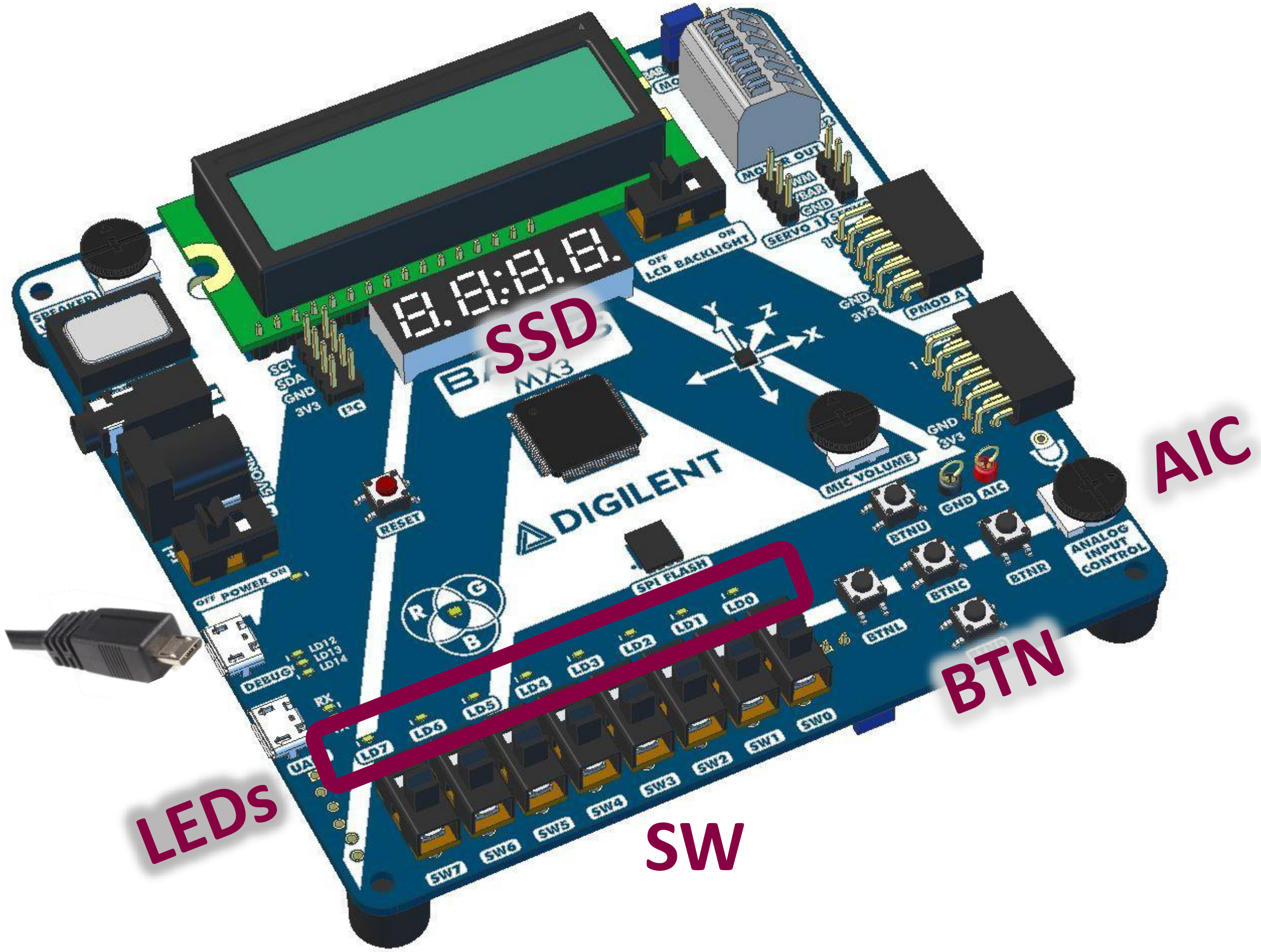
TALLINNA
TEHNIKAÜLIKOOL

Microcontroller „Basys MX3“ and TASKs

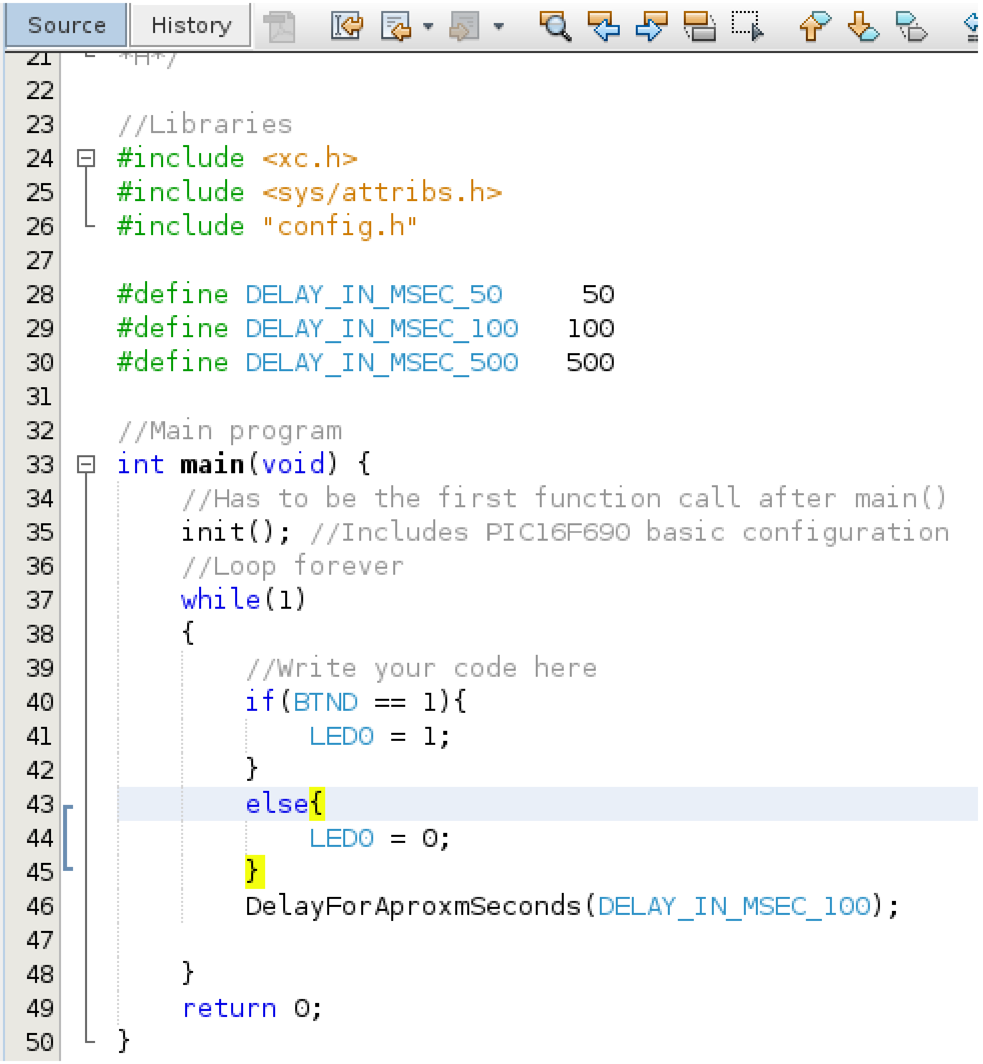
Hardi Selg

04.04.2022

What we will learn to program



Simple test program

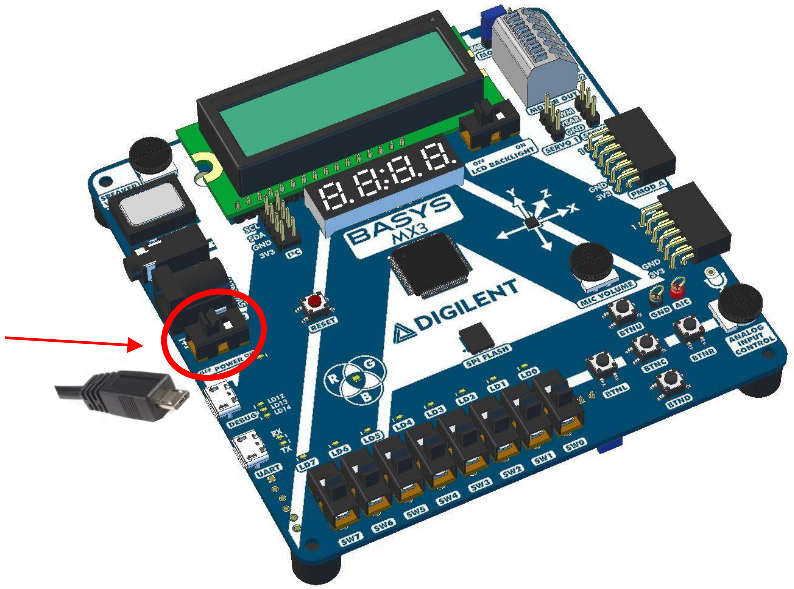


```c

//Libraries
#include <xc.h>
#include <sys/attribs.h>
