



MAX3263X SDK Getting Started In Eclipse User Guide

UG6245; Rev 0; 4/16

Abstract

This tutorial is a brief guide to getting started with Eclipse CDT IDE included in the Maxim ARM Cortex Toolchain. This guide provides steps on how to create, import, build and debug a project using the MAX3263X SDK.

CONTENTS

Introduction	4
1 Create a Workspace	5
2 Import Example Projects	6
3 Create New Project from Existing Project.....	8
3.1 Copy/Rename Project.....	8
3.2 Import Project	9
4 Build/Debug Project.....	11
4.1 Build Project	11
4.2 Debug Project	12
5 Register View.....	15
6 Add Header/Source Files to Project.....	17
7 Create New Project from Scratch.....	19
7.1 Create New Project	19
7.2 Setup Debug Configuration	22
7.3 Add Files to Project	26
8 Troubleshooting Unresolved Symbols.....	29
8.1 Add Include Paths to Project	29
8.1.1 Manually Add Include Paths.....	29
8.1.2 Export/Import Include Paths.....	30
9 Trademarks.....	33

FIGURES

Figure 1. Eclipse Workspace Launcher.	5
Figure 2. Import Project.....	6
Figure 3. Select Import Source.....	6
Figure 4. Select Project to Import and Copy.....	7
Figure 5. Project Folder Rename.....	8
Figure 6. Project Files.	8
Figure 7. Import Project	9
Figure 8. Select Import Source.....	9
Figure 9. Select Project to Import.	10
Figure 10. Build Icon.	11
Figure 11. Console Window.	11
Figure 12. Debug Drop-down.	12
Figure 13. Debug Configurations Window.	12
Figure 14. Debug Drop-down.....	13

Figure 15. Debug Perspective.....	13
Figure 16. Add Breakpoint.....	14
Figure 17. Debugging Toolbar.....	14
Figure 18. Change Perspectives.....	14
Figure 19. EmbSys Register View.....	15
Figure 20. Show EmbSys Registers.....	16
Figure 21. View Register and Bit Values.....	16
Figure 22. Add New Header File to Project.....	17
Figure 23. New Header File Window.....	17
Figure 24. Add Source file to Makefile.....	18
Figure 25. New C Project.....	19
Figure 26. New Project Settings.....	20
Figure 27. Project Properties-Build Settings.....	21
Figure 28. Project Properties-Preprocessor Settings.....	22
Figure 29. Debug Icon Drop-Down.....	22
Figure 30. New Debug Configuration.....	23
Figure 31. Debug Configurations- Main Tab.....	23
Figure 32. Debug Configurations - Debugger Tab.....	24
Figure 33. Debug Configurations - Startup Tab.....	25
Figure 34. Debug Configuration - Common Tab.....	26
Figure 35. New Source File Window.....	27
Figure 36. Copy/Paste Makefile.....	27
Figure 37. Makefile Source Files.....	28
Figure 38. Source file with Key on Icon.....	29
Figure 39. Add Include Paths.....	30
Figure 40. Export Include Paths Window.....	31
Figure 41. Import Include Paths Window.....	32

Introduction

This tutorial is a brief guide to getting started with Eclipse CDT IDE using the MAX3263X SDK included in the Maxim ARM Cortex Toolchain. Eclipse is an integrated development environment (IDE) that supports C and C++ language through the C/C++ Development Tooling (CDT) plugin. The MAX3263X software development kit (SDK) includes the MAX3263X library, firmware examples and documentation to help get users up and running on the MAX32630/31 ARM® Cortex® microcontroller. This guide provides steps on how to create, import, build and debug a project in Eclipse using the MAX3263X SDK.

1 Create a Workspace

Open Eclipse by going to the Start Menu and navigating to the folder name specified during installation, e.g., All Programs\Maxim Integrated. After displaying the splash screen, Eclipse asks for a workspace location (**Figure 1**); the default workspace is usually sufficient. It is recommended not to select a workspace in the Maxim Integrated installation directory, -e.g., C:\Maxim. Also, workspace paths with spaces do not function properly in Eclipse.

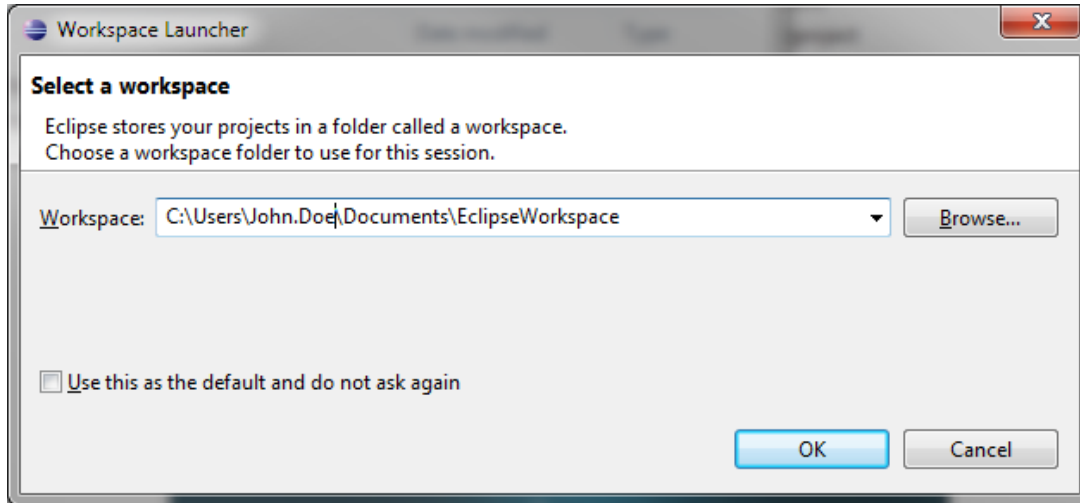


Figure 1. Eclipse Workspace Launcher.

Each time Eclipse opens, you are prompted to select a workspace location. Selecting a location creates a workspace if one does not already exist. Select the **Use this as the default and do not ask again** checkbox to bypass this prompt in the future. The first time a workspace is opened, a welcome window appears. Close this welcome window to see the C/C++ workspace perspective.

2 Import Example Projects

1. In Eclipse, right click in the **Project Explorer** area and select **Import** or go to **File > Import** (Figure 2).

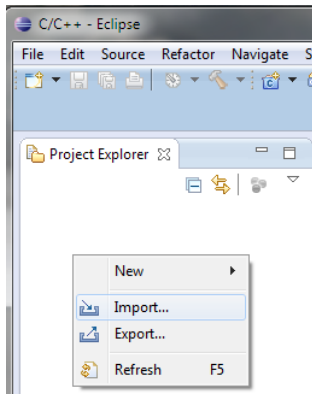


Figure 2. Import Project.

2. On the Import window under **General**, select **Existing Projects into Workspace** and click **Next** (Figure 3).

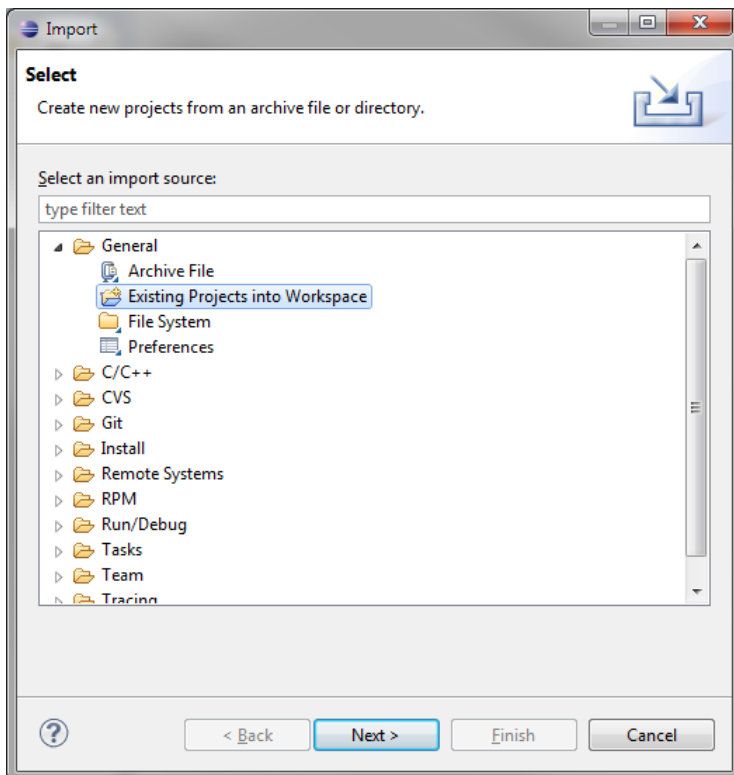


Figure 3. Select Import Source.

3. Click the **Browse** button to select the Maxim directory (C:\Maxim\Firmware\MAX3263X\Applications\EvKitExamples).
4. The projects should show up in the **Projects** list. Check the projects to import and then check **Copy projects into workspace** under Options (**Figure 4**).
5. Click **Finish**.
6. The project should appear in the Project Explorer on the left. Refer to [Build/Debug Project](#) to run the example.

