Lab 2 Summary

Schuster, Mark

Section 1540

09/30/2017

B) Pre-Lab Answers:

1) 2 MHz
2) To set the CLK to 16 MHz:
   a) Enable the 32Mhz CLK
   b) Wait until its enable flag is set
   c) Write the "IOREG" signature to CPU_CCP
   d) Set the prescaler to 2 to divide 32MHz by 2
   e) Write the "IOREG" signature to CPU_CCP again
   f) Select the 32Mhz CLK
3) One would use the phase locked loop (PLL) to augment the 32Mhz by a factor of 4 to produce a 128Mhz CLK. Then this would be selected as the input for the system CLK. Prescaler A would be set to 1 and the resulting CLK_{PER4} would be at a frequency of 128MHz. Then prescaler B would be set to 2 which would result in CLK_{PER2} outputting a CLK signal at 64Mhz. Finally, prescaler C would be set to 2 which would result in the CPU's clock running at 32Mhz.
4) Timers are useful because they let the CPU work on another process without having to constantly decrement a counter. Thus, when a timer is up, an interrupt can be triggered and the CPU can then immediately operate on a separate task.
5) From 4k to 16k or 0x004000 to 0xFFFFFF
6) It is called 3 port ALE1 because is uses the pins of three ports (Ports J, K, And H) to output the proper addresses. It is AEL1 because a latch is used for multiplexing between bytes 0 and 1 of the SRAM address.
7) As stated in my answer for problem 6, ports J, K, and H are used for the 3-port setup. Port K is used to transmit bits [7:0] and [15:8] of the SRAM address as determined by whether the AEL signal is true or not. Port J only transmits bits [7:0] of the SRAM address. Finally, port H is used to indicate what actions are being taken by the EBI along with bits [19:16] of the SRAM address. Bit 0 represents the Write signal, Bit 1 represents the Read signal, and Bit 2 represents the ALE signal. Bit 3 is unused, and bits [7:4] are multiplexed between the chip select signals and bits [19:16] of the SRAM address.
8) 256 Kilobits. I looked up the data sheet, and then calculated for safe measure:
   $15\ address\ pins * 1\ byte\ per\ address \rightarrow 2^{15} = 32{,}768\ bytes\ per\ address\ or\ a\ 32KB\ SRAM$
9) No for 32K as this is the size of our SRAM and all address lines will be used. Yes for 64K as there will be one extra bit, meaning 2^1 or 2 aliases.

C) Problems encountered:

At first, I was unable to select the 32MHz clock, but I realized that the CPU_CPP reg has to have "IOREG" written to it directly before both the prescaler and clock control are written to.

D) Future Work/Applications:

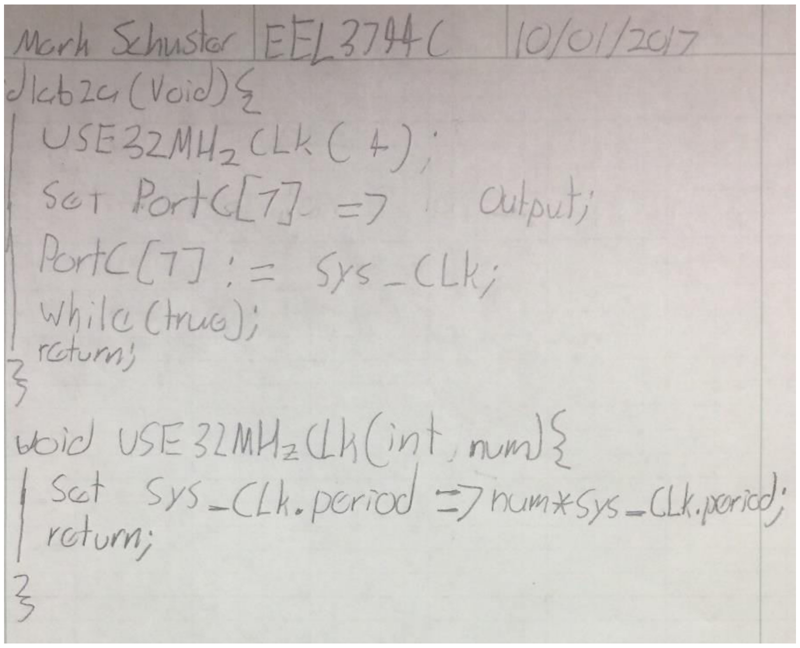
I really enjoyed using the interrupts and I look forward to the next lab where we focus on them to a greater extent.

E) Schematics:

Not applicable for this lab.

F) Pseudocode/Flowcharts:

Part A Pseudocode:

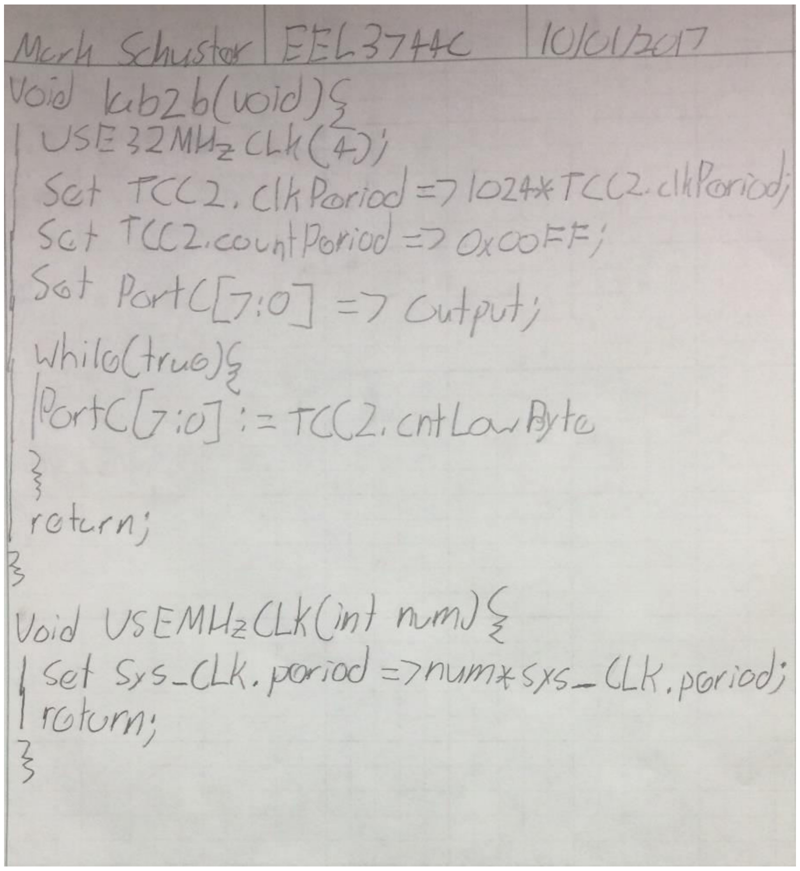


Mark Schuster | EEL3744C | 10/01/2017

```
void lab2a(Void){
    USE32MHzCLK(4);
    Set PortC[7] => Output;
    PortC[7] := Sys_CLk;
    while(true);
    return;
}

void USE32MHzClk(int num){
    Set Sys_CLk.period => num*Sys_CLk.period;
    return;
}
```

Part B Pseudocode:

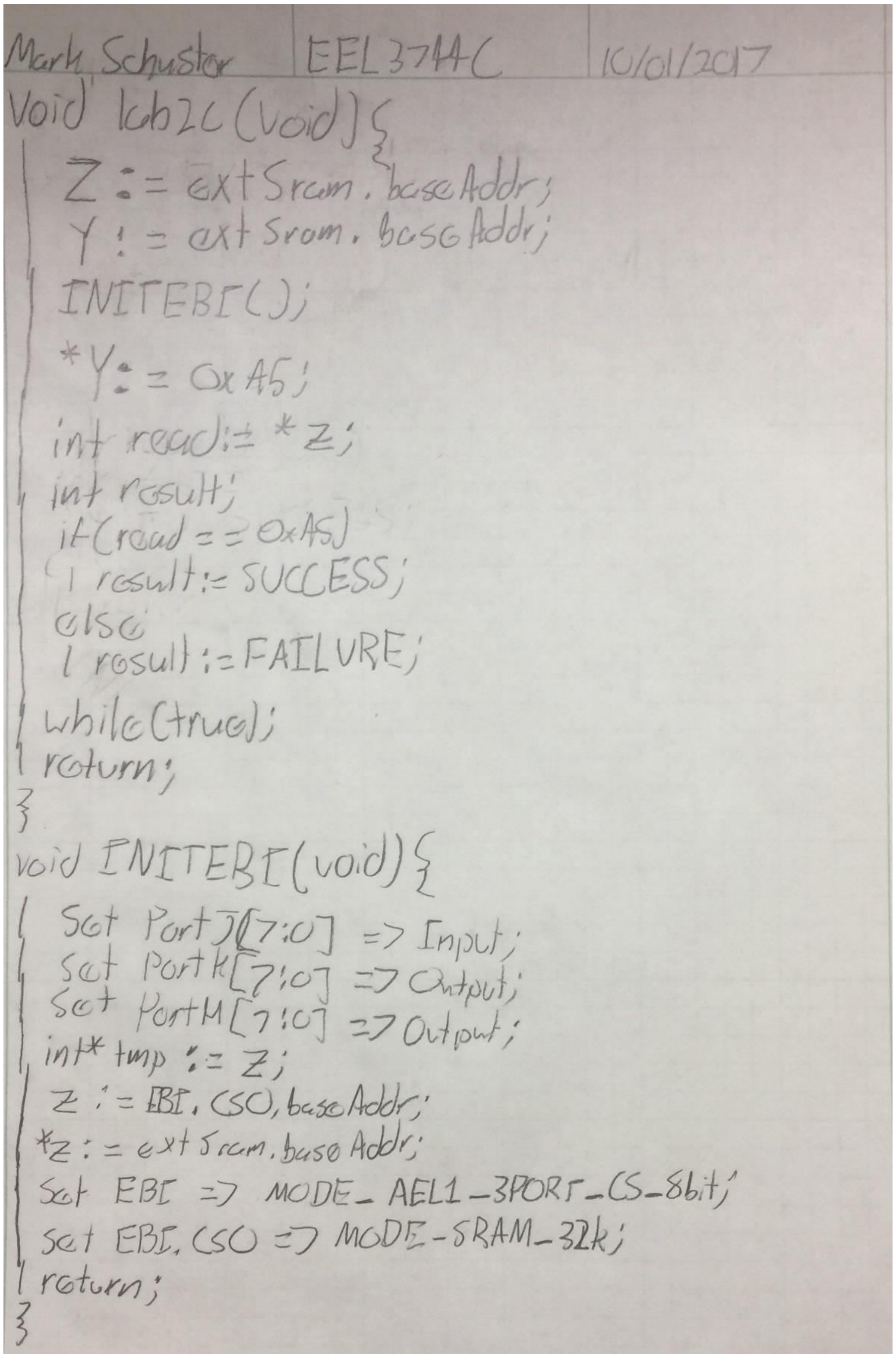


Mark Schuster | EEL3744C | 10/01/2017

```
Void lab2b(void){
    USE32MHzCLK(4);
    Set TCC2.clkPeriod => 1024*TCC2.clkPeriod;
    Set TCC2.countPeriod => 0x00FF;
    Set PortC[7:0] => Output;
    while(true){
        PortC[7:0] := TCC2.cntLowByte
    }
    return;
}

Void USEMHzCLK(int num){
    Set Sys_CLK.period => num*Sys_CLK.period;
    return;
}
```

