ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

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

Practice leaflet: Module_2-9 SevenSegmentDisplay

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 Where other published and unpublished source materials have been used, these have been acknowledged.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 with seven segment displays.

Chapter 1: **Overview**

Table 1. Overview

Title / short summary	**9. Seven segment display**
Expected learning outcomes	• The student will be able to connect seven segment displays on the microcontroller • The student will be able to handle code for seven segment displays • The student will be able to design simple circuits with seven segment displays • The student will be able to load and animate a microcontroller program in the Proteus Design Suite
Keywords	Seven segment, display, LED, outputs
Duration	The duration of the module_2-9 is 3 hours • Presentation of the module_2-9 by the teacher, 30 minutes • 1st activity, three seven-segment displays, 30 minutes • 2nd activity, input binary number – output to seven segment displays, 45 minutes • 3rd activity, real time clock, 75 minutes
Involved	**The teacher:** Presents the slides associated with the module_2-9 and answers question **The students:**

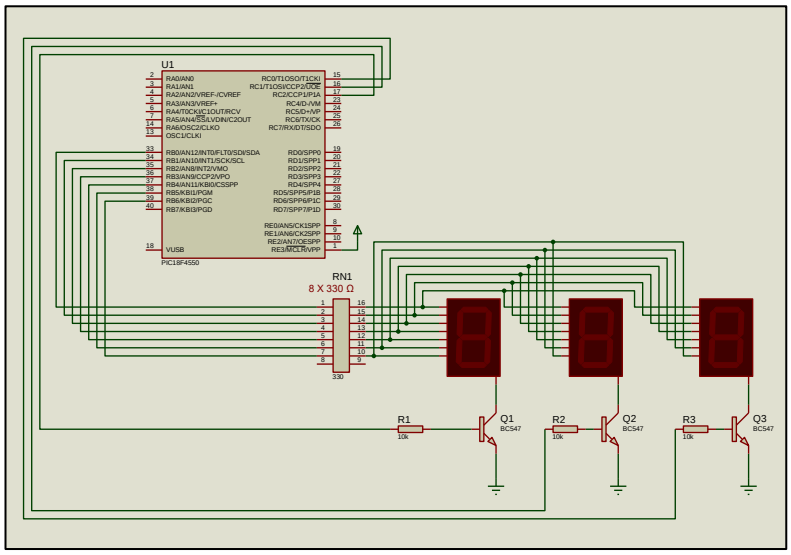
	Draw circuits in Proteus Schematic, write programs in C language, load programs to a microcontroller and run the simulation using the Proteus Design Suite
Assignment	At the end of the Module_2-9 will be given: • Open Project
Educational tools and equipment	• **Material:** PC • **Software:** CCS C compiler, 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, Module_2-2 and Module_2-8
Educational content	• CCS C Compiler manual (C Compiler Reference Manual) • MICROCHIP, PIC18F2455/2550/4455/4550 Data Sheet • Module_2-9 slides • Module_2-9 Evaluation leaflet • Module_2-9 Open project leaflet • Module_2-9 Programs, Schematic Proteus (Compressed folder)
Tips	***Tip.*** *Common anode vs common cathode seven segment display*

Chapter 2: **Activities**

2.1 Activity 1. Three seven segment displays

The purpose of the activity is to display the number 123 on three seven-segment displays.

