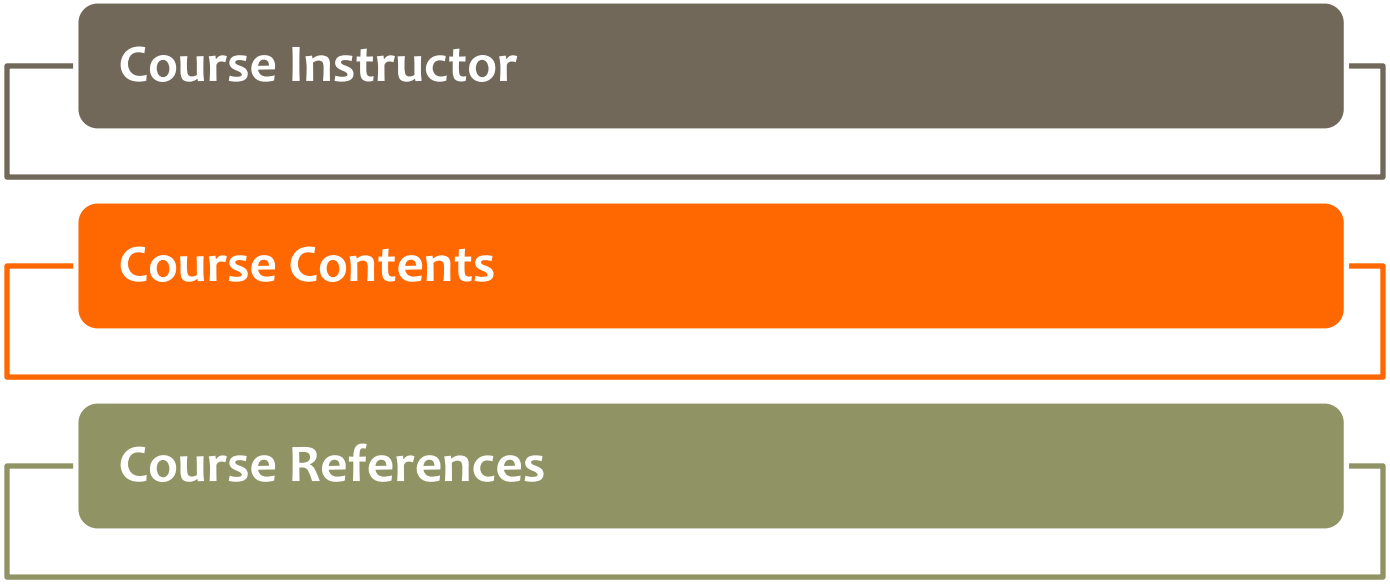
Microcontrollers in practice

Instructor:

Dr. Ahmad El-Banna

SUMMER 2016

Preface



Course Instructor

- **Dr. Ahmad EL-Banna**
  - B.Sc. in Telecommunications and Electronics, Fac. of Eng. at Shoubra, Benha Univ. 2005.
  - 9-month Diploma in Embedded Systems, ITI, 2008.
  - M.Sc. in Telecommunications and Electronics, Fac. of Eng. at Shoubra, Benha Univ. 2011.
  - PhD. in Telecommunications and Electronics, E_JUST Univ., 2014.
  - Visiting Researcher , Wireless Communications Lab, Osaka University, 2013-2014.
  - Find more at
    - www.bu.edu.eg/staff/ahmad.elbanna

Your turn !

- About You
  - Name
  - Graduation
    - Year
    - Univ

Course Contents

- Concepts and Principals of Embedded Systems.
- Basics of Microcontrollers.
- Design of various Embedded Systems.
- Real-Time Concepts for Embedded Systems.
- Developing Embedded Systems.

Course References

- John Catsoulis, **Designing Embedded Hardware,** 2005.
- Qing Li and Carolyn Yao, **Real-Time Concepts for Embedded Systems**, 2003.
- Michael Barr, **Programming Embedded Systems in C and C++**, 1999.
- Muya, **Interfacing PIC Microcontrollers**, 2006.

Detailed Contents

INTRODUCTION TO MICROCONTROLLER

Headlines

- What's Microcontroller?
- Where is it?
- What can it do?
- Basic terminology
- A deep look
- IDEs, welcome
- Hello World ☺

What is a Microcontroller?

- Simply a computer on a chip.
- A single chip, self-contained computer which incorporates all the basic components of a personal computer on a much smaller scale.
  - The main consequence of the microcontroller's small size is that its resources are far more limited than those of a desktop personal computer.
  - A microcontroller is not the same as a microprocessor.
    - A microprocessor is a single chip CPU used within other computer systems.
    - A microcontroller is itself a single chip computer system.

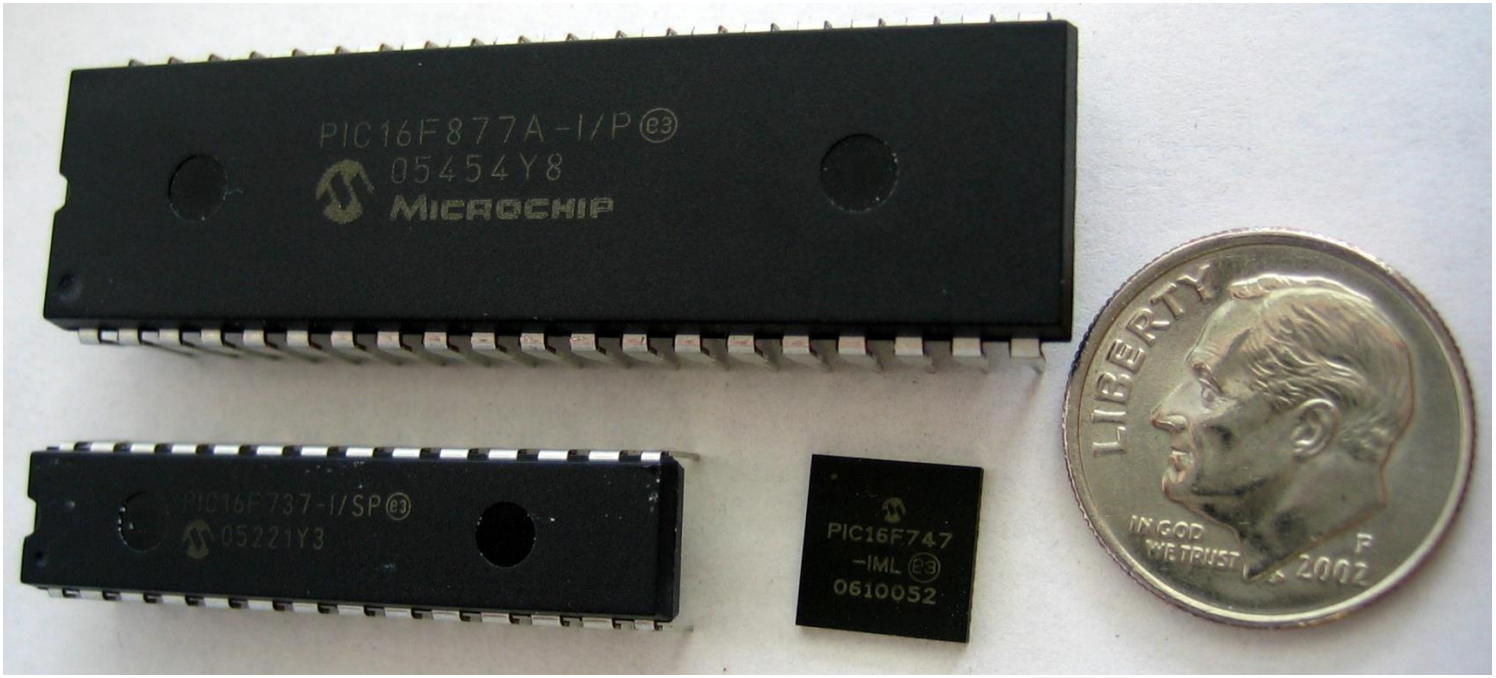
Where is it ?

- Appliances
- Automobiles
- Scientific instruments
- Computer peripherals
- ….. i.e. everywhere

What can it do

- Automatic control
- Data storage
- Data transfer
- ..

A close look


PIC16F877A-I/P
05454Y8
MICROCHIP
PIC16F737-I/SP
05221Y3
PIC16F747
-IML
0610052
LIBERTY
IN GOD WE TRUST
2002

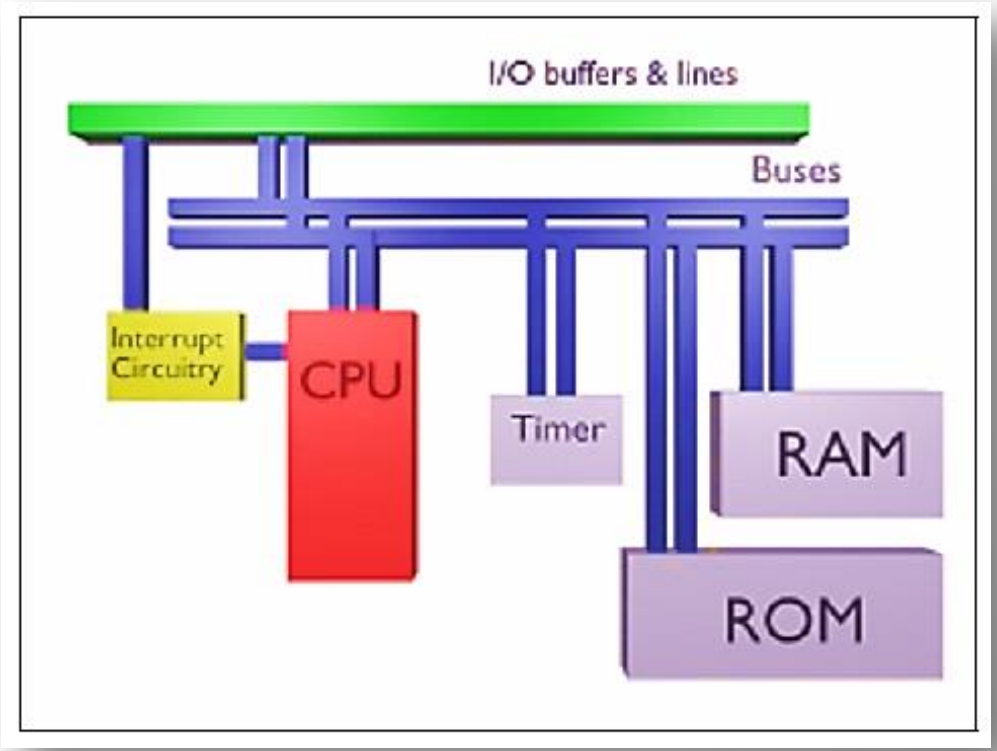
Basic Terminology

- You should know the following terms:
  - Embedded Systems.
  - Real-time systems (Hard vs. Soft)
  - Buses (address, data, control)
  - Interrupts (hardware vs. software)
  - CISC/RISC processors
  - Memory (RAM/ROM/EPROM/EEPROM/Flash)
  - Registers
  - Stack
  - Assembly & Machine code

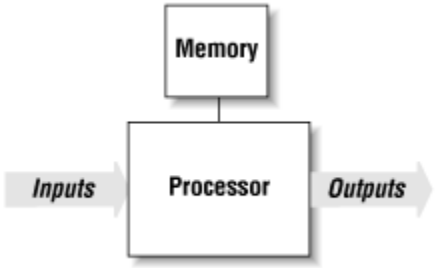
A deep look

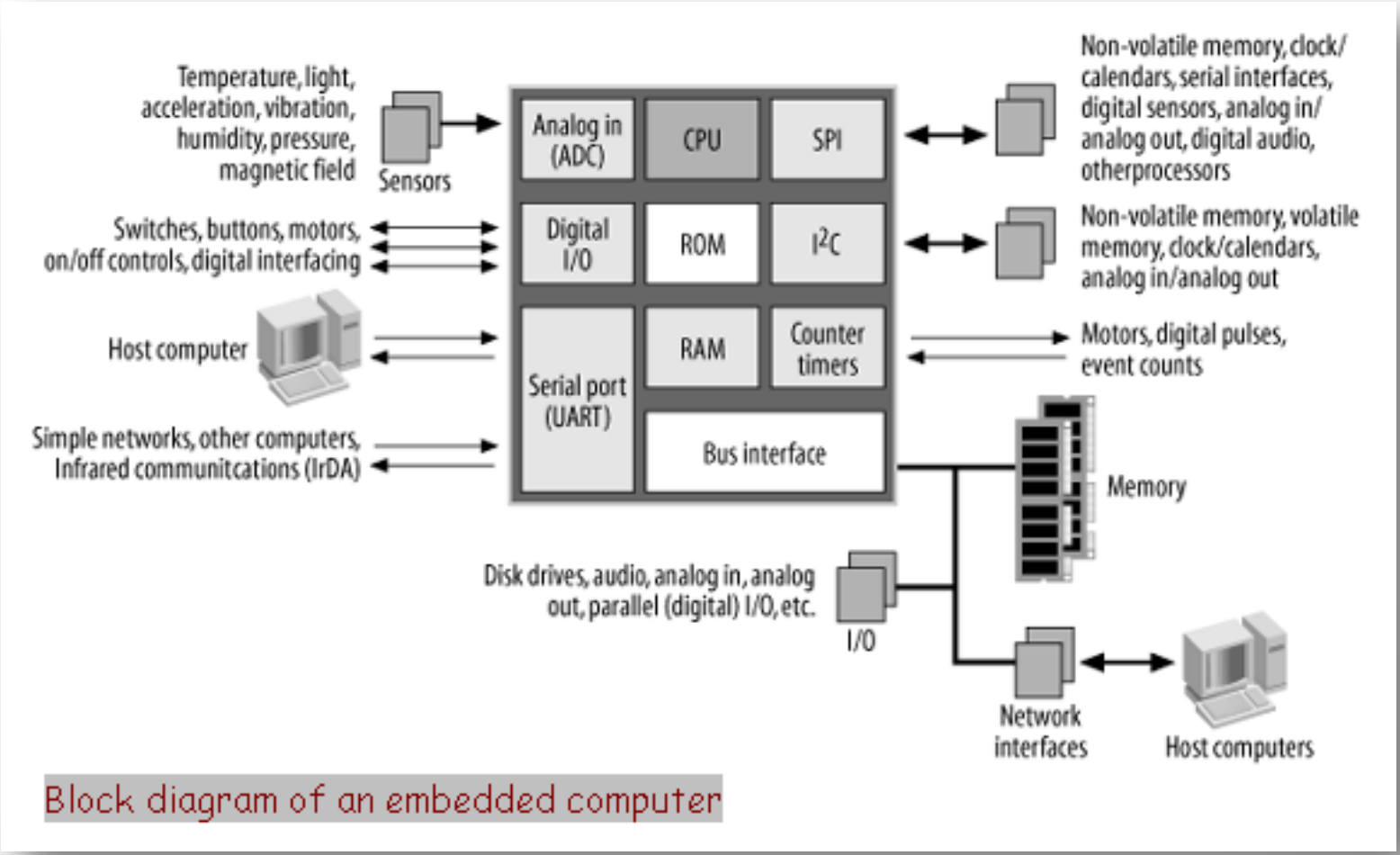
A microcontroller has seven main components:

