Getting Started with Eclipse

UG6245; Rev 1, 4/19

Abstract

This tutorial is a brief guide to getting started with the 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 Low Power Arm Micro Toolchain.

Table of Contents

List of Figures

List of Tables

Introduction

This document is a brief guide to getting started with Eclipse. Eclipse® is an integrated development environment (IDE) that supports the C and C++ languages through the C/C++ Development Tooling (CDT) plugin.

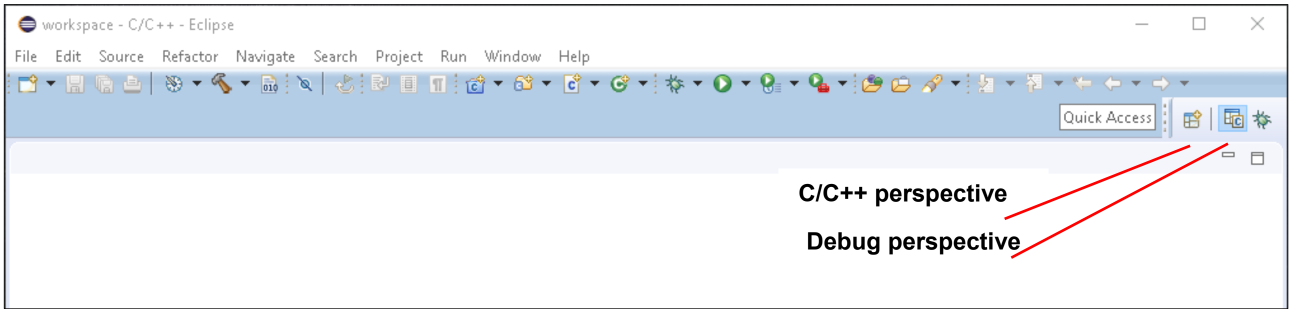
This document provides steps about using Eclipse to import, build, and debug example projects. It also provides instructions for creating new projects based on existing example projects and projects created entirely from scratch.

Changing the Default Workspace Location

By default, Eclipse creates a workspace to store your project files in (C:/User/<*user_name*>/workspace) during installation.

To change the default workspace location, complete the following steps:

1. Navigate to the **Eclipse installation** folder at C:\Maxim\Eclipse.
2. Double-click eclipse.bat. The Eclipse home page appears in the C/C++ perspective by default. **Note**: Only the top portion of the Eclipse home page with the menu and icon bars is shown in **Figure 1**.

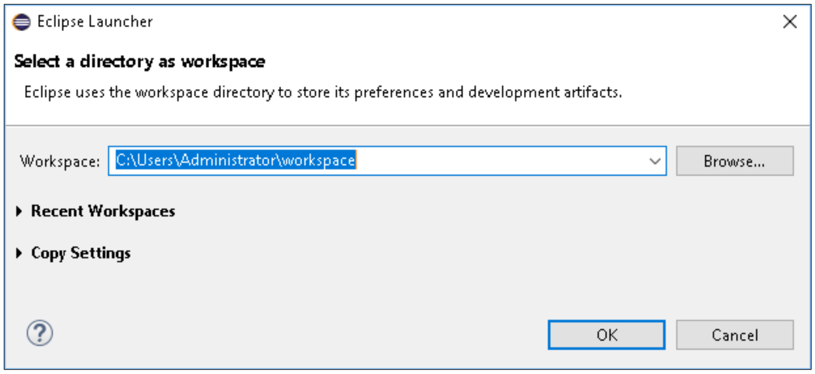


C/C++ perspective

Debug perspective

Figure 1. Eclipse home page.

3. Select **File** > **Switch Workspace** > **Other** from the menu bar. The **Eclipse Launcher** dialog box appears, as shown in **Figure 2**.



Select a directory as workspace

Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\Administrator\workspace

Figure 2. Eclipse Launcher dialog box.

4. Specify the directory path to your workspace location in the **Workspace** field. **Note**: Workspace directory locations with spaces do not function properly in Eclipse.
5. Click **OK**.

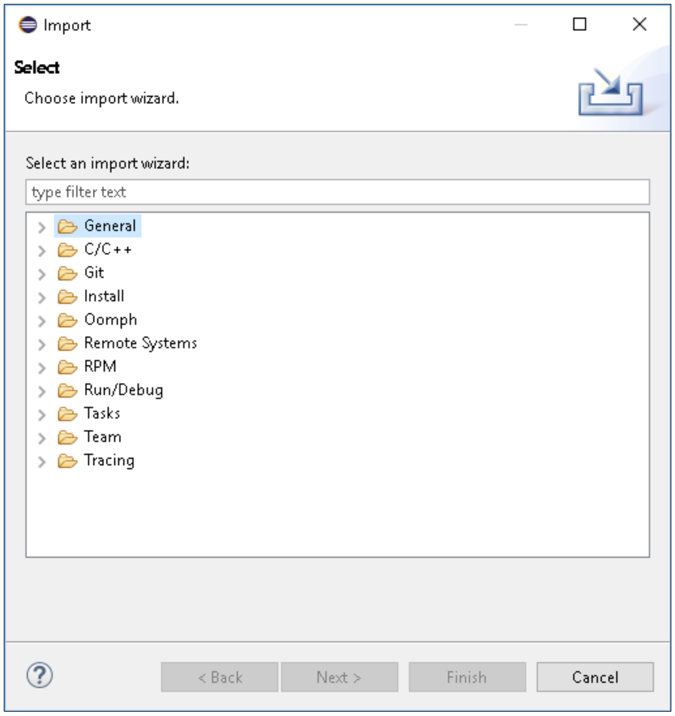
Eclipse is a registered trademark of Eclipse Foundation, Inc.

Importing Device Examples

The Maxim Arm® Cortex® Toolchain contains many examples demonstrating the features of the supported devices. Each device has its own set of examples and libraries that are separated into directories based on part numbers. These device directories are located in C:\Maxim\Firmware\<*device*>\Applications\EvKitExamples.

To begin working with the examples for your chosen device, complete the following steps:

1. Select **File** > **Import** from the menu bar. The **Import** dialog box appears, as shown in **Figure 3**.



Import

Select

Choose import wizard.

