VUT FEKT
[BPA-DE2] Digital Electronics 2
Date: Monday, Oktober 12.10.2021

Name and Surname: Arjanit Ismajli
Person ID: 240005

Data types in C :

1.Tabele

Data type	Number of bits	Range	Description
Unit8_t	8	0,1,...,255	Unsigned 8-bit integer
Int8_t	8	-128, +127	Signed 8-bit integer
Unit16_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-precison 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.

Name and Surname: Arjanit Ismajli
Person ID: 240005

GPIO library

1.Difference between the declaration and the definition of the function in C

-The declaration is a statement that assures the compiler of the existing variable so that the compiler can proceed for further compilation without requiring the complete details about the variable.

-On the other hand, the definition is a statement that explains the compiler on where and how much storage to create for the variable. Thus, this is the main difference between Declaration and Definition in C.

```
#ifndef
GPIO_H
          # define GPIO_H


          /************************************************************************
           *
           * GPIO library for AVR-GCC.
           * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
           *
           * Copyright (c) 2019-Present Tomas Fryza
           * Dept. of Radio Electronics, Brno University of Technology, Czechia
           * This work is licensed under the terms of the MIT license.
           *
           ************************************************************************/


          /**
           * @file
           * @defgroup fryza_gpio GPIO Library <gpio.h>
           * @code #include "gpio.h" @endcode
           *
           * @brief GPIO library for AVR-GCC.
           *
           * 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.
```

Name and Surname: Arjanit Ismajli
Person ID: 240005

```
 * @author Tomas Fryza, Dept. of Radio Electronics, Brno University
 *         of Technology, Czechia
 * @copyright (c) 2019-2021 Tomas Fryza, This work is licensed under
 *                the terms of the MIT license
 * @{
 */




/* Includes ----------------------------------------------------------*/
#include <avr/io.h>




/* Function prototypes -----------------------------------------------*/
/**
 * @name Functions
 */


/**
 * @brief  Configure one output pin in Data Direction Register.
 * @param  reg_name Address of Data Direction Register, such as &DDRB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return none
 */
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num);




/* GPIO_config_input_nopull */
void GPIO_config_input_nopull(volatile uint8_t *reg_name, uint8_t
pin_num);


/**
 * @brief  Configure one input pin and enable pull-up.
 * @param  reg_name Address of Data Direction Register, such as &DDRB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return none
 */
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t
pin_num);
```

Name and Surname: Arjanit Ismajli
Person ID: 240005

```
/**
 * @brief  Write one pin to a low value.
 * @param  reg_name Address of Port Register, such as &PORTB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return none
 */
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num);




/* GPIO_write_high */


void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num);


/* GPIO_toggle */
void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num);


/**
 * @brief  Read a value from input pin.
 * @param  reg_name Address of Pin Register, such as &PINB
 * @param  pin_num  Pin designation in the interval 0 to 7
 * @return Pin value
 */
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num);


/** @} */


#endif
```

Name and Surname: Arjanit Ismajli
Person ID: 240005

```
/*
 *
 * GPIO library for AVR-GCC.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-Present Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/


/* Includes ----------------------------------------------------------*/
#include "gpio.h" //NEDED for prototypes
#include "avr/sfr_defs.h" //NeDED for set bit
/* Function definitions ----------------------------------------------*/
/**********************************************************************
 * Function: GPIO_config_output()
 * Purpose:  Configure one output pin in Data Direction Register.
 * Input:    reg_name - Address of Data Direction Register, such as &DDRB
 *           pin_num - Pin designation in the interval 0 to 7
 * Returns:  none
 **********************************************************************/
void GPIO_config_output(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name | (1<<pin_num);
}


/**********************************************************************
 * Function: GPIO_config_input_nopull()
 **********************************************************************/


/**********************************************************************
 * Function: GPIO_config_input_pullup()
 * Purpose:  Configure one input pin and enable pull-up.
 * Input:    reg_name - Address of Data Direction Register, such as &DDRB
 *           pin_num - Pin designation in the interval 0 to 7
 * Returns:  none
 **********************************************************************/
void GPIO_config_input_pullup(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num);  // Clear Data Direction Register
    reg_name++;                     // Change pointer to Data Register
    *reg_name = *reg_name & ~(1<<pin_num);   // Clear Data Register
}
```

Name and Surname: Arjanit Ismajli
Person ID: 240005

```
/*************************************************************************
 * Function: GPIO_write_low()
 * Purpose:  Write one pin to a low value.
 * Input:    reg_name - Address of Port Register, such as &PORTB
 *           pin_num - Pin designation in the interval 0 to 7
 * Returns:  none
 ************************************************************************/
void GPIO_write_low(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name & ~(1<<pin_num);
}


/*************************************************************************
 * Function: GPIO_write_high()
 ************************************************************************/
void GPIO_write_high(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name | (1<<pin_num);// set output to high
}
/*************************************************************************
 * Function: GPIO_toggle()
 ************************************************************************/
void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)
{
    *reg_name = *reg_name ^ (1<<pin_num);// set output to high
}
/*************************************************************************
 * Function: GPIO_read()
 ************************************************************************/
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
{
    if(bit_is_set(*reg_name,pin_num))


        return 1;
        else
        return 0;

}
```

```
/*
 * Alternately toggle two LEDs when a push button is pressed. Use
 * functions from GPIO library.
```

Name and Surname: Arjanit Ismajli
Person ID: 240005

```
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-Present Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/


/* Defines -----------------------------------------------------------*/
#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define LED_YELLOW PC0
#define BLINK_DELAY 500
#define BUTTON  PD0
#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
Unit8_t value = 0 ;

/* Function definitions ----------------------------------------------*/
/**********************************************************************
 * Function: Main function where the program execution begins
 * Purpose:  Toggle two LEDs when a push button is pressed. Functions
 *           from user-defined GPIO library is used.
 * Returns:  none
 **********************************************************************/
int main(void)
{
    // Green LED at port B
    GPIO_config_output(&DDRB, LED_GREEN);
    GPIO_write_low(&PORTB, LED_GREEN);


    // Configure the second LED at port C
    GPIO_config_output(&DDRC, LED_YELLOW);
    GPIO_write_low(&PORTC, LED_YELLOW);
    // Configure Push button at port D and enable internal pull-up resistor
GPIO_config_input_pullup(&DDRD,BUTTON) ;
```

Name and Surname: Arjanit Ismajli
Person ID: 240005

```
    // Infinite loop
    while (1)
    {
        // Pause several milliseconds
        _delay_ms(BLINK_DELAY);
                Value=GPIO_read(&PORTD,BUTTON) ;
If(value==1)
{
   GPIO_toggle(&PORTB,LED_GREEN);
GPIO_toggle(&PORTC,LED_YELLOW) ;
}

        // WRITE YOUR CODE HERE
    }


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

3.Scheme of traffic light

Name and Surname: Arjanit Ismajli
Person ID: 240005