ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Output 2: Online Course for Microcontrollers: syllabus, open educational resources

Practice leaflet: Module_2-3 external – RB port change interrupts

Lead Partner: International Hellenic University (IHU)

Authors: Theodosios Sapounidis [IHU], Aristotelis Kazakopoulos [IHU], Aggelos Giakoumis [IHU], Sokratis Tselegkaridis [IHU]

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Executive summary

In this Module we will use PIC18F4550 external interrupts.

Chapter 1: **Overview**

Table 1. Overview

Title / short summary	**3. External interrupts – RB port change interrupts**
Expected learning outcomes	• The student will be able to handle external interrupts: ○ INT0 (RB0) ○ INT1 (RB1) ○ INT2 (RB2) • The student will be able to handle RB port change interrupts (on pins RB4~RB7) • The student will be able to load and animate a microcontroller program in the Proteus Design Suite
Keywords	External interrupts, RB interrupts
Duration	The duration of the module_2-3 is 3 hours • Presentation of the module_2-3 by the teacher, 30 minutes • 1st activity, flash a LED with INT1, 40 minutes • 2nd activity, create a moving dot with INT2, 35 minutes • 3rd activity, RB4~RB7 on change interrupt, 35 minutes • 4th activity, a simple alarm system, 40 minutes

Involved	**The teacher:** Presents the slides associated with the module_2-3 and answers question **The students:** Draw circuits in Proteus Schematic, write programs in C language, load programs to a microcontroller and run the simulation using the Proteus Design Suite
Assignment	At the end of the Module_2-3 will be given: • Open Project
Educational tools and equipment	• **Material:** PC • **Software:** CCS C compiler, Proteus Design Suite
Prerequisites / pre-existing knowledge	• The student must be familiarized with the Proteus Design Suite (link1) • The student must be completed Module_2-1 and Module_2-2
Educational content	• CCS C Compiler manual (C Compiler Reference Manual) • MICROCHIP, PIC18F2455/2550/4455/4550 Data Sheet • Module_2-3 slides • Module_2-3 Evaluation leaflet • Module_2-3 Open project leaflet • Module_2-3 Programs, Schematic Proteus (Compressed folder)

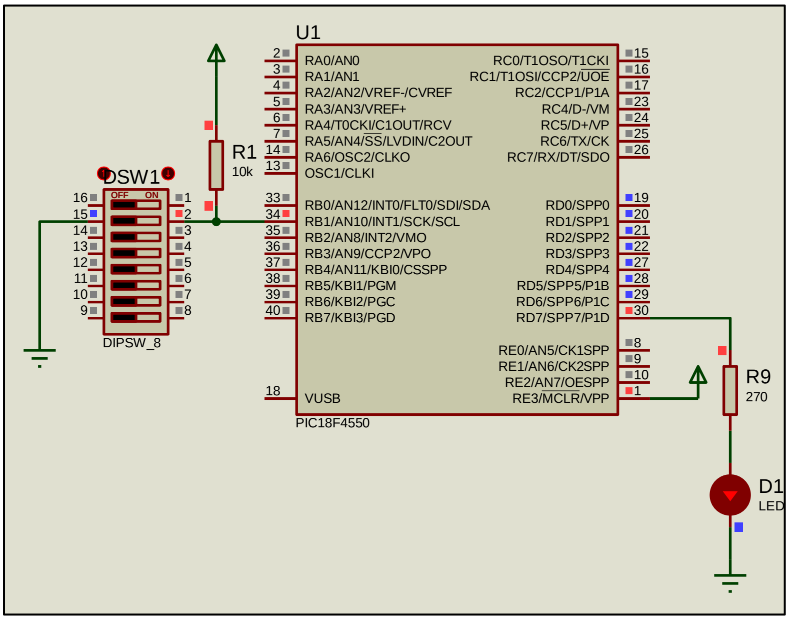
Tips	***Tip.*** *Requirement / operation of the compiler about RB change interrupts.* *The value of PORTB (or one of the pins RB4, RB5, RB6, RB7) must be read to clear the interrupt flag (IF).* *If the IF is not cleared when exiting the interrupt service routine (ISR), the IF is raised and the program re-enters the ISR.*

Chapter 2: **Activities**

2.1 Activity 1. Flash a LED

The purpose of this activity is to flash a LED twice, through the interrupt service routine of RB1 (INT1).

Table 2. Activity 1

Activity 1st (40 minutes)	**Step 1.** The circuit is drawn in the Proteus Design Suite. **Step 2.** The program in C language is written. **Step 3.** The program is compiled with the use of CCS C compiler to the microcontroller machine code. **Step 4.** The machine code is loaded to the microcontroller. **Step 5.** The animation is activated. **Step 6.** Modifications and discussion.
Step 1 (10 minutes)	Draw the circuit of the picture in the Proteus Design Suite.  U1 PIC18F4550 RA0/AN0 RA1/AN1 RA2/AN2/VREF-/CVREF RA3/AN3/VREF+ RA4/T0CKI/C1OUT/RCV RA5/AN4/SS/LVDIN/C2OUT RA6/OSC2/CLKO OSC1/CLKI RB0/AN12/INT0/FLT0/SDI/SDA RB1/AN10/INT1/SCK/SCL RB2/AN8/INT2/VMO RB3/AN9/CCP2/VPO RB4/AN11/KBI0/CSSPP RB5/KBI1/PGM RB6/KBI2/PGC RB7/KBI3/PGD VUSB RC0/T1OSO/T1CKI RC1/T1OSI/CCP2/UOE RC2/CCP1/P1A RC4/D-/VM RC5/D+/VP RC6/TX/CK RC7/RX/DT/SDO RD0/SPP0 RD1/SPP1 RD2/SPP2 RD3/SPP3 RD4/SPP4 RD5/SPP5/P1B RD6/SPP6/P1C RD7/SPP7/P1D RE0/AN5/CK1SPP RE1/AN6/CK2SPP RE2/AN7/OESPP RE3/MCLR/VPP R1 10k DSW1 DIPSW_8 OFF ON R9 270 D1 LED *Figure 1. INT1 - flash a LED*

Step 2
(10 minutes)

