Lab 7

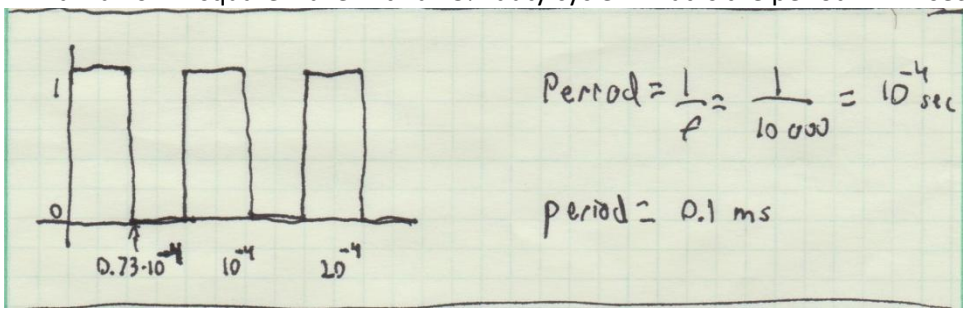
Reyna, Ricardo

Section 75C9

07/24/2015

- Prelab Questions:

1. Draw a 10kHz square wave with a 73% duty cycle. What is the period in miliseconds(ms)?




2. 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.

*The limiting factor is the amount of bits the CNT can contain. The XMEGA can generate lower frequencies more precise than higher ones, because higher frequencies overflow the TC faster than lower frequencies.*

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

*The prescaler divides the clock by a value before it's fed to the TC. The counts are stored in the CNT register.*

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 waveforms using your XMEGA? Do you need to add any extra hardware?

*The difference is that in frequency mode the period of the wave is controlled by the CCA register; meanwhile in the Single/Dual Slope PWM the period is controlled by the value of the PER register. The PWM mode can be used to emulate the Frequency Generation mode when the duty cycle is set to 50%. Sine waves can be generated by connecting a custom circuit that takes PWD as an input and outputs as a variable voltage. The circuit is found on the LCD specs.*

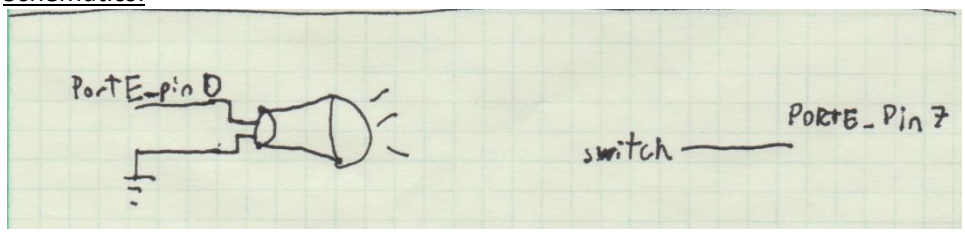
- Problems Encountered:

None in this lab.

- Future Work/Applications:

In this we used timers to generate different frequencies to produce sounds. We also combined the timers with interrupts.

- Schematics:




- Decoding Logic:
  None in this lab

- Pseudo code:
  Part A:
  initialize switch
  initialize speaker

  check if the switch is on to play sound

  PartB:
  Initialize everything
  Use an interrupt on the timer.
  Check if * or # to play the song, otherwise play a single note
  If the note is selected also display it on the lcd
  If a song is play feed the string into a function that will take each char into another function to play each note individually
  Repeat infinitely