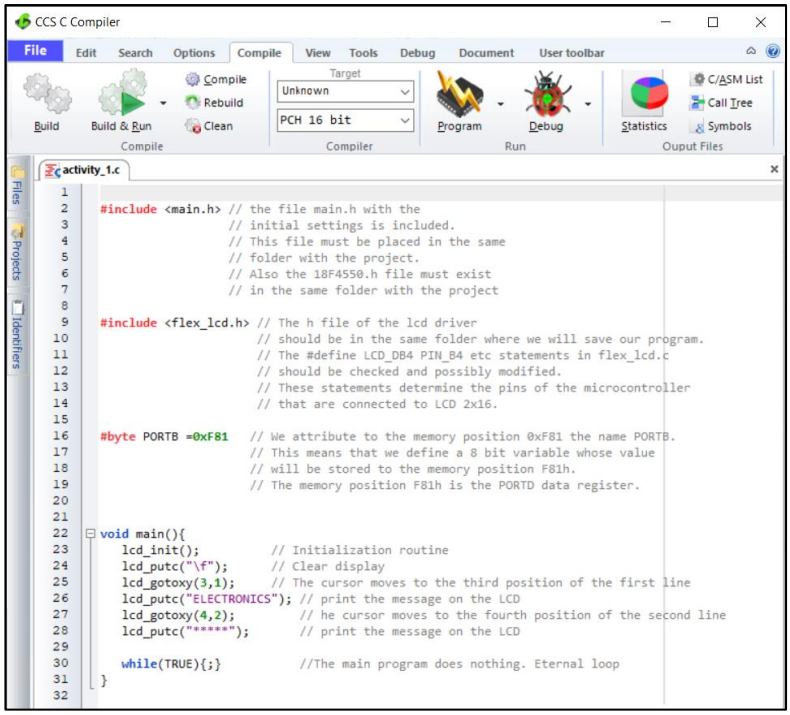
Table 2. Activity 1

Activity 1st (30 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 1 (12 minutes)	Draw the circuit of the picture in the Proteus Design Suite. Three common-cathode 7-segment displays are connected to PORTB of the 18F4550 microcontroller as shown in the figure. The base of the transistor corresponding to the right display is connected to PIN_C0, the base of the transistor corresponding to the middle display is connected to PIN_C1, and the base of the transistor corresponding to left display is connected to PIN_C2.  *Figure 1. Connections*

Step 2
(8 minutes)

Write in CCS Compiler the program in C language

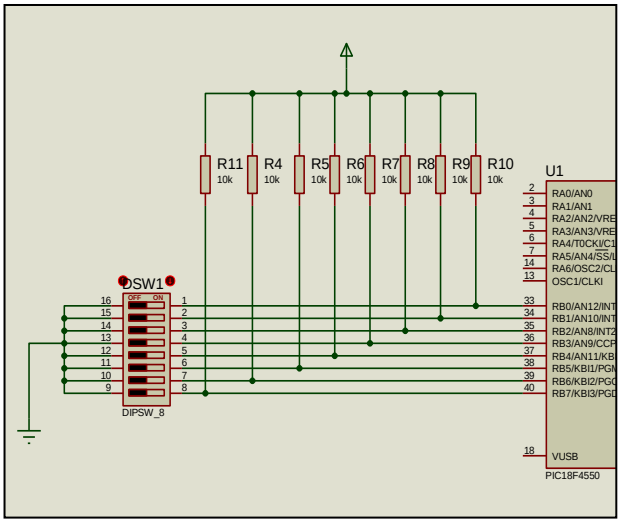
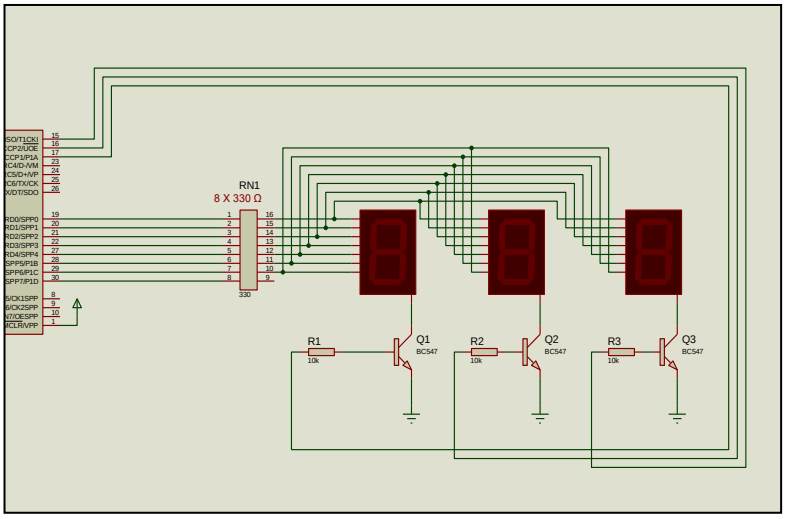
```
#include <main.h>
#byte PORTB=0xF81
//define PORTB data register
#byte PORTC=0xF82
//define PORTC data register

void main()
{
 set_tris_b(0x00);      // define PORTB as output
 set_tris_c(0x00);      // define PORTC as output

 while (TRUE){
 // the codes of each digit are sent every 5 ms
 // by activating the corresponding display

//……Number 3
PORTC=0b00000001; //activate the right display
PORTB=0b01001111; //code for "3"
delay_ms(5);

//…….Number 2
PORTC=0b00000010; // activate the middle display
PORTB=0b11011011; //code for "2".
delay_ms(5);

//……Number 1
PORTC=0b00000100; //activate the left display
PORTB=0b00000110; //code for "1"
delay_ms(5);
    }
}
```

