EECE.3170: Microprocessor Systems Design I

Summer 2016

Lecture 13: Exam Practice Problems

June 20, 2016

1. Complete this ISR so that if a timer interrupt has occurred, the LEDs will be updated to show the next value in the pattern stored in the array `st[]`, going back to the first value (`0b0001`) after showing the eighth (`0b1001`). If a switch interrupt has occurred, clear all LEDs and reset to the initial state. Assume the LEDs are wired to Port C, as on the development board used in HW 6, and that "SWITCH" and "DOWN" are appropriately defined.

Assume the use of the following global variables—`st[]` holds the list of values to be displayed on the LEDs, while `i` is the current index into that array. Assume `i` initially holds the value 0:

```
unsigned char st[8] = {0b0001, 0b0010, 0b0100, 0b0101,
                       0b1010, 0b0100, 0b1000, 0b1001};
unsigned char i;

void interrupt ISR(void) {

    if (IOCAF) {                          // SW1 was pressed

        _____________________________     // Clear flag in software
        __delay_ms(5);                    // Delay for debouncing
        if (SWITCH == DOWN) {             // If switch still pressed

          _____________________________   //   clear LEDs

          i = 0;                          //   and reset i
        }
    }

    if (INTCONbits.T0IF) {                // Timer 0 interrupt

        _____________________________     // Clear flag in software

        _____________________________     // Update LEDs to show
                                          //   current st[] value
        _____________________________     // Increment i

        if (_____________________________) // If i exceeds max index
            i = 0;                        //   reset i
    }
}
```

2. This function performs an analog to digital conversion and uses the least significant bits of the result, which are stored in ADRESL, to determine the program operation as follows:
   - If the lowest two bits are 01, toggle the lowest LED, which is wired to the least significant bit (bit 0) of Port C.
     - That bit can be accessed either through the `PORTC` or `LATC` register; to access bit 0, use `PORTCbits.PORTC0` or `LATCbits.LATC0`
   - If the lowest two bits are 10, toggle the second LED, which is wired to bit 1 of Port C.
   - If the lowest two bits are 11, turn both the first and second LEDs on.

Assume the ADC is configured to produce a right-justified result, so the lowest bits of ADRESL are the least significant bits of the conversion result.

```
void read_adc(void) {
    unsigned char lobits;           // Variable to hold lowest 2
                                    //   bits of ADC result
    __delay_us(5);                  // Wait for ADC cap to settle

    _____________________________   // Start conversion
    while (GO) continue;            // Wait until conversion done

    _____________________________   // lobits = lowest two bits
                                    //   of ADC result
    if (lobits == 0b01) {           // In this case, toggle
                                    //   lowest LED (FILL IN
                                    //   SPACE TO LEFT WITH
                                    //   APPROPRIATE CODE, WHICH
                                    //   MAY USE MULTIPLE LINES)


    }
    else if (lobits == 0b10) {      // In this case, toggle
                                    //   second LED (FILL IN
                                    //   SPACE TO LEFT WITH
                                    //   APPROPRIATE CODE, WHICH
                                    //   MAY USE MULTIPLE LINES)


    }
    else if (lobits == 0b11) {      // In this case, turn first &
                                    //   second LEDs on (FILL IN
                                    //   SPACE TO LEFT WITH
                                    //   APPROPRIATE CODE, WHICH
                                    //   MAY USE MULTIPLE LINES)


    }
}
```