Select an import wizard:

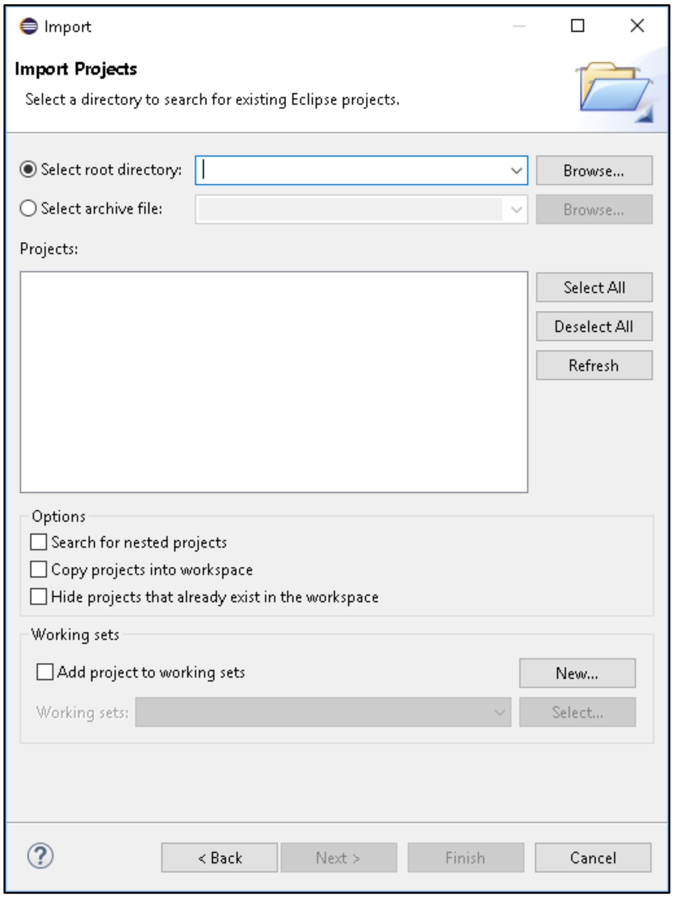
type filter text

Figure 3. Import dialog box.

2. Expand the **General** folder and select **Existing Projects into Workspace**.

Arm and Cortex are registered trademarks of Arm Limited.

3. Click **Next**. The **Import Projects** dialog box appears, as shown in **Figure 4**.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

Working sets:

Figure 4. Import Projects dialog box.

4. Click **Browse**. The **Browse For Folder** dialog box appears.
5. Scroll the list and select the root directory containing the device examples you want to import.
6. Click **OK**. The **Import Projects** dialog box reappears showing the root directory you selected in the **Select root directory** field and a list of device examples in the **Projects** field.
7. Select the check box to the left of each device example you want to import.
8. Select the **Copy projects into workspace** check box.
9. Click **Finish** to import the device examples into your workspace folder.

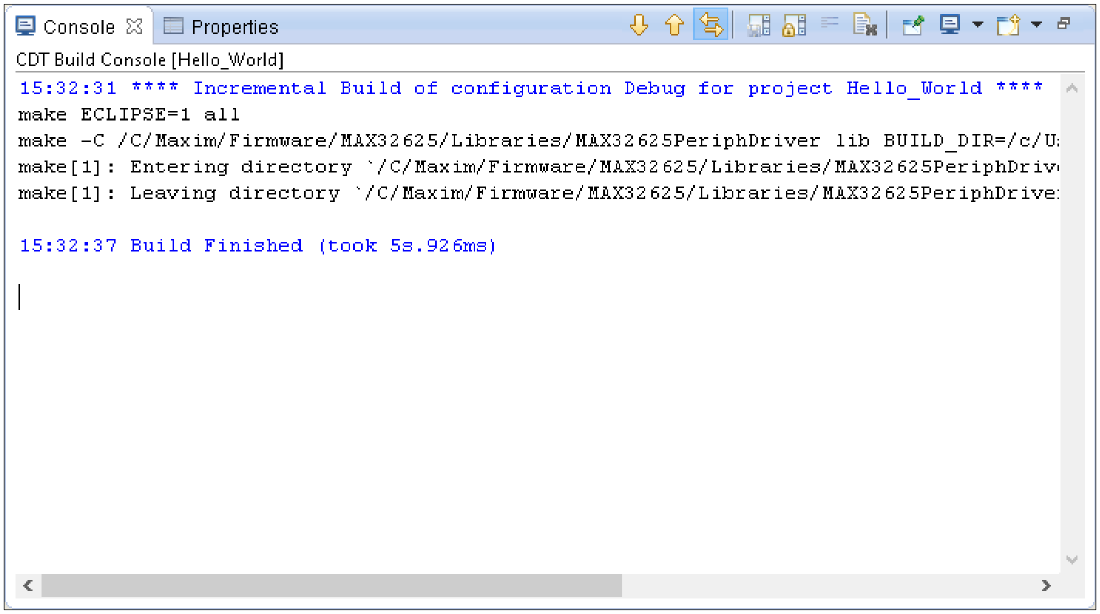
Building a Project

This section describes the steps to build one of the imported device projects called Hello_World. See *Importing Device Examples* for instructions about importing Eclipse sample projects into your workspace.

To build a project, complete the following steps:

1. Select **Window** > **Show View** > **Project Explorer** from the menu bar. The **Project Explorer** tab appears on the Eclipse home page.
2. Select **Hello_World** > **Build Project** on the **Project Explorer** tab.

3. Select **Window** > **Show View** > **Console**. The results of the build appear on the **Console** tab, as shown in **Figure 5**.



```
15:32:31 **** Incremental Build of configuration Debug for project Hello_World ****
make ECLIPSE=1 all
make -C /C/Maxim/Firmware/MAX32625/Libraries/MAX32625PeriphDriver lib BUILD_DIR=/c/U
make[1]: Entering directory `/C/Maxim/Firmware/MAX32625/Libraries/MAX32625PeriphDriv
make[1]: Leaving directory `/C/Maxim/Firmware/MAX32625/Libraries/MAX32625PeriphDrive

15:32:37 Build Finished (took 5s.926ms)
```

Figure 5. Console tab showing build results for Hello_World.

Debugging a Project

Before debugging a project, make sure your Maxim evaluation kit (EV kit) is properly connected and powered, as shown in **Figure 6** and **Table 1**.