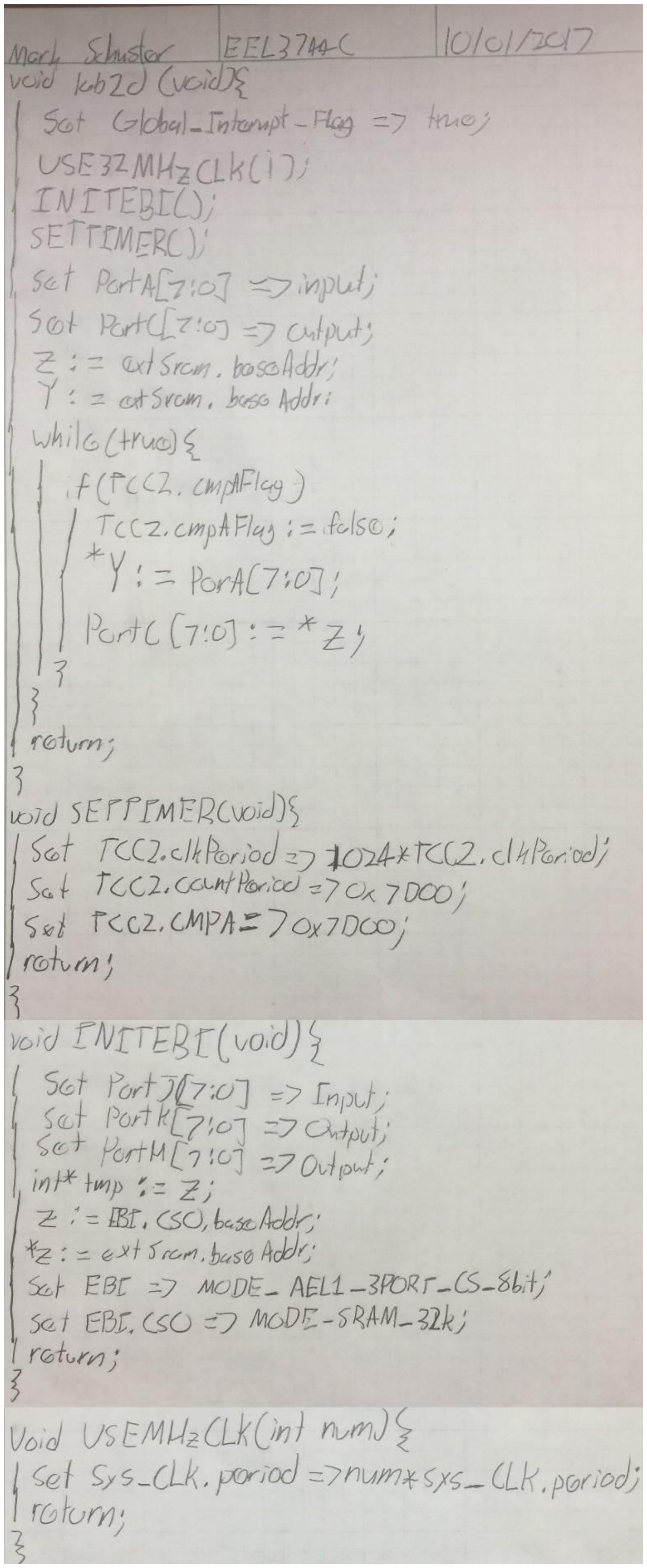
Part C Pseudocode:



Mark Schuster | EEL3744C | 10/01/2017

```
void lab2C(void){
    Z := extSram.baseAddr;
    Y := extSram.baseAddr;
    INITEBI();
    *Y := 0xA5;
    int read := *Z;
    int result;
    if(read == 0xA5)
        result := SUCCESS;
    else
        result := FAILURE;
    while(true);
    return;
}
void INITEBI(void){
    Set PortJ[7:0] => Input;
    Set PortK[7:0] => Output;
    Set PortM[7:0] => Output;
    int* tmp := Z;
    Z := EBI.CS0.baseAddr;
    *Z := extSram.baseAddr;
    Set EBI => MODE_AEL1_3PORT_CS_8bit;
    Set EBI.CS0 => MODE_SRAM_32k;
    return;
}
```

Part D Pseudocode:



```
Mark Schuster | EEL3744C | 10/01/2017
void lab2d (void){
    Set Global_Interrupt_Flag => true;
    USE32MHzCLK(1);
    INITEBI();
    SETTIMER();
    set PortA[7:0] => input;
    set PortC[7:0] => output;
    Z := extSram.baseAddr;
    Y := extSram.baseAddr;
    while(true){
        if(TCC2.cmpAFlag)
            TCC2.cmpAFlag := false;
            *Y := PorA[7:0];
            PortC[7:0] := *Z;
        }
    }
    return;
}
void SETTIMER(void){
    Set TCC2.clkPeriod => 1024*TCC2.clkPeriod;
    Set TCC2.countPeriod => 0x7D00;
    Set TCC2.CMPA => 0x7D00;
    return;
}
void INITEBI(void){
    Set PortJ[7:0] => Input;
    set PortK[7:0] => Output;
    Set PortH[7:0] => Output;
    int* tmp := Z;
    Z := EBI.CS0.baseAddr;
    *Z := extSram.baseAddr;
    Set EBI => MODE_AEL1_3PORT_CS_8bit;
    set EBI.CS0 => MODE_SRAM_32k;
    return;
}
Void USEMHzCLK(int num){
    set Sys_CLK.period => num*Sys_CLK.period;
    return;
}
```

