CPE301 – SPRING 2019

Design Assignment 2B

Student Name: Rocky Yasuaki Gonzalez
Student #: 5003229733
Student Email: gonzar14@unlv.nevada.edu
Primary Github address: https://github.com/rockyg1995/ihswppdar.git
Directory: C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA2B

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Multifunction Shield
Atmega328PB Xplained Mini
Wire Connector
Micro USB Cable (Power Supply)

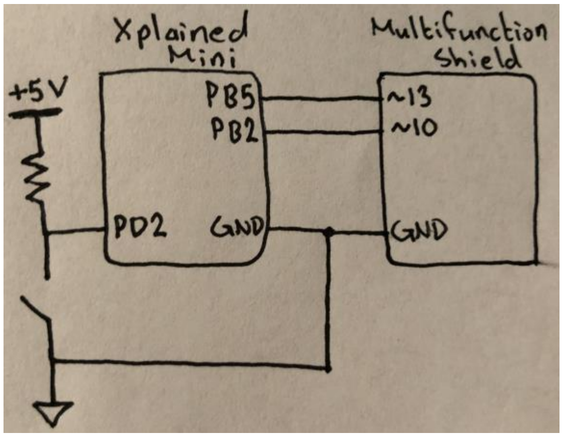

Figure 1 – Block Diagram/Pin Connections

2. INITIAL/DEVELOPED CODE OF TASK 1/A

```
; DA2B - Rocky Gonzalez.asm
;
; Created: 3/9/2019 8:39:31 AM
; Author: rocky

.ORG 0                      ; Location for reset
       JMP Main             ; Label Address for Main
.ORG 0x02                   ; Location for external interrupt 0
       JMP EX0_ISR          ; Label Address for External Interrupt 0

Main:
       ; Interrupt Related Initialization
       LDI R20, HIGH(RAMEND)       ; Load Upper RAM end address
       OUT SPH, R20                ; Initialize Upper Stack Pointer
       LDI R20, LOW(RAMEND)        ; Load Lower Ram end address
       OUT SPL, R20                ; Initialize Lower Stack Pointer
       LDI R20, 0x2                ; Make 'INT0' falling edge triggered
       STS EICRA, R20              ; External Interrupt Control Register A 'xxxx 0010'
       SBI PORTD, 2                ; Activate pull-up in 'Interrupt 0' (PD2)
       SBI DDRB, 5                 ; Set PB5 as an Output (Interrupt LED)
       CBI PORTB, 5                ; Set PB2 as 'High'
       LDI R20, (1<<INT0)          ; Enable 'INT0'
       OUT EIMSK, R20              ; Store into 'EIMSK'
       SEI                         ; Enable Global Interrupts
       ; Main Function Initialization
       SBI DDRB, 2                 ; Set PB2 as an Output
       SBI PORTB, 2                ; Set PB2 as 'High'
Repeat:
       CAll delay435ms      ; Subroutine to Delay program 435ms
       CBI PORTB, 2         ; Toggle LED 'Low' at PORTB.2
       CAll delay290ms      ; Subroutine to Delay program 290ms
       SBI PORTB, 2         ; Toggle LED 'High' at PORTB.2
       RJMP Repeat          ; Repeat LED Squarewave Period (725ms)

delay435ms:                 ; Delay Program '290ms'
       Push R16             ; Store R16 into Stack
       Push R17             ; Store R17 into Stack
       Push R18             ; Store R18 into Stack
       LDI R18, 15          ;
L3:    LDI R17, 200         ; 15
L2:    LDI R16, 232         ; 200*15
L1:    NOP                  ; 232*200*15
       NOP                  ; 232*200*15
       NOP                  ; 232*200*15
       NOP                  ; 232*200*15
       NOP                  ; 232*200*15
       NOP                  ; 232*200*15
       NOP                  ; 232*200*15
       NOP                  ; 232*200*15
       DEC R16              ; 232*200*15
       BRNE L1              ; 232*200*15
       DEC R17              ; 200*15
       BRNE L2              ; 200*15
       DEC R18              ; 15
       BRNE L3              ; 15
       Pop R18              ; Restore R18 from Stack
       Pop R17              ; Restore R17 from Stack
       Pop R16              ; Restore R16 from Stack
       RET                  ; Complete Subroutine

delay290ms:                 ; Delay Program '290ms'
       Push R16             ; Store R16 into Stack
       Push R17             ; Store R17 into Stack
       Push R18             ; Store R18 into Stack
       LDI R18, 10          ;
M3:    LDI R17, 200         ; 10
M2:    LDI R16, 232         ; 200*10
M1:    NOP                  ; 232*200*10
       NOP                  ; 232*200*10
       NOP                  ; 232*200*10
       NOP                  ; 232*200*10
       NOP                  ; 232*200*10
       NOP                  ; 232*200*10
       NOP                  ; 232*200*10
       NOP                  ; 232*200*10
       DEC R16              ; 232*200*10
       BRNE M1              ; 232*200*10
       DEC R17              ; 200*10
       BRNE M2              ; 200*10
       DEC R18              ; 10
       BRNE M3              ; 10
       Pop R18              ; Else, Restore R18 from Stack
       Pop R17              ; Restore R17 from Stack
       Pop R16              ; Restore R16 from Stack
       RET                  ; Complete Subroutine

delay1s:                    ; Delay Program '290ms'
       Push R16             ; Store R16 into Stack
       Push R17             ; Store R17 into Stack
       Push R18             ; Store R18 into Stack
       LDI R18, 32          ;
N3:    LDI R17, 200         ; 32
N2:    LDI R16, 250         ; 200*32
N1:    NOP                  ; 250*200*32
       NOP                  ; 250*200*32
       NOP                  ; 250*200*32
       NOP                  ; 250*200*32
       NOP                  ; 250*200*32
       NOP                  ; 250*200*32
       NOP                  ; 250*200*32
       NOP                  ; 250*200*32
       DEC R16              ; 250*200*32
       BRNE N1              ; 250*200*32
       DEC R17              ; 200*32
       BRNE N2              ; 200*32
       DEC R18              ; 32
       BRNE N3              ; 32
       Pop R18              ; Restore R18 from Stack
       Pop R17              ; Restore R17 from Stack
       Pop R16              ; Restore R16 from Stack
       RET                  ; Complete Subroutine

EX0_ISR:
       IN R21, PORTB ;      ; Read in Value of PB5 to R21
       LDI R22, (1<<5)      ; '0010 0000' for toggling PB5
       EOR R21, R22         ; XOR Current Status of Interrupt LED
       OUT PORTB, R21       ; Output Interrupt LED
       CAll delay1s         ; Subroutine to Delay program 1000ms
RETI                        ; Return to Original Placement in Code
```