- Program Code:
  Part A:

```
/*
* Lab7_A_RSR.c
*
* Created: 7/20/2015 4:20:45 PM
*  Author: stefano92
Lab 6 part A
Name: Ricardo Stefano Reyna
Section#: 75C9
TA: Khaled Hassan
Description: This program is to check the timer and check the output sound.
*/


#include <avr/io.h>
#include <avr/interrupt.h>
#include "ebi_driver.h"
#define F_CPU 2000000
#define CS0_Start 0x4000
#define CS1_Start 0x1B0000
#define LCD_COM 0x1B1000
#define LCD_DAT 0x1B1001

void init_timer();
```

```
void init_switch();

int main(void)
{
	init_switch();
	init_timer();

	uint8_t input_check;

	while(1) {
		//Checks when the switch is on or off
		input_check = PORTE.IN & 0x80;
		if(input_check == 0x80) {
			TCE0.CTRLB = 0x11;
		}

		else {
			TCE0.CTRLB = 0x01;
		}
	}

	return 0;
}

void init_timer() {
	PORTE.DIRSET = 0x01;
	TCE0.CNT = 0x00;
	TCE0.CTRLA = 0x01;
	TCE0.CTRLB = 0x11;
	TCE0.CCA = 0x23B;
}

void init_switch() {
	PORTE.DIRCLR = 0x80;
}


	PartB:
/*
 * Lab7_B_RSR.c
 *
 * Created: 7/21/2015 3:17:24 PM
 *  Author: stefano92
  Lab 7 part B
  Name: Ricardo Stefano Reyna
  Section#: 75C9
  TA: Khaled Hassan
  Description: This program depending on the keypad will select a note to be played or a
full song.
 */


#include <avr/io.h>
#include <avr/interrupt.h>
#include "ebi_driver.h"
#define F_CPU 2000000
#define CS0_Start 0x4000
#define CS1_Start 0x1B0000
```

```c
#define LCD_COM 0x1B1000
#define LCD_DAT 0x1B1001

void init_EBI();
void init_lcd();
void check_BF();
void out_string(char *str);
void init_timer();
void init_SP();
void k_init();
char get_key();
void soundStop();
void delay();
void checknotes(char note, int stop);

void noise(char note);
void music(char* song, int stop);

int main(void)
{
	init_SP();
	init_EBI();
	init_timer();
	init_lcd();
	k_init();

	char input = '&';

	char *song1 = "03603601B86803153"; //Song of time
	char *song2 = "365130p365136p031"; //Lugia's song
	int time1 = 20000;
	int time2 = 25000;

    while(1)
    {
		input = get_key();
		//This checks if you want to play a song or just a beep
		switch(input) {
			case '*':
				__far_mem_write(LCD_COM, 0x01);
				check_BF();
				out_string("Song of Time");
				__far_mem_write(LCD_COM, 0xC0);
				check_BF();
				out_string("Ocarina of Time");
				music(song1, time1);
				break;
			case '#':
				__far_mem_write(LCD_COM, 0x01);
				check_BF();
				out_string("Lugia's song");
				__far_mem_write(LCD_COM, 0xC0);
				check_BF();
				out_string("Pokemon");
				music(song2, time2);
				break;
			default:
				noise(input);
```

```
                        break;
                }
        }
}

void init_EBI() {
        PORTH_DIR = 0x37;
        PORTH_OUT = 0x33;
        PORTK_DIR = 0xFF;

        EBI.CTRL = EBI_SRMODE_ALE1_gc | EBI_IFMODE_3PORT_gc;            // ALE1
multiplexing, 3 port configuration

        EBI.CS0.BASEADDRH = (uint8_t) (CS0_Start>>16) & 0xFF;
        EBI.CS0.BASEADDRL = (uint8_t) (CS0_Start>>8) & 0xFF;            // Set CS0 range
to 0x004000 - 0x004FFF
        EBI.CS0.CTRLA = EBI_CS_MODE_SRAM_gc | EBI_CS_ASPACE_4KB_gc;       // SRAM mode, 4k
address space

        // BASEADDR is 16 bit (word) register. C interface allows you to set low and high
parts with 1
        // instruction instead of the previous two
        EBI.CS1.BASEADDR = (uint16_t) (CS1_Start>>8) & 0xFFFF;          // Set CS1 range
to 0x1B0000 - 0x1BFFFF
        EBI.CS1.CTRLA = EBI_CS_MODE_SRAM_gc | EBI_CS_ASPACE_64KB_gc;
}


void init_lcd() {
        check_BF();
        __far_mem_write(LCD_COM,0x38);
        check_BF();
        __far_mem_write(LCD_COM,0x0F);
        check_BF();
        __far_mem_write(LCD_COM,0x01);
        check_BF();

}

void k_init(){
        PORTF_PIN4CTRL = 0x18;
        PORTF_PIN5CTRL = 0x18;
        PORTF_PIN6CTRL = 0x18;
        PORTF_PIN7CTRL = 0x18;
        PORTF_DIRSET = 0x0F; //Low bits are outputs
        PORTF_DIRCLR = 0xF0; //High bits are inputs
}

void check_BF() {
        volatile uint8_t temp_val = 0;
        while(1) {
                temp_val = __far_mem_read(LCD_COM);
                if((temp_val & 0x80) == 0x00) {
