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:
   1. Enable the 32Mhz CLK
   2. Wait until its enable flag is set
   3. Write the “IOREG” signature to CPU\_CCP
   4. Set the prescaler to 2 to divide 32MHz by 2
   5. Write the “IOREG” signature to CPU\_CCP again
   6. 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 CLKPER4 would be at a frequency of 128MHz. Then prescaler B would be set to 2 which would result in CLKPER2 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:
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 2­­­1 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:

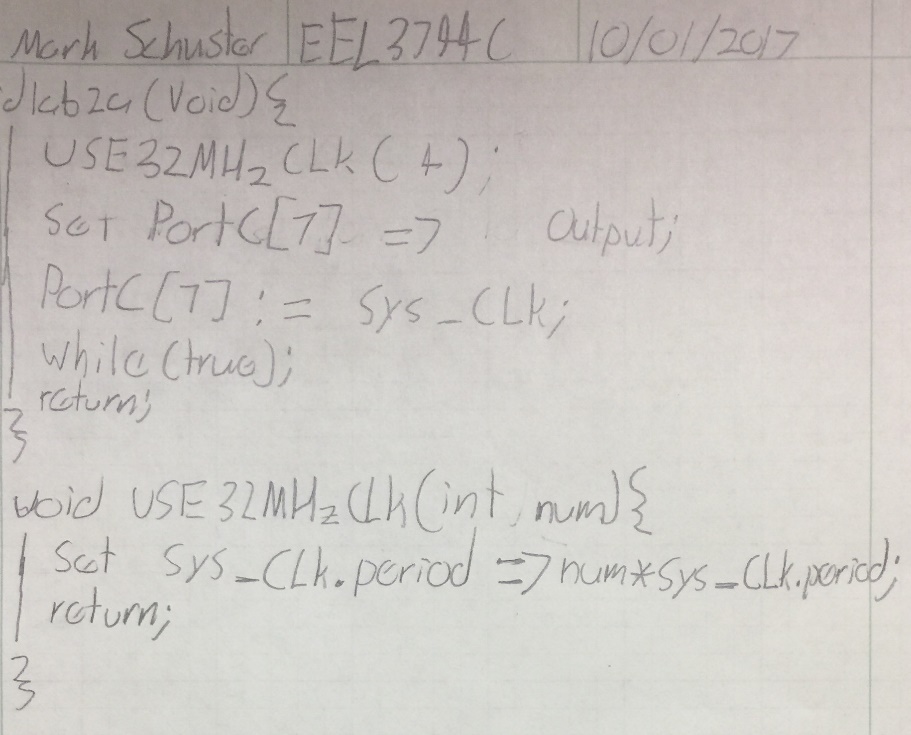
I really enjoyed using the interrupts and I look forward to the next lab where we focus on them to a greater extent.