#include "config.h"

#define DELAY_IN_MSEC_50     50
#define DELAY_IN_MSEC_100   100
#define DELAY_IN_MSEC_500   500

//Main program
int main(void) {
    //Has to be the first function call after main()
    init(); //Includes PIC16F690 basic configuration
    //Loop forever
    while(1)
    {
        //Write your code here
        if(BTND == 1){
            LED0 = 1;
        }
        else{
            LED0 = 0;
        }
        DelayForAproxmSeconds(DELAY_IN_MSEC_100);

    }
    return 0;
}
```

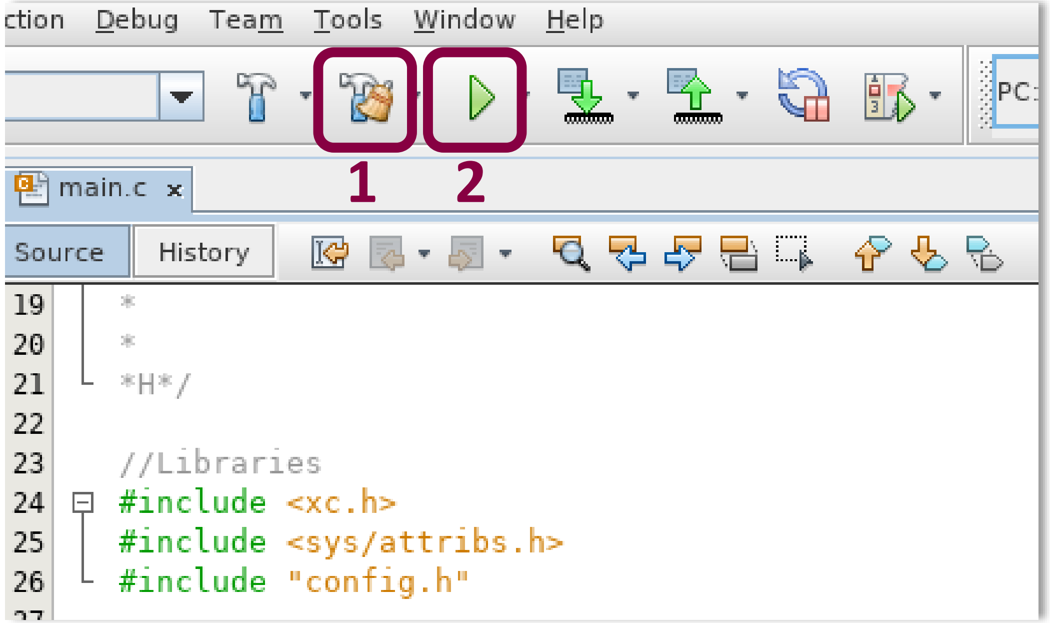
- When you push button „**BTND**“ down, LED “**LD0**” should light up.
- When you release it, it will dim out.

