Parallel Ports (GPIOs)

Laboratory Report Practice 5

Sebastian Medina Garza A01139531

Ricardo Cárdenas García A01232465

Eduardo Daniel Pérez Caro A00823082

Prof. Matías Vázquez

Monterrey NL México 05/04/2022

Objectives

The main objective of this laboratory is to become familiar with the basic instructions for configuring and operating the microcontroller general purpose input/output (GPIO) ports. The student will create the microcontroller firmware that runs the basic operational logic behind a game called Wack-a-mole.

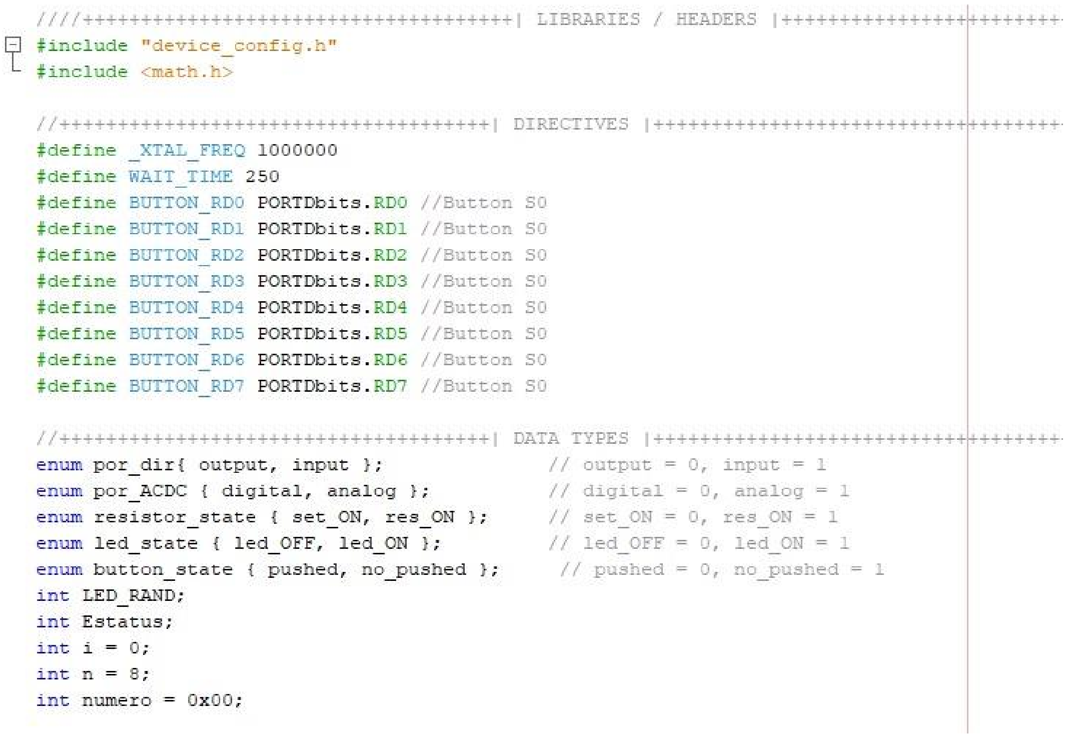
Introduction

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

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 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.

