Student Name: Prachi Patel

Student #: 5002380222

Student Email: patelp3@unlv.nevada.edu

Primary Github address: https://github.com/prachi173/da\_sp18

Directory: [https://github.com/prachi173/da\_sp18/tree/master/Design Assignments/DA2C](https://github.com/prachi173/da_sp18/tree/master/Design%20Assignments/DA2C)

Youtube Link: <https://www.youtube.com/watch?v=-ABx4CKwbtY&feature=youtu.be>

The following are required for successful completion of the design assignment:

* 1. a. AVR C code that has been compiled and working for all four tasks. Verify the period and duty cycle of the waveforms in simulation and emulation.
  2. b. The C code should be well documented with explanation of every instruction.
  3. c. A word document that contains the code with comments, complete schematics, that includes the AVR, components connected on the breadboard and LED should be included. Follow the template provided.
  4. d. A snapshot of the board with connected components and a video of the complete LED bar blink sequence should be recorded and uploaded to Youtube and the line to be provided for each task.
  5. e. The git directory should have DA2\DA2T1, DA2\DA2T2, … \_folders, with one doc file and video link file.

|  |
| --- |
| /\*  **\* DA2C\_T1A\_C.c**  \*  \* Created: 3/21/2019 8:51:54 PM  \* Author : patel  \*/  #include <avr/io.h>  #include <util/delay.h>  #define F\_CPU 16000000UL  int main(void)  {  uint8\_t OVFCount = 0; //initialize OVF Counter  DDRB = 0xFF; //set PB2 to output  TCCR0B = (1<<CS00) | (1<<CS02); // prescaler 1024  TCNT0 = 0x01; // reset timer    while(1)  {  while((TIFR0 & 0x01) == 0){} //do nothing if flag is 0  OVFCount++; //increment OVFCounter  if (OVFCount <= 13) //until ovfcounter reaches 13 timer cycles)  {  TCNT0 = 0x01; //keep reseting to 0x01 (counts from 1 to 256 so 255 times)  TIFR0 = 0x01; //reset overflow flag  }  if(OVFCount == 13) //when the overflow reaches 13 cycles  {  PORTB = 0x00; //set PORTB to 0    }  if(13 < OVFCount < 22) //until overflow reaches 22  {  TCNT0 = 0x01; //keep reseting timer  TIFR0 = 0x01; //keep resetting overflow flag  }  if(OVFCount == 22) //when the overflow reaches 22 cycles  {  PORTB = 0x04; //set PORTB2 on.  OVFCount = 0; //reset overflow counter  }  }  }A screenshot of a social media post  Description automatically generated  A screenshot of a social media post  Description automatically generatedA screenshot of a social media post  Description automatically generated |
| /\*  **\* DA2C\_T1B\_C.c**  \*  \* Created: 3/21/2019 10:51:54 PM  \* Author : patel  \*/  #include <avr/io.h>  #include <util/delay.h>  #define F\_CPU 16000000UL  int main(void)  {  uint8\_t OVFCount = 0; //initialize OVF Counter  DDRB = 0x04; //set PB2 to output  DDRC = 0x00; //set PortC to input  PORTC = 0x02; //pull up PinC1 to read input  TCCR0B = (1<<CS00) | (1<<CS02); // prescaler 1024  // TCCR0A = 0x00; // Normal Mode  TCNT0 = 0x03; // Reset timer to start at 3 (counts 253 times)    while(1)  {  OVFCount = 0; //reset overflow to 0  PORTB = 0x00; //set portb to 0. keeps it off until prompted  if(PINC & (1<<PINC1)) //if PINC1 is on  {  PORTB = 0x04; //set PortB2 to on  TCNT0 = 0x03; //Reset Timer to 3 but continue ovfcount    while(OVFCount < 40) //until overflow reaches 40 cycles  {  while((TIFR0 & 0x01) == 0){} //do nothing if overflow flag is 0  OVFCount++; //increment overflow counter  if(OVFCount < 40) //otherwise until it reaches 40  {  TIFR0 = 0x01; //reset overflow  TCNT0 = 0x03; //reset timer  }  }    }  }  }  A screenshot of a cell phone  Description automatically generated |
| /\*  **\* DA2C\_T2A.c**  \*  \* Created: 3/22/2019 10:00:25 PM  \* Author : patel  \*/  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #define F\_CPU 16000000UL  volatile int OVFCount; //intialize global counter so we can use it in all functions  int main(void)  {  DDRB = 0xFF; //set PORTB to output  TIMSK0 |= (1<<TOIE0); //enable overflow interrupt  TCNT0 = 0x20; //start timer at 0x20 or 32 (224 counts)  sei(); //enable interrupts  TCCR0B |= (1<<CS00) | (1<<CS02); //prescaler 1024    while (1)  {  //main loop. we have nothing other than interrupt function.  }  }  ISR(TIMER0\_OVF\_vect) //overflow interrupt function  {  OVFCount++; //increment overflow  if(OVFCount < 15) //until timer reaches 15 timer cycles (3360)  {  TCNT0 = 0x20; //reset timer  TIFR0 = 0x01; //reset overflow flag  }  if(OVFCount == 15) //when timer cycles 15 times  {  PORTB = 0x00; //set PORTB to 0  }  if(15 < OVFCount & OVFCount < 25) //between 15 and 25 cycles, reset timer and overflow flag)  {  TCNT0 = 0x20;  TIFR0 = 0x01;  }  if(OVFCount == 25) //at 25 cycles  {  PORTB = 0x04; //set PORTB2 on  OVFCount = 0; //reset overflow counter  }  }  A screenshot of a cell phone  Description automatically generatedA screenshot of a cell phone  Description automatically generated |
