Lab 2

Michael Arboleda
Lab Section: 7F34

June 7, 2017

b. Answers to all pre-lab questions

1) What is the default clock frequency of the XMEGA?

ANS: The default clock frequency of the XMEGA is 2MHz .

2) What would you need to do to run the XMEGA at 16 MHz?

ANS: To run the XMEGA at 16 MHz would need to set the clock to 32MHz and then divide by 2 using a prescaler .

3) The XMEGA is rated to run at frequencies up to 32 MHz, but some peripherals run up to either 2x or 4x of this frequency. How can the XMEGA be configured to run at 32 MHz, the 2x peripheral clock at 64 MHz, and the 4x peripheral clock at 128 MHz without using any external clock sources?

ANS: One way to get values higher than 32MHz is to use PPL. PPLCTRL, the PPL control register allows to set a multiplication factor. This includes values from time 1 to times 31. Thus if 64MHz and 128MHz are need, the PPL would have to be set to time 2 and times 4 respectively.

4) Why are timers useful?

ANS: Timers are useful because they are more accurate than software delays . They work in tandem with the processors clock. The timers in ATxMEGA provide accurate programming execution timing, thus making a programs timing accurate, which is important to time-sensitive programs.

c. Problems Encountered

I could not get my clock frequency to output. This was because I did not use a dirset to make the port an output

d. Future Work/Applications

After finishing the lab it is clear that there is much that can be done. I am interested in making a program that cycles through all the colors that the LED can make.

e. Schematics

N/A

g. Pseudocode/Flowcharts

Pseudocode for lab2a.asm:

```
MAIN:

* Equate numbers
* Set registers to hold constants
* Call Change_CLK_4HZ subroutine

WHILE(true){}
END


SUBROUTINE Change_CLK_4HZ
    * Enable the new oscillator

WHILE(32Hkz OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
    * Write the IOREG signature to the CPU_CCP reg
    * Set Prescalar
    * Set clock output to port c
* Return to program
```

Pseudocode for lab2b.asm:

```
MAIN:

* Equate numbers
* Set registers to hold constants
* Set Port F to write
* Set TOP value for counter
* Set Prescalar for counter

WHILE(TRUE){
    * Output counter value
}
```

Pseudocode for lab2c.asm:

```
MAIN:

* Equate numbers
* Set registers to hold constants

* Call Change_CLK_32HZ subroutine
* Set TOP value for counter
* Set Prescalar for counter
* Set Port D to write

WHILE(TRUE){
    if(Counter is 0){
        * Turn all lights on
    }
    if(Counter equals red max time){
        * Turn Red LED Off
    }
    if(Counter equals green max time){
        * Turn Green LED Off
    }
    if(Counter equals blue max time){
        * Turn Blue LED Off
    }
}

END


SUBROUTINE Change_CLK_4HZ
   * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
    * Return to program
```

h. Program Code

Code for lab2a.asm:

```
; Lab 2 Part A
; Name:          Michael Arboleda
; Section:       7F34
; TA Name:       Wesley Piard
; Description: Changes clock frequency of up
;
; lab2a.asm
; Created: 5/21/2017 3:34:53 PM

.include "ATxmega128A1Udef.inc"

; address equates

; Constant equates
.equ new_clock_freq = 0b00000010
.equ div_value = 0b00001100
.equ CLKEVOUT_config = 0b00000001

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

MAIN:
        call Change_CLK_4HZ
Never_End:
        rjmp Never_End


;*********************SUBROUTINES************************************
; Subroutine Name: Change_CLK_4HZ
;
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: R17, R18, R19, R20
Change_CLK_4HZ:

; Enable the new oscillator
ldi R16, new_clock_freq         ; Load R16 with the clk-freq config 0x02
sts OSC_CTRL, r16               ; Set the clk config

;Wait for the right flag to be set in the OSC_STATUS reg

; While flag is not set
While_32_flag:
```

```
        lds R17, OSC_STATUS             ; Load Status Flag
        and R17, R16                    ; Bit-mask with 00000010
        cp R17, R16                     ; Compare Mask and Value
        brne While_32_flag              ; Restart loop if flag not set

; Write the IOREG signature to the CPU_CCP reg
        ldi R17, CCP_IOREG_gc    ; Load IOREG into R17
        sts CPU_CCP, R17         ; Store IOREG into CPU CCP

;Select the new clock source in the CLK_CTRL reg
        ldi R17, CLK_SCLKSEL_RC32M_gc; load 32 MHz internal osc config
        sts CLK_CTRL, R17               ; Store config in clk control

;Set Prescalars
        ; Write the IOREG signature to the CPU_CCP reg
        ldi R17, CCP_IOREG_gc    ; Load IOREG into R17
        ldi R18, div_value       ; Load divide config into r18
        sts CPU_CCP, R17         ; Store IOREG into CPU_CPP

        ; Set Prescalar
        sts CLK_PSCTRL, R18             ; Store Prescaler config

;Set clock to port c
        ldi R20, 0xFF
        sts PORTC_DIRSET, R20

        ldi R19, CLKEVOUT_config        ; Set CLKEVOUT config
        sts PORTCFG_CLKEVOUT, R19       ; MAke clock output
        ret
```