3. INITIAL/DEVELOPED CODE OF TASK 2/A

```
/*
 * DA2B - C.c
 *
 * Created: 3/9/2019 11:04:58 AM
 * Author: rocky
 */

#define F_CPU 16000000             // Slow Rate of delay from 1MHz to 8MHz
#include <avr/io.h>                // Standard AVR Library
#include <avr/interrupt.h>         // AVR library containing interrupt functions
#include <util/delay.h>            // AVR library containing _delay_ms() function

#define LED 0b00000100             // Modify LED Bit here (Currently PB2)
#define ILED 0b00100000            // Modify Interrupt LED Bit here (Currently PB5)

int main(void) {                   // Main Function
       DDRB = LED | ILED;          // Set LED (PB2) and ILED (PB5) as an Output
       PORTD |= 1<<2;              // Activate Pull-up in PD2
       PORTB |= LED;               // Set LED as 'High'
       PORTB &= ~ILED;             // Set ILED as 'Low'
       EICRA = 0x2;                // Set as Falling Edge Trigger
       EIMSK = (1<<INT0);          // Enable External Interrupt 0
       sei();                      // Enable Global Interrupts
```

```
        while (1) {                 // Loop LED Toggle, Period of 735ms (60% High, 40% Low)
                _delay_ms(435);     // Delay Program by 435ms (Multiply '16' because 16MHz)
                PORTB &= ~LED;      // Turn OFF LED (High-to-Low)
                _delay_ms(290);     // Delay Program by 290ms (Multiply '16' because 16MHz)
                PORTB |= LED;       // Turn ON LED (Low-to-High)
        }
}

ISR (INT0_vect)     {               // External Interrupt 0
        PORTB ^= (1<<5);            // Toggle PB5 LED ON and OFF
        _delay_ms(1000);            // Toggle every 1000ms
}
```

4. SCHEMATICS

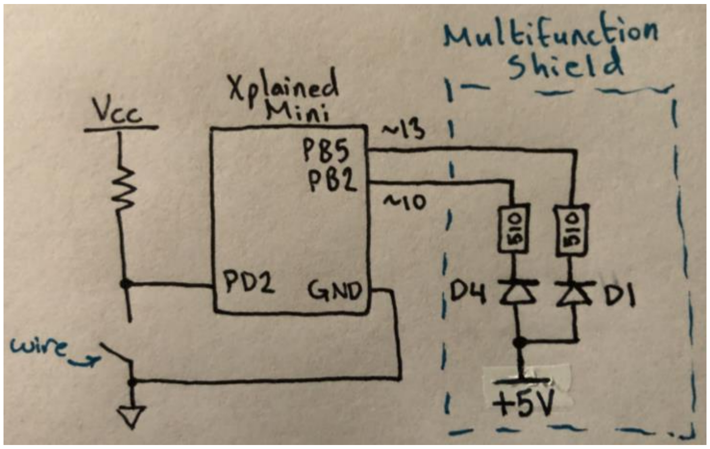

Figure 2 – Schematic Connections for including External Interrupt

5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Using a Logic Analyzer, we are able to observe that the C coding has a better response since the timer is more accurate that the Assembly Loop made. The following waveforms are the demonstrated C and Assembly code from Task 1 in DA2A except with an external interrupt now included:

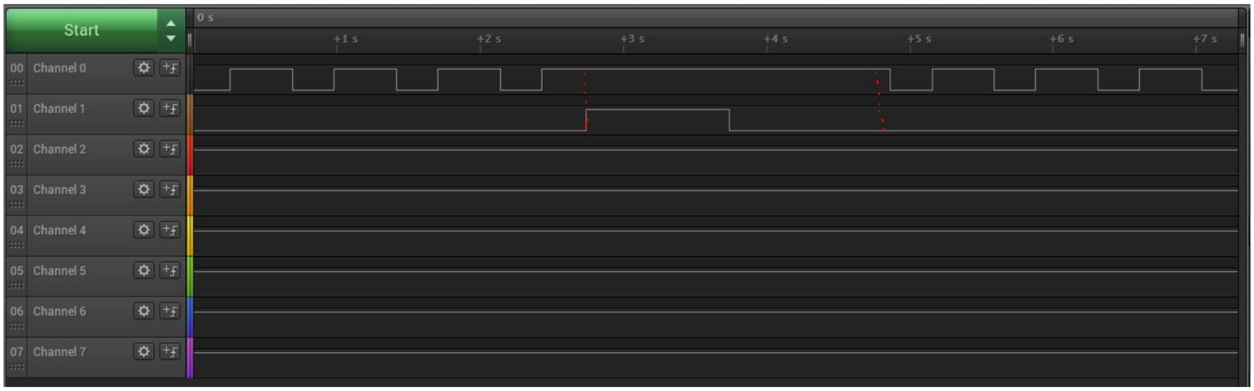

Figure 3a – Output Waveform of Task 2.A1/DA2A (Assembly Coding)

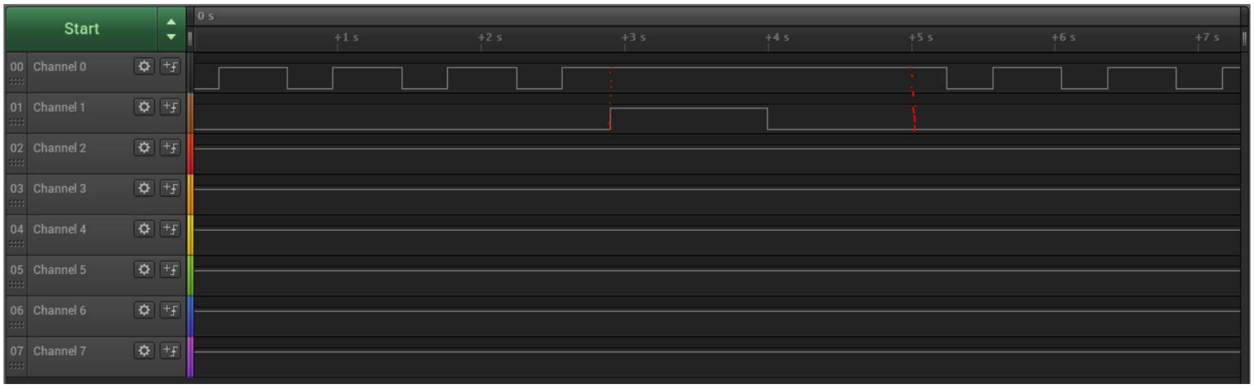

Figure 3b – Output Waveform of Task 2A.1/DA2B (C Coding)

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Each Demo utilizes the same set up which includes a Shield Attachment placed on top of the Xplained Mini PB and a red wire connector from PD2 ready to connect to Ground. This set up is shown in *Figure 2*:

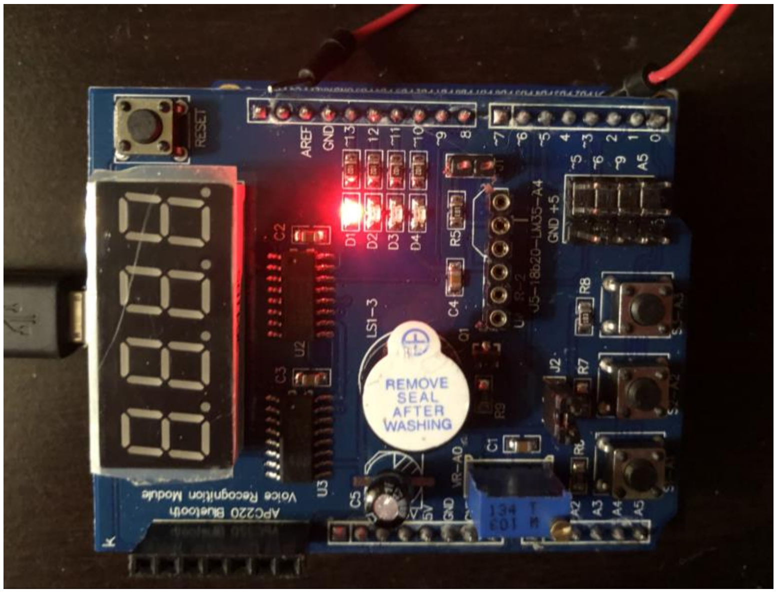

Figure 4 – Wire attached from PD2 (Xplained Mini Board)

7. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=gcYfJsiCccI

8. GITHUB LINK OF THIS DA

https://github.com/rockyg1995/ihswppdar/tree/master/DesignAssignments/DA2B

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".
Rocky Gonzalez