

## Experiment 0: Getting Started

### 1.0 Introduction

In ECE362, students will learn fundamentals of microcontrollers, including their operation, and usage. In addition to the lecture portion of the course, students are expected to complete a series of lab experiments using a microcontroller platform and IDE. Computers are provided in the course laboratory facilities to assist students in completing labs, however, the course has also been designed to allow students to perform experiment exercises on their own computers in the comfort of their own homes. These instructions are intended to serve as a guide to setting up the microcontroller development environment used in ECE362 to provide a consistent user experience between home and the laboratory.

### 2.0 Tools and Software

At the time of this writing, the following tools and equipment are needed to perform the ECE362 experiment exercises:

- **STM32F0-Discovery Development Board:** The course uses the STM32F0-Discovery board, [available from STMicroelectronics](#) [1]. The board is a low-cost development platform for the ARM Cortex M0-based STM32F051R8 microcontroller. The board is available from many online vendors; some vendors are provided below:
  - + [ST Microelectronics](#) [2]
  - + [DigiKey](#) [3]
  - + [Mouser](#) [4]
  - + [Arrow Electronics](#) [5]
  - + [Newark](#) [6]
  - + [Octopart](#) [7]
- **Software:** Embedded software development for ECE362 is focused around System Workbench, which consists of the Eclipse IDE and the STMicroelectronics OpenSTM32 environment. Eclipse is open source and cross-platform, and should be usable on Windows, Mac OSX, and Linux. This software is already installed on the computers in the lab. Refer to the instructions in Section 3 to learn how to install it on your own computer.

Later experiments require additional equipment that is included in the development kit you purchase.

### 3.0 System Workbench Setup Instructions

(Remember that you do not need to install System Workbench on a lab computer. It has been done for you.)

For embedded software development within ECE362, the open-source Eclipse IDE will be used, alongside the “System Workbench for STM32” Eclipse plugin (supported by STMicroelectronics). Full instructions for the installation of this software is OpenSTM32 installation instructions, available [on the OpenSTM32 web site](#) [8]. While it is possible to install components individually, the lab staff finds that the environment works easiest and best if a pre-integrated system is installed. The following platform-specific instructions will allow you to do that.

### 3.1 Installation on Linux

The specific version of Linux supported is Ubuntu LTS. Other distributions may work, but they may involve additional steps not detailed here. Download the latest version of System Workbench from:

[http://www.ac6-tools.com/downloads/SW4STM32/install\\_sw4stm32\\_linux\\_64bits-latest.run](http://www.ac6-tools.com/downloads/SW4STM32/install_sw4stm32_linux_64bits-latest.run)

Then invoke the installation script with the following command:

```
bash install_sw4stm32_linux_64bits-latest.run
```

You will be prompted for installation locations as well as the system password to install the STLink driver configuration. Depending on your particular computer, you may need to issue an additional command for the software to work properly:

```
sudo apt install lib32ncurses5
```

The installation script should set up a link on your desktop by which you can invoke System Workbench. If not, invoke it with [the path to the installation directory]/eclipse.

### 3.2 Installation on MacOS

Download the latest version of System Workbench from:

[http://www.ac6-tools.com/downloads/SW4STM32/install\\_sw4stm32\\_macos\\_64bits-latest.run](http://www.ac6-tools.com/downloads/SW4STM32/install_sw4stm32_macos_64bits-latest.run)

Then invoke the installation script with the following command (in a terminal shell):

```
bash install_sw4stm32_macos_64bits-latest.run
```

The installation script should set up a link on your desktop by which you can invoke System Workbench. If not, invoke it with [the path to the installation directory]/eclipse.

### 3.2 Installation on Windows

Download the latest version of System Workbench from:

[http://www.ac6-tools.com/downloads/SW4STM32/install\\_sw4stm32\\_win\\_64bits-latest.exe](http://www.ac6-tools.com/downloads/SW4STM32/install_sw4stm32_win_64bits-latest.exe)

Then invoke the installer.

#### 3.2.1 STLink Driver Installation on Windows

The STM32F0-Discovery development board includes an integrated programmer, known as STLink, which can be used to program the onboard microcontroller or other microcontrollers. In order to interact with the STLink programmer for programming and debugging functions, an appropriate driver must be installed. Acquire the driver from [STMicroelectronics](#) [9], extract the resulting zip file, and install the driver.

## 4.0 Configuring Eclipse Workspace Preferences

Now that all software tools needed for the STM32F0 have been installed, the Eclipse environment should be configured for student use. The workspace preferences configuration instructions are heavily based on a set of instructions from the GNU ARM Eclipse community, which can be found [here](#) [10]. To access the Eclipse Preferences menu, select **Window >> Preferences** from the Eclipse main menu.

### 4.0.1 General Workspace Preferences

The first destination is Workspace Preferences, located at **General >> Workspace** in the Preferences menu. Uncheck the box which says “Build automatically” (in these experiments, the build process will be initiated via a command). Additionally, check the box titled “Save automatically before build”.

Text encoding and newline delimiters can vary from operating system to operating system, and should be standardized for the purposes of this course. At the bottom of the Workspace preferences pane, there are two fields with radio buttons: “Text file encoding” and “New text file line delimiter”. Ensure that the Text file encoding field is specified as UTF-8, while the New text file line delimiter is specified to use Unix-style endings. Once all of these changes have been made, click **Apply**.

All of these settings are summarized in figure 1, below:

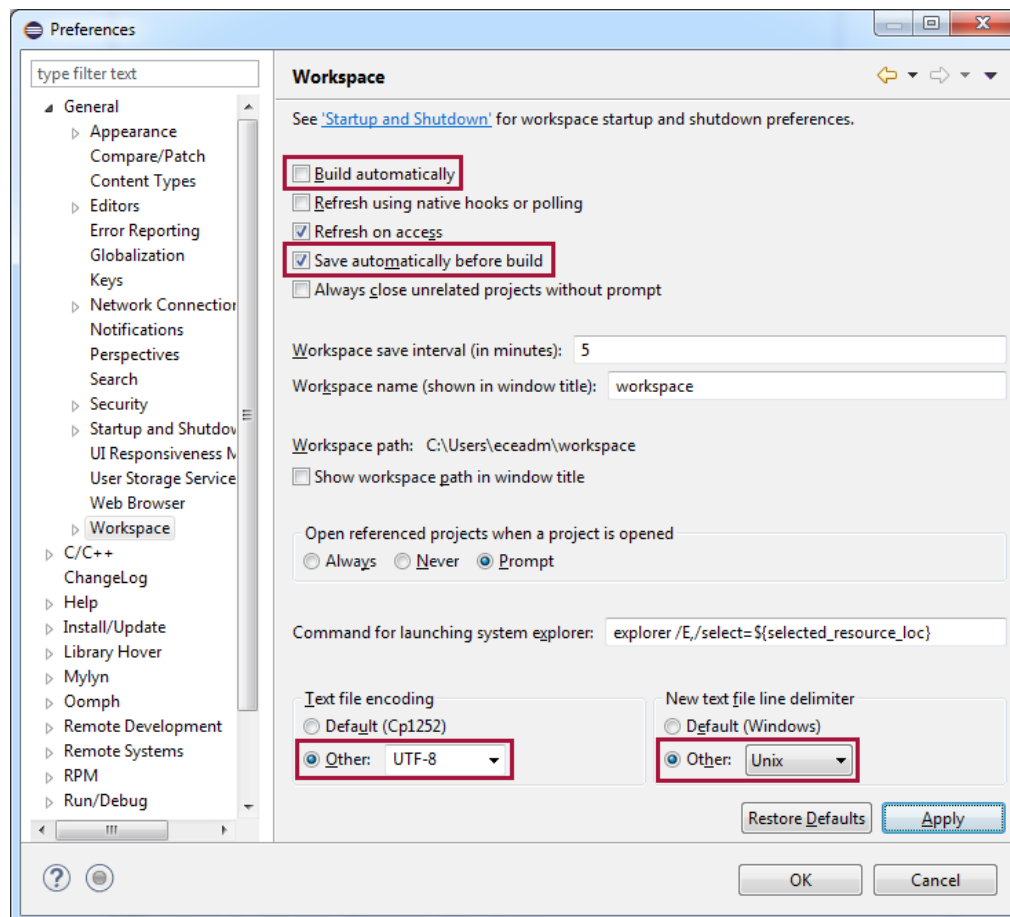


Fig. 1. General Workspace Preferences Configuration

#### 4.0.2 General Text Editor Preferences

Occasionally, source code will have to be printed. When this is done, it is helpful to know where the edge of a page would be, that lines of code can be written of appropriate lengths. Under the preferences hierarchy, navigate to **General >> Editors >> Text Editors**. In the Text Editors preferences pane, click the box labeled “Show print margin” and ensure that the “Print margin column” field is set to 80. Once these preferences have been made, click **Apply**.

