

This is a  $\mu$ Vision2 tutorial for use with the lab assignments. **Bold type** indicates a sequence of keystrokes or mouse clicks.

### Creating a project

Start  $\mu$ Vision 2.

**Start -> Programs -> Electrical and Computer Engineering Applications -> Keil -> Keil  $\mu$ Vision2**

If you haven't already created a project for lab 4, do so now.

**Project -> New Project**

Select “\\minerfiles.mst.edu\dfs\users\((S:)” and then “\CpE214\lab4” to put the file in your AFS account. Call the project something like “lab4.uv2”.

You will be prompted to select the device. Select 89C52 from Atmel.

**Atmel -> 89C52**

Add the assembly/C source file from your preliminary to the project.

**Project -> Targets, Groups, Files -> Groups/Add Files -> Source Group 1 -> Add Files to Group**

Find the assembly/C source file you created for your preliminary and click Add, then Close, then OK.

You may now open the assembly/C source file by clicking the "+" next to Target 1, then the "+" next to Source Group 1. The name of your file should appear. Double-click on it and a window will open with your source code.

In  $\mu$ Vision2 HEX files are not generated by default. You may change this by selecting "Target 1" from the Project Window (the window on the left), then

**Project -> Options for Target 'Target 1' -> Output**

Make sure the "Create HEX File" checkbox is checked.

You need to assemble your code before you start debugging.