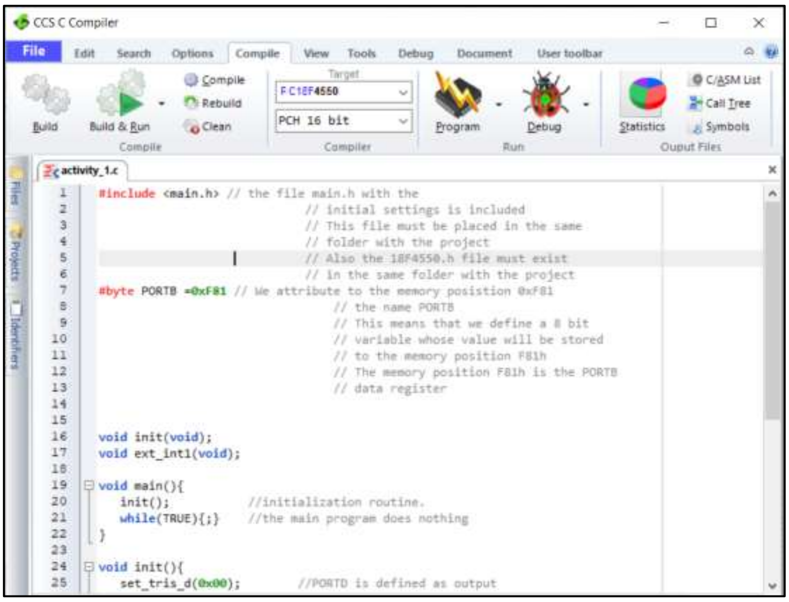
Write in CCS C Compiler the program in C language

```
#include <main.h> // the file main.h with the
                             // initial settings
is included
                             // This file must be
placed in the same
                             // folder with the
project
                             // Also the 18F4550.h
file must exist
                             // in the same folder
with the project
#byte PORTD =0xF83 // We attribute to the memory
position 0xF83
                                 // the name PORTD
                                 // This means
that we define a 8 bit
                                 // variable whose
value will be stored
                                 // to the memory
position F83h
                                 // The memory
position F83h is the PORTD
                                 // data register

void init(void);
void ext_int1(void);

void main(){
   init();           //initialization routine
   while(TRUE){;}       //the main program does
nothing
}


void init(){
   set_tris_d(0x00);        //PORTD is defined as
output
   PORTD = 0b00000000;         //The PORTD data
register is given the value 0
   ext_int_edge(1, L_TO_H); //Activation of the
interrupt from RB1
                            //during the
transition from 0 to 1 (raising edge)
   enable_interrupts(GLOBAL);     //Enable global
interrupts
   enable_interrupts(INT_EXT1);  //Enable external
interrupt by RB1
}

#INT_EXT1 HIGH    //External interrupt by RB1
void ext_int1(){
   int i;
   for(i=1;i<3; i++){      //performed twice
      output_high(PIN_D7); //LED is on
      delay_ms(200);       //wait for 0.2s
      output_low(PIN_D7);  //LED is of
      delay_ms(200);       //wait for 0.2s
```

```
        }
    }
```

Step 3
(4 minutes)

Compile the program in C in order to create the program in the microcontroller machine code (hex file).



```
#include <main.h> // the file main.h with the
                              // initial settings is included
                              // This file must be placed in the same
                              // folder with the project
                              // Also the 18F4550.h file must exist
                              // in the same folder with the project
#byte PORTB =0xF81 // We attribute to the memory posistion 0xF81
                                  // the name PORTB
                                  // This means that we define a 8 bit
                                  // variable whose value will be stored
                                  // to the memory position F81h
                                  // The memory position F81h is the PORTB
                                  // data register


void init(void);
void ext_int1(void);

void main(){
   init();           //initialization routine.
   while(TRUE){;}    //the main program does nothing
}

void init(){
   set_tris_d(0x00);         //PORTD is defined as output
```

Figure 2. CCS C Compiler, translation to machine code (hex file)

Step 4
(1 minutes)

Load to the microcontroller the hex file (program in machine code) that was created from the CCS C Compiler.

Step 5
(5 minute)

Run the simulation and check the correct operation of the circuit.

Step 6
(10 minutes)

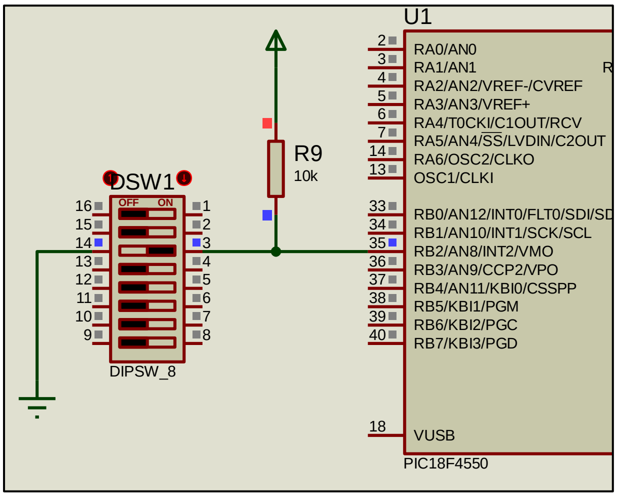
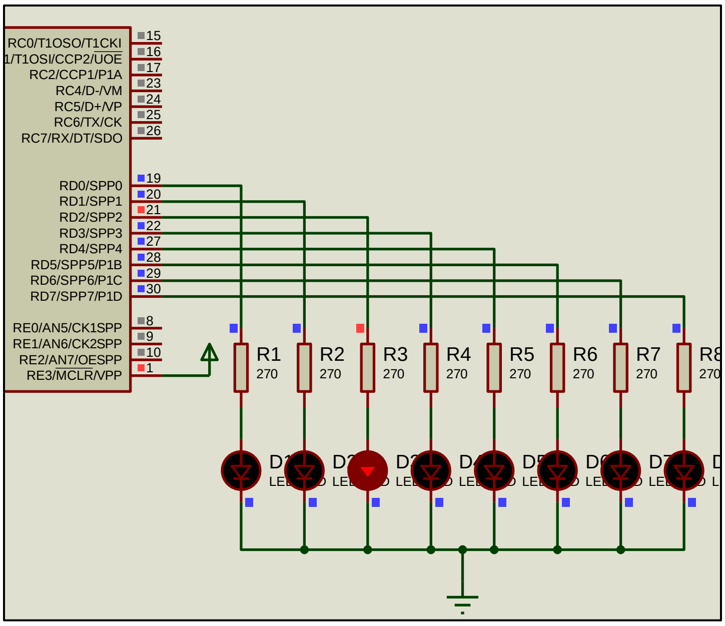
Suggested modifications and discussion:

- modify the code and circuit accordingly so that the interrupt occurs from RB0. Run and check the simulation

2.2 Activity 2. Create a moving dot

The purpose of this activity is to interrupt the main program. In the main program 8 LEDs flash. When an interrupt occurs from INT2, the LEDs create a moving dot.