| /\*  **\* DA2C\_T2B.c**  \*  \* Created: 3/22/2019 11:24:03 PM  \* Author : patel  \*/  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #define F\_CPU 16000000UL  volatile int OVFCount; //intialize global counter so we can use it in all functions  int main(void)  {  DDRB = 0x04; //set PB2 to output  DDRC = 0x00; //set PortC to input  PORTC = 0x02; //pull up PINC1 to read input  TCCR0B = (1<<CS00) | (1<<CS02); // prescaler 1024  TIMSK0 |= (1<<TOIE0); //enable overflow interrupt  // TCCR0A = 0x00; //normal mode is default  TCNT0 = 0x03; //set timer to start at 3 or 253 counts  OVFCount = 0; //initialize counter  sei(); //enable interrupt    while(1)  {  PORTB = 0x00; //set PORTB2 to 0  if(PINC & (1<<PINC1)) //if PINC1 is ON  {  PORTB = 0X04; //set PORTB2 to ON  TCNT0 = 0X03; //reset timer to 3  OVFCount = 0; //reset overflow counter  while(OVFCount < 39) //until timer has cycled 39 times, do nothing  {    }    }  }  }  ISR(TIMER0\_OVF\_vect) //overflow interrupt function  {    OVFCount++; //increment overflow  if(OVFCount < 39) //until less 39  {  TCNT0 = 0x03; //reset timer  TIFR0 = 0x01; //reset overflow flag  }  }  A screenshot of a cell phone  Description automatically generated |
| /\*  **\* DA2C\_T3A.c**  \*  \* Created: 3/23/2019 1:40:33 AM  \* Author : patel  \*/  /\*  \* DA2C\_T2A.c  \*  \* Created: 3/22/2019 10:00:25 PM  \* Author : patel  \*/  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #define F\_CPU 16000000UL  volatile int OVFCount; //intialize global counter so we can use it in all functions  int main(void)  {  OVFCount = 0; //initialize overflow counter  OCR0A = 227; //compare value at 227  TCCR0A |= (1<<WGM01); //CTC mode    DDRB = 0xFF; //set PORTB to output  TIMSK0 |= (1<<OCIE2A); //compare interrupt enable  TCCR0B |= (1<<CS00) | (1<<CS02); //prescaler 1024  sei();  while (1)  {  //main loop. no function until interrupt  }  }  ISR(TIMER0\_COMPA\_vect) //timer compare interrupt function  {  OVFCount++; //increment overflow counter  if(OVFCount == 15) //when timer cycles 253 counts 15 times  {  PORTB = 0x00; //set PORTB to 0  }    if(OVFCount == 25) //when timer cycles 253 counts 25 times  {  PORTB = 0x04; //set PORTB2 to ON  OVFCount = 0; //reset overflow counter  }  }  A screenshot of a social media post  Description automatically generatedA screenshot of a social media post  Description automatically generated |
| /\*  **\* DA2C\_T3B.c**  \*  \* Created: 3/23/2019 1:56:02 AM  \* Author : patel  \*/  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #define F\_CPU 16000000UL  volatile int OVFCount; //intialize global counter so we can use it in all functions  int main(void)  {  OVFCount = 0; //initialize overflow counter  OCR0A = 240; //set compare value at 240  TCCR0A |= (1<<WGM01); //CTC Mode  DDRC = 0X00; //Set PORTC to input  PORTC = 0XFF; //pull up PORTC to read input  DDRB = 0xFF; //Set PORTB to output  TIMSK0 |= (1<<OCIE2A); //enable compare interrupt  TCCR0B |= (1<<CS00) | (1<<CS02); //prescaler 1024  sei();  //similar to the codes before, here the timer counts from 0 to 240 and the OVFCount increments evertime this timer reaches 240 and goes back to 0. at 41 overflow count, total timer count is 240x41 = 9840. 41 was rounded from 40.6  //the counter difference between Normal Mode and CTC mode is that normal counts from reset timer upto MAX (256) while CTC counts from 0 upto the compare value (OCR0A)    while(1)  {  PORTB = 0x00; //set PORTB2 to 0  if(PINC & (1<<PINC1)) //when PINC1 is on  {  PORTB = 0X04; //set PORTB2 to on  TCNT0 = 0X00; //reset timer  OVFCount = 0; //reset overflow count (to count time AFTER the interrupt)  while(OVFCount < 41) //until overflow count is , do nothing  {    }    }  }  }  ISR(TIMER0\_COMPA\_vect) //compare interrupt  {  OVFCount++;  //increment overflow. the above function could have been implemented here too but since only the delay uses overflow counter, I didn't.  }A screenshot of a social media post  Description automatically generated |

**Assembled board with AVR programmer and Logic Analyzer for verification**

A circuit board

Description automatically generated

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Prachi Patel