G) Program Code:

lab2a.asm:

```
; Lab 2 part A
; Name:         Mark L. Schuster
; Section #:    1540
; TA Name:      Christopher Crary
; Description:  Sets the system clock to the 32Mhz clock
;               with a given prescaler.

.nolist                           ; Included for fun.
.include "ATxmega128A1Udef.inc"   ;
.list                             ;

.def clkPrescaler = r17           ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.
.equ CLKEN = 0b0010               ; Enables the 32Mhz CLK.
.equ IOREG = 0xD8                 ; The value that sets the CPU_CCP reg to 'IOREG' mode.
.equ CLKPS = 0b00001100           ; Value that sets Prescaler A to 4.
.equ CLKSEL = 1                   ; Value to select the 32MHz CLK.
.equ CLKOUT = 0b00000001          ; Value to output the CLK signal to port C.
.equ MSBOUT = 0b10000000          ; Value to set the MSB of a port to output.

.org 0x0000
    rjmp init                     ; Start at 0x0000 and jump to program.

.org 0x200
init:
    ldi clkPrescaler, CLKPS       ; Load the prescaler value and call USE32MHzCLK.
    rcall USE32MHzCLK             ;

main:
    ldi r16, MSBOUT               ; Set the MSB of port C to output.
    sts PORTC_DIRSET, r16         ;
    ldi r16, CLKOUT               ; Output the CLK signal to port C,
    sts PORTCFG_CLKEVOUT, r16     ;

run:
    jmp run                       ; Loop endlessly.

.org 0x300
;*********************SUBROUTINES**************************************
; Subroutine Name: USE32MHzCLK
; Sets the external 32MHz as the active clock for the device
; Inputs: r17 as the desired prescaler for the clock
; Outputs: None
; Affected: r16, r17
USE32MHzCLK:
    push r16                      ; Preserve the values of r16, r17.
    push r17                      ;
    ldi r16, CLKEN                ; Load the CLK enable value and store it in the CLK control.
    sts OSC_CTRL, r16             ;

checkReady:
    lds r16, OSC_STATUS           ; This section pulls the oscillator status reg and constantly
    andi r16, CLKEN               ; checks if the 32Mhz CLK is ready yet.
    cpi r16, CLKEN                ;
    breq clockSel                 ; If it is move on, if not loop continuously.
    rjmp checkReady               ;

clockSel:
    ldi r16, IOREG                ; Write 'IOREG' to the CPU_CCP to allow the CLK Prescaler
    sts CPU_CCP, r16              ; to be written to.
    sts CLK_PSCTRL, clkPrescaler  ;
    sts CPU_CCP, r16              ; Write 'IOREG' to the CPU_CCP to allow the CLK Control
    ldi r16, CLKSEL               ; to be set to output the 32 MHz.
    sts CLK_CTRL, r16             ;
    pop r17                       ; Restore the values of r16 and r17.
    pop r16                       ;
    ret                           ; return.
```

lab2b.asm:

```
; Lab 2 part B
; Name:         Mark L. Schuster
; Section #:    1540
; TA Name:      Christopher Crary
; Description:  Setup a timer/counter to run at 1024 times the period
;                                       of the system clock and out put the lower byte of the
;                                       count to be analyzed.

.nolist                             ; Included for fun.
.include "ATxmega128A1Udef.inc"     ;
.list                               ;

.def clkPrescaler = r17             ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.
.equ CLKEN = 0b0010                 ; Enables the 32Mhz CLK.
.equ IOREG = 0xD8                   ; The value that sets the CPU_CCP reg to 'IOREG' mode.
.equ CLKPS = 0b00001100             ; Value that sets Prescaler A to 4.
.equ CLKSEL = 1                     ; Value to select the 32MHz CLK.
.equ CLKOUT = 0b00000001            ; Value to output the CLK signal to port C.
.equ TCSEL = 0b0111                 ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.
.equ TCPER = 0x00FF                 ; Value of the TC period.
.equ ALLOUT = 0xFF                  ; 8-bit vector that will set an 8-bit GPIO port to output.

.org 0x0000
    rjmp init                       ; Start at 0x0000 and jump to program.

.org 0x200
init:
    ldi clkPrescaler, CLKPS         ; Load the prescaler value and call USE32MHzCLK.
    rcall USE32MHzCLK               ;

counterInit:
    ldi r16, TCSEL                  ; Enable the TC and set its period to be 1024 times that of the system CLK.
    sts TCC2_CTRLA, r16             ;
    ldi r16, low(TCPER)             ; Load the period of the TC into the TC's period regs.
    sts TCC2_LPER, r16              ;
    ldi r16, high(TCPER)            ;
    sts TCC2_HPER, r16              ;

portcInit:
    ldi r16, ALLOUT                 ; Set port C to output.
    sts PORTC_DIRSET, r16           ;

main:
    lds r16, TCC2_LCNT              ; Load the lower byte of the TC's count and output it
    sts PORTC_OUT, r16              ; to port C.
    rjmp main                       ; Repeat that action forever.

.org 0x300
;*********************SUBROUTINES**************************************
; Subroutine Name: USE32MHzCLK
; Sets the external 32MHz as the active clock for the device
; Inputs: r17 as the desired prescaler for the clock
; Outputs: None
; Affected: r16, r17
USE32MHzCLK:
    push r16                        ; Preserve the values of r16, r17.
    push r17                        ;
    ldi r16, CLKEN                  ; Load the CLK enable value and store it in the CLK control.
    sts OSC_CTRL, r16               ;

checkReady:
    lds r16, OSC_STATUS             ; This section pulls the oscillator status reg and constantly
    andi r16, CLKEN                 ; checks if the 32Mhz CLK is ready yet.
    cpi r16, CLKEN                  ;
    breq clockSel                   ; If it is move on, if not loop continuously.
    rjmp checkReady                 ;

clockSel:
    ldi r16, IOREG                  ; Write 'IOREG' to the CPU_CCP to allow the CLK Prescaler
    sts CPU_CCP, r16                ; to be written to.
    sts CLK_PSCTRL, clkPrescaler    ;
    sts CPU_CCP, r16                ; Write 'IOREG' to the CPU_CCP to allow the CLK Control
    ldi r16, CLKSEL                 ; to be set to output the 32 MHz.
    sts CLK_CTRL, r16               ;
    pop r17                         ; Restore the values of r16 and r17.
    pop r16                         ;
    ret                             ; return.
```