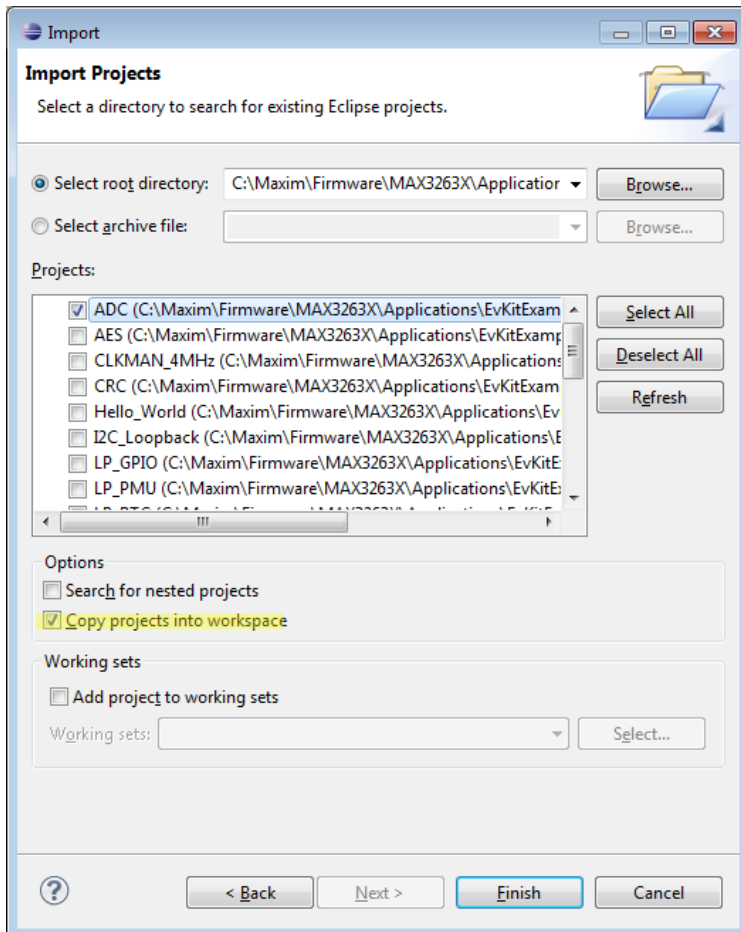


Figure 4. Select Project to Import and Copy.

3 Create New Project from Existing Project

The simplest way to start a new project is to copy an existing project containing all the proper settings. Follow the steps outlined in the sections below.

- 3.1 Copy/Rename Project
- 3.2 Import Project

3.1 Copy/Rename Project

Note: If an example project has already been imported into the workspace, close and restart Eclipse before continuing.

1. To create a new project from an existing project, first copy the existing project into the new workspace directory; one of the EV kit firmware examples provided in the Maxim directory, e.g., C:\Maxim\Firmware\MAX3263X\Applications\EvKitExamples, is copied in this example. Then, rename the project folder to the desired project name (**Figure 5**).

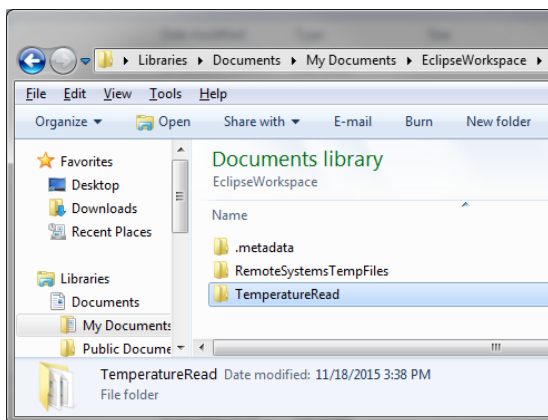


Figure 5. Project Folder Rename.

2. Inside the project folder, delete the build folder and rename the *.launch file to the project name, e.g., TemperatureRead.launch (**Figure 6**).

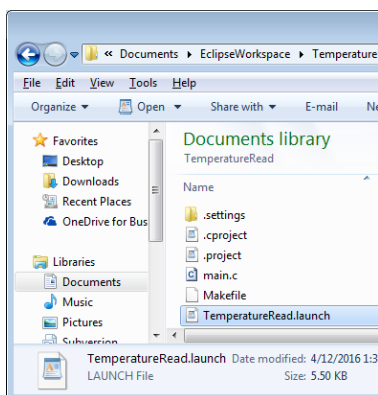


Figure 6. Project Files.

3. Open .cproject, .project and *.launch files in a text editor, and replace the old project name with the new project name, e.g., Replace SPIM_MX25 with TemperatureRead.

3.2 Import Project

1. In Eclipse, right click on **Project Explorer** area, and select **Import** or go to **File > Import** (Figure 7).

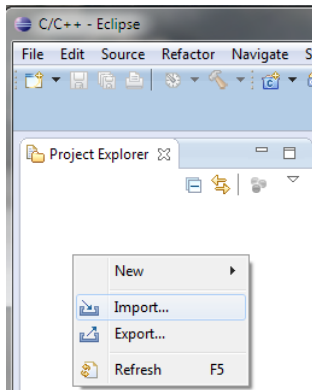


Figure 7. Import Project.

2. On the Import window, under **General** select **Existing Projects into Workspace** and click **Next** (Figure 8).

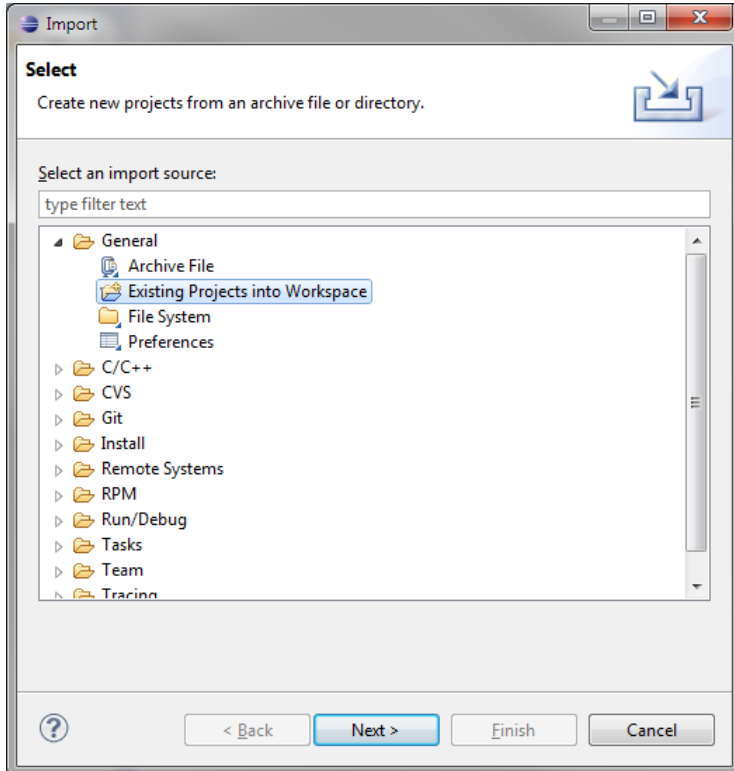


Figure 8. Select Import Source.

3. If the new project is in the workspace directory, uncheck **Copy projects into workspace**. If the new project is outside the workspace directory, check the **Copy projects into workspace**.
4. Select the project to import using the **Browse** button to select the project directory. *Note:* If the project files were renamed correctly, the project should appear in the Projects box with a checkmark. If you get an error saying the project already exists, uncheck **Copy projects into workspace**, then click **Refresh** (Figure 9).
5. Click **Finish**.

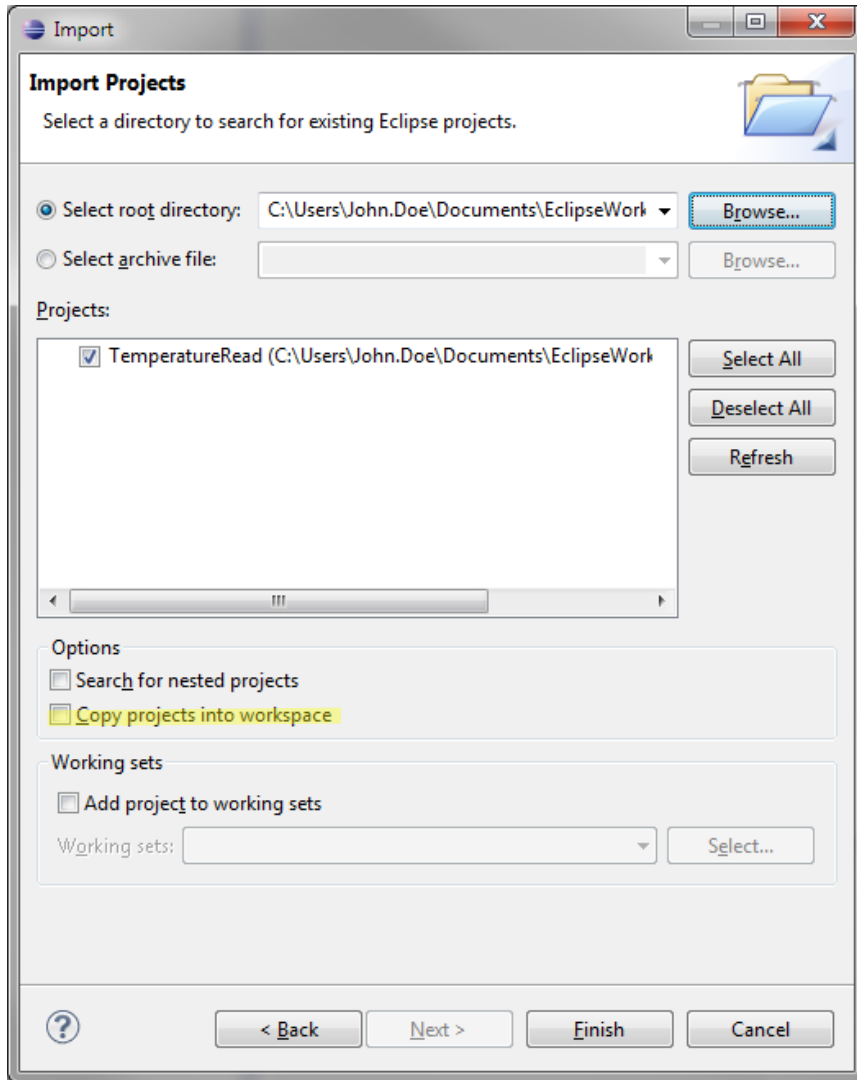


Figure 9. Select Project to Import.

4 Build/Debug Project

To build and debug a project in Eclipse, follow the steps outlined in the sections below.

