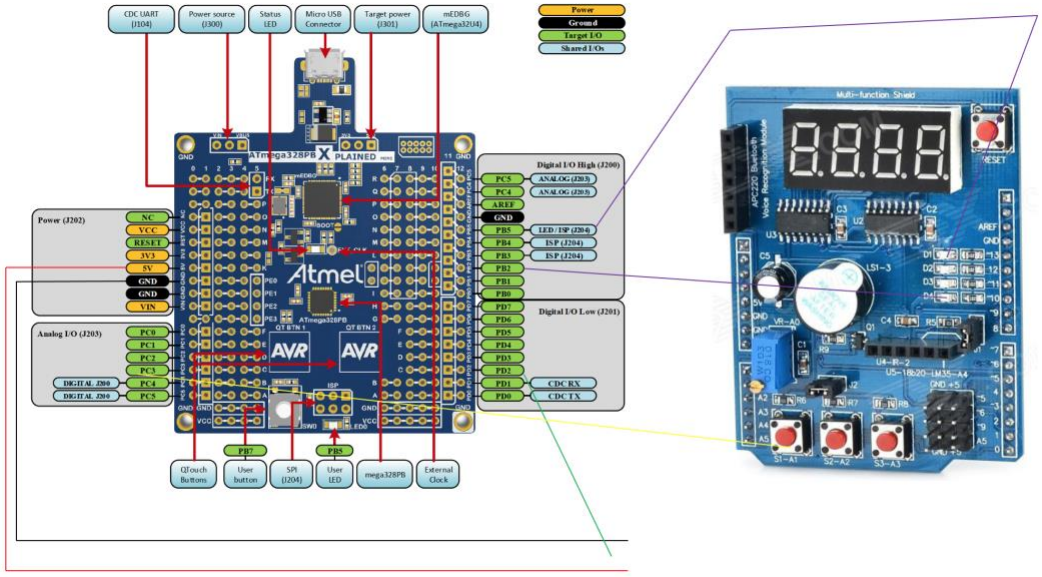
CPE301 – SPRING 2022

Design Assignment 2

Student Name: David Nakasone
Student #: 2001646072
Student Email: nakasd3@unlv.nevada.edu
Primary Github address: https://github.com/davenakasone
Directory: https://github.com/davenakasone/cpe301_David_Nakasone

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

List of Components used: Atmega 328pb and shield
Block diagram with pins used in the Atmega328P:



Using PB.5 as polling LED on PC.3, PB.2 as interrupt LED on PD.2 (INT0)

2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1:4

```
.org 0x2
    JMP int0_isr

.def rTemp = r16          ; general purpose helper register
.def rInner = r17         ; counter for inner loop
.def rInnerTune = r18     ; tunner for inner loop
.def rMiddle = r19        ; counter for middle loop
.def rMiddleTune = r20    ; tunner for middle loop
.def rOuter = r21         ; counter for outer loop
.def rOuterTune = r22     ; tunner for outerr loop
.def rLED_pol = r23       ; maintains state of LED for polling
.def rLED_int = r24       ; maintains state of LED for interrupt
.def rCalls = r25         ; counter for calling delay025

.equ LOOP_INNER = 128     ; lowest loop level, iterations to execute
```

```
Output
Show output from: Build

    [.cseg] 0x000000 0x0000ac    172      0    172    32768    0.5%
    [.dseg] 0x000100 0x000100      0      0      0     2048    0.0%
    [.eseg] 0x000000 0x000000      0      0      0     1024    0.0%
    Assembly complete, 0 errors, 0 warnings
  Done executing task "RunAssemblerTask".
Done building target "CoreBuild" in project "AssemblerApplication2.asmproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' !=
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\User
Done building target "Build" in project "AssemblerApplication2.asmproj".
Done building project "AssemblerApplication2.asmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

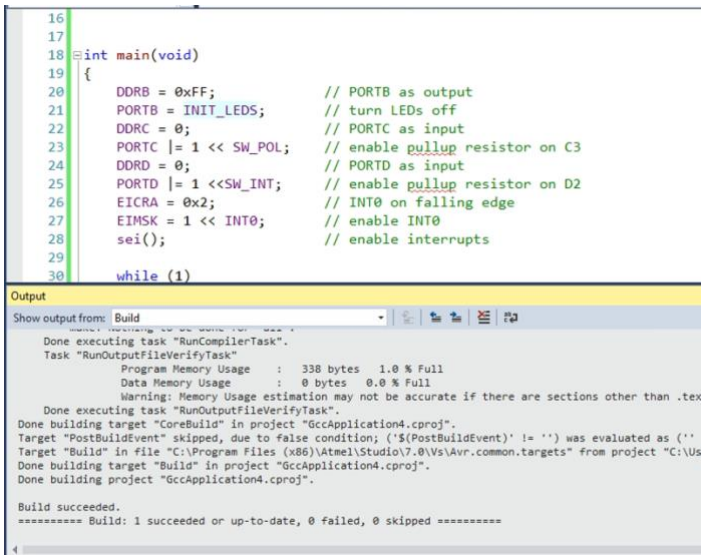
Task 1, ASM, successful build