- Central processing unit (CPU)
- ROM
- RAM
- Input and Output
- Timer
- Interrupt circuitry
- Buses



A deep look

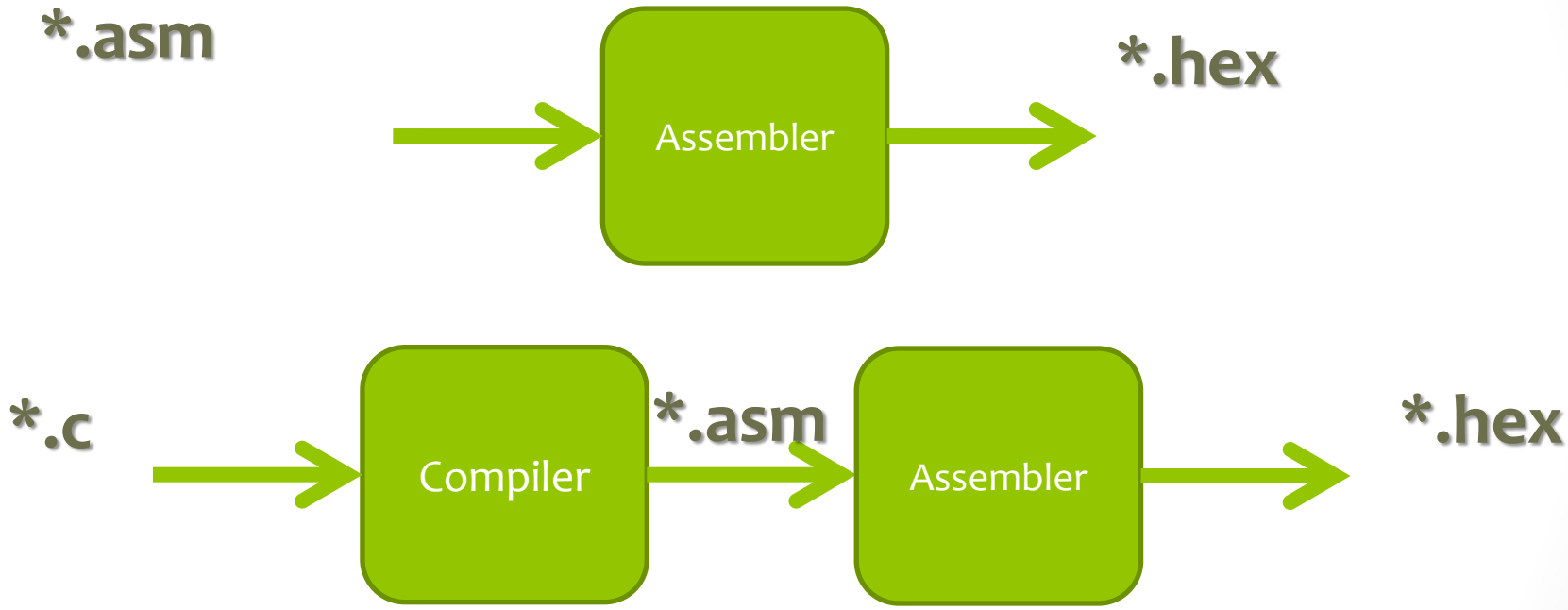




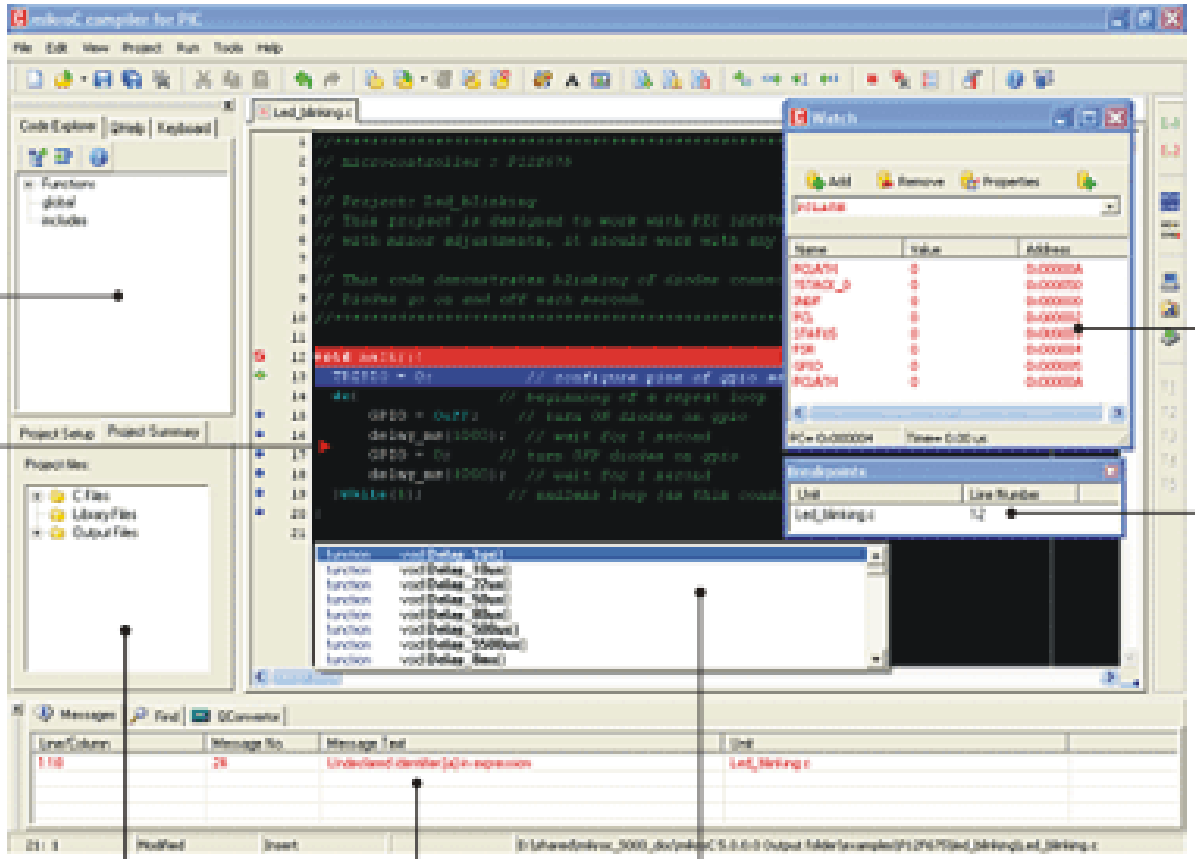
Block diagram of an embedded computer

IDE OVERVIEW

IDE

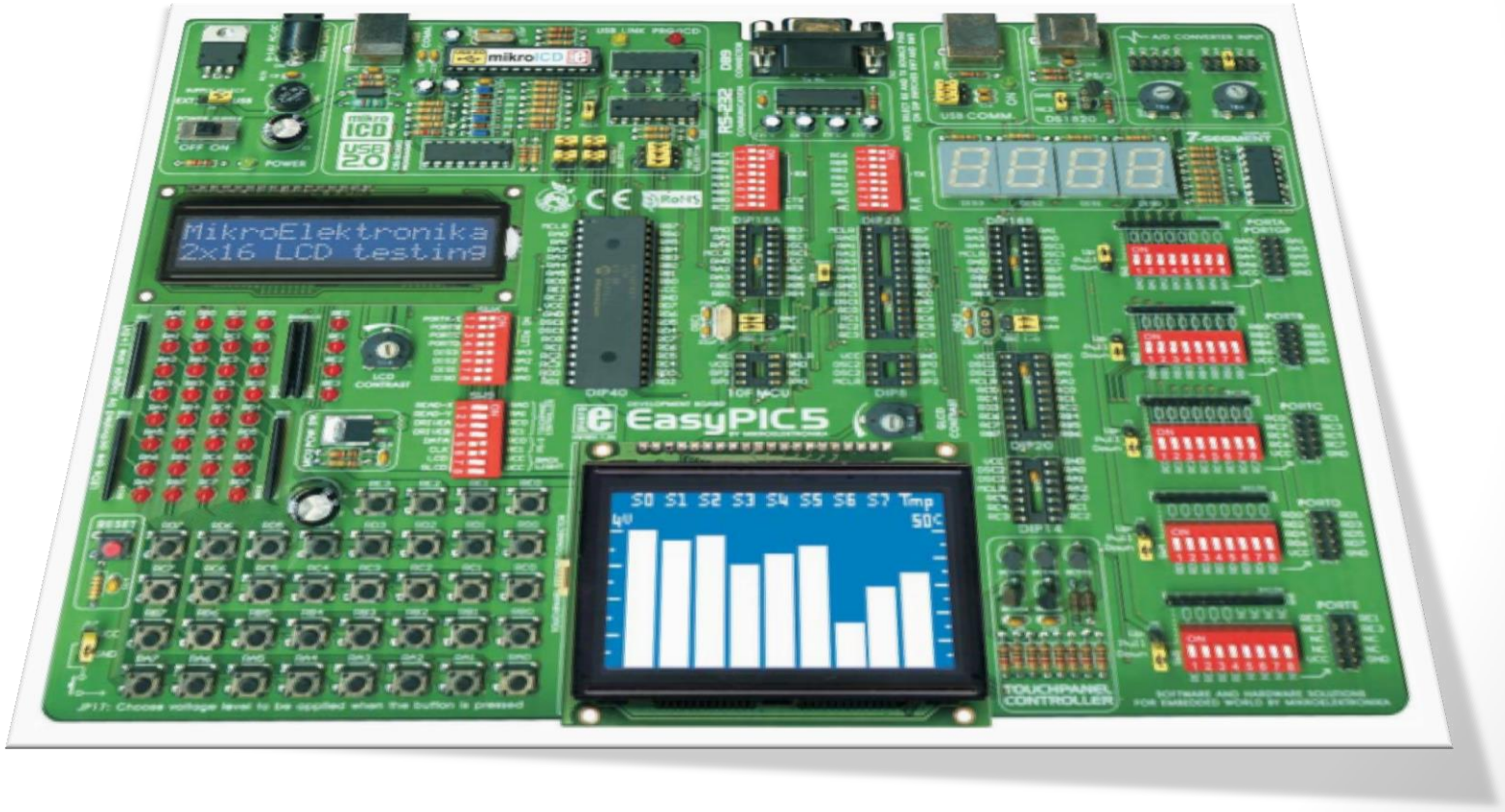


IDE

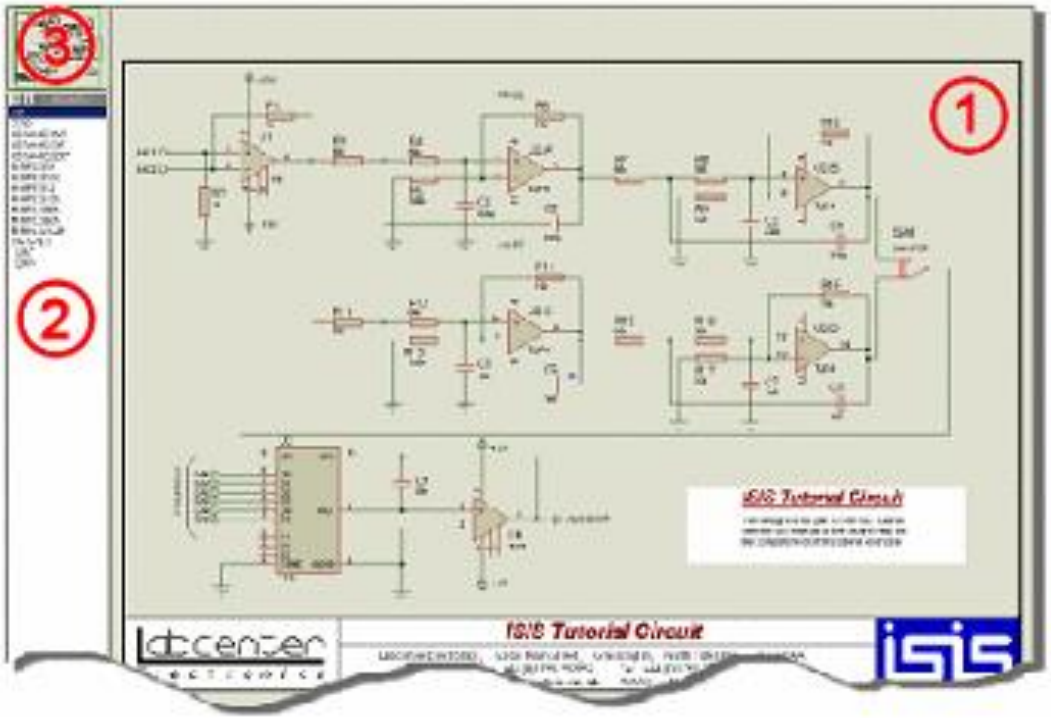

Code Explorer
Code Editor
Debugger watch window
Breakpoints Window
Project Summary
Error Window
Code Assistant

