Education Ministry of Moldova

Technical University

Computer Science and Microelectronics

Report

Embedded Systems

laboratory work #2

Author:
Bîrcu Maxim

Supervisor:
Bragarenco Andrei

November 20, 2016

Introduction

Topic

General Purpose Input/Output registers on AVR.

Objectives

1. Understanding GPIO
2. LED connection
3. Button connection
4. LCD Display device Task integration

Task

Write a C program and schematics for Micro Controller Unit (MCU) using led which will be turned on by pushing on button and turned off when button is released. Additionally use LCD Display which will display current state of led.

1 General-purpose input/output (GPIO)

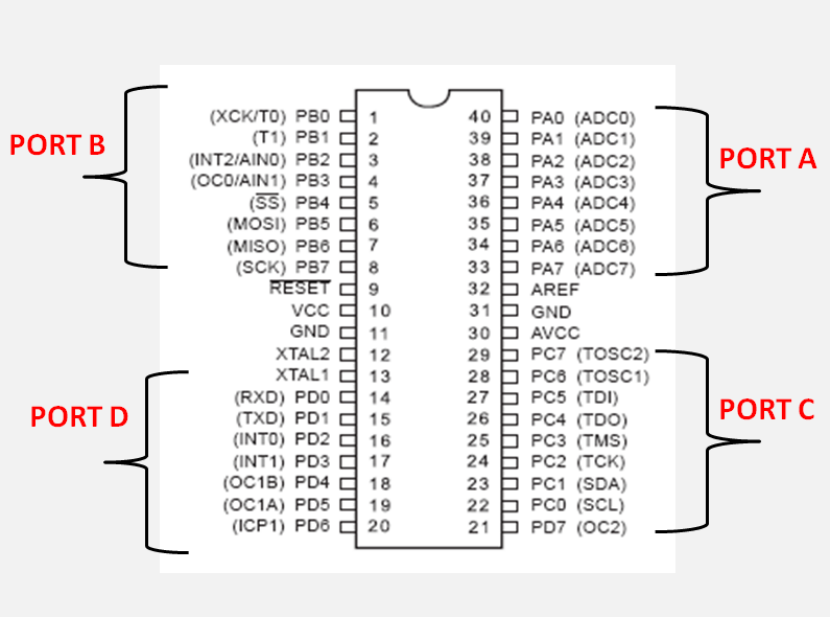
1.1 Definition

General-purpose input/output (GPIO) is a generic pin on an integrated circuit or computer board whose behavior including whether it is an input or output pin is controllable by the user at run time.

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.

1.2 Atmega32 GPIO

Atmega32 has 8-bit port, i.e. it has 8 pins in a single port. Each bit represents a pin i.e. bit 0 represents pin 0 on that port and so on. As you can see in the diagram given below, Atmega32 has 4 ports named as A, B, C & D. Each of these ports has 8-pins (micro-controller pins).

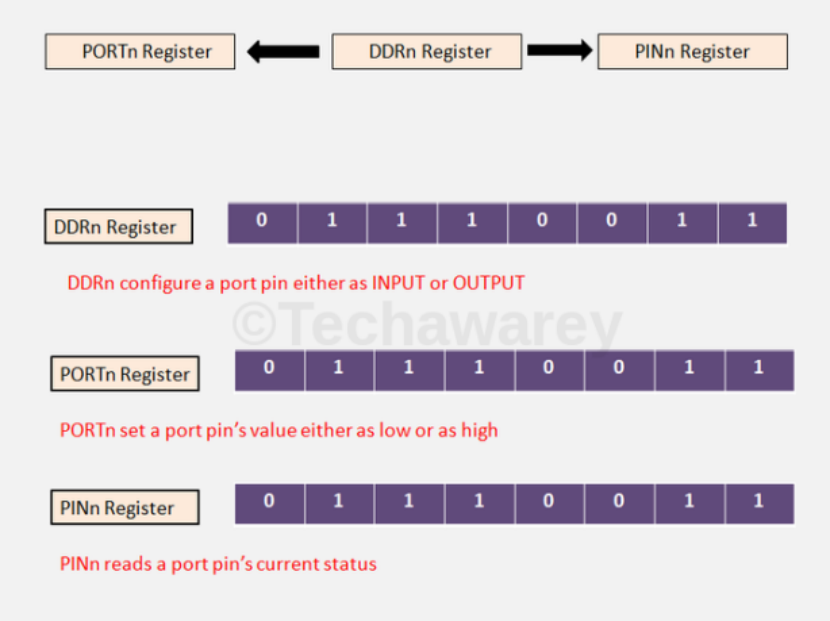


1.3 GPIO REGISTERS

Every GPIO has three registers associated with it in order to control a particular pin. For AVR micro-controllers these registers are:

