

1 Table of Contents

Lab assignment 3	1
1.1 Preparation tasks. Submit:	2
1.1.1 Table with data types:	2
1.2 GPIO library. Submit:	3
1.2.1 C code of the application main.c:	3
1.2.2 Listing of library source file gpio.h:	5
1.2.3 Listing of library source file gpio.c:	6
1.2.4 Screenshot_1 of SimulIDE circuit:	8
1.2.5 The difference between the declaration and the definition of the function in C. Give an example: 9	

Lab assignment 3

1.1 Preparation tasks. Submit:

1.1.1 Table with data types:

Data type:	Number of bits	Range	Description:
uint8_t	8	0, 1, ..., 255	Unsigned 8-bit integer
int8_t	8	-128, +127	Signed 8-bit integer
uint16_t	16	0, 1, ..., 65535	Unsigned 16-bit integer
int16_t	16	-32768, + 32767	Signed 16-bit integer
float	32	-3.4e+38, ..., 3.4e+38	Single-precision floating-point

Void pointer size varies system to system. If the system is 16-bit, size of void pointer is 2 bytes. If the system is 32-bit, size of void pointer is 4 bytes. If the system is 64-bit, size of void pointer is 8 bytes.

Here is an example of how to find the size of the void pointer in the C language:

```
#include <stdio.h>
```

```
int main() {  
    void *ptr;  
  
    printf("Pointer size value is: %d", sizeof(ptr));  
    return 0;  
}
```

Output:

"Pointer size value is: 8"

GitHub: <https://github.com/ShalaKreshnik>**Dear Professor, I'm still waiting for you to accept my invitation on GitHub (I made my repository private)*

```
/* second LED */
// WRITE YOUR CODE HERE

GPIO_config_output(&DDRC, LED_RED);
GPIO_write_high(&PORTC, LED_RED); // LED off, because active-high

/* push button */
// WRITE YOUR CODE HERE
GPIO_config_input_pullup(&DDRD, BTN);

// Infinite loop
while (1)
{
    // Pause several milliseconds
    _delay_ms(BLINK_DELAY);

    // WRITE YOUR CODE HERE
    if(GPIO_read(&PORTD, BTN)== 1) // This will check if the button (PIN 0
OF PORTD) has been pressed (Return value is 1 in the function)
    {
        GPIO_toggle(&PORTB, LED_GREEN); // Toggle Green LED PIN 5 of PORTB
        GPIO_toggle(&PORTC, LED_RED);    // Toggle RED LED PIN 0 of PORTC
    }
}

// Will never reach this
return 0;
}
```


GitHub: <https://github.com/ShalaKreshnik>**Dear Professor, I'm still waiting for you to accept my invitation on GitHub (I made my repository private)*

```
void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num); // Declaration of
function (To assign it a value of 1) with parameters *reg_name and pin_num (void does
not return any value)
```

```
void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num); // Declaration of
function (To toggle pin) with parameters *reg_name and pin_num (void does not return
any value)
```

```
/* GPIO_toggle */
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num);

#endif
```