lab2c.asm:

```
; Lab 2 part C
; Name:         Mark L. Schuster
; Section #:    1540
; TA Name:      Christopher Crary
; Description:  Initialize the SRAM's mapped address to 0x200000 using the EBI

.nolist                           ; Included for fun.
.include "ATxmega128A1Udef.inc"   ;
.list                             ;

.equ EBIBASEADDR = 0x200000       ; Value of the base address of where the external SRAM is to be mapped.
.equ EBIINIT = 0b01000001         ; Value to set the EBI to the 3-port, AEL1 with chip select mode and 8 bit reg sizes.
.equ ALL_OUT = 0xFF               ; 8-bit vector that will set an 8-bit GPIO port to output.
.equ ALL_IN = 0x00                ; 8-bit vector that will set an 8-bit GPIO port to input.
.equ EBI_PORTH_OUT = 0b11110111   ; 8-bit vector that will set all bits of port H as output aside from bit 3.
.equ EBISRAMEN = 0b00011101       ; Value to set the EBI to SRAM mode with a 32K address space.

.org 0x0000
    rjmp init                     ; Start at 0x0000 and jump to program.

.org 0x200
init:
    ldi ZL, byte3(EBIBASEADDR)    ; Load the SRAM base address into both the Z and Y pointers.
    out CPU_RAMPZ, ZL             ;
    ldi ZH, byte2(EBIBASEADDR)    ;
    ldi ZL, byte1(EBIBASEADDR)    ;
    ldi YL, byte3(EBIBASEADDR)    ;
    out CPU_RAMPY, YL             ;
    ldi YH, byte2(EBIBASEADDR)    ;
    ldi YL, byte1(EBIBASEADDR)    ;
    rcall INITEBI                 ; Call the INITEBI subroutine.

loop:
    ldi r16, 0xA5                 ; Load 0xa5 into a reg and write it to the SRAM.
    st Y, r16                     ;
    ld r17, Z                     ; Load the written value back from the SRAM.

    rjmp loop                     ; Loop endlessly.

.org 0x300
;********************SUBROUTINES**************************************
; Subroutine Name: INITEBI
; Initialized the EBI
; Inputs: None
; Outputs: None
; Affected: r16, Z
INITEBI:
    push r16                       ; Preserve r16 and the Z pointer.
    mov r16, ZL                    ;
    push r16                       ;
    mov r16, ZH                    ;
    push r16                       ;
    lds r16, CPU_RAMPZ             ;
    push r16                       ;
    ldi r16, ALL_IN                ; Set port J to input.
    sts PORTJ_DIRSET, r16          ;
    ldi r16, ALL_OUT               ; Set port K to output.
    sts PORTK_DIRSET, r16          ;
    ldi r16, EBI_PORTH_OUT         ; Set all port H to output expect for bit 3 which goes unused.
    sts PORTH_DIRSET, r16          ;
    ldi r16, EBIINIT               ; Set the EBI to the 3-port, AEL1 with chip select mode with 8 bit reg sizes.
    sts EBI_CTRL, r16              ;
    ldi ZL, 0x00                   ; Clear Z's ramp, and set Z to the address of chip 0's base address.
    out CPU_RAMPZ, ZL              ;
    ldi ZL, low(EBI_CS0_BASEADDR)  ;
    ldi ZH, high(EBI_CS0_BASEADDR) ;
    ldi r16, byte2(EBIBASEADDR)    ; Then load the chosen base address into chip 0's base address's location.
    st Z+, r16                     ;
    ldi r16, byte3(EBIBASEADDR)    ;
    st Z, r16                      ;
    ldi r16, EBISRAMEN             ; Specify that chip 0 is SRAM with a 32K address space.
    sts EBI_CS0_CTRLA, r16         ;
    pop r16                        ; Restore r16 and the Z pointer.
    mov ZL, r16                    ;
    out CPU_RAMPZ, r16             ;
    pop r16                        ;
    mov ZH, r16                    ;
    pop r16                        ;
    mov ZL, r16                    ;
    pop r16                        ;
    ret                            ; Return.
```

