RC Filters and Basic Timer Functionality

Learning Objectives:

The student who successfully completes this lab will be able to:

- Build circuits using passive components (resistors and capacitors) from a schematic diagram.
- Explain the concept of a high or low-pass filter and what its corner frequency is
- Construct a low-pass or high-pass RC filter with a given corner frequency.
- Explain terms associated with counter/timers, such as clock, top, bottom, overflow, and clock prescaler.
- Explain how the hardware timer integrated with an ATmega16 can be used to output a square wave with a given frequency.

Components:

Qty.	Item
1	Atmel ATmega16 microcontroller with STK500 interface boards
1	Programming cable and power supply
1	Solderless breadboard
1	1 kΩ resistor
1	0.1μF capacitor (may be polarized)
1	10μF capacitor (electrolytic, so it is polarized)
1	Piezo speaker (Murata PKM17EPP-2002-130 (Mouser 81-PKM17EPP-2002))

Introduction

RC Filter Theory

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, Z_C, is $1/(j\omega C)$, 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 can be derived from the voltage divider relationship formed by the resistor and capacitor. Equation 2 solves equation 1 for the ratio of Vo to Vi (V_O/V_i), which is called the 'transfer function' of the filter.

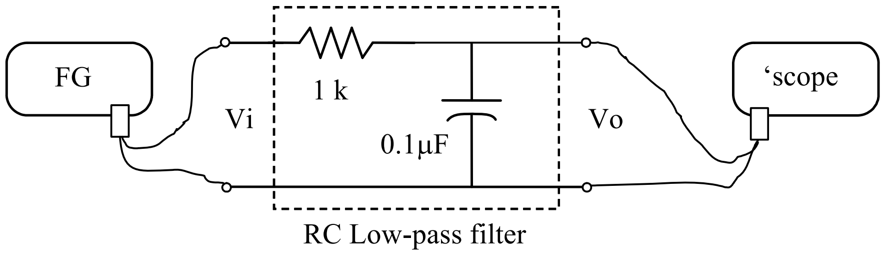

Figure 1. RC low-pass filter. The resistor and capacitor function together as a *frequency-dependent* voltage divider. The function generator will provide a time varying signal whose frequency can be varied, and the oscilloscope will enable you to measure the signal at the output of the circuit. Make sure that you use 'scope probes with the 'scope. Ask your lab instructor if you are unsure about which probes are 'scope probes.

$$Vo = \frac{1/(j\omega C)}{R + 1/(j\omega C)} Vi \qquad \textbf{(1)} \qquad\qquad \frac{Vo}{Vi} = \frac{1}{1 + j\omega RC} \qquad \textbf{(2)}$$

The transfer function is a measure of how much of the input voltage is 'transferred' to the output. **<u>What happens to the value of the transfer function as the frequency of the input voltage increases? Does it increase or decrease?</u>** For a given sinusoidal input voltage Vi, the magnitude of the output voltage Vo is given by the product of Vi and the magnitude of the transfer function (evaluated at the frequency of Vi).

An important property of filters is the corner frequency, or cut-off frequency. This is the frequency at which the attenuation of the signal through the filter starts to increase sharply. For a passive RC filter, (high or low-pass) the corner frequency is **1/RC** radians/sec. By substituting ω=1/RC into equation 2, you can see that the denominator goes to 1+1j, and consequently the magnitude of Vo/Vi becomes 0.707.

We will verify equation 2 by the following experiment.

<u>Procedure:</u>

[Note: the lab stations are set up so that half of the lab team can work on the circuit on the left side of the bench, and the other half of the team can work on programming the microcontroller on the right side with the computer. Be aware that exams will include information on both aspects, so all team members should switch roles and be familiar with what the other team members are doing.]

