Lab 3

Michael Arboleda
Lab Section: 7F34

June 13, 2017

b. Answers to all pre-lab questions

1) How many TC0 channels are necessary to control all three LEDs in the uPADs RGB LED?
ANS: . .

c. Problems Encountered

.

d. Future Work/Applications

.

e. Schematics

N/A

g. Pseudocode/Flowcharts

Pseudocode for lab1b.asm:

```
MAIN:

    * Equate numbers
    * Set registers to hold constants

    WHILE(true){
        * Grab data from buttons
        * AND Data with #4 to isolate 3rd bit
        IF(S1 is pressed){
            * run subroutine display
        }
        ELSE-IF(AND #8 to isolate4th bit
          then check if S2 is pressed){
              * run subroutine fetch_store
        }
    }
END

SUBROUTINE DISPLAY:
    * Set PortC to be able write
    * Load LED data from memory
    * Display LEDs
    * Return to program

SUBROUTINE FETCH_STORE:
    * Set Port A to read
    * Read switches
    * Store data in memory
    * Return to program
```

Pseudocode for lab1c.asm:

```
MAIN:

    * Equate numbers
    * Set registers to hold constants

    WHILE(true){
        * Set to write
        * Toggle LEDs off
        * Call DELAYx10ms subroutine
        * Toggle single LED on
        * Call DELAYx10ms subroutine
    }
END


SUBROUTINE DELAY_10ms
    * Load number of iterations for loop as counter
    FOR(
        Counter ;
        Counter does not equal 0 ;
        Decrement interation by 1
    )
    {No Body}
    * Return to program


SUBROUTINE DELAYx10ms
    * Parameter: multiplier to 10ms
    FOR(
        X := 0;
        X does not equal multiplier;
        Increment X by 1
    )
    {
        * Call DELAY_10ms
    }
    * Return to program
```

Pseudocode for lab1d.asm:

```
MAIN:

    * Equate numbers
    * Set registers to hold constants

    WHILE(true){
        * Set pointer to KITT Table
        FOR(
            x := 0;
            x is not equal to table size;
            increment x by 1
        )
        {
            * LOAD table data
            * Increment z pointer

            * Turn LED on and Delay
            * Turn LED off and Delay
            * Turn LED on and Delay
            * Turn LED off and Delay
            * Turn LED on and Delay
        }
    }

END

SUBROUTINE DELAY_10ms
    * Load number of iterations for loop as counter
    FOR(
        Counter     ;
        Counter does not equal 0 ;
        Decrement interation by 1
        )
    {No Body}
    * Return to program
```