Table 3. Activity 2

Activity 2nd (35 minutes)	**Step 1.** The circuit is drawn in the Proteus Design Suite. **Step 2.** The program in C language is written. **Step 3.** The program is compiled with the use of CCS C compiler to the microcontroller machine code. The machine code is loaded to the flash memory of the microcontroller. **Step 4.** The animation is activated.
Step 1 (15 minutes)	Draw the circuit of the picture at the Proteus Design Suite.  U1 RA0/AN0 RA1/AN1 RA2/AN2/VREF-/CVREF RA3/AN3/VREF+ RA4/T0CKI/C1OUT/RCV RA5/AN4/SS/LVDIN/C2OUT RA6/OSC2/CLKO OSC1/CLKI RB0/AN12/INT0/FLT0/SDI/SD RB1/AN10/INT1/SCK/SCL RB2/AN8/INT2/VMO RB3/AN9/CCP2/VPO RB4/AN11/KBI0/CSSPP RB5/KBI1/PGM RB6/KBI2/PGC RB7/KBI3/PGD VUSB PIC18F4550 R9 10k DSW1 OFF ON DIPSW_8 *Figure 3(a). INT2 and LEDs*  RC0/T1OSO/T1CKI 1/T1OSI/CCP2/UOE RC2/CCP1/P1A RC4/D-/VM RC5/D+/VP RC6/TX/CK RC7/RX/DT/SDO RD0/SPP0 RD1/SPP1 RD2/SPP2 RD3/SPP3 RD4/SPP4 RD5/SPP5/P1B RD6/SPP6/P1C RD7/SPP7/P1D RE0/AN5/CK1SPP RE1/AN6/CK2SPP RE2/AN7/OESPP RE3/MCLR/VPP R1 270 R2 270 R3 270 R4 270 R5 270 R6 270 R7 270 R8 270 D1 D2 D3 D4 D5 D6 D7 LED *Figure 3(b). INT2 and LEDs*

Step 2
(10 minutes)

Write in CCS C Compiler the program in C language

```
#include <main.h> // the file main.h with the
                             //  initial  settings
is included
                             // This file must be
placed in the same
                             //  folder  with  the
project
                             // Also the 18F4550.h
file must exist
                             // in the same folder
with the project
#byte PORTD =0xF83 // We attribute to the memory
position 0xF83
                                 // the name PORTD
                                 //   This   means
that we define a 8 bit
                                 //       variable
whose value will be stored
                                 // to the memory
position F83h
                                 //   The   memory
position F83h is the PORTD
                                 // data register

void init(void);
void ext_int2(void);

void main(){
   init();        //initialization routine
   while(TRUE){   //flash 8 LEDs
      PORTD=0b11111111;
      delay_ms(100);
      PORTD=0b00000000;
      delay_ms(100);
   }
}

//initialization routine
void init(){
   set_tris_d(0x00);            //PORTD is defined
as output
   PORTD = 0b00000000;           //The PORTD data
register is given the value 0

   ext_int_edge(2, H_TO_L);       //Activation  of
the interrupt from RB2
                                //during      the
transition from 1 to 0 (falling edge)

   enable_interrupts(GLOBAL);     //Enable  global
interrupts
   enable_interrupts(INT_EXT2);  //Enable external
interrupt by RB2
}

//external interrupt by RB2
#INT_EXT2
```

	void ext_int2() { //moving dot
	    PORTD=0b00000000; delay_ms(200);
	    PORTD=0b10000000; delay_ms(200);
	    PORTD=0b01000000; delay_ms(200);
	    PORTD=0b00100000; delay_ms(200);
	    PORTD=0b00100000; delay_ms(200);
	    PORTD=0b00010000; delay_ms(200);
	    PORTD=0b00001000; delay_ms(200);
	    PORTD=0b00000100; delay_ms(200);
	    PORTD=0b00000010; delay_ms(200);
	    PORTD=0b00000001; delay_ms(200);
	    PORTD=0b00000010; delay_ms(200);
	    PORTD=0b00000100; delay_ms(200);
	    PORTD=0b00001000; delay_ms(200);
	    PORTD=0b00010000; delay_ms(200);
	    PORTD=0b00100000; delay_ms(200);
	    PORTD=0b01000000; delay_ms(200);
	    PORTD=0b10000000; delay_ms(200);
	    PORTD=0b00000000; delay_ms(200);
	}

Step 3 (5 minutes)	Use the CCS C Compiler to translate the programm from C language to the microcontroller machine code. Load to the microcontroller the hex file (machine code) that was created from the CCS Compiler.
Step 4 (5 minutes)	Run the simulation and check the correct operation of the circuit.

2.3 Activity 3. RB4~RB7 on change interrupt

The purpose of this activity is to handles interrupts by state changes in RB4, RB5, RB6, and RB7. When an interrupt occurs, the corresponding LED connected to the PORTD is activated.

Table 4. Activity 3

Activity 3rd (35 minutes)	**Step 1.** The circuit is drawn at the Proteus Design Suite. **Step 2.** The program in C language is written. **Step 3.** The program is compiled with the use of CCS C compiler to the microcontroller machine code (the hex.file is created). The program in machine code is loaded to the microcontroller. **Step4.** The animation is activated.

Step 1
(12 minutes)

Draw the circuit of the picture in the Proteus Design Suite

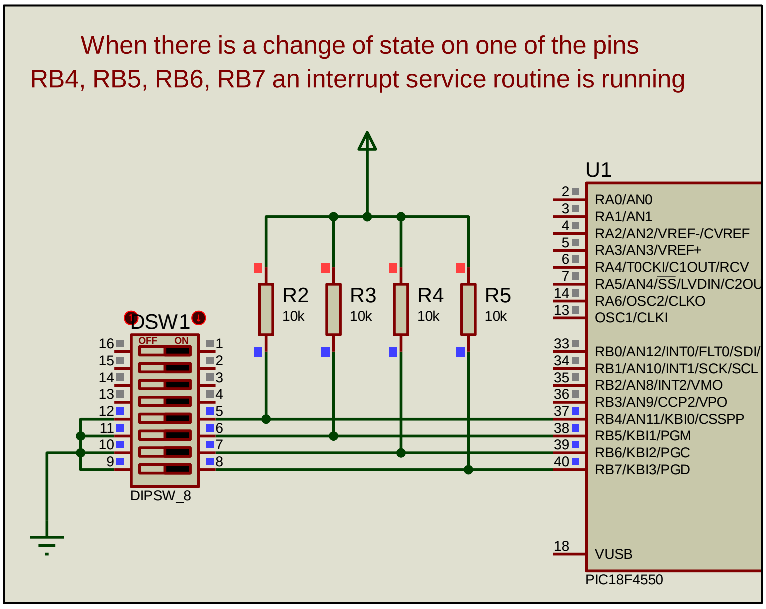

When there is a change of state on one of the pins
RB4, RB5, RB6, RB7 an interrupt service routine is running