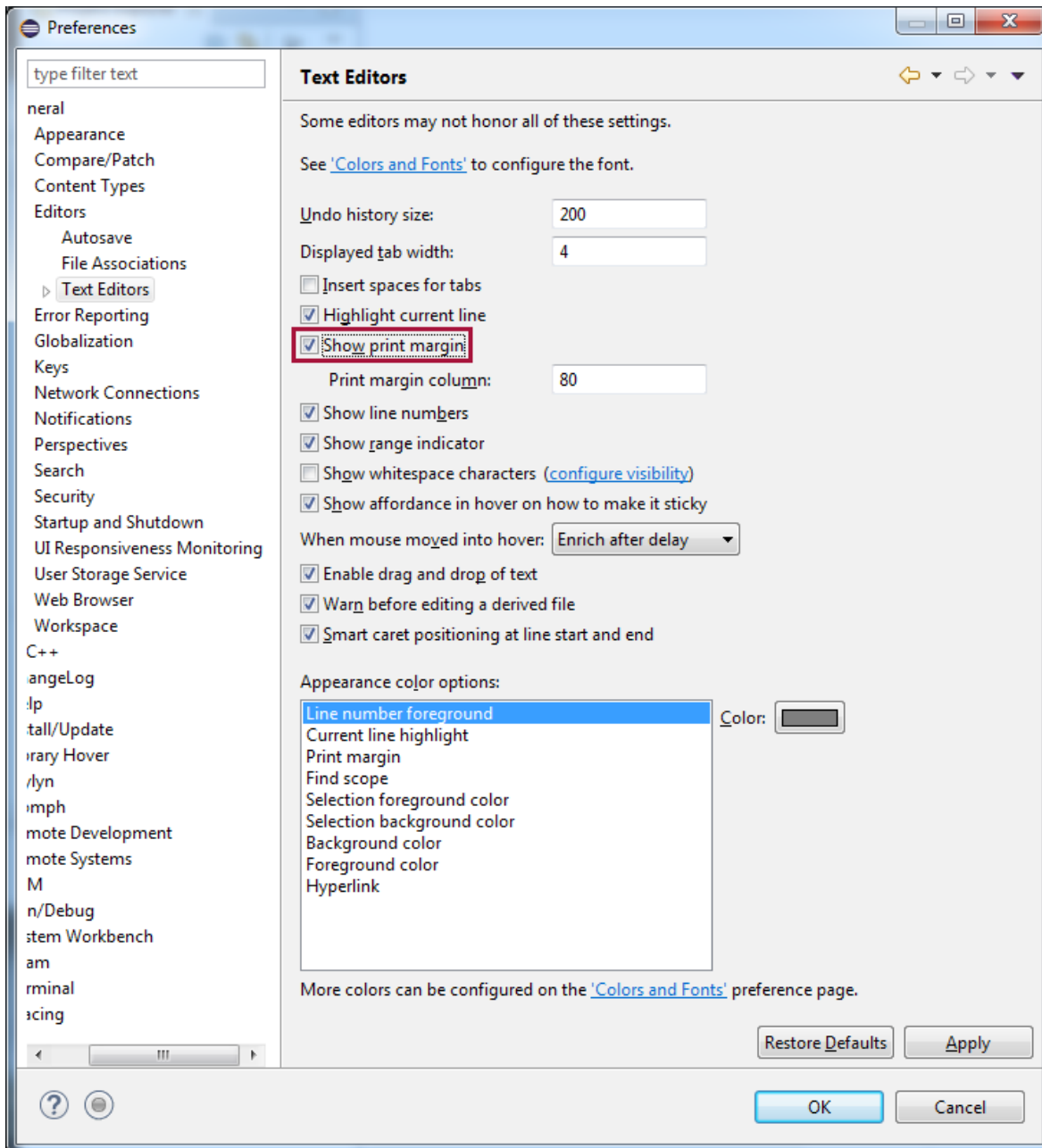
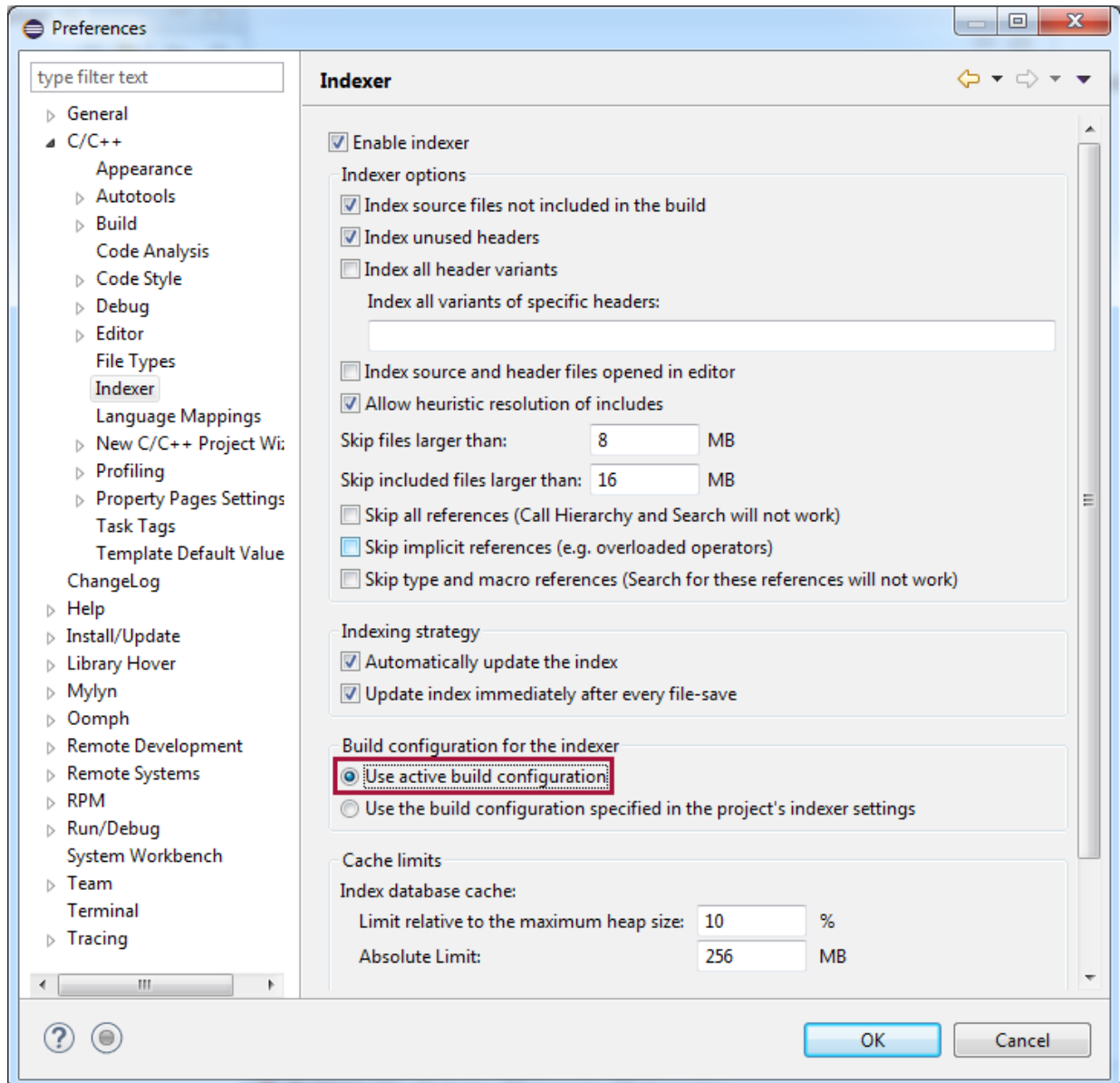