- 4.1 Build Project
- 4.2 Debug Project

4.1 Build Project

Before a build, Eclipse can show errors of unresolved include paths and symbols. The project setting have been set to find the include paths and symbols based on the Makefile output after a build. Before the first build, right click on the project in the **Project Explorer** and select **Clean Project**. After the clean completes, right-click on the project again and select **Build Project** or click the **Build** (hammer) icon. Projects can have multiple build configurations shown in the Build icon drop-down (**Figure 10**).

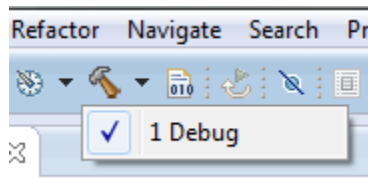


Figure 10. Build Icon.

The Console window shows the progress of the build and the results. If the project builds successfully the Console says **Build Finished** without any errors (**Figure 11**).

If there are source files in the project still showing errors of symbols that cannot be resolved, go to Project > C/C++ Index > Freshen Files or Project > C/C++ Index > Rebuild. If neither of these work, go to [Troubleshooting Unresolved Symbols](#) section.

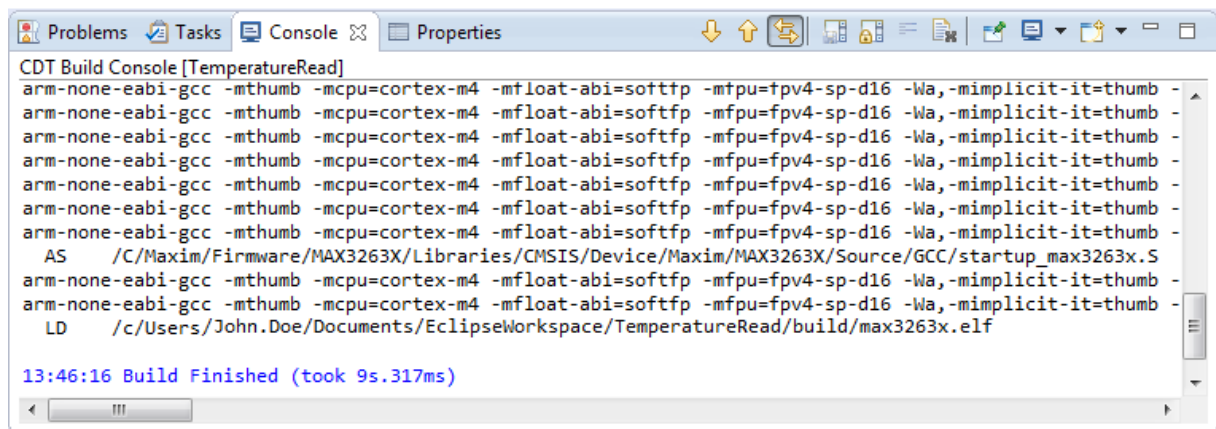


Figure 11. Console Window.

4.2 Debug Project

There is a debug configuration for each sample project provided. To debug, click the drop down on the bug icon and select **Debug Configurations...** (Figure 12).

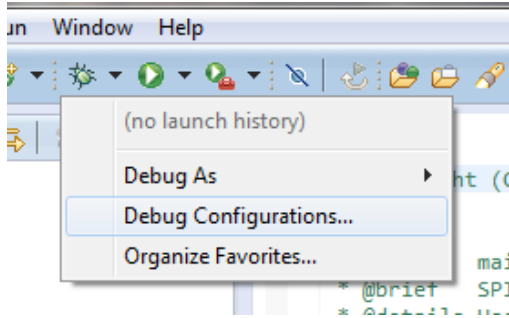


Figure 12. Debug Drop-down.

From here, select the project name under the **GDB OpenOCD Debugging** category, e.g., **TemperatureRead**, and click **Debug** (Figure 13).

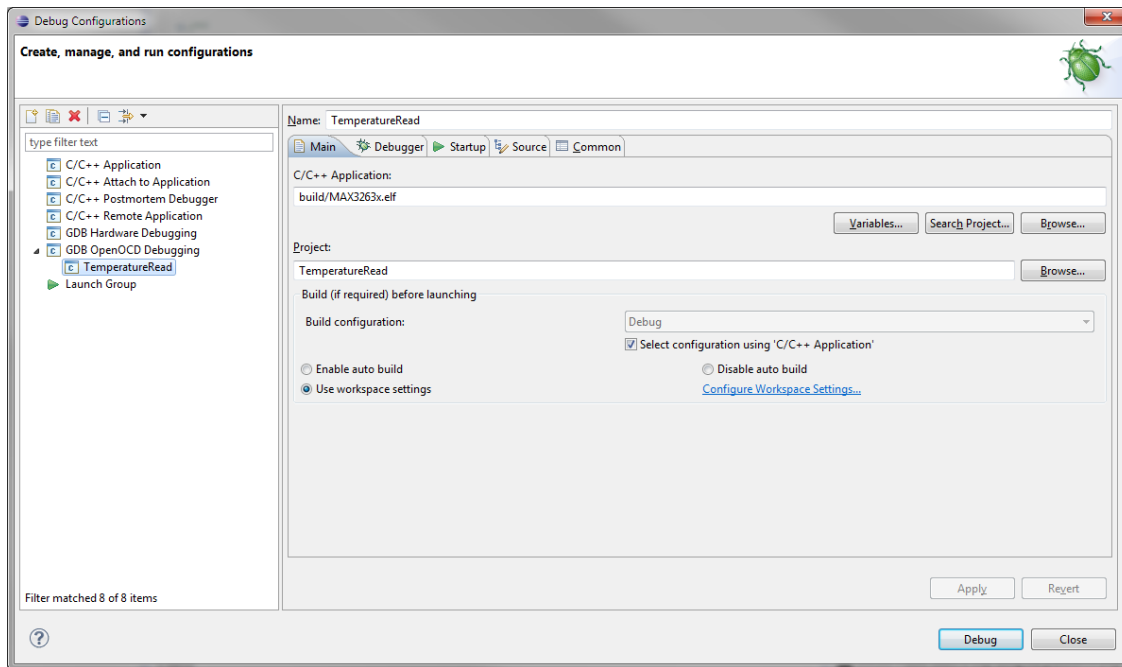


Figure 13. Debug Configurations Window.

After a debug configuration is launched, it is added to the debug configuration launch history in the Debug (bug) icon drop-down (Figure 14).

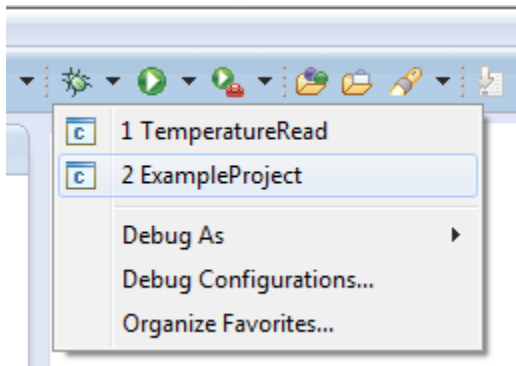


Figure 14. Debug Drop-down.

Eclipse opens the Debug perspective automatically and attempts to build, load, and run the program. By default, a breakpoint is set at the first line of main (Figure 15).

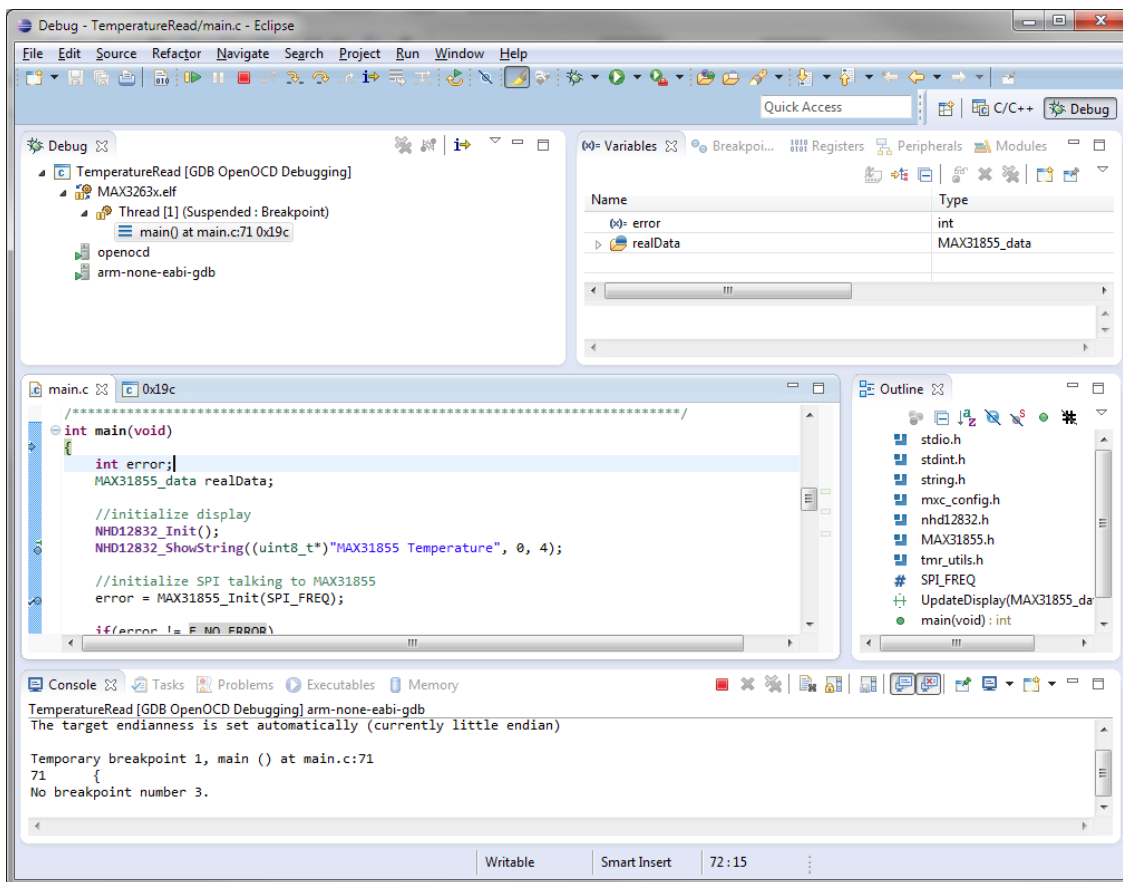
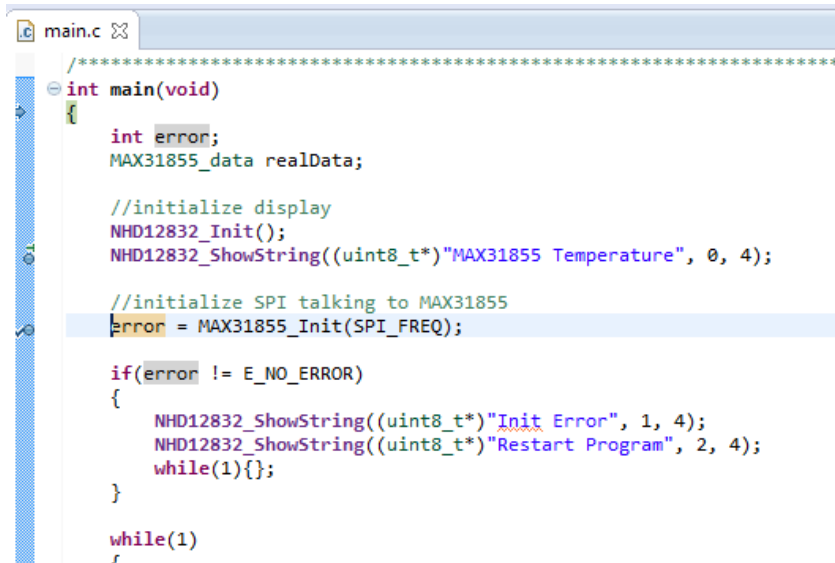