lab2d.asm:

```
; Lab 2 part D
; Name:         Mark L. Schuster
; Section #:    1540
; TA Name:      Christopher Crary
; Description:  Program that establishes external SRAM at 0x200000 and writes the current switch values to the SRAM
;               every second. Additionally, the current value at the location of the stored switch values is outputted
;               to the LEDs every second.

.nolist                         ; Included for fun.
.include "ATxmega128A1Udef.inc" ;
.list                           ;

.equ EBIBASEADDR = 0x200000     ; Value of the base address of where the external SRAM is to be mapped.
.equ EBIINIT = 0b01000001       ; Value to set the EBI to the 3-port, AEL1 with chip select mode and 8 bit reg sizes.
.equ ALL_OUT = 0xFF             ; 8-bit vector that will set an 8-bit GPIO port to output.
.equ ALL_IN = 0x00              ; 8-bit vector that will set an 8-bit GPIO port to input.
.equ EBI_PORTH_OUT = 0b11110111 ; 8-bit vector that will set all bits of port H as output aside from bit 3.
.equ EBISRAMEN = 0b00011101     ; Value to set the EBI to SRAM mode with a 32K address space.
.def clkPrescaler = r17         ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.
.equ CLKEN = 0b0010             ; Enables the 32Mhz CLK.
.equ IOREG = 0xD8               ; The value that sets the CPU_CCP reg to 'IOREG' mode.
.equ CLKPS = 0b00000000         ; Value that sets Prescaler A to 4.
.equ CLKSEL = 1                 ; Value to select the 32MHz CLK.
.equ CLKOUT = 0b00000001        ; Value to output the CLK signal to port C.
.equ TCSEL = 0b0111             ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.
.equ TCPER = 0x7D00             ; Value of the TC period.
.equ TCCMPAINT = 0b00010000     ; Value to init the TC compare A reg.
.equ TCCMPA_INTFLAGLOC = 4      ; Location of the compare interrupt flag in the TC's interrupt flags reg.

.org 0x0000
    rjmp init                   ; Start at 0x0000 and jump to program.

.org 0x100
init:
    sei                         ; Set the global interrupt bit.
    rcall INITEBI               ; Call INITEBI.
    ldi clkPrescaler, CLKPS     ; Load the prescaler input for USE32MHzCLK and then call it.
    rcall USE32MHzCLK           ;
    rcall SETTIMER              ; Call SETTIMER.
portcInit:
    ldi r16, ALL_IN             ; Set 'initValue' to 8-bit GPIO input vector.
    sts PORTA_DIRSET, r16       ; Set PORTA to be input.
    ldi r16, ALL_OUT            ; Set 'initValue' to 8-bit GPIO output vector.
    sts PORTC_DIRSET, r16       ; Set PORTC to be output.
    ldi ZL, byte3(EBIBASEADDR)  ; Load the base address of the external SRAM into the Z
    out CPU_RAMPZ, ZL           ; and Y pointers.
    ldi ZH, byte2(EBIBASEADDR)  ;
    ldi ZL, byte1(EBIBASEADDR)  ;
    ldi YL, byte3(EBIBASEADDR)  ;
    out CPU_RAMPY, YL           ;
    ldi YH, byte2(EBIBASEADDR)  ;
    ldi YL, byte1(EBIBASEADDR)  ;

main:
    lds r16, TCC2_INTFLAGS      ; Check the compare A interrupt flag of the TC.
    mov r17, r16                ;
    andi r16, TCCMPAINT         ;
    cpi r16, TCCMPAINT          ;
    breq toggle                 ; If it is set, move to toggle.
    rjmp main                   ; If not, loop to check again.
toggle:
    cbr r17, TCCMPA_INTFLAGLOC  ; Clear the interrupt flag from the TC's interrupt flags reg.
    sts TCC2_INTFLAGS, r17      ;
    lds r16, PORTA_IN           ; Pull the current switch values and store them in the
    st Y, r16                   ; external SRAM.
    ld r17, Z                   ; Load the switch values back from SRAM and output them to
    sts PORTC_OUT, r17          ; the LEDs.
    rjmp main                   ; Loop to wait for the flag to be set again.



.org 0x300
;********************SUBROUTINES**************************************
; Subroutine Name: USE32MHzCLK
; Sets the external 32MHz as the active clock for the device
; Inputs: r17 as the desired prescaler for the clock
; Outputs: None
; Affected: r16, r17
USE32MHzCLK:
    push r16                    ; Preserve the values of r16, r17.
    push r17                    ;
    ldi r16, CLKEN              ; Load the CLK enable value and store it in the CLK control.
    sts OSC_CTRL, r16              ;

checkReady:
    lds r16, OSC_STATUS            ; This section pulls the oscillator status reg and constantly
    andi r16, CLKEN                ; checks if the 32Mhz CLK is ready yet.
    cpi r16, CLKEN                 ;
    breq clockSel                  ; If it is move on, if not loop continuously.
    rjmp checkReady                ;

clockSel:
    ldi r16, IOREG                 ; Write 'IOREG' to the CPU_CCP to allow the CLK Prescaler
    sts CPU_CCP, r16               ; to be written to.
    sts CLK_PSCTRL, clkPrescaler   ;
    sts CPU_CCP, r16               ; Write 'IOREG' to the CPU_CCP to allow the CLK Control
    ldi r16, CLKSEL                ; to be set to output the 32 MHz.
    sts CLK_CTRL, r16              ;
    pop r17                        ; Restore the values of r16 and r17.
    pop r16                        ;
    ret                            ; return.

.org 0x350
;*********************SUBROUTINES**************************************
; Subroutine Name: INITEBI
; Initialized the EBI
; Inputs: None
; Outputs: None
; Affected: r16, Z
INITEBI:
    push r16                       ; Preserve r16 and the Z pointer.
    mov r16, ZL                    ;
    push r16                       ;
    mov r16, ZH                    ;
    push r16                       ;
    lds r16, CPU_RAMPZ             ;
    push r16                       ;
    ldi r16, ALL_IN                ; Set port J to input.
    sts PORTJ_DIRSET, r16          ;
    ldi r16, ALL_OUT               ; Set port K to output.
    sts PORTK_DIRSET, r16          ;
    ldi r16, EBI_PORTH_OUT         ; Set all port H to output expect for bit 3 which goes unused.
    sts PORTH_DIRSET, r16          ;
    ldi r16, EBIINIT               ; Set the EBI to the 3-port, AEL1 with chip select mode with 8 bit reg sizes.
    sts EBI_CTRL, r16              ;
    ldi ZL, 0x00                   ; Clear Z's ramp, and set Z to the address of chip 0's base address.
    out CPU_RAMPZ, ZL              ;
    ldi ZL, low(EBI_CS0_BASEADDR) ;
    ldi ZH, high(EBI_CS0_BASEADDR);
    ldi r16, byte2(EBIBASEADDR)    ; Then load the chosen base address into chip 0's base address's location.
    st Z+, r16                     ;
    ldi r16, byte3(EBIBASEADDR)    ;
    st Z, r16                      ;
    ldi r16, EBISRAMEN             ; Specify that chip 0 is SRAM with a 32K address space.
    sts EBI_CS0_CTRLA, r16         ;
    pop r16                        ; Restore r16 and the Z pointer.
    mov ZL, r16                    ;
    out CPU_RAMPZ, r16             ;
    pop r16                        ;
    mov ZH, r16                    ;
    pop r16                        ;
    mov ZL, r16                    ;
    pop r16                        ;
    ret                            ; Return.

.org 0x450
;*********************SUBROUTINES**************************************
; Subroutine Name: SETTIMER
; Initialized TCC2 by setting its period and compare A value.
; Inputs: None
; Outputs: None
; Affected: r16
SETTIMER:
    push r16                       ; Preserve r16.
    ldi r16, TCSEL                 ; Enable the TC and set its period to be 1024 times that of the system CLK.
    sts TCC2_CTRLA, r16            ;
    ldi r16, low(TCPER)            ; Load the period of the TC into the TC's period regs and load the same
    sts TCC2_LPER, r16             ; value onto the TC's compare A reg.
    sts TCC2_LCMPA, r16            ;
    ldi r16, high(TCPER)           ;
    sts TCC2_HPER, r16             ;
    sts TCC2_HCMPA, r16            ;
    pop r16                        ; Restore r16.
    ret                            ; Return.
```