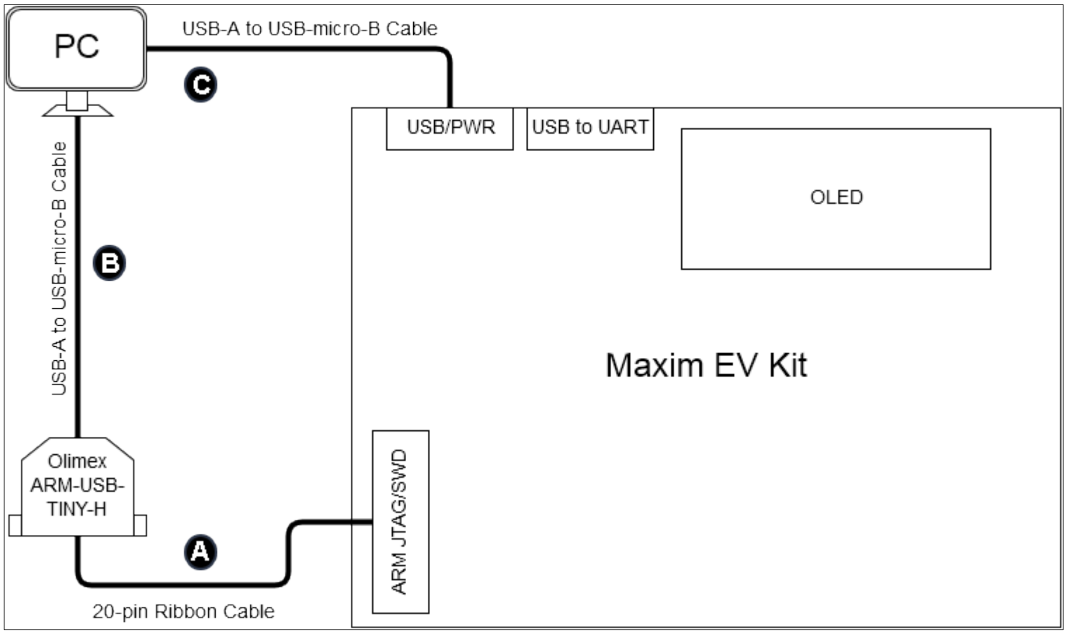


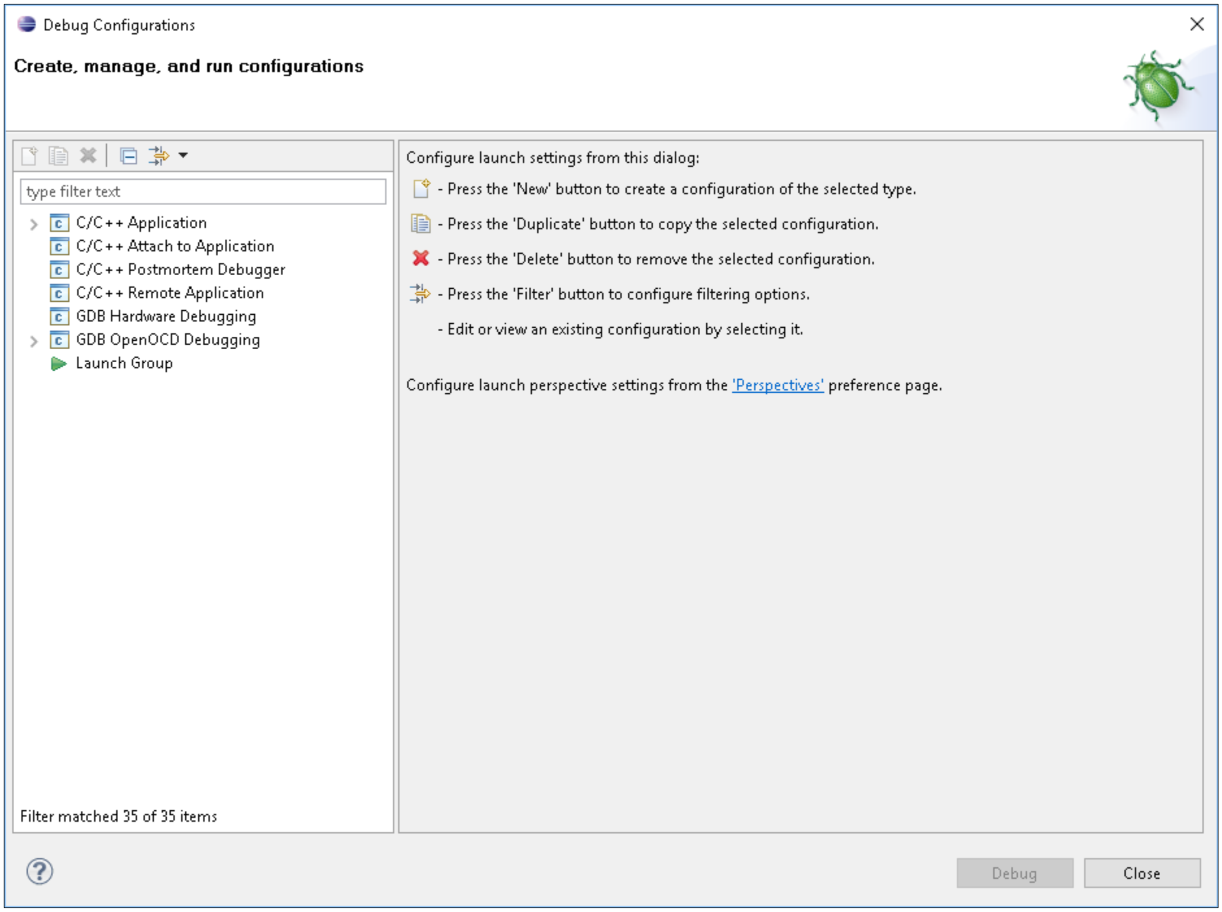
Figure 6. Maxim EV kit connections.

Table 1. EV Kit Components

Component	Description
A	20-pin ribbon cable between the Olimex ARM-USB-TINY-H adapter and the connector labeled “ARM JTAG/SWD” on the EV kit
B	USB-A to USB-B cable between the PC running Eclipse and the Olimex ARM-USB-TINY-H adapter. **Note:** Make sure that you have the most recent drivers for the adapter installed on your PC.
C	USB-A to USB-micro-B cable between the PC running Eclipse and connector labeled “USB/PWR” on the EV kit

To debug a project, complete the following steps. The TemperatureRead project is used as an example.

