

# CSS332: Microcontrollers and Applications

## Final Mock Exam

curated by The Peanuts

Name ..... ID ..... Section ..... No .....

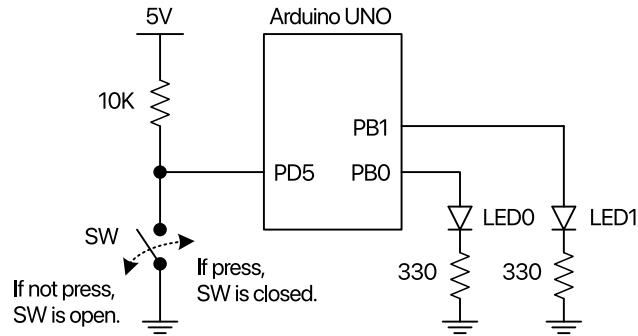
**Conditions:** Semi-Closed Book (A4 Both Sides)

**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.

*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  $4000 \mu\text{s}$ .

### Part A: Timer Calculations

- a) What value should be loaded into TCNT0 to generate an overflow interrupt every  $4000 \mu\text{s}$  with a 16 MHz clock and a prescaler of 1024?
- b) How many Timer0 overflow interrupts are required to achieve the 0.5-second toggle for LED1?

## Part B: Complete the Assembly Program

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

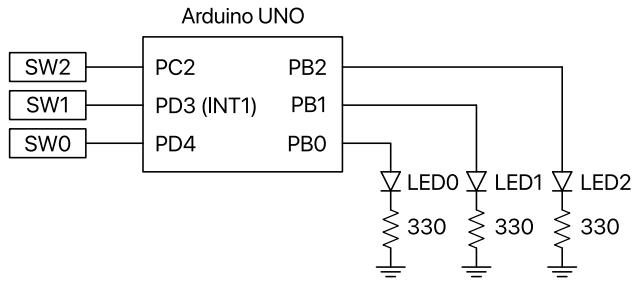
```
1  .ORG 0x00      ; Set the origin for the reset vector
2  JMP _____ ; Jump to the main program
3
4  .ORG _____
5  JMP _____ ; Jump to the Timer0 Overflow Interrupt Service
               Routine
6
7  .ORG 0x100    ; Start of the main program
8  MAIN:
9  LDI R16, HIGH(RAMEND)
10 OUT SPH, R16
11 LDI R16, LOW(RAMEND)
12 OUT SPL, R16
13
14 CALL PIN_SETUP      ; Call subroutine to set up pins
15
16 LDI R16, _____ ; Enable Timer0 overflow interrupt
17 STS TIMSK0, R16 ; Store in Timer Interrupt Mask Register
18 _____ ; Enable global interrupts
19
20 CALL TIMERO_SETUP ; Call subroutine to set up Timer0
21 LDI R21, _____ ; Load counter value for 0.5-second
                  toggling
22
23 LOOP:
24     _____ PIND, 5 ; Check if switch is pressed
25     RJMP L_OFF ; If switch is not pressed, jump to turn
                   LED0 off
26     RJMP L_ON ; Otherwise, turn LED0 on
27
28 L_OFF:
29     _____ PORTB, 0 ; Turn LED0 off
30     RJMP LOOP ; Loop back
31
32 L_ON:
33     _____ 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 (LEDO)
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 TIMERO_SETUP:
48     LDI R20, -----      ; Load initial value for Timer0
49     OUT TCNT0, R20        ; Set Timer0 initial value
50     LDI R20, -----      ; Set normal mode (TCCROA = ?)
51     OUT TCCROA, R20        ; Store in Timer Control Register A
52     LDI R20, -----      ; Set pre-scaler to 1024 and start
53     LDI R20, -----      ; Timer0 (TCCROB = ?)
54     OUT TCCROB, R20        ; Store in Timer Control Register B
55     RET                  ; Return from subroutine
56
57 .ORG 0x200 ; Start of Timer0 Overflow Interrupt Service Routine
58 TO_OV_ISR:
59     DEC R21              ; Decrement overflow counter
60     BRNE HERE             ; If not zero, skip the toggling part
61     LDI R21, -----      ; Reset counter for 0.5-second cycle
62     IN R17, PORTB          ; Read current PORTB state
63     LDI R18, -----      ; Load mask for PB1 (LED1)
64     ----- R17, R18        ; Toggle PB1 state
65     OUT PORTB, R17        ; Output new state to PORTB
66
67 HERE:
68     LDI R18, -----      ; Reload initial Timer0 value
69     OUT TCNT0, R18        ; Store in Timer0 counter register
70     -----                  ; Return from interrupt