1. DDRn – Data Direction Register
2. PORTn – Port Output data Register
3. PINn – Port Input Register

n - Indicates the port name i.e. A, B, C & D



1.3.1 DDRn Register

Data Direction Register configures data direction of a port or a port pin. I mean a port will used for input or output.

Writing a value 0 configure that port pin as INPUT and writing a value 1 configures a port pin as OUTPUT.

1.3.2 PORTn Register

PORTn register is used for two purposes :

1. To output data when port is configured as output
2. To activate/deactivate internal pull-up registers when port is configures as input

1.3.3 PIN Register

The PINn register keeps the status of all the pins in that port. By reading this register we can get the current logic level (0 or 1) on the pin. When the pin is configured as input, this register tells what logic level is being given on that pin, whether it's 0 or 1. When the pin is configured as output, this register tells what logic level is being driven out.111

2 Micro Controller

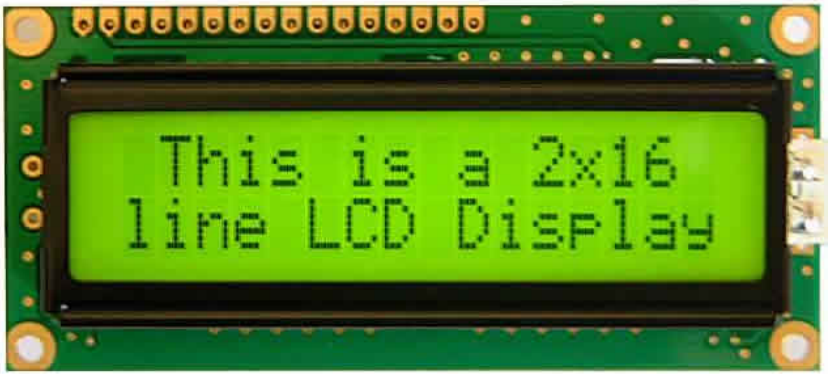
2.1 Definition

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.[1] LCD's 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.

2.2 Usage

LCD's 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 smart-phones. 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.

2.3 LCD Display LM016L



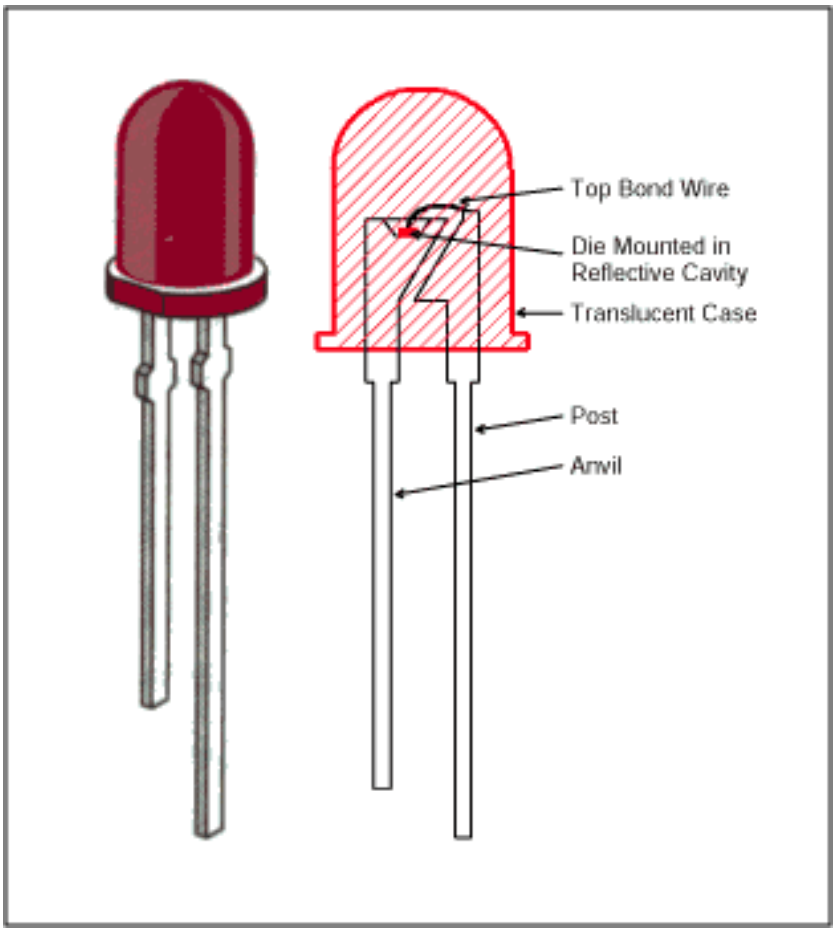
2.4 Character LCD pins with 1 Controller