```
; cpe301, s22, da2, David Nakasone
; f_cpu = 16MHz, each clock period is T_cpu 0.00625 us
; inner loop runs from 0:LOOP_INNER
; middle loop runs from 0:LOOP_MIDDLE
; outer loop runs from 0:LOOP_OUTER, making a delay of 0.25 seconds
; reach any time desired with another loop that tracks calls to "delay025"
; tunning to micro-second range is possible
; no clocks allowed

.include <m328pbdef.inc>
.org 0 ; reset point
   JMP main

.org 0x2
   JMP int0_isr

.def rTemp = r16          ; general purpose helper register
.def rInner = r17         ; counter for inner loop
.def rInnerTune = r18     ; tunner for inner loop
.def rMiddle = r19        ; counter for middle loop
.def rMiddleTune = r20    ; tunner for middle loop
.def rOuter = r21         ; counter for outer loop
.def rOuterTune = r22     ; tunner for outerr loop
.def rLED_pol = r23       ; maintains state of LED for polling
.def rLED_int = r24       ; maintains state of LED for interrupt
.def rCalls = r25         ; counter for calling delay025

.equ LOOP_INNER = 128     ; lowest loop level, iterations to execute
.equ TUNE_INNER = 128     ; tuning adjustment applied to inner loop
.equ LOOP_MIDDLE = 128    ; middle loop level, iterations to execute
.equ TUNE_MIDDLE = 128    ; tunning adjustment applied to outer loop
.equ LOOP_OUTER = 34      ; top loop level, iterations to achieve 0.25 sec delay
.equ TUNE_OUTER = 160     ; tunning adjustment applied to outter loop
.equ SPINZ = 255          ; extra NOPs
.equ SW_POL = 3           ; switch for polling on PC.3
.equ LED_POL = 2          ; LED for polling on PB.2
.equ CALLS_POL = 5        ; on polling, delay025 must be called 5 times to produce 1.25 s delay
```

```
.equ SW_INT = 2          ; switch for interrupt on PD.2
.equ LED_INT = 5         ; LED for interrupt on PB.5
.equ CALLS_INT = 2       ; on interrupt, delay025 must be called 2 times to produce 0.5 s delay
.equ LED_INIT = 0x3C     ; turns off LEDs on shield, revese biased

main:
   ; initialize SP
   LDI rTemp, LOW(RAMEND)
   OUT SPL, rTemp
   LDI rTemp, HIGH(RAMEND)
   OUT SPH, rTemp

   LDI rCalls, 1       ; initialize calling to at least one 250ms delay
   SBI DDRB, LED_INT   ; portB.5 as output for interrupt LED
   SBI DDRB, LED_POL   ; portB.2 as output for polling LED
   LDI rTemp, LED_INIT
   OUT PORTB, rTemp    ; start LEDs off, reversed biased
   LDI rTemp, 0
   OUT DDRC, rTemp     ; portC as input
   OUT DDRD, rTemp     ; portD as input
   SBI PORTC, SW_POL   ; portC.3 pullup resistor enabled
   SBI PORTD, SW_INT   ; portD.2 pullup resistor enabled

