Reykjavík University

Embedded System Programming

T-738-EMBE

Assignment 1.2: Digital IO driver

26. August 2021

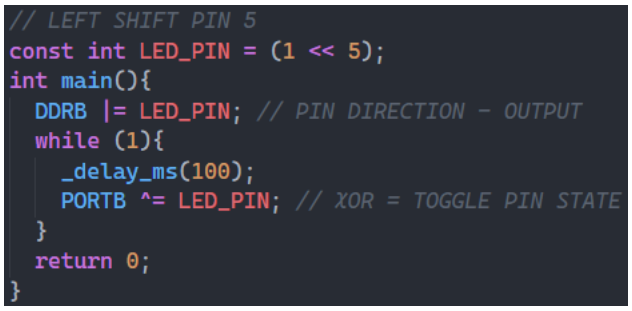
Student:
Sindri Þór Jónsson

Instructor:
Torfi Þórhallsson

1 Part

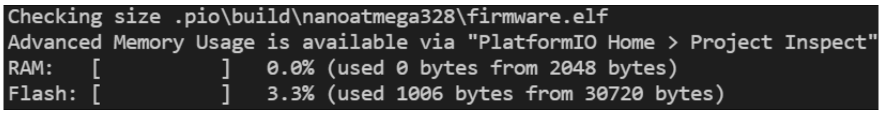
Create an empty Arduino project (containing the main() function). Write a program that blinks the LED using only the register operations from lecture 1.3 Peripherals_Digital_Analog_IO. From the board specifications or the image of the board in lecture 1.4 we see that the Arduino Nano has a (red) LED connected to PB5, that is pin 5 on port B.

To get the inbuilt LED to blink we need to ouput current to the LED pin. This is done by setting the pin direction in the data direction register (DDR) to output and then the state of that pin to high (PORT).
Since the LED is bound to pin 5 in port B the following commands are used to toggle the LED every 100ms:



```
// LEFT SHIFT PIN 5
const int LED_PIN = (1 << 5);
int main(){
  DDRB |= LED_PIN; // PIN DIRECTION - OUTPUT
  while (1){
    _delay_ms(100);
    PORTB ^= LED_PIN; // XOR = TOGGLE PIN STATE
  }
  return 0;
}
```

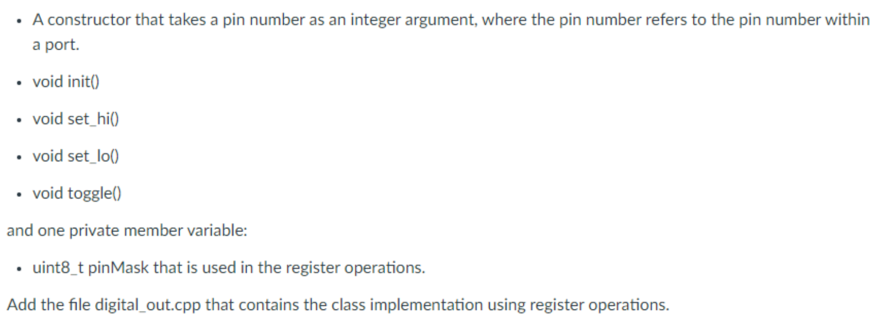
The I/O register names are defined in the included <avr/io.h> package and the delay function in <util/delay.h>.
The memory and code size can be seen in the following image:



```
Checking size .pio\build\nanoatmega328\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [          ]   0.0% (used 0 bytes from 2048 bytes)
Flash: [          ]   3.3% (used 1006 bytes from 30720 bytes)
```

2 Part

Add the file digital_out.h to the project that defines the driver API. In this file declare the C++ class Digital_out that has the following member functions:



- A constructor that takes a pin number as an integer argument, where the pin number refers to the pin number within a port.
- void init()
- void set_hi()
- void set_lo()
- void toggle()

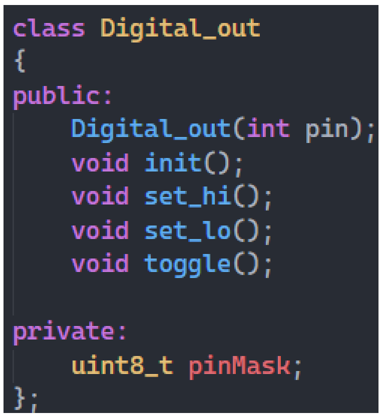
and one private member variable:

- uint8_t pinMask that is used in the register operations.

