Introduction

This document sets out to explain the logic and methodologies used in programming the Suspension Calculator. Its aim is to make it easy for the reader to understand the code, and to ensure if they program additions, they follow the same structure to the current code.

Figure 1, sets out the heirachy of the windows for the program. The program initially has two open windows; 'open' and 'main'.
Open has two purposes. Firstly, it introduces the user to a few key concepts of the calculator. Secondly, it ensures their is always a window open in the software, as if all windows are closed the application will terminate, which would happen when the user navigates between Main and some of the other windows.
Main, is the main window that the user will be using. From here the user can access any of the 19 calculation windows, clear the data, or exit the program. If the user opens any of the 19 calculation windows the calculation window opens and main closes, and if the user wants to go back to main they have to do so via a button on the new window, which will close the new window. This way, the program ensures that the user is not entering data in two locations, which could lead to problems.
Regarding the 7 windows seen in the right column of Figure 1, these are not truly accurate as in reality their are 21. However, for simplicity the lower 5 represent multiple windows, that have been given a more general name.

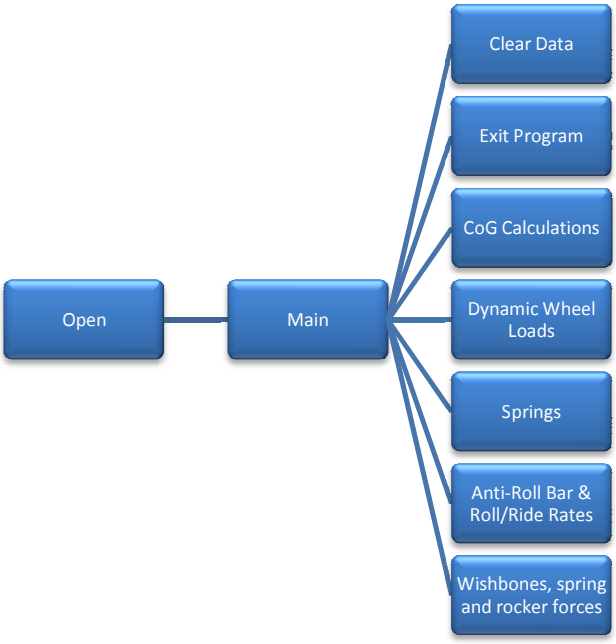

Figure 1: Window Hierarchy

Structure of windows

The windows that perform the calculations have all been programmed in a consistent way for simplicity. They first retrieve the data, and then await the user to press one of two buttons.

'Return' will take them back to the 'main' window, without performing any changes to the data or calculations.

'Calculate' will call 5 methods as shown in the code below:

```
public void calculate_Click(object sender, EventArgs e)
{
    input();

    baddata();

    calculate();

    output_inputted();

    output_outputted();
}
```

'input()' will retrieve all values the user has entered and update the variable 'data'.

'baddata()' will then check that all the data stored relevant to the calculations in this window make sense, for example, that are non-negative if applicable or that their will be no divisions by zero. If an error does occur, it will tell the user of the problem in a pop up window. This routine may occasionally be called after 'calculate()'.

'calculate()' will perform all the necessary calculations.

'output_inputted()' has two purposes. The first is to fill up the Textbox fields with the values that have already been assigned to the data when the user opens the window (this method is also called when the window is initialized). The second is to update the fields when the user presses calculate, although this is generally not necessary, it is done to prevent errors.

'output_outputted()' this routine outputs the calculated data. This is generally done onto the same screen; however, in the case their is too much data to do this neatly it is done on a new screen. In this case the routine is called, 'outputscreen()' and it merely opens a new window passing on the data. The new window, which has no other purpose than the following, then outputs the data.

In some cases any of the above 5 methods, may have subroutines embedded within.

Variables

Another important concept lies in how the variables were programmed. As multiple windows may need to access the same data, and many are intertwined, it was decided to make all variables accessible across the whole application.