Figure 4(a). INTRB and LEDs

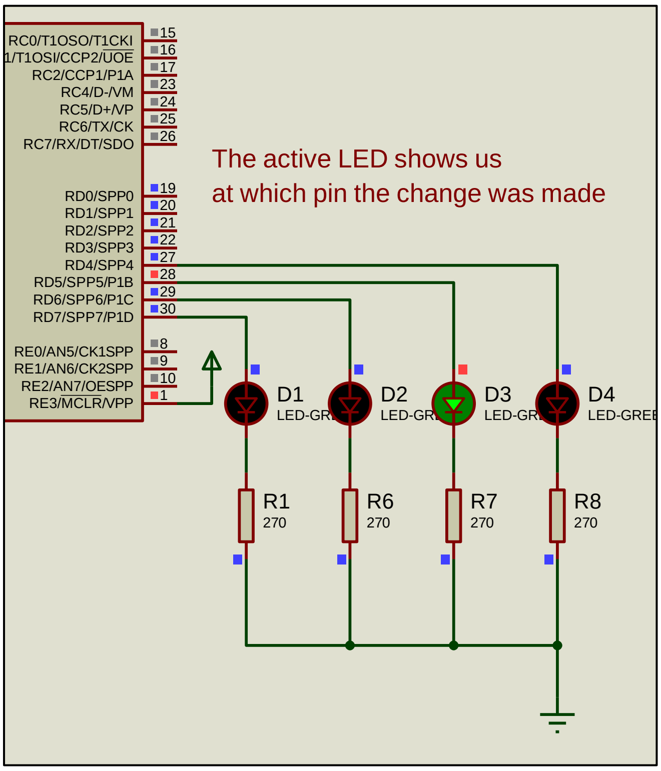

The active LED shows us
at which pin the change was made

Figure 4(b). INTRB and LEDs

Step 2
(13 minutes)

Write in CCS C Compiler the program in C language

```
#include <main.h> // the file main.h with the
                              //  initial  settings
is included
                              // This file must be
placed in the same
                              //  folder  with  the
project
                              // Also the 18F4550.h
file must exist
                              // in the same folder
with the project
#byte PORTD =0xF83 // We attribute to the memory
position 0xF83
                                  // the name PORTD
                                  //   This   means
that we define a 8 bit
                                  //       variable
whose value will be stored
                                  // to the memory
position F83h
                                  //   The   memory
position F83h is the PORTD
                                  // data register

#byte PORTB=0xF81  // We attribute to the memory
position 0xF81
                                  // the name PORTD
                                  //   This   means
that we define a 8 bit
                                  //       variable
whose value will be stored
                                  // to the memory
position F81h
                                  //   The   memory
position F81h is the PORTD
                                  // data register

void  rb(void)  ;   //Interrupt  service  routine
statement (from RB4, RB5, RB6, RB7)
void init(void);
int8 lastPORTB;  //Global variable to hold the last
value of PORTB

void main(){
   init();     //call the initialization routine
   while(TRUE){;}  //the main program does nothing
}

//initialization routine
void init(){
   set_tris_d(0x00);           //PORTD is defined
as output
   PORTD = 0b00000000;          //The PORTD data
register is given the value 0
   lastPORTB=PORTB;
   enable_interrupts(GLOBAL);      //Enable global
interrupts
   enable_interrupts(INT_RB);      //Enable change
interrupt by RB4, RB5, RB6, RB7
}

//PORTB change interrupt
#INT_RB
void rb (void){
   int8 changes;                       //Define an
8bit variable
   changes = lastPORTB ^ PORTB;      //The changed
bit becomes 1 and appears in the corresponding
position in the change variable
   lastPORTB=PORTB;                 //The new PORTB
value is transferred to the lastPORTB variable
   PORTD=changes;                     //The changed
bit is displayed in PORTD
   delay_ms (100);                      //delay to
avoid bounces
}
```

Step 3 (5 minutes)	Compile the program in order to create the hex.file (program in machine code). Load the program (hex.file) to the microcontroller.
Step 4 (5 minutes)	Run the simulation and check the correct operation of the circuit.

2.4 Activity 4. Simple alarm system

The purpose of this activity is to create a simple alarm system. The system sensors are simulated by 4 switches connected to RB4 ~ RB7. The alarm works as follows: a switch in RB0 arms or disarms the system. If the system is armed and one of the 4 switches changes state, then the microcontroller activates an LED (or a buzzer) for 6 seconds. The sensor / switch that gave the alarm is displayed in PORTD.

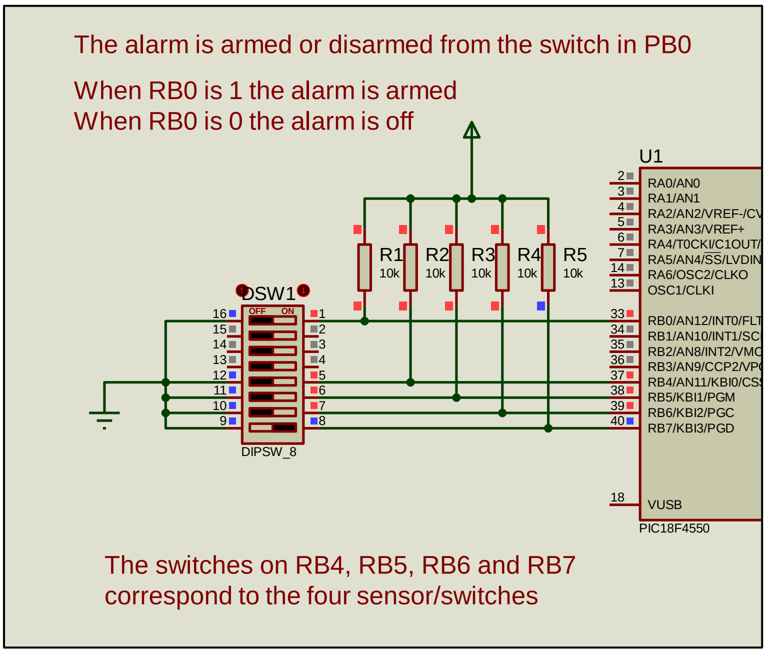
Table 5. Activity 4

Activity 4rd (40 minutes)	**Step 1.** The circuit is drawn at the Proteus Design Suite. **Step 2.** The program in C language is written. **Step 3.** The program is compiled with the use of CCS C compiler to the microcontroller machine code (the hex.file is created). The program in machine code is loaded to the microcontroller.