H) Appendix:

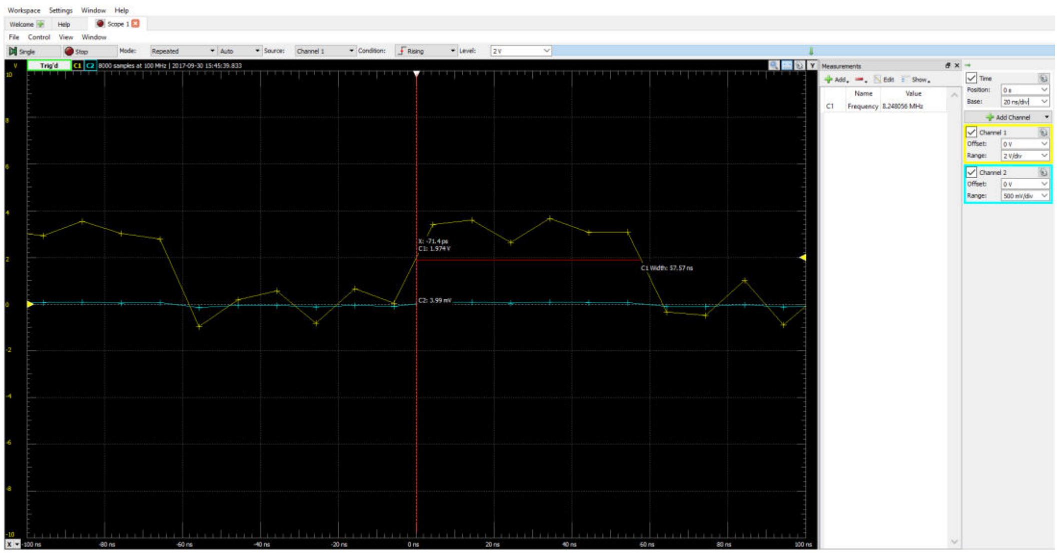
Files:

- lab2.pdf
- lab2a.asm
- lab2b.asm
- lab2c.asm
- lab2d.asm

Screenshots:

Parts A:

8 MHz clock signal:



Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 2 V

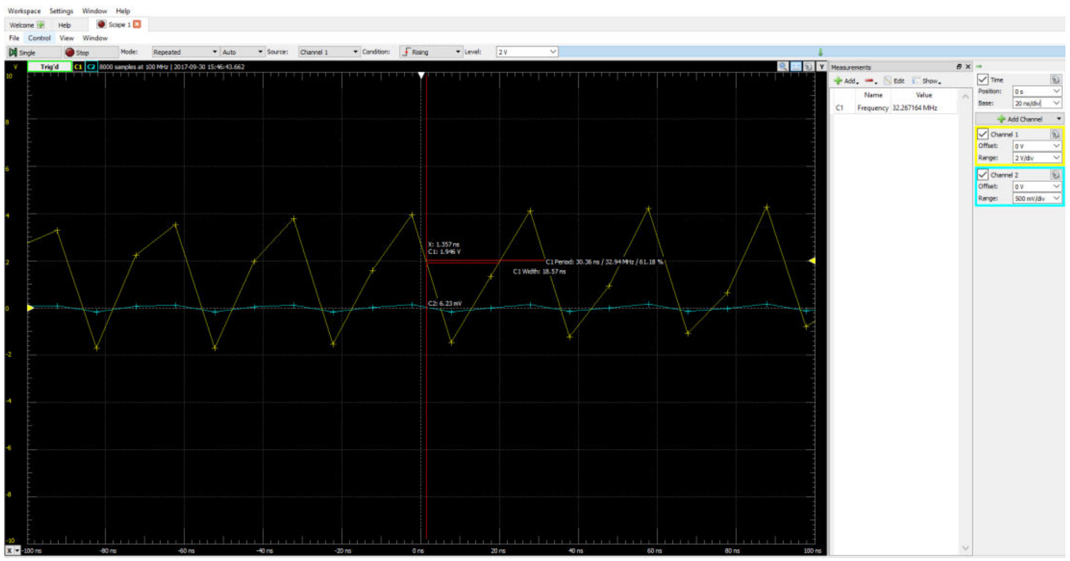
8000 samples at 100 MHz | 2017-09-30 15:45:39.833