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

2.2 Activity 2. Input binary number – output to seven segment displays

The purpose of the activity is to continuously read the value of PORTB which is used as an input and display its value in the decimal number system in the 3 seven-segment displays.

Table 3. Activity 2

Activity 2nd (45 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.  *Figure 3(a). Dip-switches and 7-segment displays*  *Figure 3(b). Dip-switches and 7-segment displays*

Step 2 (20 minutes)	Write in CCS C Compiler the program

```
#include <main.h>
#byte PORTB =0xF81
#byte PORTC =0xF82
#byte PORTD =0xF83

void main(){
   // Define PORTB as inpout
   // DEfine PORTC and PORTD as output
   set_tris_b(0xFF);
   set_tris_c(0x00);
   set_tris_d(0x00);

   int8 table[16] ={
   //Table of codes to display in a 7-segment
display
         0b00111111, //0
         0b00000110, //1
         0b01011011, //2
         0b01001111, //3
         0b01100110, //4
         0b01101101, //5
         0b01111101, //6
         0b00000111, //7
         0b01111111, //8
         0b01101111, //9
         0b01110111, //A
         0b01111100, //B
         0b00111001, //C
         0b01011110, //D
         0b01111001, //E
         0b01110001};//F

   int input_value;
   // variable to store the input value
   int monades;
   // variable to store the units of the input
value
   int decades;
   // variable to store the tens of the input value
   int ekatontades;
   // variable to store the hundredths of the input
value


    while (TRUE){
    // the codes of each digit are sent every 5 ms
    // by activating the corresponding display

      input_value=PORTB;
      ekatontades=input_value/100;
      decades=(input_value-ekatontades*100)/10;
      monades=input_value-ekatontades*100-
decades*10;

      PORTC=0b00000100;      //activate the left
display
```

	PORTD=table[ekatontades]; delay_ms(5); PORTC=0b00000010; //activate the middle display PORTD=table[decades]; delay_ms(5); PORTC=0b00000001; //activate the right display PORTD=table[monades]; delay_ms(5); } }
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. Real time clock

In this activity we want to develop a real time clock. The system will work as follows: Initially the indication will be 12:00. The indicator will be done in two phases. For one second the hour will be displayed and for one second the minutes will be displayed. In order to be able to separate the indication between hours and minutes, when the indication on the two rightmost indicators is the hour, the leftmost indicator should show the "Ω", while when the indication is the minutes, the leftmost indicator should have the indication "Π".