Step4. The animation is activated.

Step 1
(15 minutes)

Draw the circuit of the picture in the Proteus Design Suite.


The alarm is armed or disarmed from the switch in PB0

When RB0 is 1 the alarm is armed
When RB0 is 0 the alarm is off

The switches on RB4, RB5, RB6 and RB7
correspond to the four sensor/switches

Figure 5(a). A simple alarm system

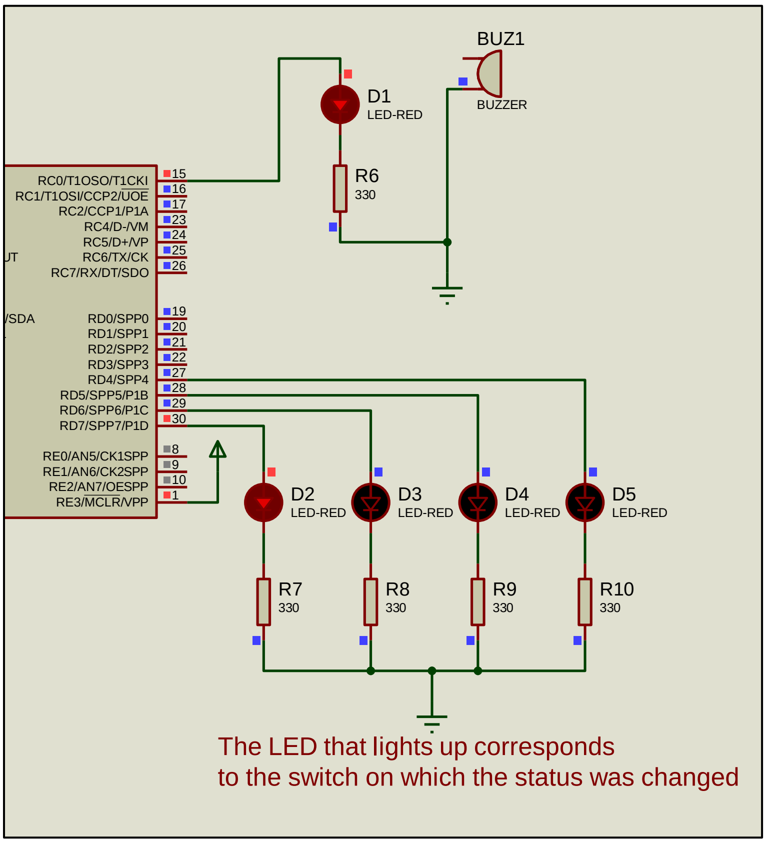

The LED that lights up corresponds
to the switch on which the status was changed

Figure 5(b). A simple alarm system

Step 2
(15 minutes)

Write in CCS C Compiler the program in C language

```
#include <main.h> // the file main.h with the
                             //  initial  settings  is
included
                             // This file must be placed
in the same
                             // folder with the project
                             // Also the 18F4550.h file
must exist
                             // in the same folder with
the project
#byte PORTD =0xF83 // We attribute to the memory position
0xF83
                                 // the name PORTD
                                 // This means that we
define a 8 bit
                                 //   variable   whose
value will be stored
                                 //   to   the   memory
position F83h
                                 // The memory position
F83h is the PORTD
                                 // data register
#byte PORTB=0xF81  // We attribute to the memory position
0xF81
                                 // the name PORTD
                                 // This means that we
define a 8 bit
                                 //   variable   whose
value will be stored
                                 //   to   the   memory
position F81h
                                 // The memory position
F81h is the PORTD
                                 // data register
#byte PORTC=0xF82  // We attribute to the memory position
0xF82
                                 // the name PORTC
                                 // This means that we
define a 8 bit
                                 //   variable   whose
value will be stored
                                 //   to   the   memory
position F82h
                                 // The memory position
F82h is the PORTC
                                 // data register

//Declaration of functions, global variables
void init (void);       //initialization routine
void rb  (void);           //interrupt service routine
statement (from RB4, RB5, RB6, RB7)

int8 lastPORTB;          //Global variable to hold the
last value of PORTB

void main(){
     init();           //call the initialization routine
     while (TRUE) {;}   //the main program does nothing
}

//interrupt service routine (change on RB4~RB7)
#INT_RB
void rb (void){
   int8 changes;       //Define an 8bit variable
   changes = lastPORTB ^ PORTB;      //The changed bit
becomes 1 and appears in the corresponding position in
the change variable
   lastPORTB=PORTB;                    //The new PORTB
value is transferred to the lastPORTB variable

   if(input(PIN_B0)==1){
      output_high(PIN_C0);    //alarm is activated
      PORTD=changes;         //The changed bit of PORTB
is displayed on PORTD | a LED is on
      delay_ms(6000);         //wait for 6 seconds
      output_low(PIN_C0);     //alarm is de-activated
      PORTD=0x00;             //LED is off
   }
}

//initialization routine
void init (void){
   set_tris_b(0xff);       // PORTB is defined as input
   set_tris_d(0x00);      // PORTD is defined as output
   set_tris_c(0x00);      // PORTC is defined as output

   enable_interrupts(GLOBAL);          //Enable  global
interrupts
   enable_interrupts(INT_RB);          //Enable  change
interrupt by RB4, RB5, RB6, RB7

   PORTD=0x00;                         //The PORTD data
register is given the value 0
   PORTC=0x00;                         //The PORTC data
register is given the value 0
   lastPORTB=PORTB;               //The new PORTB value
is transferred to the lastPORTB variable
}
```

Step 3 (5 minutes)	Compile the program in order to create the hex.file (program in machine code). Load the program (hex.file) to the microcontroller.
Step 4 (5 minutes)	Run the simulation and check the correct operation of the circuit.

Chapter 3: **Recapitulation**

☞ The schematic of the circuits was drawn with Proteus Design Suite

☞ External interrupts - RB port change interrupts were used to implement applications: flash a LED, create a moving dot, simple alarm system.

☞ The programs in C was written in CCS C Compiler.

☞ An interrupt service routine was used.

☞ The programs in C was compiled to the microcontroller machine code (hex file).

☞ The machine code was “loaded” to the microcontroller and the animation was activated.