1. Select **Window** > **Show View** > **Project Explorer** from the menu bar. The **Project Explorer** tab appears on the Eclipse home page.
2. Select **TemperatureRead** > **Debug As** > **Debug Configurations** on the **Project Explorer** tab. The **Debug Configurations** window appears, as shown in **Figure 7**.



Configure launch settings from this dialog:

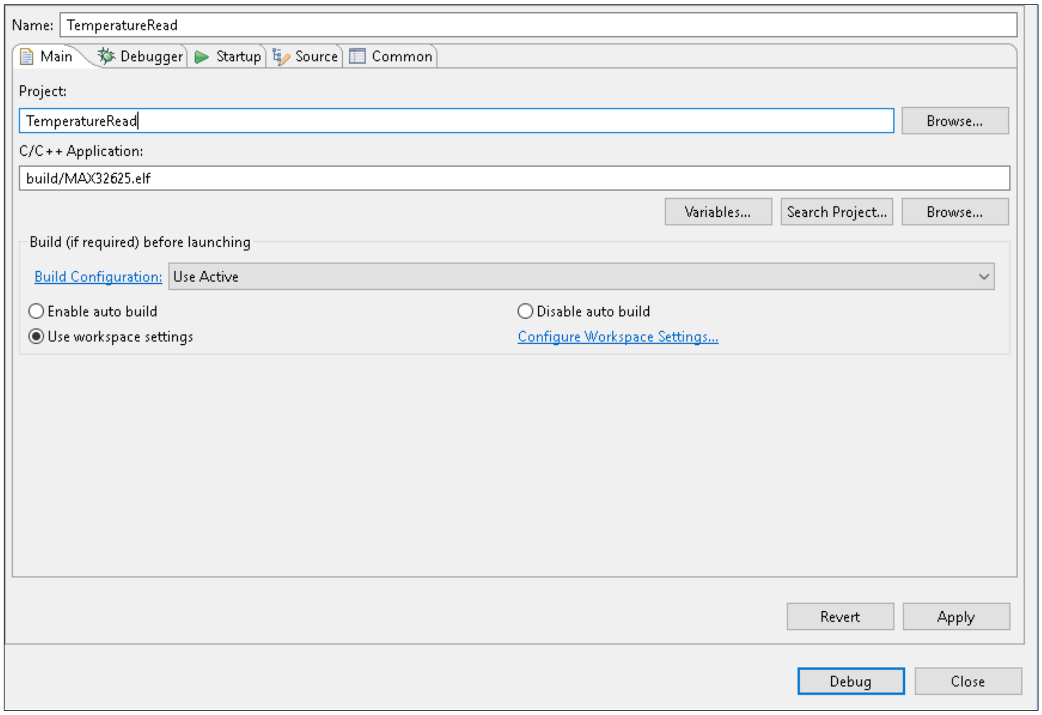
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Filter matched 35 of 35 items

Figure 7. Debug configurations window.

3. Select **GDB OpenOCD Debugging** > **TemperatureRead** in the left navigation panel. The right-hand panel reappears, showing the debug settings specific to the TemperatureRead project, as shown in **Figure 8.**



Name: TemperatureRead

Project: TemperatureRead

C/C++ Application: build/MAX32625.elf

Build (if required) before launching

Build Configuration: Use Active

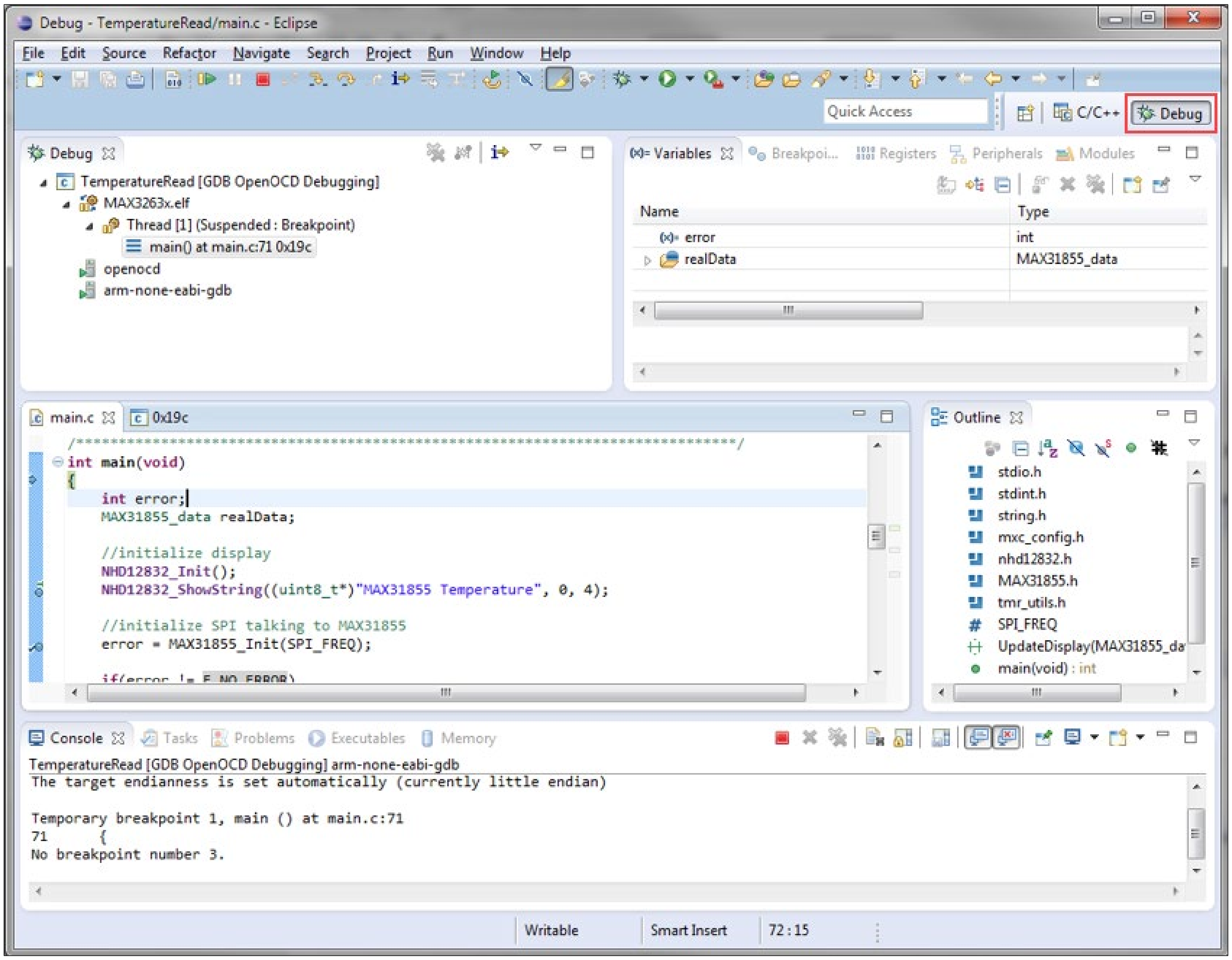
○ Enable auto build

● Use workspace settings

○ Disable auto build

Figure 8. TemperatureRead debug specifications.