   LDI rTemp, 0x2
   STS EICRA, rTemp         ; INT0 is falling edge triggered
   LDI rTemp, (1 << INT0)
   OUT EIMSK, rTemp         ; enable INTO
   SEI                      ; enable interrupts


main_loop:
   IN rTemp, PINC       ; get value of PINC, using polling
   SBIS PINC, SW_POL    ; (button pushed) ? start delay : keep polling
   RCALL poll_hit       ; starts 1.25 second delay by calling delay025 x 5
   SBI PORTB, LED_POL   ; ensure LED is not bouncing
   RJMP main_loop       ; poll again


poll_hit:
```

```
    LDI rCalls, CALLS_POL    ; prepare # of calls to delay025
    CBI PORTB, LED_POL       ; turn LED on
    RCALL call_delay025      ; wait 1.25 seconds
    SBI PORTB, LED_POL       ; turn LED off
    RET                      ; returning for more polling



call_delay025:
    RCALL delay025       ; make one 250ms delay
    DEC rCalls
    BRNE call_delay025   ; (calls complete) ? returning : keep calling
    LDI rCalls, 1        ; initialize rCalls
    RET                  ; control to caller
delay025:
    LDI rInner, LOOP_INNER          ; initialize inner loop
    LDI rInnerTune, TUNE_INNER      ; initialize tuning parameter for inner loop
    LDI rMiddle, LOOP_MIDDLE        ; initialize middle loop
    LDI rMiddleTune, TUNE_MIDDLE    ; initialize tuning parameter for middle loop
    LDI rOuter, LOOP_OUTER          ; initialize outer loop
    LDI rOuterTune, TUNE_OUTER      ; initialize tuning parameter for outer loop
    LDI rTemp, SPINZ                ; initialize single NOPs
outer_loop:
    RCALL outer_loop_tune    ; apply NOPs specified by TUNE_OUTER
    RCALL middle_loop        ; iterate exterior loop C through A and B
    DEC rOuter
    BRNE outer_loop          ; (LOOP_OUTER repititions complete) ? sub-routine complete : goto outter_loop
    RCALL spin_isolated      ; execute single NOPs
    RET                      ; returning out of this sub-routine
outer_loop_tune:
    NOP                          ; kill a cycle
    DEC rOuterTune
    BRNE outer_loop_tune         ; (TUNE_OUTER time cycles executed) ? spins complete : keep spinning
    LDI rOuterTune, TUNE_OUTER   ; reset
    RET                          ; returns to outter_loop after spins are complete
middle_loop:
    RCALL middle_loop_tune      ; apply NOPs specified by TUNE_MIDDLE
    RCALL inner_loop            ; complete one iteration of interior loop
    DEC rMiddle
```

```
    BRNE middle_loop            ; (LOOP_MIDDLE repetitions occured) ? break : goto middle_loop
    LDI rMiddle, LOOP_MIDDLE    ; reset
    RET                         ; returning to loopC
middle_loop_tune:
    NOP                         ; kill a cycle
    DEC rMiddleTune
    BRNE middle_loop_tune       ; (TUNE_MIDDLE time cycles executed) ? spins complete : keep spinning
    LDI rMiddleTune, TUNE_MIDDLE    ; reset
    RET                         ; returns to middle_loop after spins are complete
inner_loop:
    DEC rInner
    BRNE inner_loop         ; (LOOP_INNER repititions occured) ? break : goto inner_loop
    LDI rInner, LOOP_INNER    ; reset
    RET                     ; returning to loopB
inner_loop_tune:
    NOP                     ; kill a cycle
    DEC rInnerTune
    BRNE inner_loop_tune        ; (TUNE_INNER time cycles executed) ? spins complete : keep spinning
    LDI rInnerTune, TUNE_INNER    ; reset
    RET                         ; returns to inner_loop after spins are complete
spin_isolated:
    DEC rTemp
    BRNE spin_isolated          ; (SPINZ single cycles) ? task complete : NOP more
    RET



int0_isr:
    LDI rCalls, CALLS_INT    ; prepare # of calls to delay025
    CBI PORTB, LED_INT       ; turn LED on
    RCALL call_delay025      ; wait 0.5 seconds
    SBI PORTB, LED_INT       ; turn LED off
    RETI



