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%20Assignments/DA2A>

Youtube Link: <https://www.youtube.com/watch?v=vsxUoKnvJu8&list=PLUJP2DprnvJAYpwXOO40Ob2qDc82yK5Ye>

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

* 1. a. AVR ASM code that has been compiled and working for all tasks. Verify the period and duty cycle of the waveforms in simulation and emulation.
  2. b. AVR C code that has been compiled and working for all tasks. Verify the period and duty cycle of the waveforms in simulation and emulation.
  3. c. The C code should be well documented with explanation of every instruction.
  4. d. 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.
  5. e. 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.
  6. f. The git directory should have DA2\DA2T1, DA2\DA2T2, … \_folders, with one doc file and video link file.

Submission:

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

1. All compiled AVR ASM codes have verified period cycle of ~725ms.

A screenshot of a computer

Description automatically generated

1. I used the delay function to set the C codes at 725 period. 435 ON + 290 OFF

This was calculated using the (TimeON / (TimeON + TimeOFF)) \* 100 = DutyCycle.

I didn’t know how to verify them using C though because the simulator doesn’t show process status for C.

1. I used the Development given to us in the lab with AVR Programmer.
2. The codes are added below in order of

TASK 1 – Assembly

TASK 1 – C

TASK 2 – Assembly

TASK 2 – C

1. Youtube Video link:

<https://www.youtube.com/watch?v=vsxUoKnvJu8&list=PLUJP2DprnvJAYpwXOO40Ob2qDc82yK5Ye>

The Dev. Board used.

