Vincent Tuason

CPE301 – SPRING 2018

Design Assignment 2

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 1 | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 2. | Code for Task 1 |  |  |
| 3. | Code for Task 2 |  |  |
| 4. | Code for Task 3 |  |  |
| 5. | Code for Task 4 |  |  |
| 6. | Code for Task 5 |  |  |
| 7. | Schematics and Components |  |  |
| 8. | VIDEO LINKS OF EACH DEMO |  |  |
| 9. | Github Link |  |  |
|  |  |  |  |
|  |  |  |  |

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

ATMega328P

AVR USB Programmer (from lab section)

Block diagram is further down the document along with the schematics

10k Resistor

1 Yellow LED

1 Push button

Breadboard

Stripped Wires

1. **Code for task 1**

;Assembly Code for Task 1

.org 0

ldi r18, 0x00 ;load 0 into r18, use this as a counter

ldi r16, 0xFF ;load 255 into r16

out DDRB, r16 ;set all bits of DDRB to 1

loop:

SBI portb, 0x02 ;Make PORTB 2 as an output

rcall delay\_250ms ;call the delay subroutine

CBI portb, 0x02 ;clear bit 2 in portb

rcall delay\_250ms ;call the delay subroutine

rjmp loop ;keep looping

delay\_250ms:

rcall delay\_1ms ;call the 1ms delay subroutine

rcall delay\_1ms ;call the 1ms delay subroutine

rcall delay\_1ms ;call the 1ms delay subroutine

rcall delay\_1ms ;call the 1ms delay subroutine

rcall delay\_1ms ;call the 1ms delay subroutine

inc r18 ;increment my counter

CPI r18, 0x32 ;compare to 50. 50 \* 5 = 250ms

brne delay\_250ms ;if counter != 50, then keep looping

ret ;if counter = 50, return

delay\_1ms:

push r16 ;save the value in r16

ldi r16,23 ;accounts for overhead

delay\_1ms1: ; 10 us/loop

nop ; [1 cycle ]

nop ;[1 cycle ]

nop ;[1 cycle ]

nop ;[1 cycle ]

nop ;[1 cycle ]

nop ;[1 cycle ]

nop ;[1 cycle ]

dec r16

brne delay\_1ms1 ; [2 cycles ]

pop r16 ;restore the value in r16

ret ;return

//C-Code for Task 1

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB |= (1<<PB2); //set PB2 as an output

while (1)

{

PORTB |= (1<<PB2); //Turn PB2 On

*\_delay\_ms*(250); //wait for 250ms

PORTB &= ~(1<<PB2); //Turn PB2 Off

*\_delay\_ms*(250); //wait for 250ms

}

return 1;

}

1. **Code for task 2**

;Assembly Code for Task 2

.org 0

LDI R16, 0x00 ;load 0 into r16

SBI DDRB, 2 ;set PB2 as an output

SBI DDRD, 2 ;set PD2 as an output

here:

in R16, PIND ;Load PIND (I/O location) to r16

out PORTB, R16 ;store register into portb (I/O location)

cpi R16, 0x04 ;compare r16 to 4

BRSH getDelay ;if its the same or higher, jump to delay for 1 second

rjmp here ;keep jumping to begin to poll for an event

getDelay:

LDI R19, 8 ;load 8 into r19

delays:

LDI R20, 125 ;load 125 into r20

delayss:

LDI R21, 250 ;load 250 into r21

delaysss:

dec r21 ;decrement r21

nop ;perform single cycle no operation

brne delaysss ;branch to delaysss if r21 != 0

dec r20 ;decrement r20

brne delayss ;branch to delaysss if r20 != 0

dec r19 ;decrement r19

brne delays ;branch to delaysss if r19 != 0

rjmp here ;relative jump to label 'here'

//C-code for Task 2

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB = (1<<PB2); //set PB2 as an output

DDRD = 0x00; //set all bits of DDRD to 0

while (1)

{

if((PIND&0x04)==0x04) //if PD2 is changes execute instructions else stay off

{

PORTB = 0xff; //Turn PB2 on

*\_delay\_ms*(1000); //wait for 1 second then turn off

}

else

PORTB = 0x00; //turn PORTB off

}

return 0;

