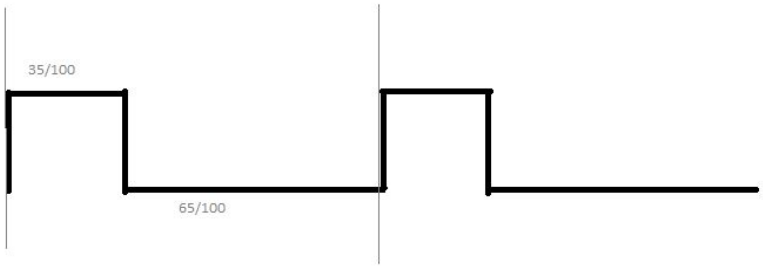
Lab 6
Jineidy Mak
Secton 7F31

Pre-Lab Questions:
1. Draw a 15 kHz square wave with a 35%% duty cycle. What is the period in milliseconds (ms)?



Period should be 66x10^3 ms with a 2 MHZ clk.

2. How does the prescaler affect the way the TC system counts per clock cycle? Where are the counts stored?

Counts are stored in the CNT register. The prescaler divides the clock. For example a prescaler of 2 will increment the CNT register every 1,000,000 (500,000 for 50% duty cycle) clk ticks instead of every 2,000,000.

3. For part A, what is the limiting factor for the precision of your frequency generation? Can your XMEGA generate some frequency ranges with higher precisions than other frequency ranges? Explain.

We are only limited by base 16, the amount of digits that are available to use when it comes to precise frequencies. Lower frequencies we are limited by the amount of bits we can use. Higher frequencies are harder to generate precisely because our available combination of bits become smaller and smaller.

4. Describe the difference(s) between the TC's Frequency Generation mode and its Single/Dual Slope PWM modes. Which mode(s) can be used to emulate the other(s)? How

could you make a sine wave or other waveform using your XMEGA by using the timer system? Do you need to add any extra hardware? How can you produce these waveforms without extra hardware?

The biggest and main difference between the two is duty cycle. You can use the PWM mode to simulate the timer by using a 50% duty cycle. We can't hope to output a perfect sine wave without external hardware. The precision of the sine wave is dependent on how many parts you can divide high and low into. Using PWM we can get voltages between 0 and 5V, but we are limited to how many divisions we can make.

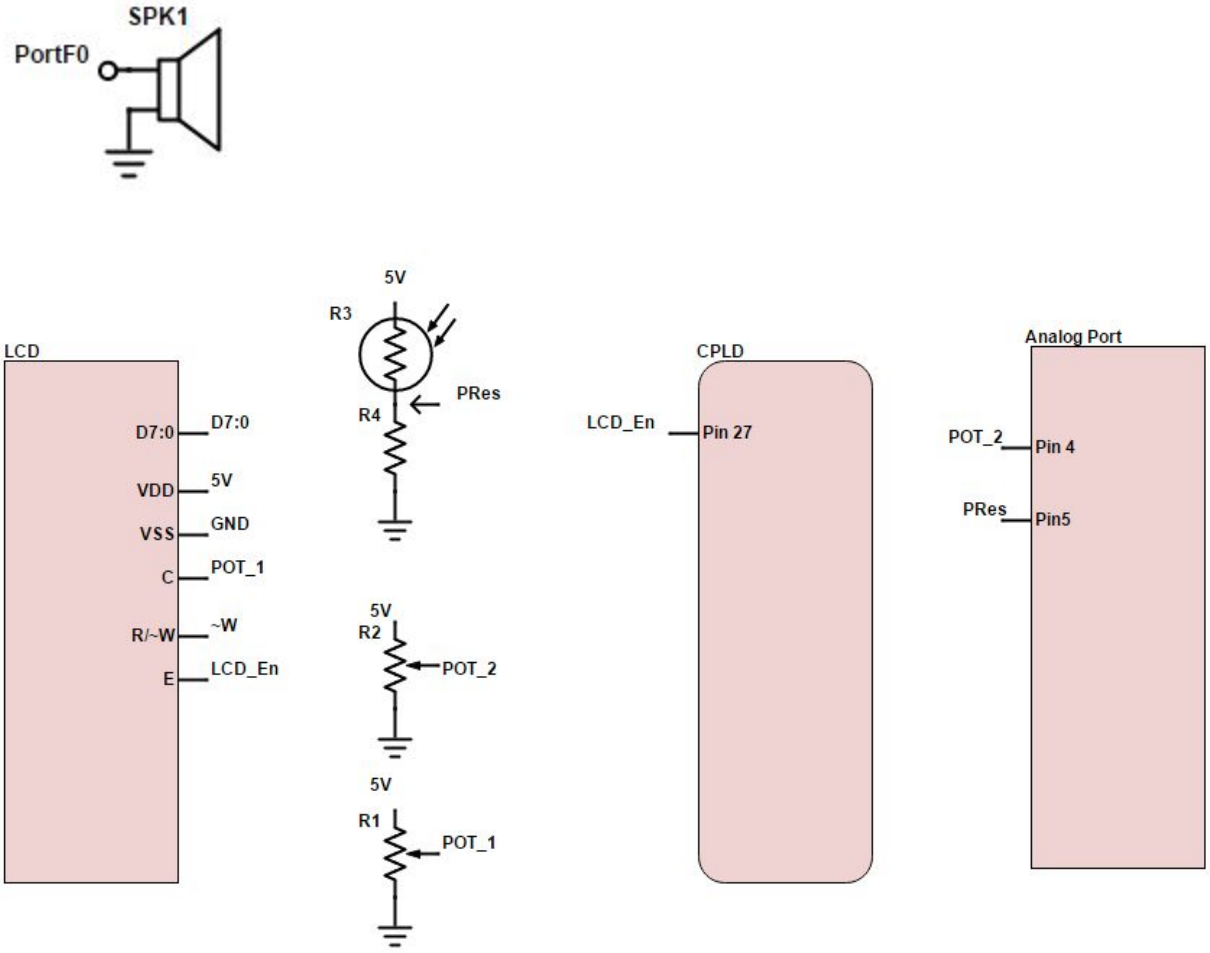
Problems Encountered:

None.

Future Work/Applications:

Some remote controls make a noise when buttons are pressed.

Schematics:


SPK1
PortF0
LCD
D7:0
D7:0
VDD
5V
VSS
GND
C
POT_1
R/~W
~W
E
LCD_En
5V
R3
PRes
R4
5V
R2
POT_2
5V
R1
POT_1
CPLD
LCD_En
Pin 27
Analog Port
POT_2
Pin 4
PRes
Pin5

Decoding Logic:

```
library ieee;
use ieee.std_logic_1164.all;

entity lab3 is
  port (

    CS0_L   : in  std_logic;
    CS1_L   : in  std_logic;
    RE_L      : in  std_logic;
    WE_L      : in  std_logic;
    A15                 : in std_logic;
    A14                 : in std_logic;
    A13                 : in std_logic;
    A12                 : in std_logic;
    LCD_En  : out std_logic

    );

end lab3;


architecture BHV of lab3 is

begin

process(CS0_L, CS1_L, RE_L, WE_L, A15, A14, A13, A12)

            variable CS0_H : std_logic;
            variable CS1_H : std_logic;
            variable A15_L : std_logic;
            variable A14_L : std_logic;
            variable A13_L : std_logic;
            variable RE_H : std_logic;
            variable WE_H : std_logic;

begin

  CS0_H := not(CS0_L);
  CS1_H := not(CS1_L);
  RE_H := not(RE_L);
  WE_H := not(WE_L);
  A15_L := not(A15);
  A14_L := not(A14);
  A13_L := not(A13);

  LCD_En <= CS1_H and (RE_H or WE_H) and ((A15_L and A14_L and A13) or (A15_L and A14));

 end process;

end BHV;
```

Pseudocode/Flowcharts:

PartA:

Initialize TC.
Initialize PortF.
If(switch = 1)
Play sound at correct frequency by entering a frequency greater than 0 into the register.
Else
Keep looping.

PartB:

Initialize TC
Initialize EBI
Initialize LCD
Initialize UART

Switch(UART)
Cases

For scales use a array of hex values and a for loop, loop through all values with delay in between.