Fig. 2. General Text Editor Preferences Configuration

### 4.0.3 C/C++ Indexer Preferences

Certain IDEs, including Eclipse, feature a software tool called an Indexer. An indexer parses source code on the fly, providing hints, auto-completion suggestions, and error reporting to the developer during the process of editing, without having to initiate the build process. In order for the indexer to provide accurate and relevant hints, the indexer must be in sync with the compiler.

In the Preferences menu, navigate to **C/C++ >> Indexer**. In the field titled “Build Configuration for the Indexer”, select the field titled “Use active build configuration”. Click **Apply**.



**Fig. 3. C/C++ Indexer Preferences Configuration**

#### 4.0.4 C/C++ Code Formatting Settings

Eclipse features a powerful integrated code formatter, which is capable of automatically formatting source code files to conform to a particular coding style of choice. Eclipse will automatically adhere to the settings of the code formatter when working on new source files. Alternatively, a file open in Eclipse can be automatically formatted using **Source >> Format** from the Eclipse main menu.

To modify the formatter settings, navigate to **C/C++ >> Code Style >> Formatter**. In the resulting pane, a drop down box will appear showing different code formatting styles, as well as a preview of what source code might look like in the resulting style. Select your style of choice, then click the **Edit...** button. A new window will appear, showcasing code formatter style configuration settings.

The built-in code formatter styles cannot be modified, so press **New** and a dialog window will open with a field labeled “Profile Name”. Provide a name for the new code formatting style (for the purposes of this exercise, the profile name “ECE362” was used). Choose “K&R C” as a template. To ensure uniformity of code across development environments, set the field labeled Tab Policy to “Spaces Only”, then click **OK**. You will be returned to the Eclipse Preferences menu. Click **Apply**.

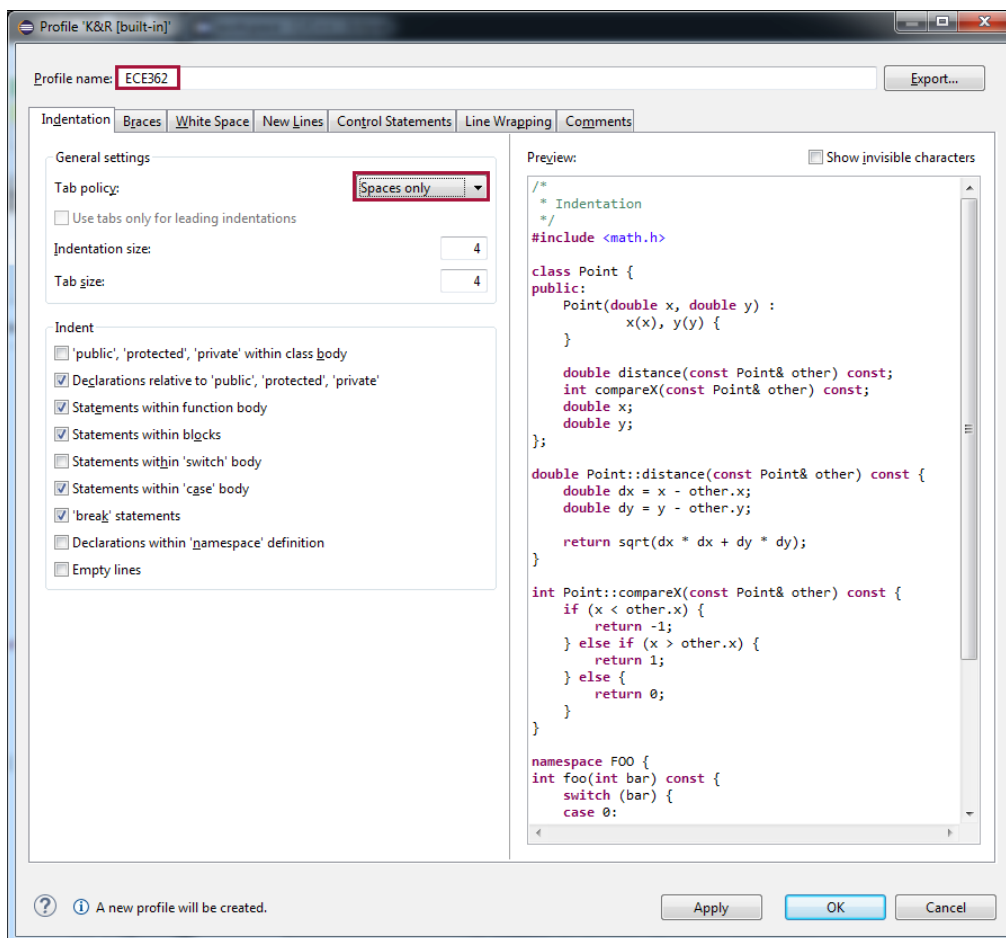


Fig. 4. Code Formatter Style Preferences Configuration

#### 4.0.5 Build Console Settings

The build console is a portion of the Eclipse UI which is displayed during the build process. Build console settings can be found under **C/C++ >> Build >> Console** in the Eclipse Preferences menu.

To ease the build process, check the boxes in the Build Console Preferences pane stating “Bring console to top when building (if present)” and “Wrap lines on the console”. Additionally, increase the field “Limit console output (number of lines):” to a higher value, such as 5000. Click **Apply**.

