**Microprocessor Systems Design**

**EEE 42101**

**Experiment 1: Environment Setup and LED Blink**

# Objectives:

* Study Basic peripherals and modules of microcontroller.
* Be familiar with the development environment.
* Put hand on the basics of compilation, linking and loading(flashing).

## Tools:

1. PC
2. Arduino Nano board
3. Testing board
4. MiniB-USB cable

Note: all material and sources of this course will be available on:

<https://github.com/ashrafmalraheem/Mircoprocessor_Course>

Feel free to download, study and modify for your own projects.

# Part 1: Compilation

Create or open the template **main.c** files. Study its content. Ensure that it will compile successfully.

You can add any code make some random arithmetic operations.

You can check the output files that you get: **main.asm, mian.o, mian.hex, ..etc**

# Part 2: Setting Registers

In the platform that you are using: Atmega328p in Arudiuno Nano boards. The LED L5 is connected to Port B pin no. 5. You should set this LED to blink at different rate.

First:

you should configure the direction register as output **DDRB.**

DDRB = 0b00100000; // 0b means binary number

Or

DDRB = 0x20; // 0x means hexadecimal number

Or

DDRB |= 1<<PINB5;

Second:

You can then set a value (0 or 1) in the port to switch on the LED.

PORTB = 0b00100000; // Set high the fifth bit, other bits will be zero!

PORTB = 0b00000000 // set low the fifth bit, other bits will be zero too!

To not affect other bits, you can use digital logic operators (OR, AND, XOR) to manipulate the value of specific bits.

PORTB ^= 0b00100000; // Set high the fifth bit, other bits will not be affected

PORTB &= ~(0b00100000) // set low the fifth bit, other bits will not be affected

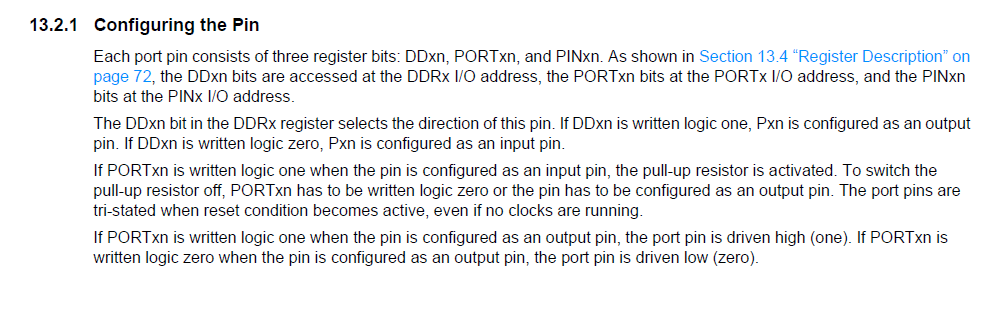


Figure 1 Atmega328p I/O pin configuring

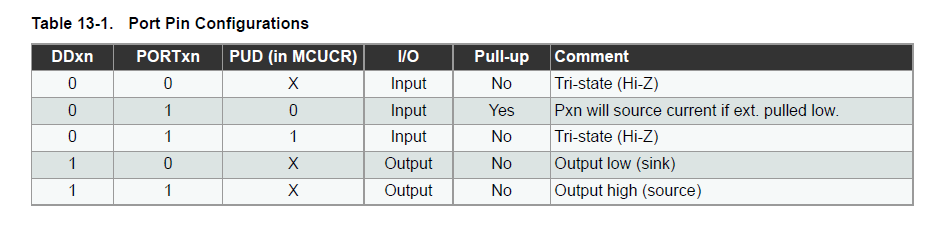


Figure 2 Atmega328p port pin configuration

Use loop to act as a delay. The delay value should be enough to see the LED blink. This Arduino have a 16MHz oscillator. Then F\_CPU is 16MHz. every one instruction will take 1/16M sec to execute.

For(int i=0;i<delay\_value;i++);

Or

while(i<delay\_value){

i++;

}

# Part 3: Include libraries

Instead of using the loops as delays now you should include a library to use the delay function.

#include <delay.h>

\_delay\_ms();

These libraries are precompiled so you can’t find their source code (delay.c). You can only find object codes (**delay.o** or **delay.a**).

You should search for (delay.h) explore it and find other functions. It is on **avr-gcc** folder in the MinGW folder in **C:\MinGW\avr8-gnu-toolchain**. If you didn’t find them, open **delay.h** and study it. You will find it as a inline static functions. What are they?

# Part 4: Make your code smarter

Instead of accessing the registers directly, now you should use functions and macros.

Define a function to set the LED on and OFF depend on the parameter that you pass to them.

## use macros

make some preprocessor macros to help you in coding.

Define macro for each bit:

#define BIT0 0b00000001

#define BIT1 0b00000010

#define BIT2 0b00000100

…etc.

Define the LED pin no.

#define RED\_LED PINB5

Define a macro function to set the LED on and off.

#define RED\_LED\_ON() PORTB |= BIT5

#define RED\_LED\_OFF() PORTB &=~BIT5

Or more general macor:

#define LED\_ON(x) PORTB ^= 1<<x // x is the pin no.

#define LED\_OFF(x) PORTB &=~(1<<x)

## use functions

void LED\_ON(); // function declaration before main

void LED\_ON(){ // function definition after main

// place your code here

}