h. Program Code

Code for lab1b.asm:

```
; Lab 1 Part B
; Name:          Michael Arboleda
; Section:       7F34
; TA Name:       Wesley Piard
; Description: Fetchs data on a switches and stores it
;                          in memory. Displays value in memory
;
; lab1b.asm
; Created: 5/21/2017 3:34:53 PM

.include ”ATxmega128A1Udef.inc”

;Stack Pointer location
.equ stack_init = 0x2FFF          ;initialize stack pointer
                                  ;(between 0x2000 & 0x3FFF)

; address equates
.equ adrs_data_bank = 0x3744                ; Address to store switch data
.equ adrs_subrout_display = 0x5000          ; Address for display subroutine
.equ adrs_subrout_fetchStore = 0x6000       ; Address for reading
                                            ; and storing subroutine

; Constant equates
.equ button_PF2 = 0b00000100      ; Configuration for S1
.equ button_PF3 = 0b00001000      ; Configuration for S2

.ORG 0x0000                       ;Code starts running from address 0x0000.
        rjmp MAIN                 ;Relative jump to start of program.

.ORG 0x0100                 ;Start program at 0x0100 so we don't overwrite
                            ;  vectors that are at 0x0000-0x00FD
MAIN:
        ldi r16, 0x00             ; LOAD r16 with 0
        ldi r17, button_PF2       ; Set bitmask 0000 0100 in r17
        ldi r18, button_PF3       ; Set bitmask 0000 1000 in r18


        ;Set Stack Pointer
        ldi YL, low(stack_init)  ; LOAD lower part of Stack Pointer
        out CPU_SPL, YL           ; initialize low byte of stack pointer
        ldi YL, high(stack_init); LOAD higher part of Stack Pointer
        out CPU_SPH, YL           ; initialize high byte of stack pointer
        ldi r17, button_PF2             ; Set bitmask 0000 0100 in r17
        ldi r18, button_PF3             ; Set bitmask 0000 1000 in r18

WHILE:
        sts PORTF_DIRCLR, r16           ; Set to write
        lds r19, PORTF_IN               ; LOAD Port F into r19
        mov r20, r19                    ; Copy Port F data into r20
        and r19, r17                    ; Isolate bit 2 in r19
;IF
        cp r19, r17             ; COMPARE r19 to r17 bitmask
        breq ELSEIF             ; BRANCH if not equal to each other

        ;IF-BODY
        call Fetch_Store                ; CALL subroutine DISPLAY
        rjmp ENDIF
;ELSE IF
ELSEIF:
        and r20, r18            ; Isolate bit 3 in r20
        cp r20, r18             ; COMPARE r20 to r18 bitmask
        breq ENDIF              ; BRANCH if not equal to each other

        ;ELSE IF-BODY
        call Display            ; CALL subroutine Fetch_Store

ENDIF:
        rjmp WHILE              ; Jump/Restart program


;————————————————————————————————————————————————————————————————————;
;                                                SUBROUTINES
;
;————————————————————————————————————————————————————————————————————;
;*******************SUBROUTINES*************************************
; Subroutine Name: Display
;
; Inputs: No direct input (from stack)
; Outputs: No direct output
; Affected: R21
.def r21_display = r21
.ORG adrs_subrout_display; Set subroutine to start at specified address
Display:
        push r21_display                ; PUSH r21 to stack
        ldi r21_display, 0xFF           ; LOAD FF to r21
        sts PORTC_DIRSET, r21_display   ; Set to write
        lds r21_display, adrs_data_bank; LOAD the data from memory
                                        ;to r21
        sts PORTC_OUT, r21_display      ; Display LED configuration
                                        ;based on r21 data
        pop r21_display                 ; POP r21 from stack
        ret                             ; return from subroutine
.undef r21_display                      ; undefine r16_by_value


;*******************SUBROUTINES************************************
; Subroutine Name: Fetch_Store
;
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
.def r21_fetch_store = r21
.ORG adrs_subrout_fetchStore; Set subroutine to start at specified address
Fetch_Store:
        push r21_fetch_store            ; PUSH r21 to stack
        ldi r21_fetch_store, 0x00       ; LOAD 0 to r21
        sts PORTA_DIRCLR, r21_fetch_store       ; Set to read
        lds r21_fetch_store, PORTA_IN   ; READ switches and store in reg
        sts adrs_data_bank, r21_fetch_store     ; STORE values in 0x3744
        pop r21_fetch_store             ; POP r21 from stack
        ret                                                             ;
.undef r21_fetch_store                  ; undefine r21_fetch_store
```

