PARALLEL PORTS (GPIO)

Introduction

Whack-a-mole is an arcade game in which players use a mallet to hit toy moles, which appear at random locations, to send they back down into their hiding holes. In this laboratory, you will develop the microcontroller firmware that runs the basic operational logic behind this game. Moreover, you will carry out experimental firmware testing to correct possible logic errors of your design, using development hardware, including the Curiosity board and additional components.

Objectives

- To become familiar with the basic instructions for configure and operate the microcontroller general purpose input/output (GPIO) ports

Activities

Software implementation

1. Create a new project in MPLAB X IDE for the Curiosity board (PIC18F45K50 µC).
2. Start your main C code with the LIBRARIEs / HEADERs section. Here, you will invoke all the required libraries and/or headers using the pre-processor directive #include.
3. Continue your C code with the DIRECTIVES section, where you will setup symbolic constants using the pre-processor directive #define, for example:

```
#define _XTAL_FREQ 1000000
```

4. Continue with the DATA TYPES section. In this section, you declare enumeration tags and define the set integer identifiers using the enum datatype (as an alternative of #define). For example, instead of using the numbers 2 and 8 throughout your code, use bbase and limit constants that belong to exponent datatype, by declaring:

```
enum exponent{bbase = 2, limit = 8};
```

5. Now write the FUNCTIONS DECLARATION section. Your code must be structured, so write function prototypes for all user-defined functions you will use. For example, if you need a function to configure the microcontroller ports, declare it as follows:

```
void portsInit(void);
```

6. It is time for the MAIN section. In your C code, the main function drives the LEDs of your minimum system card to represent that a toy mole has shown up, in a continuous and random fashion, by turning on one LED out of the 8-LED set. To emulate the use of the mallet to hit the toy moles, the player will push one push-button of the minimum system card, so the firmware must be able to read the button's state.

a. E.g. suppose that LED_4 turns on and the player presses in time button_4, thus you must indicate the match by displaying on the LED array the Knight Rider (Larson display) effect. A flow diagram of the Whack-a-Mole code is shown in Figure 1.

7. Finally, write the FUNCTIOS section. Here, you must write all function definitions of all user-declared functions. You can use the flow diagram shown Figure 2 as a starting point for your code.

Hardware implementation

1. Connect PortB of the PIC18 to the LED array of your minimum system.
2. Connect PortD of the PIC18 to the push-button array of your minimum system.
3. Your connection setup must look like the one shown in Figure 3.