References

CCS C Compiler Manual. Ccsinfo.com. (2021). Retrieved from https://www.ccsinfo.com/downloads/ccs_c_manual.pdf.

PIC18F2455/2550/4455/4550 Data Sheet. Ww1.microchip.com. (2006). Retrieved from https://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf.

Proteus Tutorial : Getting Started with Proteus PCB Design (Version 8.6). Youtube.com. (2017). Retrieved from https://www.youtube.com/watch?v=GYAHwYUUs34.

Simple LED Circuits. Electronics Hub. (2017). Retrieved from https://www.electronicshub.org/simple-led-circuits/.

Appendix. Figures with high resolution

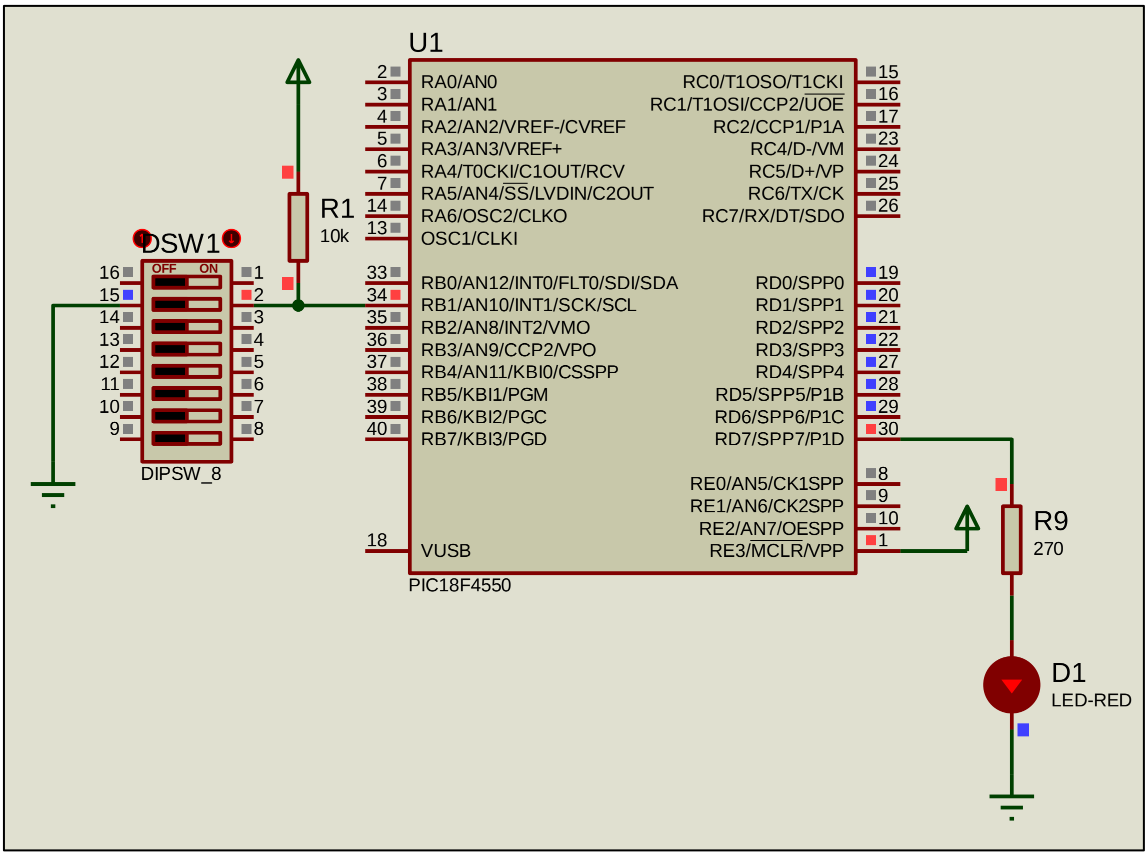
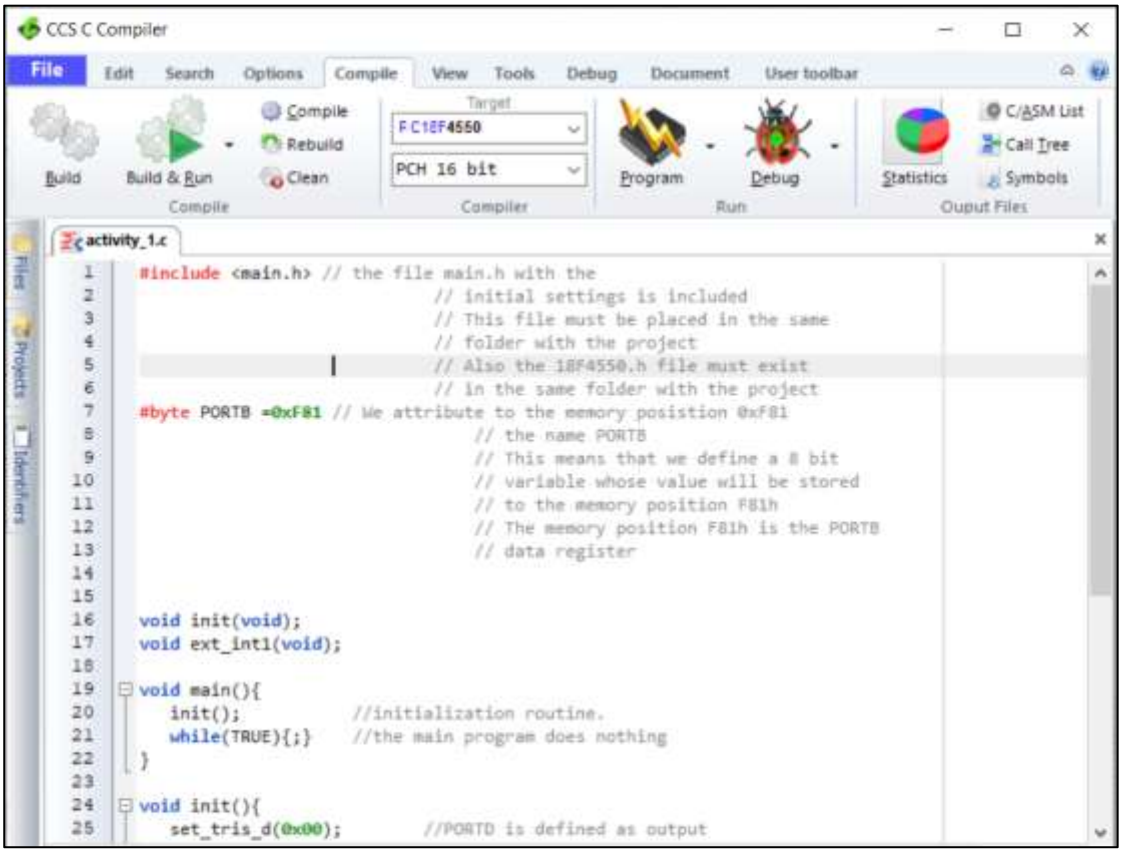