**Project -> Build target**

**OR F7**

If you make changes to your build options, you may need to use

**Project -> Rebuild all target files**

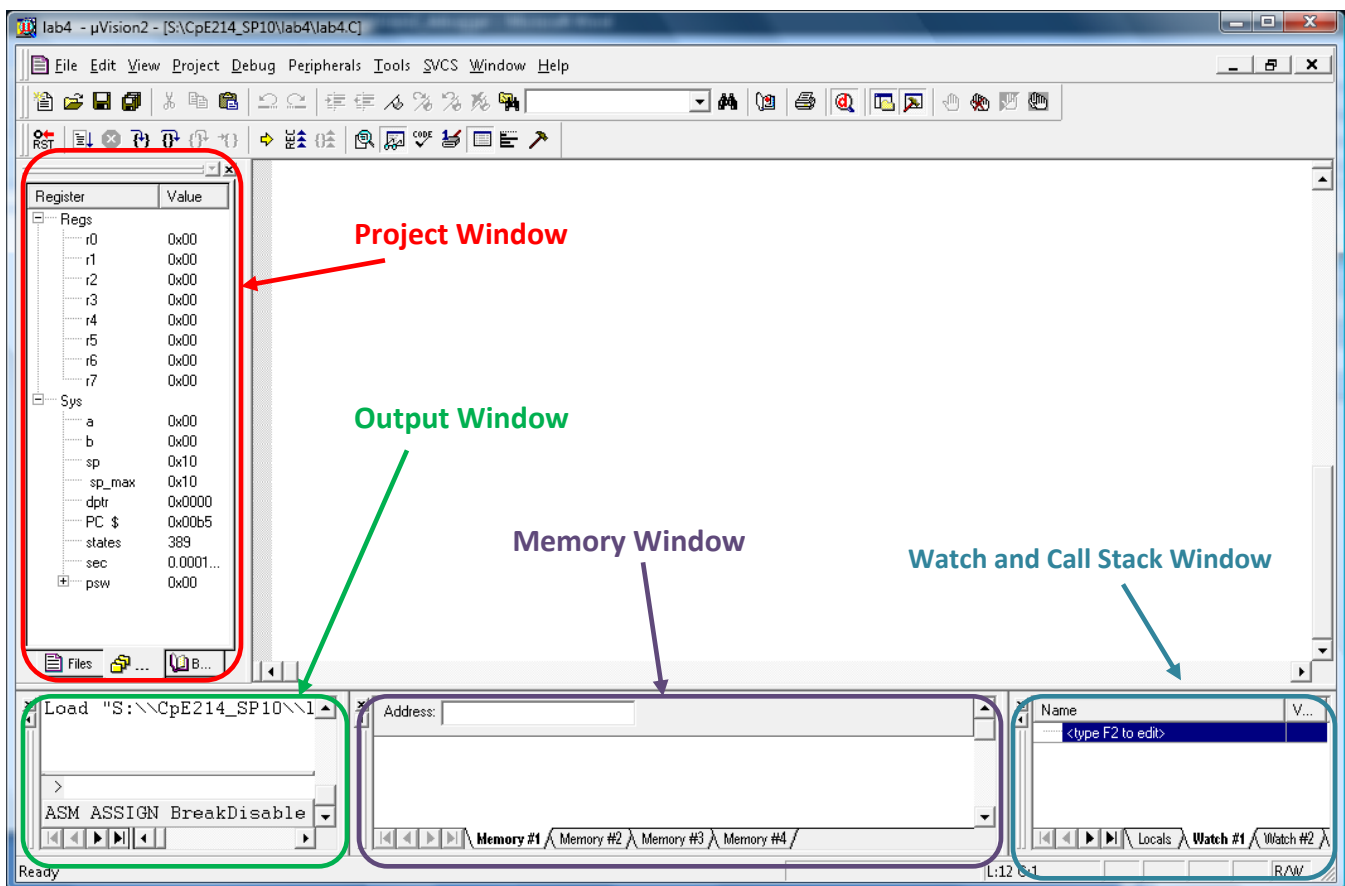
## Debugging your program

Now you are ready to start the debugger.

**Debug -> Start/Stop Debug Session**  
**OR Ctrl-F5**

For this experiment, you will need a few debugging windows open. You can tell which options are selected because they will be highlighted in the View menu. Under the View menu, make sure the following options are selected:

- Debug Toolbar
- Project Window
- Output Window
- Watch and Call Stack Window
- Memory Window




**Figure 1. Required debugging windows.**

If any other windows are selected, you may unselect them because they will probably not be useful in this experiment.

In the upper left window (Project Window) make sure the second tab (icon of three folders) is selected. You can view all the important 8051 registers such as the Accumulator and the Program Counter from this window. These registers will be highlighted every time they change.

In the lower left window (Output Window) there is a place you can type commands. It uses the greater-than character (>) as a prompt. This is a left-over from an older version of the software, but it is still useful. If you don't see the prompt, click on the Command tab. In order to see the contents of any variable that you have defined in your code, type "ws ***VariableName***" into this command window and press Enter. If you called the XSEG at 7F55H "sevensseg", use that instead of "***VariableName***." Now if you click on the "Watch #1" tab in the Watch and Call Stack Window (lower right corner) you will see sevensseg listed, as well as its value (probably 0x00 since you haven't written anything to it yet). Every time sevensseg gets written to, you will see the watch window get updated.

The memory window is in the middle of the screen, at the bottom. You can display memory ranges here. Click on one of the memory tabs (Memory #1, for example) and type "X:5000H" in the textbox labelled "Address:". This tells uVision2 to start displaying external memory at address 5000H. You could also show "C"ode memory with C:address, internal "D"irect memory with D:address, and internal "I"ndirect memory with I:address.

You are now set up to trace your first program. You can single step through your program using **F11**, or the symbol that looks like an arrow pointing into two braces: .

Every time you single-step, you will see an arrow move down to the next instruction in your program. You should be able to see the values you load into memory at 5000H appearing in the memory window, one at a time. You will also see DPTR changing in the Register window.

Instead of single-stepping, you may want to run through the program and just stop at certain points. You can do this by setting a breakpoint at the lines you're interested in checking. Just double click on a line in the source window and a red square should appear next to the line to indicate a breakpoint. Now if you run the program, it will stop at that breakpoint.

To run the program continuously, press the Run button. It looks like a piece of notebook paper next to a down arrow. You can also use **F5**.

After the program starts running, it will continue until it hits a breakpoint. If it doesn't hit a breakpoint, you will have to stop your program by clicking the Break button. It looks like a Stop sign. You can also use **Esc**.

To reset the program to the beginning, you can use the "RST" button. You can also use **Peripherals -> RESET CPU**.

If you make any changes to your file, you should exit the debugger, rebuild your project, and then start the debugger again.

## Taking a screenshot

Once you have verified that your program works, you may take a screenshot of the working simulation to include in your lab notebook. First press the **PrintScreen** key to capture the screen. Then start Paint or some other image program. Paint is included with Windows and may be started by clicking

**Start -> Programs -> Accessories -> Paint.**

Note that there are TWO "Accessories" submenus in the "Programs" menu on the CLC computers. If you can't find Paint in one menu, look for the other one.

Once you have started Paint or some other image program, select **Edit -> Paste** to view your screenshot. You can then print out the screenshot or save it to a file.