EasyPic5 Development Board

Features



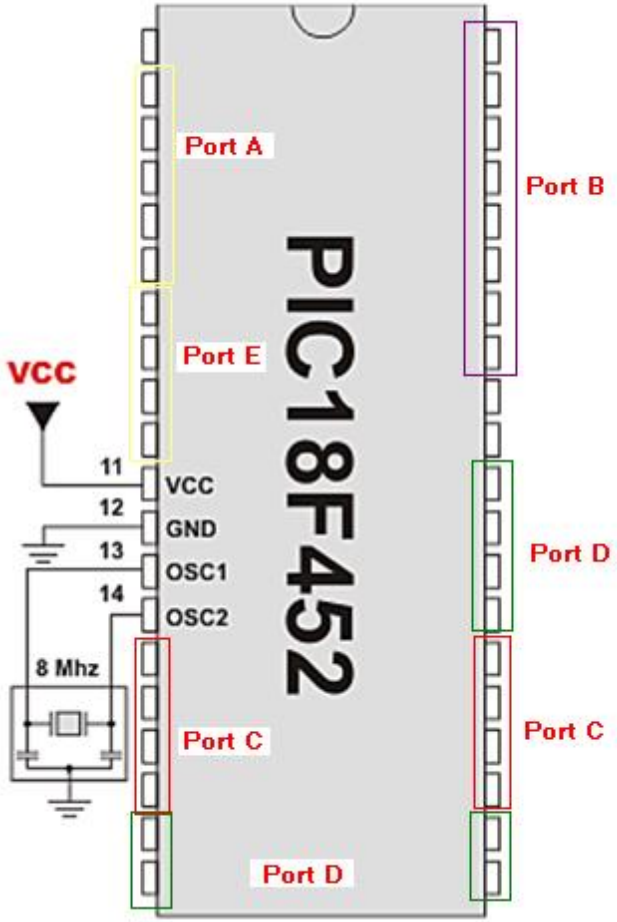
Proteus Design Suite



1. **Editing Window**
2. **Object Selector**
3. **Overview Window**

BLINKING LEDS

Basic Circuit

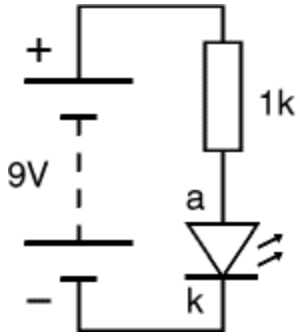


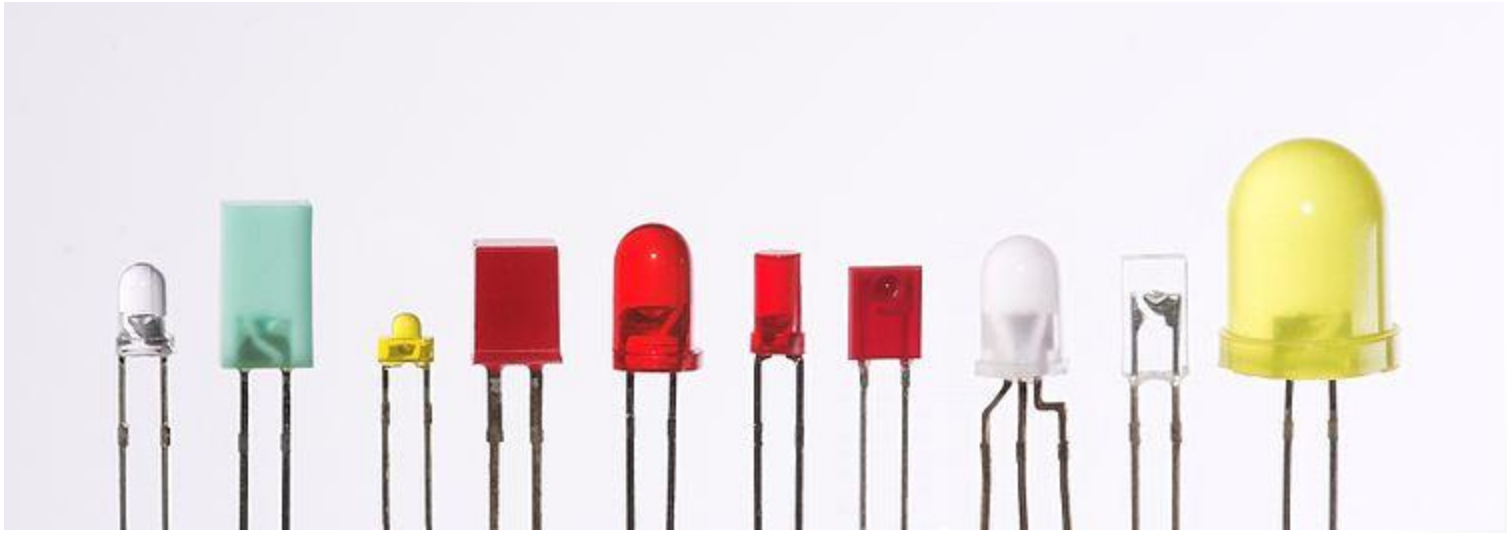
How to control?

- Microcontroller uses **ROM** for storing the programs, constants,… etc
- It uses **RAM** for storing variables
- It uses **registers** for controlling its peripherals and hence its operation.

LEDs

Symbol





Led Displays

Bargraph

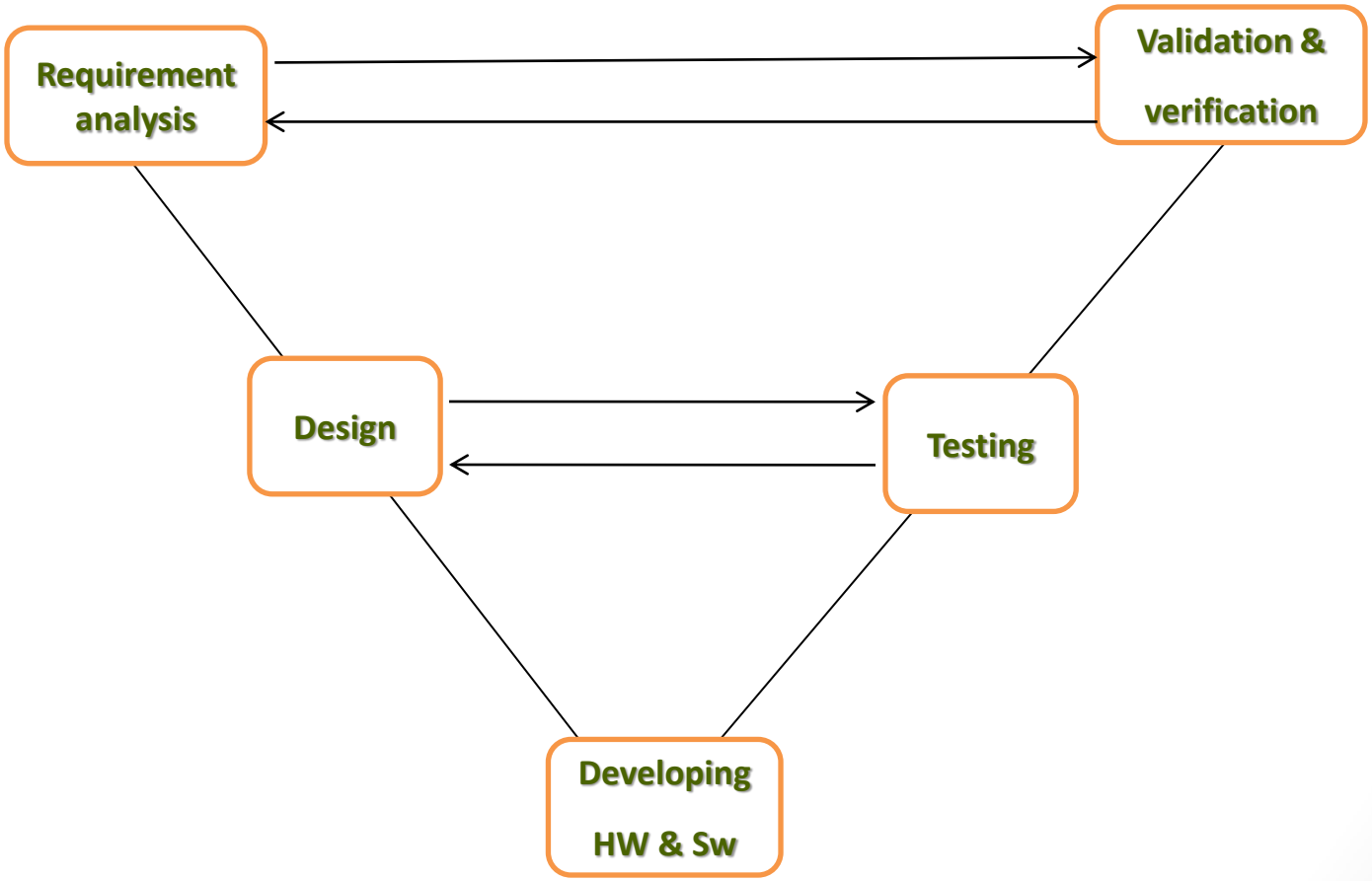
7-segment

Starburst

Dot matrix

Photographs © Rapid Electronics

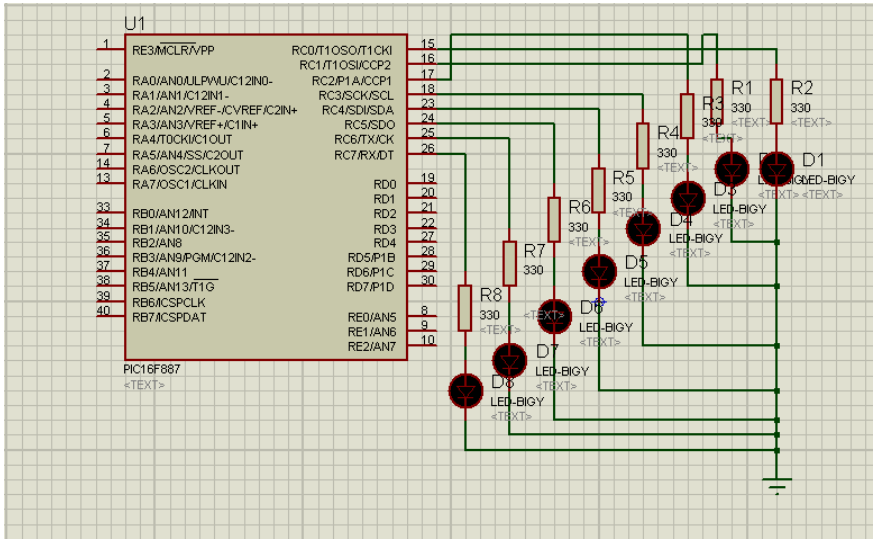
Project Managing

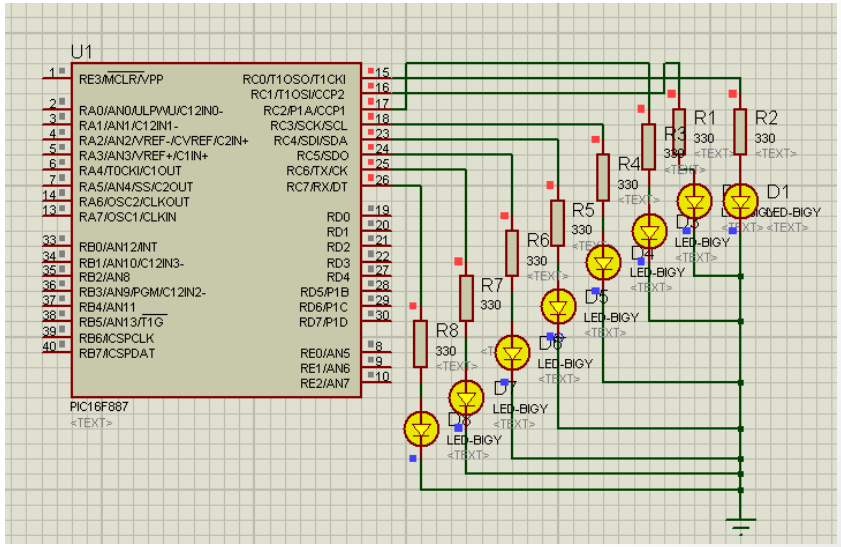


Project1

- Requirement analysis
  - Toggle leds on port c every 1 sec
- Design
  - Need a micro, crystal, resistors and leds
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj1_leds1





PORTS Registers

- TRISX register is used to configure port x direction as output/input
  - Ex: TRISB = 0; // configure PORTB as output
  - Ex: TRISB = 255; // configure PORTB as input

- PORTX register contains the values of port x pins in reading/writing operations
  - Ex: PORTC = 0xF0; // initialize PORTC pins to be 11110000
  - Ex: PORTC = 0b01010101; // initialize PORTC pins to be 01010101

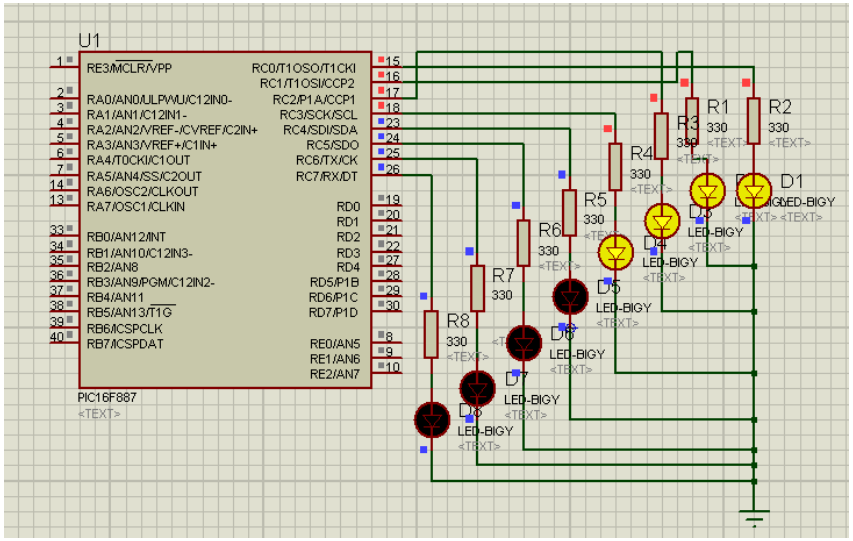
Pseudo code

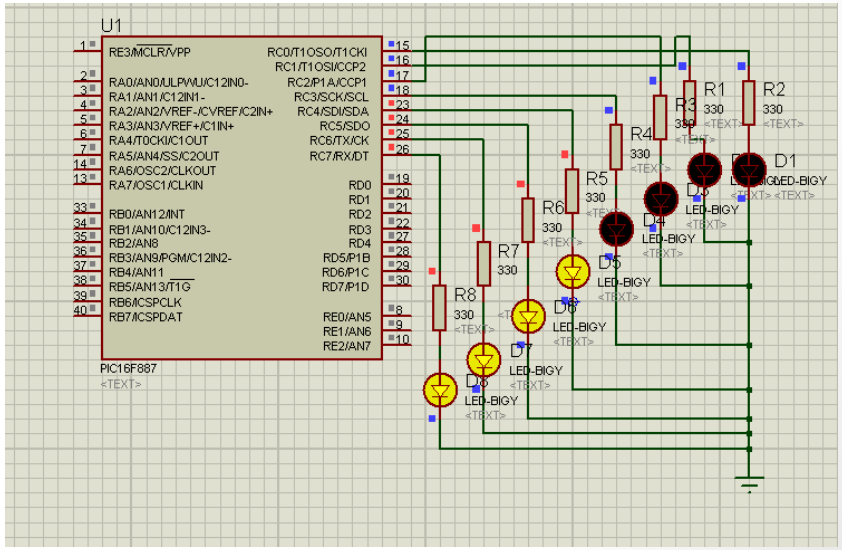
- Start
- set port direction as output
- Initialize port value
- Toggle the port
- Delay to watch
- Loop infinitely

code

```
1  /*
2   * Project name:
3       LED_Blinking (Simple 'Hello World' project)
4   * Copyright:
5       (c) MikroElektronika, 2005-2008
6   * Description:
7       This is a simple 'Hello World' project. It turns on/off diodes connected to
8       PORTC. It uses bitwise negation to toggle PORTC pins.
9   * Test configuration:
10      MCU:              PIC16F887
11      Dev.Board:        EasyPIC5
12      Oscillator:       HS, 08.0000 MHz
13      Ext. Modules:     -
14      SW:               mikroC v8.0
15  * NOTES:
16      None.
17  */
18
19 void main() {
20   PORTC = 0;                 // Initialize PORTC
21   TRISC = 0;                 // Configure PORTC as output
22
23   while(1) {
24     PORTC =  ~PORTC;         // toggle PORTC
25     Delay_ms(1000);          // one second delay
26   }
27 }
28
```