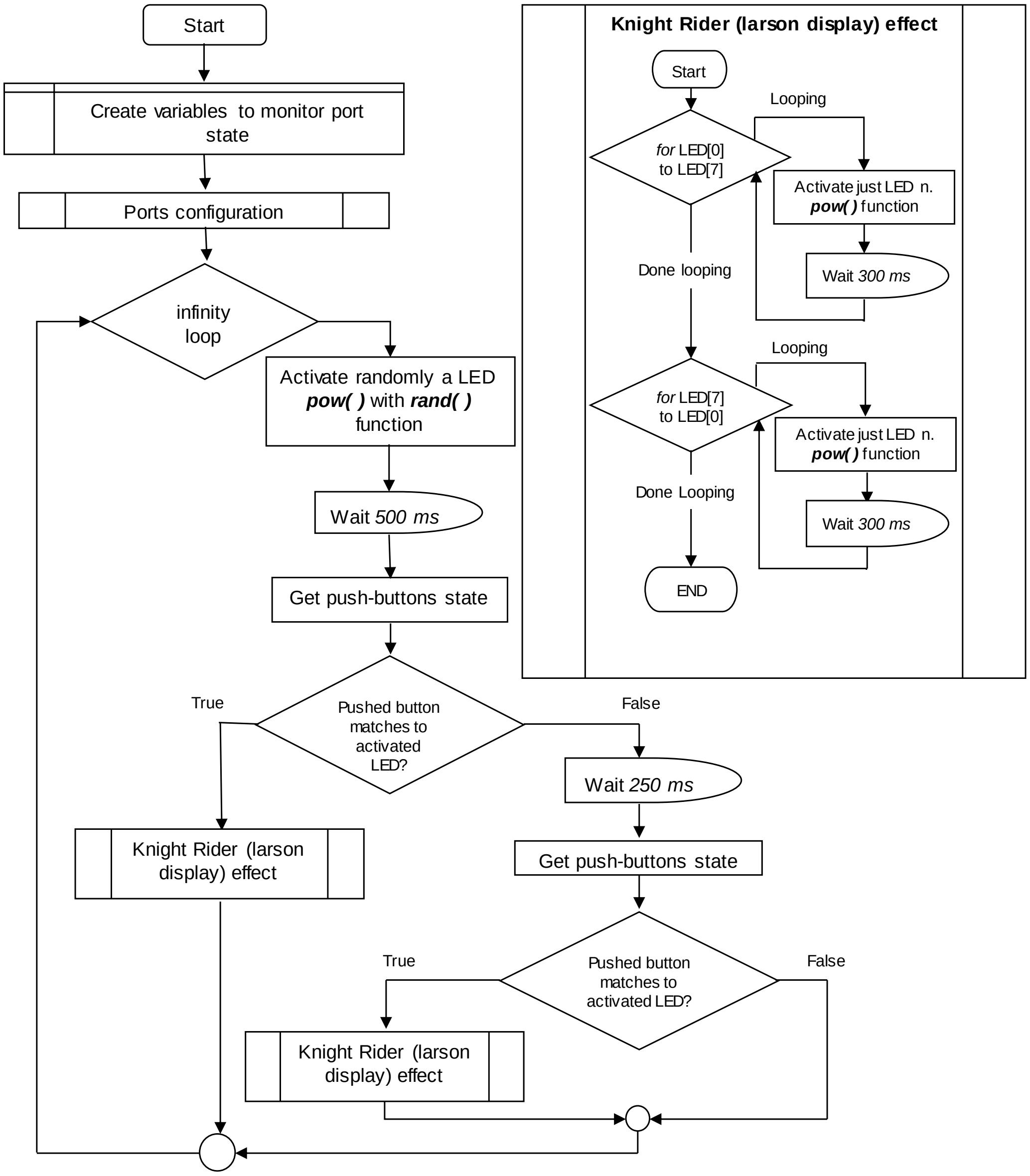
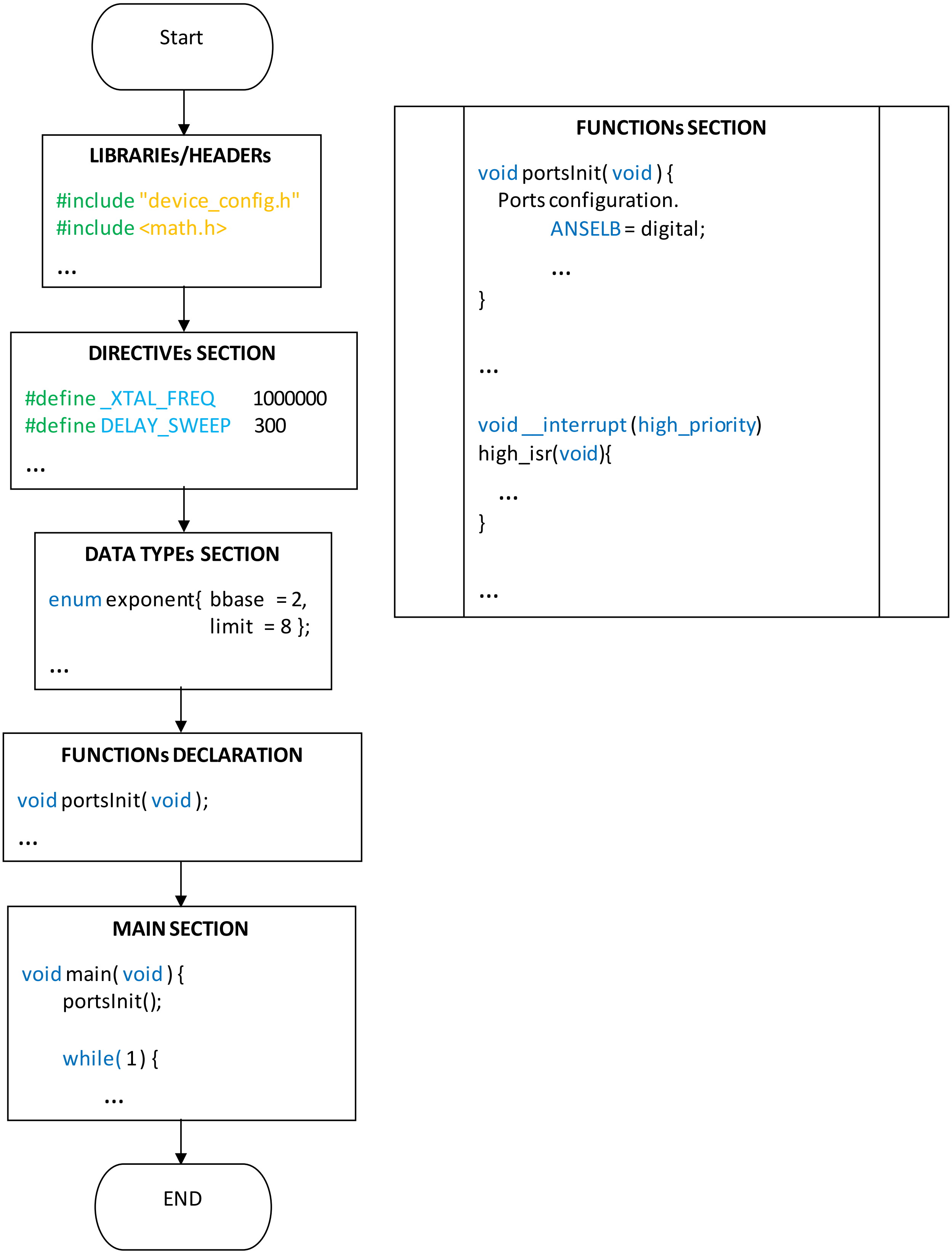

Start
Create variables to monitor port state
Ports configuration
infinity loop
Activate randomly a LED *pow( )* with *rand( )* function
Wait *500 ms*
Get push-buttons state
True
Pushed button matches to activated LED?
False
Knight Rider (larson display) effect
Wait *250 ms*
Get push-buttons state
True
Pushed button matches to activated LED?
False
Knight Rider (larson display) effect
Knight Rider (larson display) effect
Start
Looping
for LED[0] to LED[7]
Activate just LED n. *pow( )* function
Wait *300 ms*
Done looping
Looping
for LED[7] to LED[0]
Activate just LED n. *pow( )* function
Wait *300 ms*
Done Looping
END

Figure 1. Flow diagram of the Whack-a-Mole game



LIBRARIEs/HEADERs

```
#include "device_config.h"
#include <math.h>

...
```

DIRECTIVEs SECTION

```
#define _XTAL_FREQ      1000000
#define DELAY_SWEEP     300

...
```

DATA TYPEs SECTION

```
enum exponent{ bbase  = 2,
               limit  = 8 };

...
```

FUNCTIONs DECLARATION

```
void portsInit( void );

...
```

MAIN SECTION

```
void main( void ) {
      portsInit();

      while( 1 ) {

            ...
```

FUNCTIONs SECTION

```
void portsInit( void ) {
   Ports configuration.
         ANSELB = digital;

         ...
}


...


void __interrupt (high_priority)
high_isr(void){

   ...
}


...
```

