TRC2400 Computer Programming
ECE2071 Computer Organisation and Programming

Laboratory Session 1

Week 2 – Semester 1 2011

IMPORTANT – MARKING
You will receive marks for preliminary work and lab completion by completing quizzes on Blackboard All quizzes receive equal marks and these will be scaled to give a final lab mark worth 10% of your final assessment. You MUST complete the preliminary work quiz BEFORE midnight of the day before your lab otherwise you will receive a zero mark for the lab exercise (both preliminary and completion mark) You must start the completion quiz before the end of your laboratory period (you will need the demonstrator to enter a password which will only be provided when you complete the lab)

1. Objectives

This laboratory provides an introduction to using Microsoft Visual Studio and some practice at writing and debugging simple programs. The short programming exercises were written to help develop your understanding of:

- Input/output statements using printf(), scanf() functions and format strings.
- Basic data types.
- Declaration of variables and constants.
- Use of assignments and evaluation of arithmetic expressions.
- Program debugging and problem solving.

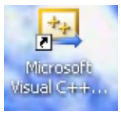
In those exercises where you are asked to write a program you should sketch your program on paper first and only enter it into the computer when you are sure that you have the logic right.

2. Preliminary work

Before coming to the lab you should complete the preliminary work quiz on Blackboard. This week's quiz will cover your understanding of the bullet points above and also the exercises in this laboratory exercise.

3. First Exercise

To start Microsoft Visual C++ 2008, double click on the desktop icon:



From the pulldown menu select "File -> New -> Project". Select "General" and "Empty Project" give your project a name (This could be Lab1_1) and select a suitable place to store your application.

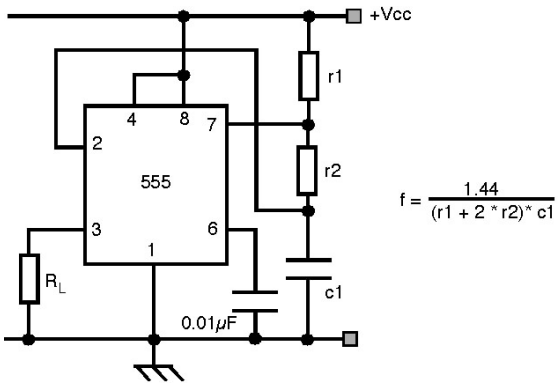
You now need to create a program file to put into your empty project. Select "File -> New -> File" and from the options Visual C++ à C++ File. Save your file in the project folder. Add your C++ file to your project by right clicking on source files and add to project.

Next cut (from the file available on MUSO) and paste the 555 timer place it in your Lab1_1.cpp file.

After you have entered the program you should compile it by selecting "Rebuild <project name>" under the "Build" pull-down tab at the top of the window. To run your program press F5. Debugging is an important

part of programming and you must learn how to do this. First of all, for successful debugging you must understand how your program is supposed to work. Then you should find out as much as you can about the error - did it cause a compiler error? did it produce the wrong output and if so what did it produce? With this information you can try to work out what kind of error would produce the observed behaviour. If you cannot see the error then try including more printf statements to monitor what the program is doing. When you think that you have found the error then correct it and try again. Do not make random changes to your program, this seldom leads to a working program.

The aim of the following program is to calculate the frequency of the following 555 timer circuit.



Insert the following (this program is available as a separate file in Blackboard - DO NOT CUT AND PAST THIS PROGRAM FROM THE PDF) into your program file:

```
// Program to calculate the frequency of a 555 timer astable oscillator
// Written by Wai Ho Li
// Date Modified: 2010-03-06
#include <stdio.h> // give access to printf and scanf functions

int main(int argc, char* argv[])
{
	float freq; // Result
	double r1, r2, c1; // Declaring Variables
	const double numerator = 1.44; // Constant numerator used for calculation

	// Getting user inputs
	printf("Enter the value of r1 in kilo Ohms: ");
	scanf("%lf", &r1):
	printf("Enter the value of r2 in kilo Ohms: ");
	scanf("%lf", &r2);
	printf("Enter the value of c1 in micro Farads: ");
	scanf("%lf", &c1);

	// Calculating results
	freq = numerator / (r1*1000 + 2.0*r2*1000) * c1*1e-6;

	// Left-justified and fixed width printing of result
	printf("\nThe oscillator frequency is: %-10.21f Hz\n", freq);

	return 0;
}
```

After you have entered the program you should compile it by selecting "Rebuild All" under the "Build" pull-down tab at the top of the window. To run your program press F5. Debugging is an important part of programming and you must learn how to do this. First of all, for successful debugging you must understand

how your program is supposed to work. Then you should find out as much as you can about the error - did it cause a compiler error? Did it produce the wrong output and if so what did it produce? With this information you can try to work out what kind of error would produce the observed behaviour. If you cannot see the error then try including more printf statements to monitor what the program is doing. When you think that you have found the error then correct it and try again. Do not make random changes to your program, this seldom leads to a working program. If all else fails then ask the demonstrators for help and explain how far you have got with your debugging.

