CSS332: Microcontrollers and Applications
Final Mock Exam

curated by The Peanuts

Name . . . . . Nonprawich I. . . . . ID . . . . . 6622772422 . . . . . Section . . . . . 4 . . . . . No . . . . . ت . . . . . .

Conditions: Semi-Closed Book (A4 Both Sides)

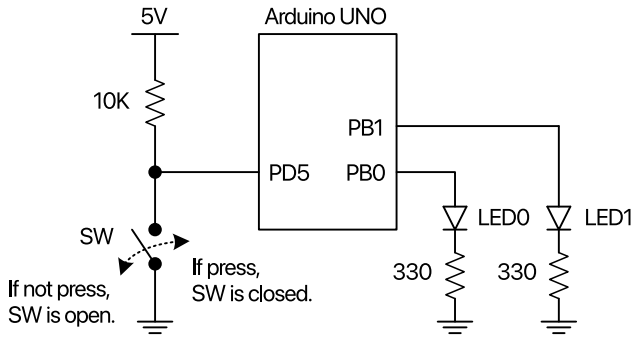
Directions:

1. This exam has 15 pages (including this page).
2. Calculators (Casio 991 Series) are allowed.
3. Write your name clearly at the top of each page.
4. You may cry silently. Loud sobs will be graded as participation.
5. If you finish early, triple-check your work.
6. Do not cheat.
7. Good luck! May the bugs forever stay in your code and not in your answers.

Rock!

~~*For solution, click here.*~~

Problem 1



You need to write an Assembly program to implement the following tasks:

- Task 1: The microcontroller monitors the status of the switch SW: if SW is pressed, LED0 turns on; if not pressed, LED0 turns off.
- Task 2: The microcontroller keeps turning LED1 on and off every 0.5 seconds using the Timer0 overflow interrupt.

The Timer0 should be set up in normal mode with a pre-scaling factor of 1024. Each Timer0 overflow will generate an interrupt every 4000 μs.

Part A: Timer Calculations

a) What value should be loaded into TCNT0 to generate an overflow interrupt every 4000 μs with a 16 MHz clock and a prescaler of 1024?

$$4000 = \left[\left(\left\lfloor\frac{(255 - A + 1) \times 1024}{3}\right\rfloor \times 3\right) + 2 + 14\right] \times 0.0625$$

$A = 193.51$

$A \approx 194 = 0xC2$ $\quad \therefore TCNT0 = 0xC2$

b) How many Timer0 overflow interrupts are required to achieve the 0.5-second toggle for LED1?