4. Click **Debug**. Eclipse opens the Debug perspective and attempts to build the program, load the program to flash, and run the program. By default, a breakpoint is set at the first line of main.c. **Figure 9** shows the TemperatureRead Debug perspective.



TemperatureRead [GDB OpenOCD Debugging]
MAX3263x.elf
Thread [1] (Suspended : Breakpoint)
main() at main.c:71 0x19c
openocd
arm-none-eabi-gdb

Name	Type
error	int
realData	MAX31855_data

```
/*****************************************************************************/
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)
```

stdio.h
stdint.h
string.h
mxc_config.h
nhd12832.h
MAX31855.h
tmr_utils.h
SPI_FREQ
UpdateDisplay(MAX31855_da
main(void) : int

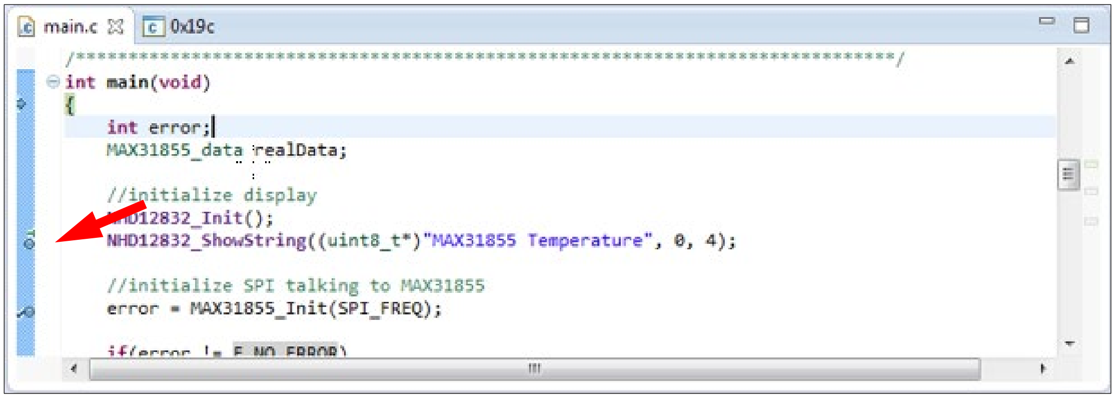
```
TemperatureRead [GDB OpenOCD Debugging] arm-none-eabi-gdb
The target endianness is set automatically (currently little endian)

Temporary breakpoint 1, main () at main.c:71
71      {
No breakpoint number 3.
```

Writable | Smart Insert | 72 : 15

Figure 9. TemperatureRead Debug perspective.

5. Set breakpoints by double-clicking the blue vertical bar next to a code line, as shown in **Figure 10**.



```
/*****************************************************************************/
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)
```

Figure 10. Setting a code line breakpoint.

6. Click **Run** > **Resume** to move from one breakpoint in the program to the next.

Connecting the EV Kit UART Device to PC to View Console Output

The serial ports on Maxim's EV kits are exposed through a USB-to-serial converter. Using a USB Standard-A to USB Micro-B cable, connect the Maxim EV kit to your PC, as shown in **Figure 11**. The appropriate drivers are installed automatically, and a serial port ID is also assigned.

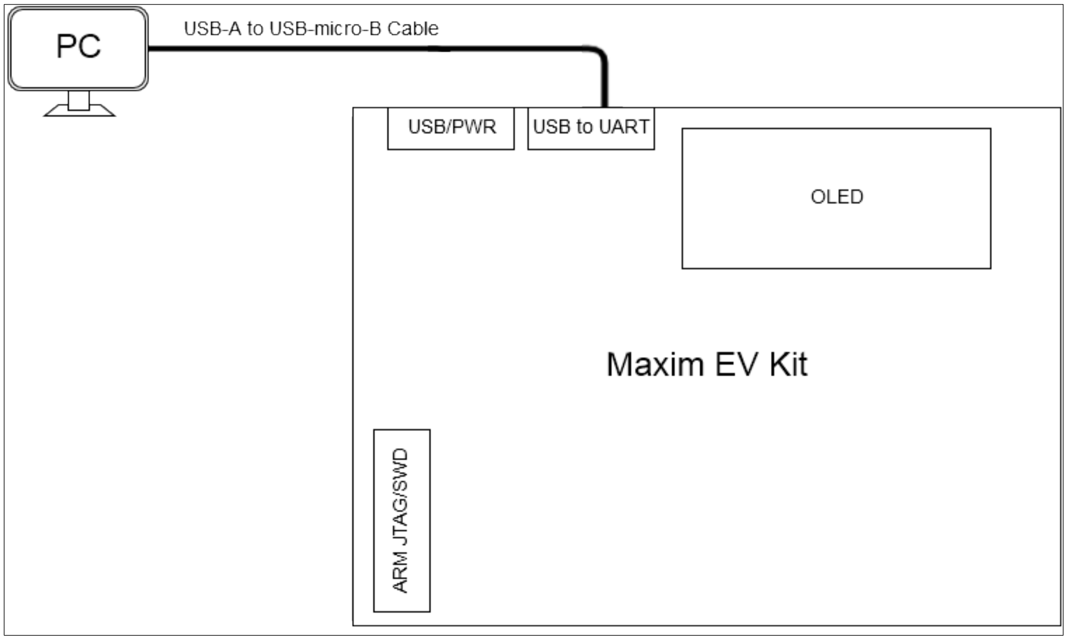


Figure 11. Serial port connection.

