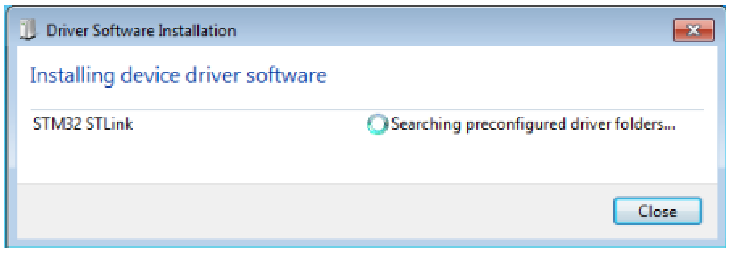
Chapter 1

Setting Up Eclipse

1.1 Configuring Eclipse on the UCT Computers

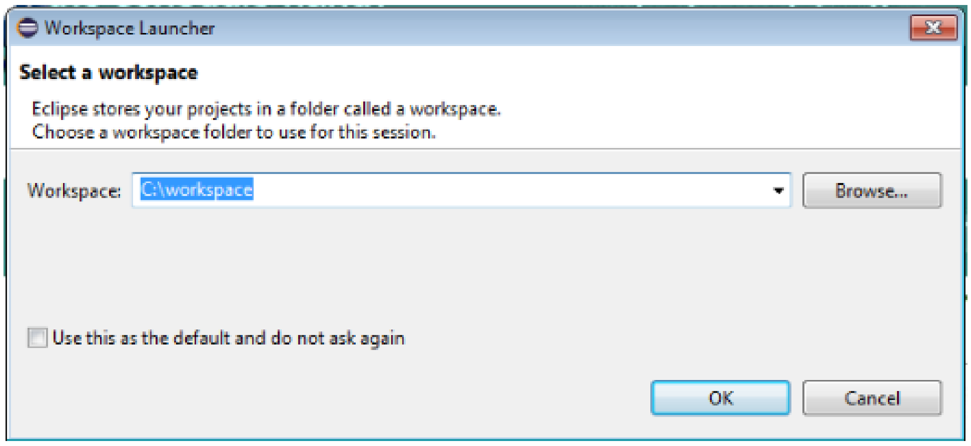
Eclipse is fully configured on the UCT campus computers but there are some steps that you need to remember before getting started. First plug in your STM32 board, Windows should start searching for a pre-configured driver once it's plugged in.



Installing device driver software

STM32 STLink   Searching preconfigured driver folders...

Windows should eventually find the drivers and you're ready to go. Launch up Eclipse, it is important you leave the default workspace as **C\:Workspace**.



Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\workspace

☐ Use this as the default and do not ask again

Then wait for **Eclipse** to load up, then create a new **STM32F0xx C\C++ Project**. The compiler will load up all the relevant files and put out a pretty empty generic template. First off, build the project so you have a set of binaries (.ELF files) to work with. You Build a project by following **Project** → **Build All** ,or selecting **Build Project** by right-clicking on the project file in the File Explorer window.

To run the code on your STM32 board, simply debug the project using the*GDB OpenOCD Debugging* configurations. The UCT computers will auto detect the project you are working on (only if you've already built it). Conveniently, the config for OpenOCD is filled in already so you can debug your code from the **Debug Perspective**.

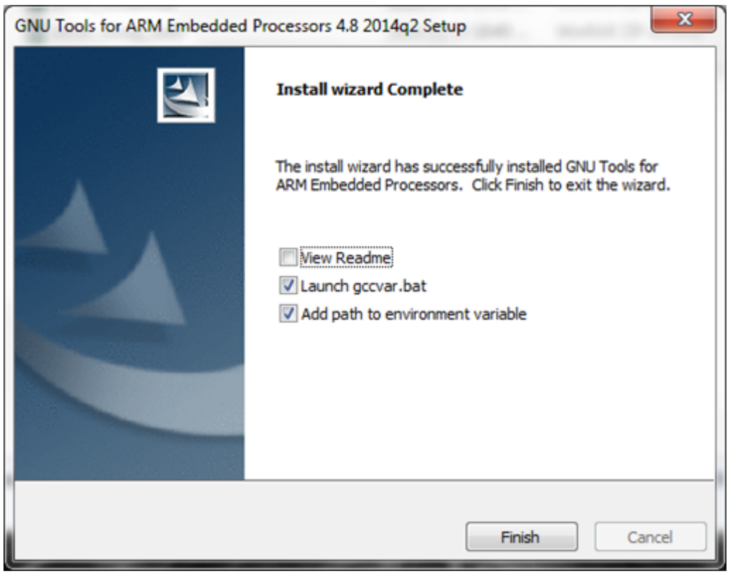
1.2 Configuring Eclipse for the STM32

The following is a guide to setting up Eclipse for Windows to debug the STM32F051 UCT Development board; I understand it is quite lengthy but it covers many details that are important when developing for the STM32. If your Eclipse is already set up, skip ahead to Perspectives in Eclipse [**?**] to see how to navigate around the Eclipse environment.
After installing Eclipse there is a section explaining Eclipses functionality and how to get the most out of your STM32 development board. Some of the installation steps may have already been completed in preparation for STM32 development in your previous courses, just skip those installations if they have already been performed on the computer you are setting up.

1.2.1 Installation

gcc-arm-none-eabi

As with regular assembly development for the STM32F051 (hence forth the STM32), the standard arm development tools are needed to assemble our written source code (main.s) into an Assembly coded object file (main.o) and finally to link that object file to the particular memory addresses of our target processor (main.elf). So download and install the **GCC-ARM-NONE-EABI toolset** from Vula.Otherwise get it from the official site[1].
Remember that this is a 32-bit version of the development tools consisting of 32-bit executables. Even if you are running a 64-bit system you will still be using these tools. Once the files are extracted you will be prompted to **"Add path to environment variable"**


GNU Tools for ARM Embedded Processors 4.8 2014q2 Setup

Install wizard Complete

The install wizard has successfully installed GNU Tools for ARM Embedded Processors. Click Finish to exit the wizard.

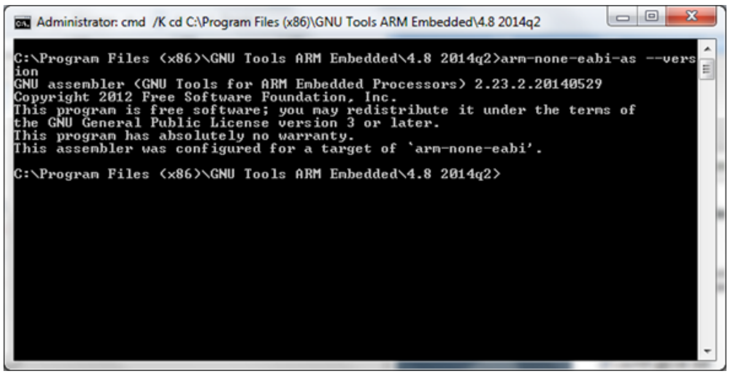
- [ ] View Readme
- [x] Launch gccvar.bat
- [x] Add path to environment variable

[1] https://launchpad.net/gcc-arm-embedded

You'll want to select this parameter, it adds the location of all the .exe's to your Windows advanced system settings so they can be run from any location without having to navigate to **C:\Program Files (x86)\Gnu Tools Arm Embedded\...**
You don't need to worry about the Readme or GCCVAR.bat, so deselect both and finish the installation. Included in the GNU Tools we've just installed are debugging, compiling and linker tools provided specifically for Arm development, its' an open source tool chain which you can read about on the official GCC-Arm-Embedded website. Just to ensure that it is installed correctly (and the system path has been added to your environment variable correctly), from your command line run;

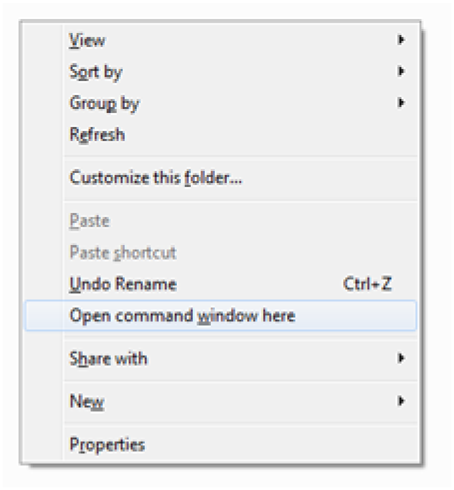
```
$ arm-none-eabi-as –version
```

It should produce some details about the current GNU assembler version (this test is applicable for any of the GCC Arm tools; **ls**, **gdb** etc . . .)

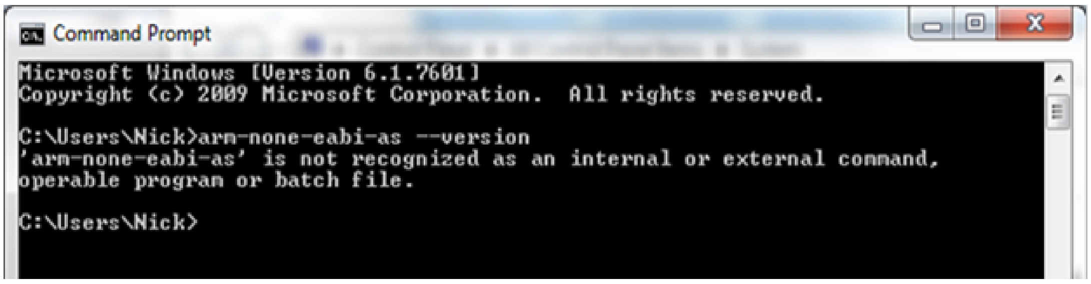


```
Administrator: cmd /K cd C:\Program Files (x86)\GNU Tools ARM Embedded\4.8 2014q2
C:\Program Files (x86)\GNU Tools ARM Embedded\4.8 2014q2>arm-none-eabi-as --vers
ion
GNU assembler (GNU Tools for ARM Embedded Processors) 2.23.2.20140529
Copyright 2012 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or later.
This program has absolutely no warranty.
This assembler was configured for a target of `arm-none-eabi'.

C:\Program Files (x86)\GNU Tools ARM Embedded\4.8 2014q2>
```

A useful trick to note is that, within any file in windows, pressing the SHIFT key and right clicking inside the folder gives you the ability to open a command window (CMD) at the files current location. This isn't really important but it might come in handy.



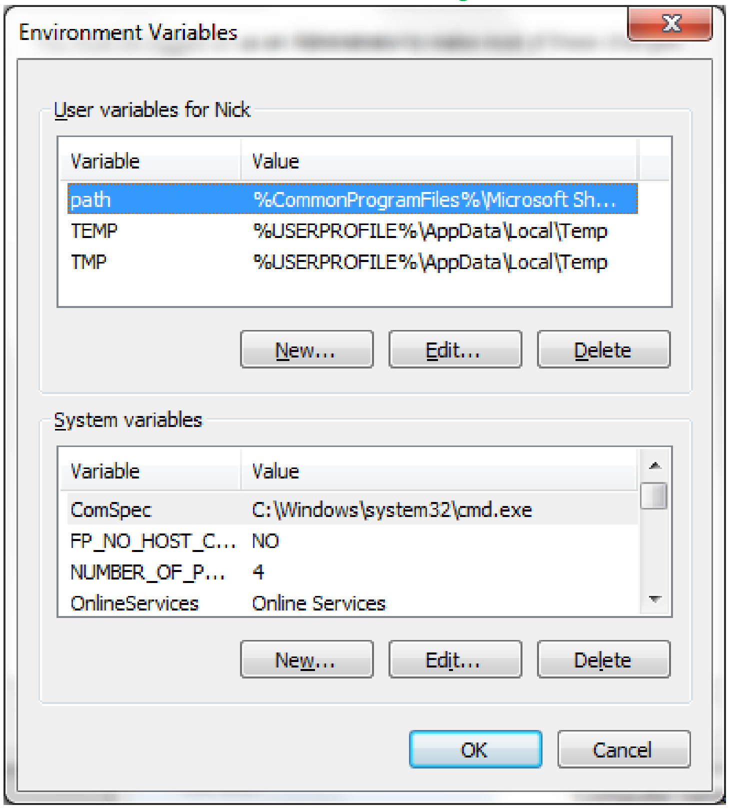
If you receive an error in which the command is unrecognized it means you either have not installed the tools correctly or the environment variable path has not been appended.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Nick>arm-none-eabi-as --version
'arm-none-eabi-as' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Nick>
```

To add the path variable you must navigate to:

Control Panel → Advanced System Settings → Environment Variables



Environment Variables

User variables for Nick

Variable	Value
path	%CommonProgramFiles%\Microsoft Sh...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	4
OnlineServices	Online Services

We want to edit the **path variable** , in the variable value paste (the semi-colon is VERY IMPORTANT):

C:\Program Files (x86)\GNU Tools ARM Embedded\4.8 2014q2\bin;

and then accept this by pressing OK until you are out of the system properties tab. Depending on where you installed your Tool Chain to be, this location may change so find the **bin** folder within that location and add that value to the path variable

Install OpenOCD

Next we need to install OpenOCD, the on chip debugging utility that lets us upload our code onto our target STM32. The debugging process takes place between the ST-Link device and the target STM32. What you may not realize is that the ST-Link is actually another specially programmed STM32 microcontroller. The ST-Link receives code from the computer, pretty much line by line, it then prepares our target STM32 to receive new code to be flashed to its memory by pulling some pins high/low in a particular order and finally it feeds our code to the target micro via a communication standard called JTAG.
Download OpenOCD-0.8.0.zip from Vula or the OpenOCD Website[2] and extract its' contents somewhere useful like in **C:\Program Files\OpenOCD**

Remember the location (file name) to which you extracted OpenOCD to, mine is called OpenOCD and is in the location listed above, and yours might be different.

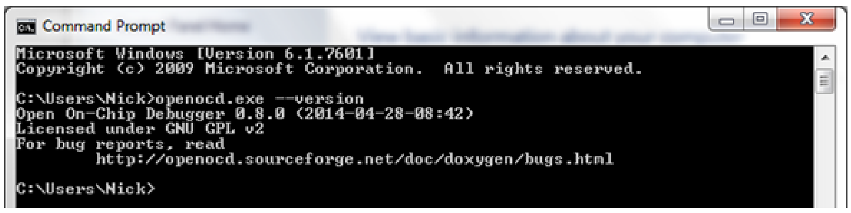
Then navigate to **C:\Program Files\OpenOCD\bin-x64** and then rename executables:
openocd-x64-0.8.0.exe to **openocd.exe**
This just makes things a bit easier for us. We now want to add OpenOCDâĂŹs path to the Environment Variables so once again, navigate to:
Control Panel → System → Advanced System Settings → Environment Variables. . .
Editing the path variable, we will see there is already a value for arm-none-eabiâĂŹs tools which was added in Step 1, so after this we append or paste in the location of OpenOCD:
(NOTE: the semi-colon is VERY IMPORTANT)
C:\Program Files\OpenOCD\bin-x64\openocd.exe;

Make sure there are no spaces between each subsequent path variable, only semi-colons. Mine looks like:
C:\ MinGW\bin;C:\ Program Files (x86)\ GNU Tools ARM Embedded\ Q2\ bin;C:\ Program Files\ OpenOCD\ bin-x64;
If there are spaces after the semi-colons, windows won't recognize the paths as system variables and it won't work!

Finally, test the path has been added correctly by running the following in the command line:

$ openocd.exe –version

And you should be rewarded with some more version details. . .

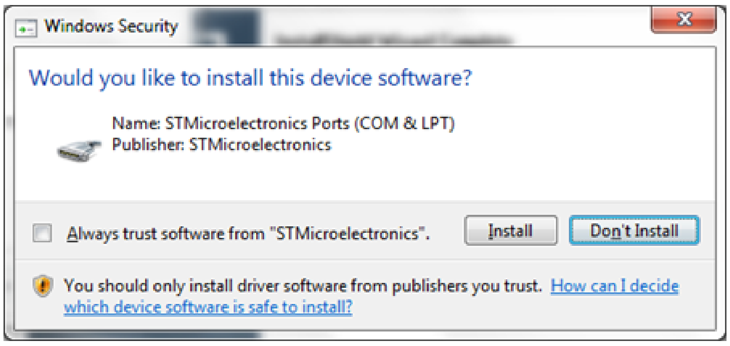


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Nick>openocd.exe --version
Open On-Chip Debugger 0.8.0 (2014-04-28-08:42)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html

C:\Users\Nick>
```

[2] http://openocd.org/

Install ST-Link Utility

As mentioned above, the on chip debugger makes use of the ST-Link device (we use the ST-Link V2 in particular) so we obviously need some drivers and programs for all this to work. Download the ST-LINK Utility from Vula or ST Electronics' Site[3] and install the utility. This will also install the ST-Link USB drivers as well as the device utility (choose to **Trust/Install** the STMicroelectronics Ports (COM & LPT) device software)


Windows Security

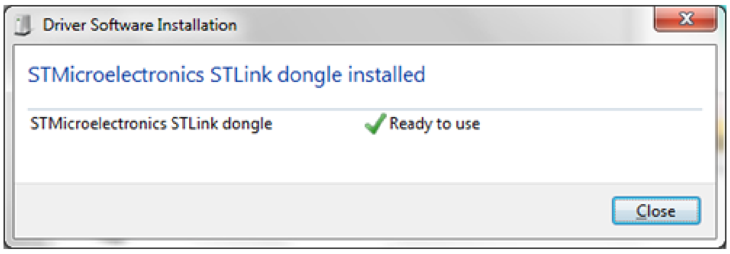
Would you like to install this device software?

Name: STMicroelectronics Ports (COM & LPT)
Publisher: STMicroelectronics

Always trust software from "STMicroelectronics". Install Don't Install

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

For some reason the installation attempts to install the device drivers twice, just click **Next** and it will take you through to the end of the installation. Now if you connect your STM32 board to your computer it should automatically detect the device (it will search for the drivers for a while and eventually give you a ready to use message)


Driver Software Installation

STMicroelectronics STLink dongle installed

STMicroelectronics STLink dongle ✓ Ready to use

Close

Whilst the USB drivers are crucial for us, the actual ST-Link utility isn't necessary but it is very useful. If you really mess up the program running on the micro or incompletely flash a program to it, it will let you erase the memory on board your STM32 micro.
Run the STM32 ST-Link Utility from wherever you installed it to and press the **Connect to the Target** button . There should now be a flashing green and red LED on your board and the table in the program will be populated with memory addresses (from 0x0800000 onwards) and hexadecimal numbers.

[3] $http : //www.st.com/content/st_com/en/products/embedded - software/development - tool - software/stsw - link004.html$