**E)** Schematics:

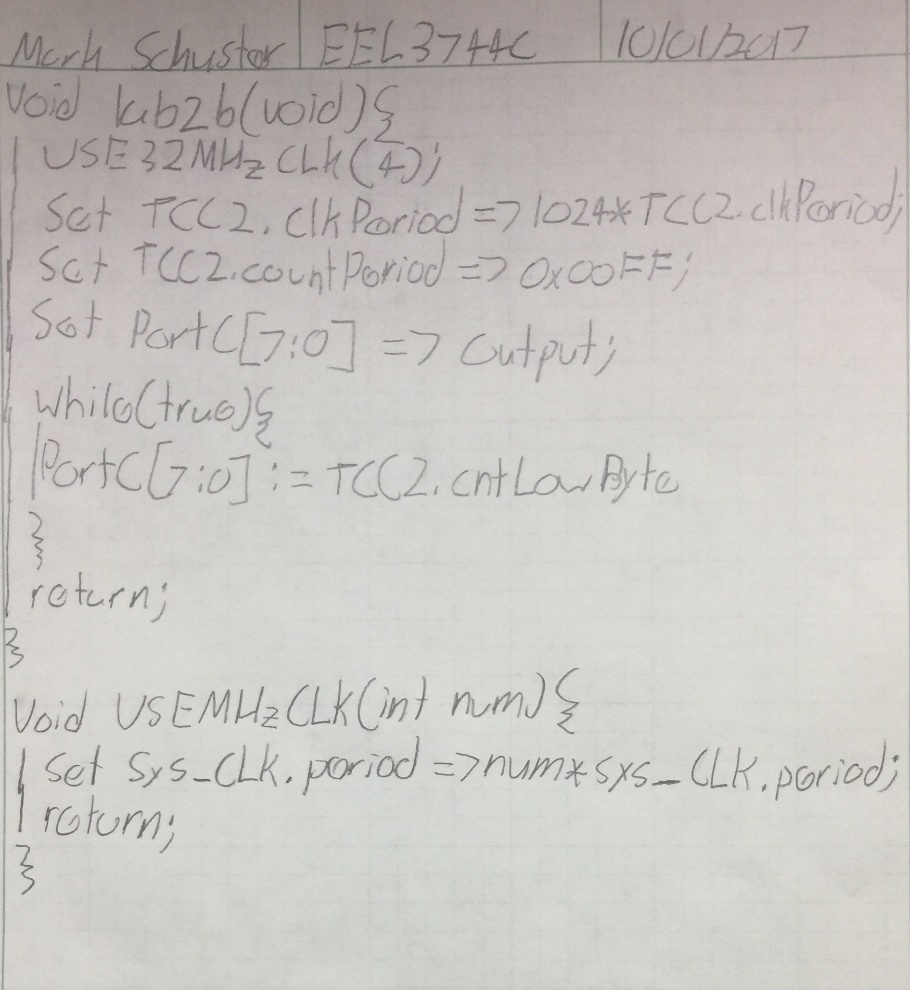
Not applicable for this lab.

**F)** Pseudocode/Flowcharts:

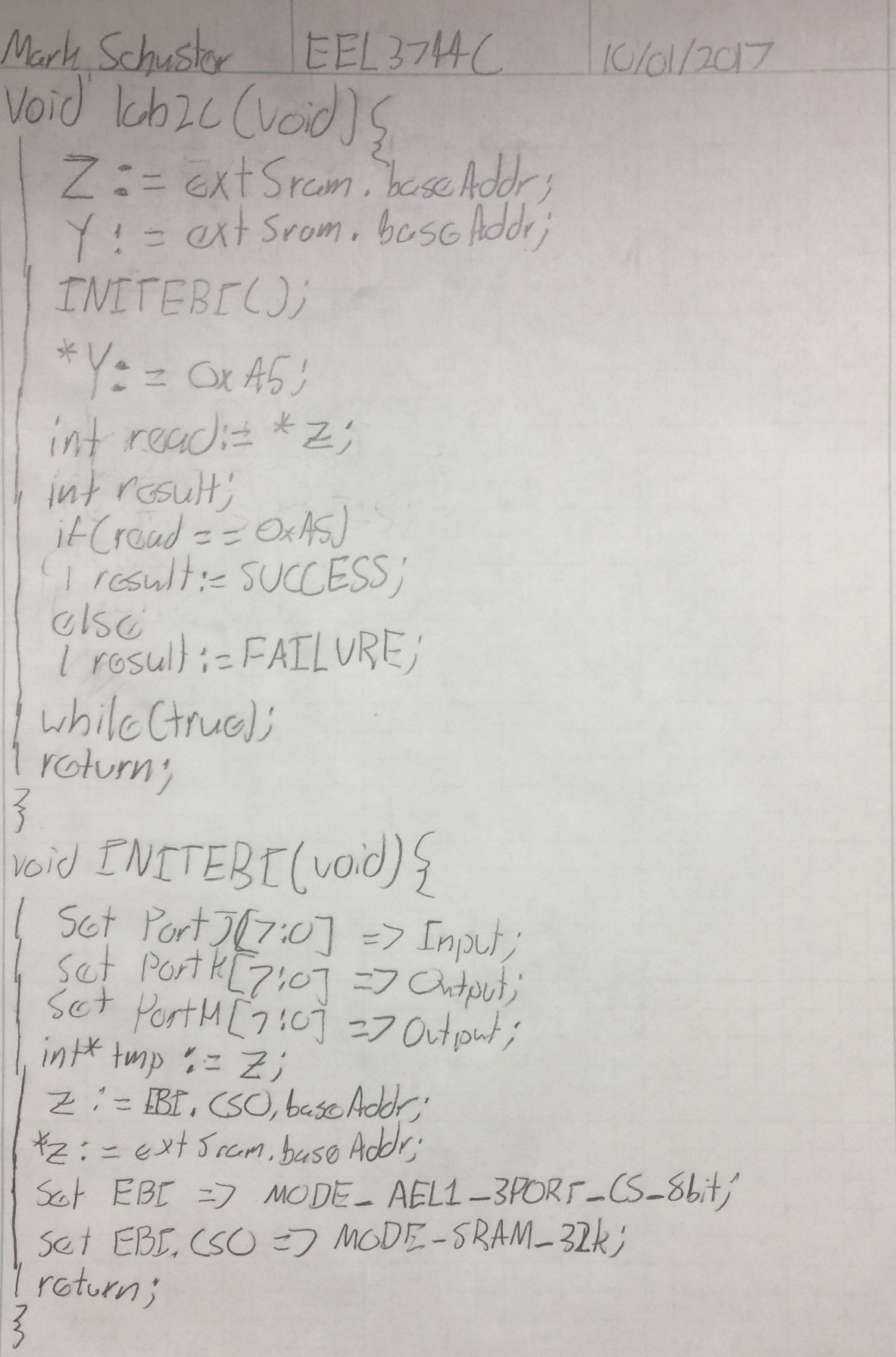
Part A Pseudocode:



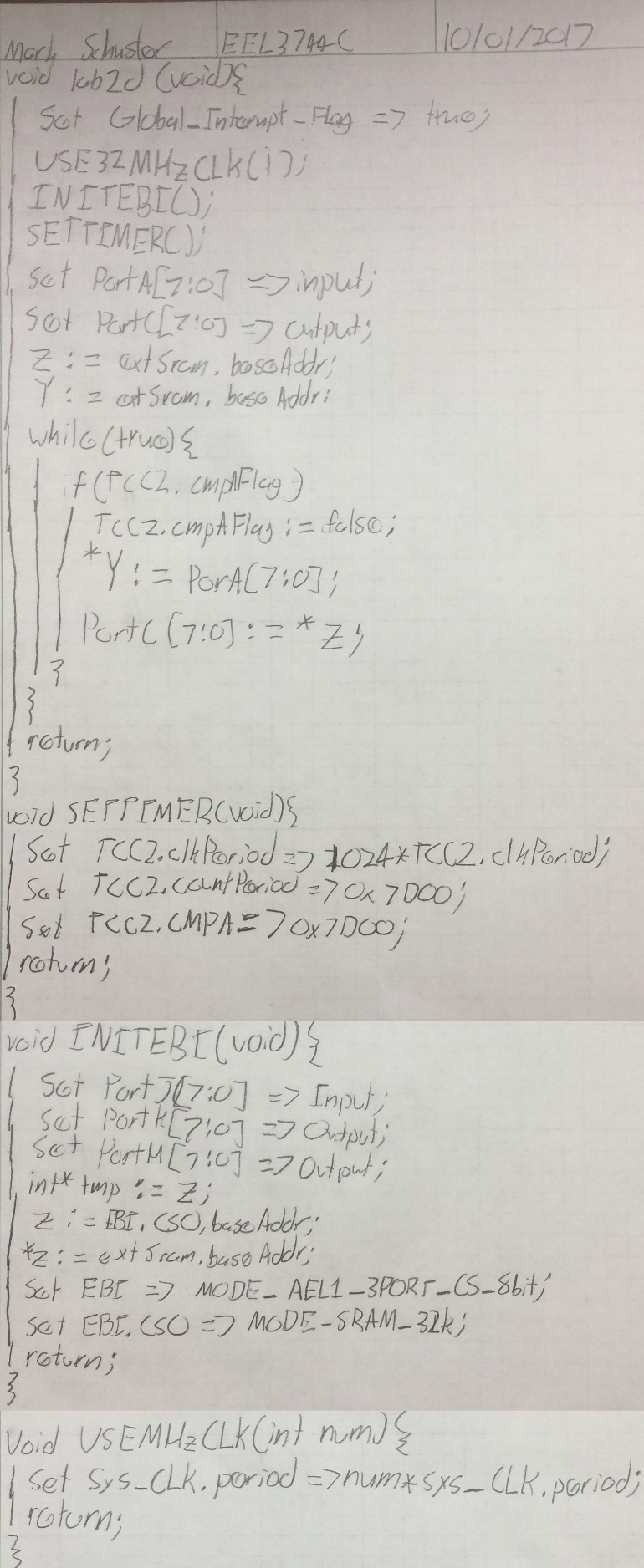
Part B Pseudocode:



Part C Pseudocode:



Part D Pseudocode:



**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:

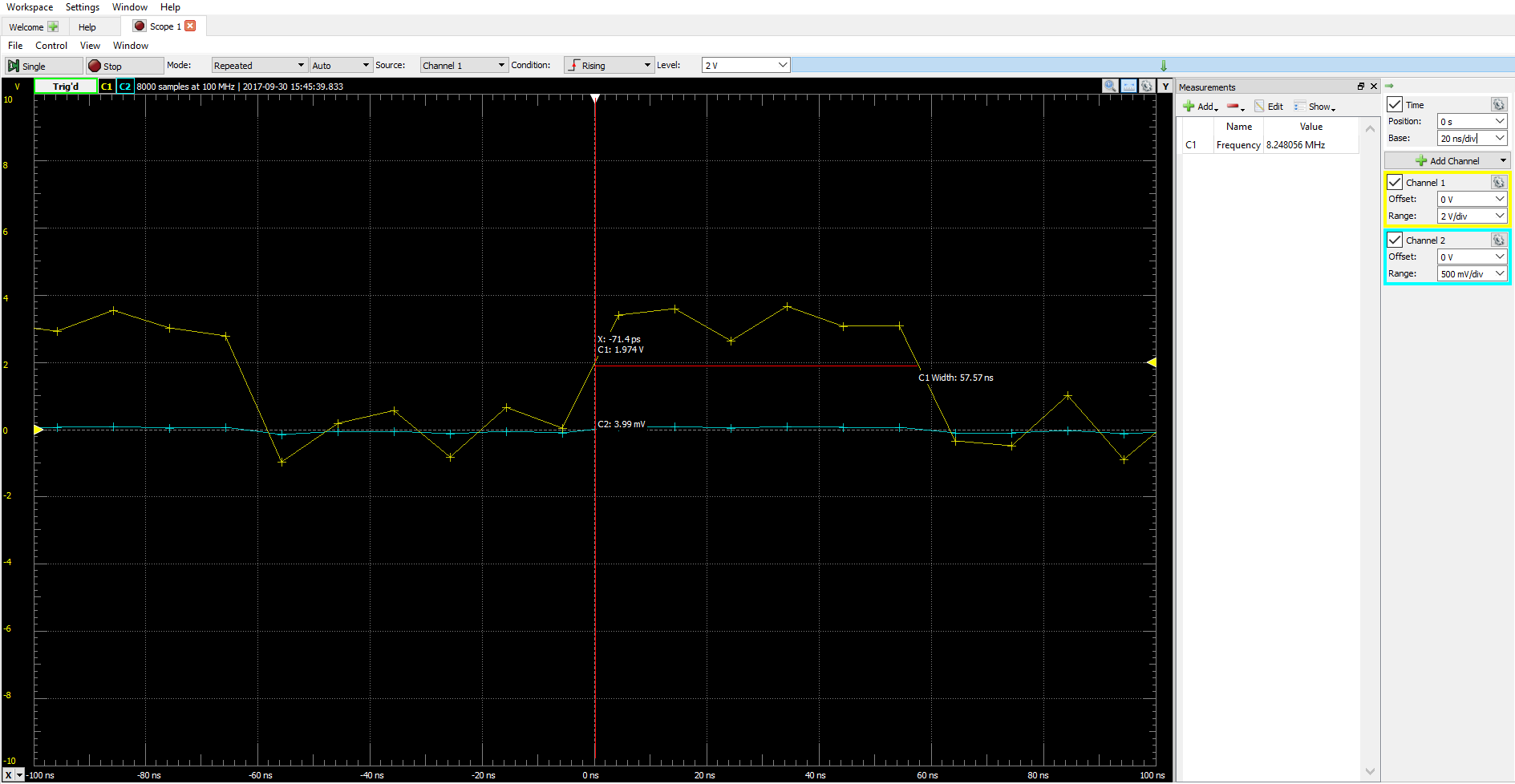
Files:

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

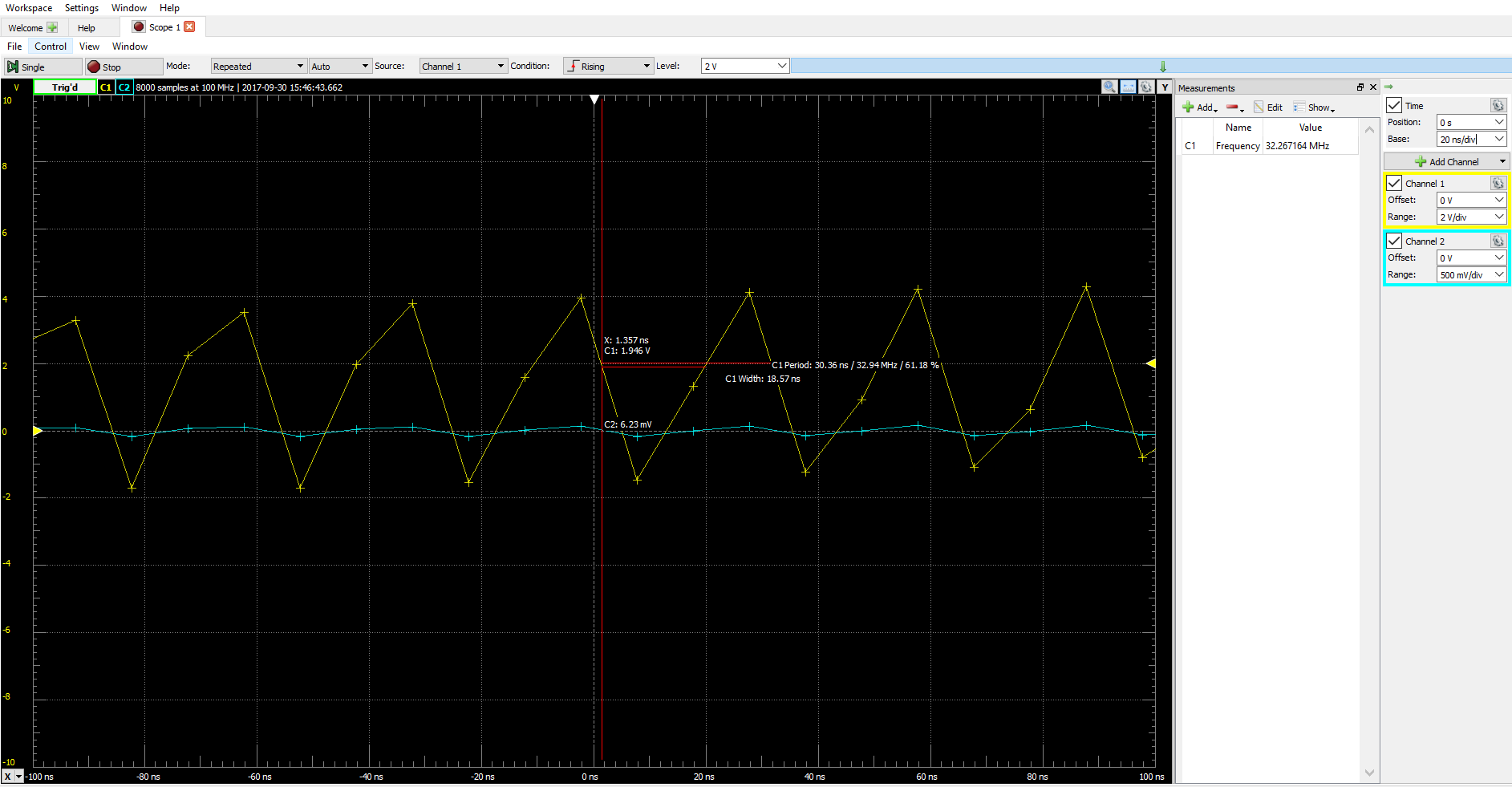
Screenshots:

Parts A:

8 MHz clock signal:



32 MHz clock signal:



Part B:

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