Code for lab1c.asm:

```
; Lab 1 Part C
; Name:         Michael Arboleda
; Section:      7F34
; TA Name:      Wesley Piard
; Description: Toggles LED based on time
;
; lab1c.asm
; Created: 5/23/2017 2:29:17 AM

.include "ATxmega128A1Udef.inc"

;Stack Pointer location
.equ stack_init = 0x2FFF        ;initialize stack pointer
                                ;(between 0x2000 & 0x3FFF)

; Constant equates
.equ button_PF2 = 0b11111101; 1 bit on
.equ multi = 1                          ; Multiplier for 10ms

.ORG 0x0000                     ;Code starts running from address 0x0000.
```

```
        rjmp MAIN               ;Relative jump to start of program.

.ORG 0x0100             ;Start program at 0x0100 so we don't overwrite
                        ;   vectors that are at 0x0000-0x00FD
MAIN:

        ;Load constants
        ldi r16, 0xFF                   ; LOAD r16 with FF
        ldi r17, 0x00                   ; LOAD r17 with 0
        ldi r18, button_PF2             ; LOAD r18 with LED data
        ldi r20, multi                  ; LOAD r19 with multiplier

;While loop
WHILE:
        ;Display Toggle off
        sts PORTC_DIRSET, r16           ; Set to write
        sts PORTC_OUT, r16      ; LED data to be displayed (LEDS off)

        call DELAYx10ms                 ; Call Delay X subroutine

        ;Display
        sts PORTC_OUT, r18      ; LED data to be displayed (1 LED on)

        call DELAYx10ms                 ; Call Delay X subroutine
        rjmp WHILE                      ; Restart loop to rerun program



;——————————————————————————————————————————————————————————————————————;
;                                       SUBROUTINES
;
;——————————————————————————————————————————————————————————————————————;
;*******************SUBROUTINES*************************************
; Subroutine Name: DELAY_10ms
;       Delays by 10 ms
; Inputs: No direct input (from stack)
; Outputs: No direct output
; Affected: No register affected

.def r1_delay = r24
.def r2_delay = r25
DELAY_10ms:
        push r1_delay                   ; Push r24
        push r2_delay                   ; Push r25
```

```
;FOR_INT none
        ldi r1_delay, 0x07             ; LOAD lower word with 0x07
        ldi r2_delay, 0x0B             ; LOAD higher word with 0x0B


FOR_IF:
        cpi r2_delay, 0x00             ; COMPARE higher word with 0
        brne  BODY_LOOP                ; if not 0, jump to loop body
        cpi r1_delay, 0x00             ; COMPARE lower word with 0
        brne  BODY_LOOP                ; if not 0, jump to loop body
        ; Since counter is 0, end loop
        rjmp END_FOR                   ; JUMP to end of loop

BODY_LOOP:
        sbiw r2_delay:r1_delay, 1      ; Subtract 1 from r25:r24
        rjmp FOR_IF                    ; JUMP back to condition check

END_FOR:

        pop r2_delay                           ; POP r21 from stack
        pop r1_delay                           ; POP r21 from stack
        ret                                    ; return from subroutine
.undef r2_delay
.undef r1_delay


;-----------------------------------------------------------------------;
;*******************SUBROUTINES************************************
; Subroutine Name: DELAYx10ms,
;       Delays by 10ms times a multiplier
; Inputs: r20
; Outputs: No direct outputs
; Affected: r19, r27

.def multiplier = r20
DELAYx10ms:
        push multiplier                ; PUSH r20
        ldi r19, 1                     ; LOAD r19 with 1

;FOR_INT
        ldi r27, 0;                    ; LOAD r27 with 0

DxFOR_IF:
        cp r27, multiplier       ; COMPARE r27 and the Multiplier
        breq DxEND_FOR           ; If r27 = multiplier then end loop

```

```
;FOR BODY
        call DELAY_10ms          ; Call DELAY_10ms to delay by 10ms
;Update
        add r27, r19             ; increment r27 by 1
        rjmp DxFOR_IF            ; jump to conditional check
DxEND_FOR:
        pop multiplier
        ret
.undef multiplier
```