A circuit board on a table

Description automatically generated

|  |
| --- |
| ; DA2A\_1ASM.asm  ;  ; Created: 3/2/2019 6:48:27 PM  ; Author : patel  ;  ; Replace with your application code  .org 0  LDI R16, 0x04 ;Loaded for the second bit of the PortB as required  LDI R18, 0XFF ;Loaded to set all the PORTB to output  OUT DDRB, R18 ;Set all pins of PORTB to Output (connected to LEDs)  LDI R17, 0 ;Loaded 0  OUT PORTB, R17 ;Set all PORTB (led's) to 0  LDI R20, 5 ;Loaded 5 or 101  STS TCCR1B, R20 ;Set prescaler to 1024, which is done by assigning 101 to TCCR1B  begin:  RCALL delay ;Calling label delay  EOR R17, R16 ;XOR R17 and R16. This is a bitmasking technique used to clear the output pins  OUT PORTB, R17 ;Output PORTB.2 (bitmask to 00000100 since PORTB is[7:0]  RJMP begin ;Jump to label begin  delay:  LDS R29, TCNT1H ;Load High bits of counter to R29  LDS R28, TCNT1L ;Load Low bits of counter to R28  CPI R28, 0X1E ;Compare the lower bit to 0x1E  BRSH body ;Branch to label body if same or higher  RJMP delay ;if lower, RJMP delay  body:  CPI R29, 0x16 ;Compare the higher bits to 0x16  BRSH done ;Branch to label done if same or higher  RJMP delay ;if lower, branch to label delay  done:  LDI R20, 0x08 ;Load 0x08 into R20  STS TCNT1H, R20 ;Load 0x08 into higher bits of counter  LDI R20, 0xD9 ;Load 0xD9 into R20  STS TCNT1L, R20 ;Load 0xD9 into lower bits of counter. This sets the counter to 0x08D9 which is 2265.  ;Since we want 60% DC, I divided them cycle into 435ms ON and 290ms OFF.  ;Using the prescaler, I calculate this times to 3397 and 2265 or 0x0D45 and 0x08D9  ;The time starts at this lower bit 0x08D9 and keeps LED ON until timer reaches 0x161E  ;Then the LED stays OFF for 290ms. The total period of this cycle is about 725ms.  RET ;Returns to label begin |
| /\*  \* DA2A\_1C.c  \*  \* Created: 3/1/2019 10:56:38 PM  \* Author : patel  \*/  #include <avr/io.h>  #define *F\_CPU* 8000000UL  #include <util/delay.h>  int main(void)  {  DDRB |= (1<<PB2);  while (1)  {  PORTB |= (1<<PB2);  *\_delay\_ms*(435); //For 60% duty cycle and period of 725ms, I used the formula [DC = (Timer\_on / (Timer\_on + Timer\_off))\*100)  PORTB &= ~(1<<PB2);  *\_delay\_ms*(290); //LED connected to PORTB2 stays on for 435ms and off for 290ms. This gives us DC = 60%.  }  return 1;  } |
| ;  ; DA2A\_2ASM.asm  ;  ; Created: 3/2/2019 10:43:41 PM  ; Author : patel  ;  ; Replace with your application code  .include <m328pdef.inc>  .org 0  LDI R20, 5 ;Loaded 5 or 101  STS TCCR1B, R20 ;Set prescaler to 1024, which is done by assigning 101 to TCCR1B  LDI R16, 0x04 ;Loaded for the second bit of the PortB as required  LDI R17, 0 ;Loaded 0 to R17  LDI R18, 0xFF ;Loaded R18 with all 1  OUT DDRC, R17 ;Set portc to input  OUT PORTC, R18 ;set pull up to active high  OUT DDRB, R16 ;Set all pins of PORTB to Output (connected to LEDs)  OUT PORTB, R17 ;Set all PORTB (led's) to 0  begin:  IN R19, PINC ;Read from PINC (switch)  CPI R19, 0x01 ;Compare to PINC to 0x01. Since the switch is connected to 0x04, it works  BRSH led ;If highers than 0x01 (or if switch ON), branch to led  RCALL begin ;or go back to begin and check again  led:  OUT DDRC, R19 ;set PortC2 to output  OUT PORTC, R19 ;set PORTC2 to ON. I did this to keep switch on for delay  EOR R17, R19 ;XOR R17 and R19  OUT PORTB, R17 ;Output PORTB.2 (bitmask to 00000100 since PORTB is[7:0]  CPI R17, 0x01 ;If R17 is bigger than 0x01  BRSH delay ;Branch to delay  delay:  LDS R29, TCNT1H ;Load High bits of counter to R29  LDS R28, TCNT1L ;Load Low bits of counter to R28  CPI R28, 0X25 ;Compare the lower bit to 0x1E  BRSH body ;Branch to label body if same or higher  RJMP delay ;if lower, RJMP delay  body:  CPI R29, 0x26 ;Compare the higher bits to 0x16  BRSH done ;Branch to label done if same or higher  RJMP delay ;if lower, branch to label delay  done:  LDI R20, 0x00 ;Load 0x08 into R20  STS TCNT1H, R20 ;Load 0x08 into higher bits of counter  LDI R20, 0x00 ;Load 0xD9 into R20  STS TCNT1L, R20 ;Load 0xD9 into lower bits of counter. This sets the counter to 0x08D9 which is 2265.Since we want 60% DC, I divided them cycle into 435ms ON and 290ms OFF.Using the prescaler, I calculate this times to 3397 and 2265 or 0x0D45 and 0x08D9. The time starts at this lower bit 0x08D9 and keeps LED ON until timer reaches 0x161E. Then the LED stays OFF for 290ms. The total period of this cycle is about 725ms.  RET ;Returns to label begin |
| /\*  \* DA2A\_2C.c  \*  \* Created: 3/2/2019 9:02:42 PM  \* Author : patel  \*/  #include <avr/interrupt.h>  #include <avr/io.h>  #define *F\_CPU* 8000000UL  #include <util/delay.h>  int main(void)  {  DDRB |= (1 << PORTB2); //set portb2 to output  DDRC &= ~(1 << PORTC2);//set portc2 to input  while (1)  {  DDRC &= ~(1 << PORTC2); //set portc to input (this is for loop)  if((PINC & (1 << PORTC2))) //if PINC2 is on  {  DDRC |= (1 << PORTC2); //Set PORTC to output  PORTC |= (1 << PORTC2);//Set PORTC2 to ON (this reads switch as ON)    PORTB |= (1<<PORTB2); //OUTPUT PB2 (LED3)  *\_delay\_ms*(1250); //Delay for 1.25s  PORTB &= ~(1<<PORTB2); //Turn off PB2 (LED3)  DDRC &= ~(1 << PORTC2); //Set Portc to input again and loop  }  else  {  PORTB &= ~(1<<PORTB2); //if PINC2 is not on, keep LED3 OFF.  }    }    return 1;  } |

**Student Academic Misconduct Policy**

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

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

Prachi Patel