$$\#loops = \frac{0.5}{4000 \times 10^{-6}} = 125$$

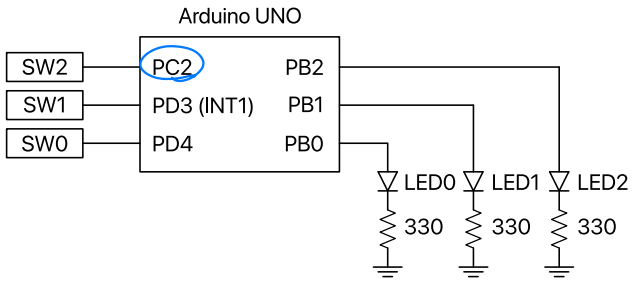
Part B: Complete the Assembly Program

Complete the following Assembly program by filling in the missing parts to implement both tasks.

```
.ORG 0x0      ; Set the origin for the reset vector
    JMP ________  ; Jump to the main program

.ORG ________
    JMP _______  ; Jump to the Timer0 Overflow Interrupt Service
        Routine

.ORG 0x100    ; Start of the main program
MAIN:
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16
    LDI R16, LOW(RAMEND)
    OUT SPL, R16

    CALL PIN_SETUP       ; Call subroutine to set up pins

    LDI R16, _____    ; Enable Timer0 overflow interrupt
    STS TIMSK0, R16      ; Store in Timer Interrupt Mask Register
    _____              ; Enable global interrupts

    CALL TIMER0_SETUP    ; Call subroutine to set up Timer0
    LDI R21, _____    ; Load counter value for 0.5-second
       toggling

LOOP:
    _____ PIND, 5     ; Check if switch is pressed
    RJMP L_OFF           ; If switch is not pressed, jump to turn
        LED0 off
    RJMP L_ON            ; Otherwise, turn LED0 on

L_OFF:
    _____ PORTB, 0    ; Turn LED0 off
    RJMP LOOP            ; Loop back

L_ON:
    _____ PORTB, 0    ; Turn LED0 on
```

```
34      RJMP LOOP               ; Loop back
35
36  ; Subroutine to configure I/O pins
37  PIN_SETUP:
38      SBI DDRB, 0             ; Set PB0 as output (LED0)
39      CBI PORTB, 0            ; Ensure LED0 is off initially
40      SBI DDRB, 1             ; Set PB1 as output (LED1)
41      CBI PORTB, 1            ; Ensure LED1 is off initially
42      CBI DDRD, 5             ; Set PD5 as input (Switch)
43      SBI PORTD, 5            ; Enable pull-up resistor on PD5
44      RET                     ; Return from subroutine
45
46  ; Subroutine to configure Timer0
47  TIMER0_SETUP:
48      LDI R20, _0xC2____      ; Load initial value for Timer0
49      OUT TCNT0, R20          ; Set Timer0 initial value
50      LDI R20, _0x00____      ; Set normal mode (TCCR0A = ?)
51      OUT TCCR0A, R20         ; Store in Timer Control Register A
52      LDI R20, _0x05____      ; Set pre-scaler to 1024 and start
            Timer0 (TCCR0B = ?)
53      OUT TCCR0B, R20         ; Store in Timer Control Register B
54      RET                     ; Return from subroutine
55
56  .ORG 0x200 ; Start of Timer0 Overflow Interrupt Service Routine
57  T0_OV_ISR:
58      DEC R21                 ; Decrement overflow counter
59      BRNE HERE               ; If not zero, skip the toggling part
60      LDI R21, 125_______     ; Reset counter for 0.5-second cycle
61      IN R17, PORTB           ; Read current PORTB state
62      LDI R18, (1<<1)_____    ; Load mask for PB1 (LED1)
63      EOR_____ R17, R18       ; Toggle PB1 state
64      OUT PORTB, R17          ; Output new state to PORTB
65
66  HERE:
67      LDI R18, 0xC2_____      ; Reload initial Timer0 value
68      OUT TCNT0, R18          ; Store in Timer0 counter register
69      RETI_____               ; Return from interrupt
```

Problem 2



You need to write a C program to implement the following tasks:

Main program

- Task 1: The microcontroller monitors the status of switch SW0: if SW0 is pressed, LED0 turns on; if not pressed, LED0 turns off.
- Task 2: The microcontroller monitors the status of switch SW1 (connected to INT1): if SW1 is pressed, LED1 toggles (on → off or off → on). The INT1 should be configured for rising-edge trigger.
- Task 3: The microcontroller monitors the status of switch SW2 (connected to PC2): LED2 toggles only after SW2 has been pressed and released 4 times (using the pin change interrupt).

Part A: Interrupt Configuration

a) For INT1 (external interrupt) with rising-edge trigger, what values should be configured for the following registers?

EIMSK = 0x10

EICRA = 0x0C

b) For the pin change interrupt on PC2, what values should be configured for the following registers?

PCICR = 0x02

PCMSK0 = 0x00

PCMSK1 = 0x04

PCMSK2 = 0x00

c) To make LED2 toggle only after 4 button presses on SW2, what initial value should variable z be set to? Explain your answer.

8

Part B: Complete the C Program

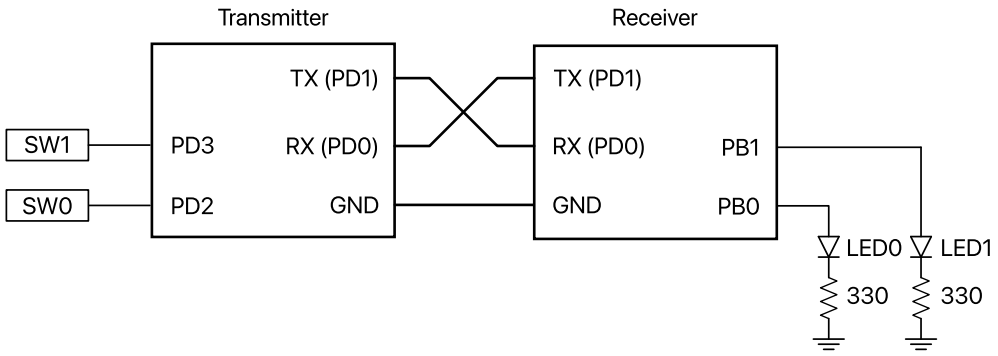
Complete the following C program to implement all three tasks. Fill in the blanks with appropriate code.

```
#include <avr/io.h>
#include <avr/interrupt.h>

unsigned char z = _8_; // Initial value for counting SW2 presses

void PIN_SETUP() {
    // Configure LED outputs
    DDRB |= (1<<0); // Set PB0 as output for LED0
    PORTB &= ~(1<<0); // Set PB0 initial state to low
    DDRB |= (1<<1); // Set PB1 as output for LED1
    PORTB &= ~(1<<1); // Set PB1 initial state to low
    DDRB |= (1<<2); // Set PB2 as output for LED2
    PORTB &= ~(1<<2); // Set PB2 initial state to low

    // Configure switch inputs with pull-up resistors
    DDRD &= ~(1<<4); // Set PD4 as input for SW0
    PORTD |= (1<<4); // Enable pull-up for PD4
    DDRD &= ~(1<<3); // Set PD3 as input for SW1 (INT1)
    PORTD |= (1<<3); // Enable pull-up for PD3
    DDRC &= ~(1<<2); // Set PC2 as input for SW2
    PORTC |= (1<<2); // Enable pull-up for PC2
}

int main() {
    PIN_SETUP();