Proj1_leds2



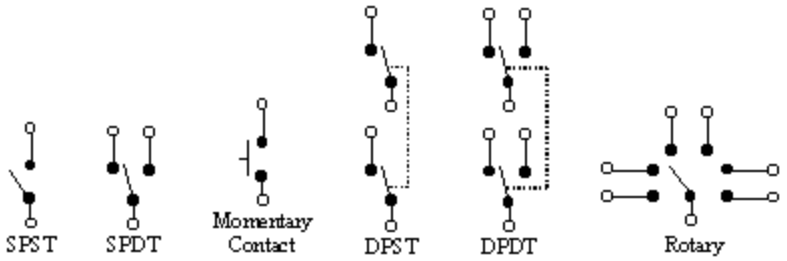


code

- Your turn ☺

BUTTONS & SWITCHES

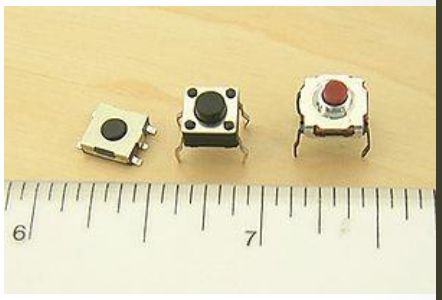
Buttons and switches

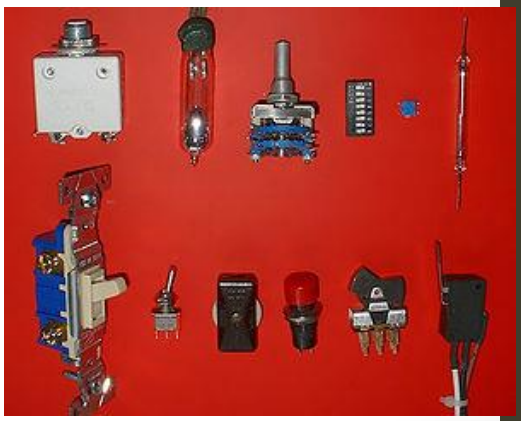


S:Single, P: Pole, D: Double, T: Throw

Temperature switch

Liquid level switch





Toggle switch

Pushbutton switch

Selector switch

Joystick switch

Lever actuator limit switch

Proximity switch

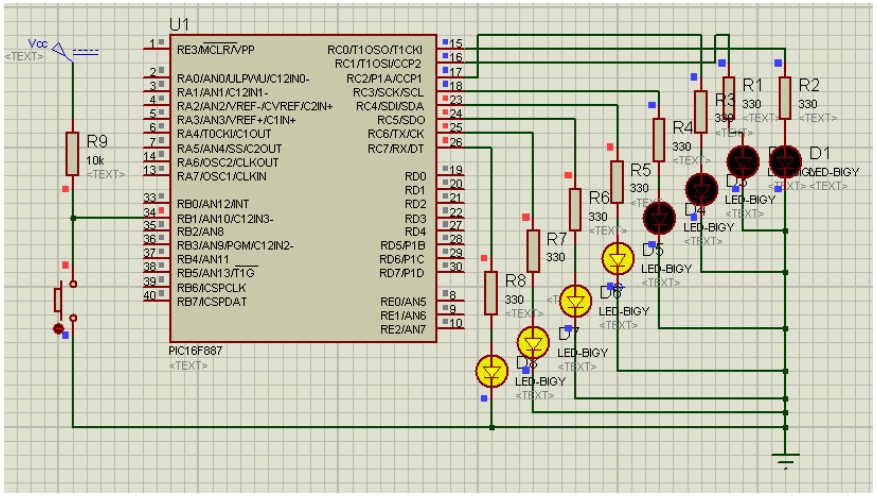
Speed switch

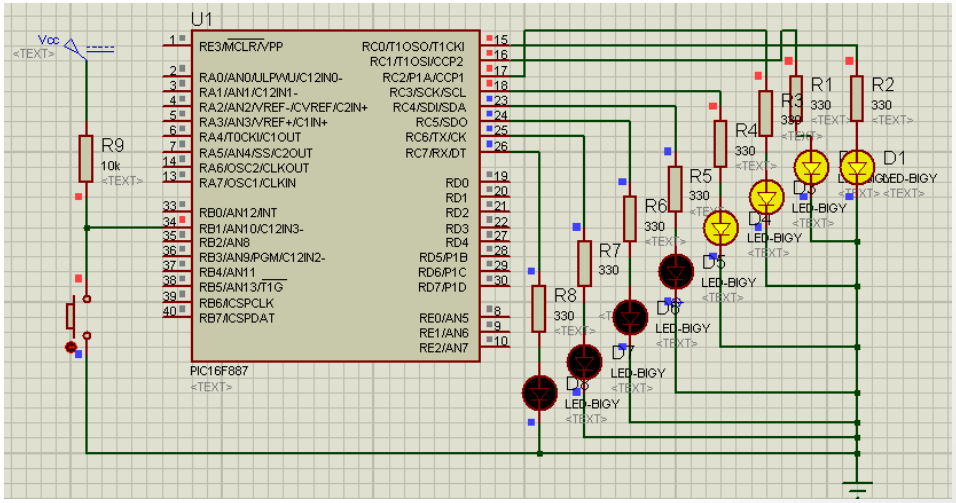
Pressure switch

Project2

- Requirement analysis
  - Toggle leds on port c when push button on RB1 is pressed
- Design
  - Need a micro, crystal, resistors, leds and a push button with pull up resistor
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj2_PB1





Pseudo code

- Start
- Configure port c direction as output
- Configure port B direction as input
- Initialize port value
- If key pressed, Toggle the port
- Delay to watch
- Loop infinitely

code

proj2_PB1

```
11      MCU:              PIC16F887
12      Dev.Board:        EasyPIC5
13      Oscillator:       HS, 08.0000 MHz
14      Ext. Modules:     -
15      SW:               mikroC v8.0
16  * NOTES:
17      None.
18 */
19
20
21
22 void main() {
23   ANSEL  = 0;                          // Configure AN pins as digital I/O
24   ANSELH = 0;
25   PORTC = 0xF0;                           // initialize PORTC
26   TRISC = 0;                           // configure PORTC as output
27   TRISB = 255;                         //// configure PORTB as input
28
29   do {
30     if (PORTB.F1 == 0) {                     // if PB is pressed
31       PORTC = ~ PORTC;
32       Delay_ms(100);     // to ignore bouncing
33     //Delay_ms(1000);    // indicate a suitable value
34     }
35
36   } while (1);
37 }
38
39
```

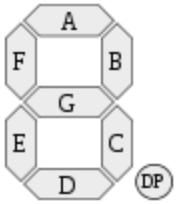
Proj2_PB2

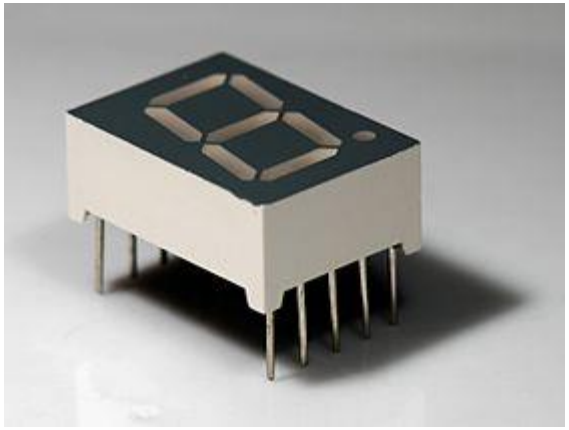
- Increment leds on every key press

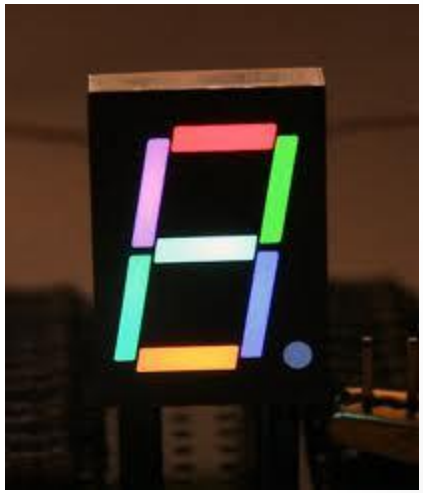
7 SEGMENTS

8 segments

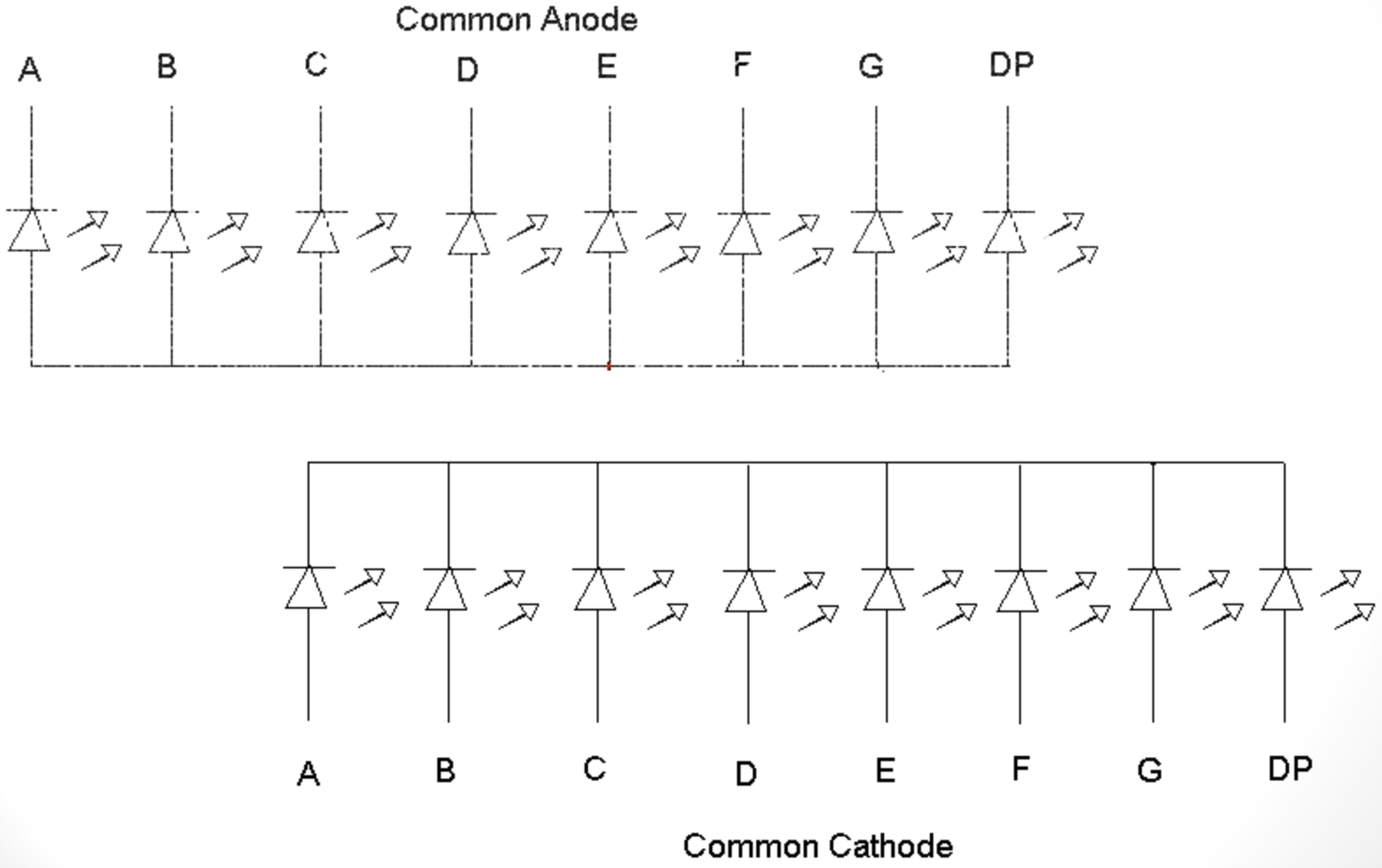
Digit	gfedcba	abcdefg	a	b	c	d	e	f	g
0	0x3F	0x7E	on	on	on	on	on	on	off
1	0x06	0x30	off	on	on	off	off	off	off
2	0x5B	0x6D	on	on	off	on	on	off	on
3	0x4F	0x79	on	on	on	on	off	off	on
4	0x66	0x33	off	on	on	off	off	on	on
5	0x6D	0x5B	on	off	on	on	off	on	on
6	0x7D	0x5F	on	off	on	on	on	on	on
7	0x07	0x70	on	on	on	off	off	off	off
8	0x7F	0x7F	on	on	on	on	on	on	on
9	0x6F	0x7B	on	on	on	on	off	on	on
A	0x77	0x77	on	on	on	off	on	on	on
b	0x7C	0x1F	off	off	on	on	on	on	on
C	0x39	0x4E	on	off	off	on	on	on	off
d	0x5E	0x3D	off	on	on	on	on	off	on
E	0x79	0x4F	on	off	off	on	on	on	on
F	0x71	0x47	on	off	off	off	on	on	on