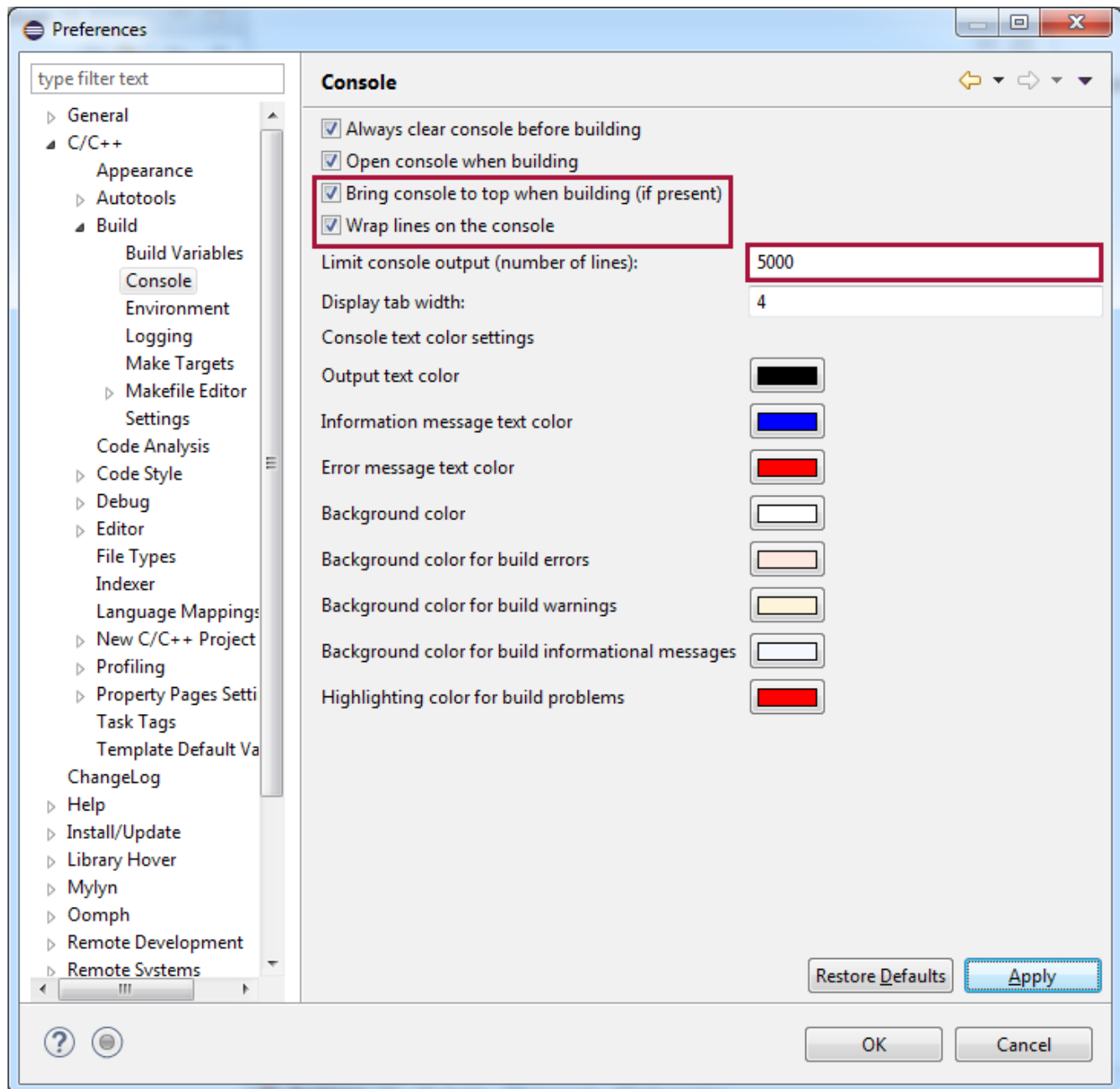
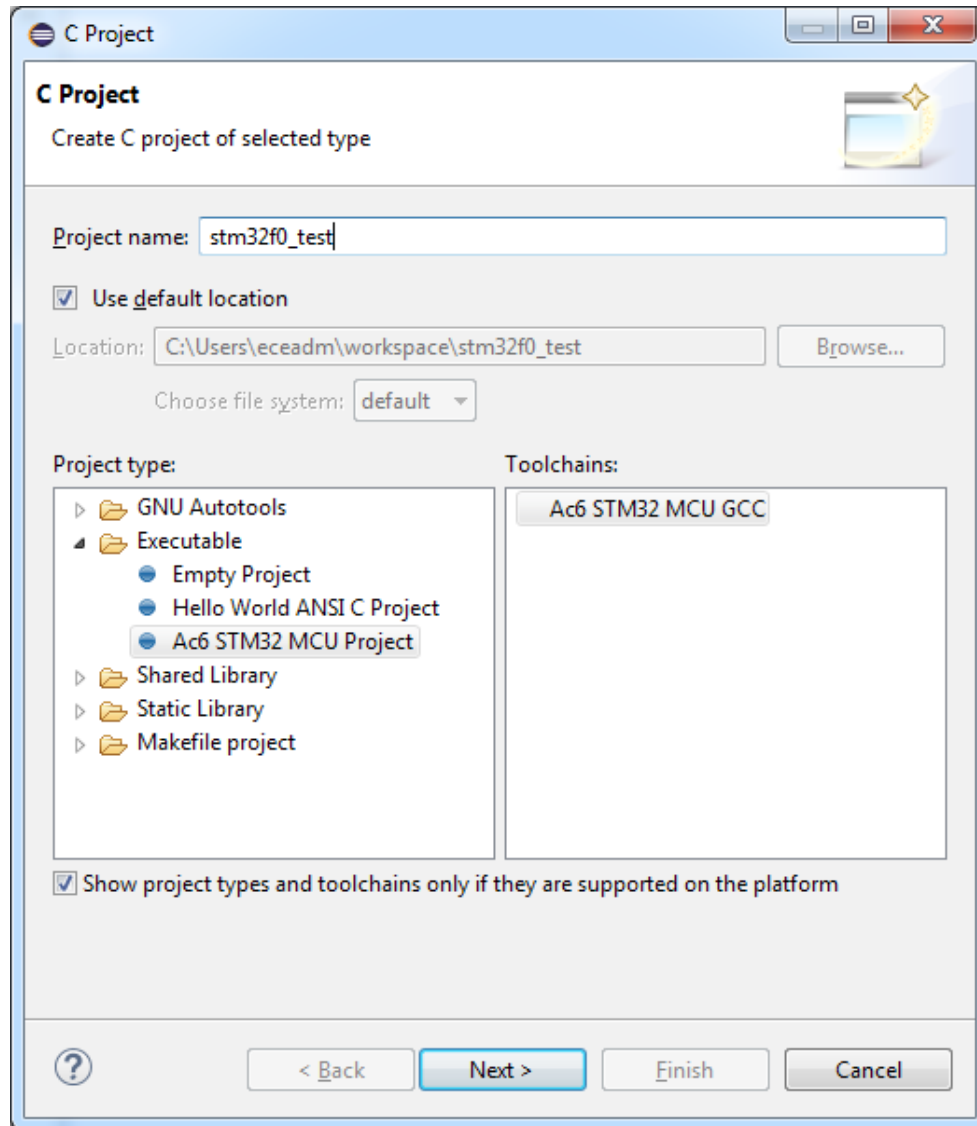


