Ministerul Educatiei al Republicii Moldova
Universitatea Tehnica a Moldovei
Filiera Anglofona

Report

at Embedded Systems

Laboratory Work #2: *Input/Output Registres. Work with LED, LCD and Button.*

Performed by: Ana-Maria Brinza

Verified by: Andrei BRAGARENCO

Chisinau 2016

Topic:

Input/Output Registres. Work with LED, LCD and Button.

Objectives:

- Understand GPIO
- Interfacing LCD
- Connecting LED
- Connect Button

Task:

Write a program. Work with LED, LCD and use a button to simulate the process of event driven programming.

General Information:

GPIO

GPIO stands for General Purpose Input/Output. It’s a standard interface used to connect microcontrollers to other electronic devices. For example, it can be used with sensors, diodes, displays, and System-on-Chip modules. Estimote Location Beacons (Hardware Revision F2.3 and later) are equipped with GPIO. It allows for providing external power supply, remote control of connected devices, broadcasting more contextual data, or defining contents for custom Bluetooth data packets.

GPIO can be used in three modes:

- input
- output
- UART interface

GPIO pins have no predefined purpose, and go unused by default. The idea is that sometimes a system integrator who is building a full system might need a handful of additional digital control lines—and having these available from a chip avoids having to arrange additional circuitry to provide them. For example, the Realtek ALC260 chips (audio codec) have 8 GPIO pins, which go unused by default.

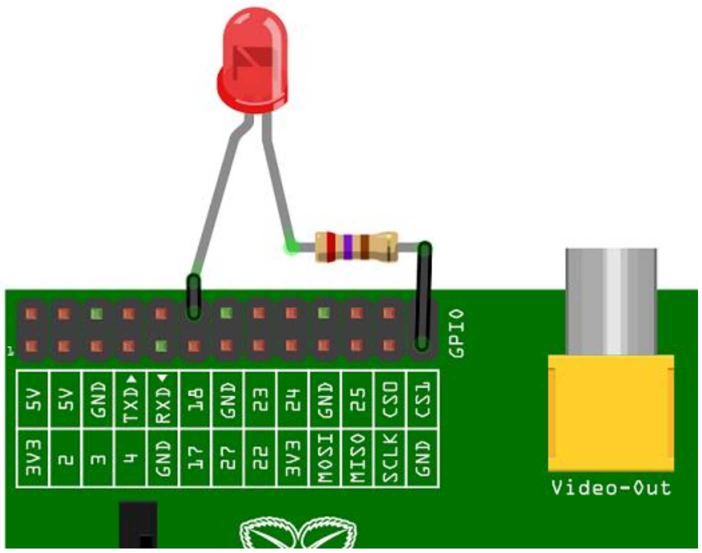


Figure 1: Representation of GPIO pins

LCD

A **liquid-crystal display** (**LCD**) is a flat-panel display or other electronic visual display that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and 7-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements.

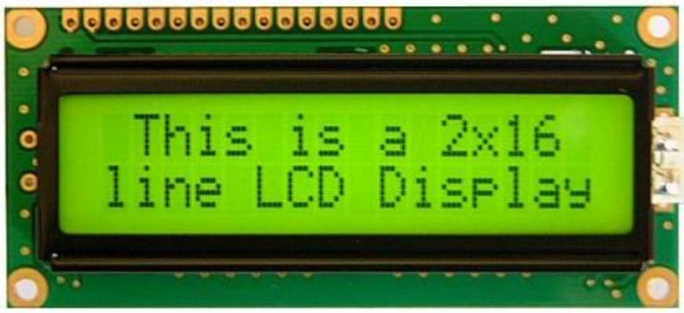


Figure 1.2: Example of LCD Display

LCD operation

In recent years the LCD is finding widespread use replacing LEDs (seven-segment LEDs or other multisegment LEDs). This is due to the following reasons:

I. The declining prices of LCDs.

2. The ability to display numbers, characters, and graphics. This is in contrast to LEDs, which are limited to numbers and a few characters.

3. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU (or in some other way) to keep displaying the data.

