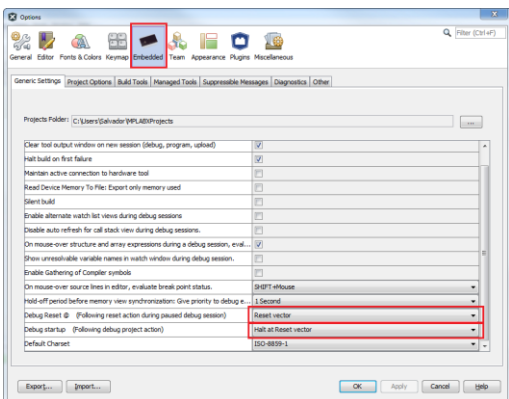


Fig. 9. Tools options

13. Go to “File” and Click on “Project properties”.

14. Ensure that the “Low voltage programming mode” option is checked.

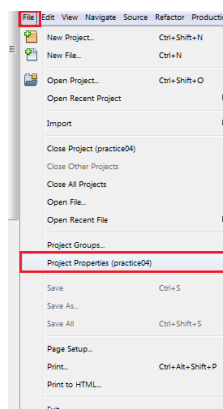


Fig. 10. Project Properties.

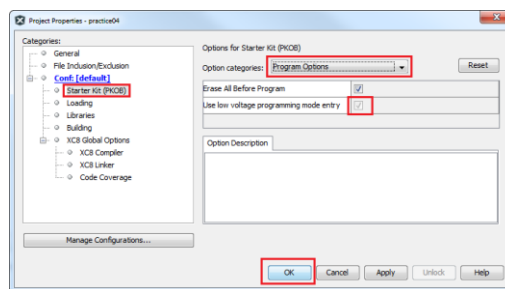


Fig. 11. Program options.

15. Select “Build for Debugging Main Project”.

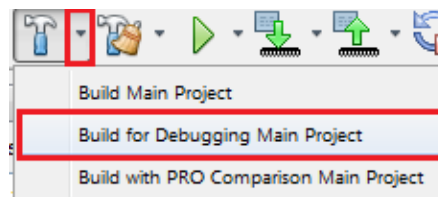


Fig. 12. Program options.

16. Select “Debug Main Project”.

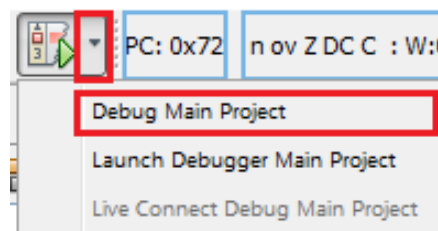


Fig. 13. Debug Main Project.

17. You can use the debug buttons to go step by step from beginning to end through the program.

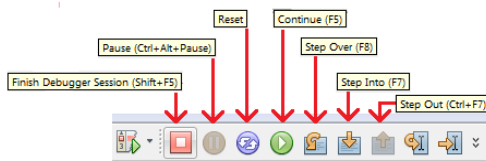


Fig. 14. Debug buttons.

18. Before start, activate the windows that let us to see what is happening inside the microcontroller. On this window you can select what you want to verify, for example general purpose file registers (GPRs), Special file registers (SFRs), etc.

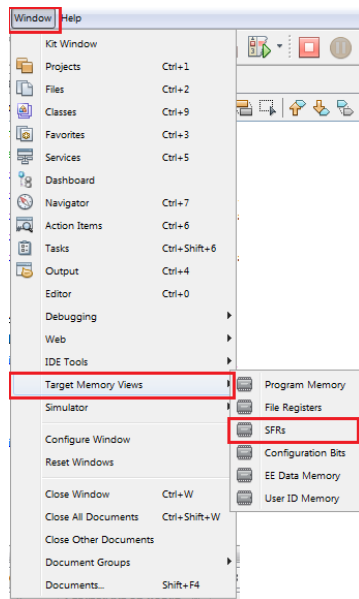


Fig. 15. Memory view.

19. Now debug the program by executing the code step-by-step.

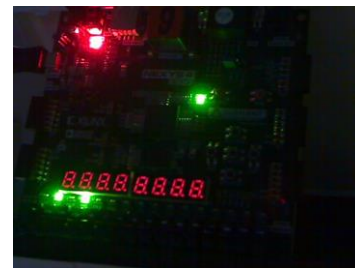
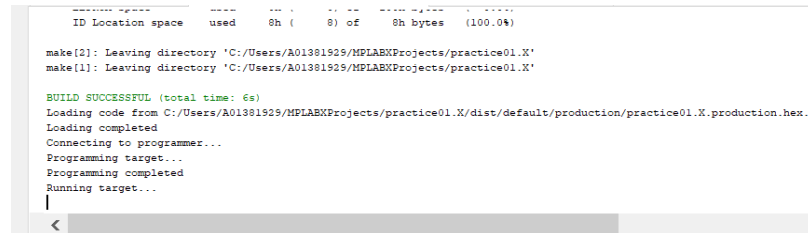
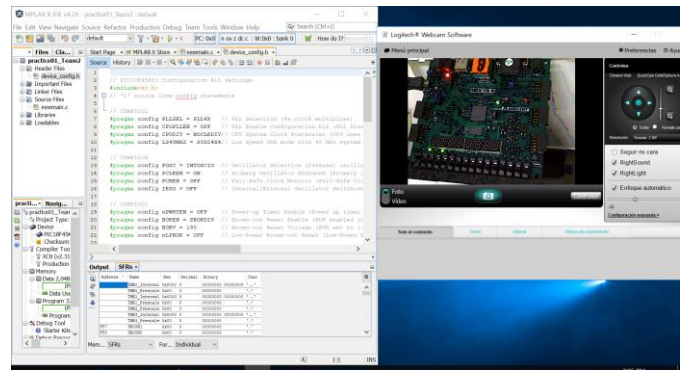
1) **Report:** For this practice, add to the Section III the following:

- image of the BUILD SUCCESSFUL window.
- image of each modified register for the portInit() function when you debug the code step by step. **TIP:** Put a breakpoint at the first line of the portInit() function and open the IO window (*Window* → *Debugging* → *IO View*).

Also, modify the Section IV and your names at the top of the document. Upload the modified document to Blackboard.

2) **File Uploads:** Upload to Blackboard your 'main.c.' The code must be commented line by line.

3) **Demonstration:** Record a video of the practice's functionality and upload it to Facebook. The video must contain a short explanation.

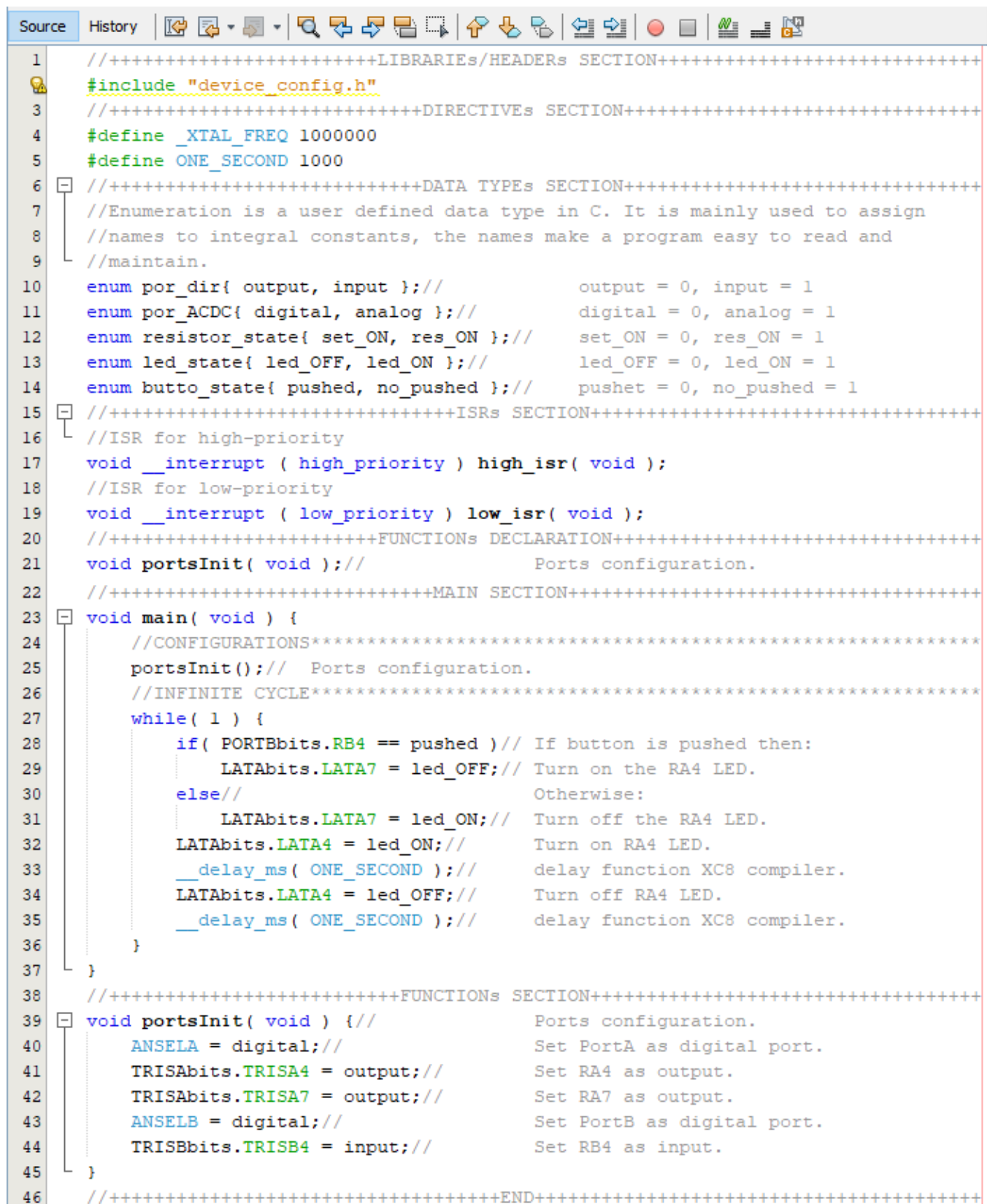


#### IV. CONCLUSION

Esta practica se nos hizo muy interesante; especialmente el concepto de controlar un microcontrolador de manera remota. El uso de las cámaras en conjunto con el escritorio remoto se me hizo muy interesante. Este tipo de tecnologías permite que las opciones para trabajar no tengan limite.

#### REFERENCES

- [1] H.-W. Huang, "PIC Microcontroller: An Introduction to Software and Hardware Interfacing," New York, USA, 1st ed., Thomson Learning Inc Inc., 2005.
- [2] Microchip, "PIC18(L)F2X/45K50 datasheet," USA, Microchip Technology Inc, 2012.
- [3] "MPLAB X IDE" <https://www.microchip.com/mplab/mplab-x-ide>



```

1 //+++++LIBRARIES/HEADERS SECTION+++++
2 #include "device_config.h"
3 //+++++DIRECTIVES SECTION+++++
4 #define _XTAL_FREQ 1000000
5 #define ONE_SECOND 1000
6 //+++++DATA TYPES SECTION+++++
7 //Enumeration is a user defined data type in C. It is mainly used to assign
8 //names to integral constants, the names make a program easy to read and
9 //maintain.
10 enum por_dir{ output, input };//          output = 0, input = 1
11 enum por_ACDC{ digital, analog };//        digital = 0, analog = 1
12 enum resistor_state{ set_ON, res_ON };//    set_ON = 0, res_ON = 1
13 enum led_state{ led_OFF, led_ON };//        led_OFF = 0, led_ON = 1
14 enum butto_state{ pushed, no_pushed };//     pushet = 0, no_pushed = 1
15 //+++++ISRs SECTION+++++
16 //ISR for high-priority
17 void __interrupt ( high_priority ) high_isr( void );
18 //ISR for low-priority
19 void __interrupt ( low_priority ) low_isr( void );
20 //+++++FUNCTIONS DECLARATION+++++
21 void portsInit( void );//                  Ports configuration.
22 //+++++MAIN SECTION+++++
23 void main( void ) {
24     //CONFIGURATIONS*****
25     portsInit();// Ports configuration.
26     //INFINITE CYCLE*****
27     while( 1 ) {
28         if( PORTBbits.RB4 == pushed )// If button is pushed then:
29             LATAbits.LATA7 = led_OFF;// Turn on the RA4 LED.
30         else// Otherwise:
31             LATAbits.LATA7 = led_ON;// Turn off the RA4 LED.
32             LATAbits.LATA4 = led_ON;// Turn on RA4 LED.
33             __delay_ms( ONE_SECOND );// delay function XC8 compiler.
34             LATAbits.LATA4 = led_OFF;// Turn off RA4 LED.
35             __delay_ms( ONE_SECOND );// delay function XC8 compiler.
36     }
37 }
38 //+++++FUNCTIONS SECTION+++++
39 void portsInit( void ) { // Ports configuration.
40     ANSELA = digital;// Set PortA as digital port.
41     TRISAbits.TRISA4 = output;// Set RA4 as output.
42     TRISAbits.TRISA7 = output;// Set RA7 as output.
43     ANSELB = digital;// Set PortB as digital port.
44     TRISBbits.TRISB4 = input;// Set RB4 as input.
45 }
46 //+++++END+++++

```

Fig. 16. Create *main.c* source file.

```

#include "DEVICE_CONFIG.H"

//DIRECTIVE

#define _XTAL_FREQ 1000000

#define ONE_SECOND 1000

//DATA TYPE

ENUM POR_DIR { OUTPUT, INPUT }; // OUTPUT=0, INPUT=1

ENUM POR_ACDC { DIGITAL, ANALOG }; // DIGITAL=0, ANALOG=1

ENUM RESISTOR_STATE { SET_ON, RES_ON }; // SET_ON=0, RES_ON=1

ENUM LED_STATE { LED_OFF, LED_ON }; // LED_OFF, LED_ON=1

ENUM BUTTO_STATE { PUSHED, NO_PUSHED }; // PUSHED =0, NO_PUSHED=1

//ISR FOR HIGH-PRIORITY

VOID __INTERRUPT(HIGH_PRIORITY) HIGH_ISR (VOID);

//ISR FOR LOW-PRIORITY

VOID __INTERRUPT (LOW_PRIORITY) LOW_ISR (VOID);

//FUNCTIONS DECLARATION

VOID PORTSINIT(VOID);

//MAIN SECTION

VOID MAIN(VOID) {

    //CONFIGURATIONS

    WHILE(1){

        IF (PORTBbits.RB4 ==PUSHED )//IF BUTTON PUSHED THEN:

            LATAbits.LATA7 =LED_OFF;//TURN ON THE RA4 LED

        ELSE// OTHERWISE:

            LATAbits.LATA4 =LED_ON;

            __DELAY_MS (ONE_SECOND); //DELAY FUNCION
XC8 COMPILER

            LATAbits.LATA4= LED_OFF;

            __DELAY_MS (ONE_SECOND); //DELAY FUNCTION
XC8 COMPILER
    }
}

```