HW#7

Mrinmoy Sarkar

ECEN-621

**Q1: Code:**

|  |
| --- |
| **#include** "msp.h"  **int** **main**(**void**)  {  // Hold the watchdog  WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;  P2->DIR |= BIT4; //Make P2.6 an output pin Green LED on MKII  P2->DIR |= BIT6; //Make P5.6 an output pin Red LED on MKII  P2->OUT &= ~BIT4; //turn green off initially  P2->OUT &= ~BIT6; //Turn red LED off initially  P3->DIR &= ~BIT5; //Make P3.5 an input pin  P3->REN = BIT5; // Enable pull resistor on P3.5  P3->OUT = BIT5; // Connect the pull resistor to Vcc (i.e. make it a pull-up)  P3->IES = BIT5; // Interrupt on high-to-low transition (Falling edged trigger on P3.5)  P3->IFG = 0; // Clear all P3 interrupt flags  P3->IE = BIT5; // Enable interrupt for P3.5 (Pin interrupt enable)  // Enable Port 3 interrupt on the NVIC (Port Interrupt Enable & Prioriterize)  NVIC->ISER[1] = BIT5;///This Sets the P3 interrupt enable bit in the ISER1 register  \_\_enable\_interrupts(); //Global Enable prioritized interrupts  **while**(1){  // Go to LPM4 (Low Power Mode #4)  \_\_sleep();  }  }  /\* Port3 ISR \*/  **void** **PORT3\_IRQHandler**(**void**)  {  // Toggling the output on the Yellow light on MKII IF P3.5 is what caused the interrupt  **if**(P3->IFG & BIT5)  {  P2->OUT ^= BIT4;  P2->OUT ^= BIT6;  }  // Delay for switch debounce  \_\_delay\_cycles(1500000); // 1/2 second delay  P3->IFG &= ~BIT5; //clear the flag so that the interrupt will not immediately happen again  } |

**Q1: UML diagram:**

**![A close up of text on a white background

Description automatically generated]()**

**Q2:Code**

|  |
| --- |
| **#include** "msp.h"  **int** **main**(**void**)  {  // Hold the watchdog  WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;  P2->DIR |= BIT1; //Make P2.6 an output pin Green LED on Launchpad  P2->DIR |= BIT2; //Make P5.6 an output pin Blue LED on Launchpad  P2->OUT &= ~BIT1; //turn green off initially  P2->OUT &= ~BIT2; //Turn blue LED off initially  P1->DIR &= ~BIT1; //Make P1.1 an input pin  P1->REN = BIT1; // Enable pull resistor on P1.1  P1->OUT = BIT1; // Connect the pull resistor to Vcc (i.e. make it a pull-up)  P1->IES = BIT1; // Interrupt on high-to-low transition (Falling edged trigger on P1.1)  P1->IFG = 0; // Clear all P1 interrupt flags  P1->IE = BIT1; // Enable interrupt for P1.1 (Pin interrupt enable)  // Enable Port 1 interrupt on the NVIC (Port Interrupt Enable & Prioriterize)  NVIC->ISER[1] = BIT3;///This Sets the P1 interrupt enable bit in the ISER1 register  \_\_enable\_interrupts(); //Global Enable prioritized interrupts  **while**(1){  // Go to LPM4 (Low Power Mode #4)  \_\_sleep();  }  }  /\* Port1 ISR \*/  **void** **PORT1\_IRQHandler**(**void**)  {  // Toggling the output on the Turquoise light on Launchpad IF P1.1 is what caused the interrupt  **if**(P1->IFG & BIT1)  {  P2->OUT ^= BIT1;  P2->OUT ^= BIT2;  }  // Delay for switch debounce  \_\_delay\_cycles(1500000); // 1/2 second delay  P1->IFG &= ~BIT1; //clear the flag so that the interrupt will not immediately happen again  } |