!NB! – The while(1) loop should always have at least **one** delay.

Running your code



Make sure that the power switch is **ON**

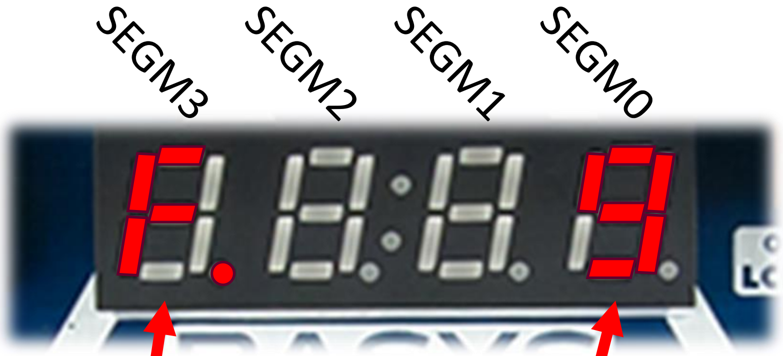


```
 *
 *
 *H*/

//Libraries
#include <xc.h>
#include <sys/attribs.h>
#include "config.h"
```

Running you code consists of 2 stages:

1. ‘Clean and Build’ Project
2. Run Project

Using 7-segment indicators



```
WriteDigits(SEGM3, 15, DOT_ON);
```

```
WriteDigits(SEGM0, 9, DOT_OFF);
```

OR

SEGM3 SEGM0

```
SSD_WriteDigits(15, 0, 0, 9, DOT_ON, 0, 0, 0);
```

To assign values to indicator, call the function **`WriteDigits`** with 3 parameters:

1. Segment name (SEGM3)
2. Numeric value to be displayed (10)
3. Choose whether the „DOT" is **ON** or **OFF**

OR

Use **`SSD_WriteDigits`** to assign values to Segment indicators at the same time

- Numeric values can be in:

  binary: 0b1010

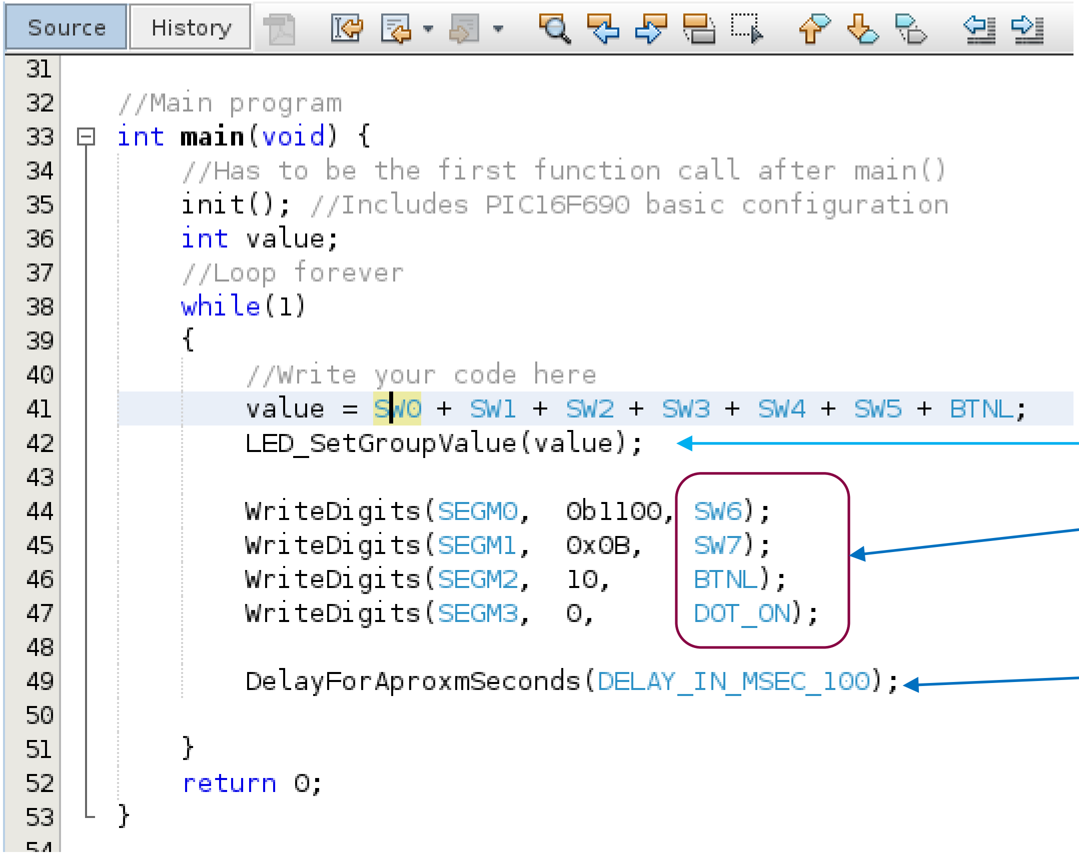
  decimal: 11

  hex: 0x0C

Additional values for indicators

Symbol	Symbol value
NULL (all segments are off)	16
- (minus)	17
FULL (all segments are on)	18
H	19
L	20
P	21
I	22
U	23
N	24

Example program



```
//Main program
int main(void) {
    //Has to be the first function call after main()
    init(); //Includes PIC16F690 basic configuration
    int value;
    //Loop forever
    while(1)
    {
        //Write your code here
        value = SW0 + SW1 + SW2 + SW3 + SW4 + SW5 + BTNL;
        LED_SetGroupValue(value);

        WriteDigits(SEGM0,  0b1100, SW6);
        WriteDigits(SEGM1,  0x0B,   SW7);
        WriteDigits(SEGM2,  10,     BTNL);
        WriteDigits(SEGM3,  0,      DOT_ON);

        DelayForAproxmSeconds(DELAY_IN_MSEC_100);

    }
    return 0;
}
```

Explanation:

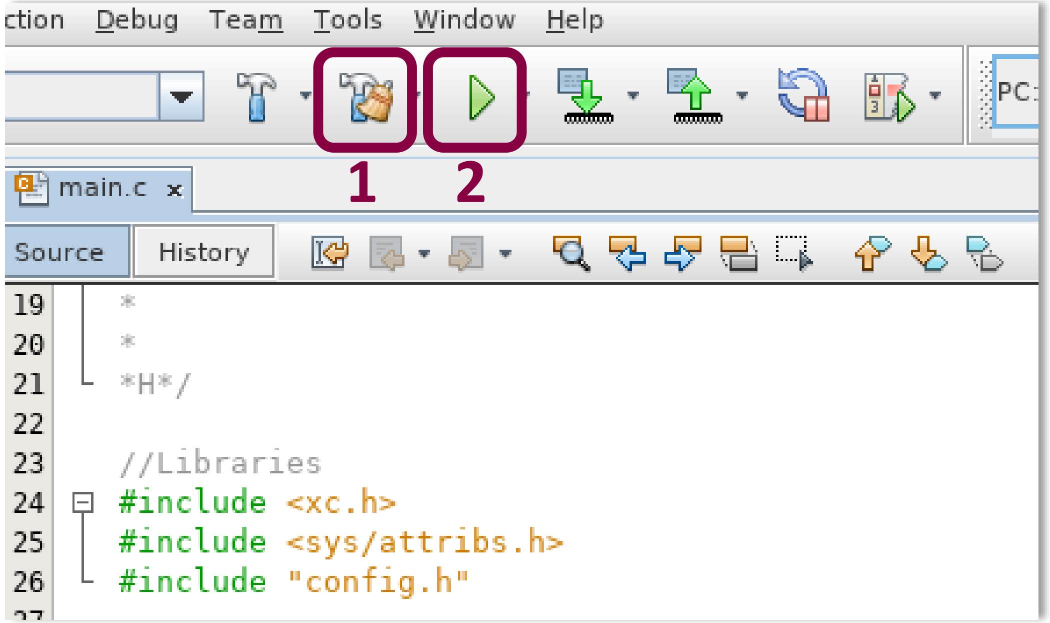
- function **LED_SetGroupValue** displays any given (parameter) numeric value as a binary number on LEDs