Pin No.	Name	Description
Pin no. 1	**D7**	Data bus line 7 (MSB)
Pin no. 2	**D6**	Data bus line 6
Pin no. 3	**D5**	Data bus line 5
Pin no. 4	**D4**	Data bus line 4
Pin no. 5	**D3**	Data bus line 3
Pin no. 6	**D2**	Data bus line 2
Pin no. 7	**D1**	Data bus line 1
Pin no. 8	**D0**	Data bus line 0 (LSB)
Pin no. 9	**EN1**	Enable signal for row 0 and 1 (1^{st}controller)
Pin no. 10	**R/W**	0 - Write to LCD module 1 - Read from LCD module
Pin no. 11	**RS**	0 - Instruction input 1 - Data input
Pin no. 12	**VEE**	Contrast adjust
Pin no. 13	**VSS**	Power supply (GND)
Pin no. 14	**VCC**	Power supply (+5V)
Pin no. 15	**EN2**	Enable signal for row 2 and 3 (2^{nd}controller)
Pin no. 16	**NC**	Not Connected

3 Light-emitting diode (LED)

3.1 Definition

A light-emitting diode (LED) is a two-lead semiconductor light source. It is a p–n junction diode, which emits light when activated.[4] When a suitable voltage is applied to the leads, 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.



3.2 Usage

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.

4 Resources

4.1 Atmel Studio

Atmel Studio 7 is the integrated development platform (IDP) for developing and debugging Atmel® SMART ARM®-based and Atmel AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and Atmel SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits. Additionally, Atmel Studio includes Atmel Gallery, an online apps store that allows you to extend your development environment with plug-ins developed by Atmel as well as by third-party tool and embedded software vendors. Atmel Studio 7 can also able seamlessly import your Arduino sketches as C++ projects, providing a simple transition path from Makerspace to Marketplace.

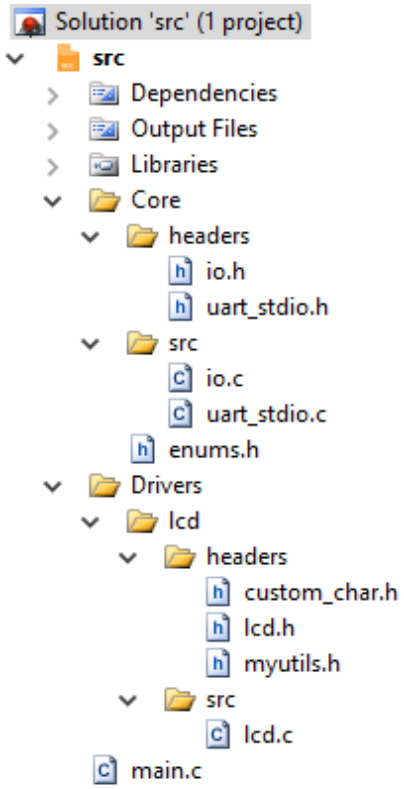
4.2 Proteus

The Proteus Design Suite is an Electronic Design Automation (EDA) tool including schematic capture, simulation and PCB Layout modules. It is developed in Yorkshire, England by Lab center Electronics Ltd with offices in North America and several overseas sales channels. The software runs on the Windows operating system and is available in English, French, Spanish and Chinese languages.