Errors - Syntax

There are two major types of errors that can occur when you are programming in C. The first type of error you will have to deal with is syntax errors. With a syntax error what you have written does not conform to the rules that C programs must adhere to. When you try to build the above program you will get an error message in the bottom output window. Double click on the line in the output window that has the error message. The position of the error will usually be highlighted. Work out what the error is and correct it.

Note that there is still a compiler warning. You can still continue and try to run a program without sorting out the warnings. This is not a good idea unless you are sure you understand why the warning occurred. Try to remove all warnings before proceeding.

In the case of the Lab1 program the compiler is worried that there will be a loss of precision when a double length result is stored in a smaller float variable. To correct this problem, the frequency variable should be declared as a "double". If the added precision is not required, the result can also be cast as a type **float** by placing `(float)` after the assignment operator.

Errors - semantic

Now after compiling the program without errors or warnings you should run it (select "Execute" under the "Build" pull-down menu item or click on the red exclamation mark or press Ctrl+F5). At the prompt type in the following values followed by 'return'.

r1 = 1 r2 = 2 c1 = 3

The correct answer should be 96Hz. The reason the program gives an incorrect result is because of a semantic error. Although the program is correct C (and therefore the compiler cannot see any syntax errors) it does not perform the correct calculation required to work out the oscillator frequency. Compare the assignment statement with the problem definition to find out what the problem is.

4. Second exercise

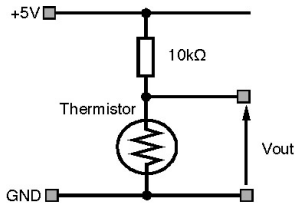
A thermistor has an electrical resistance that shows a large variation with temperature. These devices are used to measure temperature. The relationship between temperature and resistance is given by the following equation:

$$R = K\,\mathrm{e}^{\frac{B}{t}}$$

where:

R = thermistor resistance
B = a constant for any particular thermistor (2500 for this exercise)
K = another constant (0.34094 for this exercise)
t = temperature in ˚K (note 0˚K = -273.16˚C).

Temperature is to be measured using the following circuit:



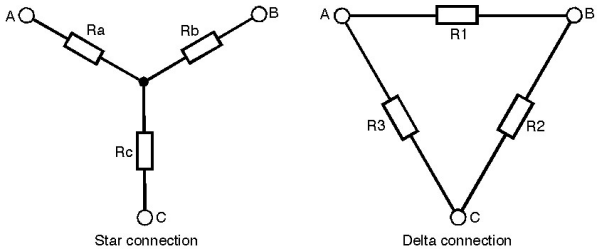
Write a program to prompt the user for a value of Vout and then to print the corresponding temperature.

Try your program with a Vout of 1.1V the temperature should be 277.14˚K (+3.98˚C).

5. Third Exercise

In analyzing three-phase power distribution networks it is often necessary to convert a delta-connected load to a star connection (and vice versa).

Write a computer program that will prompt the user for the resistance of the three elements of a star-connected load and output the corresponding values of an electrically equivalent delta connected load:



Note that so far as connections to the points A, B and C are concerned the two circuits and electrically equivalent.

$$R_1 = \left(\frac{\frac{1}{R_a}\frac{1}{R_b}}{\frac{1}{R_a}+\frac{1}{R_b}+\frac{1}{R_c}} \right)^{-1}$$

$$R_2 = \left(\frac{\frac{1}{R_b}\frac{1}{R_c}}{\frac{1}{R_a}+\frac{1}{R_b}+\frac{1}{R_c}} \right)^{-1}$$

$$R_3 = \left(\frac{\frac{1}{R_a}\frac{1}{R_c}}{\frac{1}{R_a}+\frac{1}{R_b}+\frac{1}{R_c}} \right)^{-1}$$

Here is some data that you can use to check your program:
If Ra=43.35Ω, Rb=1.57Ω, Rc=81.98Ω then R1=45.75Ω, R2=86.52Ω and R3=2388.92Ω

Please note that marks will not be allocated to people who do not attend their allocated lab and complete the appropriate quizzes by their deadline. Under no circumstances will marks be recorded after the laboratory period is finished.

RAR 15/2/2011