Assignment 1.2

Steinarr Hrafn Höskuldsson

August 25, 2022

Part 1

The following code accomplishes the task of blinking the onboard LED using only register operations.

```
#include <avr/io.h>
#include <util/delay.h>
//#include <stdint.h>

int main(){

  // we are interested in pin 5 on PORT B.
  uint8_t mask = (1 << 5);

  // first, set DDR bit as 1
  DDRB |= mask;

  while (true)
  {
    _delay_ms(1000); // wait 1 second
    PORTB |= mask; // set high
    _delay_ms(1000); // wait 1 second
    PORTB &= ~mask; // set high
  }

  return 0;
}
```

main.cpp

It uses no RAM at compile time and uses only 176 bytes of Flash memory.

Part 2

A digital out driver was constructed with these two files:

```
#include <stdint.h>

class Digital_out
{
public:
    Digital_out(int pin);
    void init();
    void set_hi();
    void set_lo();
    void toggle();
    const bool state() {return _state;}


private:
    bool _state;
    uint8_t pinMask;
};
```

digital_out.h

```
#include "digital_out.h"
#include <avr/io.h>

Digital_out::Digital_out(int pin){
    pinMask = (1<<pin);
}

void Digital_out::init(){
    DDRB |= pinMask;
}

void Digital_out::set_hi()
{
    PORTB |= pinMask;
}

void Digital_out::set_lo()
{
    PORTB &= ~pinMask;
}

void Digital_out::toggle()
{
    PORTB = PORTB ^ pinMask;
}
```

digital_out.cpp

The program given like the program from Part 1 uses no RAM at compile time and this time only 162 bytes of Flash.

Part 3

A digital in driver was constructed with these two files:

```
#include <stdint.h>

class Digital_in
{
public:
    Digital_in(int pin);
    void init();
    void init(bool pull_up);
    bool is_hi();
    bool is_lo();
    bool state();

private:
    uint8_t pinMask;
};
```

digital_in.h

```
#include "digital_in.h"
#include <avr/io.h>

Digital_in::Digital_in(int pin)
{
    pinMask = (1 << pin);
}

void Digital_in::init()
{
    // no pull up asked for
    init(false);
}

void Digital_in::init(bool pull_up)
{
    DDRB &= ~pinMask; // DDR bit to 0
    if (pull_up)
    {
        PORTB |= pinMask; // PORT bit to 1 for pull up
    }
    else
    {
        PORTB &= ~pinMask;
    }
    // set DDR bit to 0
}

bool Digital_in::is_hi()
{
    return (PINB & pinMask);
}

bool Digital_in::is_lo()
{
    return !is_hi();
}
```

digital_in.cpp

and this program accomplishes the task as outlined.

```
#include <util/delay.h>
#include <digital_out.h>
#include <digital_in.h>

int main()
{
  Digital_out led(5); // PB5 Arduino Nano built-in LED on D13

  led.init();

  Digital_in button(1); // PB1, pin 9 on Arduino Nano

  button.init(true); // use internal pull up

  // led.set_hi();
  while (true)
  {
    _delay_ms(150);
    // led.toggle();
    if (button.is_lo())
    {
      led.set_lo();
    }
    else
    {
      led.toggle();
    }
  }
}
```

main.cpp