**CG2271 Real Time Operating Systems**

**Lab 1 – ARM Cortex M0+ GPIO Programming – ANSWER BOOK**

|  |  |  |
| --- | --- | --- |
|  | Name | Student ID (Axxxxx) |
| 1. | John Woo Yi Kai | A0272561E |
| 2. | Rishi Moorthy | A0273618X |
| 3. | Wang Yanjie | A0276202M |
| 4. | Kenneth Wong Cun Wi | A0303203A |
| 5. | Kuek Yeau Hao Jonathan | A0258485M |

Question 1. (1 marks)

The LEDs are active low. When the pins are set to LOW, the 3V3 powers the LED and the pins sink the current, turning the LEDs on. When the pins are pulled HIGH at 3V3, there is no potential difference and thus current across the LEDs, turning the LEDs off. Hence, the LEDs are considered active low.

Question 2. (1 marks)

SW3 is active low. PTA4 is pulled to HIGH by the 10k pull-up resistor when the SW3 is open. When the switch is pressed (active), the current is shorted to ground, such that PTA4 is LOW at 0V. Hence, SW3 is active low.

Question 3. (4 marks)

|  |  |
| --- | --- |
| Line | Brief Explanation |
| SIM->SCGC5 |=  SIM\_SCGC5\_PORTE\_MASK; | This line turns on the clock for GPIOE using the System Clock Gating Control Register 5 (SCGC5). This is done by using the `|=` operator on the bitmask for SIM\_SCGC5\_PORTE\_MASK, which sets bit 5 of SCGC5, turning the clock on. |
| PORTE->PCR[RED\_PIN]&= ~PORT\_PCR\_MUX\_MASK; | This line clears the MUX bits for the pin used for Red LED, which is important just in case of trailing set bits from before. This is done by first accessing pin x of port E, where x is the pin defined for the Red LED. Using the `&=` operator and negating the MUX mask stated, to clear the specific bits with `1` in the mask to 0. |
| PORTE->PCR[RED\_PIN]= PORT\_PCR\_MUX(1); | This line chooses GPIO function for the pin used for Red LED. This is done by first accessing pin x of port E also, where x is the pin defined for the Red LED. Since we want to choose GPIO for this pin (Port E Pin 1) for the Red LED, we use the ALT1 function to configure PTE1 with MUX = 1, which is what this line does. |
| GPIOE->PDDR |=  (1 << RED\_PIN); | This line sets the red pin as output. |

Question 4. (4 marks)

**void** **initGPIO**() {

  SIM->SCGC5 |= (SIM\_SCGC5\_PORTA\_MASK | SIM\_SCGC5\_PORTD\_MASK | SIM\_SCGC5\_PORTE\_MASK);  
  PORTE->PCR[RED\_PIN] &= ~PORT\_PCR\_MUX\_MASK;  
  PORTE->PCR[RED\_PIN] |= PORT\_PCR\_MUX(1);  
  PORTE->PCR[BLUE\_PIN] &= ~PORT\_PCR\_MUX\_MASK;  
  PORTE->PCR[BLUE\_PIN] |= PORT\_PCR\_MUX(1);  
  PORTD->PCR[GREEN\_PIN] &= ~PORT\_PCR\_MUX\_MASK;  
  PORTD->PCR[GREEN\_PIN] |= PORT\_PCR\_MUX(1);  
  PORTA->PCR[SWITCH\_PIN] &= ~PORT\_PCR\_MUX\_MASK;  
  PORTA->PCR[SWITCH\_PIN] |= PORT\_PCR\_MUX(1);  
  
  GPIOA->PDDR &= ~(1 << SWITCH\_PIN);  
  GPIOE->PDDR |= (1 << RED\_PIN | 1 << BLUE\_PIN);  
  GPIOD->PDDR |= (1 << GREEN\_PIN);

}

Question 5. (3 marks)

Note to TA: -1 mark for each incorrect or missing “turn on”

void ledOn(TLED led) {  
  switch(led) {  
  case RED:  
    // Code to turn on RED LED  
    GPIOE->PCOR |= (1 << RED\_PIN);  
    break;  
  
  case GREEN:  
    // Code to turn on GREEN LED  
    GPIOD->PCOR |= (1 << GREEN\_PIN);  
    break;  
  
  case BLUE:  
    // Code to turn on BLUE LED  
    GPIOE->PCOR |= (1 << BLUE\_PIN);  
    break;  
  }  
}

Question 6 (3 marks)

Note to TA: -1 mark for each incorrect or missing “turn off”

void ledOff(TLED led) {  
  switch(led) {  
  case RED:  
    // Turn off RED led here  
    GPIOE->PSOR |= (1 << RED\_PIN);  
    break;  
  
  case GREEN:  
    // Turn off GREEN led here  
    GPIOD->PSOR |= (1 << GREEN\_PIN);  
    break;  
  
  case BLUE:  
    // Turn off BLUE led here  
    GPIOE->PSOR |= (1 << BLUE\_PIN);  
    break;  
  }  
}

Question 7 (1 mark)

   while(1) {  
      if(!(GPIOA->PDIR & (0b1 << SWITCH\_PIN))) {  
          switch(count) {  
          case 0:  
            ledOn(RED);  
            break;  
          case 1:  
            ledOn(GREEN);  
            break;  
          case 2:  
            ledOn(BLUE);  
            break;  
          case 3:  
            ledOff(BLUE);  
            break;  
          case 4:  
            ledOff(GREEN);  
            break;  
          case 5:  
            ledOff(RED);  
            break;  
  
          default:  
            count=0;  
  
          }  
      }  
          count = (count + 1) % 6;  
    }

Demo: 3 marks

**TOTAL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / 20**