4. Ease of programming for characters and graphics.

LCD pin descriptions

The LCD discussed in this section has 14 pins. The function of each pin is given in Table from Figure 2 shows the pin positions for various LCDs.

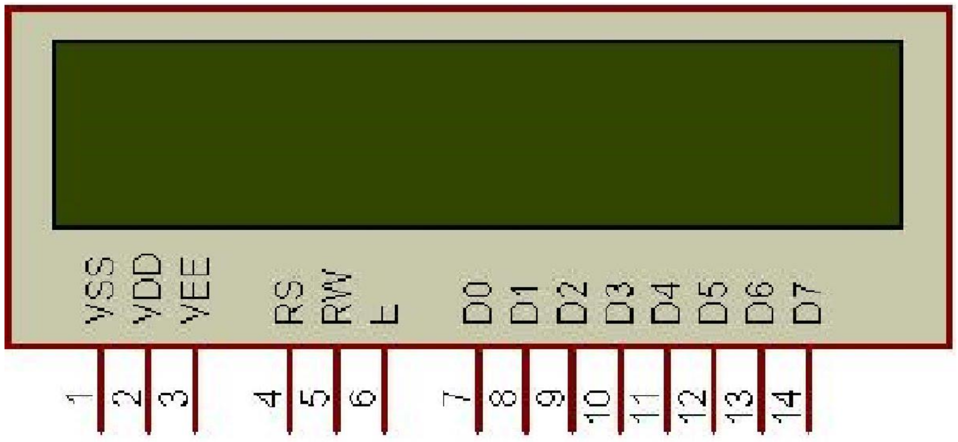


Figure 2.0: Representation of LM016L

Vcc, Vss, and Vee

While V cc and V ss provide +5V and ground, respectively, VEE is used for controlling LCD contrast.

RS, register select

There are two very important registers inside the LCD. The RS pin is used for their selection as follows. If RS = 0, the instruction command code register is selected, allowing the user to send commands such as clear display, cursor at home, and so on. If RS = 1 the data register is selected, allowing the user to send data to be displayed on the LCD.

RW, read/write

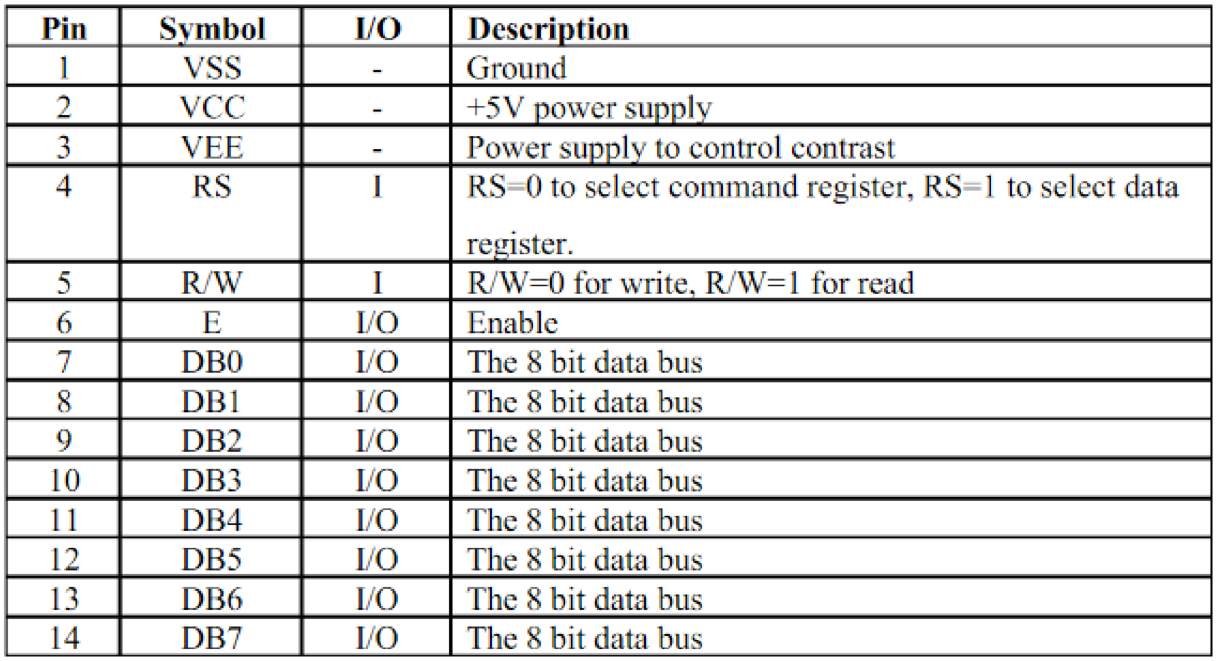
R/W input allows the user to write information to the LCD or read information from it. R/W = 1 when reading; R/W = 0 when writing.