Code for lab2b.asm:

```
; Lab 2 Part B
; Name:          Michael Arboleda
; Section:       7F34
; TA Name:       Wesley Piard
; Description: Test Timer/Counter on uPad
;
; lab2b.asm
; Created: 6/6/2017 4:56:30 AM

.include "ATxmega128A1Udef.inc"

; address equates

; Constant equates
.equ clk_div = 0b00000111

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

MAIN:
        ldi R16, 0xFF           ; LOAD 0xFF into R16
        ldi R17, 0x00           ; LOAD 0x00 into R17
        ldi R18, clk_div        ; LOAD Clk prescaler into R18

        sts PORTF_DIRSET, R16   ; Set Port F to write

        sts TCF0_PER, R16       ; Set Lower Bits of TOP
        sts (TCF0_PER + 1), R17 ; Set Higher Bits of TOP

        sts TCF0_CTRLA, R18     ; Set Prescalar for Counter

Never_End:
        lds R19, TCF0_CNT; LOAD the data from memory
                                                    ;to r19
        sts PORTF_OUT,  R19     ; Output counter value

        rjmp Never_End          ; Jump to restart output loop
```

Code for lab2c.asm:

```
; Lab 2 Part C
; Name:          Michael Arboleda
; Section:       7F34
; TA Name:       Wesley Piard
; Description: Display Different colors on uPad LED
;
; lab2c.asm
; Created: 6/7/2017 3:36:05 AM

.include "ATxmega128A1Udef.inc"

; address equates

; Constant equates
.equ new_clock_freq = 0b00000010
.equ clk_div = 0b00000110
.equ counter_check_Red = 0x00
.equ counter_check_Green = 0xFF
.equ counter_check_Blue = 0xFF
        ; COLORS
.equ BIT4 = 0x10
.equ RED = ~(BIT4)
.equ RED_OFF = 0b00010000
.equ BIT5 = 0x20
.equ GREEN = ~(BIT5)
.equ GREEN_OFF = 0b00100000
.equ BIT6 = 0x40
.equ BLUE = ~(BIT6)
.equ BLUE_OFF = 0b01000000
.equ BIT456 = 0x70
.equ WHITE = ~(BIT456)


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

MAIN:
        ; Run at 32MHz
        call Change_CLK_32HZ    ; Change Clk to 32MHz

        ; Set up Counter
        ldi R16, 0xFF           ; LOAD)0xFF into R16
        ldi R17, 0x00           ; LOAD 0x00 into R17
        ldi R18, clk_div        ; LOAD Clk prescaler into R18
```

```
        sts TCF0_PER, R16               ; Set Lower Bits of TOP
        sts (TCF0_PER + 1), R17         ; Set Higher Bits of TOP

        sts TCF0_CTRLA, R18             ; Set Prescalar for Counter

        ; Set PORT D for output
        ldi R16, BIT456 ;load a four bit value (PORTD is only four bits)
        sts PORTD_DIRSET, R16 ;set all the GPIO's in the four bit
                              ; PORTD as outputs

While_Loop:
        lds R19, TCF0_CNT; LOAD the counter value

        ; If 0, turn on
        cpi R19, 0x00           ; Compare counter value with 0
        brne IF_RED             ; If not 0, jmp to red check
        ; Turn light on
        ldi R18, WHITE          ; LOAD white LED config
        sts PORTD_OUT, R18      ; Output LED config

IF_RED:
        cpi r19, counter_check_Red; Compare counter to red max time value
        brne IF_GREEN           ; If not equal, jump to green check
        ; Turn RED LED off
        ldi R20, RED_OFF        ; Load Red_OFF config
        lds R21, PORTD_OUT      ; Load Port D values
        or R21, R20             ; OR Red_OFF config with Port D values
        sts PORTD_OUT, R21      ; Output LED config

IF_GREEN:
        cpi r19, counter_check_Green; Compare counter to green max
                              ;time value
        brne IF_BLUE            ; If not equal, jump to Blue check
        ; Turn GREEN LED off
        ldi R20, GREEN_OFF      ; Load Green_OFF config
        lds R21, PORTD_OUT      ; Load Port D values
        or R21, R20             ; OR Green_OFF config with Port D values
        sts PORTD_OUT, R21      ; Output LED config

IF_BLUE:
        cpi r19, counter_check_Blue
        brne END_IF             ; If not equal, jump to end of loop
        ; Turn BLUE LED off
        ldi R20, BLUE_OFF       ; Load Blue_OFF config
        lds R21, PORTD_OUT      ; Load Port D values
```

```
        or R21, R20      ; OR Blue_OFF config with Port D values
        sts PORTD_OUT, R21       ; Output LED config

END_IF:
        rjmp While_Loop                  ; Jump to restart loop


;*********************SUBROUTINES************************************
; Subroutine Name: Change_CLK_32HZ
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
Change_CLK_32HZ:
        ;Push Values
        push R17 ; PUSH r17 to stack
        push R18 ; PUSH r18 to stack
        push R19 ; PUSH r19 to stack
        push R20 ; PUSH r20 to stack

; Enable the new oscillator
ldi R16, new_clock_freq          ; Load R16 with the clk-freq config 0x02
sts OSC_CTRL, r16                        ; Set the clk config

;Wait for the right flag to be set in the OSC_STATUS reg
; While flag is not set
While_32_flag:
        lds R17, OSC_STATUS              ; Load Status Flag
        and R17, R16                     ; Bit-mask with 00000010
        cp R17, R16                      ; Compare Mask and Value
        brne While_32_flag               ; Restart loop if flag not set

; Write the IOREG signature to the CPU_CCP reg
        ldi R17, CCP_IOREG_gc    ; Load IOREG into R17
        sts CPU_CCP, R17                 ; Store IOREG into CPU CCP

;Select the new clock source in the CLK_CTRL reg
        ldi R17, CLK_SCLKSEL_RC32M_gc; load 32 MHz internal osc config
        sts CLK_CTRL, R17                ; Store config in clk control

        ;Pop Values
        pop R20 ; POP r20 from stack
        pop R19 ; POP r19 from stack
        pop R18 ; POP r18 from stack
        pop R17 ; POP r17 from stack
        ret
```