Program Code:

Part A:

```
/*
 * Lab6a.c
 *
 * Created: 7/18/2016 4:03:04 PM
 * Author : James Mak
 */

#include <avr/io.h>
#include "ebi_driver.h"
#include <string.h>
#define F_CPU 2000000
#define CS0_Start 0x8000
#define CS1_Start 0x420000
#define LCD_Start 0x422000
#define LCD_Write 0x422001

void TC_INIT(void)
{
	//We are using Port E TC because it is the only port available that isn't taken.
	PORTE_DIRSET = 0x01; //Set Pin 1 to output.
	TCE0.CTRLA = TC_CLKSEL_OFF_gc;  //Turn of the Counter.
	TCE0.CTRLB = TC_WGMODE_FRQ_gc | TC0_CCAEN_bm;  //Waveform FRQ generation mode and Compare
enable A set.
	TCE0.CTRLE = TC_BYTEM_NORMAL_gc; //Normal mode TC.
	TCE0.CTRLFCLR = TC0_DIR_bm;  //Incrementing Counter.
	TCE0.CNT = 0; //Start count at zero.

}

void Delay_XuS(int a) // Delay a certain amount of uS.
```

```
{
	for(int i = 0; i < a; i++);
}

void BF_POLL(void)
{
	uint8_t BF = __far_mem_read(LCD_Start);

	while((BF & 0x80) == 0x80) //Poll the busy flag (bit 7 until it is clear.
	{
		BF = __far_mem_read(LCD_Start);
	}
}

void LCD_INIT(void) // LCD_INIT from lab 5.
{
	__far_mem_write(LCD_Start, 0x38); //Configure function.
	BF_POLL();
	__far_mem_write(LCD_Start, 0x0F); //Configure display.
	BF_POLL();
	__far_mem_write(LCD_Start, 0x01); //Clear display.
	BF_POLL();
	__far_mem_write(LCD_Start, 0x06); //Configure entry mode.
	BF_POLL();
}

void OUT_CHAR(char character) // OUT_CHAR from lab 5.
{
	__far_mem_write(LCD_Write, character);
	BF_POLL();
	Delay_XuS(400);
}

void OUT_STRING(char str[]) //Store the characters in strings. OUT_STRING from lab 5.
{
	__far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
	BF_POLL();
	Delay_XuS(400);

	for(int i = 0; i < strlen(str); i++)
	{
		OUT_CHAR(str[i]);

		if(i == 16)
		{
			__far_mem_write(LCD_Start, 0xC0); // Go to second line.
			BF_POLL();
			Delay_XuS(400);
		}
	}
}

int main(void)
{
   TC_INIT();
	LCD_INIT();
	PORTF.DIRCLR = 0x01; //Set port F pin as switch input.
```

```
   while (1)
   {
                        TCE0.CCA = 0x01E4; //The count is always compared with this register.

                        if((PORTF_IN & 0x01) == 0x01)
                        {
                                TCE0.CTRLA = TC_CLKSEL_DIV1_gc; // Turn on the counter which should count/pulse until
the period set in CCA.
                        }
                        else
                        {
                                TCE0.CTRLA = TC_CLKSEL_OFF_gc; // Turn off the counter.
                        }
   }
}
```