Table 4. Activity 3

Activity 3rd (75 minutes)	**Step 1.** The circuit is drawn at the Proteus Design Suite. **Step 2.** Configuration for Timer0 **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 (10 minutes)	Draw the circuit of the Picture 1 in the Proteus Design Suite
Step 2 (10 minutes)	The Timer0 overflow interrupt method will be used to measure the time. Specifically, 5ms will be used as a time base, that is, an interruption from Timer0 will occur every 5ms. When 200 interruptions have passed, 1 second will have elapsed. Let's assume that: Fclock=48MHz και Prescaler=1 To calculate what value timer0 should take we need to solve the equation: $$(65536 - y) * \frac{1}{\frac{F_{clock}}{4}} * Prescaler = 5ms <=> \mathbf{y} \approx \mathbf{5536}$$ Where y is the initial value that Timer0 should have.
Step 3 (40 minutes)	Write in CCS C Compiler the program `#include <main.h>` `#byte PORTB =0xF81` `#byte PORTC =0xF82` `#byte PORTD =0xF83` `// Variable definitions` `int8  des=0;` `//variable to select one of the three displays` `int8 seconds=0;` `int8 minute=0;` `int8 hour=12;` `int8 counter=200;` `//variable to count interrupts` `int1 flag=0;` `//choose to display hours or minutes` `//Table of codes to display in a 7-segment display` `int8 table[16] =  {  0b00111111, //0` `0b00000110,  //1` `0b01011011, //2` `0b01001111, //3` `0b01100110, //4` `0b01101101, //5` `0b01111101, //6` `0b00000111, //7` `0b01111111, //8` `0b01101111, //9` `0b01101011, //Ω` `0b00110111, //Π` `};`

```
int8 dig[3] = {1,2,4};
//Table for driving a single display from PORTC
//PORTC applies 5V to the base of only 1 of the 3
transistors


// Function declaration
void timer0_int(void);
void init (void);

void main()
{
     init();
     while (1){ }
}


// Interrupt Service Routine
#INT_TIMER0 HIGH
void timer0_int(void){
   int16 mon,dec,eka;
   //variable to display digits in the 7-segment
displays

   set_timer0(5536);
   //Timer0 initial value to have interrupts occur
every 5ms

   counter--;
   //Counter is decremented by 1 and reset every
   //200 * 5 msec = 1 sec

   if (counter == 0){
      seconds++;
      counter = 200;
      flag^=1;
      if (seconds > 59){
         seconds = 0;
         minute++;
         if (minute > 59){
             minute = 0;
             hour++;
         }
         if (hour >24){
             hour = 0;
         }
      }
   }
   if (flag == 0){
      //if flag=0 then the minutes are displayed
      dec = (int8)minute / 10;
      mon =  minute - dec * 10;
      eka = 11;
   }
   if (flag == 1){
      //if flag=1 then the hours are displayed
      dec = (int8)hour / 10;
      mon =  hour - dec * 10;
      eka = 10;
   }
```

	`` des = ++des%3; /* This variable takes the values sequentially 0, 1, 2, 0, 1, 2, 0, 1, … so that they are selected from the dig[des] array sequentially the values … 0000 0001, 0000 0010, 0000 0100 … and to activate the displays in order */ PORTC = dig[des]; if (des==0){ PORTB = table[mon]; } if (des==1){ PORTB = table[dec]; } if (des==2){ PORTB = table[eka]; } } // Initialization routine void init (void){ set_tris_b(0x00); set_tris_c(0x00); PORTB = 0; PORTC = 0; counter = 200; seconds = 0; minute =0; hour = 12; des =0; flag = 0; SETUP_TIMER_0(T0_INTERNAL | T0_DIV_1); set_timer0(5536); enable_interrupts(INT_TIMER0); enable_interrupts(GLOBAL); } ```
Step 4 (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 5 (10 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
- ☞ The programs in C was written in CCS C compiler.
- ☞ 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