  `LED_SetGroupValue(5);`

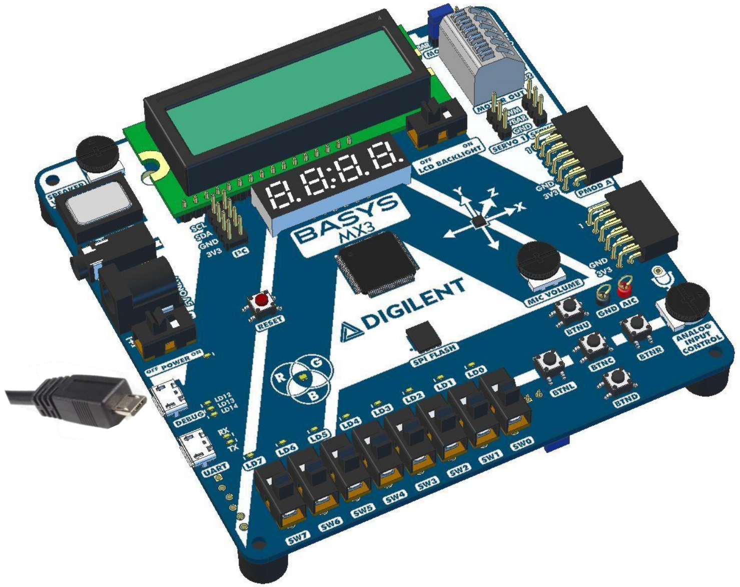
  ○ ○ ○ ○ ○ ● ○ ●

- Indicator '**DOT**'s can also be directly tied to either **switches**, **buttons**, or given a **constant** value
- Each iteration is delayed by 100ms

Running your code



```
 *
 *
 *H*/

//Libraries
#include <xc.h>
#include <sys/attribs.h>
#include "config.h"
```



Running you code consists of 2 stages:

1. ‘Clean and Build’ Project
2. Run Project

ADC value

- Function **ADC_AnalogRead()** which returns unsigned integer value.
- The value is calculated based on the potentiometer (**AIC**) position.
- Possible values from the function **ADC_AnalogRead()** are **0** - **255**

```
Example:

ADC_result = ADC_AnalogRead();
```

RGB LED

- Function **RGBLED_SetValue(R, G, B)** can be used to set values for the RGB LED.
- Parameters **R**, **G**, **B** are type of **unsigned int**, which have size of one byte, values ranging from 0 – 255.

```
Example:

unsigned char red = 64;
unsigned char blue = 255;
…
RGBLED_SetValue(red, 0, blue);
```

Lab task 1 – Controlling LED’s

- Write a program, that assigns a simple logic element to 6 LED’s witch has 2 inputs (switches).
  The same inputs can be used for all of the logic elements
- List of logic elements to be implemented:

AND, NAND, OR, NOR, XOR, XNOR

- Example: **SW0** and **SW1** are assigned to inputs of an 2 input AND

gate. So if both inputs are ON LED **LD0** will light up.