To determine the serial port ID assigned to the Maxim EV kit on Microsoft® Windows®, complete the following steps:

Microsoft and Windows are registered trademarks of Microsoft Corporation.

1. Select **Start** > **Control Panel** > **Device Manager**. The Device Manager window appears, as shown in **Figure 12**.

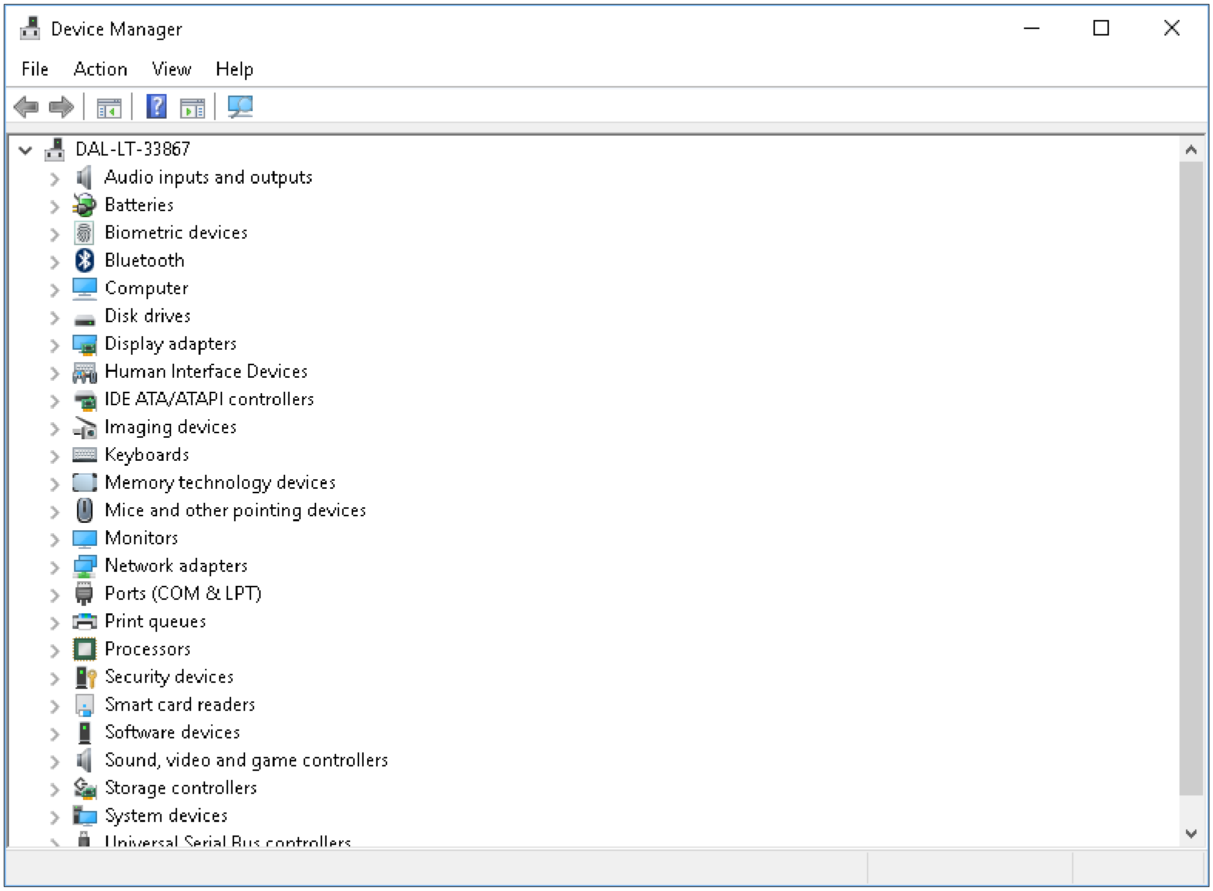
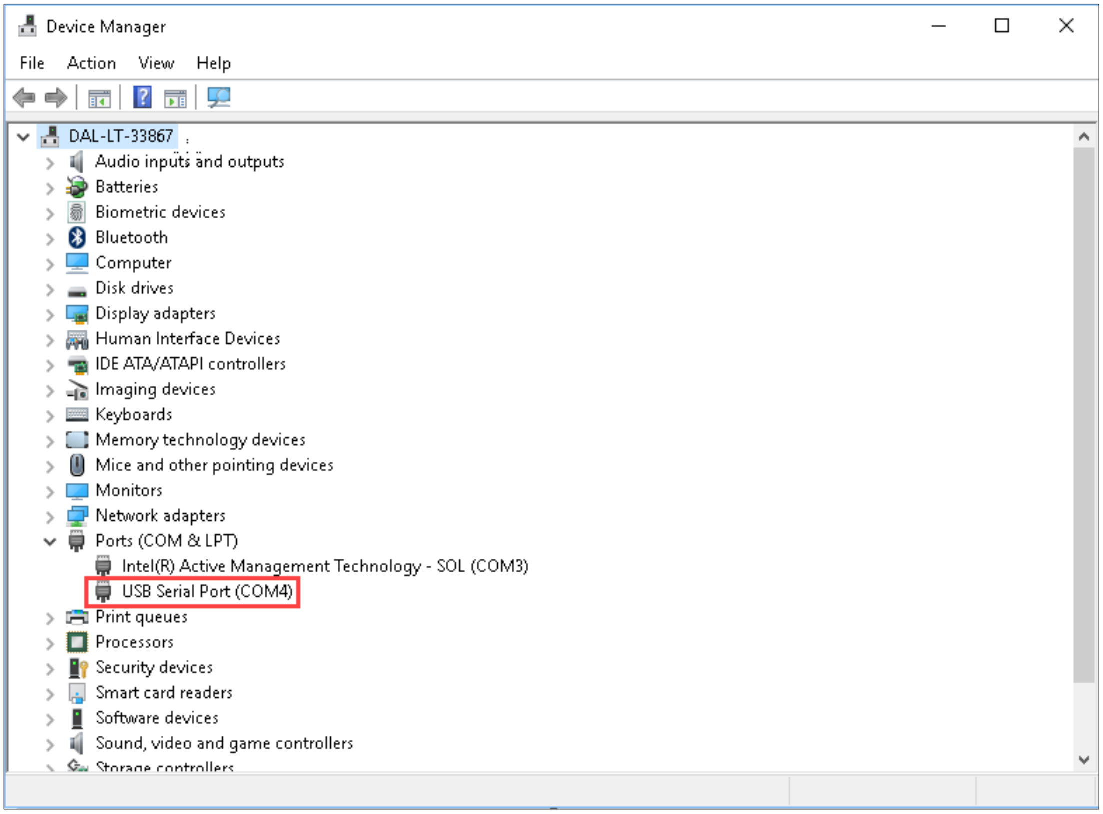


Figure 12. Device manager window.