1. Construct the <u>single-stage RC Low-pass filter circuit</u> shown in Figure 1 above. Pay attention to the connection of the leads of the capacitor. The curved line on schematic shows that the capacitor is ***polarized*** (which may or may not be the case depending on the type of capacitor you were given by your lab instructor). Not all capacitors are polarized, but electrolytic ('lytic, for short) capacitors are. 'Lytics look like little aluminum cans. If you look at the side of the can, you should see a minus sign. The lead closest to the minus sign corresponds to the lead with the curved line on the schematic. It will also be the shorter of the two leads on a polarized capacitor that has not had its leads trimmed. **It is extremely important that the lead nearest the minus sign always be kept at a lower potential (toward ground) than the other lead of the capacitor.** If you reverse the polarity, the capacitor can explode, so take extra precaution, and double-check how you have wired the capacitors before you apply power to your circuit. There are other capacitors, such as ceramic capacitors, which are not polarized, and the connection to the leads are interchangeable. Discern what kind of capacitor you have, and connect it accordingly!

2. On the function generator, output a sine wave, and set the amplitude of Vi to 5 volts peak-to-peak (V_{p-p}) at 500 Hz (What termination should you set the function generator output to, 50 ohms or **High Z**?). Measure and record the amplitude of Vi and Vo using the 'scope. Repeat these measurements with the frequency of Vi set to 1.6 kHz and then 10 kHz. Compare the ratio of the input voltage and the output voltage (i.e., Vo/Vi) with the magnitude of the transfer function (from equation 2) evaluated at the corresponding frequency. **<u>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"?</u>**

   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. Recall that since the transfer function from equation 2 is a complex quantity, you can think of it as a vector in the complex plane. As you know, vectors have magnitude and direction. The phase lag of the circuit shown in Figure 2 is the angle of the transfer function vector, 1/(1+jωCR), evaluated at the frequency of the signal with respect to the real axis in the complex plane. We will verify this relation by the following experiment:

3. Set the 'scope to dual-trace mode. Connect two 'scope probes to the circuit with the same grounding points, but have one measure the output of the function generator and the other measure the voltage across the capacitor. Using the 'scope, measure the phase lag of this circuit of Figure 2 at Vi=1.6 kHz. The following steps may help you make this measurement:
   1). Use the position knob on the 'scope to center each channel around the horizontal (time) axis
   2). Change the volts/div adjustment to maximize the size of the signals on the screen
   3). Adjust the Horizontal Delay to center a peak (or zero crossing) of the Vi waveform on the vertical axis
   4). Using Measure buttons, determine the period of the Vi waveform
   5). Use the Cursor button and cursors for t1 and t2 to determine Δt between the peaks (or zero crossings) of Vi and Vo

   Use Intuilink software or other means to take a picture of the 'scope trace for the lab report. **Does the theoretical value match the observed value?**

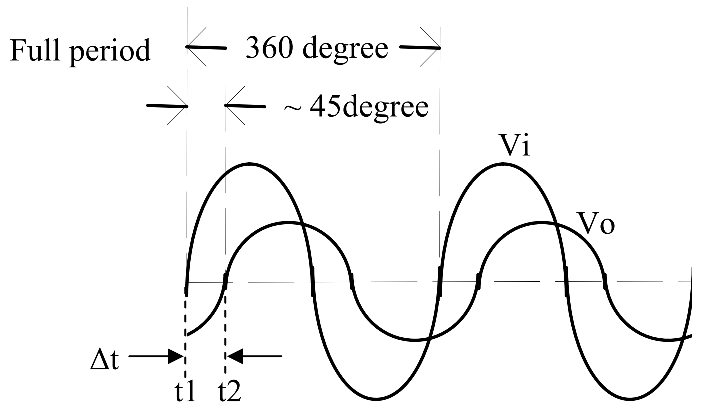

Figure 2. Phase lag of Vo. The output voltage is delayed (shifted later in time). The amount of time shift depends on the value of R and C, and the frequency of the input signal. One period of the waveform represents 360º, so the phase shift is just the percentage of time delay multiplied by 360º. You can use the time cursors on the 'scope to help in making the phase lag measurement.