Part B:

```
/*
 * Lab6b.c
 *
 * Created: 7/18/2016 4:03:04 PM
 * Author : James Mak
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include "ebi_driver.h"
#include <string.h>
#define F_CPU 2000000
#define CS0_Start 0x8000
#define CS1_Start 0x420000
#define LCD_Start 0x422000
#define LCD_Write 0x422001
#define bsel 107
#define bscale -5

int CMA[] = {0x0719,0x5FF,0x510,0x3C9,0x0301,0x0283,0x01E4};
int SCL[] = {0x03C9,0x0360,0x0301,0x02D6,0x0283,0x0240};

void EBI_INIT(void)
{
        PORTH.DIR = 0x37;
        PORTH.OUT = 0x33;
        PORTK.DIR = 0xFF;

        EBI.CTRL = EBI_CS_MODE_SRAM_gc | EBI_IFMODE_3PORT_gc; //3 port (H,J,K) SRAM mode.

        EBI.CS0.BASEADDRH = (uint8_t) (CS0_Start>>16) & 0xFF;
        EBI.CS0.BASEADDRL = (uint8_t) (CS0_Start>>8) & 0xFF;
        EBI.CS0.CTRLA = EBI_CS_MODE_SRAM_gc | EBI_CS_ASPACE_16KB_gc; //SRAM mode 16k address space.

        EBI.CS1.BASEADDR = (uint16_t) (CS1_Start>>8) & 0xFFFF;
        EBI.CS1.CTRLA = EBI_CS_MODE_SRAM_gc | EBI_CS_ASPACE_64KB_gc; //SRAM mode 64k address space.
}
```

```
void TC0_INIT(void)
{
	//We are using Port E TC because it is the only port available that isn't taken.
	PORTE_DIRSET = 0x01; //Set Pin 1 to output.
	TCE0.CTRLA = TC_CLKSEL_DIV1_gc;  //Turn off the Counter.
	TCE0.CTRLB = TC_WGMODE_FRQ_gc | TC0_CCAEN_bm;  //Waveform FRQ generation mode and Compare
enable A set.
	TCE0.CTRLE = TC_BYTEM_NORMAL_gc; //Normal mode TC.
	TCE0.CTRLFCLR = TC0_DIR_bm;  //Incrementing Counter.
	TCE0.CNT = 0; //Start count at zero.

}

void TC1_INIT(void)
{
	TCE1.CTRLA = TC_CLKSEL_DIV256_gc; //Turn off the Counter.
	TCE1.CTRLB = 0x00; //Normal Waveform mode.
	TCE1.CTRLE = TC_BYTEM_NORMAL_gc; //Normal mode TC.
	TCE1.CTRLFCLR = TC1_DIR_bm; //Incrementing Counter.
	TCE1.INTCTRLA = TC_OVFINTLVL_LO_gc; //Enable low level interrupt.
	PMIC_CTRL = PMIC_LOLVLEN_bm; //Enable low level interrupts.
	sei();
}

ISR(TCE1_OVF_vect)
{
	TCE0_CCA = 0x00;
	TCE1_PER = 0x00; //Set the duration register to zero.
	TCE1_INTFLAGS = 0x01; //Clear the flag.
}

void Delay_XuS(int a) // Delay a certain amount of uS.
{
	for(int i = 0; i < a; i++);
}

void BF_POLL(void)
{
	uint8_t BF = __far_mem_read(LCD_Start);

	while((BF & 0x80) == 0x80) //Poll the busy flag (bit 7 until it is clear.
	{
		BF = __far_mem_read(LCD_Start);
	}
}

void LCD_INIT(void) // LCD_INIT from lab 5.
{
	__far_mem_write(LCD_Start, 0x38); //Configure function.
	BF_POLL();
	__far_mem_write(LCD_Start, 0x0F); //Configure display.
	BF_POLL();
	__far_mem_write(LCD_Start, 0x01); //Clear display.
	BF_POLL();
	__far_mem_write(LCD_Start, 0x06); //Configure entry mode.
	BF_POLL();
```

```
}

void OUT_CHAR(char character) // OUT_CHAR from lab 5.
{
	__far_mem_write(LCD_Write, character);
	BF_POLL();
	Delay_XuS(400);
}

void OUT_STRING(char str[]) //Store the characters in strings. OUT_STRING from lab 5.
{

	Delay_XuS(400);

	for(int i = 0; i < strlen(str); i++)
	{
		OUT_CHAR(str[i]);

		if(i == 16)
		{
			__far_mem_write(LCD_Start, 0xC0); // Go to second line.
			BF_POLL();
			Delay_XuS(400);
		}
	}
}

void UART_INIT(void)
{
	PORTQ.DIRSET = 0x0A;
	PORTQ.OUTCLR = 0x0A;
	PORTD.DIRSET = 0x08;
	PORTD.OUTSET = 0x08;
	PORTD.DIRCLR = 0x04;
	USARTD0.CTRLB = 0x18;
	USARTD0.CTRLC = 0x03;
	USARTD0.BAUDCTRLA = (bsel & 0xFF);
	USARTD0.BAUDCTRLB = ((bscale << 4) & 0xF0) | ((bsel >> 8) & 0x0F );

}

uint32_t IN_CHAR(void)
{
	uint8_t busyflag = USARTD0_STATUS;

	if((busyflag & 0x80) == 0x80)
	{
		return USARTD0_DATA;
	}
	else
		return 0x00;
}

int main(void)
{
	EBI_INIT();
   TC0_INIT();
```

```
        TC1_INIT();
        LCD_INIT();
        UART_INIT();
        uint8_t key = 0x00;

while (1)
{
                key = IN_CHAR();
                switch(key)
                {
                        case 0x30 :
                           __far_mem_write(LCD_Start, 0x01);// Select function clear display, clear display.
                        BF_POLL();
                                OUT_STRING("A6");
                                BF_POLL();
                                __far_mem_write(LCD_Start, 0xC0);
                                BF_POLL();
                                OUT_STRING("1760.00 Hz");

                                TCE0.CCA = 0x0240;
                                TCE1.PER = 0x897;

                        break;
                        case 0x31 :
                                __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
                                BF_POLL();
                                OUT_STRING("C6");
                                BF_POLL();
                                __far_mem_write(LCD_Start, 0xC0);
                                BF_POLL();
