DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 3

Name: Demirkan Korbey Baglamac

1)

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, …, 65 535	Signed 16-bit integer
`int16_t`	16	-32 768, .., +32 767	16-bit integer
`float`	32	-3.4e+38, ..., 3.4e+38	Single-precision floating-point
`void`	-	-	-

Example Code:

```c
#include <avr/io.h>

// Function declaration (prototype)
uint16_t calculate(uint8_t, uint8_t);

int main(void)
{
    uint8_t a = 156;
    uint8_t b = 14;
    uint16_t c;

    // Function call
    c = calculate(a, b);

    while (1)
    {
    }
    return 0;
}

// Function definition (body)
uint16_t calculate(uint8_t x, uint8_t y)
{
    uint16_t result;    // result = x^2 + 2xy + y^2

    result = (x*x) + (2*x*y) + (y*y);

    return result;
}
```

2) Listing of library source file gpio.c,

```c
/* 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);
}

/*--------------------------------------------------------------------*/
void GPIO_config_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
}

/*--------------------------------------------------------------------*/
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)
}

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

/*--------------------------------------------------------------------*/
void GPIO_toggle(volatile uint8_t *reg_name, uint8_t pin_num)
{
       *reg_name = *reg_name ^ (1<<pin_num); // Toggle bit (xor)
}

/*--------------------------------------------------------------------*/
uint8_t GPIO_read(volatile uint8_t *reg_name, uint8_t pin_num)
{
       uint8_t result;

       result = *reg_name>>pin_num;
       return result;
}
```

C code of the application main.c,

```c
/* 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

    /* second LED */
    GPIO_config_output(&DDRC, LED_RED);
       GPIO_write_low(&PORTC, LED_RED); //LED off, because active-low

    /* push button */
    GPIO_config_input_pullup(&DDRD, BTN);

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

        if(!(GPIO_read(&PIND, BTN)))
              {
              GPIO_toggle(&PORTB, LED_GREEN);

              GPIO_toggle(&PORTC, LED_RED);
              }
    }

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

In your words, describe the difference between the declaration and the definition of the function in C. Give an example,

The difference between them is function declaration only contains the name of the function and the parameters. But the function definition, as you can understand from the name, contains the body of the function.

For example,

`int add(int, int);` is a function decleration (prototype). But,

`int add(int a, int b) { return a + b; }` is a function definition.