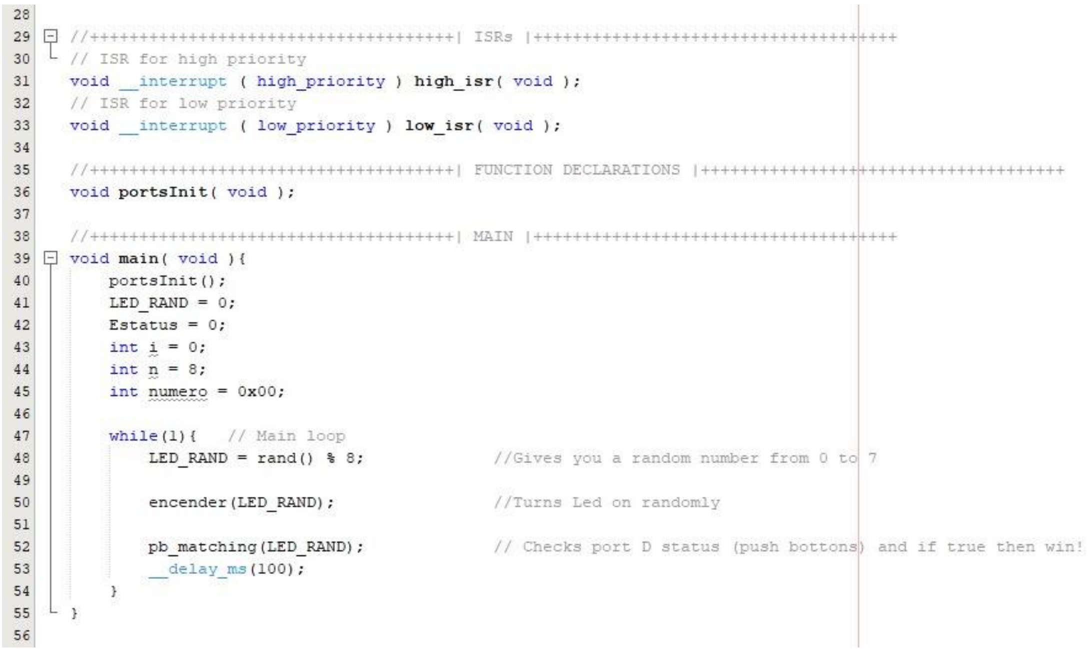
MPLAB Libraries / Directives / Data Types

In the first section (figure 1) it’s shown the libraries, it’s where the libraries are defined, in this case are two; device_config.h and math.h. Below the libraries are the directives section, this is in order to reference the pin we want to execute in code. For example PORTDbits.RD3 instead of writing the name of the port it's easy to identify as BUTTON_RD3. The last part of this section is named data types, it’s in order to have two states; input (1) and output (0). And in last are the int, that are the ones the Pat tide fine depths variables we’re going to use later.



```c
////+++++++++++++++++++++++++++++++++++++++| LIBRARIES / HEADERS |++++++++++++++++++++++++
#include "device_config.h"
#include <math.h>

//+++++++++++++++++++++++++++++++++++++++| DIRECTIVES |++++++++++++++++++++++++++++++++++++++
#define _XTAL_FREQ 1000000
#define WAIT_TIME 250
#define BUTTON_RD0 PORTDbits.RD0 //Button S0
#define BUTTON_RD1 PORTDbits.RD1 //Button S0
#define BUTTON_RD2 PORTDbits.RD2 //Button S0
#define BUTTON_RD3 PORTDbits.RD3 //Button S0
#define BUTTON_RD4 PORTDbits.RD4 //Button S0
#define BUTTON_RD5 PORTDbits.RD5 //Button S0
#define BUTTON_RD6 PORTDbits.RD6 //Button S0
#define BUTTON_RD7 PORTDbits.RD7 //Button S0

//+++++++++++++++++++++++++++++++++++++++| 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 button_state { pushed, no_pushed };       // pushed = 0, no_pushed = 1
int LED_RAND;
int Estatus;
int i = 0;
int n = 8;
int numero = 0x00;
```