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

$$Vo = \frac{R}{R + 1/(j\omega C)} Vi \qquad \textbf{(3)} \qquad\qquad \frac{Vo}{Vi} = \frac{j\omega RC}{1 + j\omega RC} \qquad \textbf{(4)}$$

5. Repeat steps 2 and 3 for the high-pass filter in Figure 3. Note that Vo is measured across the resistor in this case.

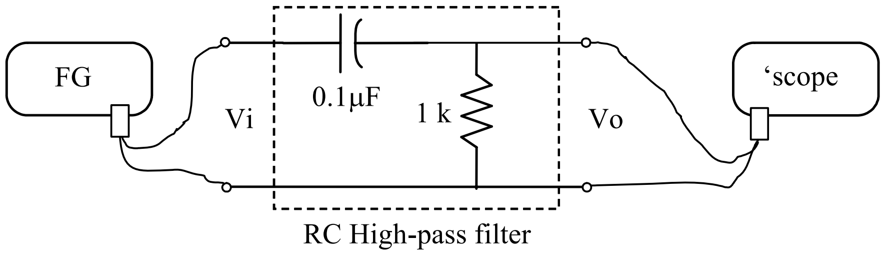

Figure 3. High-pass filter. The resistor and capacitor together function as a frequency-dependent voltage divider. Note that the locations of the resistor and capacitor are interchanged in comparison to the low-pass filter shown in Figure 1.

Add a DC offset (any value) to the function generator output. **Does this DC offset affect the output (Vo) of the high-pass filter? If so, how? Record your measurements for Vi and Vo after you add the DC offset.** 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 ATmega16 Timer to Output a Frequency

Basic introduction to IC timers/counters

One of the most common and useful peripheral devices in a microcontroller is the timer/counter. These peripherals can be used to count pulses of a signal. Timer/counters are extremely useful for keeping time and outputting signals when paired with a **clock signal.** A clock signal is a very uniform set of pulses generated by a device that occur at a constant frequency, such as that shown in Figure 4 below. Frequently these signals are derived from crystals or RC oscillators tuned to provide a particular output frequency. You have probably heard of the 'clock speed' of a computer or other microprocessor. Microcontrollers, timer/counters, and many digital integrated circuits (ICs) need a clock signal to function. You can think of the clock signal like the heart beat of a sequential logic device.

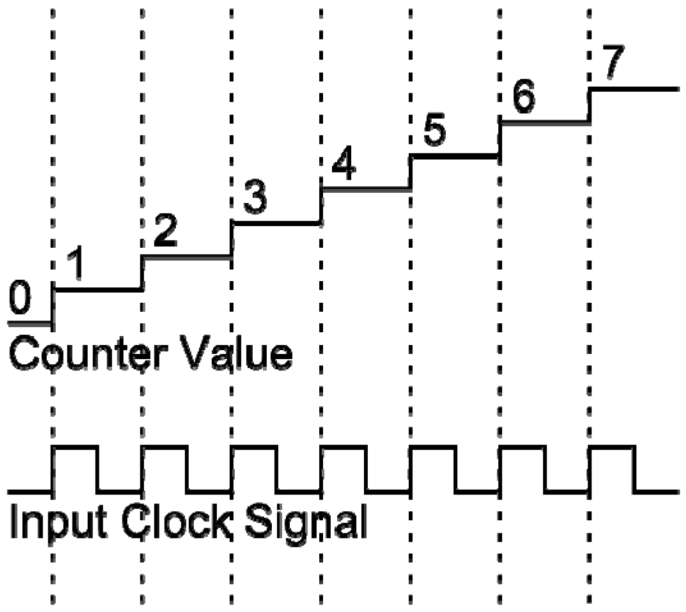

Figure 4. Clock signal input to a counter and the resulting count value as a function of time. Each time the clock signal has a rising edge, the counter's value increases by one.

A timer is a specialized counter, which is given a clock signal. The device increments an internal *register* (think of a register like a variable in a computer program) each time the clock signal changes. The counter increases until it reaches a maximum value known as the **TOP** value of the counter. The most basic counters are unidirectional, that is they will only count up or down, starting from either the TOP value or the BOTTOM (minimum) value. A unidirectional counter (usually called either an ‘up-counter’ or a ‘down-counter’) will **overflow** on the next clock pulse after it has reached its top or bottom value, and it will then reset to the respective opposite value. There are also “up-down counters”, which will count to the TOP value, and then back down to the BOTTOM value. Because counters exist in the digital world, their counts are represented in binary form, and as a result, people talk about the *resolution* of a counter in terms of the size of their internal register in number of bits.

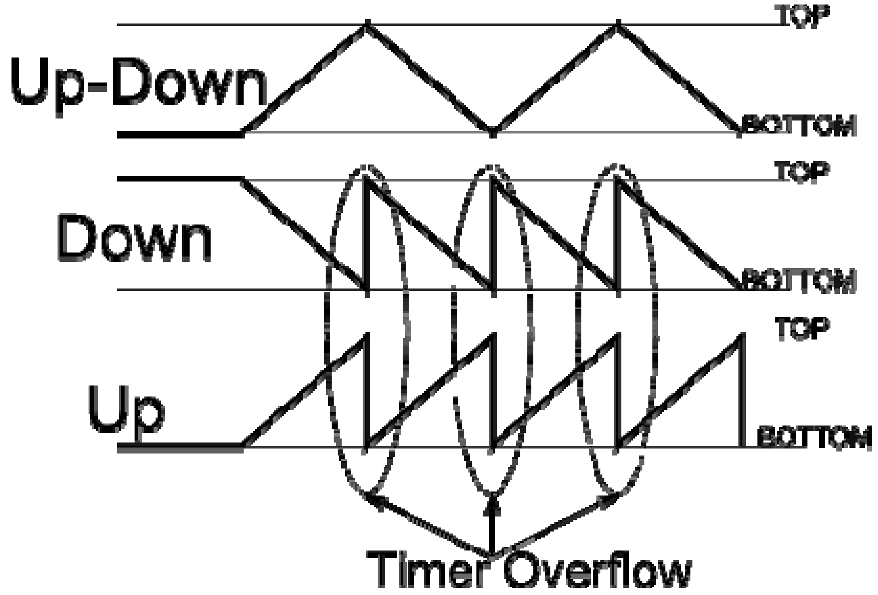

Figure 5. The three main types of counters and their count values as a function of time. Note that 'up' and 'down' counters will loop around from top to bottom, this is called an overflow.

The ATmega16 microcontroller features three built-in timer/counters. These are independent circuit units in the microcontroller, and therefore they can function in a fairly autonomous fashion. Your program code can set up the various attributes of the counter, such its input source (be it the chip’s on board clock or an outside clock signal), how it should count, as well as whether it should behave as an up, down, or up-down counter. The ATmega16 features two 8-bit counters and one 16-bit counter, and the resolution of the counter can also be modified by the use of a prescaler. A **prescaler** allows one to scale down the clock signal by a specified amount. For example, in this lab we will use a pre-scaler of 8. This means that only every 8th clock pulse will cause the counter to be incremented. Thus, coupled with our 8Mhz clock, this means that the counter will increment at a rate of 1MHz. Consequently, one counter tick is a millionth of second, so 1,000 counter ticks is 0.001 seconds, etc. **If we are using a 16-bit counter and the above mentioned clock and prescaler, how much time will it take for the counter to reach it maximum value and then overflow?**

Now we want to use the ATmega16 microcontroller to output a tone and see the effect of filtering the resulting signal.

1. Build the circuit shown below in Figure 6 on the solderless breadboard. Use a jumper wire (female socket on one end, stripped wire at the other end or kluge an equivalent using a 2-wire female jumper and a short length of hookup wire stripped at both ends) to connect the ATmega16 pin PD4 to the solderless breadboard. Note that the 10 μF capacitor is POLARIZED. Remember to look for the minus sign on the side, and connect the corresponding lead appropriately in the circuit.

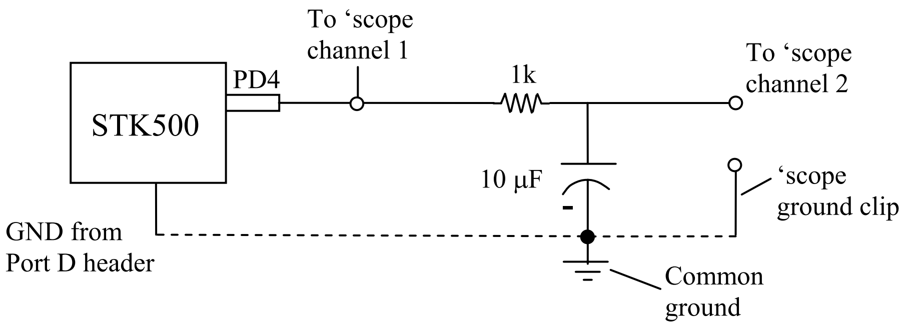

Figure 6. ATmega16 circuit with RC Filter. The 'scope is used to display the output from the ATmega16 before and after the filter. Note that the ground shown is the common ground of the power supply. One (not both, they are internally wired together) of the ground clips from the 'scope probes must be connected to the power supply common ground.

2. Using AVR Studio create a new project with descriptive name (ex. 'RCfilterLab')
   - Select AVR GCC and type a descriptive name into the Project name field.
   - You will be using existing sources so uncheck the Create initial file box.
   - Check the Create folder box.
   - Browse to ensure the Location (where your project will be saved) is your USB drive. Files left on lab computers can be erased at any time!!!
   - Select AVR Simulator in the Debug platform field.
   - Select ATmega16 in the Device field. (scroll down, it is there)
3. Configure your project settings using the Project -> Configuration Options drop down menu
   - Change the **Frequency** box to **8000000**. (8 million without the commas)
   - Change the **Optimization** box to **–O2**.
   - Click on the **Include Directories button** found in the left side panel.
   - Click on the New Folder icon (upper right corner) and use the browse ellipse "…" to add your project folder you created in Step #2 (appears as .\ ), then click, 'OK'
4. Use a web browser to navigate to the course website and download the supplied C file, main_RC.c, into the project folder you created in step 2.
5. Switch to AVR Studio to add the main_RC.c file to your project. Find the Project File Manager Window (look for AVR GCC and a file tree in the upper left hand corner).
   - Right click on **Source files → Add Existing Source File(s)…** and locate the **main_RC.c** file that you just copied to your project folder.
6. Now, please take a moment to read through the code, especially the comments. The code has been designed to give you insight into what each line is doing. If you understand the code, you will be able to implement your own timer and counter in other projects. Pay special attention to

any references the comments make to the datasheet for the microcontroller. Timers will come up throughout the semester.

7. Program the microcontroller (using Build → Build, or F7) with this new code (use the “AVR” icon on the toolbar). *Be sure to verify that you’re using the hex file associated with this project while you’re on the Program Tab.* AVR Studio does NOT update this automatically when you open a new project. If someone else has used the AVR Studio recently, it may still be pointing to their hex file. Double-check that the fuses and the Oscillator and ISP speeds are set as instructed in the Introduction to the ATmega16 lab.
8. Connect a 10-pin jumper between port C and the Switches header (check the alignment of the red stripe etc…). Also connect the PORTA headers pins to the LED headers pins, and then power up the board, and press the reset button.
9. Connect the leads of the piezo speaker to PD4 and common ground. You should hear an audible tone when you hold down one of the switches on the STK500, and a different tone for each of the switches. The tone should stop when no button is pressed or if more than one button is pressed.
10. Look at the signal on channels 1 and 2 using the oscilloscope, and measure their voltages and frequencies. Acquire the ‘scope display for your lab report using IntuiLink software (or other means). Comment on the behavior of the system.