Figure 15. Debug Perspective.

Breakpoints can be set by double-clicking next to the desired line on the blue vertical bar (Figure 16).



```
main.c
/*****
int main(void)
{
    int error;
    MAX31855_data realData;

    //initialize display
    NHD12832_Init();
    NHD12832_ShowString((uint8_t*)"MAX31855 Temperature", 0, 4);

    //initialize SPI talking to MAX31855
    error = MAX31855_Init(SPI_FREQ);

    if(error != E_NO_ERROR)
    {
        NHD12832_ShowString((uint8_t*)"Init Error", 1, 4);
        NHD12832_ShowString((uint8_t*)"Restart Program", 2, 4);
        while(1){};
    }

    while(1)
    {
```

Figure 16. Add Breakpoint.

There is a debugging toolbar shown in the **Debug** perspective that can be used to step through code (Figure 17). The icons in this toolbar are (from left to right): Resume (F8), Suspend (disabled), Terminate (Ctrl+F2), Disconnect (disabled), Step Into (F5), Step Over (F6), and Step Return (F7-disabled).



Figure 17. Debugging Toolbar.

The user can also manually switch between the **Debug** and **C/C++** perspectives (Figure 18).



Figure 18. Change Perspectives.

5 Register View

Included in the Eclipse distribution is the Embedded Systems Register View (EmbSysRegView) plugin that allows the user to view the Cortex Microcontroller Software Interface Standard System View Description (CMSIS-SVD) file for the product being used (e.g., MAX32630). This file provides an itemized listing of the product's registers. To configure the plugin in Eclipse, select **Window > Preferences**. Under **C/C++ > Debug** select **EmbSys Register View**. In the drop-downs select the following: **SVD(CMSIS)** for Architecture, **MaximIntegrated** for the Vendor, and **MAX32630** for the Chip. Click OK (Figure 19).

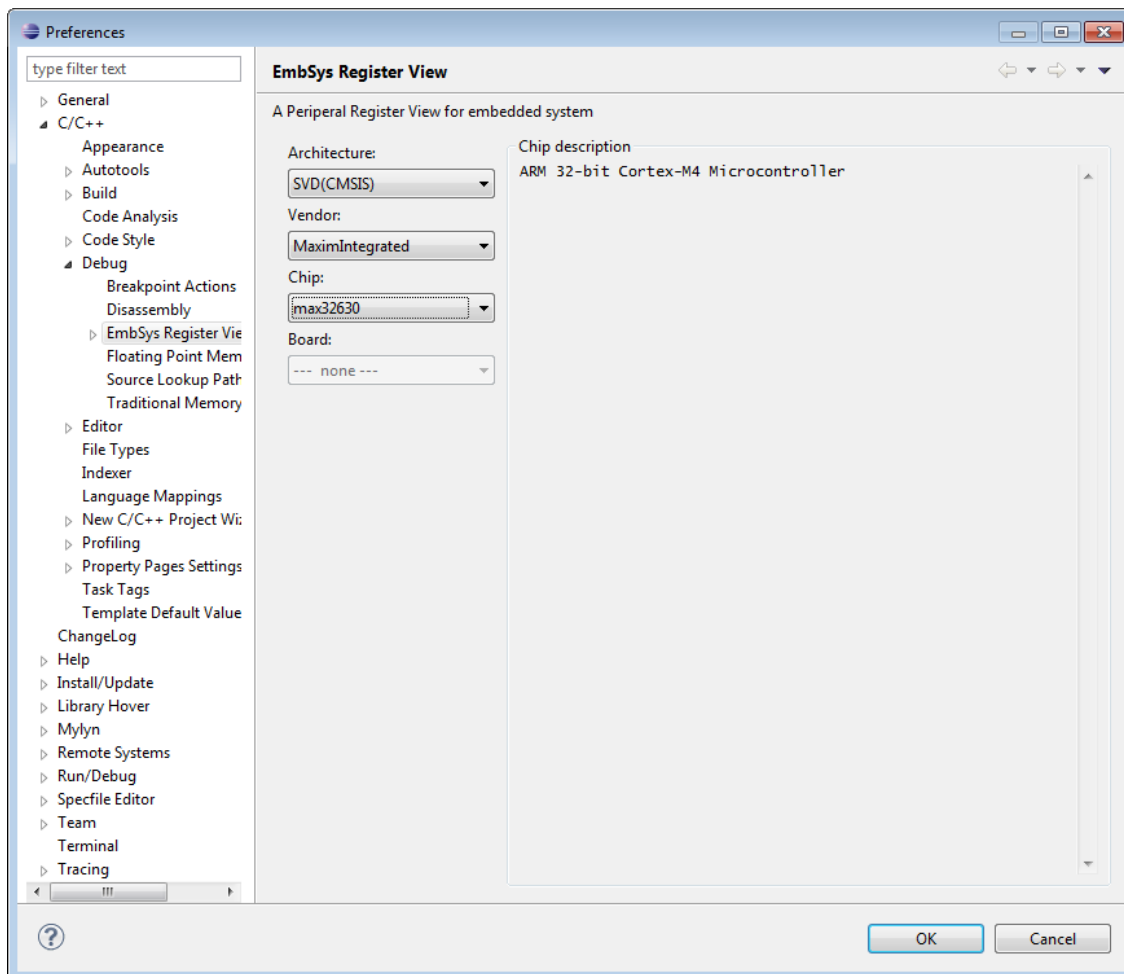


Figure 19. EmbSys Register View.

6 Add Header/Source Files to Project

To add more header/source files to a project, right click on the project and select **New > Header File** or **New > Source File** (Figure 22).

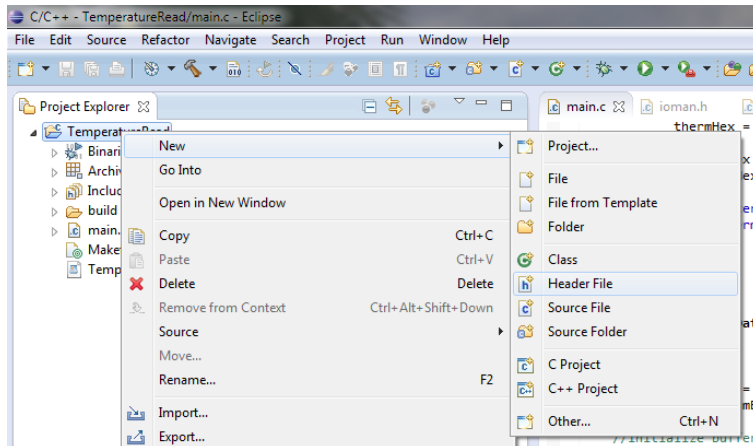


Figure 22. Add New Header File to Project.

In the New Header (or Source) File window, keep the project folder as the Source folder and name the file with *.h for header, or *.c for source file. Keep the **Template** as default (Figure 23).

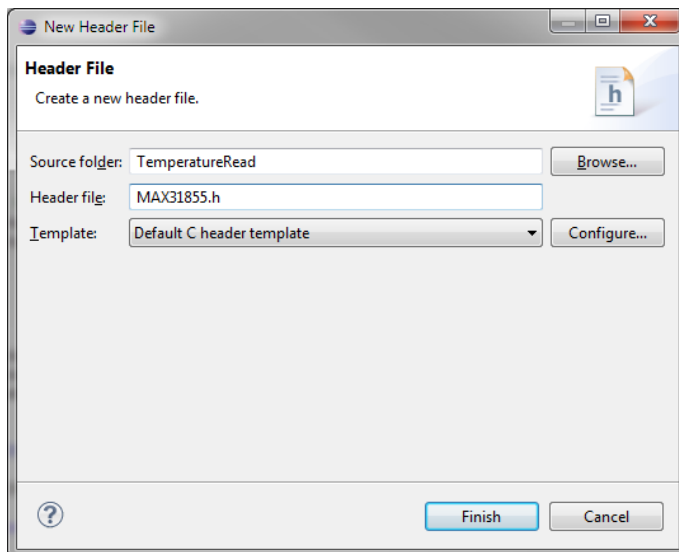


Figure 23. New Header File Window.