Figure 1. INT1 - flash a LED



```
#include <main.h> // the file main.h with the
                              // initial settings is included
                              // This file must be placed in the same
                              // folder with the project
                              // Also the 18F4550.h file must exist
                              // in the same folder with the project
#byte PORTB =0xF81 // We attribute to the memory posistion 0xF81
                                  // the name PORTB
                                  // This means that we define a 8 bit
                                  // variable whose value will be stored
                                  // to the memory position F81h
                                  // The memory position F81h is the PORTB
                                  // data register


void init(void);
void ext_int1(void);

void main(){
   init();           //initialization routine.
   while(TRUE){;}    //the main program does nothing
}

void init(){
   set_tris_d(0x00);        //PORTD is defined as output
```

Figure 2. CCS C Compiler, translation to machine code (hex file)

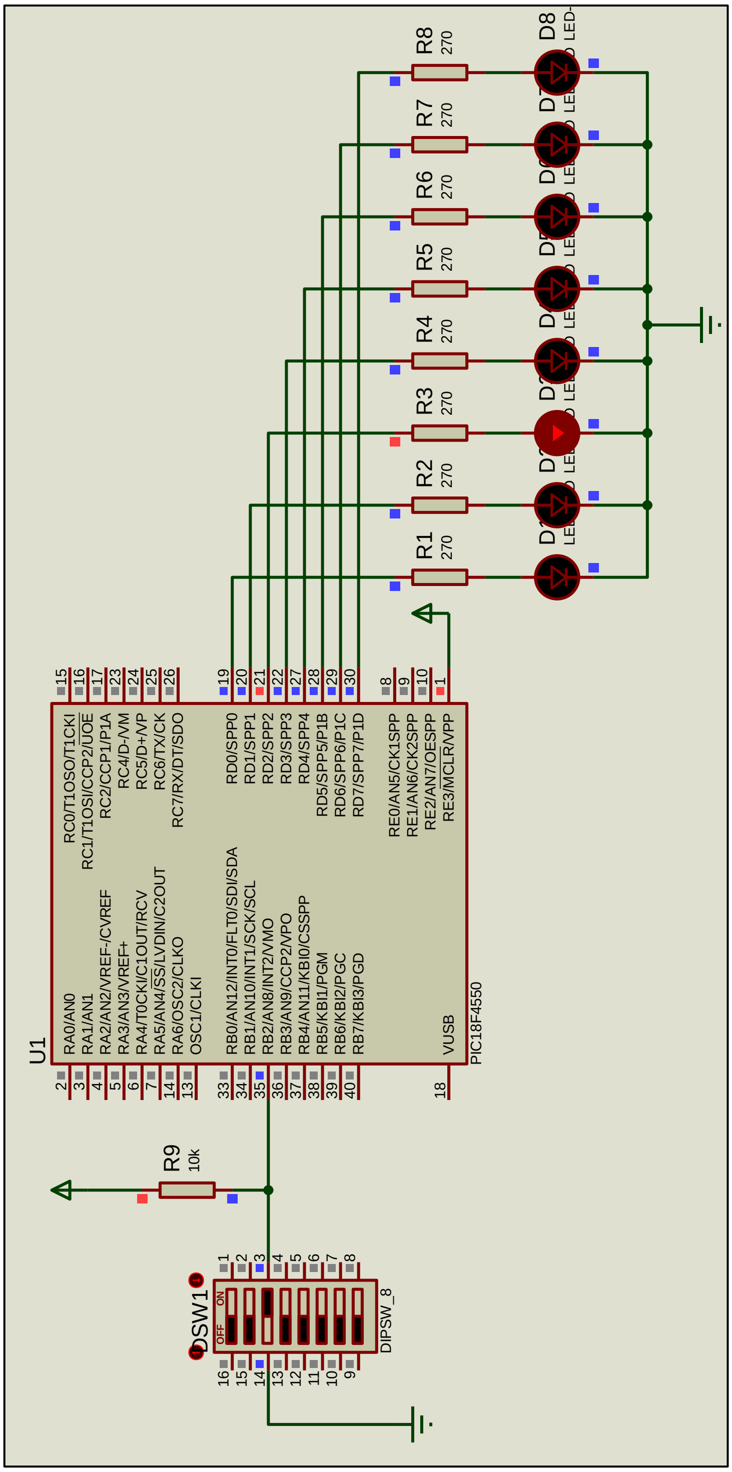
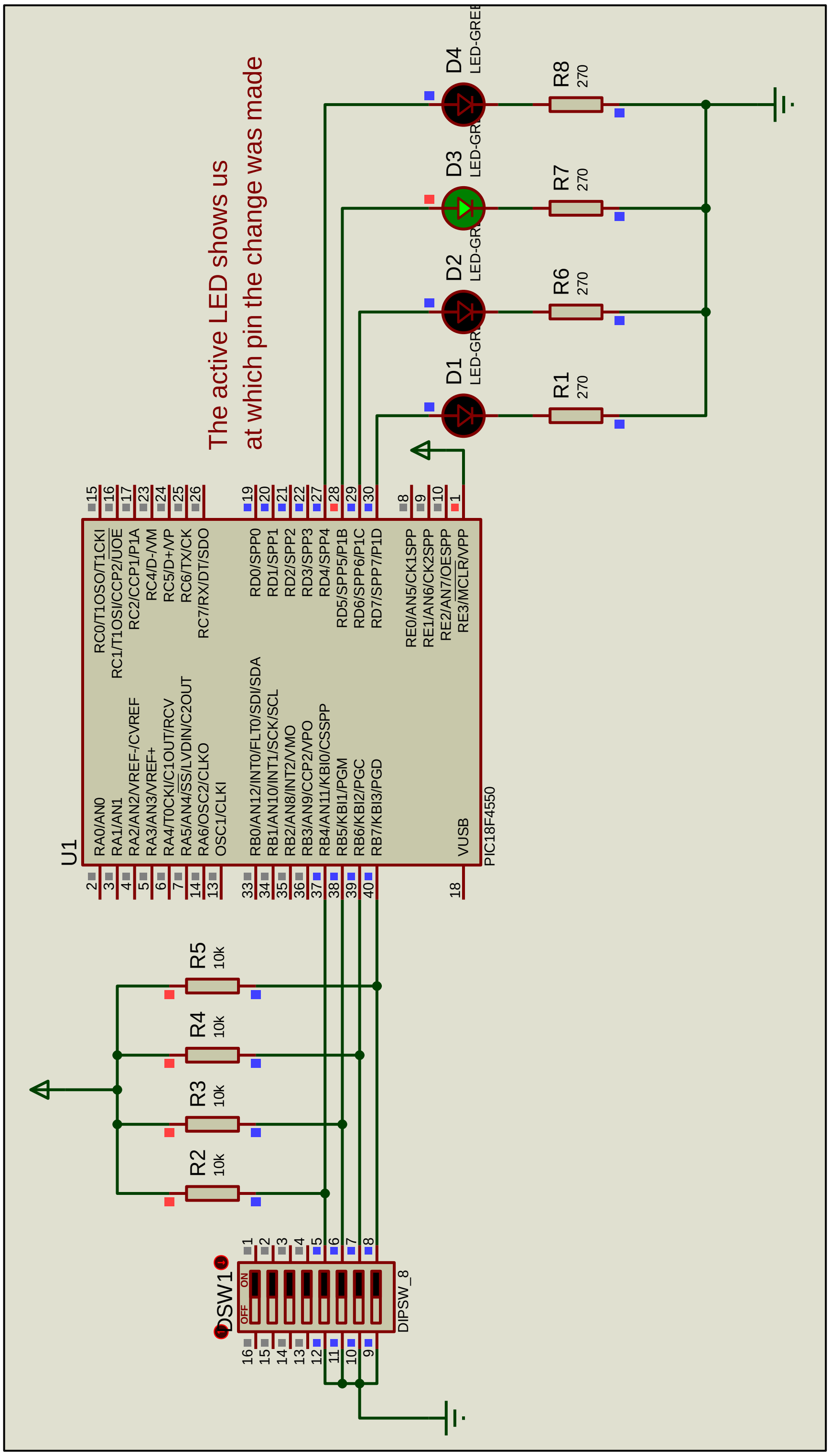
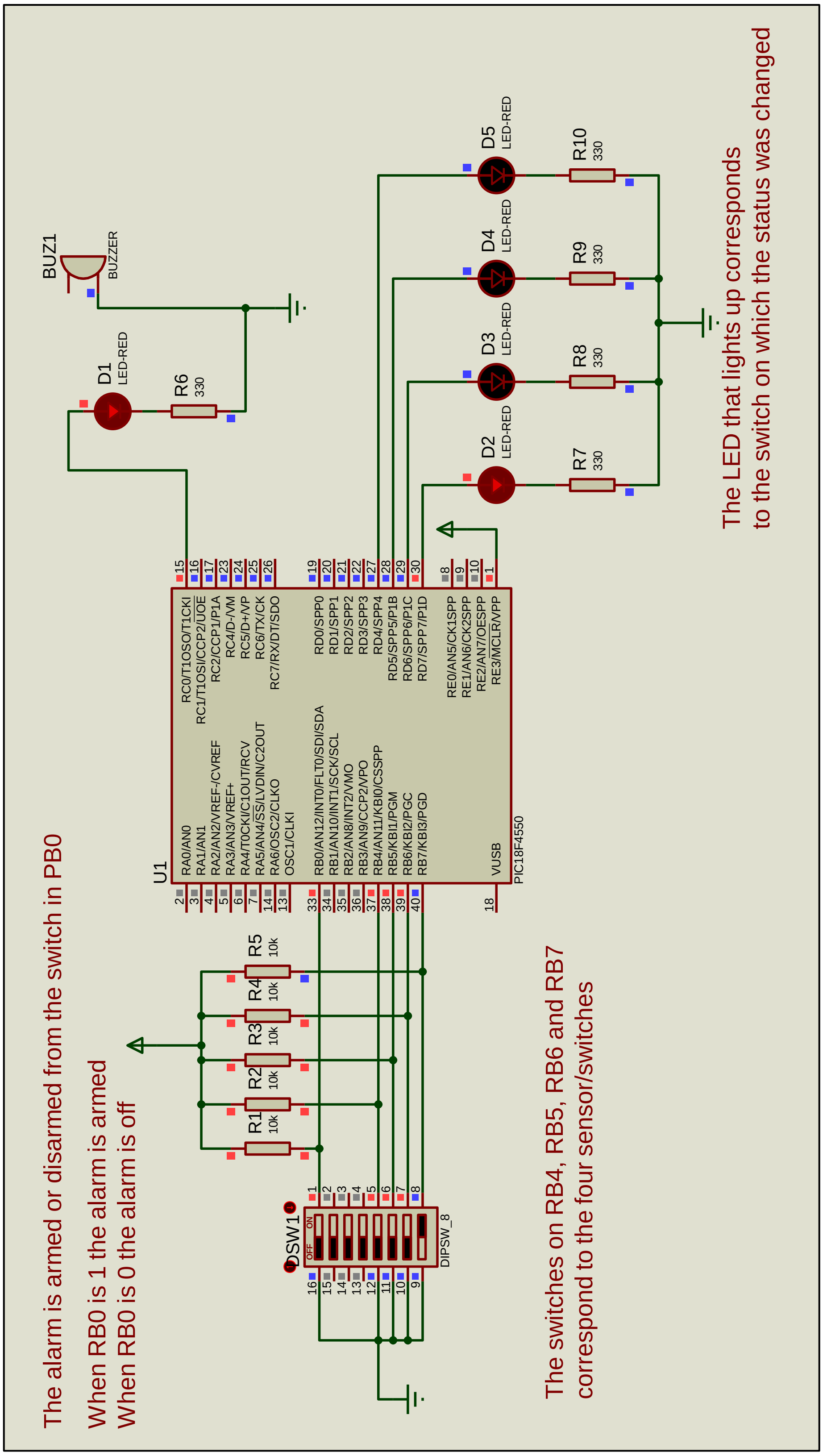


Figure 3. *INT2 and LEDs*



The active LED shows us at which pin the change was made

Figure 4. INTRB and LEDs


The alarm is armed or disarmed from the switch in PB0

When RB0 is 1 the alarm is armed
When RB0 is 0 the alarm is off

The switches on RB4, RB5, RB6 and RB7 correspond to the four sensor/switches

The LED that lights up corresponds to the switch on which the status was changed

Figure 5. A simple alarm system