2. Expand **Ports (COM & LPT)**. The COM port that was created by the USB driver is shown in **Figure 13**. In this case, the COM serial port is 4.



DAL-LT-33867
- Audio inputs and outputs
- Batteries
- Biometric devices
- Bluetooth
- Computer
- Disk drives
- Display adapters
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Imaging devices
- Keyboards
- Memory technology devices
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Intel(R) Active Management Technology - SOL (COM3)
  - USB Serial Port (COM4)
- Print queues
- Processors
- Security devices
- Smart card readers
- Software devices
- Sound, video and game controllers
- Storage controllers

Figure 13. COM port created by the USB driver.

Opening a Terminal Program

To view the output messages from an example application, you need a serial terminal application on your PC. If you do not already have a preferred terminal program, PuTTY is a free, open-source application that works on Linux®, UNIX®, OS X®, and Windows.

Once you install your terminal application, use it to open the serial port that you identified in **Connecting the Device to a PC**. Specify the values shown in **Table 2** for the serial port parameters:

Table 2. Terminal Application Serial Port Parameter Values

Parameter	Value
Speed	115200 baud
Data bits	8
Parity bits	None
Stop bits	1
Flow control	None

Note: After opening and configuring the serial port, any print statements you run appear in the terminal window.

Creating New Projects

You can create a new project by copying and renaming an existing project or creating a project from scratch.

Note: When using the first option, you must import the new project into Eclipse. Once imported, the new project appears on the **Project Explorer** tab.

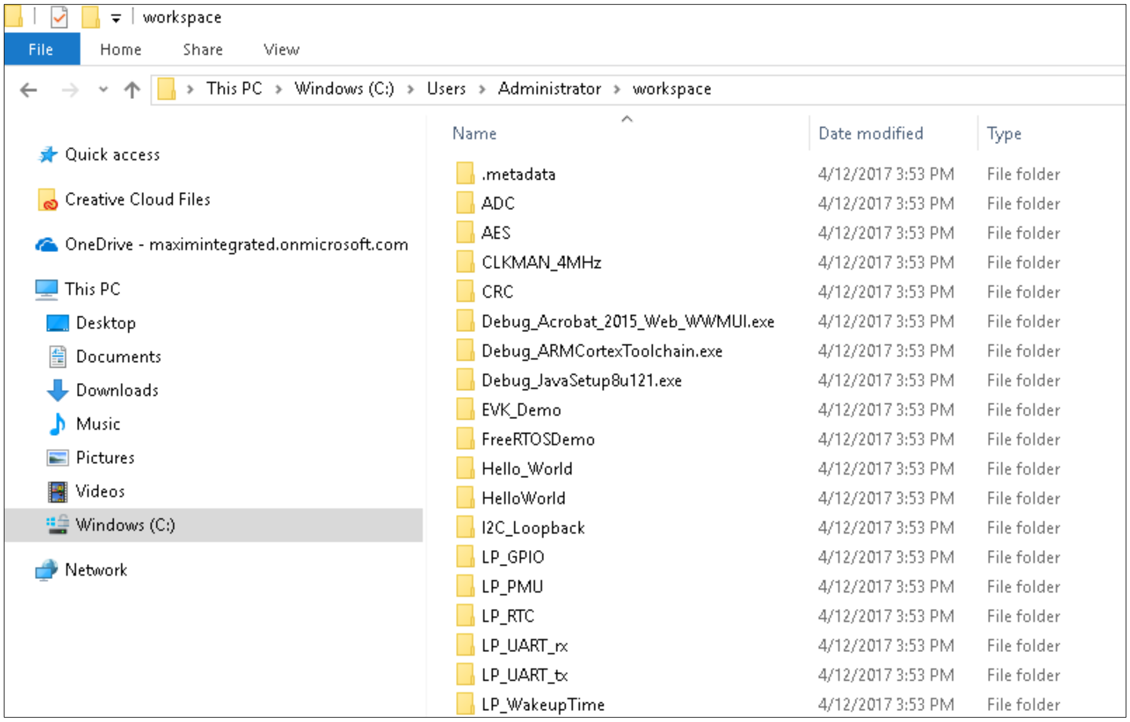
Copying and Renaming an Existing Project

This section describes the steps you need to copy and rename one of the Eclipse example projects (Watchdog) located in C:\Maxim\Firmware\<*device*>\Applications\EvKitExamples. If you have not imported any example projects into your workspace, see the instructions in *Importing Device Examples*.

To copy and rename an existing project, complete the following steps:

Linux is a registered trademark of Linus Torvalds.
Mac and OS X are registered trademarks of Apple Corporation.
UNIX is a registered trademark of The Open Group.

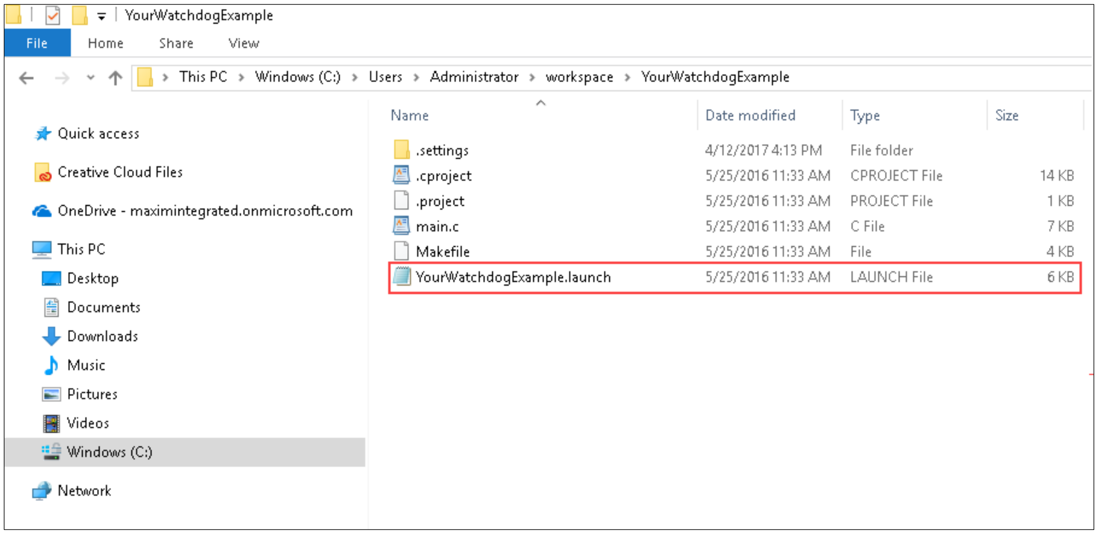
1. Navigate to C:\Users\<*user_name*>\workspace, as shown in **Figure 14.**