E, enable

The enable pin is used by the LCD to latch information presented to its data pins. When data is supplied to data pins, a high-to-low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450 ns wide.

D0-D7

The 8-bit data pins, DO-D7, are used to send information to the LCD or read the contents of the LCD's internal registers.



Pin	Symbol	I/O	Description
1	VSS	-	Ground
2	VCC	-	+5V power supply
3	VEE	-	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register.
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8 bit data bus
8	DB1	I/O	The 8 bit data bus
9	DB2	I/O	The 8 bit data bus
10	DB3	I/O	The 8 bit data bus
11	DB4	I/O	The 8 bit data bus
12	DB5	I/O	The 8 bit data bus
13	DB6	I/O	The 8 bit data bus
14	DB7	I/O	The 8 bit data bus

Figure 2.1: LCD Pin Description

LCDs are used in a wide range of applications including computer monitors, televisions, instrument panels, aircraft cockpit displays, and indoor and outdoor signage. Small LCD screens are common in portable consumer devices such as digital cameras, watches, calculators, and mobile telephones, including smartphones. LCD screens are also used on consumer electronics products such as DVD players, video game devices and clocks. LCD screens have replaced heavy, bulky cathode ray tube (CRT) displays in nearly all applications. LCD screens are available in a wider range of screen sizes than CRT and plasma displays, with LCD screens available in sizes ranging from tiny digital watches to huge, big-screen television set.

The most commonly used LCDs found in the market today are 1 Line, 2 Line or 4 Line LCDs which have only 1 controller and support at most of 80 charachers, whereas LCDs supporting more than 80 characters make use of 2 HD44780 controllers.

Most LCDs with 1 controller has 14 Pins and LCDs with 2 controller has 16 Pins (two pins are extra in both for back-light LED connections). Pin description is shown in the table below.

In this laboratory work was used the LM016L lcd.

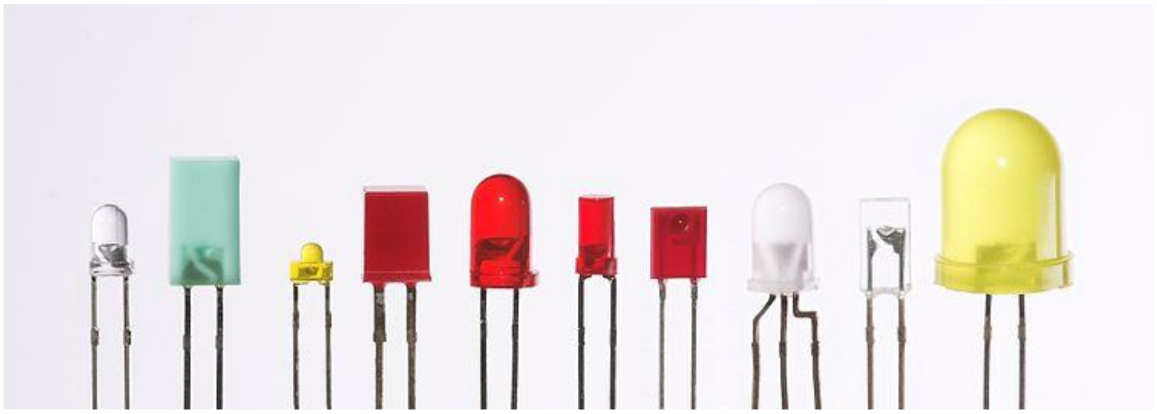
LED

A **light-emitting diode** (**LED**) is a two-lead semiconductor light source. It is a p–n junction diode, which emits light when activated. When a suitable voltage is applied to the leds, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

An LED is often small in area (less than 1 mm^2) and integrated optical components may be used to shape its radiation pattern.