}

1. **Code for task 3**

**//C-Code for Task 3**

#include <avr/io.h>

int main(void)

{

DDRB = 0x04; //set pb2 as an output

TCCR0A = 0x02; //set CTC mode

TCCR0B = 0x05; //prescale 1024

TCNT0 = 0x00; //initialize timer 0 to 0

OCR0A = 64; //continously compared to timer 0

while (1)

{

if(TIFR0 & (1 << OCF0A)) //if there is an overflow

PORTB ^= 0x04; //turn pb2 on

TIFR0 |= (1 << OCF0A); //clear flag

}

}

1. **Code for task 4**

**;Assembly Code for Task 4 (This one does not work, but this is what I have for it)**

.include "m328pdef.inc"

.org 0

jmp main

.org 0x16

jmp T0\_OV\_ISR

.org 0x100

main:

SBI DDRB, 2 ;PB2 as an output

SBI PORTB, 0 ;activate pull up of PB0

ldi r20, (1<<TOIE0) ;enable timer0 overflow interrupt

STS TIMSK0, R20

sei ;set enable interrupt

LDI R16, 12

out TCNT0, R16 ;load timer with 12 for next round

ldi r20, 0x01

out TCCR0B, R20 ;internal clock, no prescaler

here:

jmp here ;keep CPU busy waiting for interrupt

.org 0x200

T0\_OV\_ISR:

ldi r16, 12

out TCNT0, R16 ;load timer with 12

IN R16, PORTB ;read PORTB

ldi r17, 0x04 ;for toggling pb2

eor r16, r17

out PORTB, R16 ;toggle pb2

reti ;return from interrupt

//C-Code for Task 4

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

ISR(TIMER0\_OVF\_vect)

{

PORTB ^= 0x04; // output to pb2

TCNT0 = 12; // 255 - (((1Mhz / 1024)\*0.25)-1)

}

int main(void)

{

DDRB = 0x04;

TIMSK0 = (1<<TOIE0); //Overflow Interrupt Enable.

TCCR0B = 5; //set prescale to 1024.

TCNT0 = 12; // 255 - (((1Mhz / 1024)\*0.25)-1)

sei(); //Interrupt enable.

while(1)

{

}

}

1. **Code for task 5**

**;Assembly for Task 5**

.include "m328pdef.inc"

jmp main ;jump to label "main"

.org INT0addr ;address of EXT\_INT0

jmp INT0\_interrupt ;go to interrupt sub routine

main:

SBI DDRB, 2 ;Set PB2 as an output

ldi r16, (1 << ISC01) | (1 << ISC00) ;rising edge toggle

STS EICRA, r16 ;store r16 into EICRA

in r16, EIMSK ;load I/O (INT0) into r16

ori r16, (1<<INT0) ;or r16 and 1

out EIMSK, r16 ;output R16 value to INT0

sei

while:

rjmp while ;poll for an event

INT0\_interrupt:

in r16, PORTB ;load (I/O Location) PORTB into r16

com r16 ;perform one's complement

out PORTB, r16 ;store r16 into I/O location Portb

call getDelay ;get a 1 second delay

ldi r16, 0x00 ;load 0 into r16

out PORTB, r16 ;turn PB2 off

reti ;return from interrupt

getDelay: ;this is a 1 second delay routine

LDI R19, 8 ;load 8 into r19

delays:

LDI R20, 125 ;load 125 into r20

delayss:

LDI R21, 250 ;load 250 into r21

delaysss:

dec r21 ;decrement r21

nop ;perform single cycle no operation

brne delaysss ;branch to delaysss if r21 != 0

dec r20 ;decrement r20

brne delayss ;branch to delaysss if r20 != 0

dec r19 ;decrement r19

brne delays ;branch to delaysss if r19 != 0

ret ;relative jump to label 'here'

//C-code for task 5

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

ISR(INT0\_vect)

{

PORTB = 0x04; //turn PB2 on

*\_delay\_ms*(1000); //wait for 1 second

PORTB = 0x00; //turn PB2 off

}

int main(void)