To build the new source file with the project, the new source file should be added to the **Makefile**. Open the **Makefile**, and add the new source file to SRCS (**Figure 24**).

```
# Source files for this test (add path to VPATH below)
SRCS = main.c
SRCS += MAX31855.c
```

Figure 24. Add Source file to Makefile.

7 Create New Project from Scratch

To create a new blank project in Eclipse, follow the steps outlined in the sections below.

- 7.1 Create New ProjectError! Reference source not found.
- 7.2 Setup Debug Configuration
- 7.3 Add Files to Project

7.1 Create New Project

1. Select File > New > C Project.
2. Type in a project name and select **Executable > Empty Project** in the **Project type**, and **Cross ARM GCC** in the **Toolchains**. Then click **Next** (Figure 25).

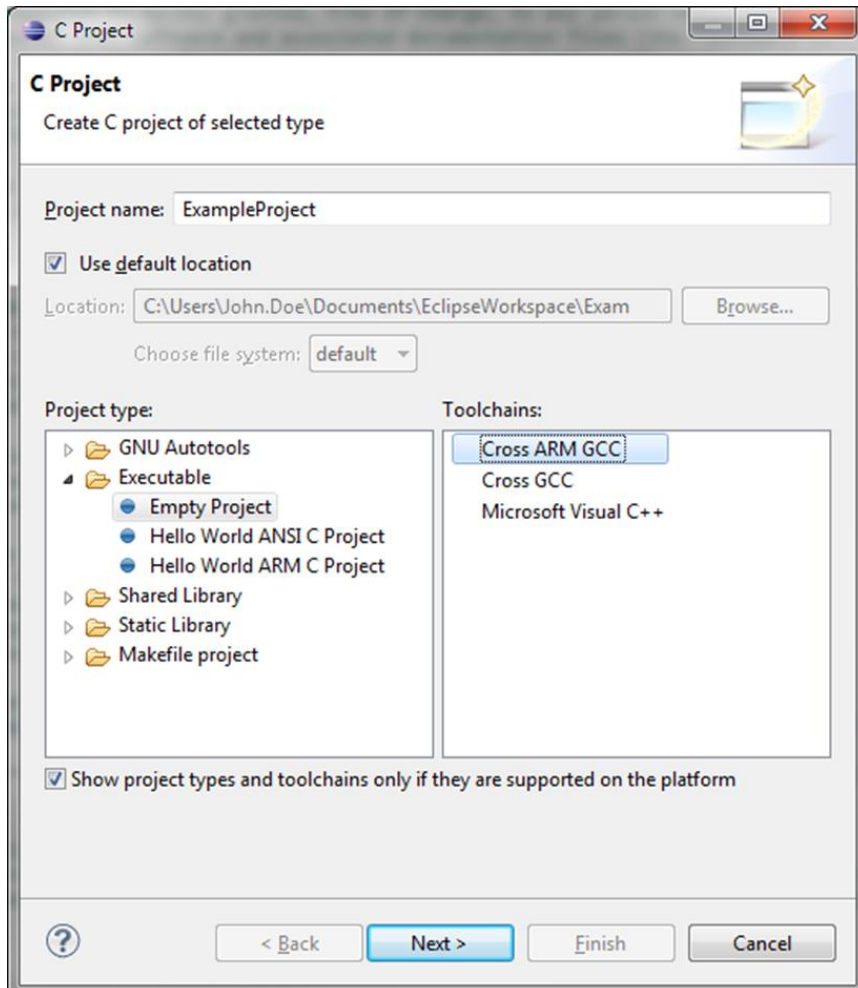


Figure 25. New C Project.

3. On the next window uncheck **Release** under Configurations, and open the **Advanced settings...** (Figure 26).

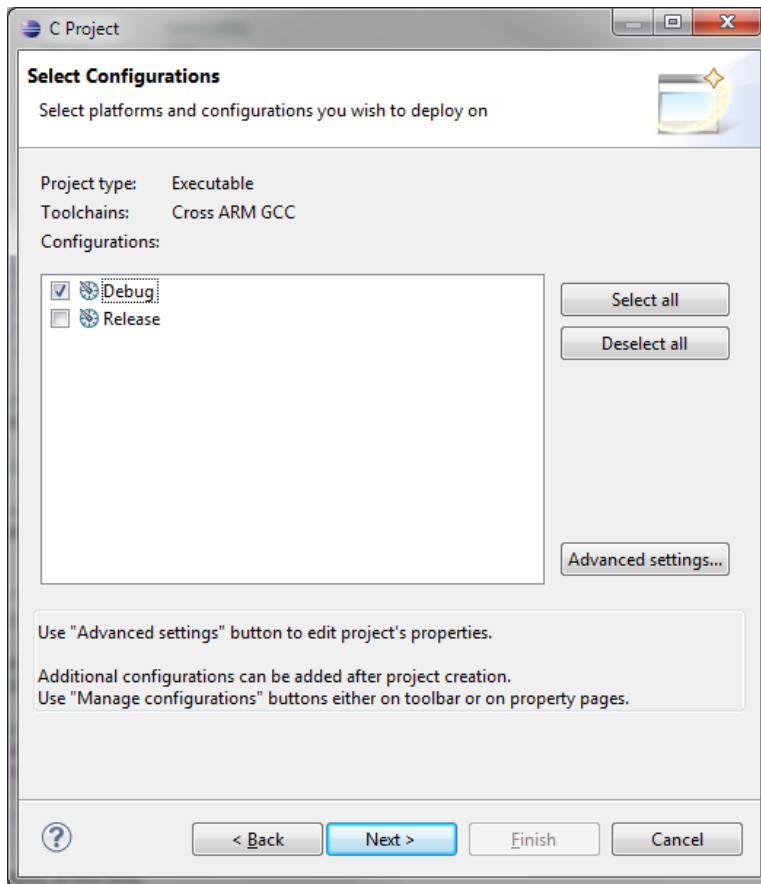


Figure 26. New Project Settings.

- Under **C/C++ Build** on the **Builder Settings** tab, deselect **Use default build command**, and enter “make ECLIPSE=1” into the Build command field (Figure 27).

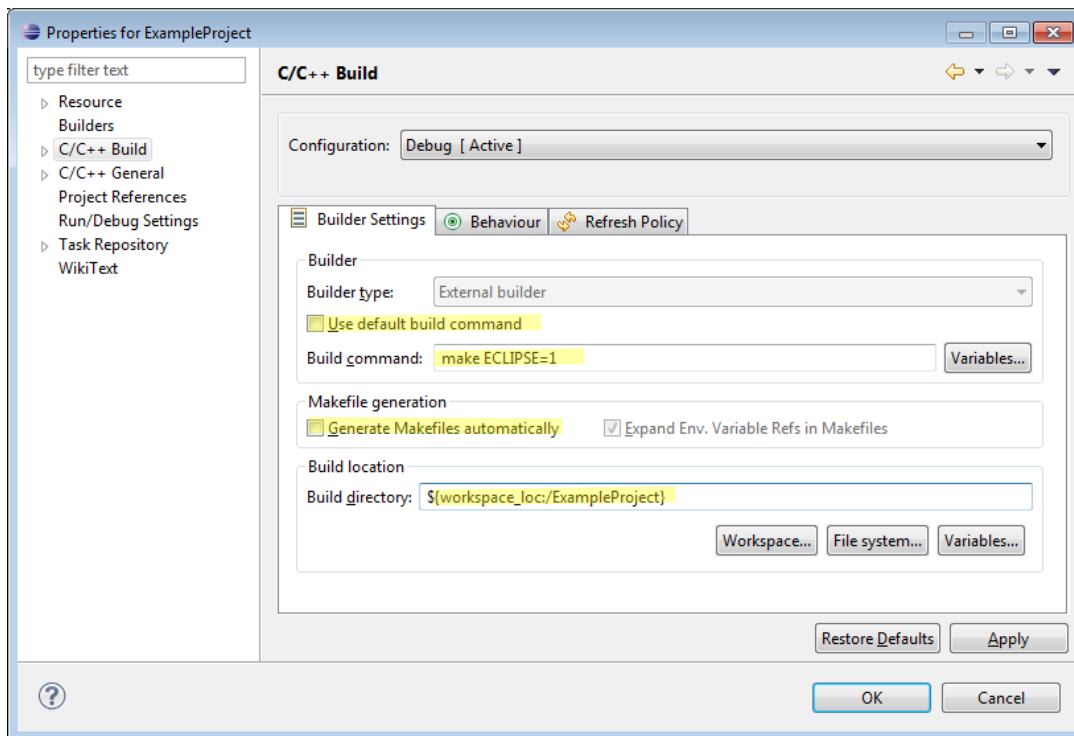


Figure 27. Project Properties-Build Settings.

- Uncheck **Generate Makefiles automatically** and remove “/Debug” from the Build directory, e.g., `${workspace_loc:/ExampleProject}`.
- Go to **C/C++ General > Preprocessor Include Paths, Macros etc.** and click on the **Providers tab**. In the list, check **CDT GCC Build Output Parser** and change the Compiler command pattern to “(.gcc)” (Figure 28).
- Use the **Move Up** button to move **CDT GCC Build Output Parser** to be second in the list and **CDT GCC Built-In Compiler Settings Cross ARM** to be third in the list (Figure 28).