X: -71.4 ps
C1: 1.974 V
C2: 3.99 mV
C1 Width: 57.57 ns

	Name	Value
C1	Frequency	8.248056 MHz

Time — Position: 0 s, Base: 20 ns/div
Channel 1 — Offset: 0 V, Range: 2 V/div
Channel 2 — Offset: 0 V, Range: 500 mV/div

32 MHz clock signal:



Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 2 V

8000 samples at 100 MHz | 2017-09-30 15:46:43.662

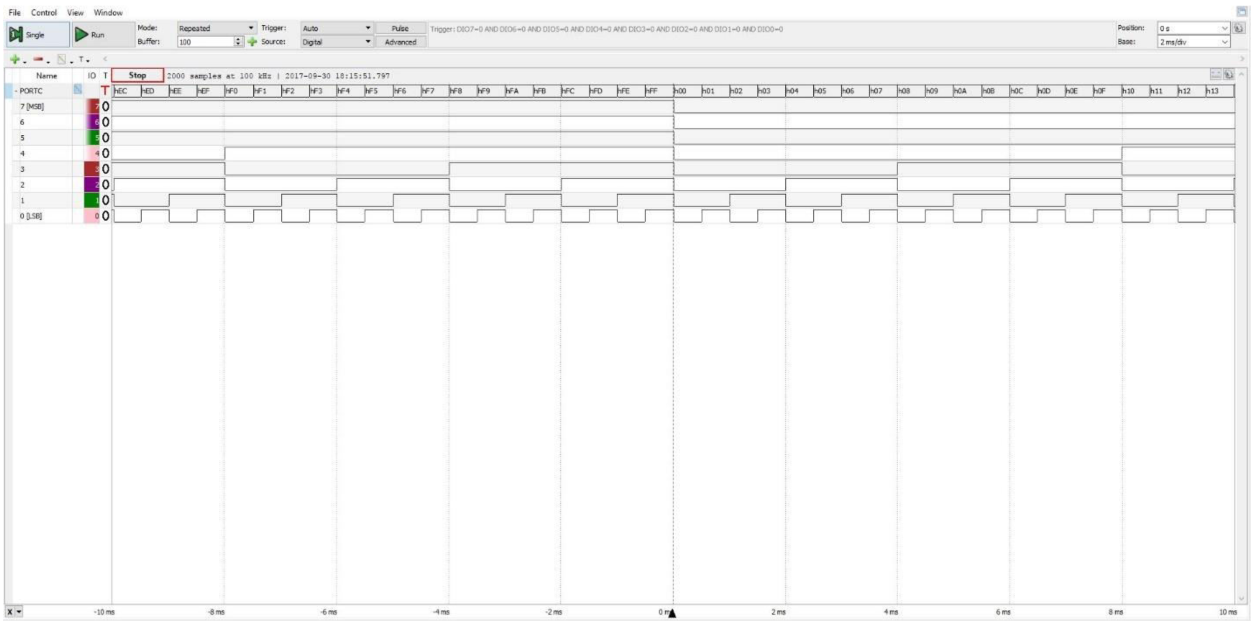
X: 1.357 ns
C1: 1.946 V
C2: 6.23 mV
C1 Period: 30.36 ns / 32.94 MHz / 61.18 %
C1 Width: 18.57 ns

	Name	Value
C1	Frequency	32.267164 MHz

Time — Position: 0 s, Base: 20 ns/div
Channel 1 — Offset: 0 V, Range: 2 V/div
Channel 2 — Offset: 0 V, Range: 500 mV/div

Part B:

Logic Analyzer displaying the low byte of the TC's count.


Mode: Repeated
Buffer: 100
Trigger: Auto
Source: Digital
Trigger: DIO7=0 AND DIO6=0 AND DIO5=0 AND DIO4=0 AND DIO3=0 AND DIO2=0 AND DIO1=0 AND DIO0=0
Position: 0 s
Base: 2 ms/div

```
2000 samples at 100 kHz | 2017-09-30 18:15:51.797
```

File
Control
View
Window
Single
Run
Pulse
Advanced
Name
IO
T
Stop
PORTC
7 [MSB]
6
5
4
3
2
1
0 [LSB]
hEC hED hEE hEF hF0 hF1 hF2 hF3 hF4 hF5 hF6 hF7 hF8 hF9 hFA hFB hFC hFD hFE hFF h00 h01 h02 h03 h04 h05 h06 h07 h08 h09 h0A h0B h0C h0D h0E h0F h10 h11 h12 h13
-10 ms
-8 ms
-6 ms
-4 ms
-2 ms
0 ms
2 ms
4 ms
6 ms
8 ms
10 ms