i. Appendix

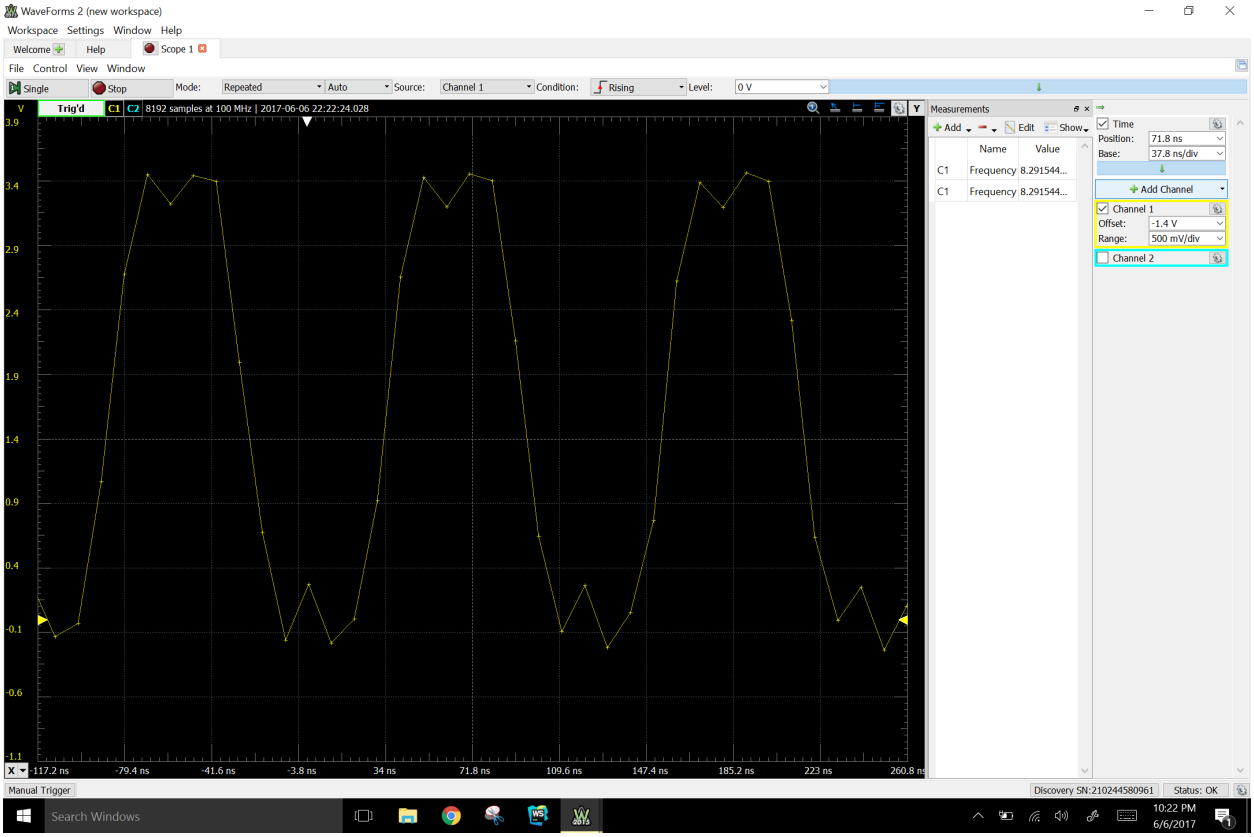

Figure 1: PART A: Clock at 8 MHz

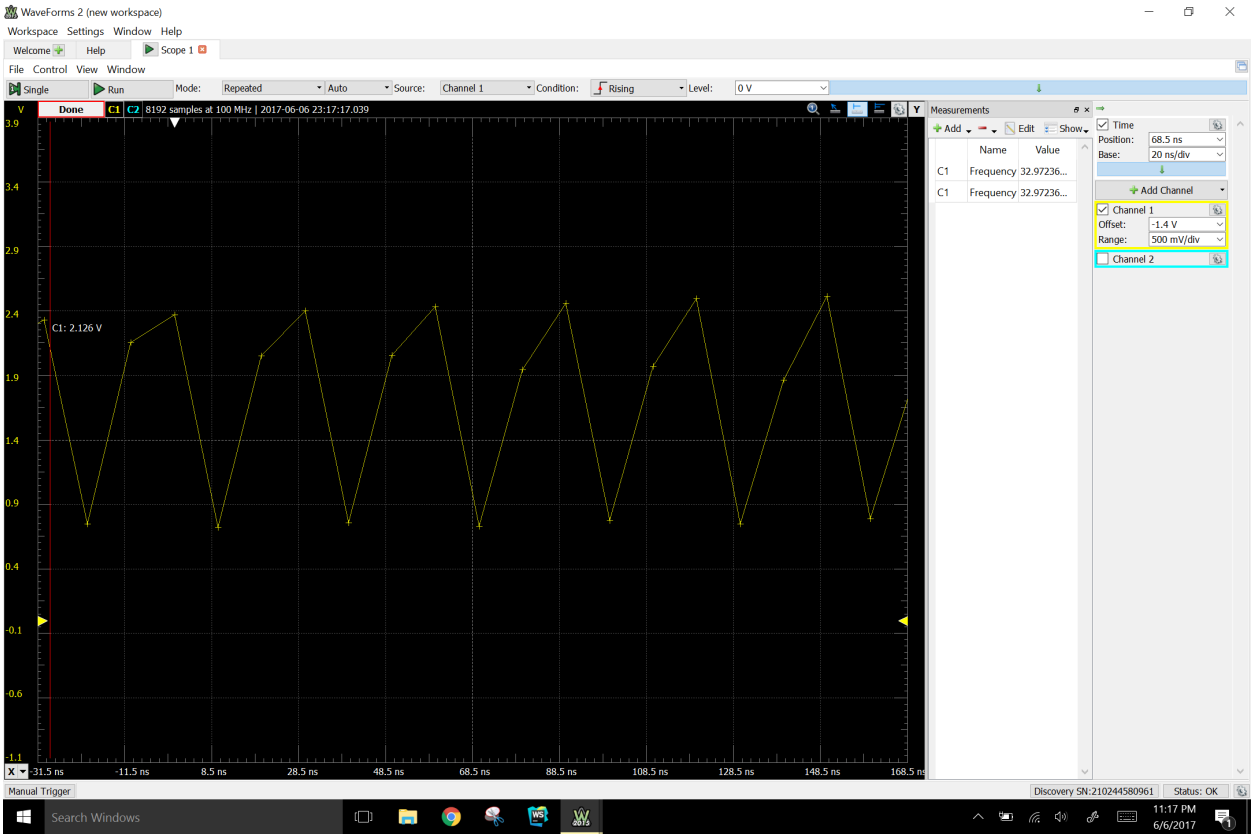

