HW#8

Mrinmoy Sarkar

ECEN-621

**Q1: Code:**

|  |
| --- |
| **#include** "msp.h"  **#define** MAX\_ADC\_VALUE 16384  **int** period\_of\_blue\_led = 16384; // fixed period for blue led  **volatile** uint16\_t JOYresults[2]; // store the ADC value of the joystick  **float** duty\_cycle = 0.5; // store the instantaneous duty cycle of the two led  **int** **main**(**void**)  {  WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // Stop WDT  // Configure GPIO for PWM output  P2->DIR |= BIT6 | BIT7; // red led and buzzer  P2->SEL0 |= BIT6 | BIT7;  P5->DIR |= BIT6; // blue led  P5->SEL0 |= BIT6;  // configure timer for pwm operation  TIMER\_A0->CCR[0] = 1000 - 1; // PWM Period  TIMER\_A0->CCTL[3] = TIMER\_A\_CCTLN\_OUTMOD\_7; // CCR3 reset/set  TIMER\_A0->CCR[3] = 750; // CCR3 PWM duty cycle  TIMER\_A0->CCTL[4] = TIMER\_A\_CCTLN\_OUTMOD\_7; // CCR4 reset/set  TIMER\_A0->CCR[4] = 500; // CCR4 PWM duty cycle  TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK  TIMER\_A\_CTL\_MC\_\_UP | // Up mode  TIMER\_A\_CTL\_CLR; // Clear TAR  TIMER\_A2->CCR[0] = period\_of\_blue\_led - 1; // PWM Period  TIMER\_A2->CCTL[1] = TIMER\_A\_CCTLN\_OUTMOD\_7; // CCR1 reset/set  TIMER\_A2->CCR[1] = 750; // CCR1 PWM duty cycle  TIMER\_A2->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK  TIMER\_A\_CTL\_MC\_\_UP | // Up mode  TIMER\_A\_CTL\_CLR; // Clear TAR  // Configure GPIO for ADC  P4->SEL1 |= BIT4; // Enable A/D channel A9  P4->SEL0 |= BIT4;  P6->SEL1 |= BIT0; // Enable A/D channel A15  P6->SEL0 |= BIT0;  // Turn on ADC14, extend sampling time to avoid overflow of results  ADC14->CTL0 = ADC14\_CTL0\_ON | ADC14\_CTL0\_MSC | ADC14\_CTL0\_SHT0\_\_192 | ADC14\_CTL0\_SHP | ADC14\_CTL0\_CONSEQ\_3;  ADC14->MCTL[0] = ADC14\_MCTLN\_INCH\_9; // ref+=AVcc, channel = A9  ADC14->MCTL[1] = ADC14\_MCTLN\_INCH\_15 | ADC14\_MCTLN\_EOS; // ref+=AVcc, channel = A15, end seq.  ADC14->IER0 = ADC14\_IER0\_IE1; // ADC interrupt enable for ADC14->MEM[1]  SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // sleep on exit  // Start conversion-software trigger  ADC14->CTL0 |= ADC14\_CTL0\_ENC | ADC14\_CTL0\_SC;  NVIC->ISER[0] = 1 << ((*ADC14\_IRQn*) & 31); // Enable ADC interrupt in NVIC module  // Enable global interrupt  \_\_enable\_irq();  // Enter LPM0  \_\_sleep();  }  // ADC14 interrupt service routine  **void** **ADC14\_IRQHandler**(**void**)  {  **if** (ADC14->IFGR0 & ADC14\_IFGR0\_IFG1)  {  JOYresults[0] = ADC14->MEM[0]; // Move A9 results, IFG is cleared. vertical joystick  JOYresults[1] = ADC14->MEM[1]; // Move A15 results, IFG is cleared. horizontal joystick  duty\_cycle = (**float**)JOYresults[1] / MAX\_ADC\_VALUE; // calculate the duty cycle  TIMER\_A0->CCR[0] = JOYresults[0] - 1; // reset the PWM Period for buzzer  TIMER\_A0->CCR[3] = duty\_cycle \* JOYresults[0]; // reset the duty cycle of red led  TIMER\_A0->CCR[4] = 0.5\*JOYresults[0]; // set the buzzer duty cycle to 50%  TIMER\_A2->CCR[1] = duty\_cycle \* period\_of\_blue\_led; // reset the duty cycle of blue led  \_\_delay\_cycles(2000); // small delay  }  } |

**Q1: UML diagram:**

**![A close up of a piece of paper

Description automatically generated]()**