Code for lab1d.asm:

```
; Lab 1 Part D
; Name:         Michael Arboleda
; Section:      7F34
; TA Name:      Wesley Piard
; Description: Displays LED in KITT pattern
;
; lab1d.asm
; Created: 5/23/2017 11:09:32 PM

.include ”ATxmega128A1Udef.inc”

.equ table_size = 9             ;Size of table

.ORG 0x0000                     ;Code starts running from address 0x0000.
        rjmp MAIN               ;Relative jump to start of program.

.CSEG                                   ;Code segment start
        .ORG 0xF000                     ; Start table address

KITT_Table:
        .db 0b01111111, 0b00111111, 0b10011111, 0b11001111,\
            0b11100111, 0b11110011, 0b11111001, 0b11111100,\
            0b11111110 ; Table Values
.DSEG                                   ;Code segment stop

.CSEG
.ORG 0x0100                 ;Start program at 0x0100 so we don’t overwrite
                            ;  vectors that are at 0x0000-0x00FD
MAIN:

; Set up constant regs
ldi R16, 1                              ; LOAD r16 with 1
ldi r18, 0xFF                           ; LOAD r18 with 0xFF

WHILE:
```

```
; Set up z?register
sts CPU_RAMPZ, R16                     ;STORING Extended Address
ldi ZL, low( KITT_Table << 1)          ;LOAD adrs to read from?low bits
ldi ZH, high ( KITT_Table << 1)        ;LOAD adrs to read from?high bits

; Set up regs
ldi r17, 0                                    ; LOAD r17 with 0

; Set leds to write
sts PORTC_DIRSET, r18                  ; Set to write

FOR_IF:
        cpi r17, table_size      ; COMPARE r17 and size of data table
        breq END_FOR             ; if r17 = size, end loop
;FOR BODY
        ;Load LED data
        elpm r19,  z+            ; LOAD From Table, increment z pointer

        ; LED Blinking
        sts PORTC_OUT, r19             ; Display LEDs (on)
        call DELAY_10ms                ; Delay by 10ms
        sts PORTC_OUT, r18             ; Display LEDs (off)
        call DELAY_10ms                ; Delay by 10ms
        sts PORTC_OUT, r19             ; Display LEDs (on)
        call DELAY_10ms                ; Delay by 10ms
        sts PORTC_OUT, r18             ; Display LEDs (off)
        call DELAY_10ms                ; Delay by 10ms
        sts PORTC_OUT, r19             ; Display LEDs (on)
        call DELAY_10ms                ; Delay by 10ms
;UPDATE
        add r17, r16                   ; ADD 1 and r17 (r17 = r17 + 1)
        rjmp FOR_IF                    ; JUMP to conditional statement
END_FOR:
        rjmp WHILE                     ; JUMP to restart program


;------------------------------------------------------------------------------;
;                                                         SUBROUTINES
;
;------------------------------------------------------------------------------;
;*******************SUBROUTINES*************************************
; Subroutine Name: DELAY_10ms
;       Delays by 10 ms
; Inputs: No direct input (from stack)
; Outputs: No direct output
; Affected: No register affected
```

```
.def r1_delay = r24
.def r2_delay = r25
DELAY_10ms:
        push r1_delay                   ; Push r24
        push r2_delay                   ; Push r25



;FOR_INT none
        ldi r1_delay, 0x07              ; LOAD lower word with 0x07
        ldi r2_delay, 0x0B              ; LOAD higher word with 0x0B



DFOR_IF:
        cpi r2_delay, 0x00              ; COMPARE higher word with 0
        brne  BODY_LOOP                 ; if not 0, jump to loop body
        cpi r1_delay, 0x00              ; COMPARE lower word with 0
        brne  BODY_LOOP                 ; if not 0, jump to loop body
        ; Since counter is 0, end loop
        rjmp DEND_FOR                   ; JUMP to end of loop

BODY_LOOP:
        sbiw r2_delay:r1_delay, 1       ; Subtract 1 from r25:r24
        rjmp DFOR_IF                    ; JUMP back to condition check

DEND_FOR:

        pop r2_delay                            ; POP r21 from stack
        pop r1_delay                            ; POP r21 from stack
        ret                                     ; return from subroutine
.undef r2_delay
.undef r1_delay
```

i. Appendix