**CG2271 Real Time Operating Systems**

**Lab 4 – Answer Book**

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

**Question 1** (2 marks)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LED** | **PIN** | **ALT** | **TPMx (TPM0, 1 or 2)** | **Channel** |
| RED | PTE31 | 3 | TPM0 | CH4 |
| GREEN | PTD5 | 4 | TPM0 | CH5 |
| BLUE | PTE29 | 3 | TPM0 | CH2 |

**Question 2** (2 marks)

Low True. Since the LEDs are active low, for the LEDs to turn on, it will be during the off-period of a PWM cycle. Using low-true PWM will have duty = % of time output is LOW, such that duty cycle maps directly to perceived brightness.

**Question 3.** (6 marks) initPWM:

// Our LEDs are connected to:

// Red: PTE31 TPM0 CH 4 ALT3

// Green: PTD5 TPM0 CH 5 ALT4

// Blue: PTE29 TPM0 CH 2 ALT3

**void** **initPWM**() {

// Turn on clock gating to TPM0

SIM->SCGC6 |= SIM\_SCGC6\_TPM0\_MASK;

// Turn on clock gating to the ports

SIM->SCGC5 |= (SIM\_SCGC5\_PORTE\_MASK | SIM\_SCGC5\_PORTD\_MASK);

// Set the pin multiplexors for PWM

PORTE->PCR[RED\_PIN] &= ~PORT\_PCR\_MUX\_MASK;

PORTD->PCR[GREEN\_PIN] &= ~PORT\_PCR\_MUX\_MASK;

PORTE->PCR[BLUE\_PIN] &= ~PORT\_PCR\_MUX\_MASK;

PORTE->PCR[RED\_PIN] = PORT\_PCR\_MUX(3); // ALT3 for TPM0\_CH4 (Red)

PORTD->PCR[GREEN\_PIN] = PORT\_PCR\_MUX(4); // ALT4 for TPM0\_CH5 (Green)

PORTE->PCR[BLUE\_PIN] = PORT\_PCR\_MUX(3); // ALT3 for TPM0\_CH2 (Blue)

// Set pins to output

GPIOE->PDDR |= (1 << RED\_PIN);

GPIOD->PDDR |= (1 << GREEN\_PIN);

GPIOE->PDDR |= (1 << BLUE\_PIN);

// Set up TPM0

// Turn off TPM0 and clear the prescalar field

TPM0->SC &= ~(TPM\_SC\_CMOD\_MASK | TPM\_SC\_PS\_MASK);

// Set prescalar of 128

TPM0->SC |= TPM\_SC\_PS(0x7);

// Select centre-aligned PWM mode

TPM0->SC |= TPM\_SC\_CPWMS\_MASK;

// Initialize count to TPM0->CNT = 0;

TPM0->CNT = 0;

// We nominally choose a PWM frequency of 250 Hz

// Calculate and set the appropriate MOD value.

TPM0->MOD = 125;

// Configure the MSB:MSA and ELSB:ELSA bits for

// all the relevant channels

TPM0->CONTROLS[4].CnSC &= ~TPM\_CnSC\_MSA\_MASK | TPM\_CnSC\_ELSB\_MASK; // Red

TPM0->CONTROLS[5].CnSC &= ~TPM\_CnSC\_MSA\_MASK | TPM\_CnSC\_ELSB\_MASK; // Green

TPM0->CONTROLS[2].CnSC &= ~TPM\_CnSC\_MSA\_MASK | TPM\_CnSC\_ELSB\_MASK; // Blue

TPM0->CONTROLS[4].CnSC |= (TPM\_CnSC\_MSB(1) | TPM\_CnSC\_ELSA(1)); // Red

TPM0->CONTROLS[5].CnSC |= (TPM\_CnSC\_MSB(1) | TPM\_CnSC\_ELSA(1)); // Green

TPM0->CONTROLS[2].CnSC |= (TPM\_CnSC\_MSB(1) | TPM\_CnSC\_ELSA(1)); // Blue

}

**Question 4.** (1 mark) startPWM:

**void** **startPWM**() {

// Fill in code to start the PWM

TPM0->SC |= TPM\_SC\_CMOD(0b01);

}

**Question 5.** (1 mark) stopPWM:

**void** **stopPWM**() {

// Fill in code to stop the PWM

TPM0->SC &= ~(TPM\_SC\_CMOD\_MASK);

}

**Question 6.** (3 marks) setPWM:

**void** **setPWM**(**int** LED, **int** percent) {

**int** value = (**int**)((percent / 100.0) \* (**double**) TPM0->MOD);

// PRINTF("value = %d\r\n", value);

**switch**(LED){

**case** *RED*:

// Statement to set the RED LED value

TPM0->CONTROLS[4].CnV = value;

**break**;

**case** *GREEN*:

// Statement to set the GREEN LED value

TPM0->CONTROLS[5].CnV = value;

**break**;

**case** *BLUE*:

// Statement to set the BLUE LED value

TPM0->CONTROLS[2].CnV = value;

**break**;

**default**:

PRINTF("Invalid LED.\r\n");

}

}

**Question 7.** (2 marks)

Modified from TPM0->MOD = 125; to the code below

TPM0->MOD = 1563;

**Difference:** At 250Hz, the LED flickers faster than what the human eye can perceive, thus, we see a smooth transition between the changes in colour. However, at 20Hz, the LED flickers slow enough for humans to perceive, thus we see the LEDs flicker while changing in colours.

**Report Total:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / 17

**Demo:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / 3

**Total:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / 20