**Q2: UML diagram**

**![A close up of text on a white background

Description automatically generated]()**

**Q3:Code(**MKII has higher priority**)**

|  |
| --- |
| **#include** "msp.h"  **int** **main**(**void**)  {  // Hold the watchdog  WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;  P2->DIR |= BIT4; //Make P2.6 an output pin Green LED on MKII  P2->DIR |= BIT6; //Make P5.6 an output pin Red LED on MKII  P2->OUT &= ~BIT4; //turn green off initially  P2->OUT &= ~BIT6; //Turn red LED off initially  P3->DIR &= ~BIT5; //Make P3.5 an input pin  P3->REN = BIT5; // Enable pull resistor on P3.5  P3->OUT = BIT5; // Connect the pull resistor to Vcc (i.e. make it a pull-up)  P3->IES = BIT5; // Interrupt on high-to-low transition (Falling edged trigger on P3.5)  P3->IFG = 0; // Clear all P3 interrupt flags  P3->IE = BIT5; // Enable interrupt for P3.5 (Pin interrupt enable)  // Enable Port 3 interrupt on the NVIC (Port Interrupt Enable & Prioriterize)  NVIC->ISER[1] = BIT5;///This Sets the P3 interrupt enable bit in the ISER1 register  P2->DIR |= BIT1; //Make P2.6 an output pin Green LED on Launchpad  P2->DIR |= BIT2; //Make P5.6 an output pin Blue LED on Launchpad  P2->OUT &= ~BIT1; //turn green off initially  P2->OUT &= ~BIT2; //Turn blue LED off initially  P1->DIR &= ~BIT1; //Make P1.1 an input pin  P1->REN = BIT1; // Enable pull resistor on P1.1  P1->OUT = BIT1; // Connect the pull resistor to Vcc (i.e. make it a pull-up)  P1->IES = BIT1; // Interrupt on high-to-low transition (Falling edged trigger on P1.1)  P1->IFG = 0; // Clear all P1 interrupt flags  P1->IE = BIT1; // Enable interrupt for P1.1 (Pin interrupt enable)  // Enable Port 1 interrupt on the NVIC (Port Interrupt Enable & Prioriterize)  NVIC->ISER[1] = BIT3;///This Sets the P1 interrupt enable bit in the ISER1 register  NVIC->IP[*PORT1\_IRQn*] = 64;//Set the interrupt priority to 64 for port P1 interrupt  NVIC->IP[*PORT3\_IRQn*] = 0;//Set the interrupt priority to 0 for port P3 interrupt  \_\_enable\_interrupts(); //Global Enable prioritized interrupts  **while**(1){  // Go to LPM4 (Low Power Mode #4)  \_\_sleep();  }  }  /\* Port1 ISR \*/  **void** **PORT1\_IRQHandler**(**void**)  {  // Blinking the output on the Turquoise light on Launchpad IF P1.1 is what caused the interrupt  **if**(P1->IFG & BIT1)  {  P2->OUT ^= BIT1;  P2->OUT ^= BIT2;  \_\_delay\_cycles(6000000); // 2 second delay  P2->OUT ^= BIT1;  P2->OUT ^= BIT2;  \_\_delay\_cycles(6000000); // 2 second delay  }  P1->IFG &= ~BIT1; //clear the flag so that the interrupt will not immediately happen again  }  /\* Port3 ISR \*/  **void** **PORT3\_IRQHandler**(**void**)  {  // Blinking the output on the Yellow light on MKII IF P3.5 is what caused the interrupt  **if**(P3->IFG & BIT5)  {  P2->OUT ^= BIT4;  P2->OUT ^= BIT6;  \_\_delay\_cycles(6000000); // 2 second delay  P2->OUT ^= BIT4;  P2->OUT ^= BIT6;  \_\_delay\_cycles(6000000); // 2 second delay  }  P3->IFG &= ~BIT5; //clear the flag so that the interrupt will not immediately happen again  } |

**Explanation using visually inspecting the output:**

In the above code PORT3 has higher priority than PORT1. The expected output is when s2 of MKII will be pressed the yellow led in MKII will be blinked and when s1 of Launchpad will be pressed the turquoise led of Launchpad will be blinked. Now, if s1 is pressed first and then s2 is pressed, we will see both turquoise and yellow light will be turned on since there is a long delay between turning on and off of the leds and the yellow led turns off first then the turquoise led. Here, we can see that by pressing s1 the PORT1 ISR will be running and in the meantime s2 will generate another interrupt since s2 has more priority, yellow led will be turned on. Therefore, with this sequence of operation, we can understand that PORT3 has more priority than PORT1. However, if we press s2 first then s1 we will never see both led turn on simultaneously. But we will see the yellow led turns on first then it turns off and then the turquoise led turns on and off.