Main program

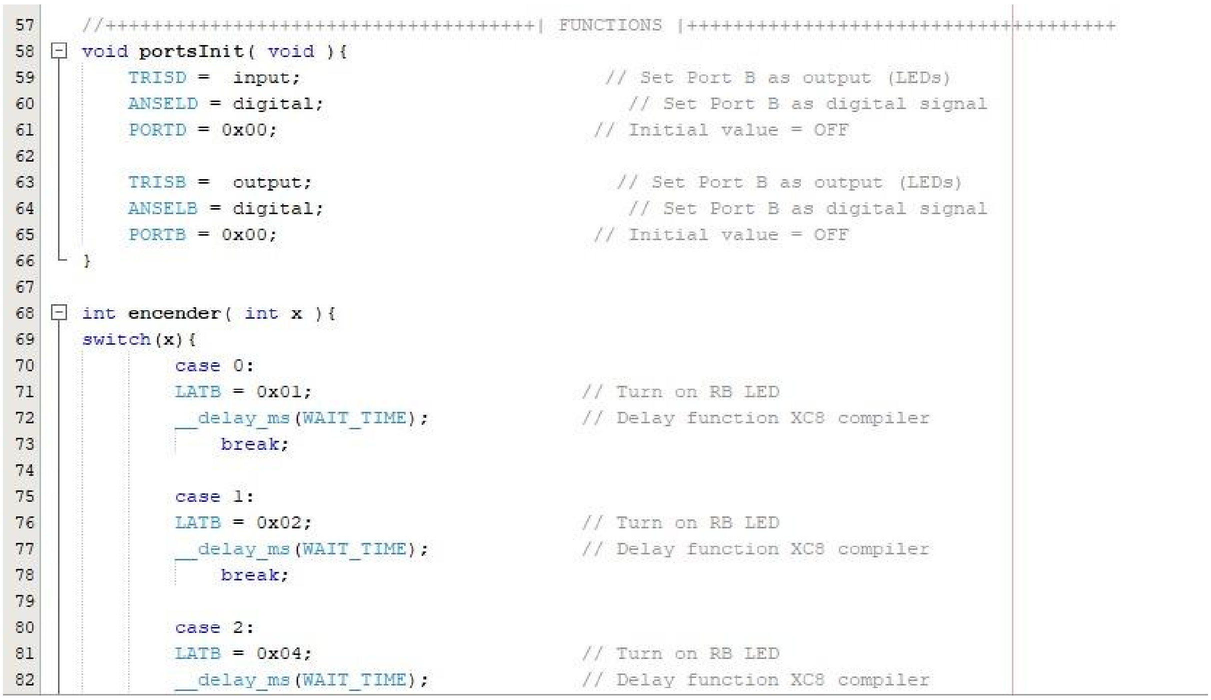


```c

//+++++++++++++++++++++++++++++++++++++++| 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();
    LED_RAND = 0;
    Estatus = 0;
    int i = 0;
    int n = 8;
    int numero = 0x00;

    while(1){   // Main loop
        LED_RAND = rand() % 8;          //Gives you a random number from 0 to 7

        encender(LED_RAND);             //Turns Led on randomly

        pb_matching(LED_RAND);          // Checks port D status (push bottons) and if true then win!
        __delay_ms(100);
    }
}

```

Here is where the main program is executed, portsInit() is the function that will inicializa the variables. LED_RAND, Estatus, and i = 0, n = 8 and número = 0x00.

