TED UNIVERSITY
Department of Computer Engineering

CMPE 451 FINAL EXAM
FALL, 2022-2023

Date: 04/01/2023, Wednesday
Time: 09:30-11:30, Duration: 120 minutes

FULL NAME : Solution

STUDENT ID :

SECTION :

SIGNATURE :

QUESTIONS	POINTS	
Q1	20	
Q2	15	
Q3	15	
Q4	20	
Q5	30	
TOTAL	100	

Please Read Carefully

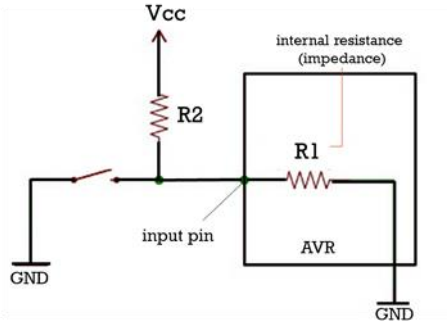
- This is a closed-book exam.
- Talking is forbidden during the exam.
- Use of electronic devices including mobile phones is forbidden during the exam.
- Exchange of items (erasers, pens, pencils, etc.) during the exam is forbidden.
- There should be total of 5 questions. Check your exam paper!
- Show your work clearly in the spaces provided.

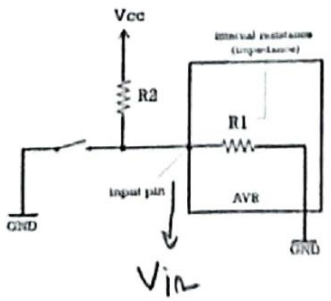
Q1. Digital Input

a- Write C code to configure bits 1 and 4 of PORTB as Input. Enable their internal pull-up resistors and read the value held in PORTB continuously.

```
DDRB = 0x00;
PORTB = 0b00010010;
int tmp;
while (1){
    tmp = PINB;
}
```

b- Determine a suitable pull-up resistor value to provide logic high and logic low to AVR microcontroller according to the state of push button. Assume that Vcc is 5V, internal impedance of AVR is 300KΩ. Show your work and calculations clearly. Writing a value only by depending on literature will not be accepted.





$$V_{in} = \frac{V_{cc}}{R_1+R_2} \cdot R_1$$

For logic high = 1, $V_{in} > \frac{V_{cc}}{2}$

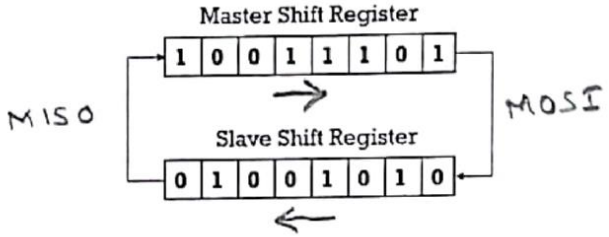
$$\frac{V_{cc}}{2} = \frac{V_{cc} \cdot R_1}{R_1+R_2}$$

$\Rightarrow 2R_1 = R_1 + R_2 \Rightarrow R_1 = R_2 = 300K\Omega$

So R_2 should be smaller than $300K\Omega$ to have $V_{in} > \frac{V_{cc}}{2}$

Q.2 Serial Communication

For the following SPI Master and Slave Shift Registers, show all steps to transmit all the data (loaded at time t) from both sides to each other. Write the names of data lines and show the direction of data transfer.



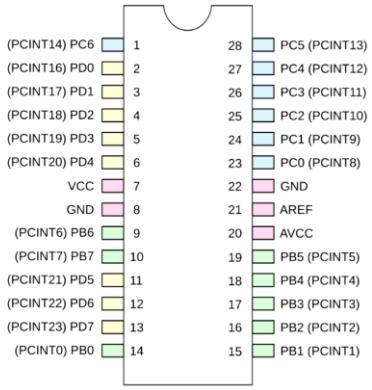
01001110 → 10100111 → 01010011
10010101 → 00101010 → 01010101

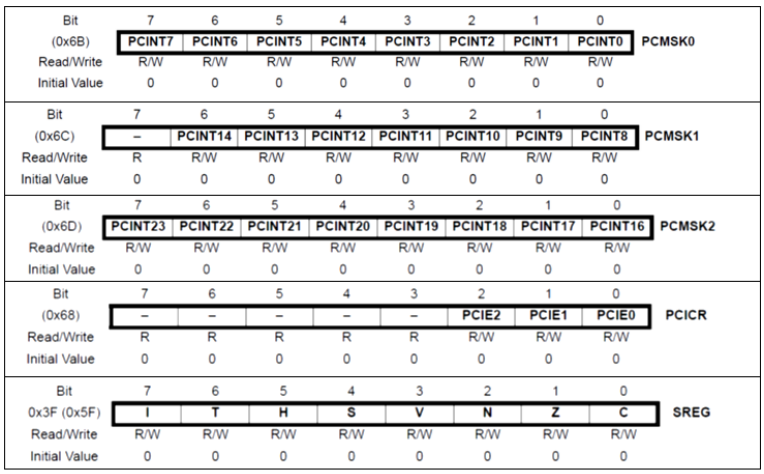
00101001 → 10010100 → 01001010
10101011 → 01010111 → 10101110

10100101 → 01010010 ← read from this side.
01011100 → 10111001

Q.3 Interrupts

Program AVR microcontroller such that LED0 blinks with a delay of 100 ms. LED1 is turned on when push button is pressed. Push button must be handled using ISR for PCINT13. LED0 is connected to bit 2 of PORTB and LED1 is to bit 4 of PORTD.





Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x6C)	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

```
// ------- Preamble -------- //
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define LED_PORT0 PORTB
#define LED0_DDR DDRB
#define LED_PORT1 PORTD
#define LED1_DDR DDRD
#define LED0 2
#define LED1 4
#define BUTTON_PORT PORTC
#define BUTTON_PIN PINC
#define BUTTON 5

ISR(PCINT1_vect)
{
/* Run every time there is a change on button */
if (bit_is_clear(BUTTON_PIN, BUTTON))
        LED_PORT1 |= (1 << LED1);
else
        LED_PORT1 &= ~(1 << LED1);
}
```

```
void initInterrupt0(void)
{
PCMSK1 | = (1<<PCINT13);   //enable PCINT13
PCICR |= (1<<PCIE1);          // PortC → Bank 1
sei(); /* set (global) interrupt enable bit */
}

int main(void)
{
// -------- Inits --------- //
LED0_DDR = 0xff; /* all LEDs active */
LED1_DDR = 0xff;
BUTTON_PORT |= (1 << BUTTON); /* pullup */
initInterrupt0();
// ------ Event loop ------ //
while (1)
        {
        _delay_ms(100);
        LED_PORT0 ^= (1 << LED0);
        } /* End event loop */
return 0; /* This line is never reached */
}
```

Q.4 Timer/Counter

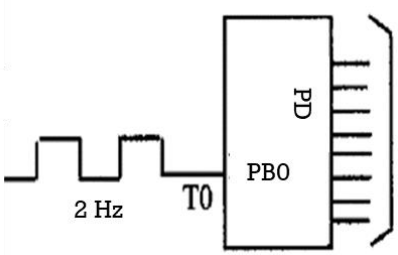
a- Write a C program to generate 800µs delay, by using Timer0, CTC mode, with prescaler = 64. Assume that XTAL = 4MHz. Information you may need is given below.

T = 1/XTAL = 1/(4x10^6)= 0.25x10^{-6}s = 0.25µs, since prescaler is 64;
64 x 0.25µs =16 µs; for 800µs delay; we need 800/16 = 50 clocks. Therefore OCR0 should be set as 50-1 = 49.

```
int main ()
{
        DDRB = 0xFF
        while (1)
        {
                T0Delay();
        }
}
```

```
Void T0Delay()
{
        OCR0 = 49;
        TCNT = 0;
        TCCR0 = 0x0B    // TCRR0 = 00001011
        while((TIFR&0x2)==0)
        TCCR0 = 0;
        TIFR = 0x02;
}
```

b- 2 Hz clock pulse is fed into pin T0 (as shown), write C code to program AVR microcontroller as a counter and display the counter on PORTD.
Note: It should count with the speed of a real clock.




```
#include "avr/io.h"

int main ( )
{
      PORTB = 0x01;          //activate pull-up of PB0

      DDRD = 0xFF;           //PORTD as output

      TCCR0 = 0x06;          //output clock source
      TCNT0 = 0x00;

      while (1)
      {
            do
            {
                  PORTD = TCNT0;
            }while((TIFR&(0x1<<TOV0))==0);//wait for TOV0 to roll ove

            TIFR = 0x1<<TOV0;       //clear TOV0

      }
```

TCCR0 (Timer/Counter Control Register)

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

CS02:00	D2	D1	D0	Timer0 clock selector
	0	0	0	No clock source (Timer/Counter stopped)
	0	0	1	clk (No Prescaling)
	0	1	0	clk / 8
	0	1	1	clk / 64
	1	0	0	clk / 256
	1	0	1	clk / 1024
	1	1	0	External clock source on T0 pin. Clock on falling edge.
	1	1	1	External clock source on T0 pin. Clock on rising edge.