{

EIMSK = 0x01; //Enable External Interrupt 0

EIFR = 0x01; //Enable External Interrupt Flag Request 0

EICRA = 0x03; //Rising edge of Int0 generates an interrupt request

sei(); //Enable 'I' Bit in status register

DDRB = 0xFF; //set all bits in DDRB to 1

while (1) //poll for an event

{

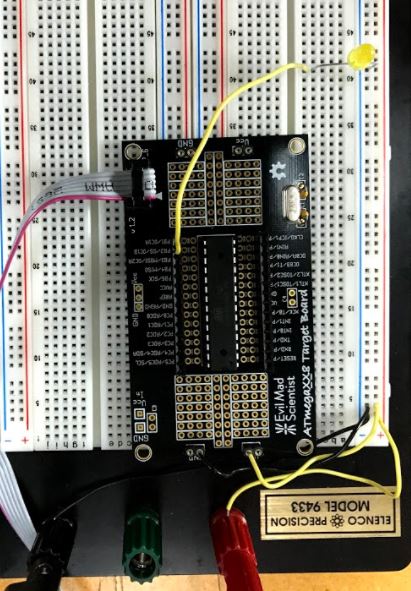
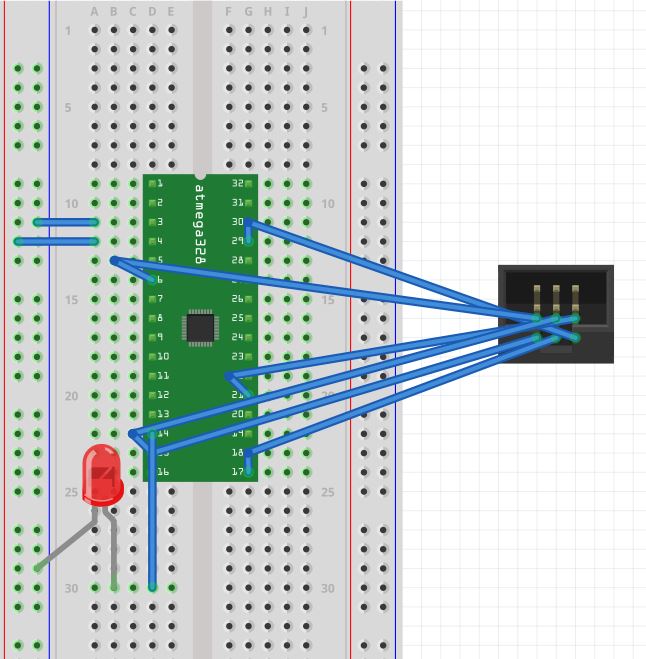
}

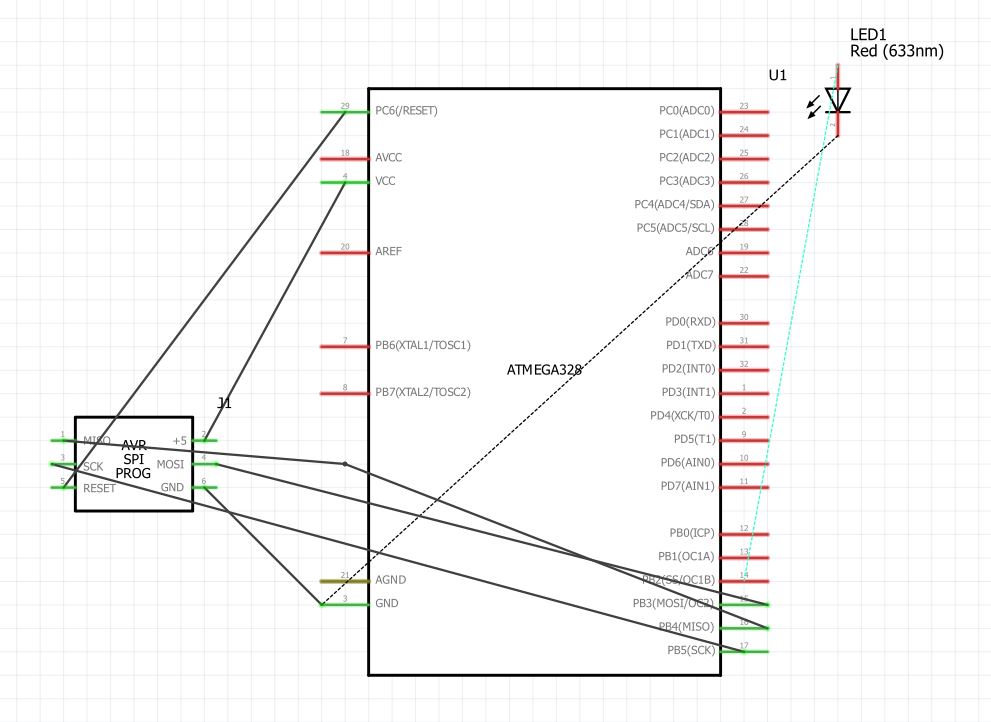
return 0;