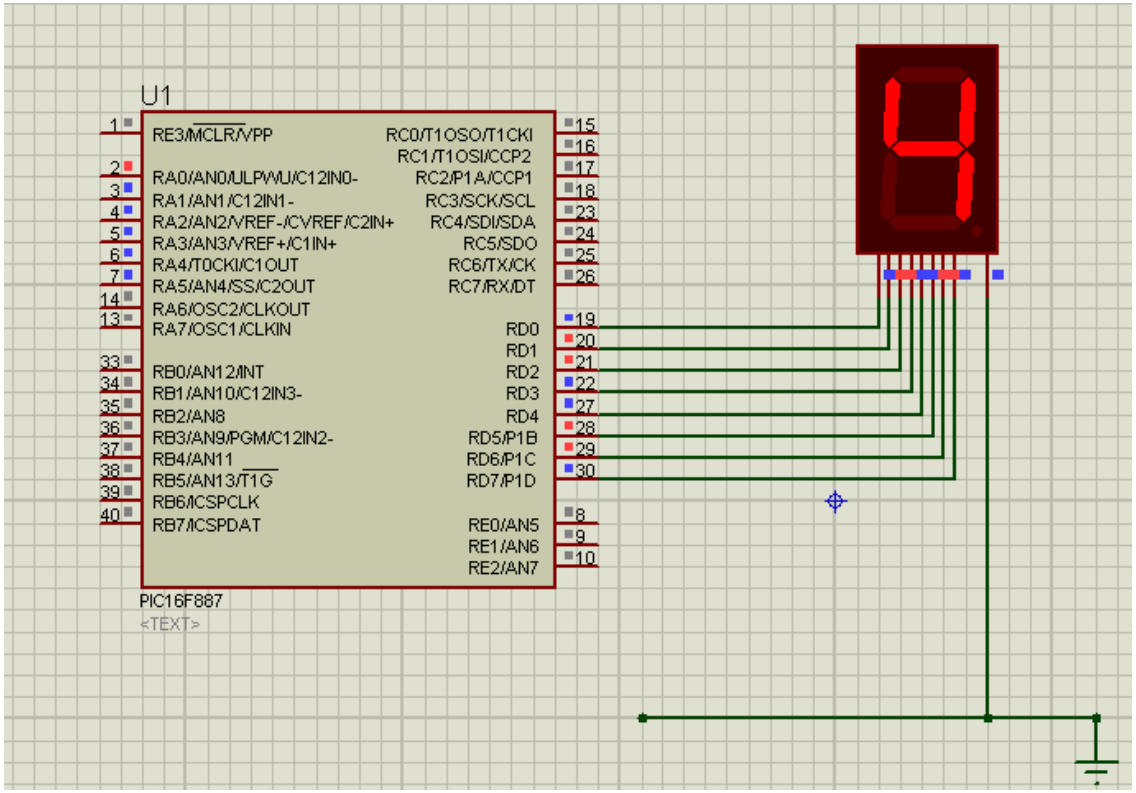
Main types



Project3

- Requirement analysis
  - Increment 7 segment display every 1 sec.
- Design
  - Need a micro, crystal, resistors, 7-segment
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj3_SS1



Pseudo code

- Start
- Configure port D direction as output
- Initialize port value
- increment the 7-seg
- Delay to watch
- Loop infinitely

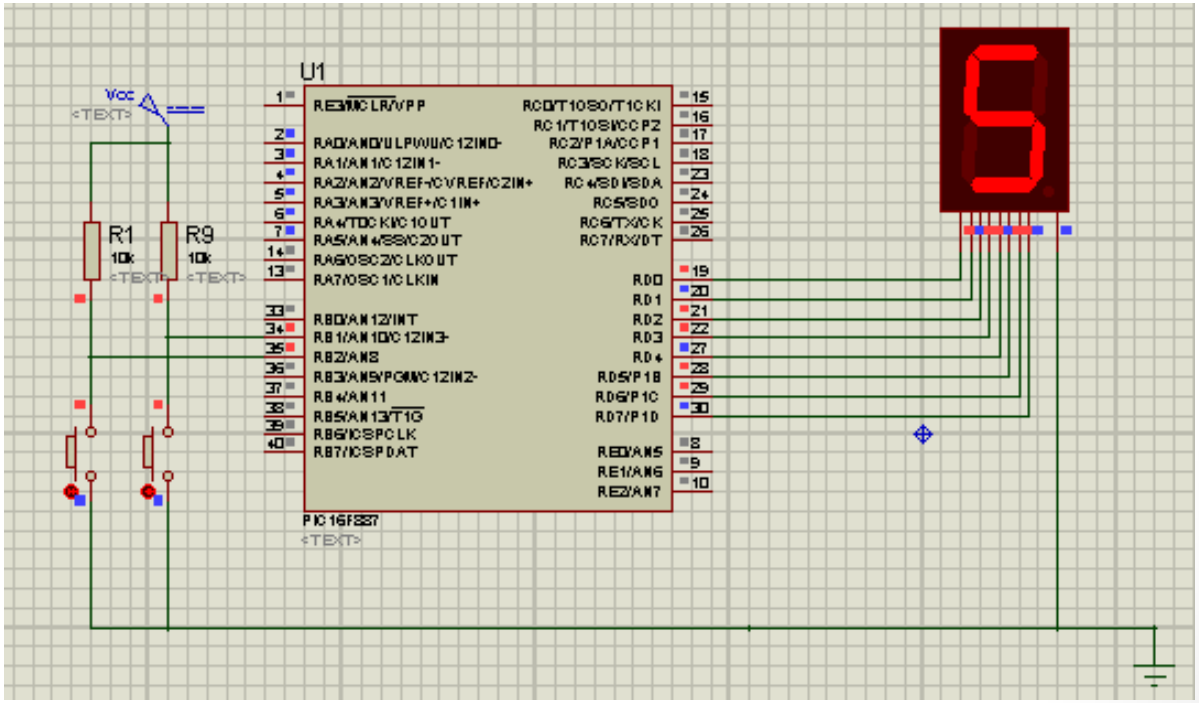
code

proj3_SS1.c

```
20
21 unsigned short i;
22 //-------------- Returns mask for common cathode 7-seg. display
23 unsigned short mask(unsigned short num) {
24   switch (num) {
25     case 0 : return 0x3F;
26     case 1 : return 0x06;
27     case 2 : return 0x5B;
28     case 3 : return 0x4F;
29     case 4 : return 0x66;
30     case 5 : return 0x6D;
31     case 6 : return 0x7D;
32     case 7 : return 0x07;
33     case 8 : return 0x7F;
34     case 9 : return 0x6F;
35   } //case end
36 }//~
37 void main() {
38   INTCON = 0;                    // Disable GIE, PEIE, INTE, RBIE, TOIE
39   PORTA  = 0;
40   TRISA  = 0;
41   PORTD  = 0;
42   TRISD  = 0;
43   do {
44     for (i = 0; i <= 9u; i++) {
45       PORTD = mask(i);           // bring appropriate value to PORTD
46       Delay_ms(1000);
47     }
48   } while (1);                   //endless loop
49 }
50
```

Proj3_SS2

- Increment 7 segment display when push button on RB1 is pressed and decrement it when push button on RB2 is pressed .



Proj3_SS3

- Display on two 7-segments on the same port (Scanning concept)

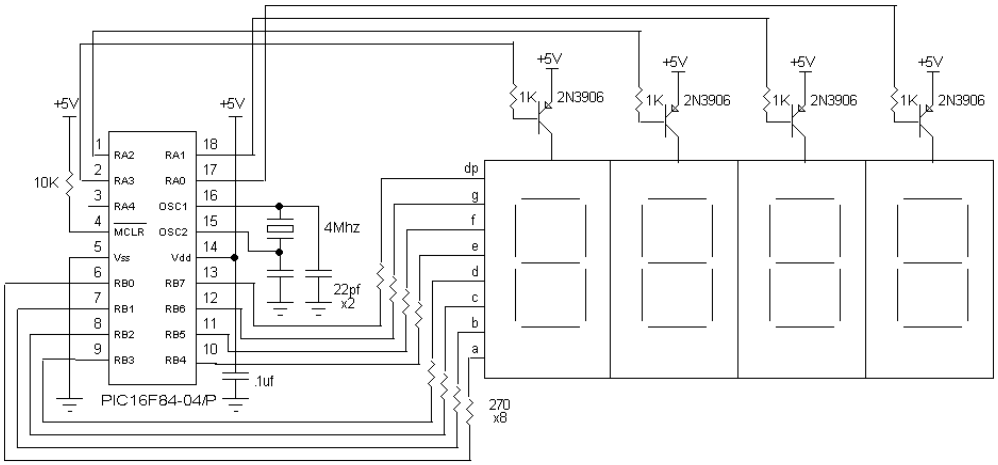
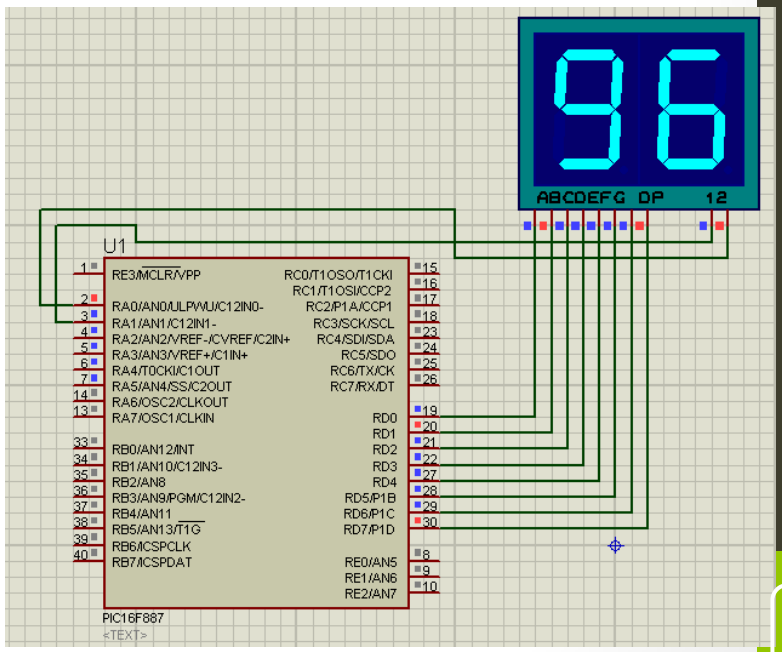
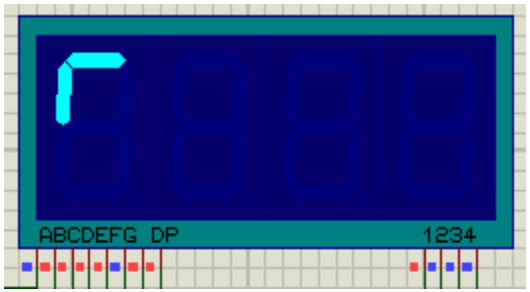


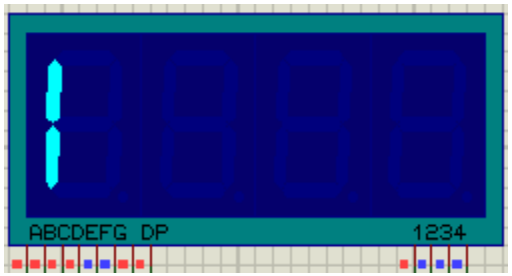
Figure 6

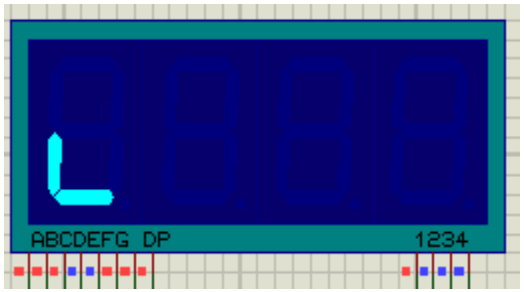


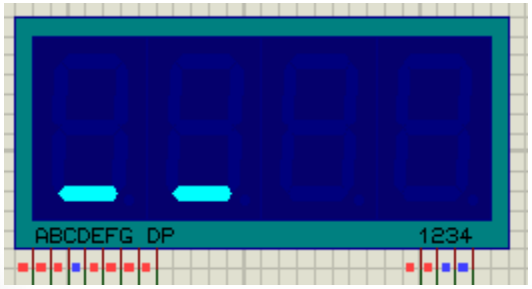
Proj3_SS4

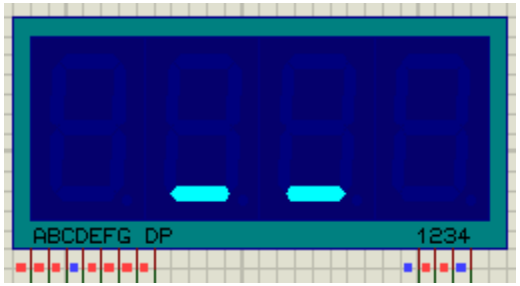
- Make a snake movement on more than one 7-segments
- Go go go ☺

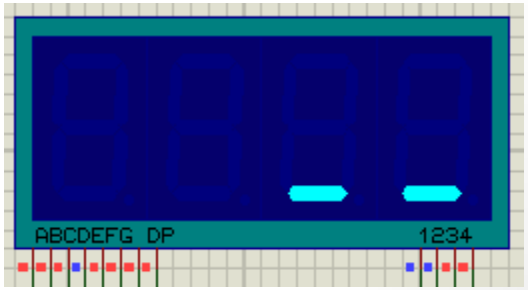






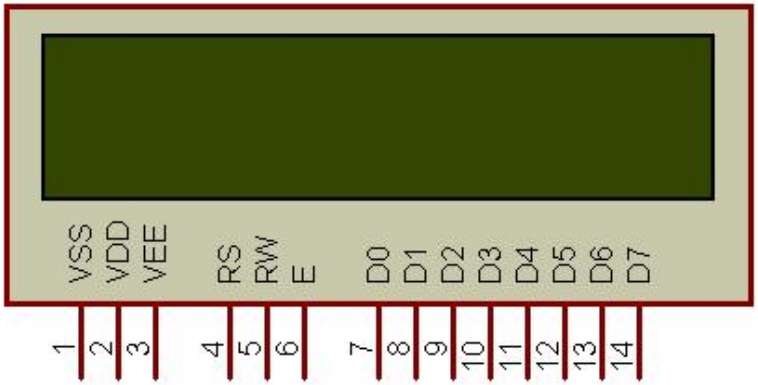






LCD

Liquid Crystal Display LCD



Character LCD type HD44780 Pin diagram

Pin No.	Name	Description
Pin no. 1	**VSS**	Power supply (GND)
Pin no. 2	**VCC**	Power supply (+5V)
Pin no. 3	**VEE**	Contrast adjust
Pin no. 4	**RS**	0 = Instruction input 1 = Data input
Pin no. 5	**R/W**	0 = Write to LCD module 1 = Read from LCD module
Pin no. 6	**EN**	Enable signal
Pin no. 7	**D0**	Data bus line 0 (LSB)
Pin no. 8	**D1**	Data bus line 1
Pin no. 9	**D2**	Data bus line 2
Pin no. 10	**D3**	Data bus line 3
Pin no. 11	**D4**	Data bus line 4
Pin no. 12	**D5**	Data bus line 5
Pin no. 13	**D6**	Data bus line 6
Pin no. 14	**D7**	Data bus line 7 (MSB)