5 Solution

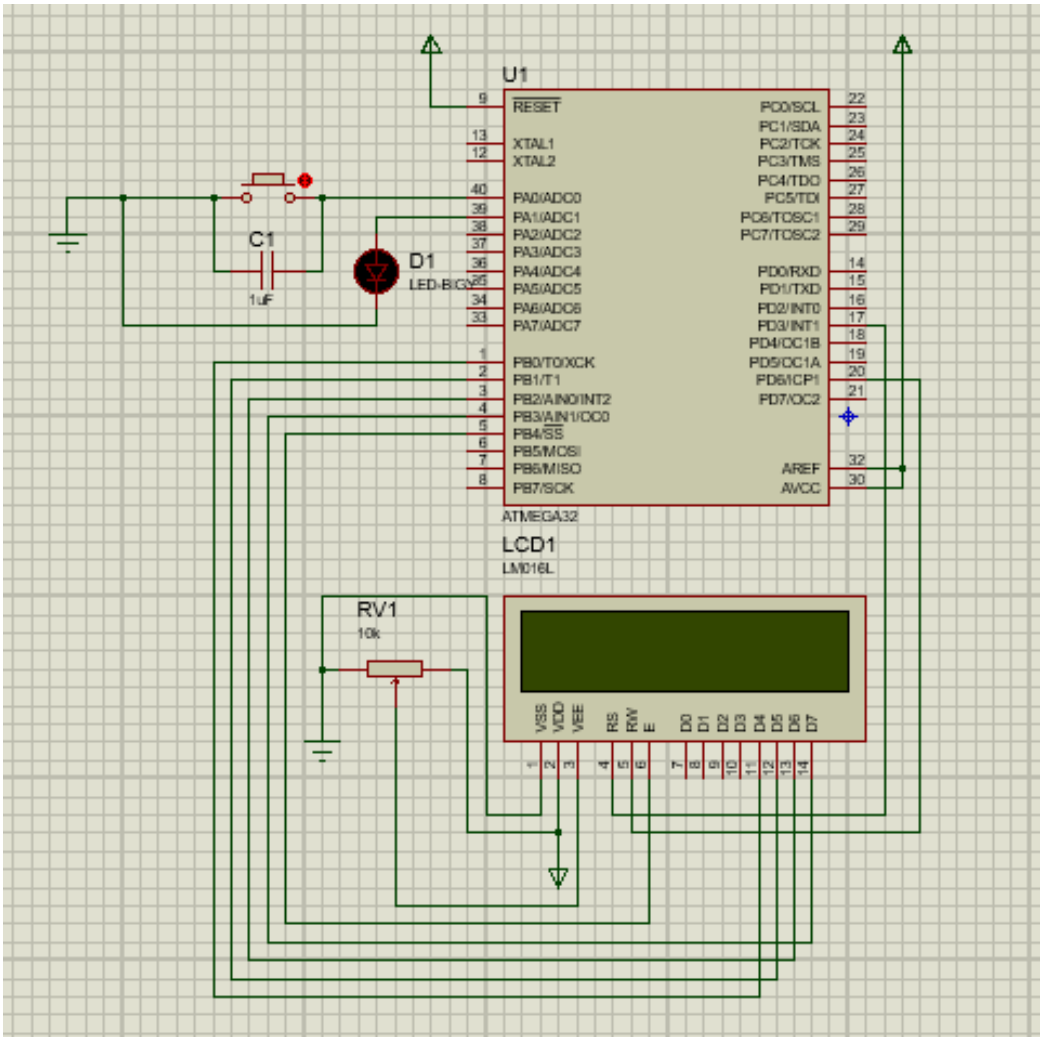
5.1 Project Structure

Project contains 2 folders Core and Drivers. In Core folder I implemented a basic structure that describes a connection to micro-controller and implemented basic function as read write connect. In the second folder Drivers I added drivers for LCD.