Appearing as practical electronic components in 1962, the earliest LEDs emitted low-intensity infrared light. Infrared LEDs are still frequently used as transmitting elements in remote-control circuits, such as those in remote controls for a wide variety of consumer electronics. The first visible-light LEDs were also of low intensity, and limited to red. Modern LEDs are available across the visible, ultraviolet, and infrared wavelengths, with very high brightness. Early LEDs were often used as indicator lamps for electronic devices, replacing small incandescent bulbs. They were soon packaged into numeric readouts in the form of seven-segment displays, and were commonly seen in digital clocks.

AVR microcontrollers such as the ATMega8515 only supply a current of about 20mA and so we can drive an LED directly from the microcontroller port eliminating the



resistor. In fact if a resistor is added the intensity of the LED will be low.

Figure 3: Representation of different LEDs

Tools and Technologies used:

Atmel Studio 7 and Proteus where used in this laboratory work.

Solution:

In this laboratory work the student familiarized with LCD interfacing, led and button usage in an Embedded System.

My project has more files:
 - main.c : the main function
- led.c and led.h – specific files for the led
- button.c and button.h – files for button

main

Main function is the entry point of the program. First of all it has the initialization: Initializes the led and button:

```
init();
LedInit();
```

Then it enters the infinite while loop, with a delay of 100 ms: `_delay_ms(100);`
In the loop it is checked if the button is pressed, with the function `isPressed()` and depending on the output of that function the Led will turn on or off.

Led

For the led I have 3 functions:

- LedInit() : this function sets the DDRA of where the LED is connected to 1 (output).
- ledOn(): set's the 1st pin of the port A to 1, therefore the LED will torn on because the current can pass through it.
- ledOff(): set's the 1st pin of the port A to 0, therefore the current can not pass through the LED and it will turn Off

Button:

For the button I have the following functions:

- init(): this function sets the DDR (data direction Register) of where the button is connected to 0 (input)
- isPressed(): this function checks if the first PIN of the PORTC is 0 or 1

Proteus Simulation

After implementing the solution in terms of C code the schemes where build in Proteus. There we have the following elements:

- ATmega32
- LCD (LM016L)
- Button
- LED-Red
- Resitor

Here is the scheme when the LED is on:

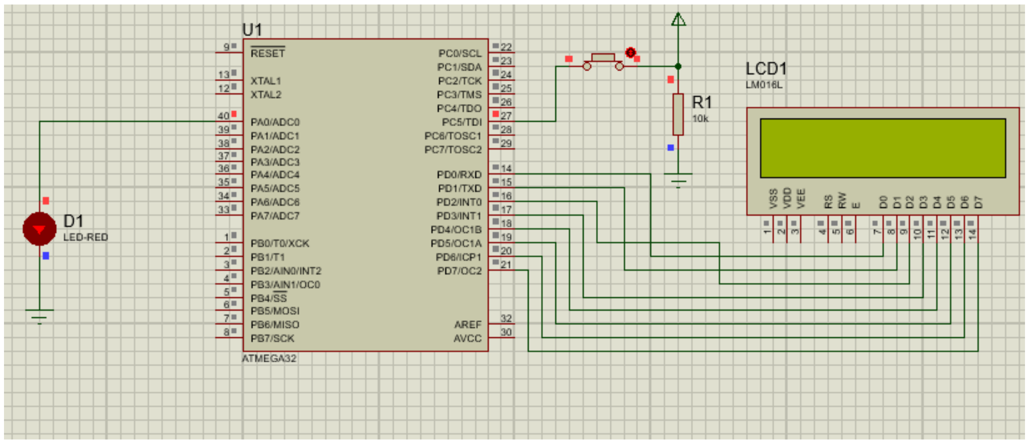


Figure 5: Construction of the scheme - Led, Button and LCD interfacing. Led is on

And another one when the LED is off:

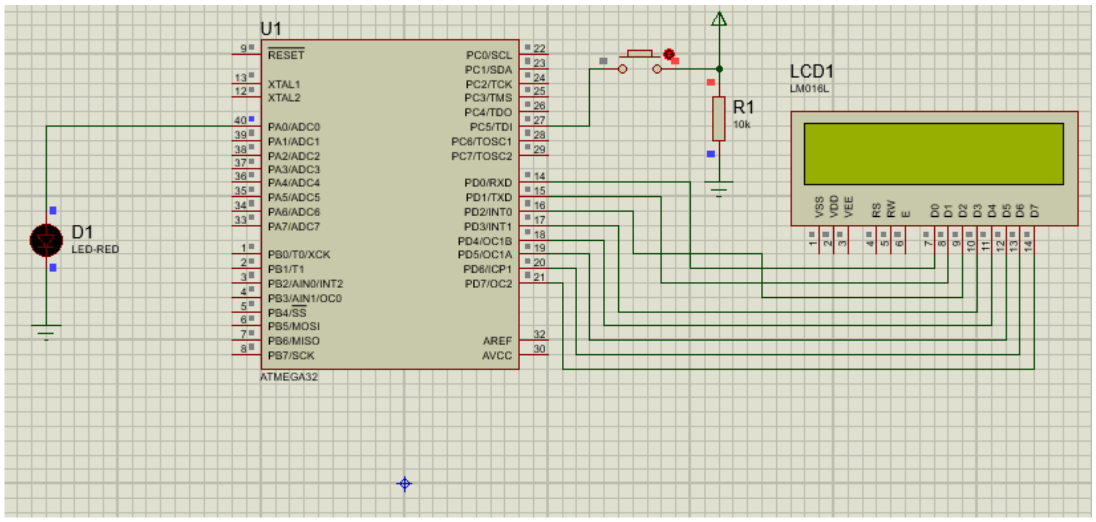


Figure 6: Construction of the scheme - Led, Button and LCD interfacing. Led is off

Conclusion:

This laboratory familiarized student with GPIO and how to work with it. I build a system that consists of: MCU(ATMEGA32), LCD, LED and button. The button allows us to turn on and off the LED (the diode). There is a resistance involved that assures the circuit security.
All these helped me to gain some general knowledge about embedded system and how they work.

Bibliography:

- https://community.estimote.com/hc/en-us/articles/217429867-What-is-GPIO-
- http://maxembedded.com/2011/06/port-operations-in-avr/
- Wikipedia

Appendix:

main.c

```
#include "led.h"
#include "uart_stdio.h"
#include "button.h"
#include <avr/delay.h>


int main() {

	init();
	LedInit();

	while(1) {
		_delay_ms(100);
		if(isPressed()) {
			ledOn();
			} else {
			ledOff();
		}
	}


	return 0;
}
```

button.h

```
#ifndef BUTTON_H_
#define BUTTON_H_
#include <avr/io.h>

int isPressed();
void init();


	#endif /* BUTTON_H_ */
```

button.c

```
 #include "button.h"

 void init() {
        DDRC &= ~(1 << PORTC5) ; //data direction register coresponding to the port C
(DDRC) initialilizes the port
 }

 int isPressed() {
        return PINC & (1<<PORTC5); //PINC register gets the reading from the input pins
of the MCU
        }
```

led.h

```
#ifndef LED_H_
#define LED_H_
#include <avr/io.h>

void LedInit();
void ledOn();
void ledOff();


#endif /* LED_H_ */#endif
```

Led.c

```
#include "led.h"

void LedInit() {
       DDRA |= (1 << PORTA0);
}

void ledOn() {
       PORTA |= (1 << PORTA0);
}

void ledOff() {
       PORTA &= ~(1 << PORTA0);
       }
```