Character LCD pins with 1 Controller

LCD operation

(a) Commands

Instruction	Code	Description
Clear display	0000 0001	Clear display and reset address
Home cursor	0000 001x	Reset display location address
Entry mode	0000 01MS	Set cursor move and display shift
Display control	0000 1DCB	Display & cursor enable
Shift control	0001 PRxx	Moves cursor and shifts display
Function control	001L NFxx	Data mode, line number, font
CGRAM address	01gg gggg	Send character generator RAM address
DDRAM address	1ddd dddd	Send display data RAM address

X	Don't care
M	Cursor move direction 1 = right 0 = left
S	Enable whole display shift = 1
D	Whole display on = 1
C	Cursor on = 1
B	Blinking cursor on = 1
P	Display shift = 1, cursor move = 0
R	Shift right = 1, shift left = 0
L	8-Bits = 1, 4-bits = 0
N	2 Lines = 1, 1 line = 0
F	5 × 10 character = 1, 5 × 8 = 0
g	Character generator RAM address bit
d	Data RAM address bit

(b) Character addresses (16 × 2 display)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

LCD operation

Hex	Binary	Type	Meaning
32	0011 0010	Function control	8-bit data, 1 line, 5×8 character
28	0010 1000	Function control	4-bit data, 2 lines, 5×8 character
0C	0000 1100	Display control	Enable display, cursor off, blink off
06	0000 0110	Entry mode	Cursor auto-increment right, shift off
01	0000 0001	Clear display	Clear all characters
80	1000 0000	DDRAM address	Reset display memory address to 00

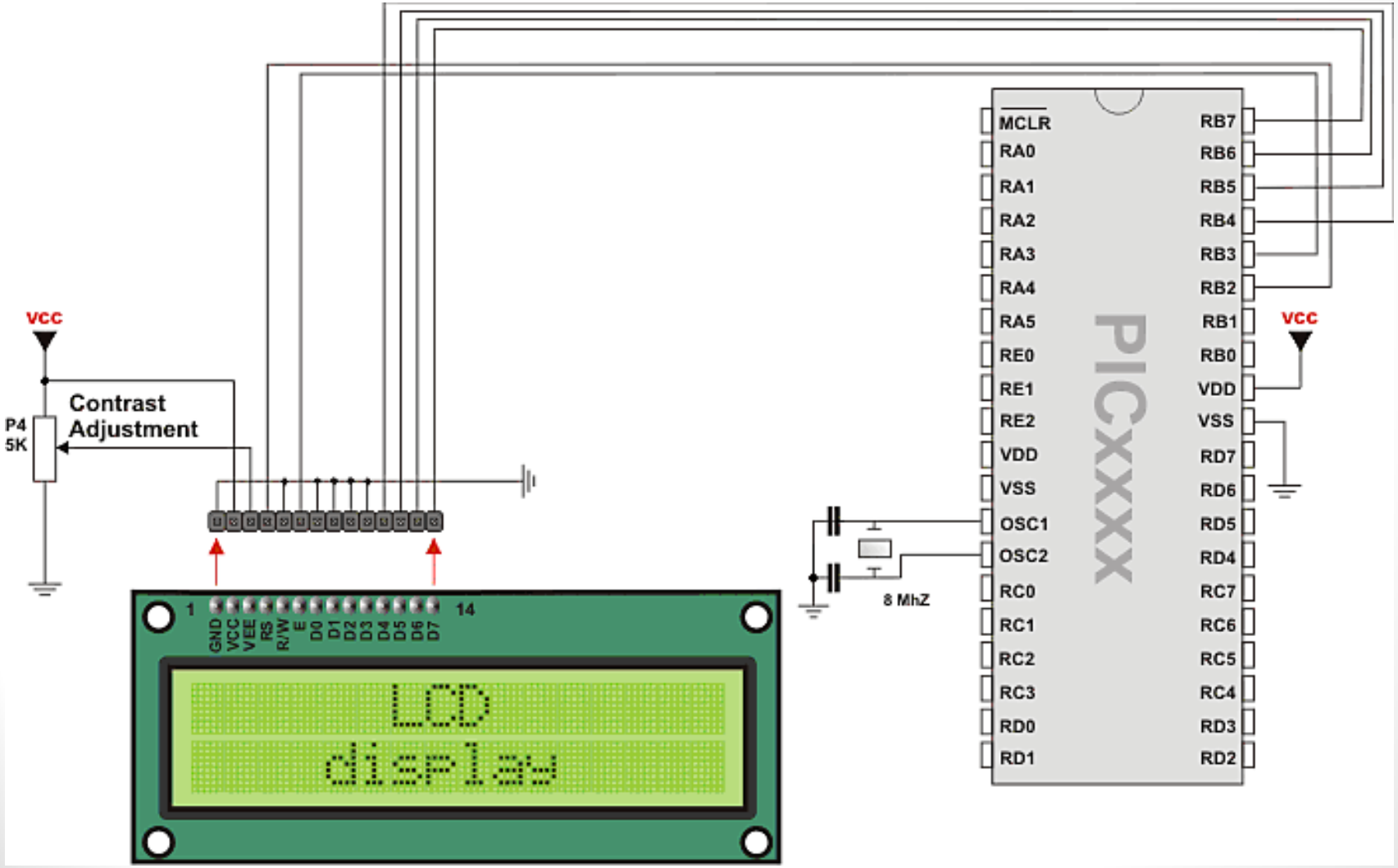
LCD initialisation command code sequence

Project4

- Requirement analysis
  - Display a welcome message on the LCD.
- Design
  - Need a micro, crystal, resistors and a LCD
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj4_lcd1

Note: 4-bit mode connection



Pseudo code

- Start
- Configure port B direction as output
- Initialize port value
- Initialize the lcd
- Write on the screen
- Delay to watch
- Loop infinitely

code

proj4_lcd1

```
5       (c) MikroElektronika, 2005-2008
6   * Description:
7       This is a simple demonstration of LCD library functions. LCD is first
8       initialized, then some text is written at the first row.
9   * Test configuration:
10      MCU:              PIC16F887
11      Dev.Board:        EasyPIC5
12      Oscillator:       HS, 08.0000 MHz
13      Ext. Modules:     LCD 2x16
14      SW:               mikroC v8.0
15  * NOTES:
16      None.
17  */
18
19  char *text = "Welcome";
20
21  void main() {
22    ANSEL  = 0;                           // Configure AN pins as digital I/O
23    ANSELH = 0;
24    Lcd_Config(&PORTB, 4, 5, 6, 3, 2, 1, 0); // Lcd_Init_EP5, see Autocomplete
25
26    LCD_Cmd(LCD_CLEAR);         // Clear display
27    LCD_Cmd(LCD_CURSOR_OFF);    // Turn cursor off
28    LCD_Out(1,6, text);         // Print text to LCD, 1st row, 1st column
29    Delay_ms(1000);
30    LCD_Out(2,3,"LCD worlde");     // Print text to LCD, 2nd row, 6th column
31  }
32
33
```

Proj4_lcd2

- Display a marquee statement on the LCD.

Proj4_lcd3

- Change the displayed statement on the LCD upon a forward/backward keys press.

ADC & DAC

Analog-to-Digital Conversion

Terminology:
analog: continuously valued signal, such as temperature or speed, with infinite possible values in between.

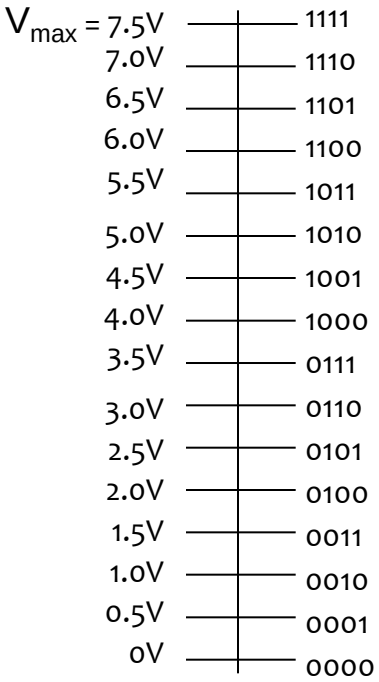
digital: discretely valued signal, such as integers, encoded in binary

analog-to-digital converter: ADC, A/D, A2D; converts an analog signal to a digital signal

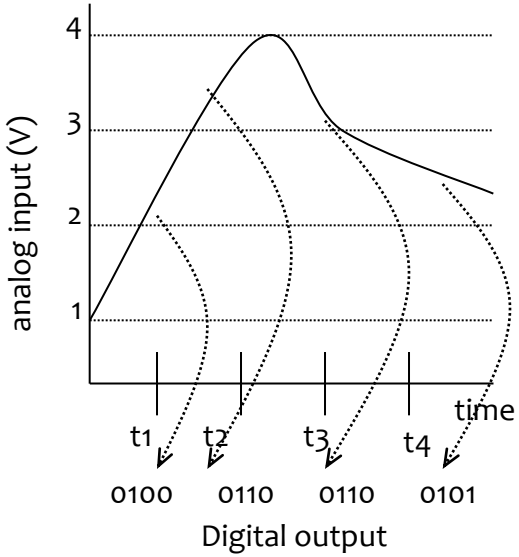
digital-to-analog converter: DAC, D/A, D2A converts a digital signal to a analog signal.

An embedded system's surroundings typically involve many analog signals.

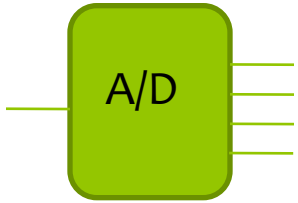
Analog-to-digital converters

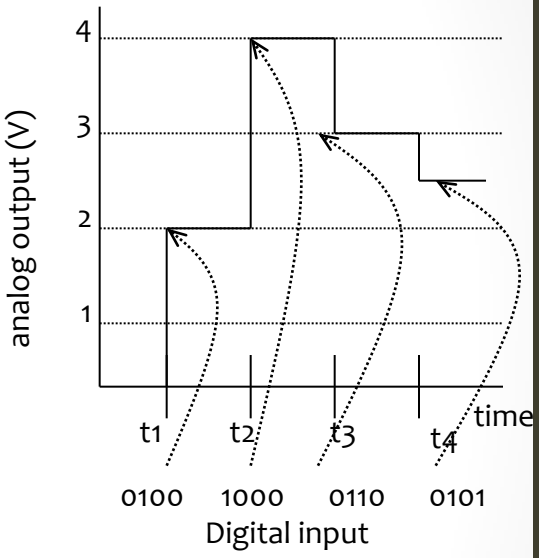


proportionality

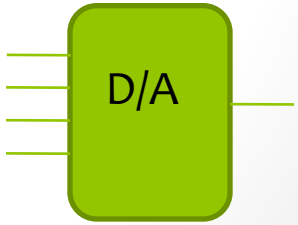


analog to digital





digital to analog



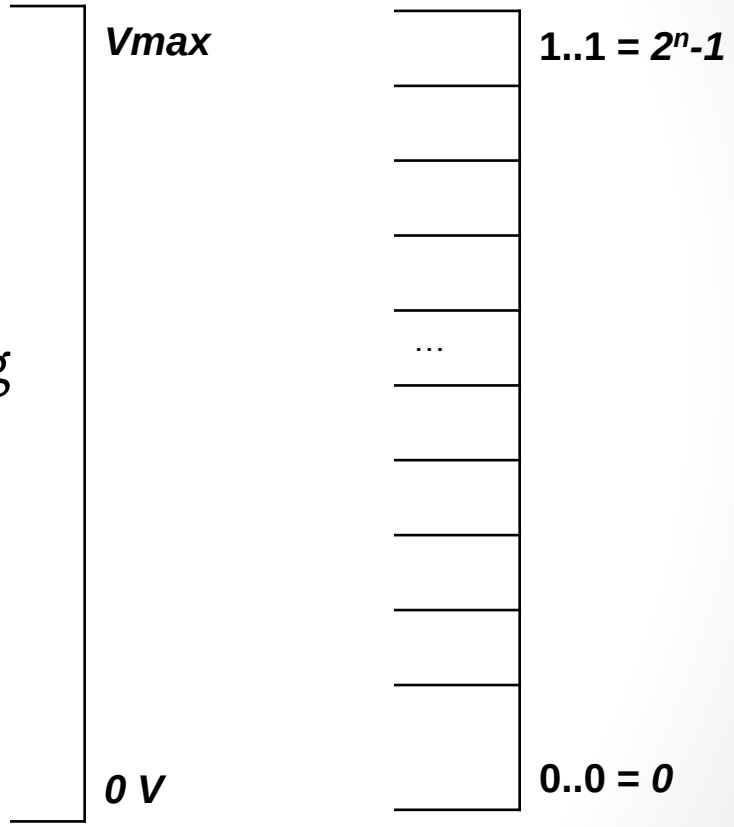
***r*, resolution: smallest analog change resulting from changing one bit**