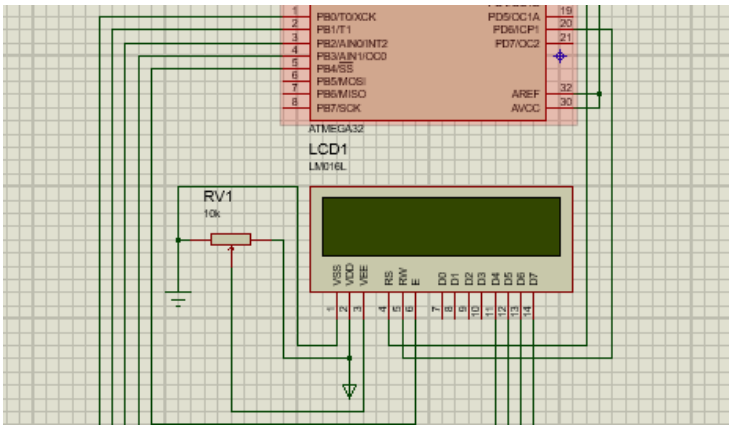


5.2 Circuit in Proteus

This is circuit in Proteus that is composed from a LCD a led and a button

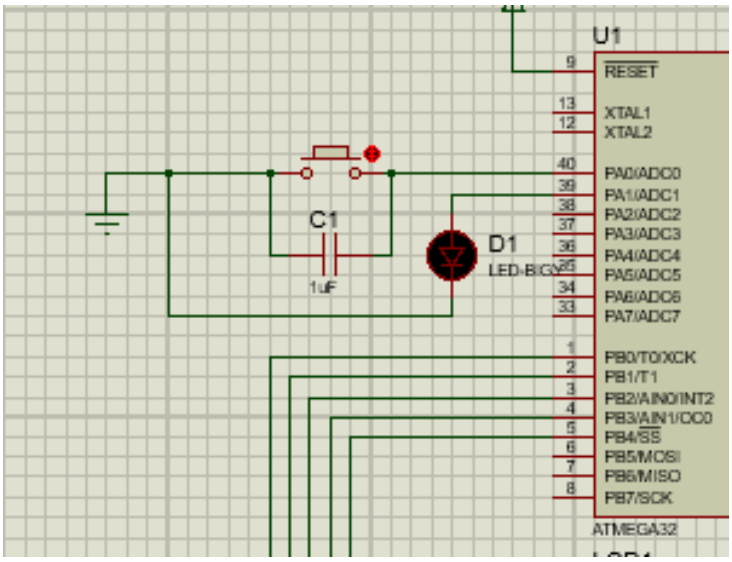


5.2.1 Let's take a look close at LCD connection



5.2.2 Now let's take a look closer at button connection

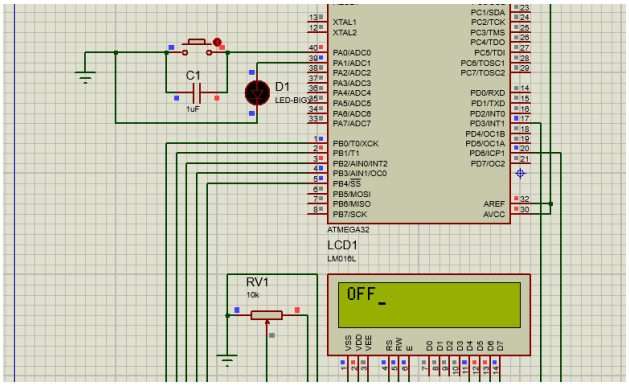
I've connected button via a capacitor to reduce button Contact Bounce problem



5.3 Simulation

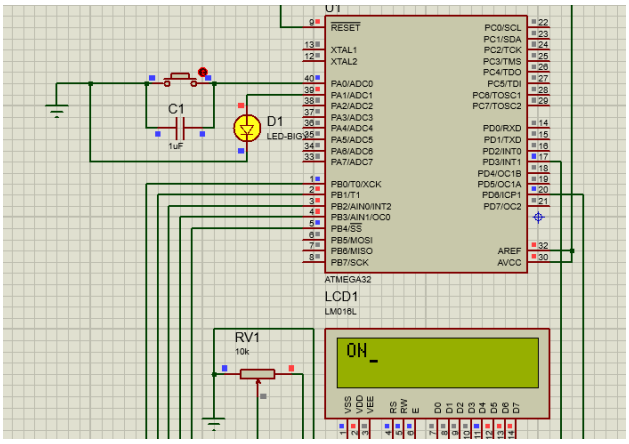
5.3.1 When button is not pressed

The led is not emitting light and LCD outputs Off.