WGM00, WGM01

D6	D3	Timer0 mode selector bits
0	0	Normal
0	1	CTC (Clear Timer on Com
1	0	PWM, phase correct
1	1	Fast PWM

TIFR

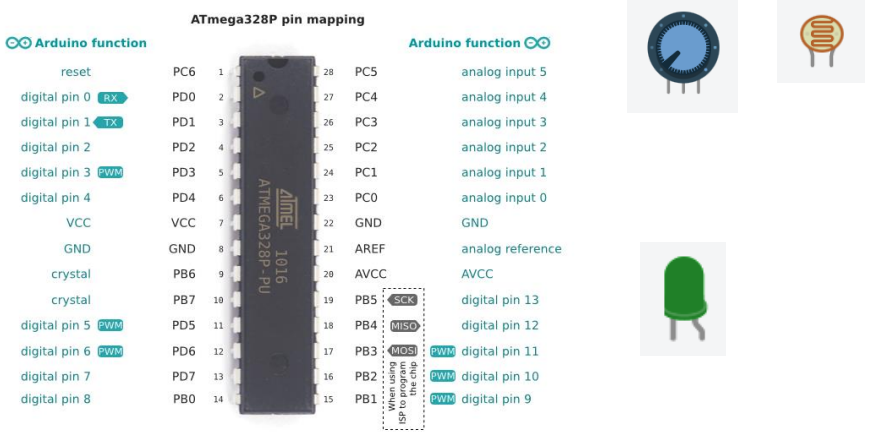
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

Q.5 ADC

Assume that you have 1 light sensor, 1 potentiometer, 1 LED, 1 AVR Atmega328p and jumpers. Design an AVR-based circuit which reads from potentiometer and light sensor. By comparing the voltages read from them, it turns on or turn off the LED. (Assume that AVR is already powered).

I- Connect the components to AVR shown below.

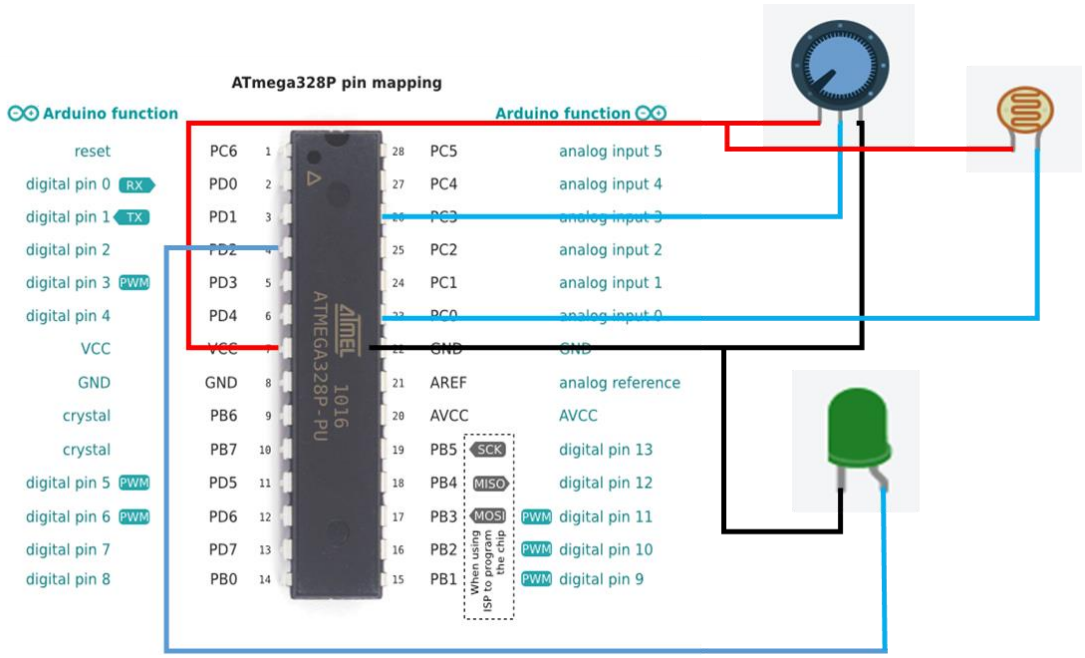
II- Write C program which compares the potentiometer and light sensor voltages, if light sensor's value is above potentiometer's value, LED should be turned on, and otherwise it should be turned off.



Required Registers

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	




```
// ------- Preamble -------- //
#include <avr/io.h>
#include <util/delay.h>
#include "pinDefines.h"
#define LED_DDR DDRD
#define LED_PORT PORTD
#define POT PC3
#define LIGHT_SENSOR PC0

uint16_t readADC(uint8_t channel)
{
ADMUX = (0xf0 & ADMUX) | channel;
ADCSRA |= (1 << ADSC);
loop_until_bit_is_clear(ADCSRA, ADSC);
return (ADC);
}

int main(void)
{
// -------- Inits --------- //
uint16_t lightThreshold;
uint16_t sensorValue;
// Set up ADC
ADMUX |= (1 << REFS0); /* reference voltage on AVCC */
ADCSRA |= (1 << ADPS1) | (1 << ADPS0); /* ADC clock prescaler /8 */
ADCSRA |= (1 << ADEN); /* enable ADC */
```

```
LED_DDR = 0xff;



// ------ Event loop ------ //
while (1)
        {
        lightThreshold = readADC(POT);
        sensorValue = readADC(LIGHT_SENSOR);
        if (sensorValue < lightThreshold)
                LED_PORT = 0x04;
        else
                LED_PORT = 0x00;
        } /* End event loop */
return 0; /* This line is never reached */
}
```