Proportional Signals

Simple Equation

Assume minimum voltage of 0 V.
Vmax = maximum voltage of the analog signal
n = number of bits for digital encoding
2^n = number of digital codes
x = analog value
d = digital encoding

$$x / Vmax = d / 2^n$$

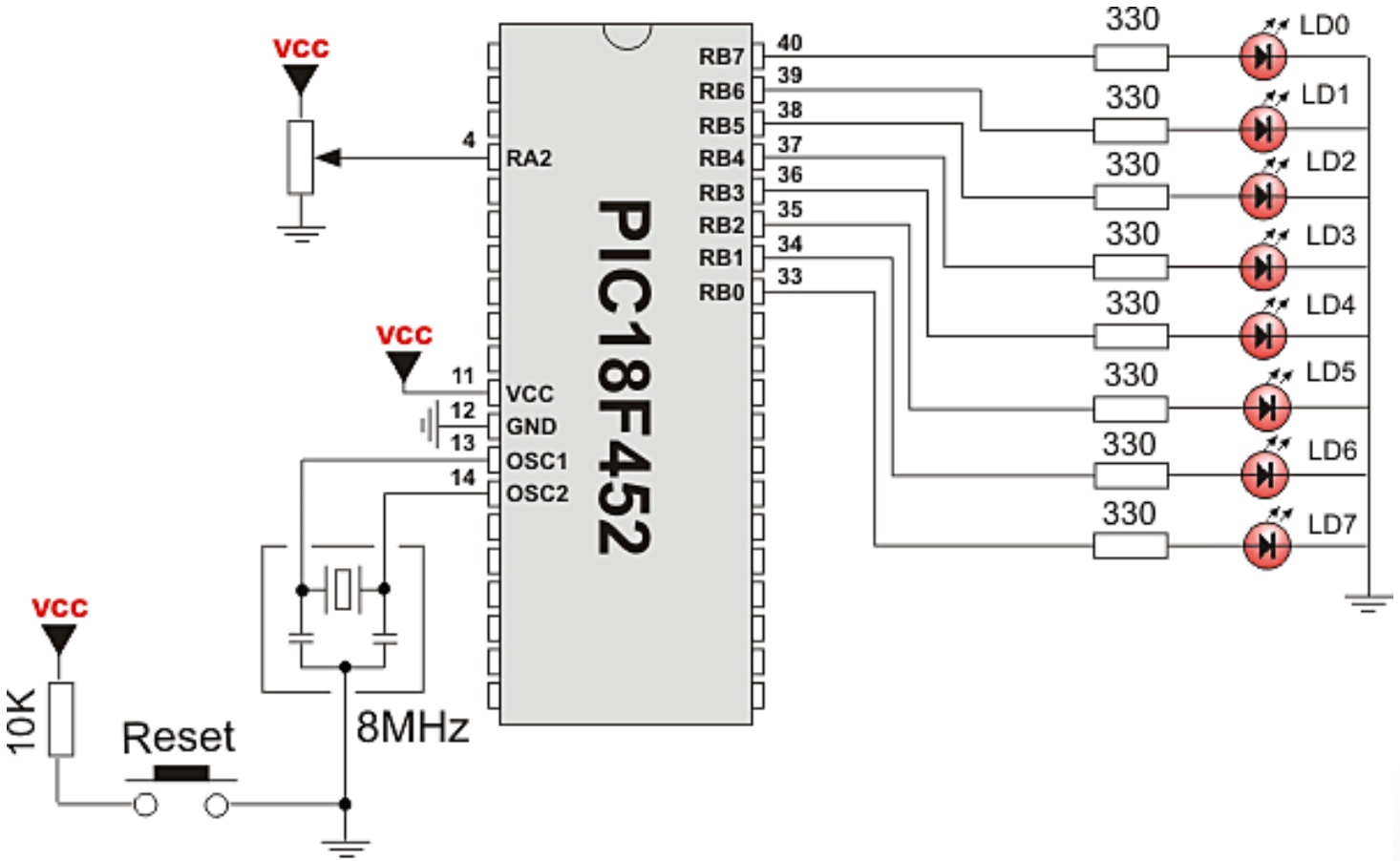


The sampling rate, sample rate, or sampling frequency defines the number of samples per second (or per other unit) taken from a continuous signal to make a discrete signal

Project5

- Requirement analysis
  - Digital Voltmeter (displays value on LEDs).
    - reads analog value from channel 2 and displays it on PORTD (lower 8 bits) and PORTB (2 most significant bits).
- Design
  - Need a micro, crystal, resistors and a LED paragraph
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj5_adc



Note: 2 most significant bits are not connected here

Pseudo code

- Start
- *Configure analog inputs and Vref*
- Configure port A direction as input
- Configure port D direction as output
- Configure *Pins RB7, RB6 as outputs*
- *Get results of AD conversion*
- *Send lower 8 bits to PORTD*
- *Send 2 most significant bits to RB7, RB6*
- Loop infinitely

code

```
unsigned int temp_res;

void main() {
  ADCON1 = 0x80;  // Configure analog inputs and Vref
  TRISA  = 0xFF;  // PORTA is input
  TRISB  = 0x3F;  // Pins RB7, RB6 are outputs
  TRISD  = 0;     // PORTD is output

  do {
    temp_res = Adc_Read(2); // Get results of AD conversion
    PORTD = temp_res;       // Send lower 8 bits to PORTD
    PORTB = temp_res >> 2;  // Send 2 most significant bits to RB7, RB6
  } while(1);
}
```

Proj5_adc_on_ LCD

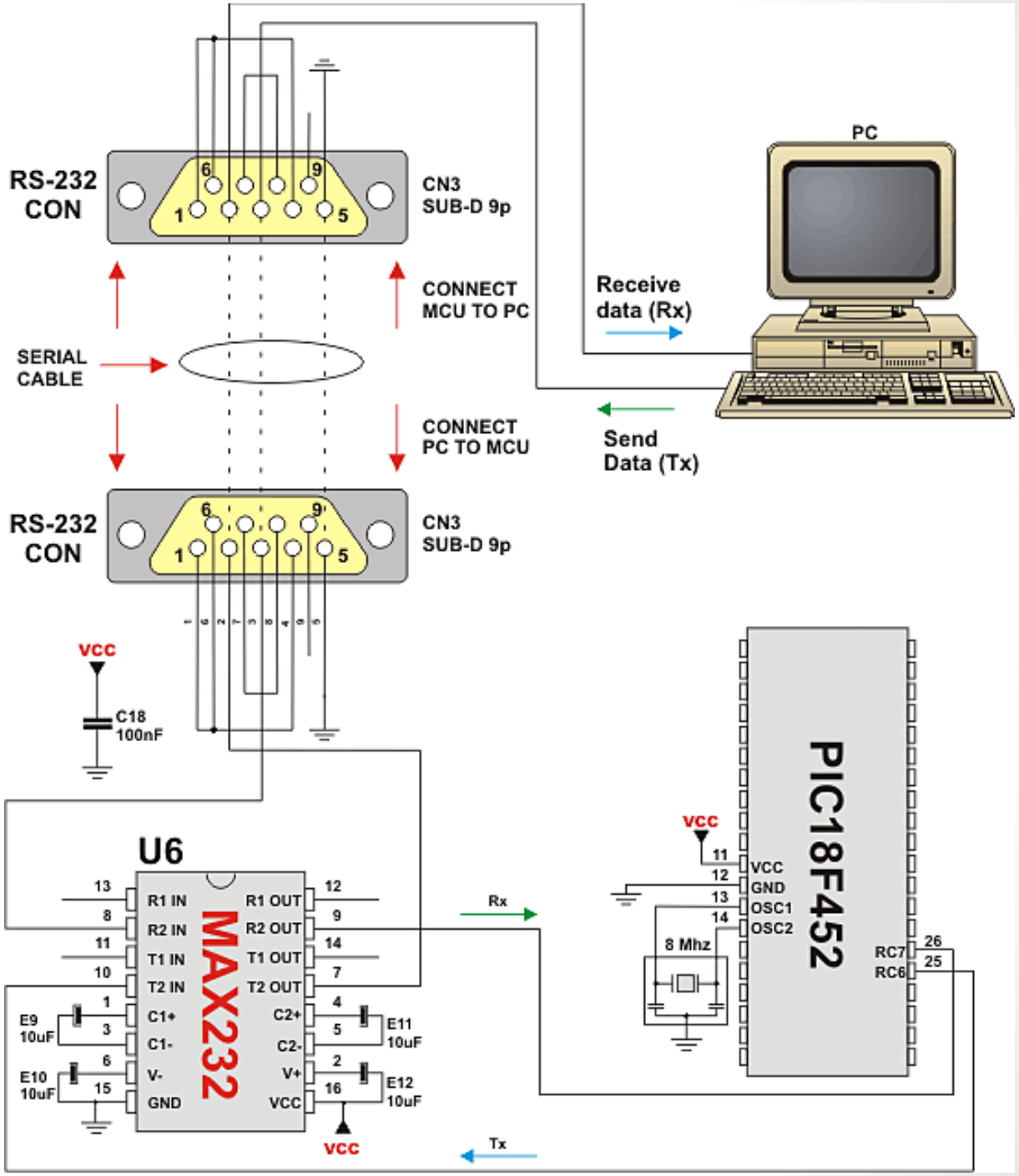
- Digital Voltmeter (displays value on LCD).
  - reads analog value from channel 2 and displays it on LCD.

USART

Project6

- Requirement analysis
  - Receive serial data from PC and send them back.
- Design
  - Need a micro, crystal, resistors and
  - a serial interface (RS232 connector and cable)
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj6_ USART


RS-232 CON
CN3 SUB-D 9p
PC
Receive data (Rx)
CONNECT MCU TO PC
SERIAL CABLE
CONNECT PC TO MCU
Send Data (Tx)
RS-232 CON
CN3 SUB-D 9p
VCC
C18 100nF
U6
MAX232
R1 IN
R2 IN
T1 IN
T2 IN
C1+
C1-
V-
GND
R1 OUT
R2 OUT
T1 OUT
T2 OUT
C2+
C2-
V+
VCC
E9 10uF
E10 10uF
E11 10uF
E12 10uF
Rx
Tx
PIC18F452
VCC
GND
OSC1
OSC2
8 Mhz
RC7
RC6

Pseudo code

- Start
- *Initialize USART module*
- *If data is received*
  - *Read the received data*
  - *Send data via USART*
- Loop infinitely

code

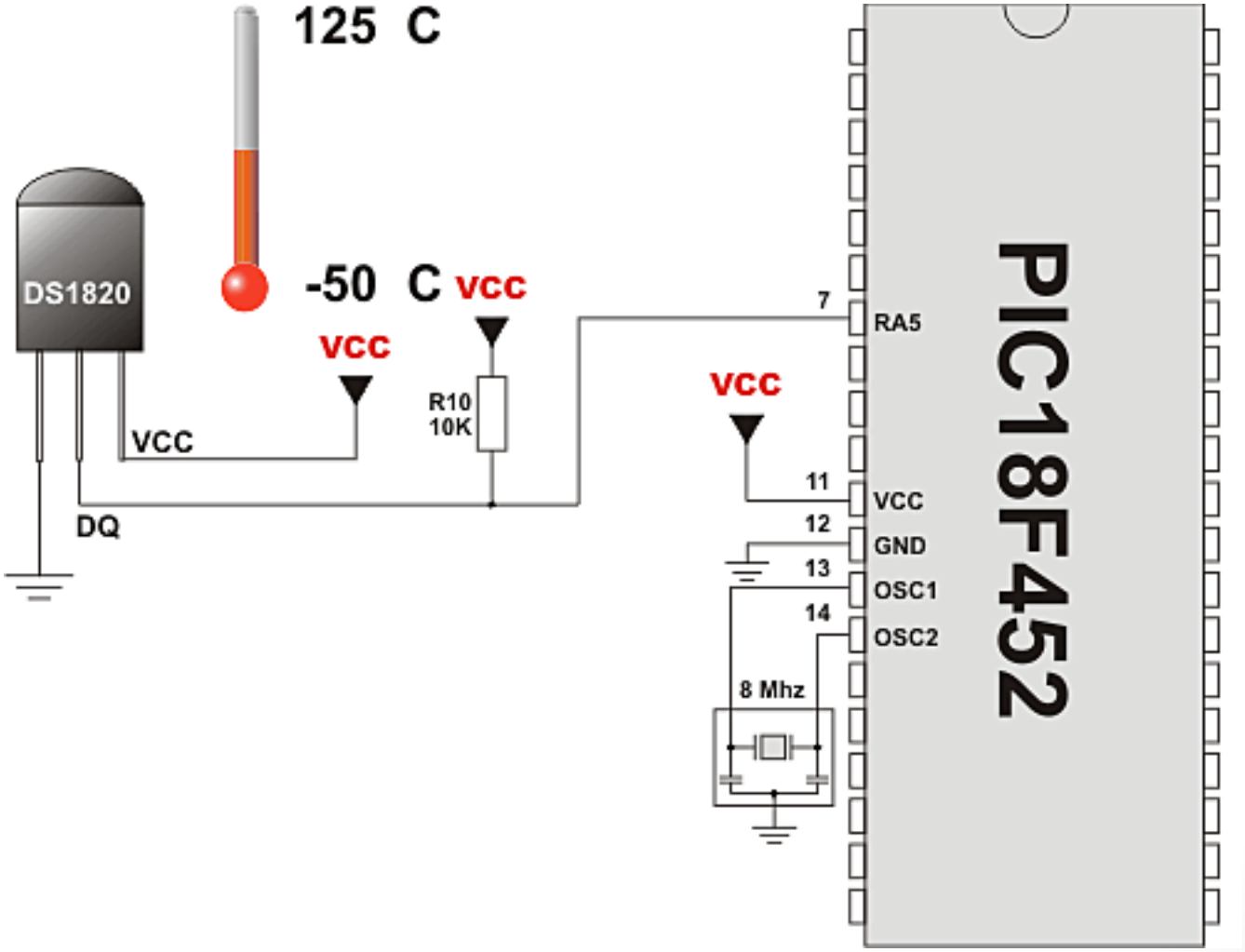
```
unsigned short i;

void main() {

  // Initialize USART module (8 bit, 2400 baud rate, no parity bit..)
  Usart_Init(2400);

  do {
    if (Usart_Data_Ready()) {    // If data is received
      i = Usart_Read();          // Read the received data
      Usart_Write(i);            // Send data via USART
    }
  } while (1);
}//~!
```

SENSORS APPLICATIONS

Project 7

- Requirement analysis
  - Reads the temperature and displays it on LCD.
- Design
  - Need a micro, crystal, resistors and
  - DS1820 temperature sensor & LCD
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj7_TempSnsr



Pseudo code

- Start
- *Set TEMP_RESOLUTION to the corresponding resolution of the DS18x20 sensor.*
- *Configure RA5 pin as digital I/O &*
  - *PORTE as input &*
  - *PORTB as output.*
- *Initialize LCD on PORTB and prepare for output.*
- *Issue one wire signals for communication between uC and the sensor*
- *Get temperature LSB &MSB*
- *Display the result on the LCD*
- Loop infinitely

code

```
// Set TEMP_RESOLUTION to the corresponding resolution of your DS18x20 sensor:
//  18S20: 9
//  18B20: 12 (default setting; can be 9,10,11,or 12)
const unsigned short TEMP_RESOLUTION = 12;

const int RES_FACTOR_1[4] = {5000, 2500, 1250, 625};
const unsigned int RES_FACTOR_2[4] = {0x0001, 0x0003, 0x0007, 0x000F};
const unsigned int RES_FACTOR_3[4] = {0x8000, 0xC000, 0xE000, 0xF000};

unsigned temp;
unsigned short  j, RES_SHIFT;

void Display_Temperature(unsigned int temp) {
  const unsigned short RES_SHIFT = TEMP_RESOLUTION - 8;
  unsigned int temp_whole, temp_fraction;
  unsigned short i;
  char text[8];

  // Isolate the fraction and make it a 4-digit decimal integer (for display)
  temp_fraction = temp & RES_FACTOR_2[RES_SHIFT - 1];
  temp_fraction = temp_fraction * RES_FACTOR_1[RES_SHIFT - 1];
  //portc = temp_fraction;
  // Handle the whole part of temperature value
  temp_whole = temp;
```