Fig. 5. Build Console Preferences Configuration

## 5.0 A Simple STM32F0 Project

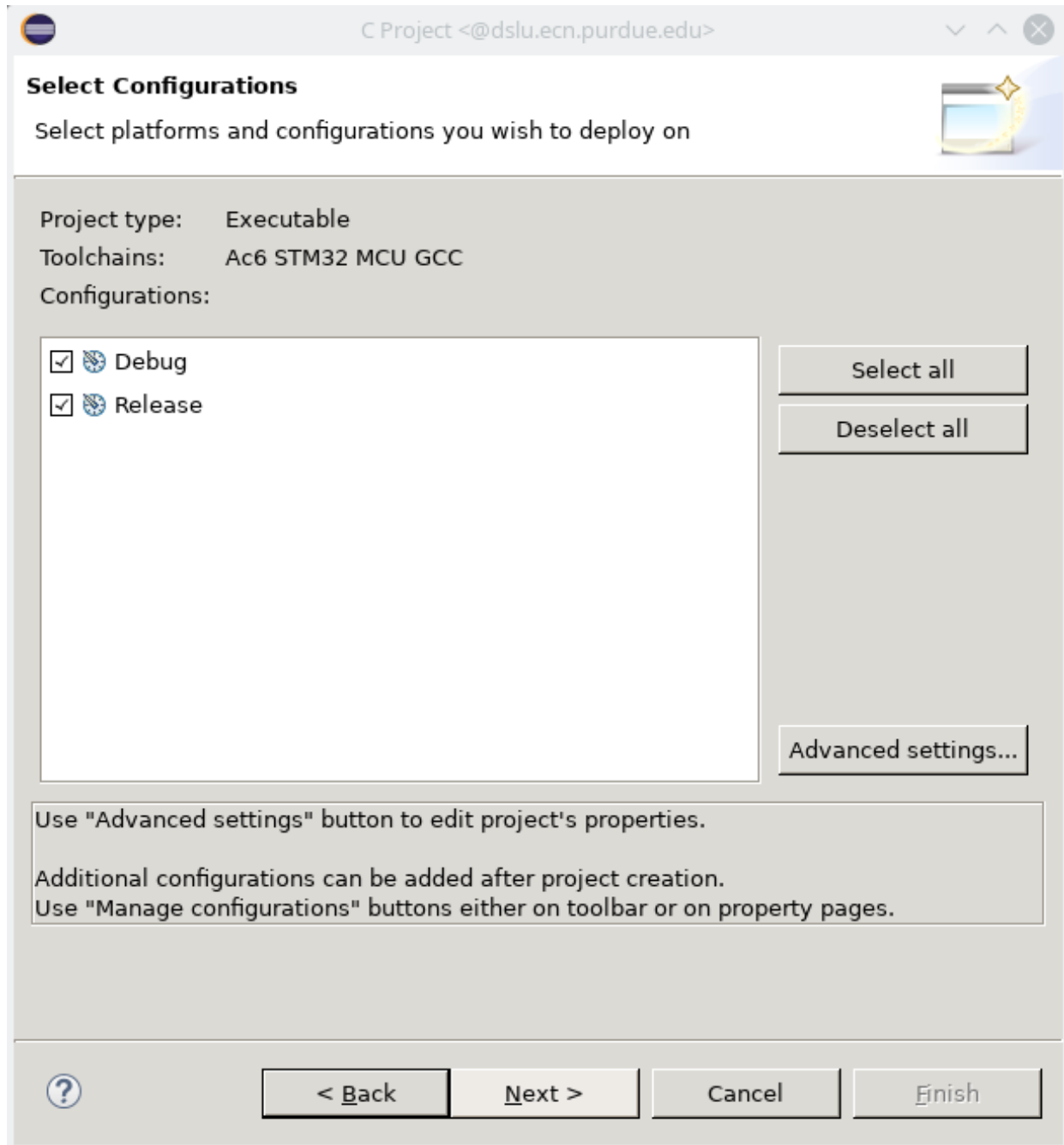
Now that the software toolchain is configured, it's time to create a simple project for the STM32F0. Connect the STM32F0 Discovery board to the computer, and start Eclipse (on lab computers, this may be called "System Workbench"). A Welcome Screen will appear, which can be closed. In the top level Eclipse menu, select **File >> New >> C Project**. This will take you to the C Project window. In that window, specify "Ac6 STM32 MCU Project", specify a **Project name** ("stm32f0\_test" was used, but you can pick any name you want) and click **Next**.



**Fig. 6. C Project Configuration Settings**



**5.1** A "Select Configuration" dialog box will appear next. It is asking if you want to prepare the "Debug" or "Release" versions of the software (or both). By default, both are selected, and you may keep this configuration. Click **Next**.



**Fig. 7. Select Configurations Dialog**

5.2 Next, a “Target Configuration” screen appears. Here, select the board “STM32F0DISCOVERY” from the series “STM32F0” (it’s always at the bottom of the drop-down menu), then click **Next**.

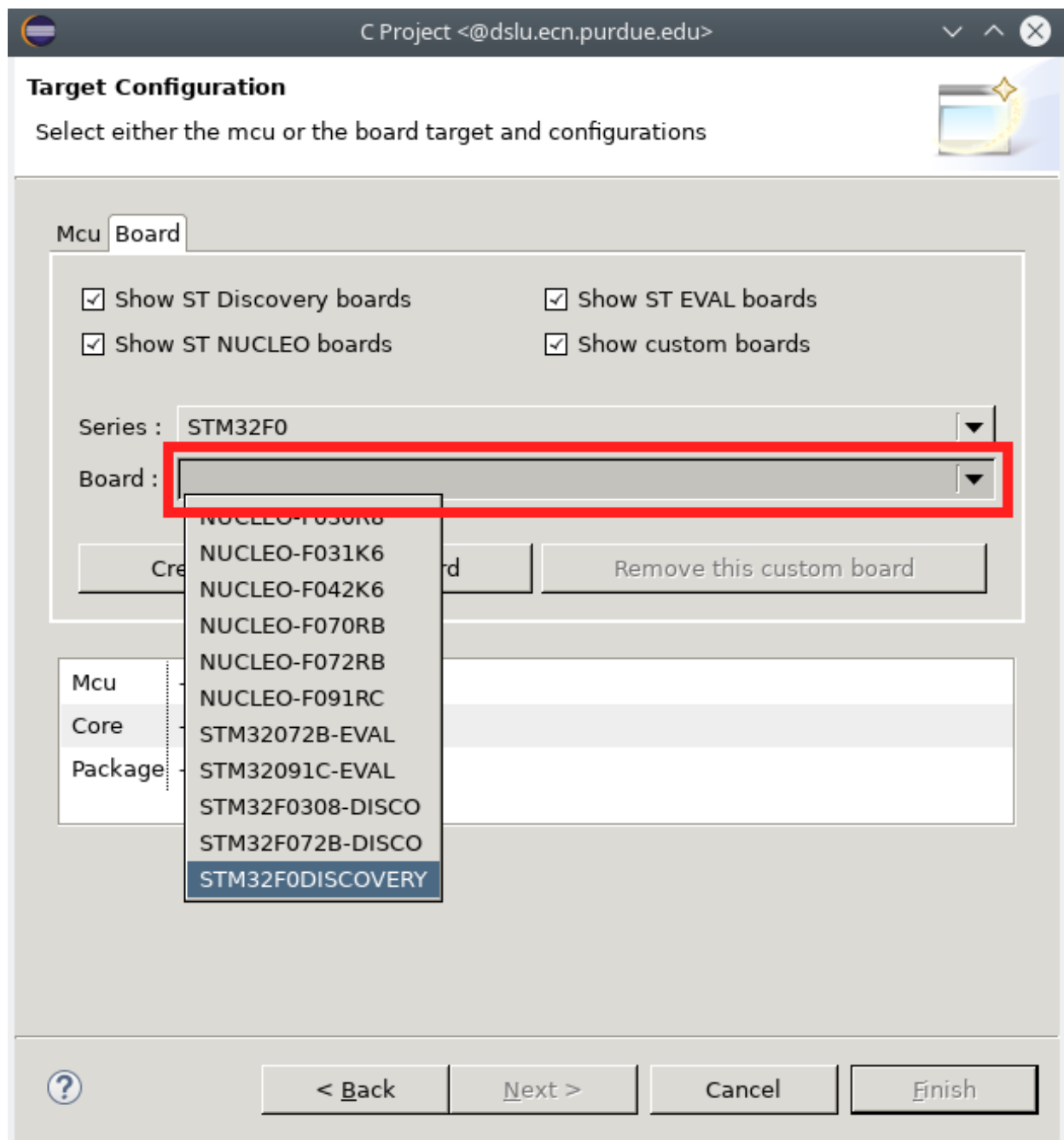
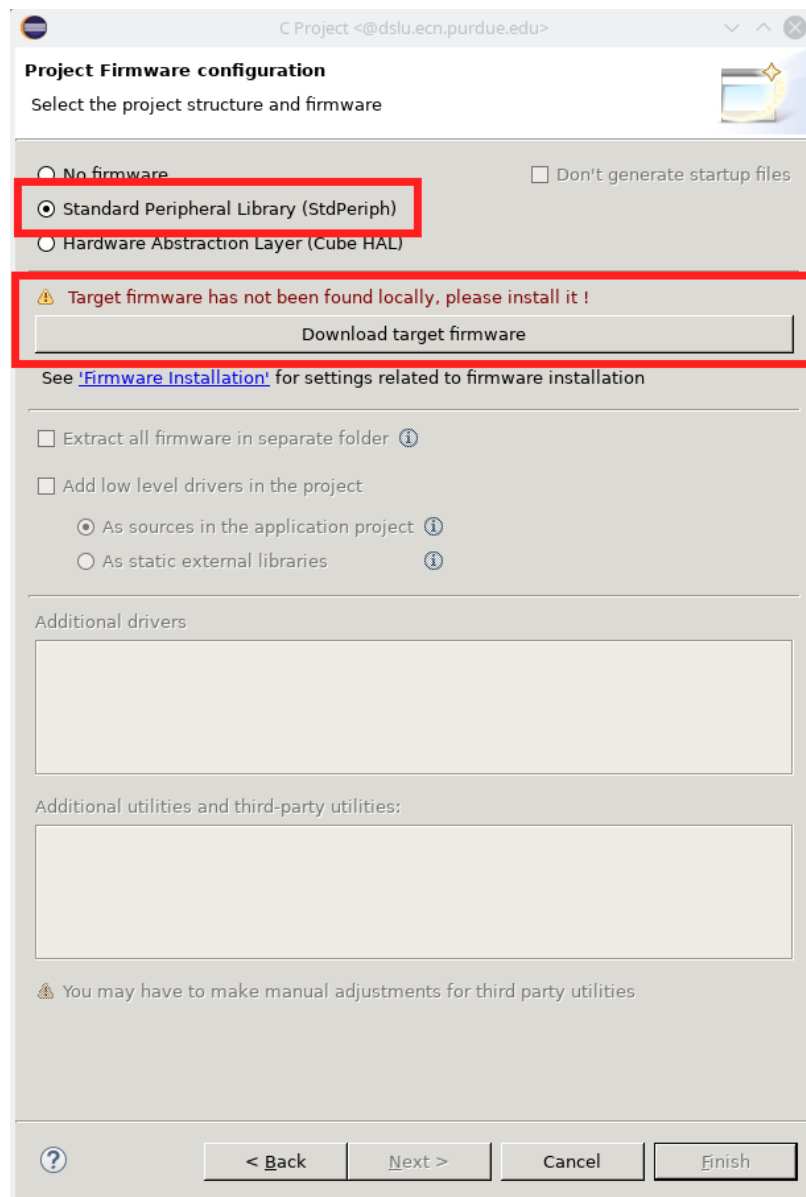