```

```
27      // Configure INT1 for rising edge trigger
28      EIMSK = _0x10_____; // Enable INT1
29      EICRA = _0x0C_____; // Set for rising edge
30
31      // Configure Pin Change Interrupt for PC2
32      PCICR = _0x02_____; // Enable PORTC pin change interrupts
33      PCMSK1 = _0x04_____; // Enable interrupt for PC2
34
35      sei(); // Enable global interrupts
36
37      while (1) {
38          if (_____~PIND & (1<<4)____) { // Check if SW0 is pressed
39              PORTB |= (1<<0); // Turn on LED0
40          } else {
41              PORTB &= ~(1<<0); // Turn off LED0
42          }
43      }
44
45      return 0;
46  }
47
48  // ISR for INT1 (Task 2)
49  ISR(_INT1_vect_____) {
50      PORTB ^= (1<<1)_____;   // Toggle LED1
51  }
52
53  // ISR for Pin Change Interrupt (Task 3)
54  ISR(PCINT1_vect_____) {
55      z--;                    // Decrement counter
56      if (z == 0) {
57          _PORTB ^= (1<<2);     // Toggle LED2
58          z = __8_____;    // Reset counter to initial value
59      }
60  }
```

Problem 3

In this exercise, you will implement serial communication between two Arduino UNO boards. Board 1 will have two switches connected, while Board 2 will have two LEDs connected. When a switch on Board 1 is pressed, it will send a command through UART to toggle the corresponding LED on Board 2.



Part A: Board 1 - Transmitter

Complete the following C program for Board 1, which will monitor the switch states and send commands to Board 2 when a switch is pressed:

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

// Function to initialize UART
void usart_init(void) {
    UCSR0B = (1<<_________);   // Enable USART transmitter
    UCSR0C = (1<<UCSZ01)|(1<<UCSZ00); // Async, 8 bits
    UBRR0L = _________;        // Baud rate = 9600
}

void pin_setup(void) {
    DDRD &= ~(1<<2); // Set PD2 as input (Switch 0)
    PORTD |= (1<<2); // Enable pull-up for Switch 0
    DDRD &= ~(1<<3); // Set PD3 as input (Switch 1)
    PORTD |= (1<<3); // Enable pull-up for Switch 1
}
```

```
18 // Function to send a character over UART
19 void usart_send(unsigned char data) {
20     while (!(________)); // Wait until UDR0 is empty
21     UDR0 = data;         // Send data
22 }
23
24 int main(void) {
25     pin_setup();
26     usart_init();
27
28     while(1) {
29         // Check if Switch 0 is pressed (PD2 is LOW)
30         if (_____________________) {
31             usart_send(__'1'_____); // Send command to toggle LED0
32             _delay_ms(200);
33         }
34
35         // Check if Switch 1 is pressed (PD3 is LOW)
36         if (_____________________) {
37             usart_send(__'2'_____); // Send command to toggle LED1
38             _delay_ms(200);
39         }
40
41         _delay_ms(50); // Small polling delay
42     }
43
44     return 0;
45 }
```

Part B: Board 2 - Receiver

Complete the following C program for Board 2, which will receive commands from Board 1 and toggle the corresponding LEDs:

```
1  void usart_init(void) {
2      UCSR0B = (1<<_________);   // Enable USART receiver
3      UCSR0C = (1<<UCSZ01)|(1<<UCSZ00); // Async, 8 bits
4      UBRR0L = _________;        // Baud rate = 9600
5  }
6
7  void pin_setup(void) {
8      DDRB |= (1<<0); // Set PB0 as output (LED0)
9      PORTB &= ~(1<<0); // Set LED0 initial state to OFF
10     DDRB |= (1<<1); // Set PB1 as output (LED1)
11     PORTB &= ~(1<<1); // Set LED1 initial state to OFF
12 }
13
14 int main(void) {
15     unsigned char received_data;
16     pin_setup();
17     usart_init();
18
19     while(1) {
20         // Wait until data is received
21         while (_____________); // Check if data is available
22
23         // Read the received data
24         received_data = _________;
25
26         // Process received commands
27         if (received_data == '1') {
28             PORTB ^= (1<<0); // Toggle LED0
29         }
30         else if (received_data == '2') {
31             ____________;     // Toggle LED1
32         }
33     }
34     return 0;
35 }
```

Part C: Questions

a) What value should be loaded into UBRR0L to achieve a baud rate of 9600 with a 16 MHz clock? Show your calculation.

$$UBRR = \frac{F_{CPU}}{16 \times Baud} - 1 = \frac{16 \times 10^6}{16 \times 9600} - 1 \approx 103 \ \#$$

b) What command (character) should be sent from Board 1 when Switch 0 is pressed to toggle LED0 on Board 2?

'1'

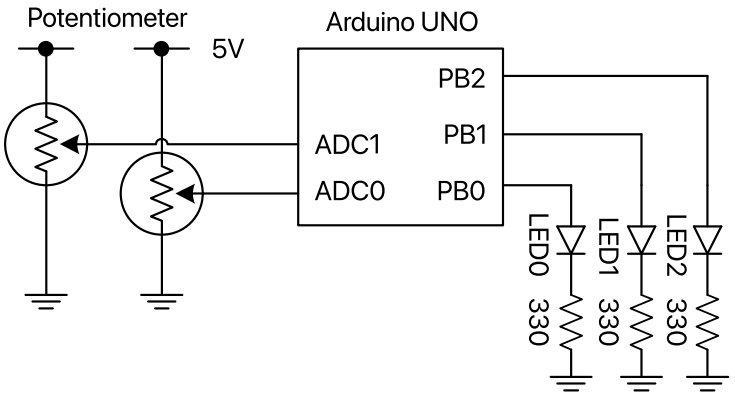
c) What command (character) should be sent from Board 1 when Switch 1 is pressed to toggle LED1 on Board 2?

'2'

d) Which pins must be disconnected before uploading code to the Arduino boards? Explain why this is necessary.

Pin PD0 (RX) and PD1 (TX) must be disconnected before uploading code to Arduino Because these pins are used for serial communication between Arduino and the computer If they remained connected, it can interfere with the upload process, or preventing code from being properly uploaded

Problem 4



You need to write a C program to implement a voltage comparator using two analog inputs on the microcontroller. The circuit consists of two potentiometers connected to ADC0 and ADC1 pins, with three LEDs connected to pins PB0, PB1, and PB2.

Write a complete C program that performs the following tasks:

1. Read the voltage values from two potentiometers connected to ADC0 and ADC1 pins.
2. Compare the two voltage values and control LEDs based on these conditions:
   (a) If voltage at ADC0 > voltage at ADC1, turn on LED1 (PB0) and turn off other LEDs.
   (b) If voltage at ADC0 < voltage at ADC1, turn on LED2 (PB1) and turn off other LEDs.
   (c) If voltage at ADC0 = voltage at ADC1, turn on LED3 (PB2) and turn off other LEDs.

Hint: You may need to configure two separate ADC channels in your program. Consider using a function to read the ADC value from a specific channel.

```c
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

// Function to initialize pins
void pin_setup(void) {
    DDRC = 0x00;   // Set PORTC as input (ADC0 and ADC1)
    DDRB = 0x07;   // Set PB0, PB1, PB2 as outputs (LEDs)
    PORTB = 0x00;  // Initialize all LEDs as OFF
}

// Function to initialize ADC0
void ADC0_setup(void) {
    ADCSRA = 0x87;   // Enable ADC, set prescaler to 128
    ADMUX = 0x40;    // Vref = AVCC, select ADC0 channel, right-justified
}

// Function to initialize ADC1
void ADC1_setup(void) {
    ADCSRA = 0x87;   // Enable ADC, set prescaler to 128
    ADMUX = 0x41;    // Vref = AVCC, select ADC1 channel, right-justified
}

// Function to read from ADC
int read_ADC(void) {
    ADCSRA |= (1 << ADSC);
    while ((ADCSRA & (1 << ADIF)) == 0);
    return ADC;
}

int main(void) {
    int adc0_value, adc1_value;

    pin_setup();

    while (1) {
        ADC0_setup();
        adc0_value = read_ADC();

        ADC1_setup();
        adc1_value = read_ADC();

        PORTB = 0x00; // Turn off all LEDs

        if (adc0_value > adc1_value) {
            PORTB = 0x01;
        } else if (adc0_value < adc1_value) {
            PORTB = 0x02;
        } else {
            PORTB = 0x04;
        }

        _delay_ms(100);
    }

    return 0;
}
```

Problem 5

In this problem, you will implement the functionality from Problems 1-4 using Arduino programming language (C++ with Arduino libraries) instead of AVR C or Assembly. Select ONE of the previous problems (1-4) and reimplement it using Arduino code.

Random Question Selection:

Visit **npwitk.com/css332-random** to randomly determine which problem (1-4) you should implement in Arduino.

Problem 1

(เลือกแล้ว)

Problem 2

Problem 3

Problem 4 ⭐⭐ (abbreviation)