[BPA-DE2] Digital Electronics 2

Assignment 3

GitHub: https://github.com/ShalaKreshnik

Name and Suriname: Kreshnik Shala

Person ID: 226108

Date: Monday, October 12, 2020

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

1 Table of Contents

a	b assignmer	nt 3	1
	1.1 Prep	paration tasks. Submit:	2
	1.1.1	Table with data types:	2
	1.2 GPI0	O 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 example:	The difference between the declaration and the definition of the function in C. Give an 9	

Lab assignment 3

[BPA-DE2] Digital Electronics 2

Assignment 3

GitHub: https://github.com/ShalaKreshnik

Date: Monday, October 12, 2020

Person ID: 226108

Name and Suriname: Kreshnik Shala

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

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"

[BPA-DE2] Digital Electronics 2

Assignment 3

GitHub: https://github.com/ShalaKreshnik

Date: Monday, October 12, 2020

Name and Suriname: Kreshnik Shala

Person ID: 226108

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

1.2 GPIO library. Submit:

1.2.1 C code of the application main.c:

```
///
                                                              111
/// 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
                                                              ///
* Alternately toggle two LEDs when a push button is pressed. Use
* functions from GPIO library.
* ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
/* Defines -----*/
#define LED_GREEN PB5
                        // AVR pin where green LED is connected
#define LED_RED
                   PC0
#define BTN
                   PD0
#define BLINK_DELAY 500
#ifndef F_CPU
#define F CPU 16000000 // CPU frequency in Hz required for delay
#endif
/* Includes -----*/
#include <util/delay.h> // Functions for busy-wait delay loops
#include <avr/io.h> // AVR device-specific IO definitions
#include "gpio.h"
                   // GPIO library for AVR-GCC
/* Function definitions -----*/
* Main function where the program execution begins. Toggle two LEDs
* when a push button is pressed. Functions from user-defined GPIO
* library is used instead of low-level logic operations.
*/
int main(void)
   /* GREEN LED */
   GPIO config output(&DDRB, LED GREEN);
   GPIO write low(&PORTB, LED GREEN); // LED off, because active-high
```

[BPA-DE2] Digital Electronics 2

Assignment 3

Person ID: 226108

Date: Monday, October 12, 2020

Name and Suriname: Kreshnik Shala

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;
}
```

[BPA-DE2] Digital Electronics 2



GitHub: https://github.com/ShalaKreshnik

Date: Monday, October 12, 2020

Person ID: 226108

Name and Suriname: Kreshnik Shala

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

1.2.2 Listing of library source file gpio.h:

```
///
                                                               ///
/// 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
#ifndef GPIO H
#define GPIO_H
* GPIO library for AVR-GCC.
* ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
* @file gpio.h
* @brief GPIO library for AVR-GCC.
* @details
* The library contains functions for controlling AVRs' gpio pin(s).
* @note
* Based on AVR Libc Reference Manual. Tested on ATmega328P (Arduino Uno),
* 16 MHz, AVR 8-bit Toolchain 3.6.2.
/* Includes -----*/
#include <avr/io.h>
/* Function prototypes -----*/
* @brief Configure one output pin in Data Direction Register.
* @param reg_name - Address of Data Direction Register, such as &DDRA,
               &DDRB, ...
* @param pin_num - Pin designation in the interval 0 to 7
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num);
/* GPIO config input nopull */
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num);
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num);
```

[BPA-DE2] Digital Electronics 2

Person ID: 226108



Assignment 3

Date: Monday, October 12, 2020

Name and Suriname: Kreshnik Shala

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
                                                         111
/// [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)</pre>
}
/*-----*/
/* 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</pre>
}
```

[BPA-DE2] Digital Electronics 2



GitHub: https://github.com/ShalaKreshnik

B 1B 226400

Name and Suriname: Kreshnik Shala

Person ID: *226108*

Date: Monday, October 12, 2020

```
/*-----*/
void GPIO config input pullup(volatile uint8 t *reg name, uint8 t pin num)
   *reg_name = *reg_name & ~(1<<pin_num); // Data Direction Register</pre>
   *reg_name++;
                             // Change pointer to Data Register
   *reg name = *reg name | (1<<pin num); // Data Register</pre>
}
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num)
{
   *reg name = *reg name & ~(1<<pin num); //Clear bit (and not)</pre>
}
/*_____*/
/* 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)</pre>
}
/*-----*/
/* GPIO_toggle */
void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)
{
     *reg_name = *reg_name ^ (1<<pin_num); // Toggle bit (xor)</pre>
/*-----*/
/* GPIO read */
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
     uint8_t result = 0; // Initiallizing result with 0
     if ((PIND&0b00000001)==pin_num) // If (PINO 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)
}
```

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

[BPA-DE2] Digital Electronics 2

Assignment 3

GitHub: https://github.com/ShalaKreshnik

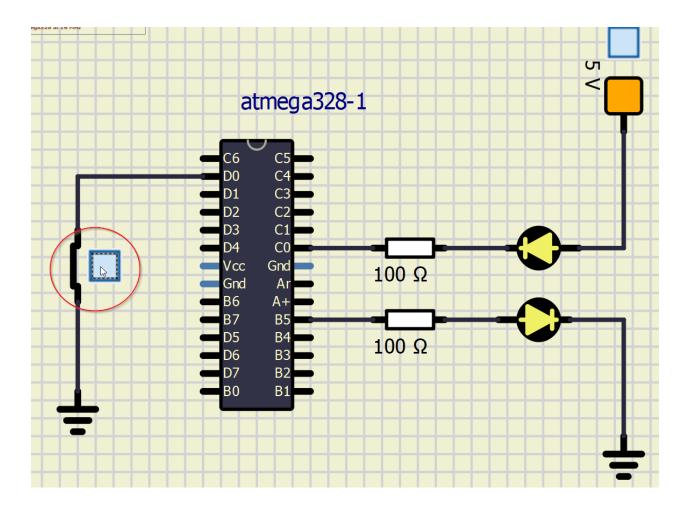
Name and Suriname: Kreshnik Shala

Person ID: 226108

Date: Monday, October 12, 2020

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

1.2.4 Screenshot_1 of SimulIDE circuit:



[BPA-DE2] Digital Electronics 2



GitHub: https://github.com/ShalaKreshnik

Date: Monday, October 12, 2020 *Dear Professor, I'm still waiting for you to accept my invitation on GitHub (I made my repository private)

Name and Suriname: Kreshnik Shala

Person ID: 226108

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&0b0000001)==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.