Fig. 8. Target Configuration Dialog

5.3 Next, the "Project Firmware configuration" dialog appears to ask you what type of firmware you should use for the project you are creating. For all projects in this course, we will use either "No Firmware" or the "Standard Peripheral Library (StdPeriph)" firmware. For this experiment, **choose "Standard Peripheral Library (StdPeriph)"**. Since this is the first time you have used this firmware type, it will warn you that the firmware is not installed and prompt you to download it. Press the "Download target firmware" button, accept the license agreement, and wait a few seconds for it to complete. *If System Workbench fails to download the StdPeriph firmware*, you may simply select "No firmware." There will be no significant difference for this week's lab exercise.

Then press **Finish**.

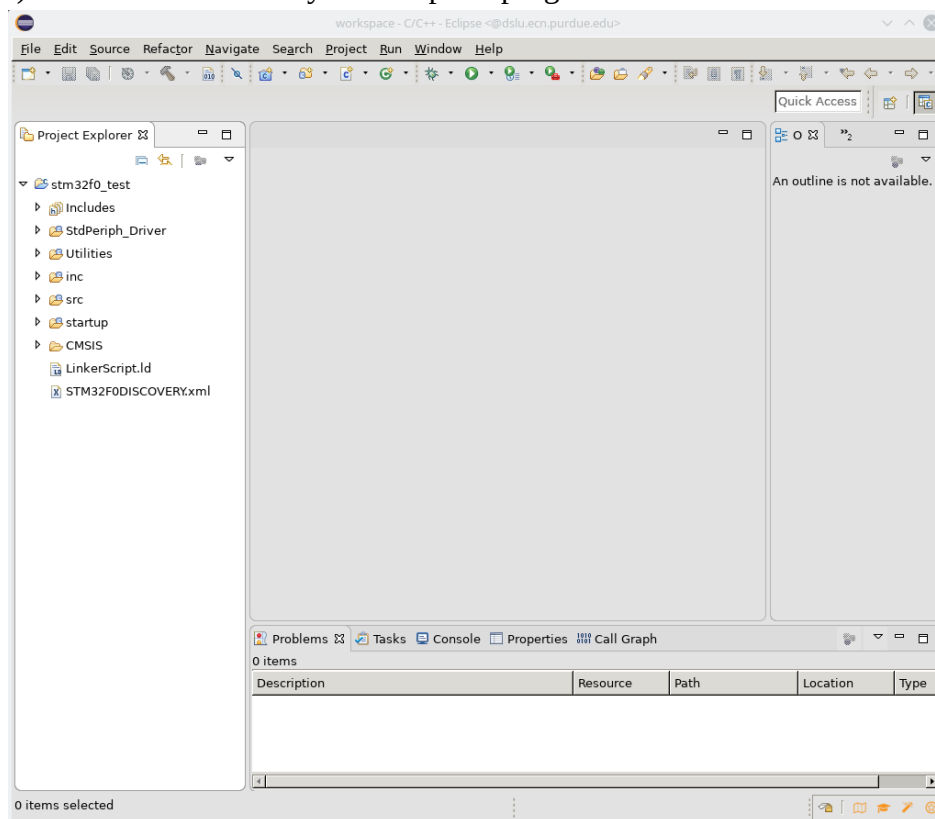


**Fig. 8. Target Configuration Dialog**

**5.4** Finally we arrive at the main project workspace where we can start manipulating files related to the software project we wish to build. Eventually, you will become accustomed to going through the steps of creating a new project and it will become an automatic action for you. (You will usually create one new project for every prelab, lab, and homework that you write.)

If you click open the "stm32f0\_test" element in the Project Explorer on the left, you will see the following top-level folders:

- Binaries: Folder containing the “release version” of your compiled program.
- Includes: Global folder containing all of the C header files utilized in the project.
- inc: Folder containing project-specific C header files used in the project.
- src: Folder containing C source files (including main) used in the project.
- startup: Folder containing device-specific startup files. Startup files are generally provided as assembly files (for the purposes of ECE362, STM32F0-series assembly files are provided).
- Debug: Folder containing the “debug version” of your compiled program.
- LinkerScript.ld is the configuration file that tells the linker where to put the segments (e.g. text and data) of the executable for your compiled program.