Add the file digital_out.cpp that contains the class implementation using register operations.

Change the application in main.cpp to use the new driver. Compile and test the application and note the new memory and code sizes. What is the added footprint of using the driver as implemented? Write down the results.

The class Digital_out is created as specified:

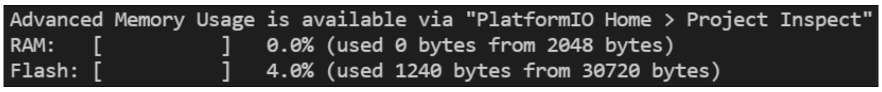


```
class Digital_out
{
public:
    Digital_out(int pin);
    void init();
    void set_hi();
    void set_lo();
    void toggle();

private:
    uint8_t pinMask;
};
```

The constructor takes in a I/O pin number on the B register as an integer and converts it into the private member variable "pinMask". The file digital_out.cpp contains the class implementation using register operations to toggle the LED every 100ms.

The memory and code size can be seen in the following image:

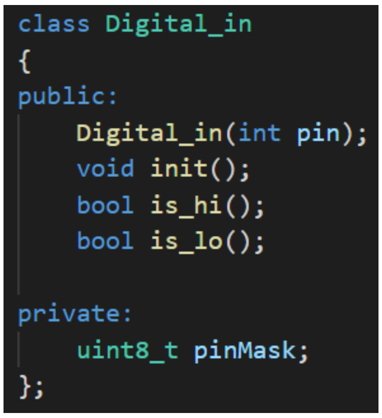


```
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [          ]   0.0% (used 0 bytes from 2048 bytes)
Flash: [          ]   4.0% (used 1240 bytes from 30720 bytes)
```

3 Part

Add the digital input driver class Digital_in following the same pattern as in Part 2. You could perhaps define the member functions bool is_hi() and bool is_lo(). Separate the driver interface definition and the driver implementation into two files: digital_in.h and digital_in.cpp. Use this class to drive a button that disables the blinking when pressed.

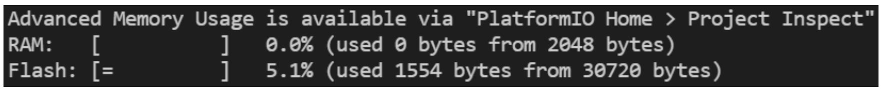
Now the header file "digitial_in.h" is added which contains the class "Digital_in" as specified:



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

private:
    uint8_t pinMask;
};
```

The constructor takes in a I/O pin number on the B register as an integer and converts it into the private member variable "pinMask". The file digital_in.cpp contains the class implementation using register operations like in part 2.
Pin1 on PORTB is enabled as the input reading for the button (switch in my case) with internal pull up enabled. With register operations in the class functions the LED is toggled every 100ms but when the button is pressed the LED turns off until the button is released.
The memory and code size can be seen in the following image:



```
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [          ]   0.0% (used 0 bytes from 2048 bytes)
Flash: [=         ]   5.1% (used 1554 bytes from 30720 bytes)
```

From the RAM and Flash usage report in each part:

	RAM(bytes)	**FLASH(bytes)**
P1	0	1006
P2	0	1240
P3	0	1554

We can see that the RAM usage calculated in platformIO is 0bytes for each part, the flash memory size increases in each part (larger code size).

P3: Video of switch and LED: VIDEO P3