**UML diagram:**

**![A picture containing text, map

Description automatically generated]()**

**Q3:Code(**Launchpad has higher priority**)**

|  |
| --- |
| **#include** "msp.h"  **int** **main**(**void**)  {  // Hold the watchdog  WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;  P2->DIR |= BIT4; //Make P2.6 an output pin Green LED on MKII  P2->DIR |= BIT6; //Make P5.6 an output pin Red LED on MKII  P2->OUT &= ~BIT4; //turn green off initially  P2->OUT &= ~BIT6; //Turn red LED off initially  P3->DIR &= ~BIT5; //Make P3.5 an input pin  P3->REN = BIT5; // Enable pull resistor on P3.5  P3->OUT = BIT5; // Connect the pull resistor to Vcc (i.e. make it a pull-up)  P3->IES = BIT5; // Interrupt on high-to-low transition (Falling edged trigger on P3.5)  P3->IFG = 0; // Clear all P3 interrupt flags  P3->IE = BIT5; // Enable interrupt for P3.5 (Pin interrupt enable)  // Enable Port 3 interrupt on the NVIC (Port Interrupt Enable & Prioriterize)  NVIC->ISER[1] = BIT5;///This Sets the P3 interrupt enable bit in the ISER1 register  P2->DIR |= BIT1; //Make P2.6 an output pin Green LED on Launchpad  P2->DIR |= BIT2; //Make P5.6 an output pin Blue LED on Launchpad  P2->OUT &= ~BIT1; //turn green off initially  P2->OUT &= ~BIT2; //Turn blue LED off initially  P1->DIR &= ~BIT1; //Make P1.1 an input pin  P1->REN = BIT1; // Enable pull resistor on P1.1  P1->OUT = BIT1; // Connect the pull resistor to Vcc (i.e. make it a pull-up)  P1->IES = BIT1; // Interrupt on high-to-low transition (Falling edged trigger on P1.1)  P1->IFG = 0; // Clear all P1 interrupt flags  P1->IE = BIT1; // Enable interrupt for P1.1 (Pin interrupt enable)  // Enable Port 1 interrupt on the NVIC (Port Interrupt Enable & Prioriterize)  NVIC->ISER[1] = BIT3;///This Sets the P1 interrupt enable bit in the ISER1 register  NVIC->IP[*PORT1\_IRQn*] = 0;//Set the interrupt priority to 0 for port P1 interrupt  NVIC->IP[*PORT3\_IRQn*] = 64;//Set the interrupt priority to 64 for port P3 interrupt  \_\_enable\_interrupts(); //Global Enable prioritized interrupts  **while**(1){  // Go to LPM4 (Low Power Mode #4)  \_\_sleep();  }  }  /\* Port1 ISR \*/  **void** **PORT1\_IRQHandler**(**void**)  {  // Blinking the output on the Turquoise light on Launchpad IF P1.1 is what caused the interrupt  **if**(P1->IFG & BIT1)  {  P2->OUT ^= BIT1;  P2->OUT ^= BIT2;  \_\_delay\_cycles(6000000); // 2 second delay  P2->OUT ^= BIT1;  P2->OUT ^= BIT2;  \_\_delay\_cycles(6000000); // 2 second delay  }  P1->IFG &= ~BIT1; //clear the flag so that the interrupt will not immediately happen again  }  /\* Port3 ISR \*/  **void** **PORT3\_IRQHandler**(**void**)  {  // Toggling the output on the Yellow light on MKII IF P3.5 is what caused the interrupt  **if**(P3->IFG & BIT5)  {  P2->OUT ^= BIT4;  P2->OUT ^= BIT6;  \_\_delay\_cycles(6000000); // 2 second delay  P2->OUT ^= BIT4;  P2->OUT ^= BIT6;  \_\_delay\_cycles(6000000); // 2 second delay  }  P3->IFG &= ~BIT5; //clear the flag so that the interrupt will not immediately happen again  } |

**Explanation using visually inspecting the output:**

In the above code PORT1 has higher priority than PORT3. The expected output is when s2 of MKII will be pressed the yellow led in MKII will be blinked and when s1 of Launchpad will be pressed the turquoise led of Launchpad will be blinked. Now, if s2 is pressed first and then s1 is pressed, we will see both turquoise and yellow light will be turned on since there is a long delay between turning on and off of the leds and the turquoise led turns off first then the yellow led. Here, we can see that by pressing s2 the PORT3 ISR will be running and in the meantime s1 will generate another interrupt since s1 has more priority, turquoise led will be turned on. Therefore, with this sequence of operation, we can understand that PORT1 has more priority than PORT3. However, if we press s1 first then s2 we will never see both led turn on simultaneously. But we will see the turquoise led turns on first then it turns off and then the yellow led turns on and off.

**UML diagram:**

**![A picture containing text, map

Description automatically generated]()**