CPE301 – SPRING 2019

Design Assignment 2B

Student Name: Allis Hierholzer

Student #: 2000160356

Student Email: hierholz@unlv.nevada.edu

Primary Github address: https://github.com/acexhp/submission\_da.git

Directory: Repository/cpe301/DesignAssignment/DA2B

Task:

1. Implement Design Assignment 2A.2 using INT0 (PD2 pin) interrupt mechanism.

-Connect a switch to PORTC.2 (active high - turn on the pull up transistor) to poll for an event to turn on the led at PORTB.2 for 1.250 sec after the event.

Submission:

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

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.

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.

c. The C code should be well documented with explanation of every instruction.

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.

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.

f. The git directory should have DA2\DA2T1, DA2\DA2T2, … folders, with one doc file and video link file.

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

* ATMEGA328P XPLAINED MINI
* MULTIFUNCTION SHIELD
* ATMEL STUDIO 7.0

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

**ASSEMBLY CODE**

; assembly.asm

; Author : hierholz

.include <m328pdef.inc>

.ORG 0 ;location for reset

JMP MAIN

.ORG 0x02 ;location for external interrupt 0

JMP EX0\_ISR

MAIN:

LDI R20, HIGH(RAMEND)

OUT SPH, R20

LDI R20, LOW(RAMEND)

LDI R20, 0X2 ;initialize stack

LDI R20, 0X2 ;make INT0 falling edge triggered

STS EICRA, R20

SBI DDRB, 5 ;set PORTB5 as output

SBI PORTB, 5 ;turns off led

SBI PORTD, 2 ;pull-up activated

LDI R20, 1<<INT0 ;enable INT0

OUT EIMSK, R20 ;bit masking

SEI ;enable interrupts

HERE: JMP HERE

EX0\_ISR:

IN R21, PORTB ;read PORTB as input

LDI R22, (1<<5) ;00100000 for toggling PB5

EOR R21, R22 ;exor to toggle led

OUT PORTB, R21 ;led turns on

RCALL delay\_1s

RCALL delay\_250ms

RETI ;returns from interrupt

delay\_1s: ;creating delay of 1s

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_100ms

ret

delay\_250ms: ;creating delay of 250ms

rcall delay\_100ms

rcall delay\_100ms

rcall delay\_50ms

ret

delay\_100ms: ;creating delay of 100ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

ret

delay\_50ms: ;creating delay of 50ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

rcall delay\_10ms

delay\_10ms: ;creating delay of 10ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

ret

delay\_1ms: ;creating delay of 1ms

rcall delays\_16MHz

rcall delays\_16MHz

rcall delays\_16MHz

rcall delays\_16MHz

rcall delays\_16MHz

rcall delays\_16MHz

rcall delays\_16MHz

rcall delays\_16MHz

ret

delays\_16MHz: ;delays 1ms for 16MHz

push r16 ;save value in r16

ldi r16, 180 ;running 1980 cycles

delays\_16MHz\_: ;1ms per loop

nop ;1 cycle

nop

nop

nop

nop

nop

nop

nop

dec r16

brne delays\_16MHz\_ ;2 cycles

pop r16 ;restore value in r16

ret

**C CODE**

/\*

\* ccode.c

\* Author : hierholz

\*/

#define *F\_CPU* 16000000UL // clock runs at 16 MHz

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

int main(void)

{

DDRB |= (1<<5); //PB5 an output (PB2/3/4 led is burnout)

PORTB |= 1<<5; //led PB5 off

PORTD |= 1<<2; //pull up activated

EICRA = 0x2; //make INTO falling edge triggered

EIMSK = (1<<INT0); //enable external interrupt 0

sei (); //enable interrupts

while (1)

{ //wait for interruption

}

}

ISR (INT0\_vect) //ISR for external interrupt 0

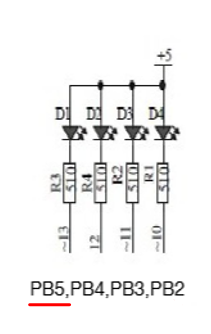
{

PORTB ^= (1<<5); //toggle PORTB.5

*\_delay\_ms*(1250);