Figure 2. Main C program structure

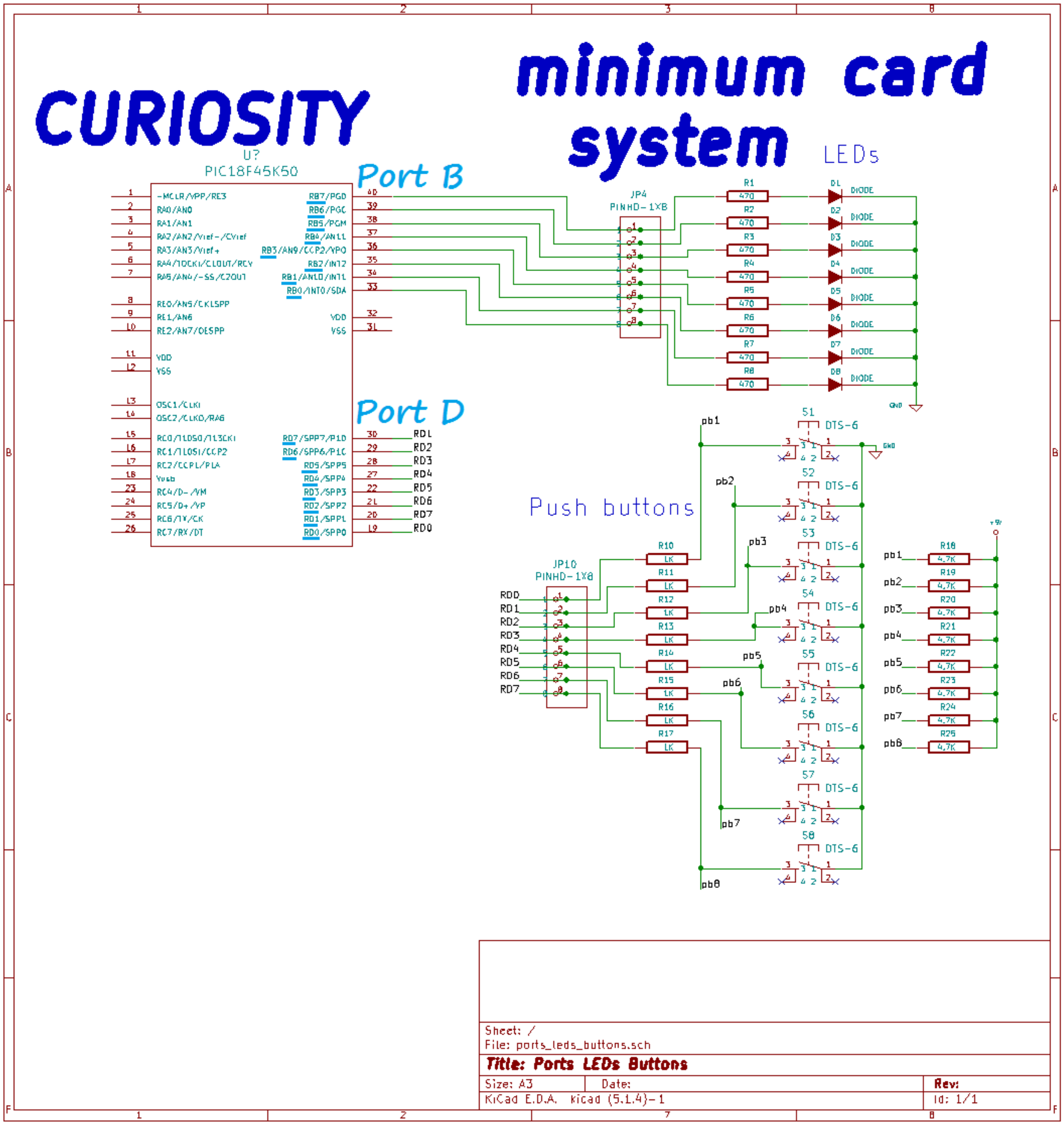


Figure 3. Schematic diagram for hardware implementation.