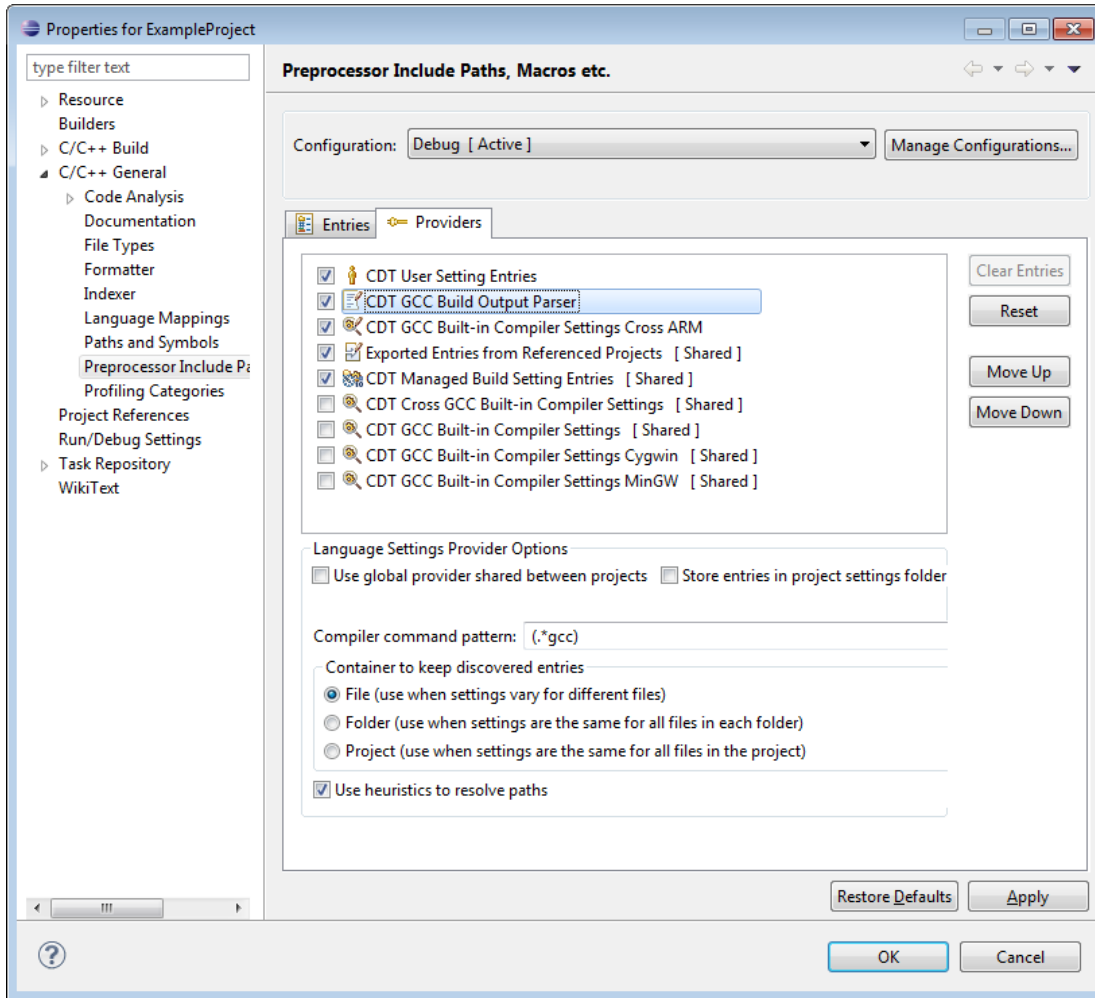


Figure 28. Project Properties-Preprocessor Settings.

8. Select OK, Next >, and then Finish.

7.2 Setup Debug Configuration

1. To add a debug configuration for the new project, select Debug Configurations... from the Debug icon drop-down (Figure 29).

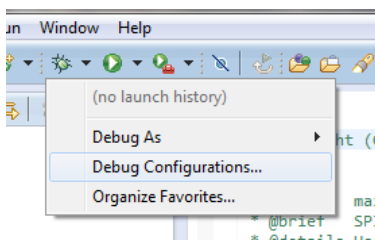


Figure 29. Debug Icon Drop-Down.

2. Right click on GDB OpenOCD Debugging and select New (Figure 30).

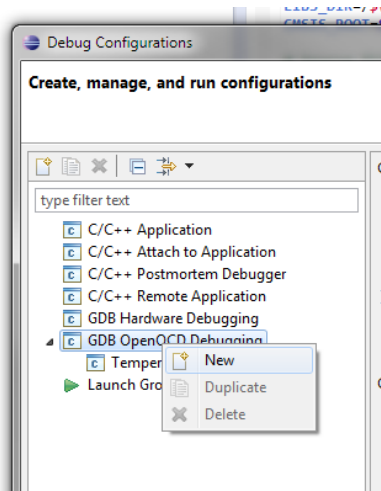


Figure 30. New Debug Configuration.

3. It is recommended to change the name of the debug configuration to match the project name. Under the **Main** tab, type "build/MAX3263X.elf" in the C/C++ Application field (Figure 31).

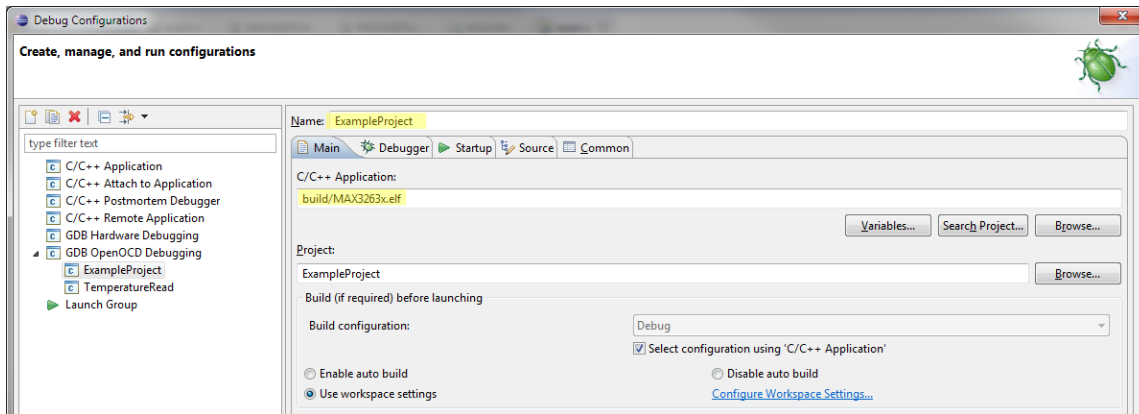


Figure 31. Debug Configurations- Main Tab.

4. Under the **Debugger** tab, change the following (Figure 32):
 - **OpenOCD Setup Executable:** openocd
 - **OpenOCD Setup Config Options:** -s `${env_var:TOOLCHAIN_PATH}/share/openocd/scripts -f interface/ftdi/olimex-arm-usb-tiny-h.cfg -f target/MAX3263X.cfg`
 - **GDB Client Setup Executable:** arm-none-eabi-gdb
 - Delete the text in **GDB Client Setup Commands**

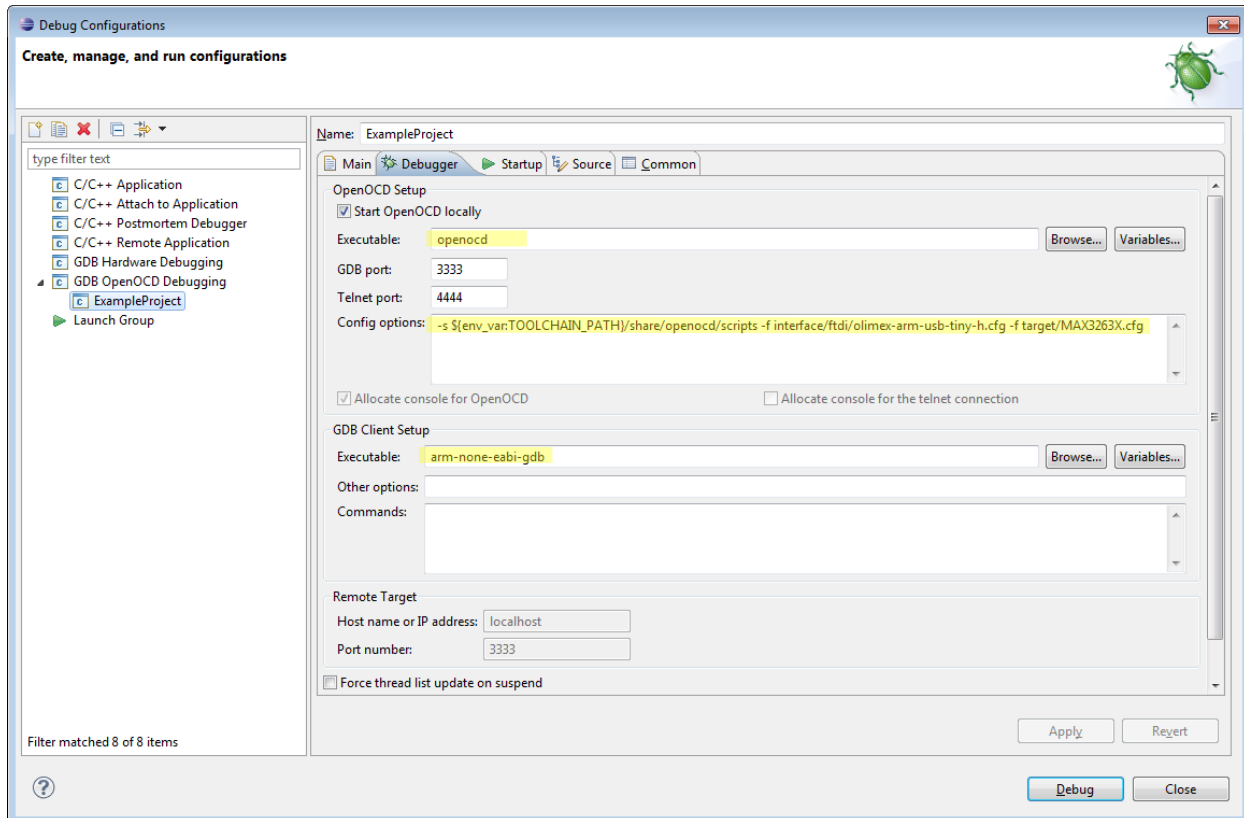


Figure 32. Debug Configurations - Debugger Tab.