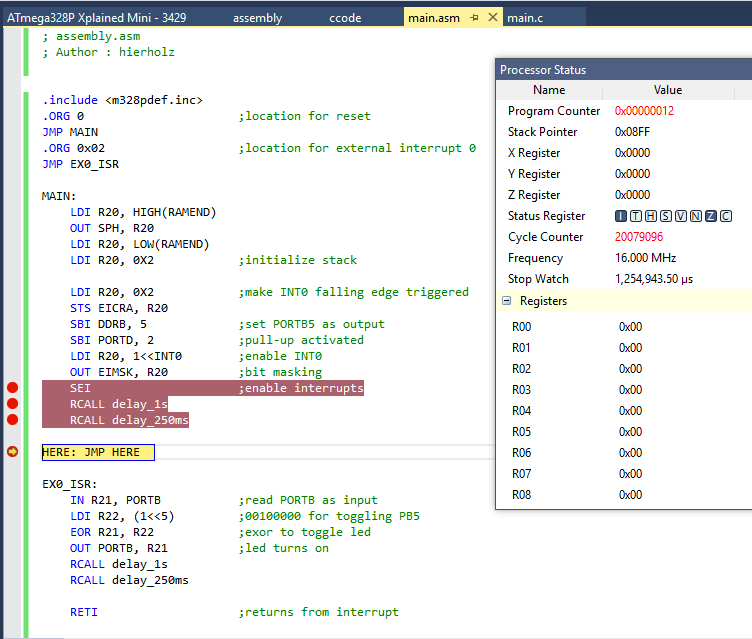
}

1. **SCHEMATICS**

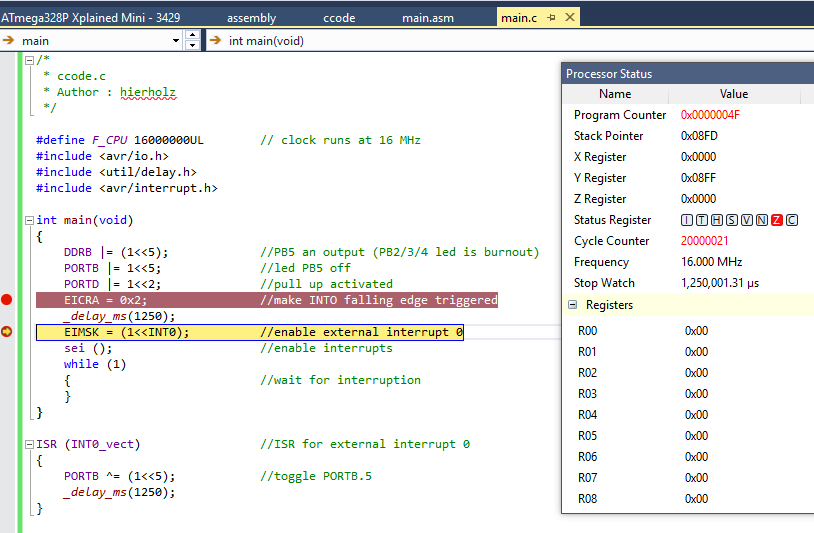


1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

**ASSEMBLY CODE**

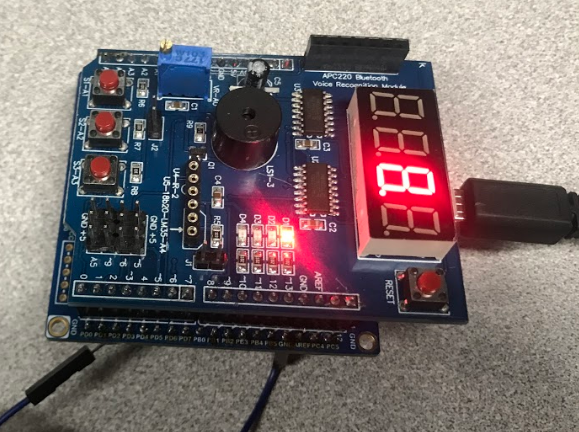


**C CODE**



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

**The board set up is the same for all exercises**

****

1. **VIDEO LINKS OF EACH DEMO**

**ASSEMBLY**

<https://youtu.be/0iSfr8PT67s>

**C CODE**

<https://youtu.be/LSQ4L6o7rGQ>

1. **GITHUB LINK OF THIS DA**

<https://github.com/acexhp/submission_da.git>

**Student Academic Misconduct Policy**

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

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

Allis Hierholzer