ECE 414 LAB ASSIGNMENT 2

Microstick II and Analog Discovery

CHIKO DHIRE (Scribe)
TAFITA RAKOTOZANDRY

AUGUST 31th,2020

INTRODUCTION

In this lab, we built a test circuit using a MicrostickII along with a DIP switch as an input device and eight LEDs as an output device. The aim of the lab was increase our familiarity with C programming using a modular structure and to become familiar with the PLIB Peripheral Library

REQUIREMENTS

The following are the requirements of the lab:

- Module portb_out shall implement an initialization function named portb_out_init()that will configure PORTB as an output port that can write all port bits that are available on the output pins of the ’128B (specifically, outputs RB0-RB5, RB7-RB11, and RB13-RB15).
- Module portb_out shall implement a function named portb_out_write(uint16_t a) that accepts a 16-bit unsigned integer as input and writes the 14 least significant bits of that value to output pins {RB15:13,RB11:RB7,RB5:RB0}. In other words, this function writes a 14-bit binary value while “skipping over” the PORTB pins that do not have external connections to the ‘128B.
- Module porta_in shall implement an initialization function named porta_in_init()that will configure PORTA as an input port using internal pullup resistors and to read switch inputs on all available PORTA pins on the ‘128B, specifically pins {RA4:RA0}.
- Module porta_in shall implement a function named porta_in_read()that returns an 8-bit unsigned integer that contains the values {3’b0, RA4:RA0}. Note that the internal pullups mean we can connect a switch to each input pin that is connected to ground; this will be read as a “1” when the switch is open, and as a “0” when the switch is closed.
- The port modules shall be tested using a circuit similar to Figure 1 but expanded to test all PORTA inputs and PORTB outputs. The schematic diagram for this circuit shall be drawn using KiCad.
- The port modules shall be tested using a main program should include initializes the port modules and then enters an infinite while (1) loop that performs the following functions:
    - When input RA4 is a logic low: turn on the output LED corresponding to the binary number encoded on switches RA3:RA0 while turning off all other outputs.
    - When input RA4 is a logic high: turn off the output LED corresponding to the binary number encoded on switches RA3:RA0 while turning on all other outputs.