                        break;
                }
        }
}
```

```
void out_string(char *str){
	int cntr = 0;
	//go through each char until null
	while(*str!= 0) {
		check_BF();
		//go to the next line when ends the line
		if(cntr == 16)
		{
			__far_mem_write(LCD_COM, 0xC0);
			check_BF();
		}
		check_BF();
		__far_mem_write(LCD_DAT,*str);
		str++;
		cntr++;
	}
	check_BF();
}

void init_timer() {
	TCE1.CTRLA = 0x05;
	TCE1.CTRLB = 0x10;
	TCE1.INTCTRLA = 0x02;
	TCE1.INTCTRLB = 0x00;

	PMIC_CTRL = 0x02;

	sei();
}

void init_SP() {
	PORTE.DIRSET = 0x01; //making portE pin2 as an output
	TCE0_CNT = 0x00;		//this is where the count is stored, we are resetting
it. (CNT=PER)
	TCE0_CTRLA = 0x00;		//setting clk to on
	TCE0_CTRLB = 0x11;		//enables CCA (FRQ mode, see 14.8.2), frequency mode
}

void soundStop(int num) {
	TCE1.CNT = 0x00;
	TCE1.PER = num;
	//Setting the PER depending on the time passed in
	while (TCE1.PER > TCE1.CNT)
	{
		TCE0.CTRLA = 0x01;
	}
}

//Interrupt for the timer
ISR(TCE1_OVF_vect) {
	TCE0.CTRLA = 0x00;
}


//Keypad
char get_key(){
	uint8_t key;
	PORTF_OUT = 0x0E;
```

```
asm("nop");
key = PORTF_IN;
if(key == 0xEE){
	return '1';
}
if(key == 0xDE){
	return '4';
}
if(key == 0xBE){
	return '7';
}
if(key == 0x7E){
	return '*';
}

PORTF_OUT = 0x0D;
asm("nop");
key = PORTF_IN;
if(key == 0xED){
	return '2';
}
if(key == 0xDD){
	return '5';
}
if(key == 0xBD){
	return '8';
}
if(key == 0x7D){
	return '0';
}
PORTF_OUT = 0x0B;
asm("nop");
key = PORTF_IN;
if(key == 0xEB){
	return '3';
}
if(key == 0xDB){
	return '6';
}
if(key == 0xBB){
	return '9';
}
if(key == 0x7B){
	return '#';
}
PORTF_OUT = 0x07;
asm("nop");
key = PORTF_IN;
if(key == 0xE7){
	return 'A';
}
if(key == 0xD7){
	return 'B';
}
if(key == 0xB7){
	return 'C';
}
if(key == 0x77){
```

```
                return 'D';
        }

        return '&';
}

void noise(char note) {
        int timenote = 15000;
        //Display on the lcd and make the noise
        switch (note)
        {
                delay();
                case '1':
                        TCE0.CCA = 0x3C1; //1046.50
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("C6");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("1046.50 Hz");
                        soundStop(timenote);
                        break;
                case '2':
                        TCE0.CCA = 0x38B; //1108.73
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("C#6/Db6");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("1108.73 Hz");
                        soundStop(timenote);
                        break;
                case '3':
                        TCE0.CCA = 0x358; //1174.66
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("D6");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("1174.66 Hz");
                        soundStop(timenote);
                        break;
                case '4':
                        TCE0.CCA = 0x328; //1244.51
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("D#6/Eb6");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("1244.51 Hz");
                        soundStop(timenote);
                        break;
                case '5':
                        TCE0.CCA = 0x2FA; //1318.51
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("E6");
                        __far_mem_write(LCD_COM, 0xC0);
```

```
        check_BF();
        out_string("1318.51 Hz");
        soundStop(timenote);
        break;
case '6':
        TCE0.CCA = 0x2D0; //1396.91
        __far_mem_write(LCD_COM, 0x01);
        check_BF();
        out_string("F6");
        __far_mem_write(LCD_COM, 0xC0);
        check_BF();
        out_string("1396.98 Hz");
        soundStop(timenote);
        break;
case '7':
        TCE0.CCA = 0x2A7; //1479.98
        __far_mem_write(LCD_COM, 0x01);
        check_BF();
        out_string("F#6/Gb6");
        __far_mem_write(LCD_COM, 0xC0);
        check_BF();
        out_string("1479.98 Hz");
        soundStop(timenote);
        break;
case '8':
        TCE0.CCA = 0x281; //1567.98
        __far_mem_write(LCD_COM, 0x01);