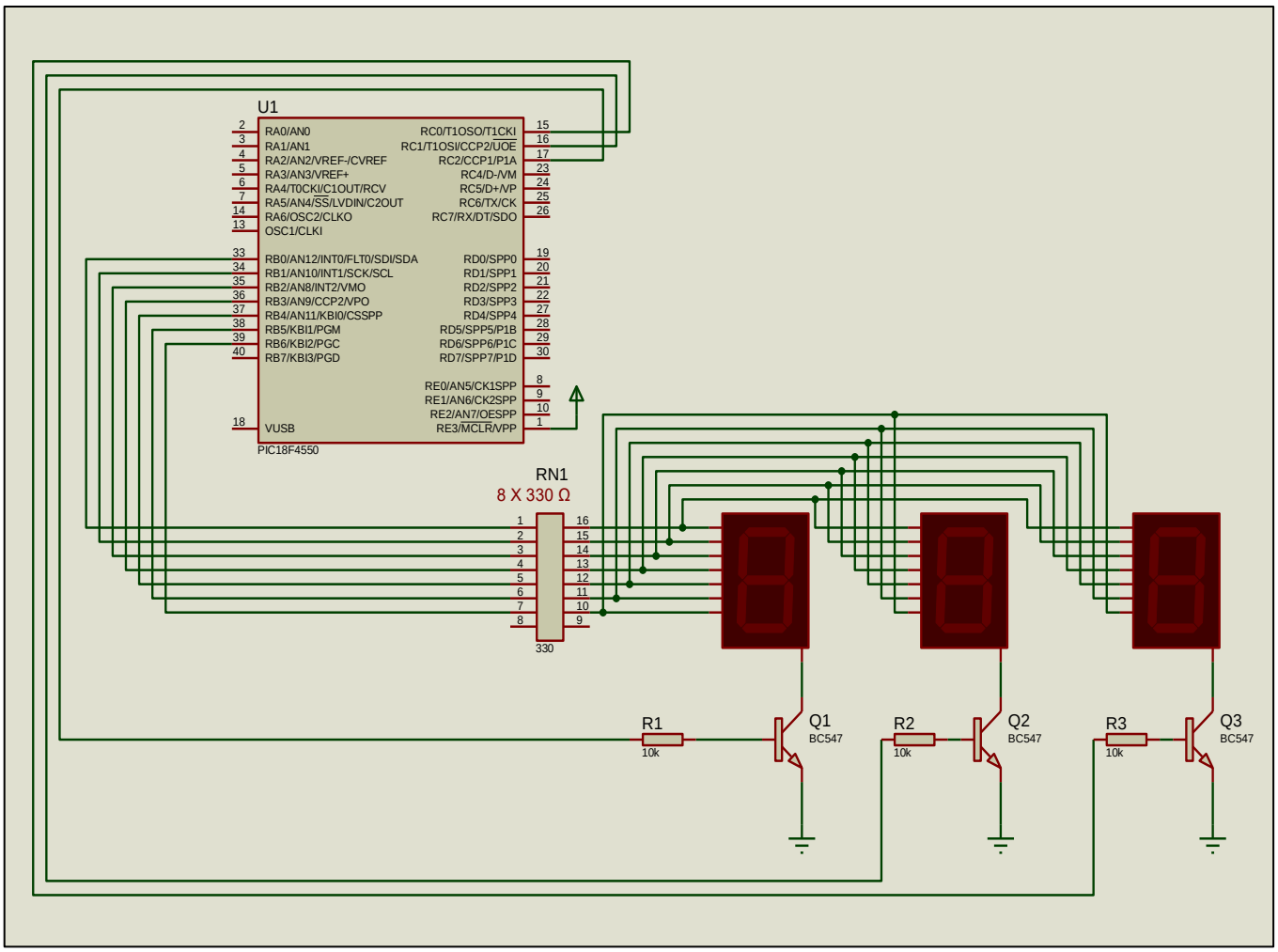


Figure 1. Connections

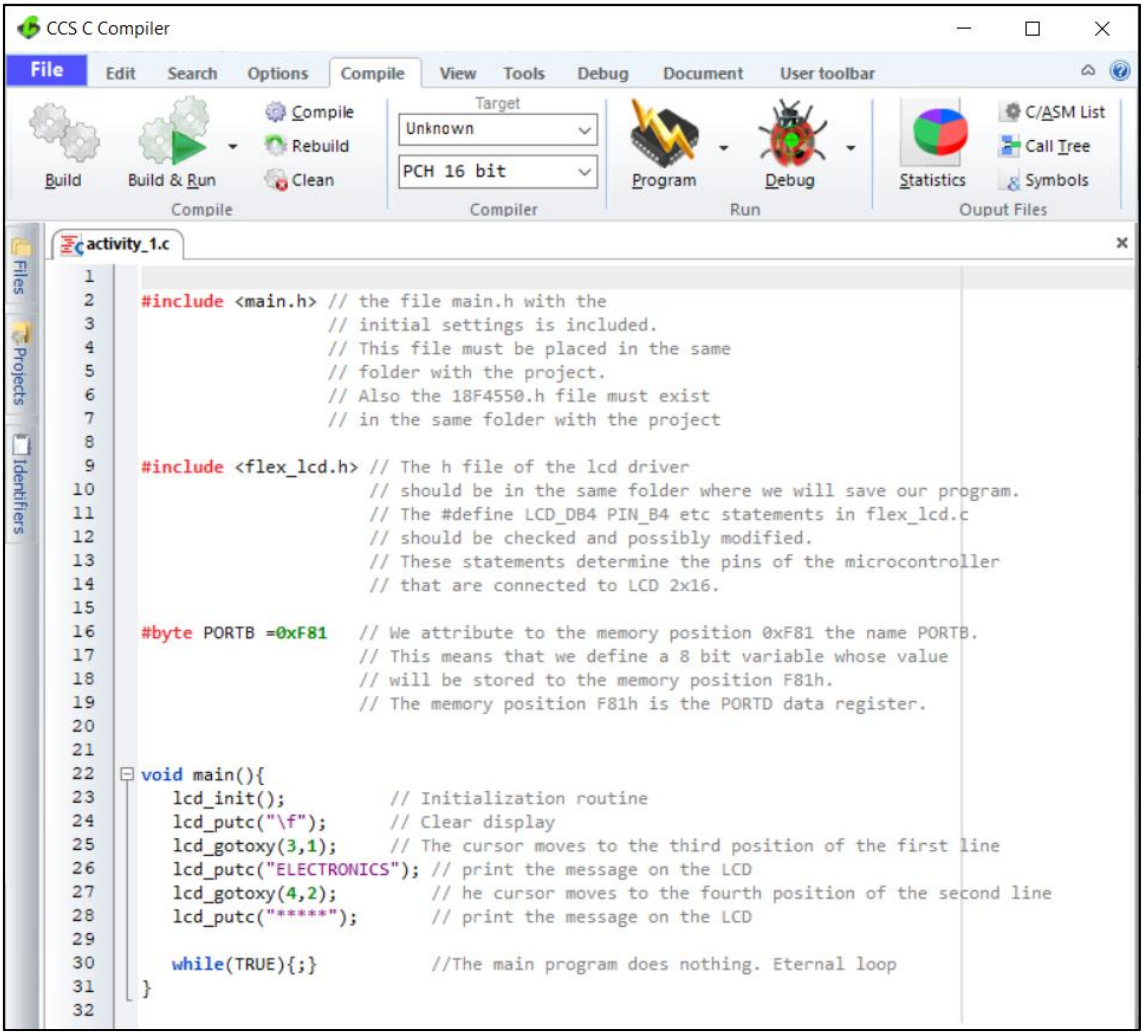


Figure 2. CCS C Compiler, translation to machine code (hex file)

