Jihan Shende
Net ID: jks550

Midterm 1 Answers

1.

a. #define PORTB(*(volatile unsigned char *)0x25)

The keyword *define* lets us access the variable PORTB as a constant value anywhere in the program. The *unsigned char* used in this code segment means the value is 8 bits long without any signed bit, 0x25 tells that is a memory location and the *volatile* keyword tells the compiler to access the memory location only when it is called upon. In this segment of the code, we are using the *volatile* keyword so that the compiler would have the access to the memory when needed.

b.

```
void GPIOToggle (int MyPort, char MyBitMask){

    switch(MyPort){
    case 0:
          PORTA |= (1 << (unsigned int) MyBitMask);
          break;
    case 1:
          PORTB |= (1 << (unsigned int) MyBitMask);
          break;
    case 2:
          PORTC |= (1 << (unsigned int) MyBitMask);
          break;
    case 3:
          PORTD |= (1 << (unsigned int) MyBitMask);
          break;
    case 4:
          PORTE |= (1 << (unsigned int) MyBitMask);
          break;
    case 5:
          PORTF |= (1 << (unsigned int) MyBitMask);
          break;
   }
}
```

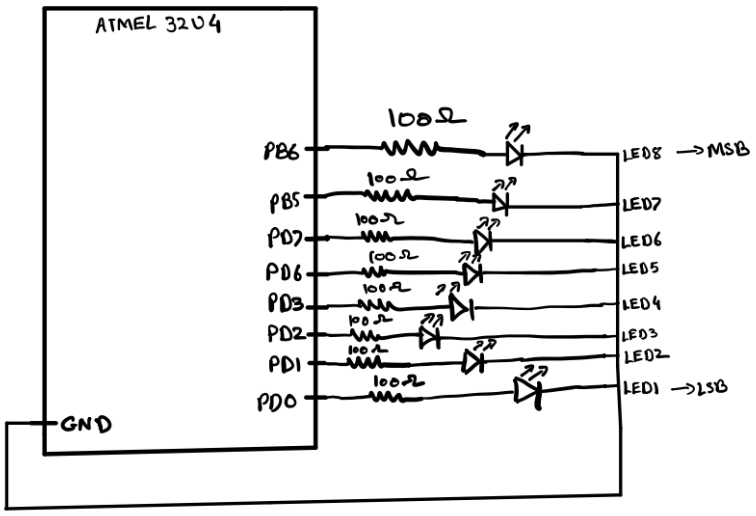
2.

```
#include <Arduino.h>
/*  Using the 8 GPIO pins on the adafruit circuit playground classic
    The selected pins are:
        PD0 (D3)  ---> LED 1 ==> LSB
        PD1 (D2)  ---> LED 2
        PD2 (RX0) ---> LED 3
        PD3 (TX1) ---> LED 4
```

```
        PD6 (D12) ---> LED 5
        PD7 (D6)  ---> LED 6
        PB5 (D9)  ---> LED 7
        PB6 (D10) ---> LED 8 ==> MSB
*/

#define DDR_D *(unsigned char*) 0x2A                    // Directional register D
address
#define DDR_B *(unsigned char*) 0x24                    // Directional register B
address
#define Port_D *(unsigned char*) 0x2B                   // address of PORTD
#define Port_B *(unsigned char*) 0x25                   // address of PORTB


void setup() {
  // put your setup code here, to run once:
    DDR_D |= 0xCF;                                     // (11001111) PD0 - PD7
expect for PD4 and PD5 would turn ON
    DDR_B |= 0x30;                                     // (00110000) PB5 and PB6
Turns ON



}

int UpdateProgressBar(char Percent){
    if ((int)Percent < 0 || (int)Percent > 100)       // Returns 1 for the out of
bounds error
        return 1;
    else {
        int LEDsON = (int)(((int)Percent*8)/100);     // Calculating Number of
LEDs ON based on percent
        if (LEDsON <= 6){
            for(int i = 0; i < LEDsON; i++){
              if (i == 4 || i == 5)
                Port_D |= (1 << (i+2));
               else Port_D |= (1 << i);               // Turning ON the PINS
sequentially based on percent
            }
        } else {
            if (LEDsON == 7){
                Port_D |= 0xCF;                      // Turning all 6 pins ON
                Port_B |= (1 << 5);                  // Turning only 5th pin ON
            } else {
                Port_D |= 0xCF;                      // Turning all 6 pins ON
                Port_B |= 0x30;                      // Turning 2 pins ON
```

```
            }
        }
        return 0;                                   // Returns 0 for a successful
run
    }
}

void loop() {
  // put your main code here, to run repeatedly:
    UpdateProgressBar(90);                          // To turn LEDs ON
}
```

Schematics of the circuit



3.

```
#include <Arduino.h>
/*
    Using adafruit circuit playground classic to code
    For this question I am using 16bit timer (Timer 1 on circuit playground
classic)
    to get the wider range  of frequencies possible
*/

#define PWMFreq 10000                                   // To set the PWM
frequency between 1kHz to 100kHz

void setup() {
```

Jihan Shende
Net ID: jks550

```
  // put your setup code here, to run once:
    DDRB |= (1<<6);                                     // setting port B6 ON(D10
pin on the board)
    TCCR1A = 0b00100011;                                // setting TCCR1A to fast
pwm with channel B set to clear at compare
            // xx   Setting Channel A to 00 for Normal port operations
               // xx    For non-inverting PWM mode
                  // xx     Channel C disconnected
                     // xx      Fast PWM, OCR1A top
    TCCR1B = 0b00011001;                                // setting TCCR1B to fast
pwm with prescaler set of 1 to get better resolution
            // xx   not necessary for this assignment
               //x  Doesn't matter
                //xx    Fast PWM bits
                  //xxx     setting no prescaler
    TCCR1C = 0b00000000;                                // Its 0x00 because we
dont need to set any register for force compare on any channels
}

int SetClkFrequency(long tFreq){
    // Setting the bounds for the frequency
    if (tFreq < 1000 || tFreq > 100000)
        return 1;
    else {
        OCR1A = (int)(8000000/tFreq);                   // setting the OCR1A
value
        OCR1B = (int)(0.5*OCR1A);                       // multipling the OCR1A
by 0.5 for a 50% duty cycle
        return 0;                                       // Returns 0 for a
successful run
    }
}

void loop() {
  // put your main code here, to run repeatedly:
  SetClkFrequency(PWMFreq);
}
```