1.2.3 Listing of library source file gpio.c:

```
/*//////////////////////////////////////
///
/// VUT FEKT                               Name and Surname: Kreshnik Shala    ///
/// [BPA-DE2] Digital Electronics 2         Person ID: 226108                  ///
/// Date: Monday, October 12, 2020          ///
/// GitHub: https://github.com/ShalaKreshnik ///
///                                          ///
*//////////////////////////////////////
*****
*
* GPIO library for AVR-GCC.
* ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
*
*****/

/* Includes -----*/
#include "gpio.h"

/* Function definitions -----*/
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name | (1<<pin_num); // Set bit (or)
}

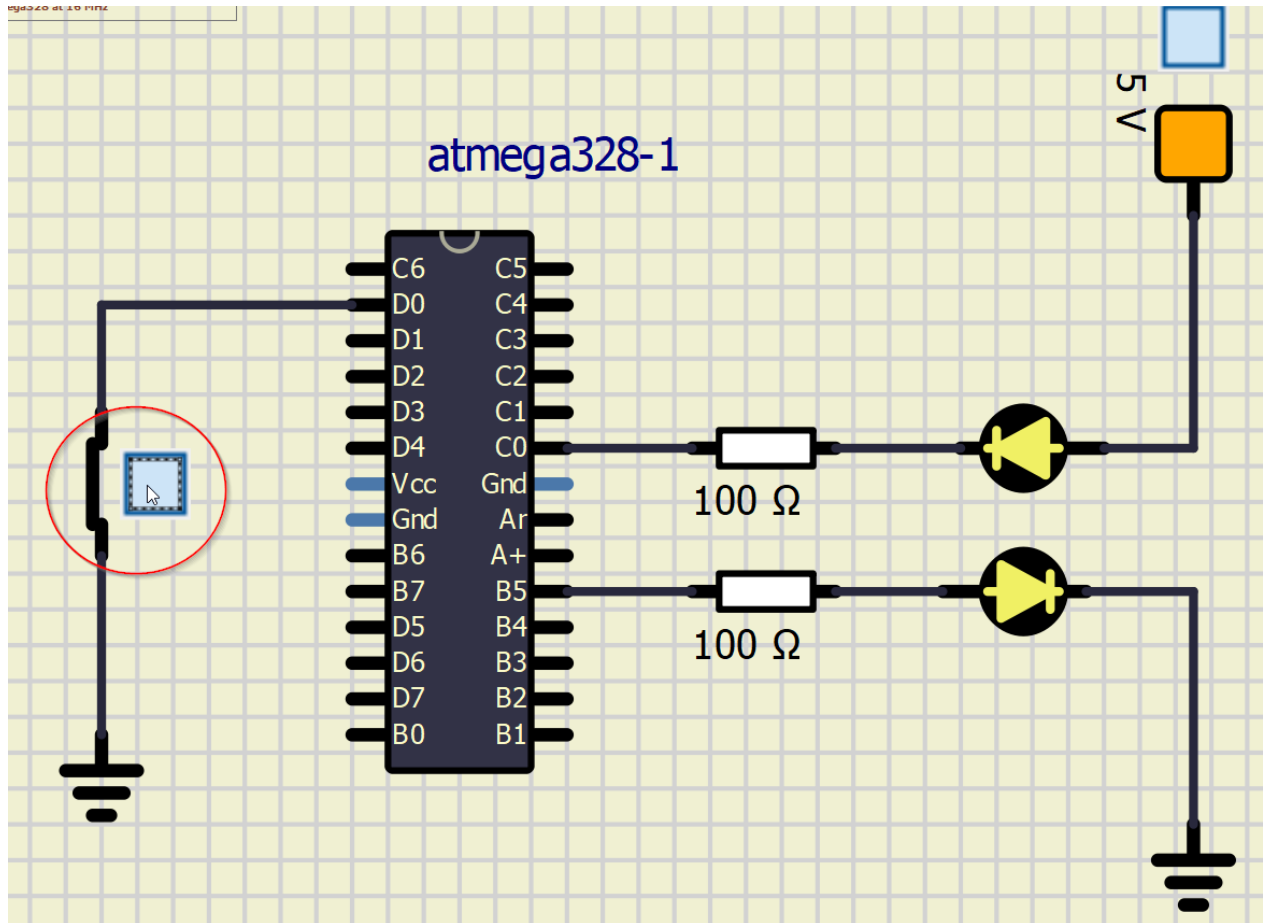
/*-----*/
/* GPIO_config_input_nopull */
void GPIO_config_input_nopull(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num); // Data Direction Register
    *reg_name++;                          // Change pointer to Data Register
    *reg_name = *reg_name & ~(1<<pin_num); // Data Register
}
```

GitHub: <https://github.com/ShalaKreshnik>

**Dear Professor, I'm still waiting for you to accept my invitation on GitHub (I made my repository private)*

```
/*-----*/  
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num)  
{  
    *reg_name = *reg_name & ~(1<<pin_num); // Data Direction Register  
    *reg_name++; // Change pointer to Data Register  
    *reg_name = *reg_name | (1<<pin_num); // Data Register  
}  
  
/*-----*/  
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num)  
{  
    *reg_name = *reg_name & ~(1<<pin_num); //Clear bit (and not)  
}  
  
/*-----*/  
/* GPIO_write_high */  
void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num)  
{  
    *reg_name = *reg_name | (1<<pin_num); // Set bit (or)  
}  
  
/*-----*/  
/* GPIO_toggle */  
void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)  
{  
    *reg_name = *reg_name ^ (1<<pin_num); // Toggle bit (xor)  
}  
  
/*-----*/  
/* GPIO_read */  
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)  
{  
    uint8_t result = 0; // Initiallizing result with 0  
    if ((PIN0 & 0b00000001) == pin_num) // If (PIN0 of PORTD is pressed)  
    {  
        result = 1; // Value of result becomes 1 if PD0 is pressed.  
    }  
    return result; // Return the value stored in result (1 OR 0)  
}
```

1.2.4 Screenshot_1 of SimulIDE circuit:



1.2.5 The difference between the declaration and the definition of the function in C. Give an example:

Declaration of a function gives details about the parameters, return type and its name to the compiler but the definition of a function tells the compiler what task it should do.

For example:

This is our function

```
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
{
    uint8_t result = 0;
    if ((PIND&0b00000001)==pin_num)
    {
        result = 1;
    }
    return result;
}
```

- **Declaration** of the function:

```
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
```

This tells us **uint8_t** is the return type, **GPIO_read** is the name of function and (**volatile uint8_t *reg_name, uint8_t pin_num**) are the parameters of the function

- **Definition** of the function:

```
uint8_t result = 0;
if ((PIND&0b00000001)==pin_num)
{
    result = 1;
}
return result;
```

The set of these lines of instructions to be performed by the compiler is the definition of the function.