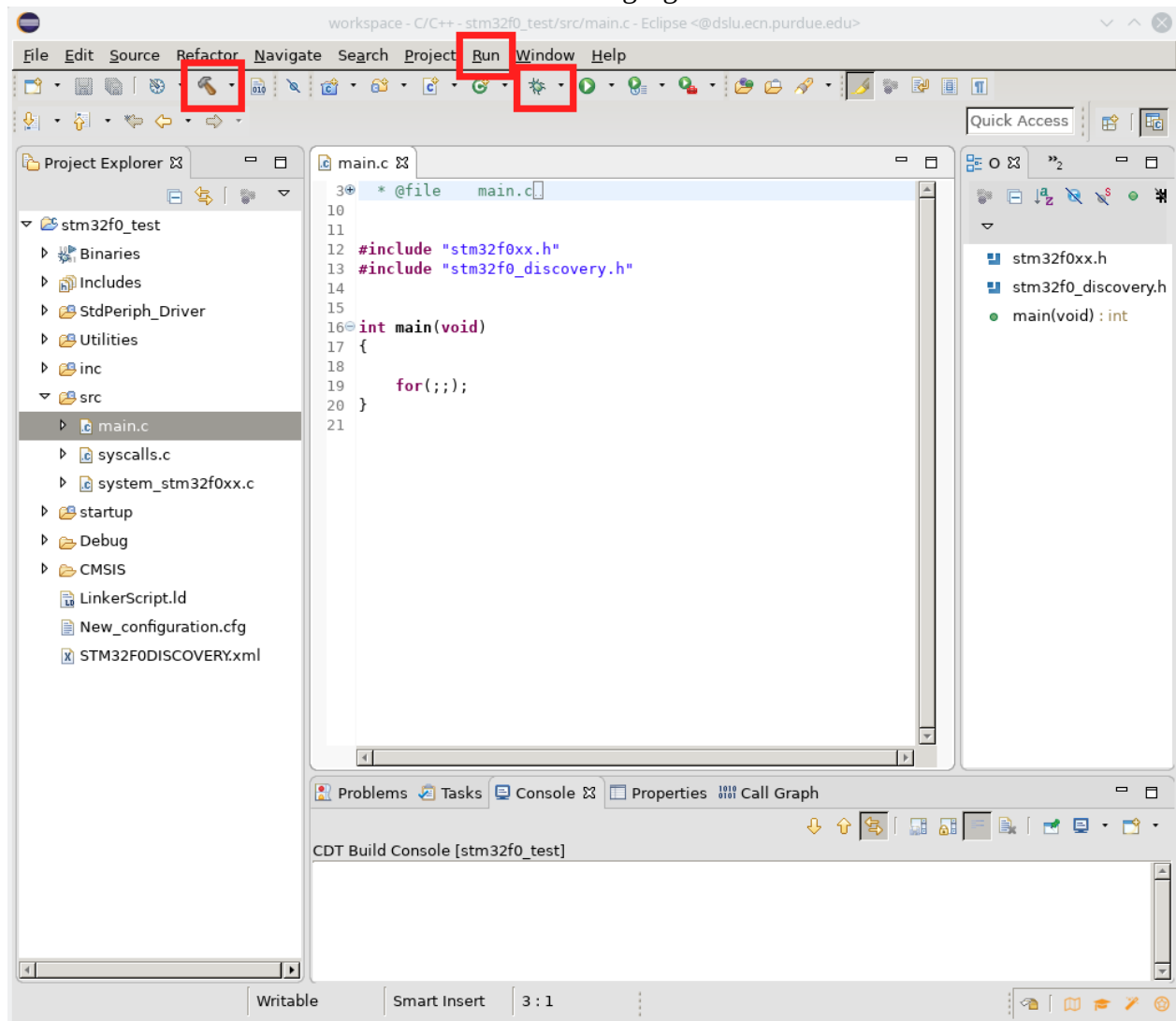


**Fig. 9. Project Workspace default view at startup.**

Double-click on the **src** folder to open it, then double-click on the **main.c** file to edit it.

## 5.5 Editing, building, and debugging a program

The editor window should look similar to the following figure:



**Fig. 10. Editor window**

System Workbench automatically created this "main.c" file for you. It includes a comment at the top of the window that looks like `"* @file main.c"` that is *folded*. If you click on the plus icon near the line number, it will unfold the comment, allowing you to view it.

If you press the hammer icon you will see many messages go by explaining that it is building firmware files and compiling the main.c file to produce an executable. We can run and debug the execution of this simple program on the development board that is at your lab station. Before we do so, we must configure the debugger.

### 5.5.1 Debug Configuration

Select the **Run >> Debug configurations...** menu item. A dialog will appear. Double-click on the "Ac6 STM32 Debugging" item on the left side of the screen to create a New\_configuration entry. Notice that the selection for "C/C++ Application" is empty in your dialog window. Press the "Search Project..." button and select the "stm32f0\_test.elf" entry that appears. (This is the executable that you just built in the previous step.) The result should look as follows:

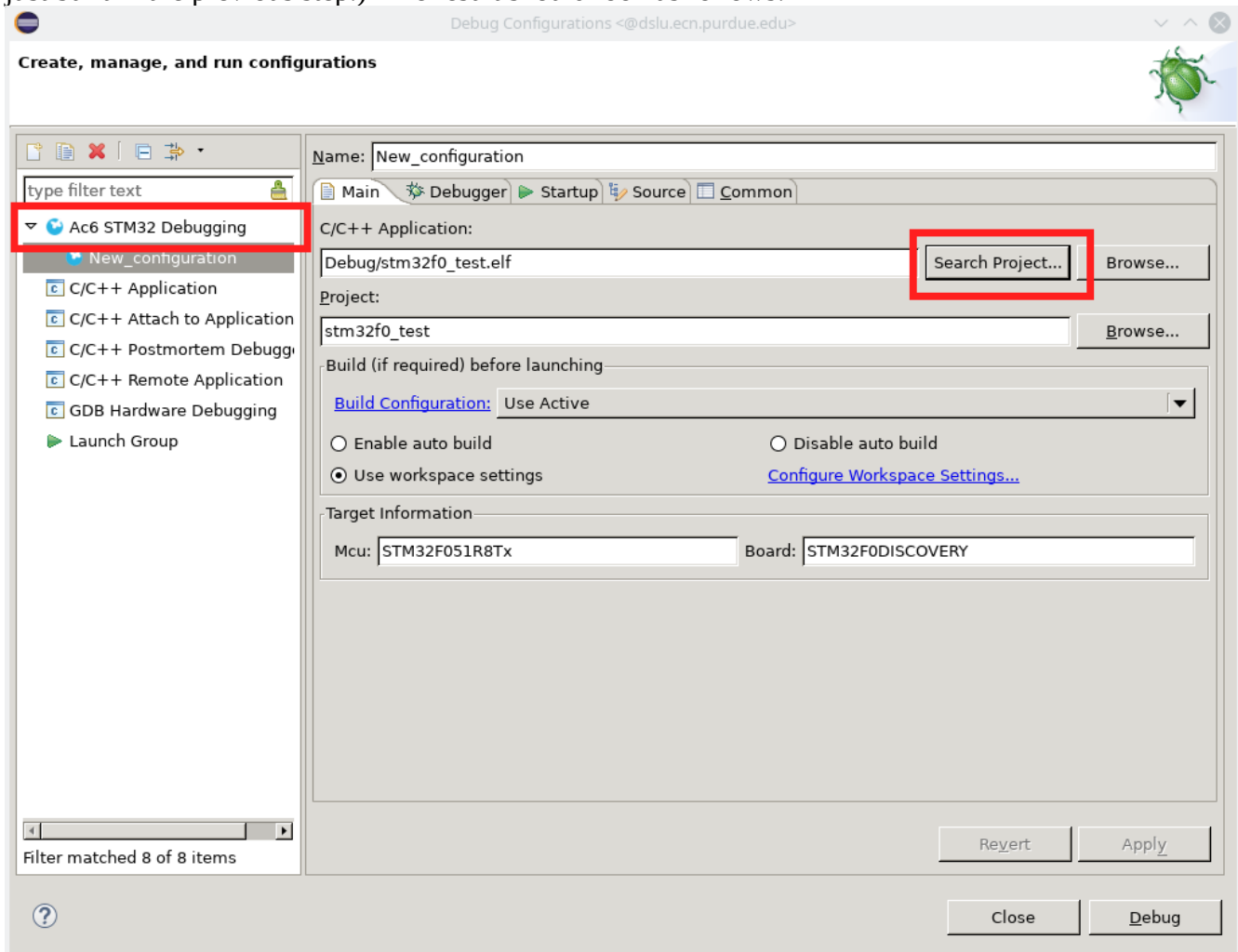


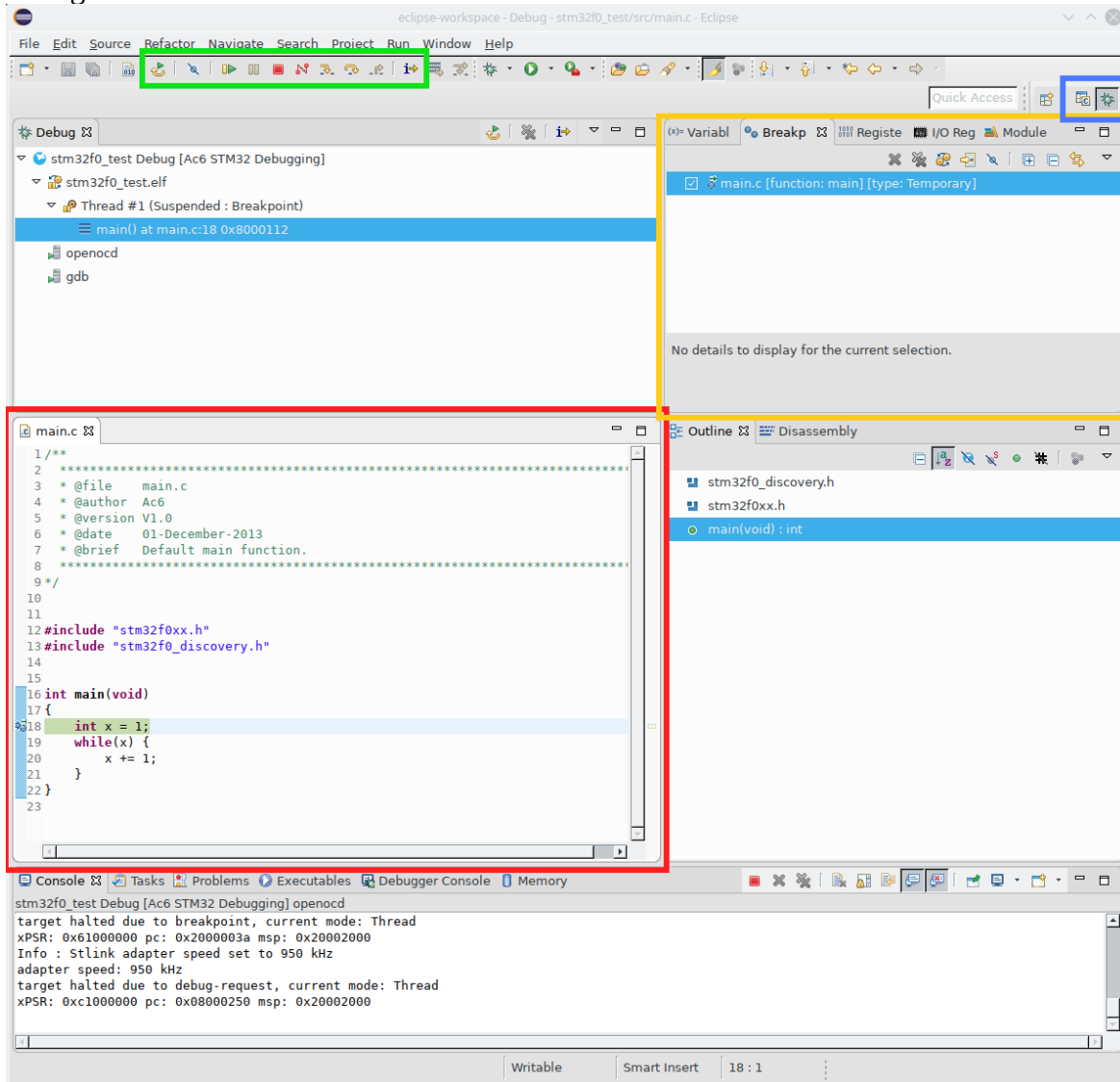
Fig. 11. Debugger configuration

Click **Apply** and **Close**. At this point, you will be able to use the debugger. Before doing so, change the main() function to look like this:

```
int main(void) {  
    int x = 1;  
    while(x) {  
        x += 1;  
    }  
}
```

### 5.5.2 Debugging

When you press the debug icon (the blue bug) below the menu bar, your project code should be rebuilt, the debugger will be invoked, and System Workbench will prompt you to change to a different *perspective*. Click **Yes** on the perspective change dialog. The new perspective for the debugger should look like Figure 12.









**Fig. 12. Debugger Perspective**

The editor is now reduced to a smaller portion of the screen (highlighted in red). There is now a series of buttons below the menu bar to control the debugger operation (highlighted in green). An information window (highlighted in orange) shows variable values, breakpoints, register contents, and I/O registers. Finally, a perspective selection buttons (highlighted in blue) can be used to switch back to the main editor perspective.

### 5.5.3 Using the Debugger

Hover over the debugger control buttons under the menu bar (highlighted in green in Figure 12) to learn their functions before doing the following steps.

- 1) Press either the "Step Into (F5)"  or "Step Over (F6)"  buttons several times. Notice how the highlighting in the editor window changes to show the statement that is ready to execute next.
- 2) In the information window in the upper right quadrant of the screen, select the "Variables" tab. Press the "Step Into" or "Step Over" buttons again to continue executing the C program. Notice how the value of the "x" variable changes.
- 3) Press the debugger control button that will "Reset the chip and restart debug session".  Notice how it does exactly what it says it does.
- 4) Press the "Terminate (Ctrl+F2)" button  and notice that all the debugger control buttons turn gray and become inoperative. To restart the debugger, press the debugger button again (small blue bug under the menu bar).
- 5) Once the debugger is restarted, press the button that will "Resume (F8)" . Notice how many of the debugger control buttons turn gray and become inoperative. At this point, your program is running unhindered on the development board.
- 6) Press the button that will "Suspend"  the debugger. Notice how the variable display shows a large value now. This happened because the microcontroller was allowed to run very quickly and update the variable as fast as it could.

**Demonstrate each of the previous 6 steps to your lab TA. Your TA will register your work for this lab.**



## 6.0 Sources Cited

- [1] *STM32F0Discovery*. ST Microelectronics. [Online]. Available:  
[http://www.st.com/content/st\\_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-kits/stm32f0discovery.html](http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-kits/stm32f0discovery.html)
- [2] *STM32F0Discovery Product Page*. ST Microelectronics. [Online]. Available:  
<https://my.st.com/esample2/app?page=basket&pn=STM32F0DISCOVERY>
- [3] *STM32F0Discovery Product Page*. DigiKey. [Online]. Available:  
<http://www.digikey.com/product-search/en?keywords=stm32f0-discovery>
- [4] *STM32F0Discovery Product Page*. Mouser. [Online]. Available:  
<http://www.mouser.com/ProductDetail/STMicroelectronics/STM32F0DISCOVERY/?qs=sGAEpiMZZMt4jl5TkKljYNdBh%2fXNAVtU>
- [5] *STM32F0Discovery Product Page*. Arrow. [Online]. Available:  
<https://www.arrow.com/en/products/stm32f0discovery/stmicroelectronics>
- [6] *STM32F0Discovery Product Page*. Newark. [Online]. Available:  
<http://www.newark.com/stmicroelectronics/stm32f0discovery/evaluation-board-cortex-m0-f0/dp/94T6483>
- [7] *STMicroelectronics STM32F0Discovery*. Octopart. [Online]. Available:  
<https://octopart.com/stm32f0discovery-stmicroelectronics-22099547>
- [8] *Installing System Workbench for STM32 from Eclipse*. OpenSTM32 Community. [Online]. Available: <http://www.openstm32.org/Installing+System+Workbench+for+STM32+from+Eclipse>
- [9] *ST-Link/V2-1 USB driver on Windows Vista, 7 and 8*. STMicroelectronics. [Online]. Available:  
<http://www.st.com/web/catalog/tools/FM147/SC1887/PF260218>
- [10] *Workspace Preferences*. GNU ARM Eclipse. [Online]. Available:  
<http://gnuarmclipse.github.io/eclipse/workspace/preferences/>