Figure 2: PART A: Clock at 32 MHz

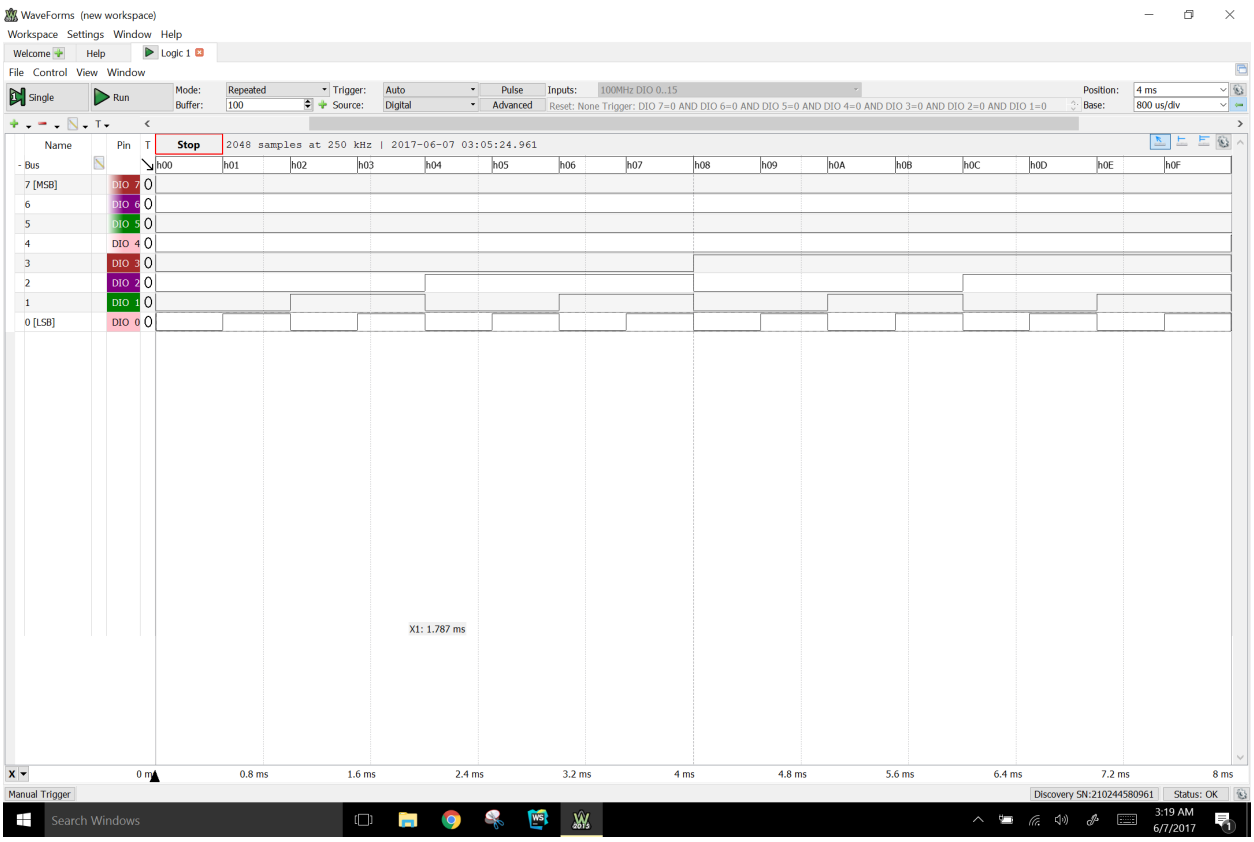

Figure 3: PART B: Counter (0-F)

LED Values

20-0-0: A pink Light
80-0-0: A red light. Not that bright
FF-0-0: A very bright red light
0-FF-0: A very bright green light
FF-FF-0: A very bright yellow light