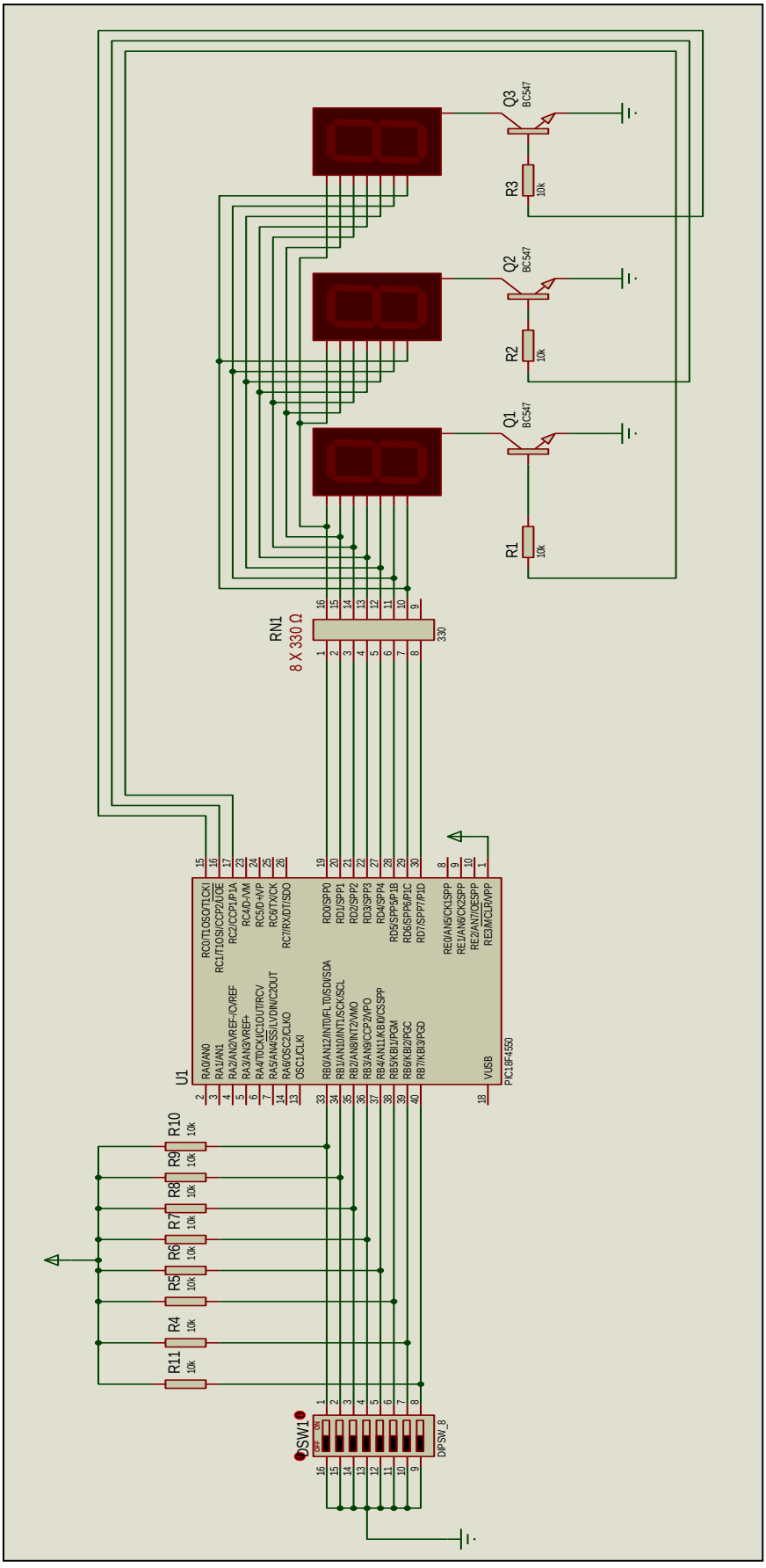


Figure 3. Dip-switches and 7-segment displays