A	B	AND	NAND	OR	NOR	XOR	XNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Lab task 2 – RGB LED (Red, Green, Blue LED)

- Seven segment display must show the Potentiometer **(AIC)** value
- Button **BTNL** press updates RGB color **red** value
- Button **BTNC** press updates RGB color **green** value
- Button **BTNR** press updates RGB color **blue** value
- Previously mentioned value is received from **AIC**

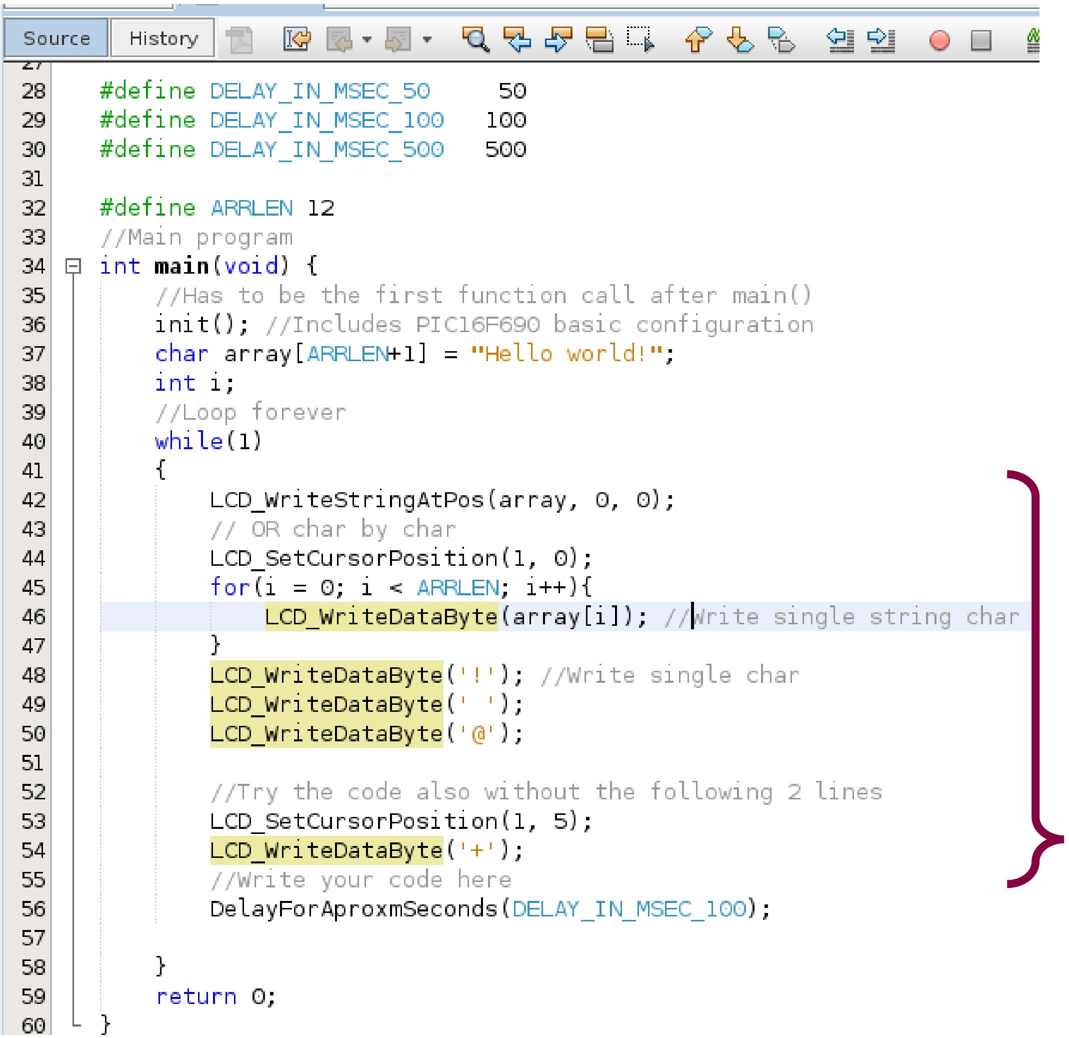
- We have intentionally corrupted **3 values** for the 7-segment display
  - They would appear as turned off
  - **Find the corrupted values and fix them**
  - Corruption is located in `config.c` file, where **digitSegments** is declared

- Example on the video *(final ~20sec)*

Lab task 3 – Egg timer

- Write a program that starts to count down from given binary value until it reaches zero. Current values must be displayed on the 7-segment indicators. Timers starting value must be given using the **SW** switches. When timer reaches ZERO the LED’s must start blinking on and off. Timers starts only when **BUTTON** is pushed.
- Example: If **SW**s have value of “00001111” and the **BUTTON** is pushed, the indicators will show value of “0015” and will start counting down until it reaches zero. Then LED’s will start to blink. LED blinking repetition is not defined.
- The values must decrease once per second
- Values that can be entered by the user must be in the range of 0 to 255 (“00000000” to “11111111”).