```

## Problem 2



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

- 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 = \_\_\_\_\_

EICRA = \_\_\_\_\_

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

PCICR = \_\_\_\_\_

PCMSK0 = \_\_\_\_\_

PCMSK1 = \_\_\_\_\_

PCMSK2 = \_\_\_\_\_

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

## Part B: Complete the C Program

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

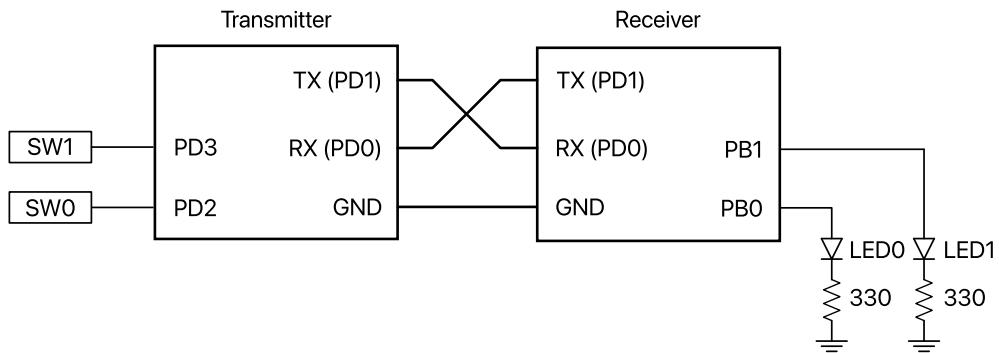
```
1 #include <avr/io.h>
2 #include <avr/interrupt.h>
3
4 unsigned char z = ___; // Initial value for counting SW2 presses
5
6 void PIN_SETUP() {
7     // Configure LED outputs
8     DDRB |= (1<<0); // Set PB0 as output for LED0
9     PORTB &= ~(1<<0); // Set PB0 initial state to low
10    DDRB |= (1<<1); // Set PB1 as output for LED1
11    PORTB &= ~(1<<1); // Set PB1 initial state to low
12    DDRB |= (1<<2); // Set PB2 as output for LED2
13    PORTB &= ~(1<<2); // Set PB2 initial state to low
14
15    // Configure switch inputs with pull-up resistors
16    DDRD &= ~(1<<4); // Set PD4 as input for SW0
17    PORTD |= (1<<4); // Enable pull-up for PD4
18    DDRD &= ~(1<<3); // Set PD3 as input for SW1 (INT1)
19    PORTD |= (1<<3); // Enable pull-up for PD3
20    DDRC &= ~(1<<2); // Set PC2 as input for SW2
21    PORTC |= (1<<2); // Enable pull-up for PC2
22 }
23
24 int main() {
25     PIN_SETUP();
26 }
```

```

27     // Configure INT1 for rising edge trigger
28     EIMSK = -----; // Enable INT1
29     EICRA = -----; // Set for rising edge
30
31     // Configure Pin Change Interrupt for PC2
32     PCICR = -----; // Enable PORTC pin change interrupts
33     ----- = -----; // Enable interrupt for PC2
34
35     sei(); // Enable global interrupts
36
37     while (1) {
38         if (-----) { // 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(-----) {
50     -----; // Toggle LED1
51 }
52
53 // ISR for Pin Change Interrupt (Task 3)
54 ISR(-----) {
55     z--; // Decrement counter
56     if (z == 0) {
57         -----; // Toggle LED2
58         z = -----; // 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:

```
1 #include <avr/io.h>
2 #define F_CPU 16000000UL
3 #include <util/delay.h>
4
5 // Function to initialize UART
6 void usart_init(void) {
7     UCSROB = (1<<______); // Enable USART transmitter
8     UCSROC = (1<<UCSZ01)|(1<<UCSZ00); // Async, 8 bits
9     UBRR0L = _____; // Baud rate = 9600
10 }
11
12 void pin_setup(void) {
13     DDRD &= ~(1<<2); // Set PD2 as input (Switch 0)
14     PORTD |= (1<<2); // Enable pull-up for Switch 0
15     DDRD &= ~(1<<3); // Set PD3 as input (Switch 1)
16     PORTD |= (1<<3); // Enable pull-up for Switch 1
17 }
```

```
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(_-----); // 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(_-----); // 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

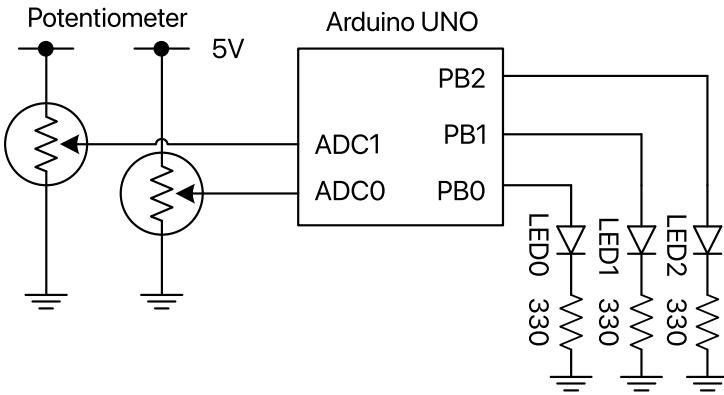
Complete the following C program for Board 2, which will receive commands from Board 1 and toggle the corresponding LEDs:

```
1 void usart_init(void) {
2     UCSROB = (1<<______); // Enable USART receiver
3     UCSROC = (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             _____; // 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.
- b) What command (character) should be sent from Board 1 when Switch 0 is pressed to toggle LED0 on Board 2?
- c) What command (character) should be sent from Board 1 when Switch 1 is pressed to toggle LED1 on Board 2?
- d) Which pins must be disconnected before uploading code to the Arduino boards? Explain why this is necessary.

## 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.

*This page is intentionally left blank for writing code*

## 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](http://npwitk.com/css332-random) to randomly determine which problem (1-4) you should implement in Arduino.

*This page is intentionally left blank for writing code*