RC Filters

Pre-lab Questions
1. What are some typical uses of RC filter circuits?
2. Given a low-pass filter with a 2 KΩ resistor and a 0.2 µF capacitor, find the corner frequency.
3. Design an RC filter that removes the high frequency noise from the signal seen below.

<table>
<thead>
<tr>
<th>Voltage, V</th>
<th>Time, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1.5</td>
</tr>
<tr>
<td>-0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>-1.5</td>
<td>0</td>
</tr>
</tbody>
</table>

Purpose
- To introduce resistors and capacitors
- To introduce the concept, construction, and utility of low-pass and high-pass RC filters
- To introduce the Atmega 128 microcontroller and its capability to output tones.

Components

<table>
<thead>
<tr>
<th>Qty.</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Atmel Atmega 128 microcontroller with STK500 and STK501 interface boards</td>
</tr>
<tr>
<td>1</td>
<td>Programming cable and power supply.</td>
</tr>
<tr>
<td>1 ea.</td>
<td>1 kΩ and 10 kΩ resistors</td>
</tr>
<tr>
<td>1</td>
<td>0.1µF capacitor</td>
</tr>
<tr>
<td>1</td>
<td>10µF capacitor</td>
</tr>
<tr>
<td>1</td>
<td>piezo speaker (Murata PKM17EPP-2002-130 (Mouser 81-PKM17EPP-2002))</td>
</tr>
</tbody>
</table>

Introduction
Resistor-Capacitor Filter Circuit

You will begin this laboratory using the function generator connected to a simple circuit consisting of a resistor and capacitor. Details of the construction are given in the Procedure below. First, consider the circuit shown in Figure 1. The impedance of a capacitor is 1/(jωC), a complex quantity, where j is the square root of -1, ω is frequency in rad/sec (2π rad/sec = 1Hz), and C is capacitance in Farads. Equation 1 is derived using the voltage divider relationship.
Equation 2 solves equation 1 for the ratio of $V_o$ to $V_i$, which is called the ‘transfer function’ of the filter.

Equation 2

$$\frac{V_o}{V_i} = 1 + j\omega RC$$

The transfer function tells how much of the input voltage is ‘transferred’ to the output.

**What happens to the value of the transfer function as the frequency of the input voltage increases? Does it increase or decrease?** For a given sinusoidal input voltage $V_i$, the magnitude of the output voltage $V_o$ is given by the product of $V_i$ and the magnitude of the Transfer Function (evaluated at the frequency of $V_i$).

We will verify equation 2 by the following experiment.

**Procedure**

1. Construct the single-stage RC Low-pass filter circuit shown in Figure 1.

2. On the function generator, set the amplitude of $V_i$ to 5 volts peak-to-peak ($V_{p-p}$) at 500 Hz (what termination should you set the function generator for, 50 ohms or High Z?), and record the amplitude of $V_o$. Repeat this measurement with the frequency of $V_i$ set to 1.6 kHz and 10 kHz. Compare the ratio of the input voltage and the output voltage (i.e., $V_o/V_i$) with the magnitude of the transfer function evaluated at the corresponding frequency.

**What can you conclude from this comparison about the relationship between the magnitude of the transfer function and the ratio of the input and output voltages? Explain. Why is this circuit called a “low-pass filter”?**

The phase lag of a circuit can be thought of as the amount of delay in the signal as it goes through the circuit. Phase lag is measured in degrees as shown in Figure 2. The phase lag of the circuit shown in Figure 2 is the angle of the transfer function, $1/(1+j\omega CR)$, evaluated at the frequency of the signal. We will verify this relation by the following experiment.
3. Set the scope to dual-trace mode. Display both \( V_i \) and \( V_o \) on the scope at the same time. (Ask the TA for help if needed). Measure the phase lag of this circuit as shown in Figure 2 at \( V_i = 1.6 \) kHz. **Does the theoretical value match the observed value?**

![Figure 2](image)

**Figure 2** Phase lag of \( V_o \). The output voltage is delayed (shifted later in time). The amount of time shift depends on the value of \( R \) and \( C \), and the frequency of the input signal. One period of the waveform represents 360º, so the phase shift is just the percentage of time delay multiplied by 360º.

4. Construct the single-stage RC High-pass filter circuit shown in Figure 3, and verify (using the voltage division formula) that \( V_o/V_i \) for this circuit is as follows:

\[
V_o = \frac{R}{R + 1/(j\omega C)} V_i \quad (3)
\]

\[
\frac{V_o}{V_i} = \frac{j\omega RC}{1 + j\omega RC} \quad (4)
\]

5. Repeat step (2) and (3) for this high-pass filter.

![Figure 3](image)

**Figure 3** High-pass filter. The resistor and capacitor together function as a frequency-dependent voltage divider. Note that the locations of the resistor and capacitor are interchanged in comparison to the low-pass filter shown in Figure 1.

Add a DC offset (any value) to the function generator output. **Does this DC offset affect the output (\( V_o \)) of the filter?** This is the reason that a capacitor can be thought of as a device that blocks a DC component, but allows the AC component to go through. This is called “AC coupling.”
Using the Atmega 128 PWM

Now we want to use the Atmega 128 microcontroller to output a tone and see the effect of filtering a Pulse Width Modulated (PWM) signal.