5.3.2 When button is not pressed

The led is emitting light and LCD outputs On.



Conclusion

In this laboratory work I've learned basic concepts of MCU GPIO. I've learned how to connect a peripheral to a MC via ports configure it as in or out and how to send and receive data through this connections.

Apendix

Main

```
/*
 * main.c
 *
 * Created: 10/22/2016 8:39:04 PM
 *  Author: max
 */
#include <avr/io.h>
#include <util/delay.h>
#include "core/headers/io.h"
#include "Drivers/lcd/headers/lcd.h"

void displayMsg(char* lcdMsg){
        LCDClear();
        LCDWriteString(lcdMsg);
}

void main(void)
{
        IO led = registerConnection(1, &DDRA, &PINA, &PORTA);
        IO button = registerConnection(0, &DDRA, &PINA, &PORTA );
        char isPressed = 0;
        LCDInit(LS_ULINE);
        setDataDirectionMode(led, write);
        setDataDirectionMode(button, read);
        writePin(button, high);

    while(1){
                if(readPin(button) == low && !isPressed){
                        writePin(led, high);
                        displayMsg("ON");
                        isPressed = 1;
                } else if (readPin(button) == high && isPressed) {
```

```
                writePin(led, low);
                displayMsg("OFF");
                isPressed = 0;
            }
        }
    }
```

IO header

```
/*
 * IO.h
 *
 * Created: 11/11/2016 8:16:37 AM
 *  Author: max
 */
#ifndef IO_H_
#define IO_H_

#include <avr/io.h>
#include <avr/delay.h>
#include "../enums.h"

typedef struct IO
{
        uint8_t pinIndex;
        uint8_t volatile *dataDirection;
        uint8_t volatile *readBuffer;
        uint8_t volatile *writeBuffer;
} IO;

IO registerConnection(uint8_t pinIndex, uint8_t volatile* dataDirection
void setDataDirectionMode(IO connection, dataDirectionMode direction);
void writePin(IO connection, pinValue pinVal);
pinValue readPin(IO connection) ;

#endif /* IO_H_ */
```

IO src

```
/*
 * IO.c
 *
 * Created: 11/11/2016 8:17:44 AM
 *  Author: max
 */

#include "../headers/io.h"

IO registerConnection(uint8_t pinIndex, uint8_t volatile* dataDirection
        IO drv;
        drv.dataDirection = dataDirection;
        drv.readBuffer  = readBuffer;
        drv.writeBuffer = writeBuffer;
        drv.pinIndex = pinIndex;
        return drv;
}

void setDataDirectionMode(IO connection, dataDirectionMode direction){
        if(direction == read){
                *connection.dataDirection &= ~(1 << connection.pinIndex
        } else {
                *connection.dataDirection |= 1 << connection.pinIndex;
        }
}

void writePin(IO connection, pinValue pinVal){
        if(pinVal == high){
                *connection.writeBuffer |= 1 << connection.pinIndex;
        } else {
                *connection.writeBuffer &= ~(1 << connection.pinIndex);
        }
}

pinValue readPin(IO connection) {
        if(bit_is_clear(*connection.readBuffer, connection.pinIndex))
```

```
                return low;
        else
                return high;
}
```

Enums

```
/*
 * enums.h
 *
 * Created: 11/11/2016 10:14:38 AM
 *  Author: max
 */


#ifndef ENUMS_H_
#define ENUMS_H_

typedef enum dataDirectionMode{
        read,
        write
} dataDirectionMode;

typedef enum pinValue{
        high,
        low
} pinValue;


#endif /* ENUMS_H_ */
```