#HW, cec32x_HW_24_HW3_embedded_C_and_GPIO, 135 points total.

HW 3. Embedded C programming and GPIO

Prob 3-1. (30 points total, 5 points each) Assume `A` is a uint16_t variable, written as 0x $H_3H_2H_1H_0$, intended to express an arbitrary number. Here H_k is the kth hexadecimal digit counted from the right to the left. Write the C code to perform the following operations to change the digit (4 bits) of interest without affecting the other digits (bits) or shift the number:

1. A = 0x$H_3H_2H_1H_0$ => A = 0x$H_3H_20H_0$ (for example A = 0x1234 => A = 0x1204)
2. A = 0x$H_3H_2H_1H_0$ => A = 0x$H_3H_2FH_0$
3. A = 0x$H_3H_2H_1H_0$ => A = 0x$H_3H_26H_0$
4. A = 0x$H_3H_2H_1H_0$ => A = 0x$H_3H_29H_0$
5. A = 0x$H_3H_2H_1H_0$ => A = 0x$H_3H_2\bar{H}_1H_0$ where 0x$\bar{H}_1 = 15-$ 0xH_1
6. A = 0x$H_3H_2H_1H_0$ => A = 0x$0H_3H_2H_1$

Prob 3-2. (10 points) Describe briefly what is a wired-AND logic connection (or wired-AND bus).

Prob 3-3. (30 points total, 5 points each) Determine what mode (push-pull, open-drain, pull-up, and pull-down) you will use for the following situations:

Soln to Prob 3-3.

1. Input, seeing HiZ as 0. (HiZ stands for high impedance, meaning the input is not connected to a source. For example, when the push button on Page 7 of class notes for Module 3 is not pressed.) Pull-down
2. Input, seeing HiZ as 1. Pull-up
3. Output, using wired-AND logic. Open-drain
4. Output, pushing current to the load from the output pin. Push-pull
5. Output, turning on the LED when the output is low from the GPIO pin.
6. Output, turning on the LED when the output is high from the GPIO pin. 6. Push-pull

5. Mainly open-drain. Push-pull will work as well---just a bit wierd.

Prob 3-4. (20 points) Consider the three-line code in the middle of page 14 of the class notes for Module 3 (below the figure for the AHB2 enable register). Answer the following questions:

1. (10 points) What is the value of `RCC_AHB2ENR_GPIOBEN` in hexadecimal?
2. (10 points) How to define the value of `RCC_AHB2ENR_GPIODEN` in a similar approach?

Prob 3-5. (20 points) Consider the control of OTYPER for GPIOC on page 15 of the class notes for Module 3. Answer the following questions:

1. (10 points) If the value of OTYPER is `0x0000_1234`. What is the value after the statement `GPIOC->OTYPER |= (1U << 2)`. (Give your solution in hexadecimal.)

2. (10 points) If we want to change the output type of Pin 2 to push-pull, write a ONE-line C statement like the one given above to perform this.

Prob 3-6 (25 points) We can better understand the code when we can explain each part easily. Consider the line of code in Section 4.3.5 on page 17 of the class notes for Module 3.

1. (5 points) What is the value of `GPIO_PIN_1`?
2. (5 points) If the key on PA1 is pressed and the value in IDR corresponding to Pin 1 is 1, what is the value of `GPIOA->IDR & GPIO_PIN_1`?
3. (5 points) Continue the above question. What is the value of `(GPIOA->IDR & GPIO_PIN_1) == GPIO_PIN_1`?
4. (5 points) If the key on PA1 is released and the value in IDR corresponding to Pin 1 is 0, what is the value of `GPIOA->IDR & GPIO_PIN_1`?
5. (5 points) Continue the above question. What is the value of `(GPIOA->IDR & GPIO_PIN_1) == GPIO_PIN_1`?

```
Soln to Prob 3-1

a. A &= 0xFF0F;
b. A |= 0x00F0;
c. A &= 0xFF0F;
   A |= 0x0060;
d. A &= 0xFF0F;
   A |= 0x0090;
e. A ^= 0x00F0;
f. A >>= 4;

Or the preferred soln:
a. A &= ~(15u << 4);
b. A |= 15u << 4;
c. A &= ~(15u << 4);
   A |= 6u << 4;
d. A &= ~(15u << 4);
   A |= 9u << 4;
e. A ^= 15u << 4;
f. A >>= 4;
```

Soln to Prob 3-2. When two or more open-drain outputs are connected together---wired together, one 0 output will lead to the entire output to be 0. This is equivalent to the AND logic operation, and hence this is called a wired-and circuit.

```
Soln to Prob 3-4.
1.
RCC_AHB2ENR_GPIOBEN = 0x0000_0002

2.
#define RCC_AHB2ENR_GPIODEN_Pos      (3U)
#define RCC_AHB2ENR_GPIODEN_Msk      (0x1U << RCC_AHB2ENR_GPIODEN_Pos)
#define RCC_AHB2ENR_GPIODEN          RCC_AHB2ENR_GPIODEN_Msk
```

```
Soln to Prob 3-5.
1.
The value of GPIOB->OTYPER is 0x0000_1234.

2.
GPIOB->OTYPER &= ~(1U << 4);
```

```
Soln to Prob 3-6.

1. GPIO_PIN_1 = 0x0000_0002

2. GPIOA->IDR & GPIO_PIN_1 = 0x0000_0002

3. True

4. GPIOA->IDR & GPIO_PIN_1 = 0x0000_0000

5. False
```