CPE301 – SPRING 2019

Design Assignment 2B

Student Name: Ali Asadi

Student #: 5004200625

Student Email: aliasadivet@gmail.com

Primary Github address: https://github.com/aliasadivet/Submitions\_DA\_AliAsadi

Directory: DessignAssignments

The goal of the assignment is use GPIO Interrupts.

Introduction

When an interrupt occurs, the current program execution is stopped, the context is saved and the control jumps to Interrupt Service Routine (the ISR). When the ISR is executed, the main program execution is continued. This is usually the case with most simple Micro-controllers. We have been working with Atmega32 for this series, however there shouldn't be much difference making it work with other AVR MCUs. Now, let us look at how to configure the External Interrupts in AVR.

Steps to configure the Interrupts:

1. Set INT1 and INT0 bits in the General Interrupt Control Register (GICR)
2. Configure MCU Control Register (MCUCR) to select interrupt type.
3. Set Global Interrupt(I-bit) Enable bit in the AVR Status Register(SREG)
4. Handle the interrupt in the Interrupt Service Routine code.

INT1 and INT0 enable the two interrupts. MCUCR helps in configuring the type of interrupt, level, edge triggered etc. The I-bit in SREG is the master control for all interrupts in AVR micro-controller. Observe the sequence it is turned on after all the interrupts are configured. This prevents, any interrupt to occur before rest of them are configured.

Programming

The following is the assembly code to implement Design Assignment 2A.2 using INT0 interrupt mechanism.

;

; DA2B.asm

;

; Created: 3/5/2019 1:51:07 AM

; Author : Ali Asadi

;

rjmp main

.org 0x0001 ; location of Interrupt service routine

rjmp INT0\_vect ; jump to INT0\_vect location

.org 0x0020

INT0\_vect:

cbi PORTB, 2 ; PORTB.2 - Output High

ldi r24, 250 ; load r24 with 250 to generate a delay of 1250mS

rcall Delay\_ASM ; call delay

sbi PORTB, 2 ; PORTB.2 - Output Low

reti ; return from thr interrupt

main:

;stack initialization

ldi r16, high (RAMEND)

out SPH, r16

ldi r16, low (RAMEND)

out SPL, r16

rcall Int\_Init\_ASM

loop:

rjmp loop

Int\_Init\_ASM:

cbi DDRD, 2 ; DDRD bit 2 = 0 INPUT Mode

sbi PORTD, 2 ; PORTD bit 2 = 1 PULLUP Enable

sbi DDRB, 2 ; PORTB.2 - Output

sbi PORTB, 2 ; PORTB.2 - Output Low

ldi r16, 0x03 ; load 0x00000011 in r16

sts EICRA, r16 ; store r16 in EICRA

sbi EIMSK, 0 ; set bit 0 in EIMSK register to enable INT0

sei ; Enable Global interrupt

ret ; return from the function

; Delay\_ASM generates a delay of 5mS for each loop

; So to generate a delay of 100mS value of 100/5 = 20 is sent via r24 register

; ldi - 1 Cycle

; sbiw - 2 cycle

; brne - 2 cycle

; so sbiw and brne loop till the value in r24 decrements to zero

; for example if r25 = 1 , then those two lines executes 1 time which is 4 cycle

; if r24 = 10, then executes 10 times which is 4 \* 10 = 40 cycle

; frequency is 8Mhz then = each cycle is 1/8Mhz = 1/8uS

; therefore 4Cycle = 4\*(1/8)uS = 0.5uS delay

; to get 5mS = 5000uS , number of cycles = 5000uS / (1/8)uS = 5000 \* 8 cycles

; but this loop already generates delay of 4 cycles

; to take that into account 5000 \* 8 / 4 = 10000 cycles

; there 10000 \* 4 = 40000cycles \* (1/8uS) = 5000 uS

Delay\_ASM:

mov r16, r24 ; move r24 to r16

ldi r24, low(5000 \* 8 / 4 ) ; load lower byte of the delay value to r24

ldi r25, high(5000 \* 8 / 4 ) ; load higher byte of the delay value to r25

sbiw r24, 1 ; subtract Immediate from word - r25:r24 - 1

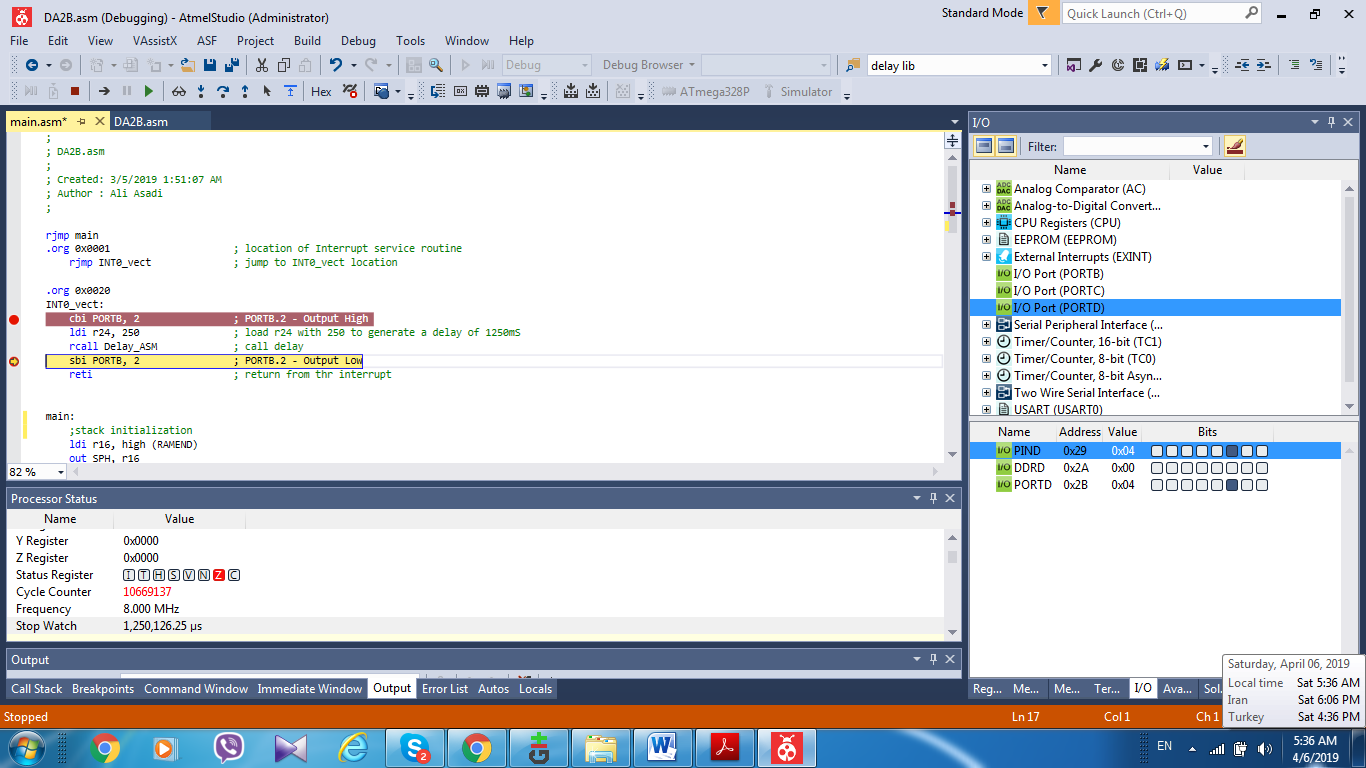
brne PC-1 ; branch if not equal ie branch till result of previous instruction is Zero

dec r16 ; decrement r16

brne PC-5 ; branch till r16 becomes zero

ret ; exit or return

The following screenshot demonstrates the simulation result of the code.



The following is the C code to implement Design Assignment 2A.2 using INT0 interrupt mechanism.