Code..

```
  // Is temperature negative?
  if ((temp_whole & 0x8000) != 0u) i = 1;  // Yes, i = 1
  else i = 0;                              // No,  i = 0
  PORTC = i;
  // Remove the fractional part
  temp_whole >>= RES_SHIFT;

  // Correct the sign if necessary
  if (i) temp_whole |= RES_FACTOR_3[RES_SHIFT - 1];

  //portd = temp_whole;
  IntToStr(temp_whole, text);  // Convert whole part to string
  Lcd_Out(2, 6, text);         // Print whole part on LCD
  Lcd_Chr_Cp('.');             // Print dot to separate fractional part


  IntToStr(temp_fraction, text); // Convert fractional part to string

  // Add leading zeroes (we display 4 digits fractional part)
  if (temp_fraction < 1000u) Lcd_Chr_Cp('0');
  if (temp_fraction < 100u)  Lcd_Chr_Cp('0');
  if (temp_fraction < 10u)   Lcd_Chr_Cp('0');

  Lcd_Out_Cp(text);            // Print fractional part on LCD

  Lcd_Chr_Cp(223);             // Print degree character
  Lcd_Chr_Cp('C');             // Print 'C' for Centigrades
}//~
```

Code…

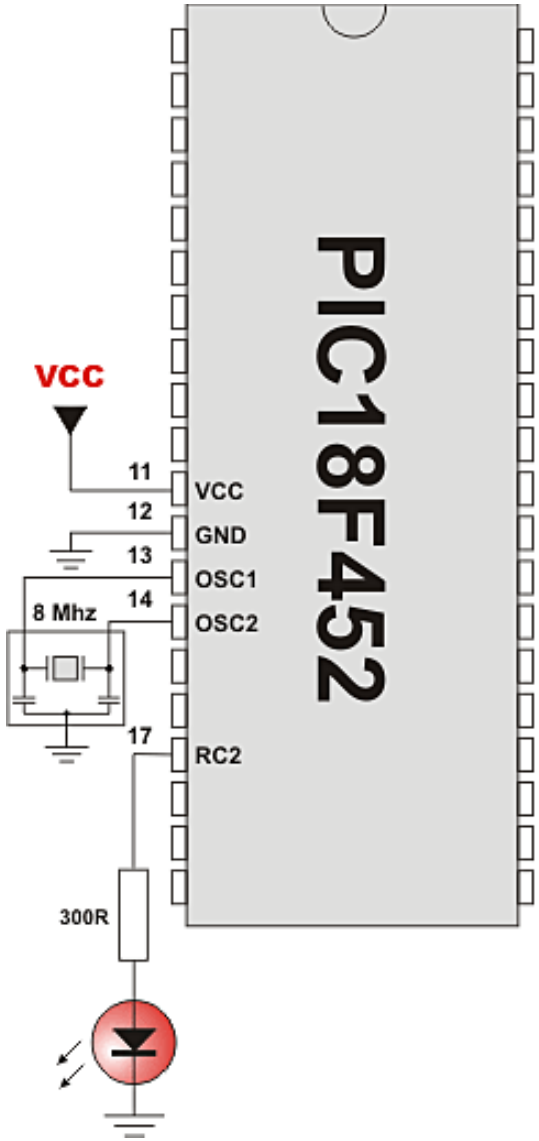
```
void main() {
  ADCON1 = 0xFF;               // Configure RA5 pin as digital I/O
  PORTE  = 0xFF;
  TRISE  = 0x0F;               // PORTE is input
  PORTB  = 0;
  TRISB  = 0;                  // PORTB is output

  // Initialize LCD on PORTB and prepare for output
  Lcd_Init(&PORTB);
  Lcd_Cmd(Lcd_CURSOR_OFF);
  Lcd_Out(1, 1, " Temperature:   ");

  do { // main loop

    Ow_Reset(&PORTE,2);        // Onewire reset signal
    Ow_Write(&PORTE,2,0xCC);   // Issue command SKIP_ROM
    Ow_Write(&PORTE,2,0x44);   // Issue command CONVERT_T
    Delay_us(120);

    Ow_Reset(&PORTE,2);
    Ow_Write(&PORTE,2,0xCC);   // Issue command SKIP_ROM
    Ow_Write(&PORTE,2,0xBE);   // Issue command READ_SCRATCHPAD
    Delay_ms(400);

    j = Ow_Read(&PORTE,2);     // Get temperature LSB
    temp = Ow_Read(&PORTE,2);  // Get temperature MSB
    temp <<= 8; temp += j;     // Form the result
    Display_Temperature(temp); // Format and display result on LCD
    Delay_ms(500);

  } while (1);
```

PWM

Project 8

- Requirement analysis
  - Changes PWM duty ratio on pin RC2 continually and observe them on a LED.
- Design
  - Need a micro, crystal, resistors and LED.
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Proj8_pwm



Pseudo code

- Start
- Initialize PORTB & PORTC as output.
- Initialize PORTA as input.
- *Initialize PWM module .*
- If *button on RA0 pressed, increment*
- If *button on RA1 pressed, decrement*
- *Update the duty cycle.*
- *Display it.*
- Loop infinitely

code

```
// microcontroller : P16F877A
// PWM module is set on RC2.

unsigned short j, oj;

void InitMain() {
  PORTB = 0;              // Set PORTB to 0
  TRISB = 0;              // PORTB is output

  ADCON1 = 6;             // All ADC pins to digital I/O
  PORTA = 255;
  TRISA = 255;            // PORTA is input

  PORTC = 0xFF;           // Set PORTC to $FF
  TRISC = 0;              // PORTC is output
  Pwm_Init(5000);         // Initialize PWM module
}//~

void main() {
  InitMain();
  j   = 80;               // Initial value for j
  oj  = 0;                // oj will keep the 'old j' value
  Pwm_Start();            // Start PWM
```

```
void main() {
  InitMain();
  j   = 80;               // Initial value for j
  oj  = 0;                // oj will keep the 'old j' value
  Pwm_Start();            // Start PWM

  while (1) {             // Endless loop
    if (Button(&PORTA, 0,1,1))   // button on RA0 pressed
      j++ ;                      //    increment j
    if (Button(&PORTA, 1,1,1))   // button on RA1 pressed
      j-- ;                      //    decrement j

    if (oj != j) {        // If change in duty cycle requested,
      Pwm_Change_Duty(j); //   set new duty ratio,
      oj = j;             //   memorize it,
      PORTB = oj;         //   and display on PORTB
    }
    Delay_ms(200);        // Slow down a bit
  }
}
```

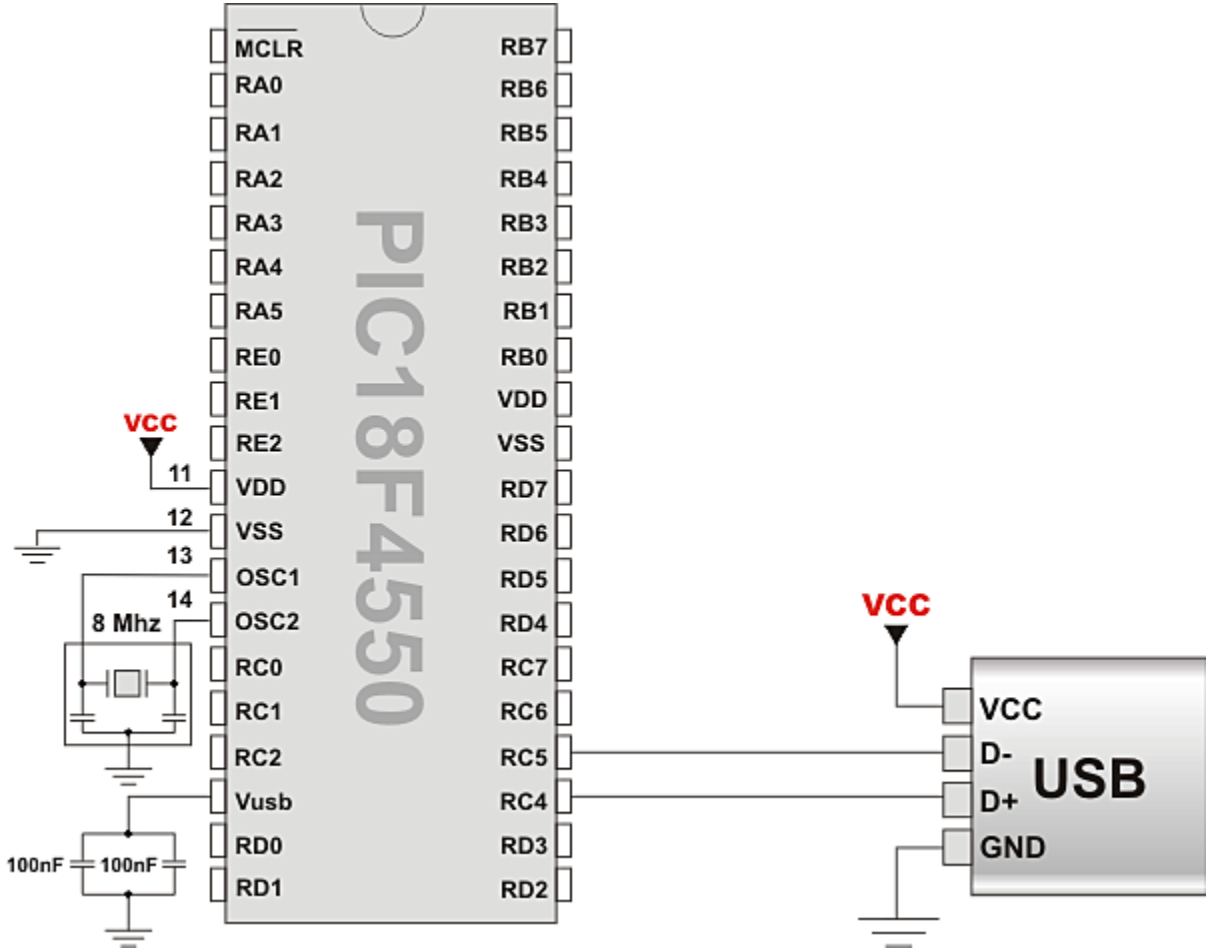
ADVANCED PROJECTS

Other Projects

- Your turn to mange the project yourself.
- Refer to mikroc libraries for codes and other details.

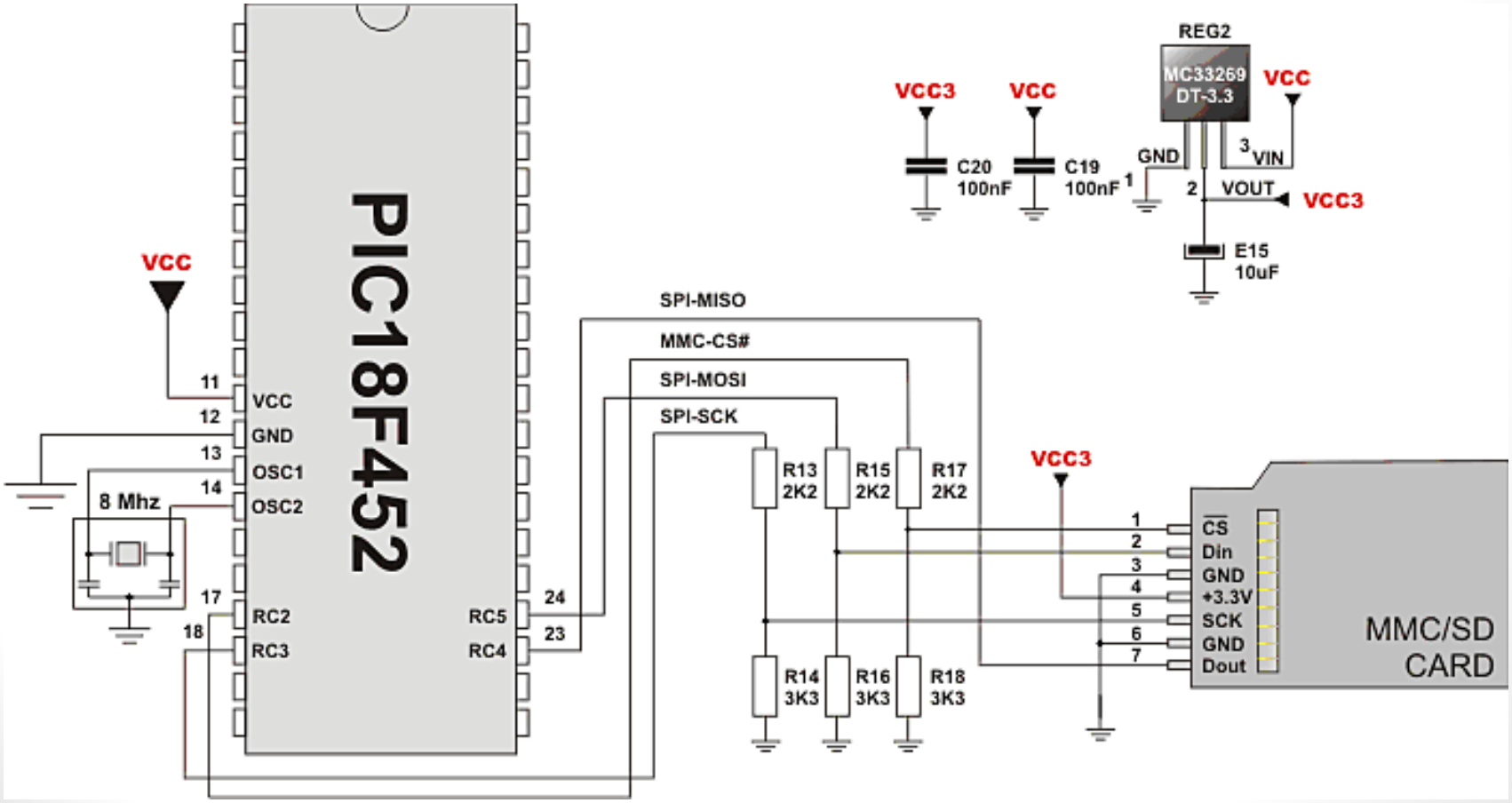
Project 9

- sends sequence of numbers 0..255 to the PC via Universal Serial Bus.



Project 10

- Creation of new file and writing down to it.
- Opening existing file and re-writing it (writing from start-of-file).
- Opening existing file and appending data to it (writing from end-of-file).
- Opening a file and reading data from it (sending it to USART terminal).
- Creating and modifying several files at once.
- Reading file contents.
- Deleting file(s).
- Creating the swap file (see Help for details).



- For more details, refer to:
  - John Catsoulis, **Designing Embedded Hardware**, 2005.
  - Qing Li and Carolyn Yao, **Real-Time Concepts for Embedded Systems**, 2003.
  - Michael Barr, **Programming Microcontrollers in practice in C and C++**, 1999.
- For inquires, send to:
  - ahmad.elbanna@feng.bu.edu.eg