**3. Introduction**

At present this document should be invariably seen as a “work-in-progress" - to the point of representing a “thinking out loud" process. As time goes by, the utility of this document and it's correctness are expected to improve. The overall intent of the document is to describe various relevant installation and deployment caveats as well as to document the scripting format used by the software to perform its functions.

**4. Background and Software Purpose**

The autoinput utility represents a very small hack which simulates mouse-movements -clicks and key-presses via a script file provided as a command-line argument.

Although it is applicable to many deployment scenarios it was initially scribbled for the purposes of aiding a project where an extremely close synchronization between various programs was desired. In particular it was sought to start (and stop) software related to an eeg device, an eye tracking device, and a stimulus instrument. This would allow the beginning of each data file captured by the various devices to be synchronised at a time zero making analysis possible.

Instead of clicking on the start buttons to begin the various pieces of software the autoinput script was programmed to take control of the mouse and keyboard and do this in the shortest time possible.

In presence of many already-existing solutions yielding similar functionality, this utility was written due to the following considerations:

- it represented a very quick hack/job and did not take much time at all

- it needed to be lightweight and easily learned/scripted with respect to its automation and general use

- it needed to be open-sourced and fully extensible for the purposes of future specializations including issues such as being launched and running in the background without any GUI/windows of its own (i.e. low-intrusion)

- low amount of overhead with respect to script-processing and overall system interaction

- simple scripting format (e.g. .csv) which is easily editable within and integrated with other applications (e.g. text editors, spreadsheets, etc.)

**5. Scripting Format**

The autoinput utility reads lines from a .CSV (comma separated values) file and executes each line as a separate command.

Commands (text contained in each line) are case sensitive.

The first value/field on a given line denotes which command it happens to be. Whether there are additional fields/values (separated by commas) depends on the command in question.

Capture.JPG

For example, the above command tells the autoinput utility to left mouse click on pixel 175 (x axis), 756 (y axis). Note – pixel location is measured from the top left corner of the screen, and the total resolution will depend on the resolution of the particular display you are using with your computer.

An empty first field (e.g. lines beginning with a coma) denote a comment line. You can type what-ever you like after the coma for your informative purposes as the line will not be executed/processed by the utility.

Capture.JPG

The double-quotes are allowed to enclose any field (including the empty/missing values). Thusly the following are both comments:

, this is a comment

"", this is a comment

Empty lines are also allowed (will be skipped).

**6. Supported commands**

There are six basic command possible with the autoinput utility. These are (1)mouse move, to move the mouse to a position on the screen, (2) mouse down, to click and hold the left mouse button down, (3) mouse up, to unclick the mouse after a mouse down command, (4) click, to click (down press and then release) the mouse button as would normally be done with a finger input, and (5) sleep, to create pauses between commands of a set length, and (6) ascii, which can be used to simulate pressing any particular key on the keyboard.

Note: at this time the utility does not support pressing the right mouse button.

Combinations of commands in sequence can be used to undertake almost any action that would normally be achieved through a person using the mouse and/or keyboard.

**6.1 mouse move**

The mouse\_move command moves the mouse pointer to a fixed position on the screen. It has two additional arguments: the x and y (pixel-wise) coordinates for the mouse pointer's new location. Allowed, coordinates are non-negative integers with top-left corner being the origin. Greater numbers move the pointer down and right respectively.

For example a complete mouse move command in a .csv file may look like:

Capture.JPG

This command would move the mouse pointer to the top-left corner of the computer's desktop.

After the autoinput is finished processing a given .csv scripting file the mouse-pointer is returned to the original location (i.e. before autoinput's invocation).

NOTE: the physical mouse will still work when the autoinput utility is in operation, which can disrupt the execution of the commands. The best thing is not to wiggle the mouse during the automated execution.

**6.2 mouse\_down**

The mouse down sends a left-click down event (i.e. user pressing-down on the left button of a mouse). It clicks and HOLDS the mouse button down. The command will look like this:

Capture.JPG

This command is not location specific, and will be executed wherever the mouse pointer is at that time. Thus this command is best used after the mouse has been moved using the mouse\_move command to where you want the button or other object pressed on the screen.

**6.3 mouse\_up**

The mouse up sends a left-click up event (i.e. user releasing the left button of a mouse). It is design to end the execution of a mouse\_down command. The command will look like this:

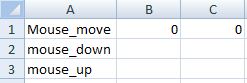
Capture.JPG

Like the mouse\_down command it too is not location specific and will be executed wherever the mouse is presently located.

**6.4 click**

The click command is designed to simulate a normal clicking motion of the left mouse button as would be achieved with a finger press. It incorporates a down press and release into the same command.

The click command is equivalent to this command sequence useing the mouse\_down and mouse\_up commands at the origin (top left of the screen):

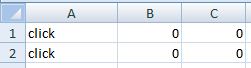


The click command does this in a more simple coding structure using this format:

Capture.JPG

This is a much easier and efficient to code.

This command can also be used to produce a ‘double click’ simply by repeating the command twice in sequence such as:



**6.5 Sleep**

The sleep command pauses the execution of the script file for the time specified in the command. The command has one argument to specify the amount of time the script should ‘sleep’, this should be a positive amount in milliseconds, for example:

Capture.JPG

This example will pause the script file for 1000 milliseconds (equivalent to 1 second).

The purpose of this command is to allow short breaks between some commands. It has been found that when starting some software (or executing a function in it) use the autoinput utility a short break is needed for the system to load the program or function into memory before any further buttons can be pressed inside that software. For example, when starting internet explorer often it will take a fraction of a second for the software to appear on screen where you can then start using it. The sleep function allows you to accommodate such occurrences into your script file.

The accuracy of this sleep command though should be taken with caution. It is unlikely that your computer’s internal clock will be accurate to 1 millisecond, and the processing time to activate the script is also likely to create some lag. Thus the sleep duration will always have some error in its execution.

**6.6. ascii**

The ascii command emulates a user pressing (down, then up) a given key on a keyboard. The key in question is specified by the second argument/value on the line. It represents a decimal value of

the ASCII symbol. For example the following command:

Capture.JPG

will make computer think that the user had pressed the ‘esc ‘key.

A full list of the standard ascii codes for the keyboard buttons can be found at: <http://www.ascii-code.com/> (as at 11 Sept 2012)

Note: Decimal ascii code is used in the autoinput utility.

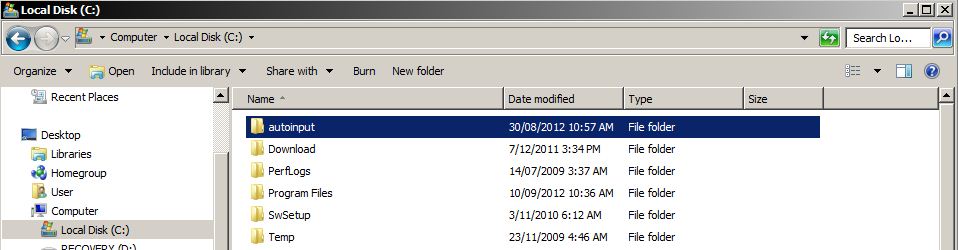
**7. Installation**

The installation process for this utility is not the same as a piece of software most users would be familiar. It requires specific files to be placed in different locations on your computer for it to function correctly. Please follow these instructions carefully as missing a step can render the utility non-functional.

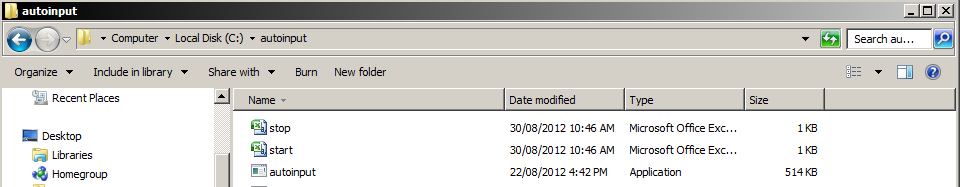
This installation will show you how to place TWO different script files onto your computer. This reflects the original purpose of the utility to both start software being used in an experimental procedure at the same time, and then stop that software at the same time after the experiment is completed. However, only one of these script files ever needs to be used if you only need one of them, so this installation process is still viable if you only need one script file.

**Step 1.**

Create a folder for the program and relevant script .csv file, on your hard drive. Call that folder ‘autoinput’:

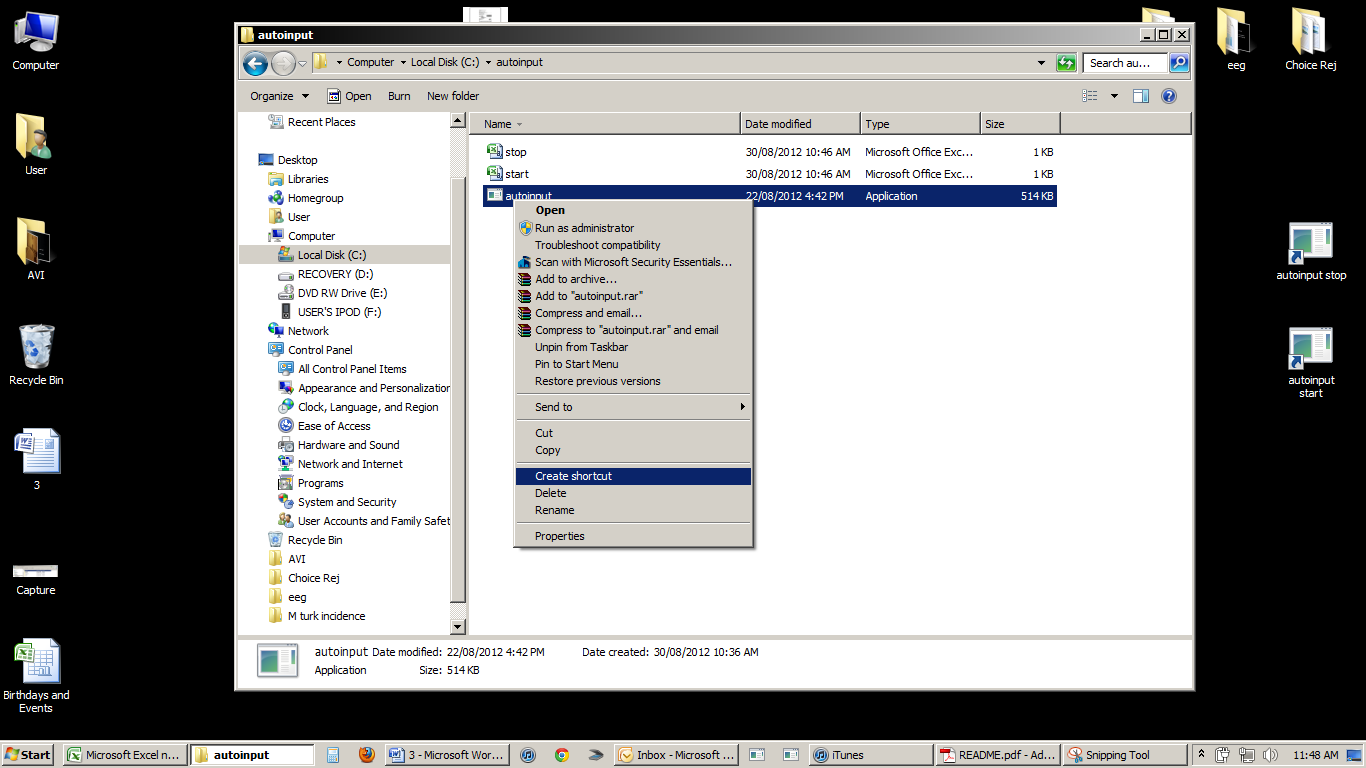


Put the autoinput.exe and script files (e.g. start.csv, stop.csv) in this folder.

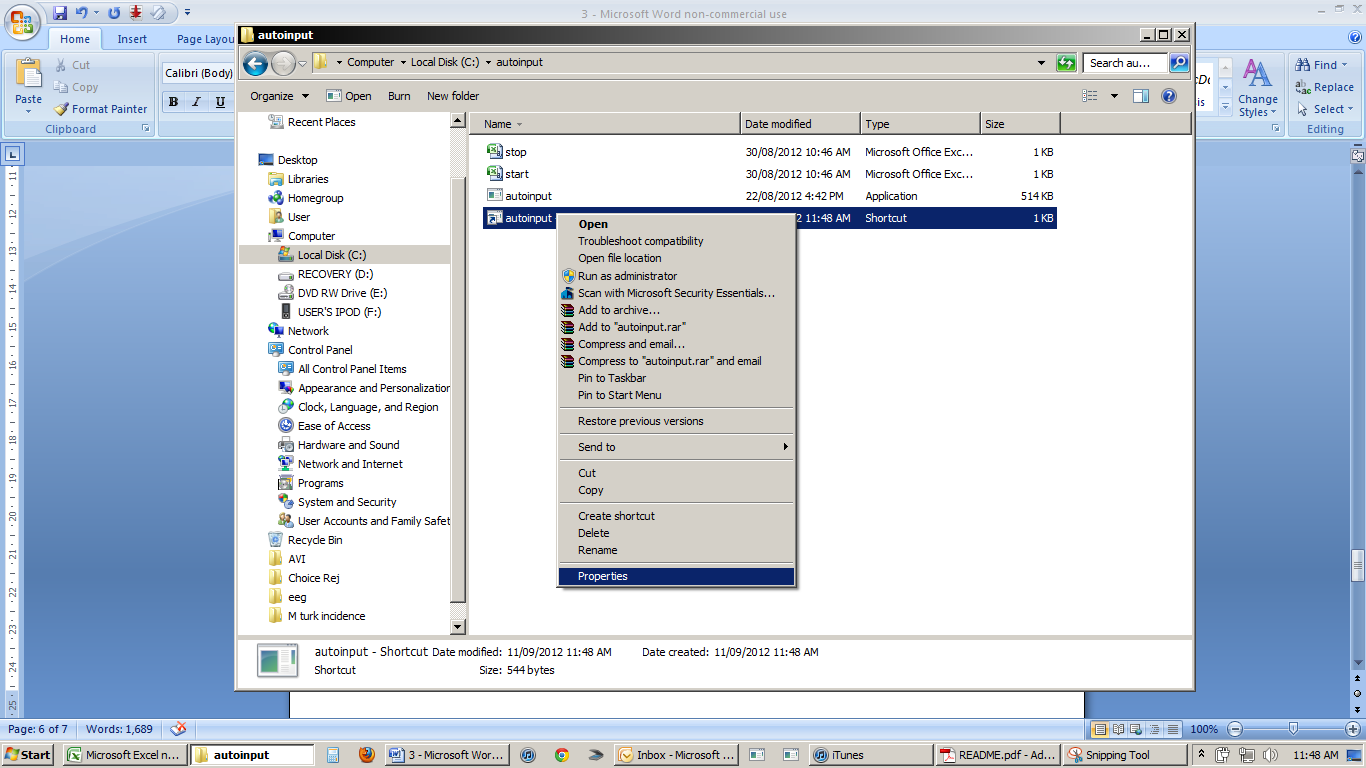


Step 2.

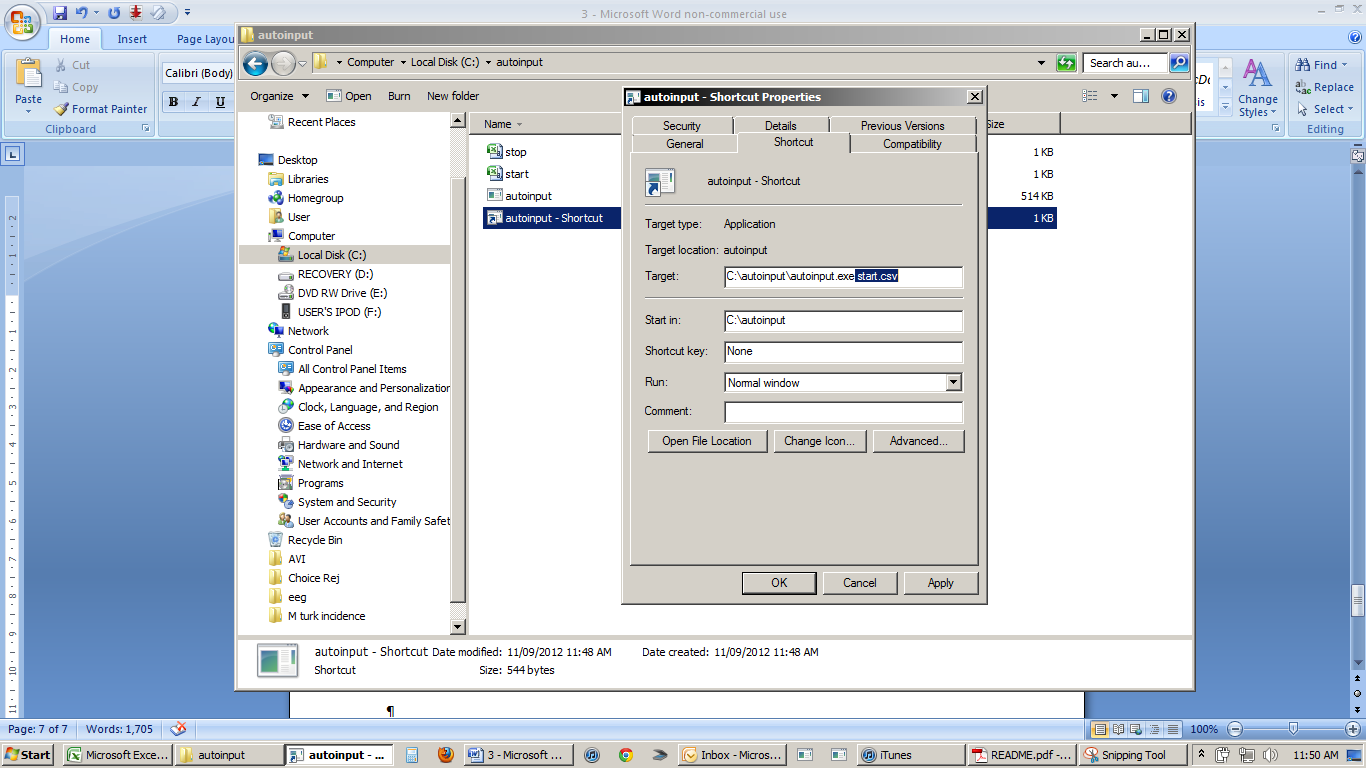
You will now need to create a shortcut to the autoinput.exe file. Do this by right clicking on the autoinput.exe icon and clicking on ‘create shortcut’.



A new shortcut icon will now appear. Right click on that new shortcut and select ‘properties’.



You will now need to change the target of the shortcut. At the end of the current text in the target cell type in a ‘space’ and then ‘start.csv’. This tells the shortcut to open both the autoinput utility (the original text in the target) AND the CSV file (the text you just added) that contains the commands for the utility to execute.

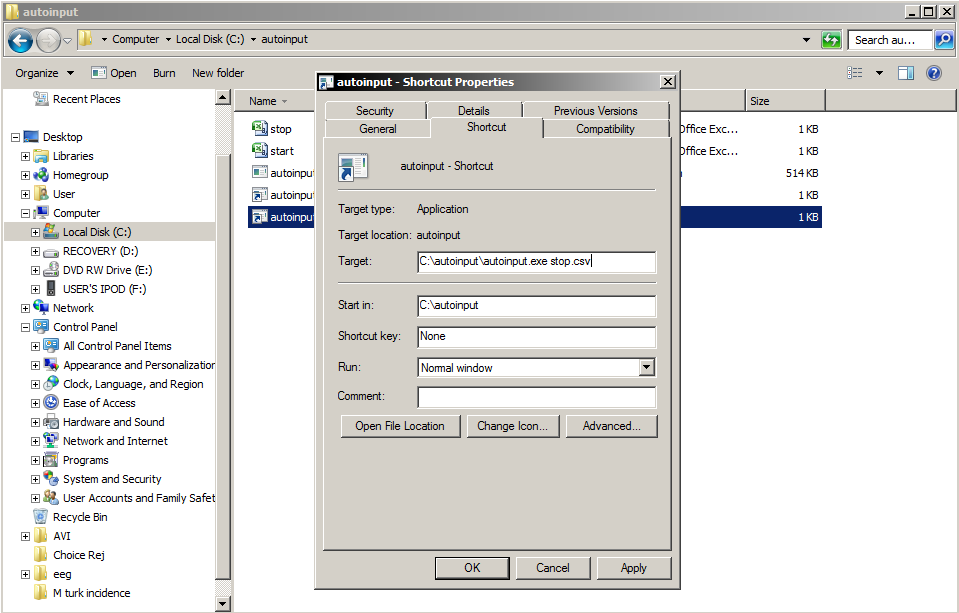


You may also like to go to the ‘general’ tab in this screen and change the name of the autoinput shortcut. In this case we will name it ‘autoinput – start’ as that is the CSV file it will execute, but the name is up to you.

Once you have made the changes click ‘OK’

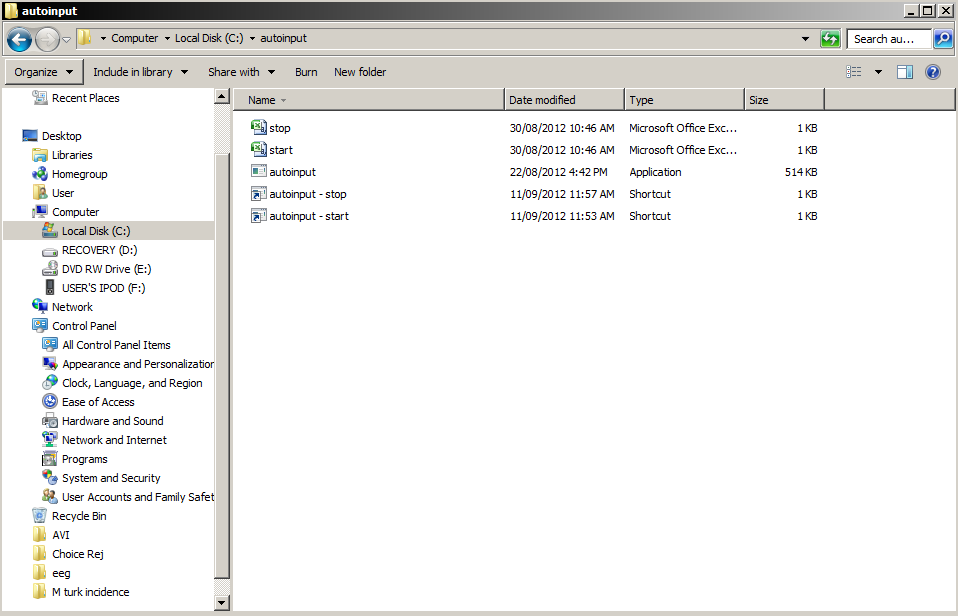
**Step 3.**

You now need to repeat the process in step 2 to create another shortcut. This time the text you add to the target is a ‘space’ followed by ‘stop.csv’. This will create another version of the utility that will execute the commands in the stop csv file.



You can also rename this shortcut before clicking OK.

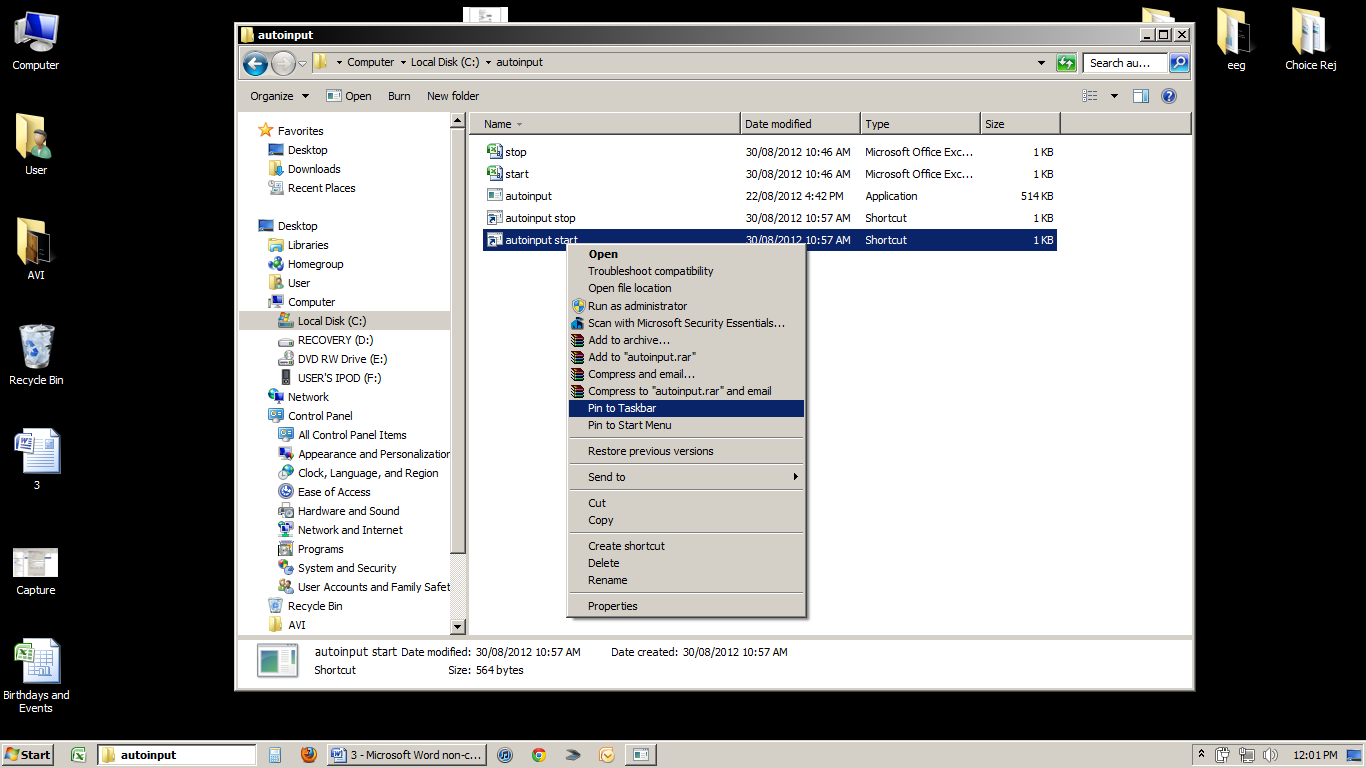
You should now have two shortcuts in the folder, one to execute to commands in the start csv and one to execute the commands in the stop csv.



**Step 4.**

You can now execute the start and stop command files just by double clicking on the shortcuts. If you like, you can ‘pin’ these to the task bar or move these to the desktop to make them more accessible during your experiment.

For example, they can be pinned to the taskbar by right clicking on each shortcut in turn and selecting ‘pin to taskbar’:



Now both of these shortcuts will appear on your taskbar where they can be easily executed at the start and end of your experiment. These can be seen below in a red box.

Untitled.gif

**Step 5.**

You should now go into the start and stop CSV files and type in the commands that you want the utility to execute. The available list of commands can be found in section 6 of this document.