LCD panel - ADV



```
#define DELAY_IN_MSEC_50      50
#define DELAY_IN_MSEC_100    100
#define DELAY_IN_MSEC_500    500

#define ARRLEN 12
//Main program
int main(void) {
    //Has to be the first function call after main()
    init(); //Includes PIC16F690 basic configuration
    char array[ARRLEN+1] = "Hello world!";
    int i;
    //Loop forever
    while(1)
    {
        LCD_WriteStringAtPos(array, 0, 0);
        // OR char by char
        LCD_SetCursorPosition(1, 0);
        for(i = 0; i < ARRLEN; i++){
            LCD_WriteDataByte(array[i]); //Write single string char
        }
        LCD_WriteDataByte('!'); //Write single char
        LCD_WriteDataByte(' ');
        LCD_WriteDataByte('@');

        //Try the code also without the following 2 lines
        LCD_SetCursorPosition(1, 5);
        LCD_WriteDataByte('+');
        //Write your code here
        DelayForAproxmSeconds(DELAY_IN_MSEC_100);

    }
    return 0;
}
```

It is possible to write onto the LCD panel in **2** different ways:

- Writing a string at once:

  `LCD_WriteStringAtPos(array, 0, 0);`

  where the arguments are: **1)** char array, **2)** line to write on, **3)** cursor position on the line

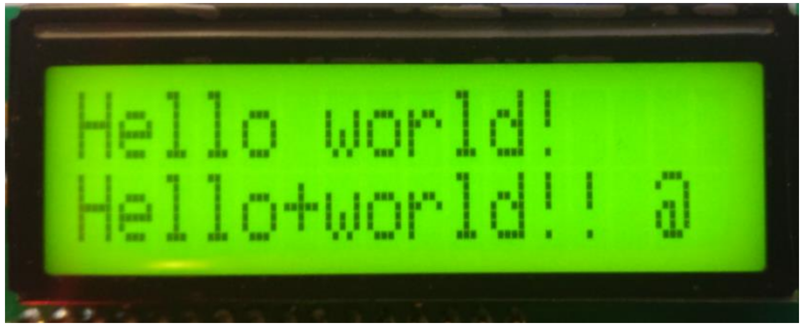
- Writing string char by char:

  Line nr

  Corsor pos

  ```
  LCD_SetCursorPosition(1, 0);
  LCD_WriteDataByte('H');
  LCD_WriteDataByte(array[1]);
  …
  ```

- LCD supports 2 lines of 16 **characters**



BCD – Binary coded decimal - ADV

Regular binary representation:

	128	64	32	16	8	4	2	1
73	0	1	0	0	1	0	0	1
163	1	0	1	0	0	0	1	1
12	0	0	0	0	1	1	0	0

BCD – binary coded decimal

	8	4	2	1	8	4	2	1	8	4	2	1
73	0	0	0	0	0	1	1	1	0	0	1	1
163	0	0	0	1	0	1	1	0	0	0	1	1
12	0	0	0	0	0	0	0	1	0	0	1	0

Max value for a BCD digit is **9**

Binary values for a BCD digit are in range of “0000” to “1001” (0 – 9)

Lab task ADVanced - Add BCD values

- Switches must be divided into groups of 4, where one group is operand 1, the other group is operand 2.
- Switch groups must return values in BCD format (max value 9)
- If invalid input is set on the switches, error message must be shown on the LCD
  - In addition calculation row must be clear
- Calculation must be show on the LCD
  - In addition error row must be clear

TALLINNA
TEHNIKAÜLIKOOL