                                OUT_STRING("1046.50 Hz");

                                TCE0.CCA = 0x03C9;
                                TCE1.PER = 0x897;

                        break;
                        case 0x32 :
                                __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
                                BF_POLL();
                                OUT_STRING("C#6/Db6");
                                BF_POLL();
                                __far_mem_write(LCD_Start, 0xC0);
                                BF_POLL();
                                OUT_STRING("1108.73 Hz");

                                TCE0.CCA = 0x0394;
                                TCE1.PER = 0x897;

                        break;
                        case 0x33 :
                                __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
                                BF_POLL();
                                OUT_STRING("D6");
                                BF_POLL();
                                __far_mem_write(LCD_Start, 0xC0);
                                BF_POLL();
                                OUT_STRING("1174.66 Hz");
```

```
            TCE0.CCA = 0x0360;
            TCE1.PER = 0x897;

break;
case 0x34 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("D#6/Eb6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1244.51");

            TCE0.CCA = 0x032F;
            TCE1.PER = 0x897;

break;
case 0x35 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("E6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1318.51 Hz");

            TCE0.CCA = 0x0301;
            TCE1.PER = 0x897;

break;
case 0x36 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("F6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1396.91 Hz");

            TCE0.CCA = 0x02D6;
            TCE1.PER = 0x897;

break;
case 0x37 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("F#6/Gb6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1479.98 Hz");

            TCE0.CCA = 0x02AE;
            TCE1.PER = 0x897;

break;
```

```
case 0x38 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("G6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1577.98 Hz");

            TCE0.CCA = 0x0283;
            TCE1.PER = 0x897;

break;
case 0x39 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("G#6/Ab6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1864.66 Hz");

            TCE0.CCA = 0x0262;
            TCE1.PER = 0x897;

break;
case 0x41 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("A#6/Bb6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1864.66 Hz");

            TCE0.CCA = 0x0220;
            TCE1.PER = 0x897;

break;
case 0x42 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("B6");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("1975.53 Hz");

            TCE0.CCA = 0x0201;
            TCE1.PER = 0x897;


break;
case 0x43 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("C7");
```

```
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("2093.00 Hz");

            TCE0.CCA = 0x01E4;
            TCE1.PER = 0x897;

break;
case 0x44 :
            __far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
            BF_POLL();
            OUT_STRING("C#7/Db7");
            BF_POLL();
            __far_mem_write(LCD_Start, 0xC0);
            BF_POLL();
            OUT_STRING("2217.46 Hz");

            TCE0.CCA = 0x01C9;
            TCE1.PER = 0x897;

break;
case 0x2A :
__far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
BF_POLL();
OUT_STRING("C Major Arpeggio");
BF_POLL();

for(int i = 0; i < 7; i++)
{
            TCE0.CCA = CMA[i];
            TCE1.PER = 0x897;

            while((TCE1_INTFLAGS & 0x01) == 0x00)
            {

            }
}

Delay_XuS(400);

for(int j = 7; j > 0; j--)
{
            TCE0.CCA = CMA[j];
            TCE1.PER = 0x897;

            while((TCE1_INTFLAGS & 0x01) == 0x00)
            {

            }
}
break;
case 0x23 :
__far_mem_write(LCD_Start, 0x01);    // Select function clear display, clear display.
BF_POLL();
OUT_STRING("C Major Scale");
BF_POLL();
```

```
                        for(int i = 0; i < 7; i++)
                        {
                                TCE0.CCA = SCL[i];
                                TCE1.PER = 0x897;

                                while((TCE1_INTFLAGS & 0x01) == 0x00)
                                {

                                }
                        }
                        break;

                        default:
                        break;
                }
        }
}
```

Appendix:

C7 Waveform