        check_BF();
        out_string("G6");
        __far_mem_write(LCD_COM, 0xC0);
        check_BF();
        out_string("1567.98 Hz");
        soundStop(timenote);
        break;
case '9':
        TCE0.CCA = 0x25D; //1661.22
        __far_mem_write(LCD_COM, 0x01);
        check_BF();
        out_string("G#6/Ab6");
        __far_mem_write(LCD_COM, 0xC0);
        check_BF();
        out_string("1661.22 Hz");
        soundStop(timenote);
        break;
case '0':
        TCE0.CCA = 0x23C; //1760.00
        __far_mem_write(LCD_COM, 0x01);
        check_BF();
        out_string("A6");
        __far_mem_write(LCD_COM, 0xC0);
        check_BF();
        out_string("1760.00 Hz");
        soundStop(timenote);
        break;
case 'A':
        TCE0.CCA = 0x21B; //1864.66
        __far_mem_write(LCD_COM, 0x01);
        check_BF();
```

```
                        out_string("A#6/Bb6");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("1864.66 Hz");
                        soundStop(timenote);
                        break;
                case 'B':
                        TCE0.CCA = 0x1FD; //1975.53
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("B6");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("1975.53 Hz");
                        soundStop(timenote);
                        break;
                case 'C':
                        TCE0.CCA = 0x1E0; //2093.00
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("C7");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("2093.00 Hz");
                        soundStop(timenote);
                        break;
                case 'D':
                        TCE0.CCA = 0x1C5; //2217.46
                        __far_mem_write(LCD_COM, 0x01);
                        check_BF();
                        out_string("C#7/Db7");
                        __far_mem_write(LCD_COM, 0xC0);
                        check_BF();
                        out_string("2217.46 Hz");
                        soundStop(timenote);
                        break;
                default:
                        break;
        }
}

void music (char* song, int stop) {
        //Grabs each char individually to pass each note
        while (*song != 0x00) {
                checknotes(*song, stop);
                song++;
        }
}

void checknotes(char note, int stop) {
        //I use this for the songs and get each note individually, I algo include a pause
        switch (note)
        {
                delay();
                case '1':
                        TCE0.CCA = 0x3C1; //1046.50
                        soundStop(stop);
                        break;
```

```
case '2':
       TCE0.CCA = 0x38B; //1108.73
       soundStop(stop);
       break;
case '3':
       TCE0.CCA = 0x358; //1174.66
       soundStop(stop);
       break;
case '4':
       TCE0.CCA = 0x328; //1244.51
       soundStop(stop);
       break;
case '5':
       TCE0.CCA = 0x2FA; //1318.51
       soundStop(stop);
       break;
case '6':
       TCE0.CCA = 0x2D0; //1396.91
       soundStop(stop);
       break;
case '7':
       TCE0.CCA = 0x2A7; //1479.98
       soundStop(stop);
       break;
case '8':
       TCE0.CCA = 0x281; //1567.98
       soundStop(stop);
       break;
case '9':
       TCE0.CCA = 0x25D; //1661.22
       soundStop(stop);
       break;
case '0':
       TCE0.CCA = 0x23C; //1760.00
       soundStop(stop);
       break;
case 'A':
       TCE0.CCA = 0x21B; //1864.66
       soundStop(stop);
       break;
case 'B':
       TCE0.CCA = 0x1FD; //1975.53
       soundStop(stop);
       break;
case 'C':
       TCE0.CCA = 0x1E0; //2093.00
       soundStop(stop);
       break;
case 'D':
       TCE0.CCA = 0x1C5; //2217.46
       soundStop(stop);
       break;
case 'p':
       TCE0.CCA = 0x01; //pause
       soundStop(10000);
       break;
default:
       break;
```

```
        }
}

void delay () {
        volatile uint32_t ticks;            // Volatile prevents compiler optimization
        for(ticks=0;ticks<=10000;ticks++);  // Convinient delay
}
```

- Appendix: