Documentation: assignment6

Connor Baker

November 6, 2016
Version 0.4a

Contents

1 Summary of Problem Specification

1.1 Abstract

Using a standard tuning frequency inputted by the user, as well as a range of values for capacitance, we calculate possible frequencies greater than the user inputted standard tuning frequency, writing the values to a random access, byte-based file.

We begin by grabbing user input, and then calculate capacitance using the formulae given in Section 2.We then find the inductance (L) using the standard tuning frequency. After that, we calculate our possible range of values for frequency (F_{min} and F_{max}), comparing each to the standard tuning frequency, writing only those that are greater to file. At this point, we calculate new values of F_{max} by incrementing C_{min} by 15 picofarad, again comparing and writing only values larger than the standard tuning frequency to file. We do this until C_{min} surpasses C_{max}. When this happens, we increment L by 2%, reset C_{min}, and again calculate values of F_{min} and F_{max} using the same process described above.

Each time we edit or do not edit the file, print out to console why. For example, if our F_{max} is larger than 16.7MHZ and we edit the file to include that new value, we should print to console that we have done so.

The program halts when the largest value of F_{max} is smaller than the standard tuning frequency. It should be noted that (conveniently) the first calculated value of F_{max} is the greatest (for our purposes at least), as it is strictly monotonic decreasing in nature, given a strictly monotonic increasing L and C_{min}, which is exactly what we have.

1.2 Assumptions

I use a pre-release version of Java 9. It is my assumption that the underlying changes in the language were nothing such that it would allow me to write something incompatible with the immediate previous release.

2 Formulae

Capacitance is denoted with C and measured in farads. Capacitance Minimum is denoted with C_{min}. Capacitance Minimum is denoted with C_{max}. Frequency is denoted with F and measured in hertz. Frequency Minimum is denoted with F_{min}. Frequency Minimum is denoted with F_{max}. Inductance is denoted with L and measured in henrys.

$$C = \sqrt{C_{min} * C_{max}} \tag{1}$$

$$F_{min} = \frac{2\pi}{\sqrt{L * C_{max}}} \tag{2}$$

$$F_{max} = \frac{2\pi}{\sqrt{L * C_{min}}} \tag{3}$$

$$L = \frac{(\frac{2\pi}{F})^2}{C} \tag{4}$$

3 Explanation of Components

3.1 Main.class

Main.class contains the main() method.

3.1.1 The main() method

The main() method consists of three main parts: the body, outer while loop, and inner while loop.

The body

The body of the method creates initializes the object we operate on (a component of TuningCircuit.class and covered in the corresponding subsection), as well as the random access byte-based file the program writes all the doubles to.

The outer while loop

The outer while loop has three duties:

1. To check for a condition that signals the termination of the program
2. To perform multiple iterations over the inner while loop
3. To increment the value of L and reset C_{min} to the initial value input by the user every time the inner while loop terminates.

The first duty is met via a simple if() statement. If the value of F_{max} is less than the user inputted standard tuning frequency, we can halt calculations because every value calculated after that point for the current or larger L and any given C_{min} will be smaller as well.

The second duty is met as this component of the main() method is a while loop.

The third duty is met by performing operations on the object using methods found in TuningCircuit.class.

The inner while loop

The inner while loop has three duties:

1. To calculate values of frequencies larger than the standard tuning frequency inputted by the user given a certain capacitance and inductance and print to file/console
2. To increment C_{min} by 15 picofarad
3. To check for a condition that signals us to break out of the inner while loop

The first duty is met by calling methods located in TuningCircuit.class.

The second duty is met by using the compound assignment operator +=.

The third duty is met in the form of a if() statement that checks for values of F_{max} that are smaller than the standard tuning frequency. When this occurs, we exit the inner while loop.

3.2 TuningCircuit.class

3.2.1 The TuningCircuit() constructor

The TuningCircuit() constructor calls several methods to get values for the variables used in the program. It calls (in order):

1. promptUserForStandardFrequency()
2. promptUserForCapacitanceMinimum()
3. promptUserForCapacitanceMaximum()
4. calculateCapacitance()
5. calculateInductance()
6. calculatefMin()
7. calculatefMax()

These methods are detailed below.

3.2.2 The promptUserForStandardFrequency() method

3.2.3 The promptUserForCapacitanceMinimum() method

3.2.4 The promptUserForCapacitanceMaximum() method

3.2.5 The calculateCapacitance() method

3.2.6 The calculateInductance() method

3.2.7 The calculatefMin() method

3.2.8 The calculatefMax() method

4 Notes

Equations (1–4) relate everything in terms of base units. That means we must convert F from megahertz to hertz (multiply by 10^6), and C from picofarad to farad (multiply by 10^{-12}).

Additionally, one can note that due to the structure of our formula for F_{max}, our values will shrink as we increment C_{min} (since we begin to divide by increasingly large numbers). This means that our first calculated F_{max} will be our largest. If we were looking for just the largest value, we would not calculate any other but the first, and save CPU cycles. However, we must calculate all F_{max} larger than our initial value of 16.7MHZ and store it in the random access file.

5 References

http://download.java.net/java/jdk9/docs/api/