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

The GPIO control registers mentioned in the lab are:

* Accessed via RCC
  + AHBENR (for enabling peripheral clocks to GPIOs)
* Accessed via GPIO channels A/B/C/…
  + MODER (for setting input/output modes)
  + OTYPER (for setting output types such as push-pull)
  + OSPEEDR (for setting output speeds. In this lab we’re setting it to “low speed”)
  + PUPDR (for setting pull-ups or pull-downs)
  + IDR (input data register for receiving data)
  + ODR (output data register for sending data)

**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?**

We would want to write “11” to the bits controlling a pin in GPIOx\_MODER to set it to analog mode.

**Examine the bit descriptions in GPIOx\_BSRR register: which bit would you want to set to clear the fourth bit in the ODR?**

We would want to set bit 19 in order to clear the fourth bit in the ODR via BSRR.

**Perform the following bitwise operations:**

**• 0xAD | 0xC7 = 10101101 | 11000111 = 11101111 = 0xEF**

**• 0xAD & 0xC7 = 10101101 & 11000111 = 10000101 = 0x85**

**• 0xAD & ~(0xC7) = 10101101 & ~(11000111) = 10101101 & 00111000 = 00101000 = 0x28**

**• 0xAD ^0xC7 = 10101101 ^ 11000111 = 01101010 = 0x6A**

**How would you clear the 5th and 6th bits in a register while leaving the others alone?**

Use an inverted and bit-shifted set. For example:

* GPIOC->MODER &= ~(1 << 5);
* GPIOC->MODER &= ~(1 << 6);

Alternately, you could use a bit mask:

* GPIOC->MODER &= ~(0x60);

**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.**

At the lowest speed setting, the maximum frequency that the GPIO pins can handle is 2MHz (assuming V\_DDIOx >= 2V)

**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)**

RCC->APB1ENR |= RCC\_APB1ENR\_TIM1EN

**• DMA1**

RCC->AHBENR |= RCC\_AHBENR\_DMA1EN

**• I2C1**

I2C1->CR1 = I2C\_CR1\_PE