program_complete:
    RJMP program_complete    ; catch



;;;;~~~~END>  main.asm
```



```
16
17
18  int main(void)
19  {
20      DDRB = 0xFF;              // PORTB as output
21      PORTB = INIT_LEDS;        // turn LEDs off
22      DDRC = 0;                 // PORTC as input
23      PORTC |= 1 << SW_POL;     // enable pullup resistor on C3
24      DDRD = 0;                 // PORTD as input
25      PORTD |= 1 <<SW_INT;      // enable pullup resistor on D2
26      EICRA = 0x2;              // INT0 on falling edge
27      EIMSK = 1 << INT0;        // enable INT0
28      sei();                    // enable interrupts
29
30      while (1)
```

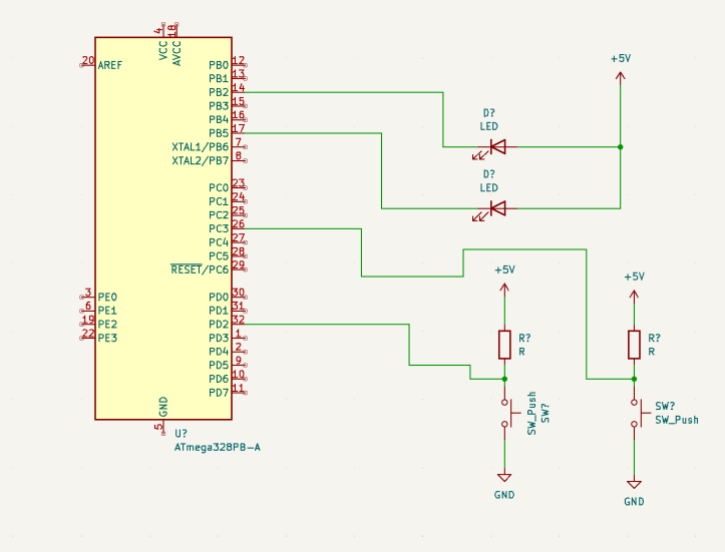
```
Output
Show output from: Build
    Done executing task "RunCompilerTask".
    Task "RunOutputFileVerifyTask"
            Program Memory Usage     :   338 bytes   1.0 % Full
            Data Memory Usage        :   0 bytes   0.0 % Full
            Warning: Memory Usage estimation may not be accurate if there are sections other than .tex
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "GccApplication4.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Us
Done building target "Build" in project "GccApplication4.cproj".
Done building project "GccApplication4.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Task 1, C, successful build

```
/*
   cpe301, da2, all tasks
 */

#define F_CPU 16000000UL    // 16 MHz for 328pb

#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>

#define SW_POL 3          // switch on C2 triggers poll
#define SW_INT 2          // switch on D2 triggers interrupt
#define LED_POL 2         // LED on B2 indicates poll detected
#define LED_INT 5         // LED on B5 indicates interrupt detected
#define INIT_LEDS 0x3C    // turns reversed biased LEDs on shield off

int main(void)
{
   DDRB = 0xFF;           // PORTB as output
   PORTB = INIT_LEDS;     // turn LEDs off
   DDRC = 0;              // PORTC as input
   PORTC |= 1 << SW_POL;  // enable pullup resistor on C3
   DDRD = 0;              // PORTD as input
   PORTD |= 1 <<SW_INT;   // enable pullup resistor on D2
   EICRA = 0x2;           // INT0 on falling edge
```

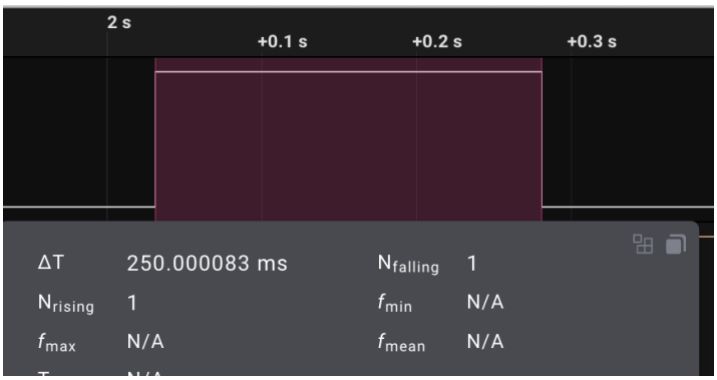
```
    EIMSK = 1 << INT0;       // enable INT0
    sei();                   // enable interrupts

    while (1)
    {

        if (!(PINC & 0x4)) // if C3 was pressed, poll triggered
        {
            PORTB &= ~(1 << LED_POL);    // turn on B2 LED
            _delay_ms(250*5);            // wait 1.25 seconds
            PORTB = INIT_LEDS;           // turn all the LEDS off
        }
    }
}


////~~~~


ISR (INT0_vect)    // isr for external interrupt 0
{
    PORTB &= ~(1 << LED_INT);    // turn on B5 LED
    _delay_ms(250*2);            // wait 0.5 seconds
    PORTB = INIT_LEDS;           // turn all the LEDS off
}


////////~~~~~~~END>  main.c
```

3. SCHEMATICS

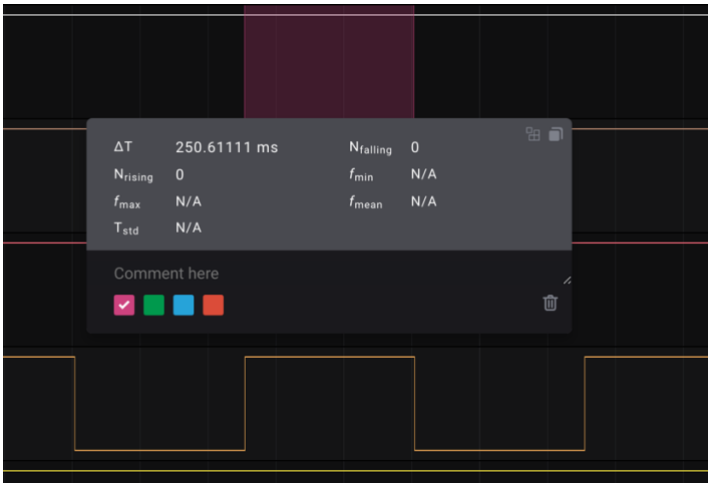


4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

The 250 ms delay determines the 1.25 and 0.5 second delays.
It was adjusted using assembly:

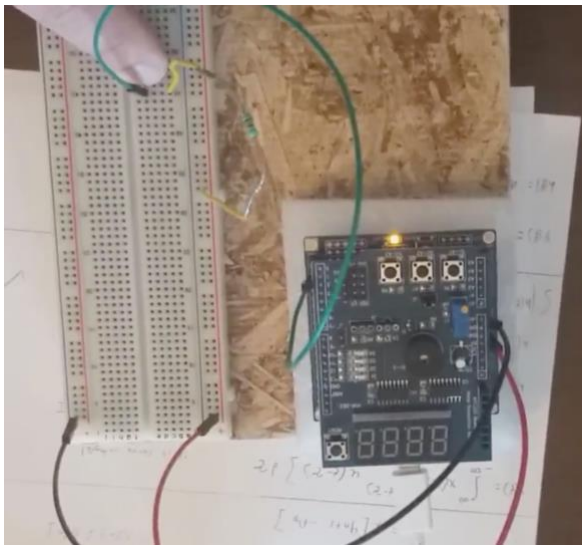


The 250 ms delay in C:

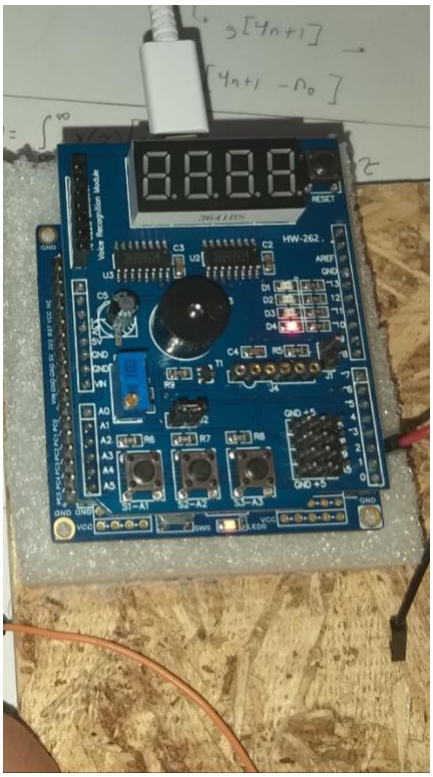


5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

The circuit using interrupt, detected by changing connection:



The circuit using polling (push-button):



6. VIDEO LINKS OF EACH DEMO

task1, 250 ms delay: https://youtu.be/0c75GUvmLo0
task2, LED activated by polling: https://youtu.be/q-BGHsYzP0Q
task3, LED activated by interrupt: https://youtu.be/cC0MpXEx9PQ
task4, logic analyzer and timing: https://youtu.be/vdnolojUFSs

7. GITHUB LINK OF THIS DA

https://github.com/davenakasone/cpe301_David_Nakasone/tree/main/Design_Assignments/DA2

Student Academic Misconduct Policy

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

"*This assignment submission is my own, original work*".
David Nakasone