/\*

\* DA2B.c

\*

\* Created: 3/6/2019 2:56:14 AM

\* Author : Ali Asadi

\*/

#include <inttypes.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/sleep.h>

#include <util/delay.h>

ISR(INT0\_vect)

{

PORTB &= ~\_BV(2); // Clear the PORTD pin 2

*\_delay\_ms*(1250); // delay 1.250 Secs

PORTB |= \_BV(2); // Set the PORTD pin 2

}

void int\_init()

{

DDRB |= \_BV(2); // PORTB pin 2 output

PORTB |= \_BV(2); // Set the PORTD pin 2

DDRD &= ~\_BV(2); // Clearing DDRx register set that pin to INPUT mode

PORTD |= \_BV(2); // setting PORTx, for an INPUT pin , enables the pull-up resister

EICRA |= 0x03; // EICRA to configure the edge of the incoming pulse 0x11 or 3 - Rising edge

EIMSK |= 1; // Enable INT0

sei(); // Enable global Interrupt

}

void main()

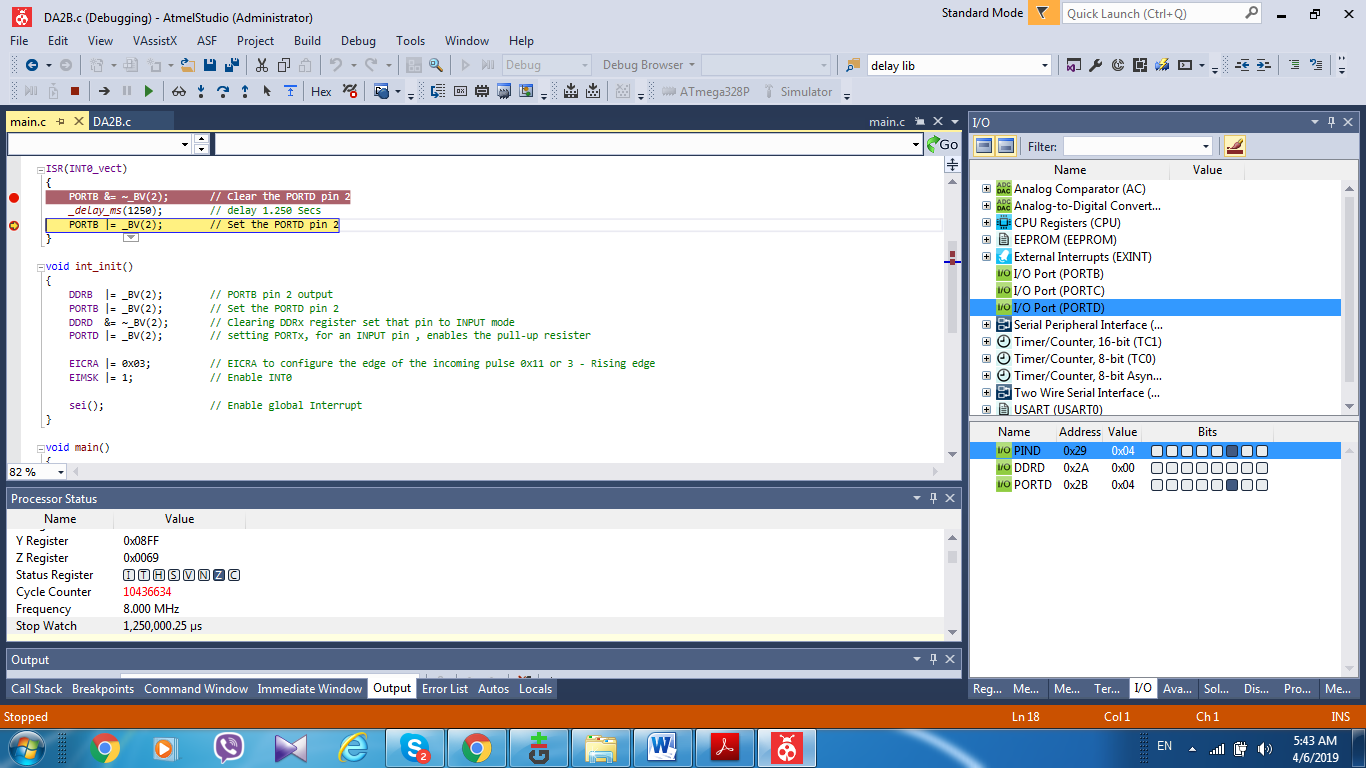
{

int\_init();

while (1);

}

As it is shown in the screenshots below, the delay time of 1.25 sec has been executed.



The simulation and emulation tesults are posted on YouTube, and can be found on linkes below:

<https://www.youtube.com/watch?v=iwlMftSIJ_s&t=22s>

<https://www.youtube.com/watch?v=64Fe2H-L384>