Chase Griswold

ECE 6780-003

TA: Bailey Martin

Post Lab_01 Questions

1.1 — Postlab 1. Please answer the following questions and hand in as your postlab for Lab 1.

1. What are the GPIO control registers that the lab mentions? Briefly describe each of their functions.

> For example, GPIOC – PUPDR is a register that sets the pull up/pull down state of the onboard resistors for a specific pin.
>
> In this lab, we used GPIOC to control the LEDs and GPIOA to control the USER button. The specific registers to set up this control were: MODER, OSPEEDR, OTYPER, and PUPDR.

2. What values would you want to write to the bits controlling a pin in the GPIOx_MODER register in order to set it to analog mode?

> Analog mode is 11
>
> For example, if I wanted to set pin 4 (in this case MODER4) of GPIOA to analog mode, I would write:
>
> GPIOA->MODER |= (1<<8);
>
> GPIOA->MODER |= (1<<9);
>
> Or we could do it like this
>
> GPIOA->MODER |= (3<<8); NOTE: this works because binary 11 = 3 in decimal.

### 8.4.1 GPIO port mode register (GPIOx_MODER) (x =A to F)

Address offset:0x00

Reset value: 0x2800 0000 for port A

Reset value: 0x0000 0000 for other ports

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODER15[1:0] | | MODER14[1:0] | | MODER13[1:0] | | MODER12[1:0] | | MODER11[1:0] | | MODER10[1:0] | | MODER9[1:0] | | MODER8[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MODER7[1:0] | | MODER6[1:0] | | MODER5[1:0] | | MODER4[1:0] | | MODER3[1:0] | | MODER2[1:0] | | MODER1[1:0] | | MODER0[1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **MODER[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

3. Examine the bit descriptions in GPIOx_BSRR register: which bit would you want to set to clear the fourth bit in the ODR?

To accomplish this, we can note from the datasheet that a 'clear' (assuming reset) of the fourth bit corresponds to 20.

To reset the corresponding ODR bit 4, we would write

GPIOx->BSRR |= (1<<20);

### 8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to F)

Address offset: 0x18

Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BR15 | BR14 | BR13 | BR12 | BR11 | BR10 | BR9 | BR8 | BR7 | BR6 | BR5 | BR4 | BR3 | BR2 | BR1 | BR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BS15 | BS14 | BS13 | BS12 | BS11 | BS10 | BS9 | BS8 | BS7 | BS6 | BS5 | BS4 | BS3 | BS2 | BS1 | BS0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit
*Note: If both BSx and BRx are set, BSx has priority.*

4. Perform the following bitwise operations:

- 0xAD | 0xC7 = 0xEF

- 0xAD & 0xC7 = 0xC5
- 0xAD & ~(0xC7) = 0x28
- 0xAD ^0xC7 = 0x6A

5. How would you clear the 5th and 6th bits in a register while leaving the other's alone?

A simple method is using &= ~ and shifting to the bits in question while specifying the specific register in question.

For example

GPIOx->MODER &= ~(1<<6) would clear the 6th bit (set it to 0, LOW) in the MODER register.

And GPIOx-MODER &= ~(1<<5) would clear the 5th bit in MODER.

6. What is the maximum speed the STM32F072R8 GPIO pins can handle in the lowest speed setting?

• Use the chip datasheet: lab section 1.4.1 gives a hint to the location. You'll want to search the I/O AC characteristics table. You will also need to view the OSPEEDR settings to find the bit pattern indicating the slowest speed.

Bits 31:0 **OSPEEDR[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output speed.
x0: Low speed
01: Medium speed
11: High speed

**Table 55. I/O AC characteristics(1)(2)**

| OSPEEDRy [1:0] value(1) | Symbol | Parameter | Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| x0 | $f_{max(IO)out}$ | Maximum frequency(3) | $C_L$ = 50 pF, $V_{DDIOx} \geq 2$ V | - | 2 | MHz |
| | $t_{f(IO)out}$ | Output fall time | | - | 125 | ns |
| | $t_{r(IO)out}$ | Output rise time | | - | 125 | |
| | $f_{max(IO)out}$ | Maximum frequency(3) | $C_L$ = 50 pF, $V_{DDIOx} < 2$ V | - | 1 | MHz |
| | $t_{f(IO)out}$ | Output fall time | | - | 125 | ns |
| | $t_{r(IO)out}$ | Output rise time | | - | 125 | |

The maximum speed is 2 MHz if the V_DDIOx is greater than or equal to 2V.

The maximum speed is 1 MHz if the V_DDIOx is less than 2V.

7. What RCC register would you manipulate to enable the following peripherals: (use the comments next to the bit defines for better peripheral descriptions)

• TIM1 (TIMER1)

### 6.4.7 APB peripheral clock enable register 2 (RCC_APB2ENR)

Bit 11 **TIM1EN:** TIM1 timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled
1: TIM1P timer clock enabled

To enable TIM1 we would need to write RCC->APB2ENR |= (1<<11);

• DMA1

### 6.4.6 AHB peripheral clock enable register (RCC_AHBENR)

Bit 0 **DMAEN:** DMA clock enable
Set and cleared by software.
0: DMA clock disabled
1: DMA clock enabled

To enable DMA1 we would need to write RCC->AHB2ENR |= (1<<0);

• I2C1

### 6.4.8 APB peripheral clock enable register 1 (RCC_APB1ENR)

Bit 21 **I2C1EN:** I2C1 clock enable
Set and cleared by software.
0: I2C1 clock disabled
1: I2C1 clock enabled

To enable I2C1 we would need to write RCC->APB1ENR |= (1<<21);