The way in which this was done was by making a class called ‘variable.cs’ and defining the approximate 300 variables within. The snippet below, shows the code in this class, but has restricted the length to that of 2 of the variables.

```
namespace SuspensionCalculator
{
    public class variable
    {
        public bool clockwise { get; set; }
        public double antirollbar_rollrate { get; set; }
    }
}
```

To make use of the variables in this class, ‘open.cs’ initialized it upon opening the calculator and sets all the values to the default ones. This class is then passed onto ‘main.cs’ and to any windows that are opened there after, if any changes are made in any window the class is updated. For simplicity in all windows the ‘variable’ class has been named ‘data’.

In order to pass this data between windows, the code at the start of each window had to be modified to the following:

```
namespace SuspensionCalculator
{
    public partial class Formname : Form //Formname is the name of the form
    {
        protected variable data;
        public Formname()
        {
            InitializeComponent();
            this.data = new variable();
        }
        public Formname(variable input)
            : this()
        {
            this.data = input;
            output_inputted(); //not relevant to variable data issue
        }
```

Aside, from the segment at the start of each window, it was necessary to pass the data into the new windows when they were opened, which was done with the following code:

```
private void button1_Click(object sender, EventArgs e)
{
    formname user = new formname(data);
    user.StartPosition = FormStartPosition.CenterParent;
    user.ShowDialog();
    this.Close();
}
```

Variable naming

The variable names have been chosen intentionally to be quite long and descriptive, making it easier for someone to find the variable they are looking for or to work out what the variable represents.

Rounding

The rounding convention used was to round numbers to the nearest whole number, or to 3 significant figures for numbers smaller than 100. However, the code only allows you to round to a specific number of digits. As such the programmer should estimate in each situation what decimal place would be needed to achieve 3 significant figures for the lowest value within a reasonable range of values for that field.

Cleardata

This button simply sets all values to their default values and has been included for the user to easily be able to clear the data and to prevent bugs.

Exit Program

This button found on the main and open screen. It is provided as an elegant way of closing the program and it uses the `'Application.Exit()' command.`

Screenlocation

All new windows have been initialized with the following code:

```
private void button1_Click(object sender, EventArgs e)
{
    formname user = new formname(data);
    user.Show();
    this.Close();
}
```

In all cases the form is called user.

Layout and graphics

With the exception of the main and open window, the software has a close to identical layout between each window. There are 3 group boxes aligned in a left column and 2 objects aligned in a right column. The left column, has ‘input’ at the top, ‘controls’ at the centre and ‘output’ at the bottom. The right column has information text at the top and an information diagram at the right. Their exact locations and size vary between windows due to different distributions in content.

The ‘input’ groupbox allows users to enter data in ‘TextBox’ fields, which are left aligned. On the right of each of the textboxes are labels describing the field. The ‘controls’ groupbox has two buttons: ‘return’ and ‘calculate’. Return is left aligned and redirects the user to the main window, calculate is right aligned and retrieves the data the user has entered and calculates the results. The ‘output’ groupbox contains right aligned label to which the software outputs the calculated values, and left aligned labels which describe the content of these fields.

The titling is also consistent between windows, the convention used is ‘Suspension Calculator | window title ‘. The background colour is always the default windows one, called ‘control’. All windows contain the products logo/icon. Finally, the 5 objects in the windows are always anchored to the top, left of the screen. This way if the user increases the screen size, the layout will be the same, just that they will have extra empty space round the bottom and right edges.

The images have been created to be as consistent as possible. The graphical content of the image (car, wheel, spring...) are drawn in black, and grey is used if a second colour is needed. All the labelling is done in the same tone of dark blue.

Tab order

The tab order used throughout the software on the first level is; input, controls, output, information text, informational window. Within these fields, the data is dealt with first, then the labels. When navigating through the data and labels the order used is top to bottom, then left to right.

Terms and conditions

The terms and conditions of using this software are provided on the open screen, and the user must confirm that they consent to them before they progress to the calculator.

Commenting code

This code has intentionally been structured in a simple and methodological way, with descriptive variable names. As such, if the theory is familiar, the code should be quite simple to understand. Nonetheless, at the top of each page of source the file name is provided with a very brief description of what the window does:

```
//banking.cs
//This window calculates the 4 wheel load transfers due to banking
```

Comments are also used throughout the code to provide clarification at places where the code may be confusing.