5. Go to the **Startup** tab and deselect **Enabled ARM semihosting** (Figure 33).

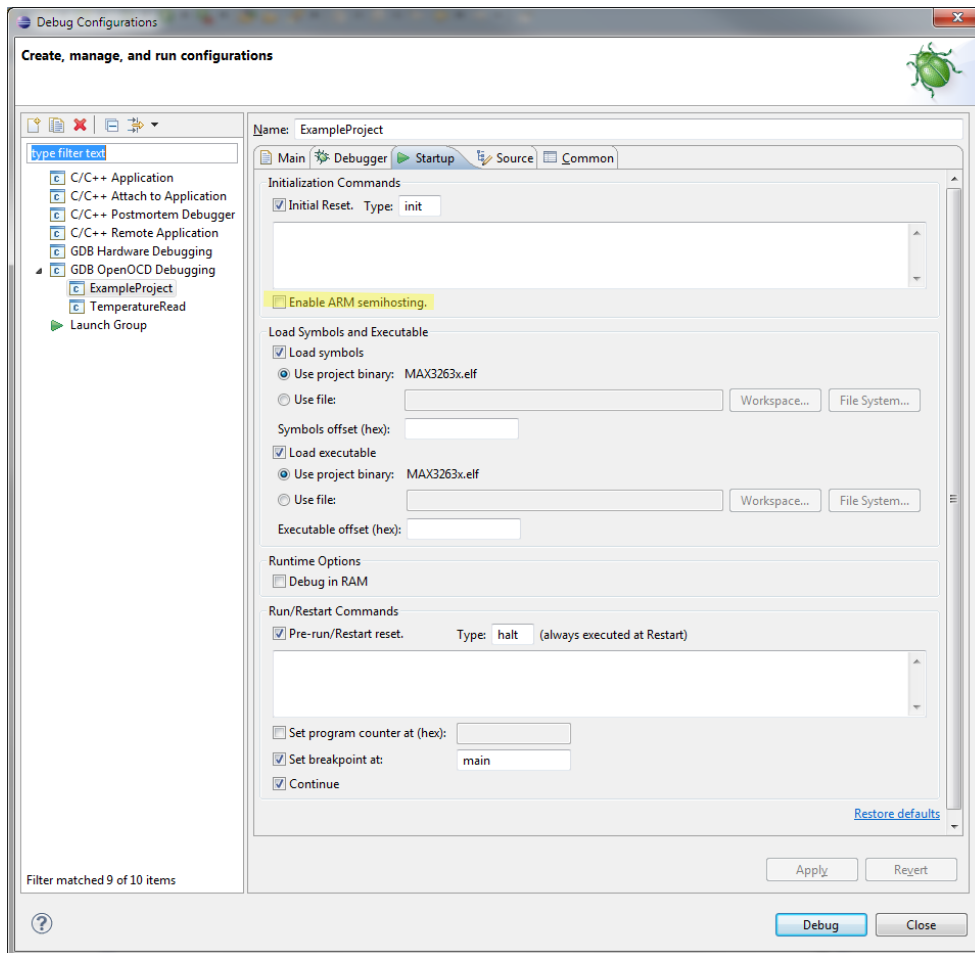


Figure 33. Debug Configurations - Startup Tab.

- Under the **Common** tab, select the **Shared file** radio button and check the **Debug** checkbox under **Display in the favorites menu** (Figure 34).

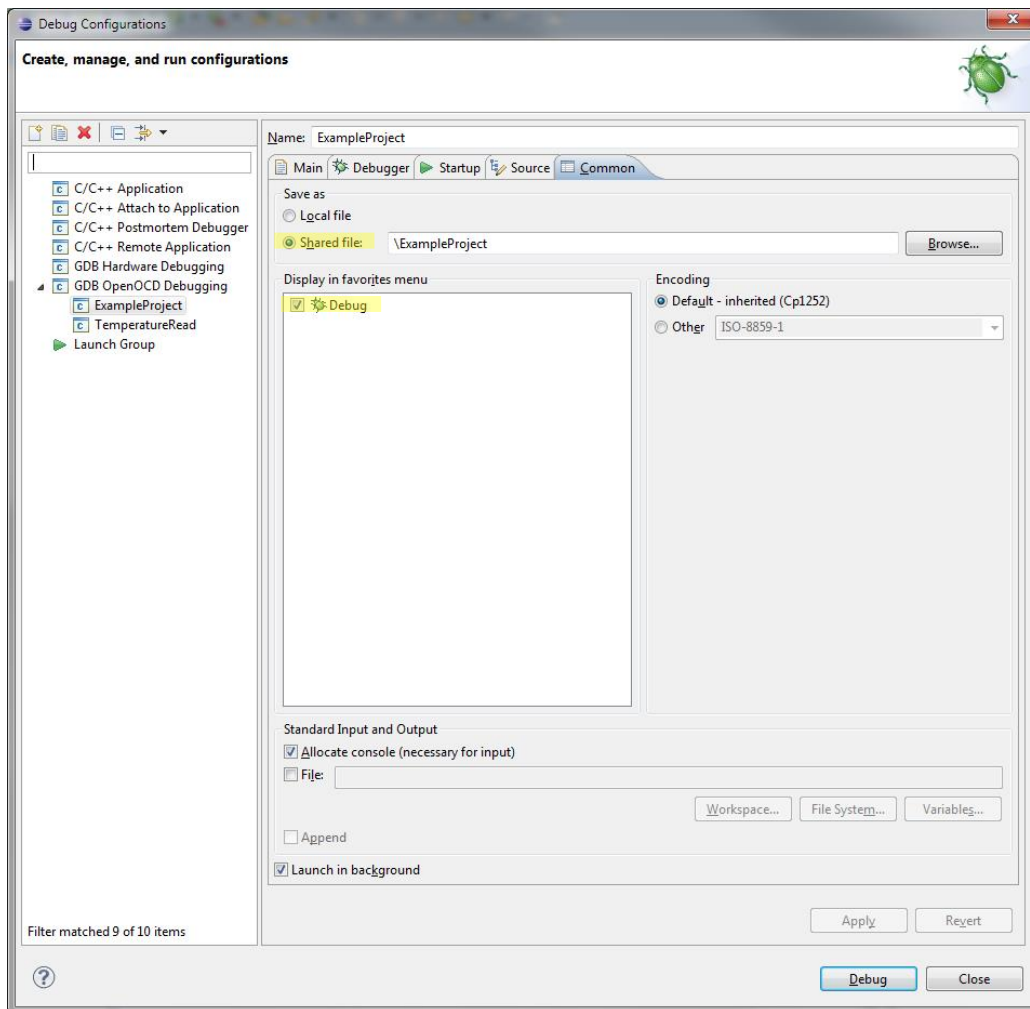


Figure 34. Debug Configuration - Common Tab.

- Click **Apply**, then **Close**. The project should now very closely match the provided example projects.

7.3 Add Files to Project

- Add a **main.c** source file to the project by right-clicking on the project, and go to **New > Source file**.

2. Ensure the correct Project folder is selected and name the file **main.c** (Figure 35).
Note: If you choose a different name, the **Makefile** must also be changed.

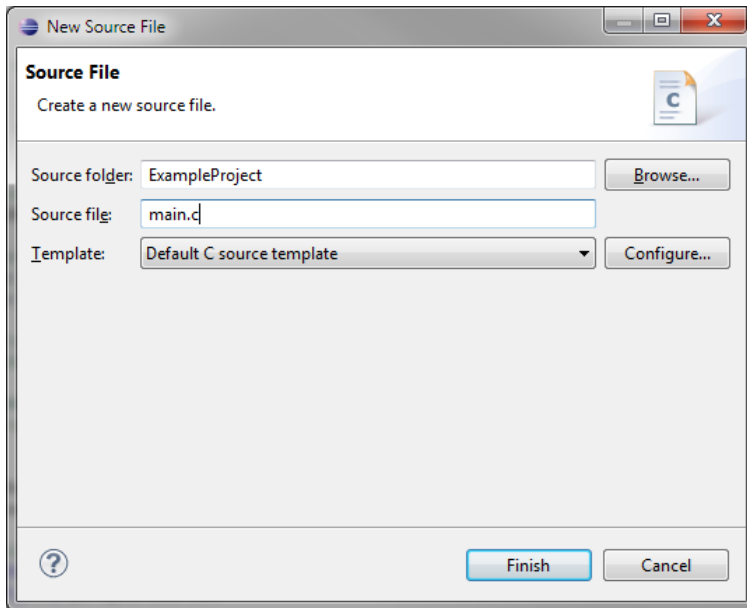


Figure 35. New Source File Window.

3. Leave the **Template** as **Default C source template** and click **Finish**. The new file should appear in the **Project Explorer**.
4. The easiest way to create a **Makefile** is to copy the one used in the EV kit examples. Go to the Maxim directory, e.g., C:\Maxim\Firmware\MAX3263X\Applications\EvKitExamples, and locate the **Makefile** used in one of the EV kit examples. If using USB or BTLE copy the Makefile from the USB or BTLE EV kit example.
5. Copy and paste the **Makefile** into the project directory (Figure 36).

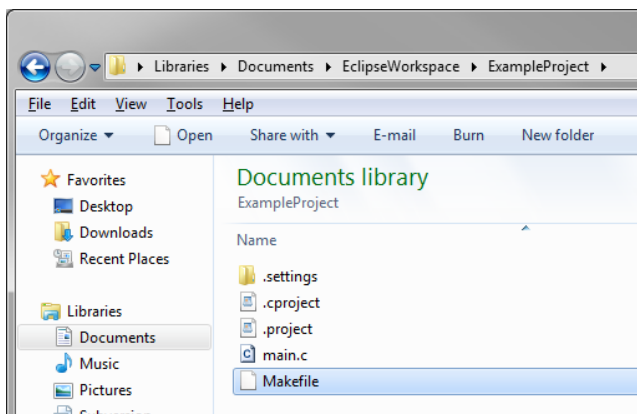


Figure 36. Copy/Paste Makefile.