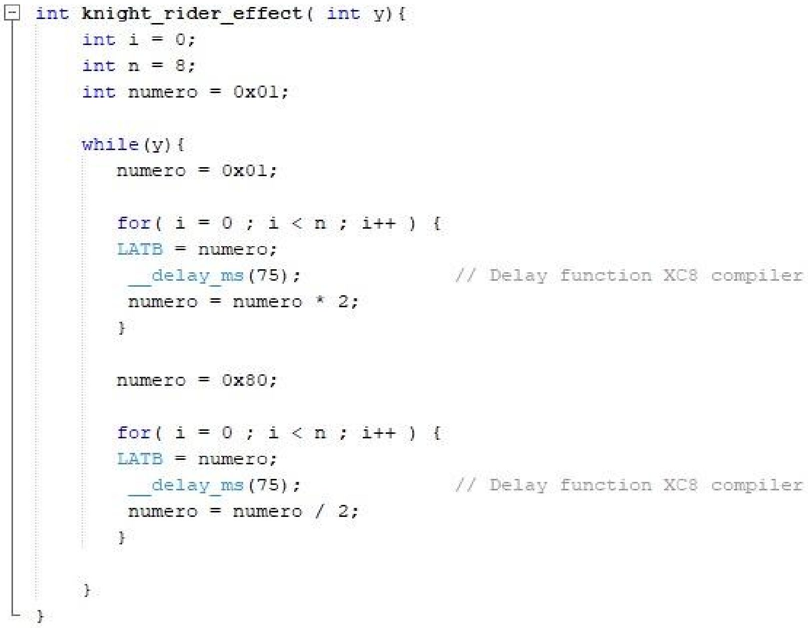
In the mail loop (while), that starts with LED_RAND is a function that will give random numbers from 0 to 7. This is followed up by encender, that basically depending on the random number that we get in the last function, it’ll turn on the corresponding led.

Functions



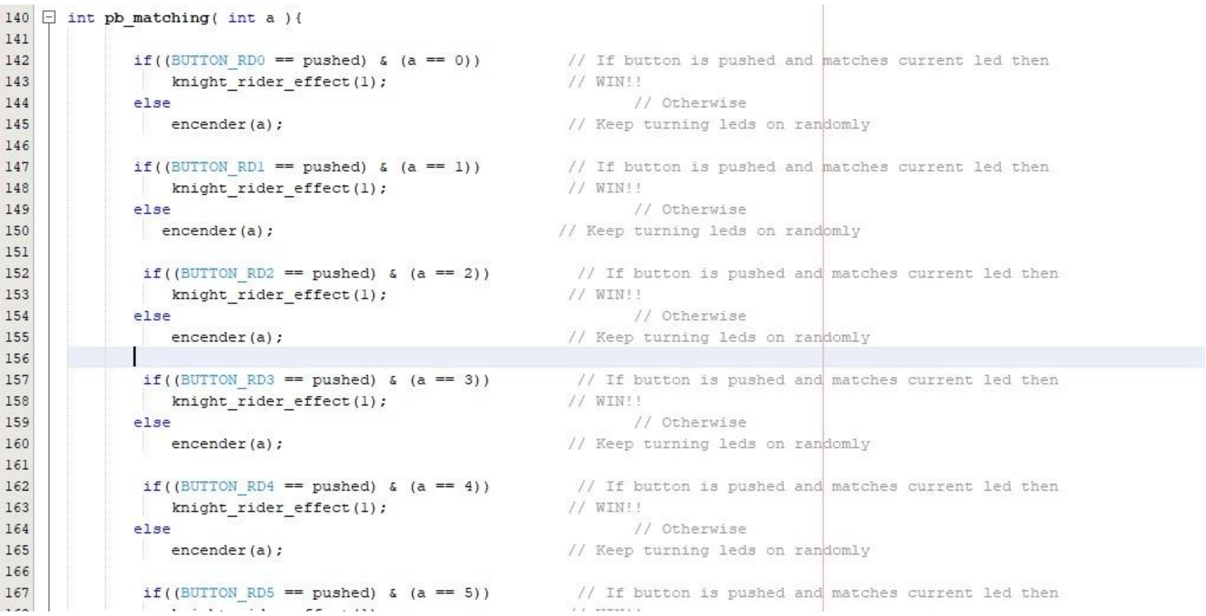
```
//+++++++++++++++++++++++++++++++++++++++| FUNCTIONS |+++++++++++++++++++++++++++++++++++++++++
void portsInit( void ){
    TRISD =  input;                          // Set Port B as output (LEDs)
    ANSELD = digital;                          // Set Port B as digital signal
    PORTD = 0x00;                           // Initial value = OFF

    TRISB =  output;                          // Set Port B as output (LEDs)
    ANSELB = digital;                          // Set Port B as digital signal
    PORTB = 0x00;                           // Initial value = OFF
}

int encender( int x ){
switch(x){
        case 0:
        LATB = 0x01;                     // Turn on RB LED
        __delay_ms(WAIT_TIME);           // Delay function XC8 compiler
            break;

        case 1:
        LATB = 0x02;                     // Turn on RB LED
        __delay_ms(WAIT_TIME);           // Delay function XC8 compiler
            break;

        case 2:
        LATB = 0x04;                     // Turn on RB LED
        __delay_ms(WAIT_TIME);           // Delay function XC8 compiler
```

This function with TRISD will set the port B as output, ANSELD is the function that sets the signal to digital and PORTD is the initial value. The function encender, will turn on the LED corresponding to the number randomly generated.



```
int knight_rider_effect( int y){
    int i = 0;
    int n = 8;
    int numero = 0x01;

    while(y){
       numero = 0x01;

       for( i = 0 ; i < n ; i++ ) {
       LATB = numero;
        __delay_ms(75);             // Delay function XC8 compiler
        numero = numero * 2;
       }

       numero = 0x80;

       for( i = 0 ; i < n ; i++ ) {
       LATB = numero;
        __delay_ms(75);             // Delay function XC8 compiler
        numero = numero / 2;
       }

    }
}
```

Knight rider will create a led lightning sequence, while y is on, it will execute the first for, that will start in I = 0 until 7, in increments of 1, this is the going secuence. And the return u



```
int pb_matching( int a ){

        if((BUTTON_RD0 == pushed) & (a == 0))        // If button is pushed and matches current led then
            knight_rider_effect(1);                  // WIN!!
        else                                                  // Otherwise
            encender(a);                             // Keep turning leds on randomly

        if((BUTTON_RD1 == pushed) & (a == 1))        // If button is pushed and matches current led then
            knight_rider_effect(1);                  // WIN!!
        else                                                  // Otherwise
           encender(a);                             // Keep turning leds on randomly

         if((BUTTON_RD2 == pushed) & (a == 2))        // If button is pushed and matches current led then
            knight_rider_effect(1);                  // WIN!!
        else                                                  // Otherwise
            encender(a);                             // Keep turning leds on randomly

         if((BUTTON_RD3 == pushed) & (a == 3))        // If button is pushed and matches current led then
            knight_rider_effect(1);                  // WIN!!
        else                                                  // Otherwise
            encender(a);                             // Keep turning leds on randomly

         if((BUTTON_RD4 == pushed) & (a == 4))        // If button is pushed and matches current led then
            knight_rider_effect(1);                  // WIN!!
        else                                                  // Otherwise
            encender(a);                             // Keep turning leds on randomly

         if((BUTTON_RD5 == pushed) & (a == 5))        // If button is pushed and matches current led then
```

pb_matching will check the push buttons that corresponds to the led that is turn on, for example if push button 2 is pressed and led 1 is on, it will still lighting random leds, in case led 2 is on and push button 2 is on, you will win the game.

Video:
https://drive.google.com/file/d/1gkgNzW-j1s-F0nV4UM8H2RUgLwZOyzCL/view?usp=drivesdk

Conclusions

Sebastian Medina

This laboratory practice was a very helpful way to get to know better how the PIC18 microcontroller works and how it interacts with the compiler of MPLAB.

Ricardo Cárdenas

In this practice we set up the curiosity with the MPLAB software,in which we create a simple game, for as simple as it seems the logic was a little tricky. I'm excited to see what we can achieve with this tools

Eduardo Perez

This practice was useful for the introduction and application of the parallel ports of the PIC18, I think this is a very powerful tool for future projects, also the understanding of the concepts was a bit hard at the beginning.

Reference

- Matiaz, V. (2022). Introduction to MPLAB X IDE. Retrieved from: **https://github.com/matias-vazquez/microcontroladores/tree/main/Lab04**