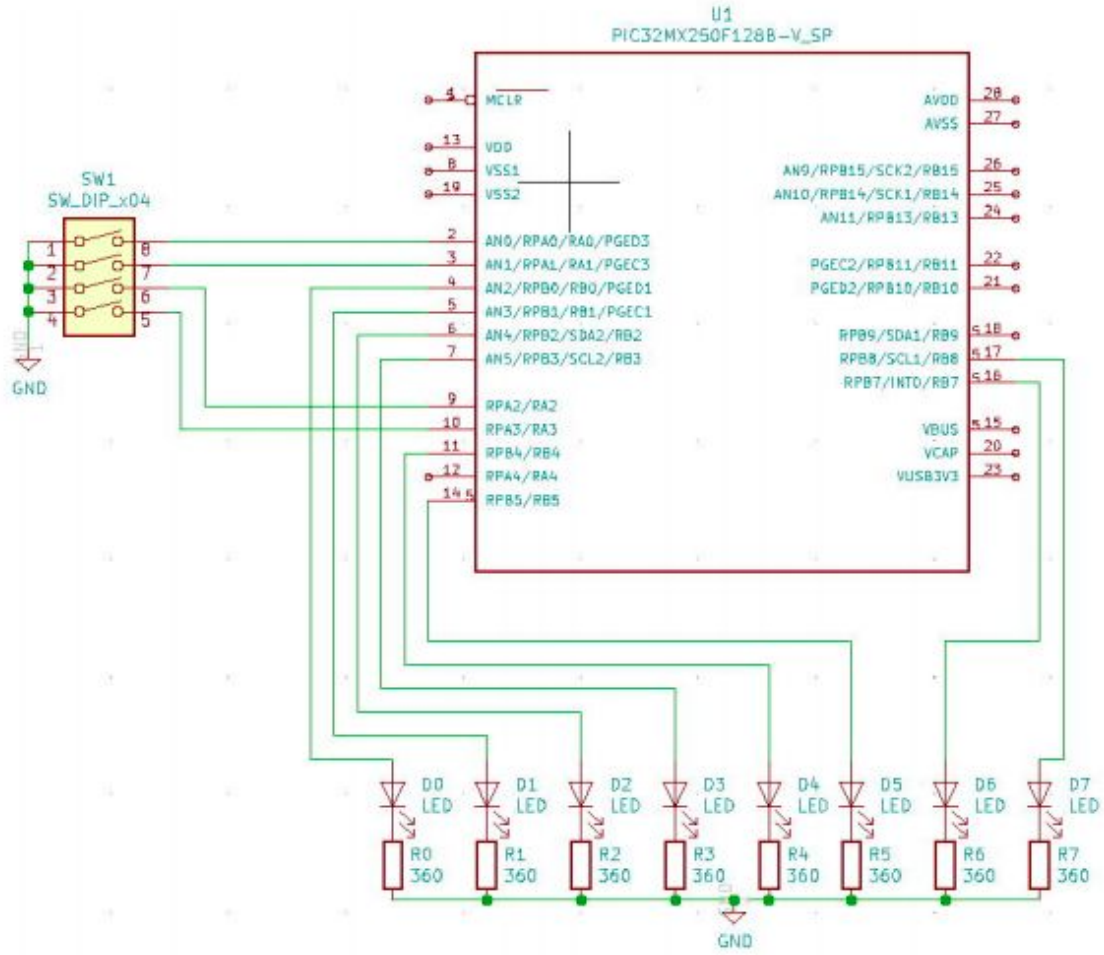

Figure 1: Partial I/O Test Circuit

DESIGN DESCRIPTION

SOFTWARE

Three main software modules were developed: porta_in,portb_out and main. **porta_in** is used to read inputs connected to PORTA on the PIC32. **portb_out** is used to write PIC32 outputs connected to PORTB. **main** makes uses of the two other models to perform the read and write task. **porta_in** and **portb_out** have respectively headers that describe the functions inside of the respective module. The detailed program can be found in the annexe section.

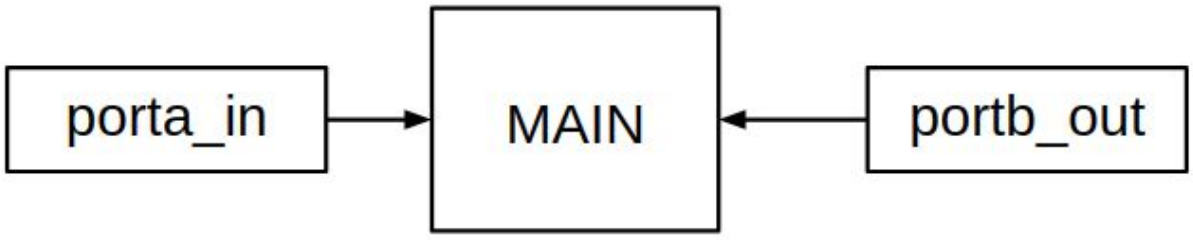

Figure2: Relationship between the program modules

HARDWARE

The hardware implementation is an enhancement of the Partial I/O Test circuit. It possesses 14 LEDs controlled by the DIP switches. The DIP switches have 4 inputs. The input combinations can go up to 16 output different combinations.

NB: The LED should be set in series with current limiting resistors to protect the circuit from overpower.

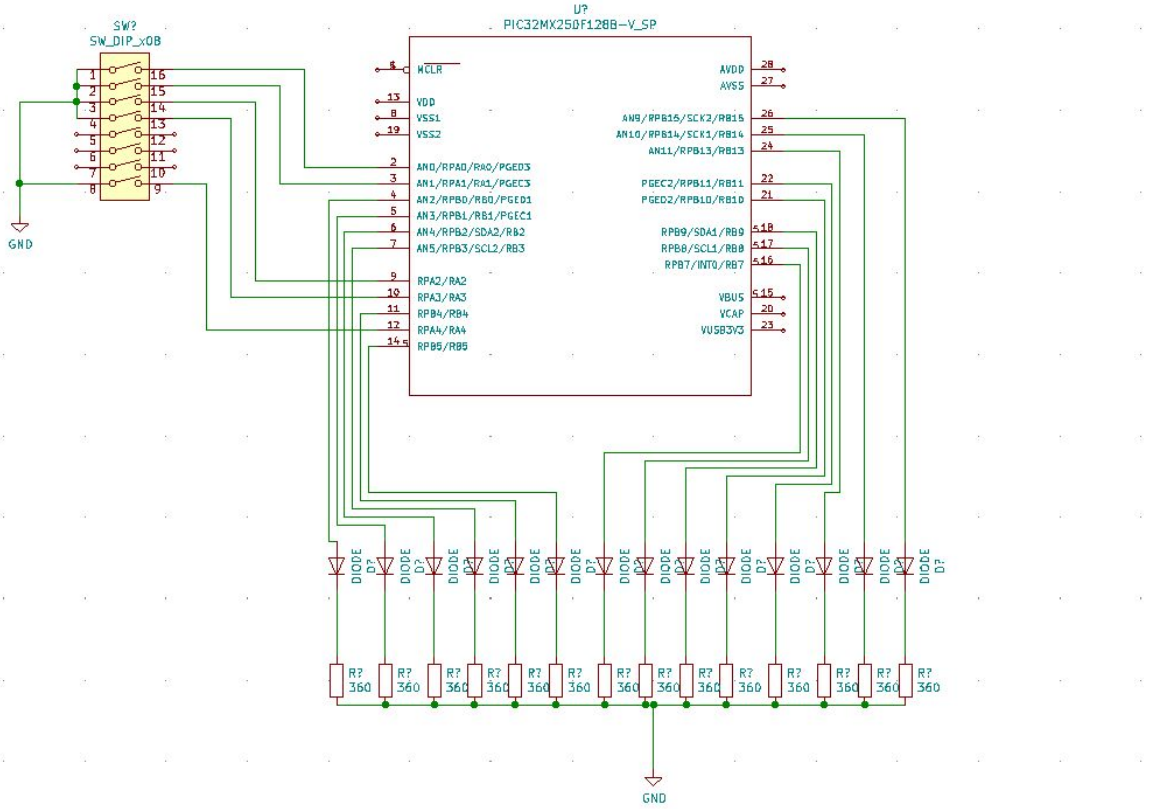

Figure3: Test circuit with 14 outputs

TEST PLAN

- Observe the number of LEDs lit and unlit when RA4 is logic low
- Observe the number of LEDs lit and unlit for the same value as above when RA4 is logic high
- Repeat the above two with different values of RA3:RA0.

TEST REPORT

- The above procedure was carried out for values 0 to 14 and they all passed.

RA4	**Binary value(x)**	**LEDs On**	**RESULT**
0	1-14	x	PASSED
1	1-14	14-x	PASSED

CONCLUSION

The PIC32 reacts with the input combinations. The test result confirms that the requirement was followed . This lab improved our understanding of using modular programming in C. It also helped us to familiarize with the PIC32's different pins and functions. We spent roughly 6 hours working in the lab.

ANNEXE

The C codes

main.c

```
//clock config
#pragma config FNOSC = FRCPLL, POSCMOD = OFF
#pragma config FPLLIDIV = DIV_2, FPLLMUL = MUL_20
#pragma config FPBDIV = DIV_1, FPLLODIV = DIV_2
#pragma config FWDTEN = OFF, JTAGEN = OFF, FSOSCEN = OFF

#include <xc.h>
#include <inttypes.h>
#include "porta_in.h"
#include "portb_out.h"
```

```
main() {
 uint8_t switches;
 uint16_t val;
 uint8_t RA4;
 porta_in_init();
 portb_out_init();

 while (1) {
 switches = porta_in_read();
//val = 0xffff;
 RA4 = switches & 0x10;

 if(RA4==0x00){
    val = 0xffff << (switches & 0x0F);

 }else if(RA4 ==0x10) {
     val = ~(0xFFFF << (switches &0x0F));
 }else{val = 0x0000;};

portb_out_write(val);
}
}
```

Port_in.c

```
#include <xc.h>
#include <inttypes.h>
#include "porta_in.h"
void porta_in_init() {
    //set a RA0: RA3 as input
    ANSELA = 0;
    TRISA = 0x1f;
    CNPUA = 0x1f;

}
uint8_t porta_in_read() {
    //return PORTA reading
    return PORTA;
```

```
}
```

Portb_out.c

```
#include <xc.h>
#include <inttypes.h>
#include "portb_out.h"

void portb_out_init() {
    //set B ports as outputs
    ANSELB = 0;
    TRISB = 0;
}
void portb_out_write(uint16_t val)
{
    //masking technique
    //the RB6 and RB7 is not available
    //temp variable creation
    uint16_t temp1;
     uint16_t temp2;
     uint16_t temp3;


     temp1 = (val & 0x003f); //R0:R5
     temp2 = (val<<1)& 0x0f80; //R7:R11
     temp3 = (val<<2)& 0xE000; //R13:R14
     LATB = temp1|temp2|temp3; //superpose them using or

}
```

The Header Files

porta_in.h

```
#ifndef PORTA_IN_H
#define PORTA_IN_H
#include <inttypes.h>
```

```
extern void porta_in_init();
extern uint8_t porta_in_read();
#endif
```

porta_out.h

```
#ifndef PORTB_OUT_H
#define PORTB_OUT_H
#include <inttypes.h>

extern void portb_out_init();
extern void portb_out_write(uint16_t val);
#endif
```