6. To see the **Makefile** in the **Project Explorer** in Eclipse, right click on the project and select **Refresh**.
7. Open the **Makefile** and ensure the source file name matches the name picked above in step 2. If there are additional files listed as source files that are not in the project, remove them (**Figure 37**).

```
# Source files for this test (add path to VPATH below)
SRCS = main.c
```

Figure 37. Makefile Source Files.

8 Troubleshooting Unresolved Symbols

The EV kit examples are setup to discover the Maxim library include paths from the output of the Makefile in the project. After a successful build the source files should not have any unresolved include paths or symbols. If the source files are still showing unresolved errors, verify the project is setup to discover the include paths by checking if the source files have a key symbol on the icon in the project explorer (Figure 38).

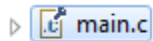


Figure 38. Source file with Key on Icon

If the source file(s) do have a key symbol try the following:

- Right click on the project and select **Clean Project**, then start a build.
- Go to **Project > C/C++ Index > Freshen Files** or **Project > C/C++ Index > Rebuild**
- Try starting a new workspace and import the project in with **Copy Project into Workspace** checked (Figure 4).
- Try adding the include file paths into the project properties. See [Add Include Paths to Project](#)

If the source file(s) do not have the key symbol try the following:

- Check if the project properties are setup to discover the include paths. Right click on the project and go to **Properties**. Go to **C/C++ General > Preprocessor Include Paths, Macros etc.** and on the **Entries** tab verify **CDT GCC Build Output Parser** is second in the list. If it is not follow steps 6-8 in [Create New Project](#) section. Then perform a clean and build on the project.
- Try adding the include file paths into the project properties. See [Add Include Paths to Project](#)

8.1 Add Include Paths to Project

The Maxim library include paths can be added to the project properties without having to discover them from a build. This can be done manually by adding all eleven include paths, or by export/importing an .xml file with all the paths.

8.1.1 Manually Add Include Paths

1. Right click on the project in the **Project Explorer** and select **Properties**.
2. Go to **C/C++ General > Paths and Symbols** and on the **Includes** tab select **GNU C** under languages (Figure 39).

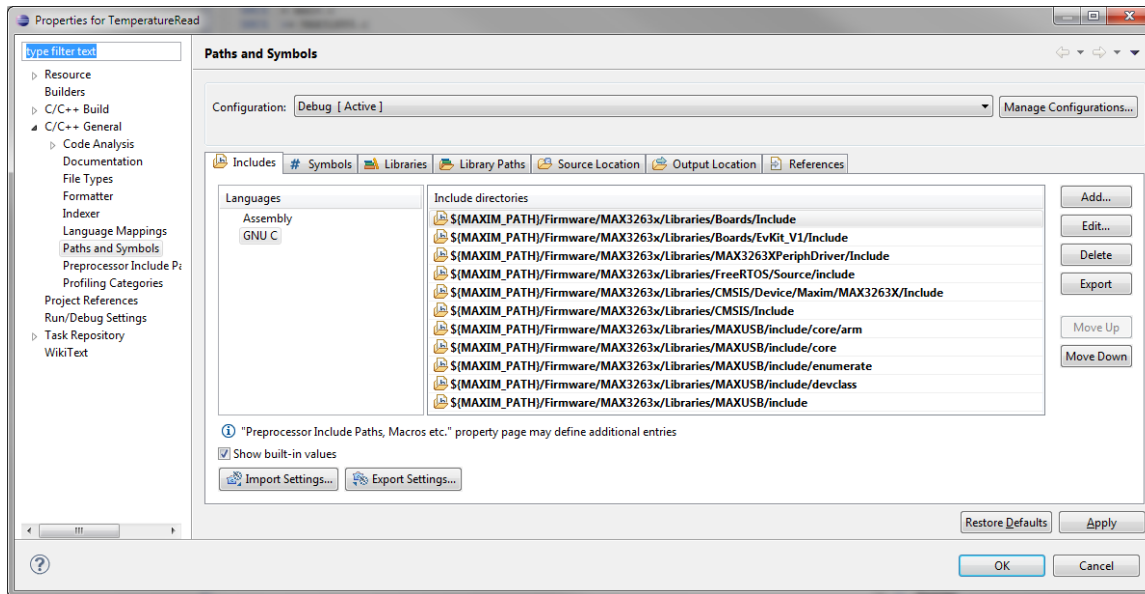


Figure 39. Add Include Paths.

3. Add all the library include paths here by clicking **Add**. See Include Paths for list of paths. *Note:* These settings can also be imported by clicking **Import Settings...** and selecting a previously saved .xml settings file. See [Export/Import Include Paths](#) for details.
4. Once all the include paths are added, click **OK** to save the changes and close the window. *Note:* If there are source files in the project still showing errors of symbols that cannot be resolved, go to Project > C/C++ Index > Freshen Files.

Include Paths:

```

${MAXIM_PATH}/Firmware/MAX3263X/Libraries/Boards/Include
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/Boards/EvKit_V1/Include
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/MAX3263XPeriphDriver/Include
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/FreeRTOS/Source/include
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/CMSIS/Device/Maxim/MAX3263X/Include
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/CMSIS/Include
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/MAXUSB/include/core/arm
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/MAXUSB/include/core
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/MAXUSB/include/enumerate
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/MAXUSB/include/devclass
${MAXIM_PATH}/Firmware/MAX3263X/Libraries/MAXUSB/include

```

8.1.2 Export/Import Include Paths

1. Import an example project with the include paths settings already added. See [Import Example Projects](#) for more details.

2. Right click on the example project in the **Project Explorer** and go to **Properties**.
3. Go to **C/C++ General > Paths and Symbols** and on the **Includes** tab click **Export Settings...**
4. Select the project to export and check **Include Paths** (Figure 40).

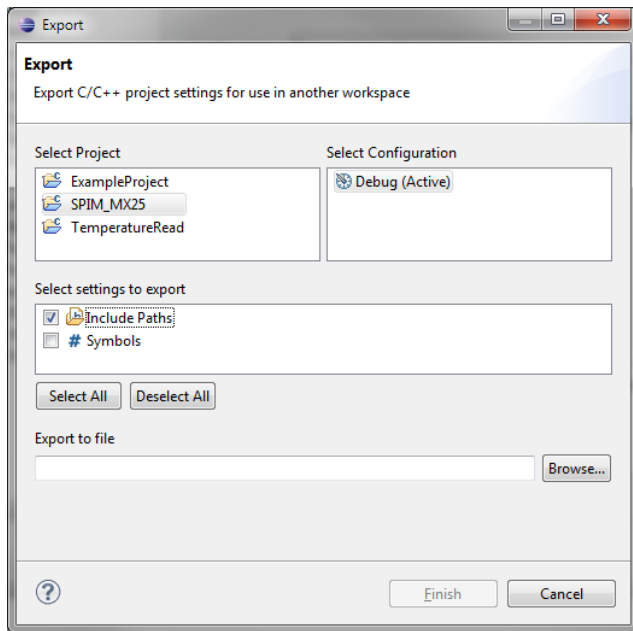


Figure 40. Export Include Paths Window.

5. Click **Browse** to select the file location and name, then click **Finish**.
6. Click **OK** to close the project properties.
7. On the new project, right click and go to **Properties**.
8. Go to **C/C++ General > Paths and Symbols** and on the **Includes** tab click **Import Settings...**

9. Navigate to the .xml files created by the export and select the new project and check **Include Paths** (Figure 41).

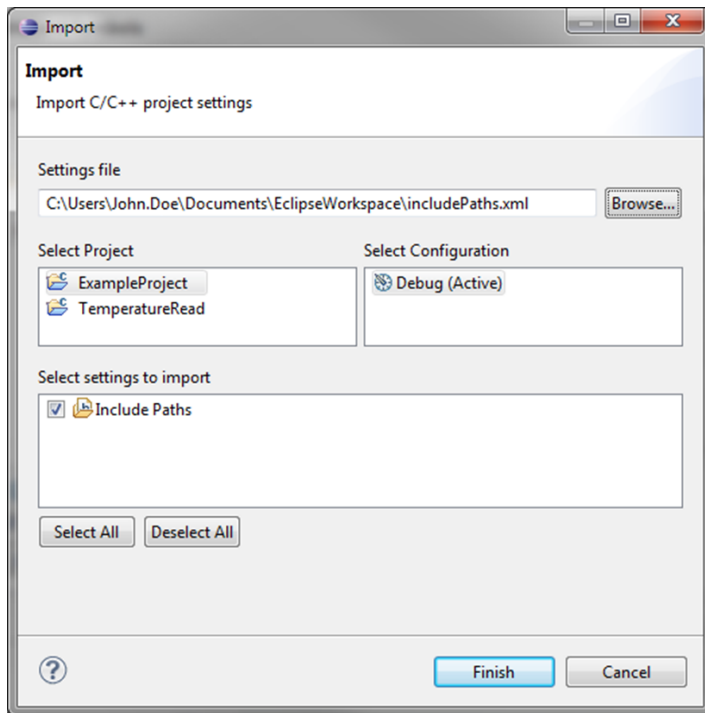


Figure 41. Import Include Paths Window.

10. Click **Finish**. Verify the **GNU C** include paths are the same as Include Paths. *Note:* If there are source files in the project still showing errors of symbols that cannot be resolved, go to Project > C/C++ Index > Freshen All Files.

9 Trademarks

ARM is a registered trademark and registered service mark of ARM Limited.
Cortex is a registered trademark of ARM Limited.

©2016 by Maxim Integrated Products, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. MAXIM INTEGRATED PRODUCTS, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. MAXIM ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering or registered trademarks of Maxim Integrated Products, Inc. All other product or service names are the property of their respective owners.