6. Build the circuit shown below in Figure 4 on the solderless breadboard. Use a jumper wire (female socket on one end, stripped wire at the other end) to connect the Atmega 128 pin PB5 to the solderless breadboard. Note that the 10 \( \mu \)F capacitor is POLARIZED. If you look at the side of the capacitor body, you will see one leg marked with a minus sign. IT IS EXTREMELY IMPORTANT THAT THE MINUS LEG ALWAYS BE KEPT AT LOWER POTENTIAL THAN THE OTHER LEG! If you reverse the polarity, the capacitor can explode, so take extra precaution, and double-check how you wire the capacitors before you apply power to your circuit. Not all capacitors are polarized, but electrolytic (‘lytic for short) capacitors are.

![Atmega 128 circuit with RC Filter](image)

**Figure 4** Atmega 128 circuit with RC Filter. The ‘scope is used to display the output from the Atmega 128 before and after the filter. Note that the ground shown is the common ground of the power supply. One of the ground clips from the ‘scope probes must be connected to the power supply common ground.

Using Programmer’s Notepad, start a new project called ‘RCfilter’. Copy the global.h file and the makefile that you used from the Introduction to the Atmega 128 lab, and save them in the folder with the new RCfilter project you just created. Add global.h and the makefile to the RC filter project. Create a new C file consisting of the listing below. Add this file to the project. Modify the makefile, so that the compiler can find the target and source files. Download the program to the Atmega 128 microcontroller:

```c
// Pulse Width Modulation (PWM) program
// causes the duty cycle of the output on pin PB5
// to continuously vary

#include <avr/io.h> // include I/O definitions (port names, pin names, etc)
#include <avr/signal.h> // include "signal" names (interrupt names)
#include <avr/interrupt.h> // include interrupt support
#include "global.h" // include our global settings
#include "uart.h" // include uart function library
#include "printf.h" // include printf function library
#include "vt100.h" // include VT100 terminal support
#include "timer.h" // include timer function library (timing, PWM, etc)
#include "pulse.h" // include pulse output support

#include <avr/interrupt.h>

void init(void);

int main(void)
```


```c
{
    char myReceivedByte;
    u16 freq = 440; // default freq is 440Hz
    init(); // initialize everything
    vt100ClearScreen(); // clear the terminal screen
    rprintf("Welcome to the RC Filter Lab\n\r");
    while(1)
    {
        if(uartReceiveByte(&myReceivedByte))
        {
            switch(myReceivedByte)
            {
            case 93:  // 93 is the ASCII character code for ]
                freq++;
                break;
            case 91:   // 91 is the ASCII character code for [
                freq--;
                break;
            case 112:  // 112 is the ASCII character code for p
                vt100SetAttr(VT100_ATTR_OFF);
                rprintf("Frequency is %d Hz\r",freq);
                break;
            default:  // if any other key is pressed
                vt100SetAttr(VT100_BLINK);
                rprintf("Invalid Command!\r");
            }
            pulseT1ASetFreq(freq); // set the output frequency
            pulseT1ARun(0);  // set number of pulses to run 0 = continuous
        }
    }
    return 0;
} // end main()

void init(void)
{
    // initialize our libraries
    timerInit(); // initialize the timer system
    uartInit();  // initialize the UART (serial port)
    uartSetBaudRate(9600); // set the baud rate of the UART
    rprintfInit(uartSendByte); // set rprintf to print to serial port
    vt100Init();   // initialize vt100 library
    // Setup I/O
    // set pulse pins as output
    sbi(DDRB, 5);   // OC1A pin
    sbi(DDRB, 6);   // OC1B pin
    pulseInit();   // initialize pulse library
} // end init()
```

7. Open a HyperTerminal window (Start → Programs → Accessories → Communications → HyperTerminal). Remember, power down the STK500, and switch the serial cable from RS232 CNTRL to RS232 SPARE, then power up the board, and press the reset button.

8. Connect the leads of the piezo speaker to PB5 and common ground. You should hear an audible tone.

9. Look at the signal on channels 1 and 2 using the oscilloscope. Grab the 'scope trace showing both waveforms using Benchlink.
10. Describe what is happening to the signals on channel 1 and 2 and the speaker when you press the ‘]’ or ‘[ ‘ keys.

11. Press the “p” key to have the program print the current frequency. **How does this compare to the oscilloscope trace of channel 1?**

12. Modify main() so that your HyperTerminal window looks like Figure 5. Have your program update the frequency display every time you press either [ or ].

![Figure 5](image)

**Figure 5** HyperTerminal Output. Modify your program to update the frequency display if you press [ or ].

13. The VT100 and rprintf libraries have several useful functions for outputting and formatting text to the terminal window. Copy the vt100.h and rprintf.h header files from AVRlib, and save them in your project directory. Explore these libraries by opening them and becoming familiar with their functions. **Change your program to display your frequency and error messages in BOLD and REVERSE.**

**Questions**

Questions are in bold letters in Procedures.