**INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY**

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente**

**Microcontrollers Laboratory**

**Practice 04**

**Names:**

**Marcos Rubén Flores Lara A01351360**

**Edgar Alejandro Rodríguez Gallegos A00822028**

**Date: June 2, 2022**

# Report 04 – Introduction to MPLAB X IDE

Practice 04

ITESM Campus Mty

Marcos Rubén Flores Lara

A01351360

Edgar Alejandro Rodríguez Gallegos A00822028

Horario: 14:30-16:00

**horizontal line**

Instructor: [Matias Vázquez Piñón](https://experiencia21.tec.mx/courses/261838/users/136680)

Due Date: 06/02/2022

**PROCEDURE:**

First we added the device configuration which was given to us by the github repository, then we created and followed the practice steps until we finally were able to simulate the code.

**DEVICE CONFIG**

// PIC18F45K50 Configuration Bit Settings

#include<xc.h>

// 'C' source line config statements

// CONFIG1L

#pragma config PLLSEL = PLL4X // PLL Selection (4x clock multiplier)

#pragma config CFGPLLEN = OFF // PLL Enable Configuration bit (PLL Disabled (firmware controlled))

#pragma config CPUDIV = NOCLKDIV// CPU System Clock Postscaler (CPU uses system clock (no divide))

#pragma config LS48MHZ = SYS24X4// Low Speed USB mode with 48 MHz system clock (System clock at 24 MHz, USB clock divider is set to 4)

// CONFIG1H

#pragma config FOSC = INTOSCIO // Oscillator Selection (Internal oscillator)

#pragma config PCLKEN = ON // Primary Oscillator Shutdown (Primary oscillator enabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config nPWRTEN = OFF // Power-up Timer Enable (Power up timer disabled)

#pragma config BOREN = SBORDIS // Brown-out Reset Enable (BOR enabled in hardware (SBOREN is ignored))

#pragma config BORV = 190 // Brown-out Reset Voltage (BOR set to 1.9V nominal)

#pragma config nLPBOR = OFF // Low-Power Brown-out Reset (Low-Power Brown-out Reset disabled)

// CONFIG2H

#pragma config WDTEN = OFF // Watchdog Timer Enable bits (WDT disabled in hardware (SWDTEN ignored))

#pragma config WDTPS = 32768 // Watchdog Timer Postscaler (1:32768)

// CONFIG3H

#pragma config CCP2MX = RC1 // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)

#pragma config PBADEN = ON // PORTB A/D Enable bit (PORTB<5:0> pins are configured as analog input channels on Reset)

#pragma config T3CMX = RC0 // Timer3 Clock Input MUX bit (T3CKI function is on RC0)

#pragma config SDOMX = RB3 // SDO Output MUX bit (SDO function is on RB3)

#pragma config MCLRE = ON // Master Clear Reset Pin Enable (MCLR pin enabled; RE3 input disabled)

// CONFIG4L

#pragma config STVREN = ON // Stack Full/Underflow Reset (Stack full/underflow will cause Reset)

#pragma config LVP = ON // Single-Supply ICSP Enable bit (Single-Supply ICSP enabled if MCLRE is also 1)

#pragma config ICPRT = OFF // Dedicated In-Circuit Debug/Programming Port Enable (ICPORT disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled)

// CONFIG5L

#pragma config CP0 = OFF // Block 0 Code Protect (Block 0 is not code-protected)

#pragma config CP1 = OFF // Block 1 Code Protect (Block 1 is not code-protected)

#pragma config CP2 = OFF // Block 2 Code Protect (Block 2 is not code-protected)

#pragma config CP3 = OFF // Block 3 Code Protect (Block 3 is not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protect (Boot block is not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protect (Data EEPROM is not code-protected)

// CONFIG6L

#pragma config WRT0 = OFF // Block 0 Write Protect (Block 0 (0800-1FFFh) is not write-protected)

#pragma config WRT1 = OFF // Block 1 Write Protect (Block 1 (2000-3FFFh) is not write-protected)

#pragma config WRT2 = OFF // Block 2 Write Protect (Block 2 (04000-5FFFh) is not write-protected)

#pragma config WRT3 = OFF // Block 3 Write Protect (Block 3 (06000-7FFFh) is not write-protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Registers Write Protect (Configuration registers (300000-3000FFh) are not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protect (Boot block (0000-7FFh) is not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protect (Data EEPROM is not write-protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Block 0 Table Read Protect (Block 0 is not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Block 1 Table Read Protect (Block 1 is not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Block 2 Table Read Protect (Block 2 is not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Block 3 Table Read Protect (Block 3 is not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protect (Boot block is not protected from table reads executed in other blocks)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

**MAIN C**

////+++++++++++++++++++++++++++++++++++++| LIBRARIES / HEADERS |+++++++++++++++++++++++++++++++++++++

#include "device\_config.h"

//+++++++++++++++++++++++++++++++++++++| DIRECTIVES |+++++++++++++++++++++++++++++++++++++

#define \_XTAL\_FREQ 1000000

#define ONE\_SECOND 1000

//+++++++++++++++++++++++++++++++++++++| DATA TYPES |+++++++++++++++++++++++++++++++++++++

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 = 0, led\_ON = 1

enum butto\_state { pushed, no\_pushed }; // pushed = 0, no\_pushed = 1

//+++++++++++++++++++++++++++++++++++++| ISRs |+++++++++++++++++++++++++++++++++++++

// ISR for high priority

void \_\_interrupt ( high\_priority ) high\_isr( void );

// ISR for low priority

void \_\_interrupt ( low\_priority ) low\_isr( void );

//+++++++++++++++++++++++++++++++++++++| FUNCTION DECLARATIONS |+++++++++++++++++++++++++++++++++++++

void portsInit( void );

//+++++++++++++++++++++++++++++++++++++| MAIN |+++++++++++++++++++++++++++++++++++++

void main( void ){

portsInit();

while(1){

if(PORTBbits.RB4 == pushed) // If button is pushed then

LATAbits.LATA7 = led\_OFF; // turn off RA4 LED

else // Otherwise

LATAbits.LATA7 = led\_ON; // turn on RA4 LED

LATAbits.LATA4 = led\_ON; // Turn on RA4 LED

\_\_delay\_ms(ONE\_SECOND); // Delay function XC8 compiler

LATAbits.LATA4 = led\_OFF; // Turn off RA4 LED

\_\_delay\_ms(ONE\_SECOND); // Delay function XC8 compiler

}

}

//+++++++++++++++++++++++++++++++++++++| FUNCTIONS |+++++++++++++++++++++++++++++++++++++

void portsInit( void ){

ANSELA = digital; // Set Port A as digital port

TRISAbits.TRISA4 = output; // RA4 as output

TRISAbits.TRISA7 = output; // RA7 as output

ANSELB = digital; // Set Port B as digital port

TRISBbits.TRISB4 = input; // Set RB4 as input

}

**RESULTS:**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

Ilustración 1: Successful build

**Tabla

Descripción generada automáticamente**

Ilustración 2: Modified registers

Tabla

Descripción generada automáticamente

Ilustración 3: Modified registers

GitHub Links:

https://github.com/A01351360/Microcontroladores/tree/main/Laboratorio%20de%20microcontroladores

**CONCLUSIONS:**

As a group conclusion we just want to stablish how complicated a simulation can be, and mostly when you’ve never interacted with any of the software’s used.

Besides that, the interaction between the hardware, the firmware and software are more complex than what it seems on a first sight. But as long as we keep looking for solutions most of the challenges, we can overcome the difficulties.