}

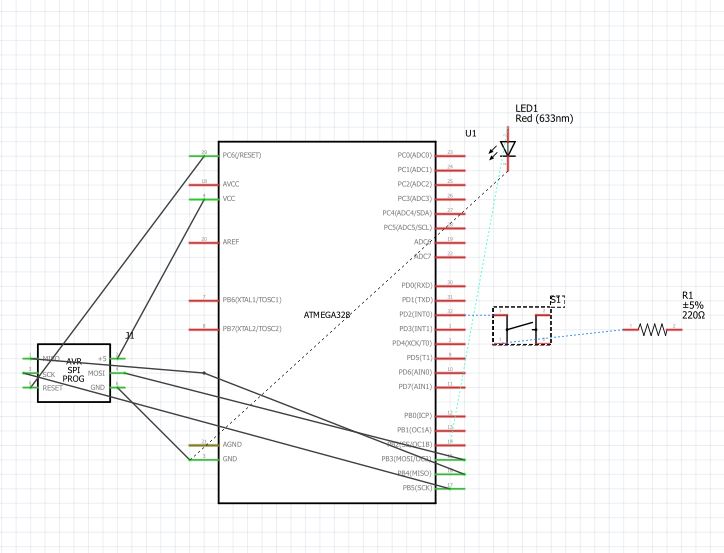
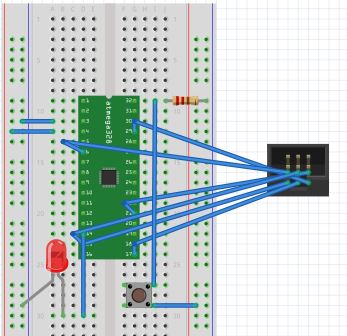
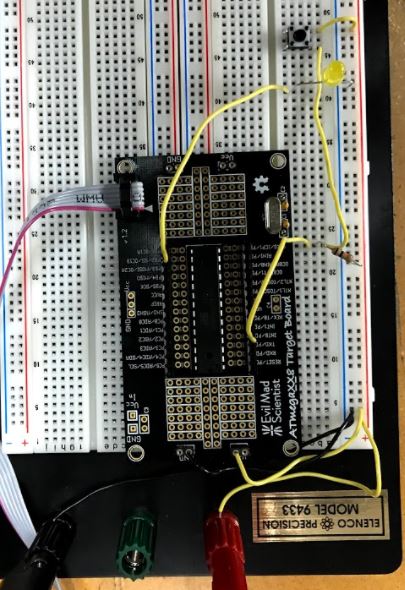
1. **Schematics, Components connected on the breadboard**

**For Tasks 1, 3 and 4**

** **

****

**For Tasks 2 and 5**

****

1. **VIDEO LINKS OF EACH DEMO**

<https://www.youtube.com/channel/UCcPVbEBGsL1quCrWGd_NTFQ/playlists?disable_polymer=1>

**Then click on DA2 Playlist**

1. **GITHUB LINK OF THIS DEMO**

<https://github.com/vason13/vasonGIT.git>

Please look at the “DA2 Final” folder. The other DA2 folder was a mistake.

**Student Academic Misconduct Policy**

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

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

Vincent Tuason