Name	Date modified	Type
.metadata	4/12/2017 3:53 PM	File folder
ADC	4/12/2017 3:53 PM	File folder
AES	4/12/2017 3:53 PM	File folder
CLKMAN_4MHz	4/12/2017 3:53 PM	File folder
CRC	4/12/2017 3:53 PM	File folder
Debug_Acrobat_2015_Web_WWMUI.exe	4/12/2017 3:53 PM	File folder
Debug_ARMCortexToolchain.exe	4/12/2017 3:53 PM	File folder
Debug_JavaSetup8u121.exe	4/12/2017 3:53 PM	File folder
EVK_Demo	4/12/2017 3:53 PM	File folder
FreeRTOSDemo	4/12/2017 3:53 PM	File folder
Hello_World	4/12/2017 3:53 PM	File folder
HelloWorld	4/12/2017 3:53 PM	File folder
I2C_Loopback	4/12/2017 3:53 PM	File folder
LP_GPIO	4/12/2017 3:53 PM	File folder
LP_PMU	4/12/2017 3:53 PM	File folder
LP_RTC	4/12/2017 3:53 PM	File folder
LP_UART_rx	4/12/2017 3:53 PM	File folder
LP_UART_tx	4/12/2017 3:53 PM	File folder
LP_WakeupTime	4/12/2017 3:53 PM	File folder

Figure 14. Imported Eclipse example projects.

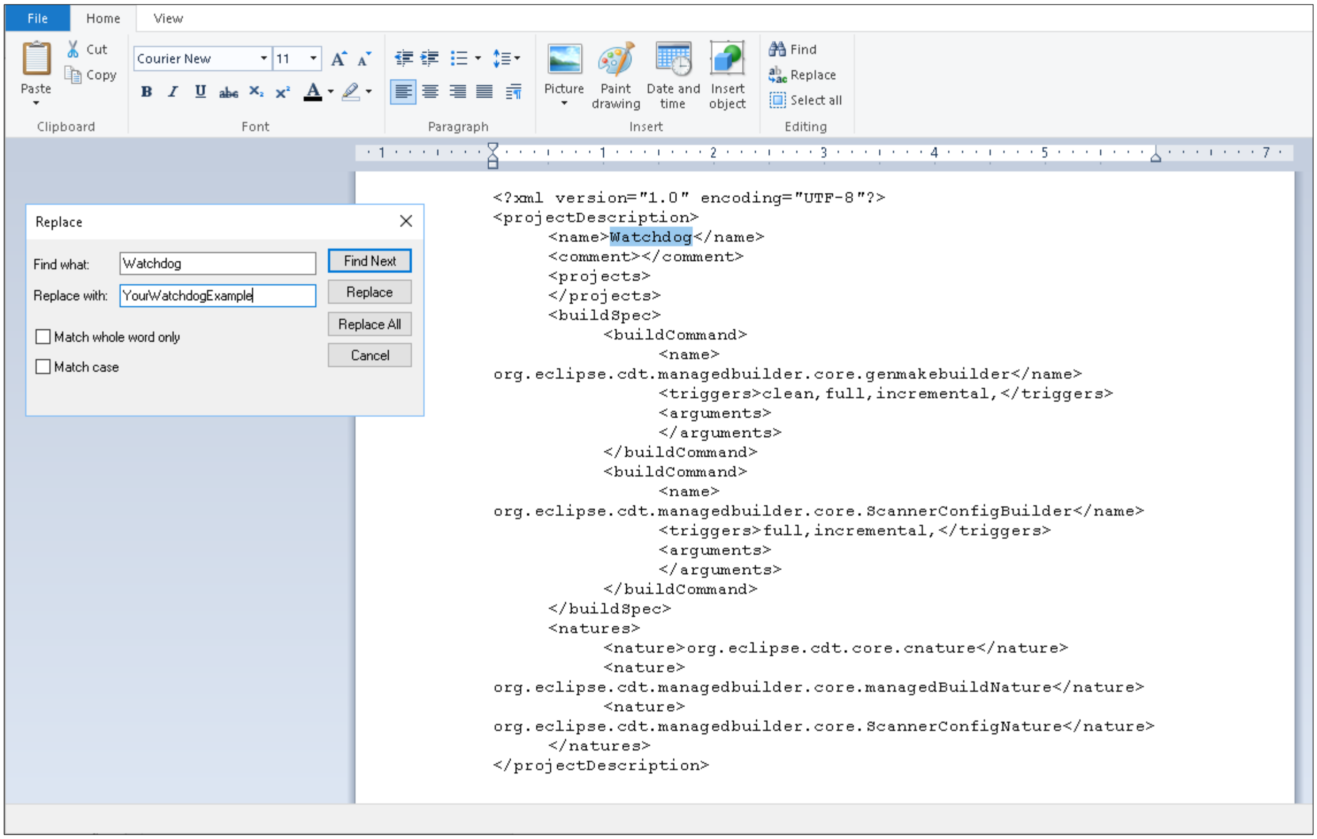
2. Copy and paste the Watchdog example project into your workspace folder.
3. Rename the copy to YourWatchdogExample.
4. Open the **YourWatchdogExample** folder.
5. Delete the build folder.
6. Rename the Watchdog.launch file to YourWatchdogExample.launch, as shown in **Figure 15**.



Name	Date modified	Type	Size
.settings	4/12/2017 4:13 PM	File folder	
.cproject	5/25/2016 11:33 AM	CPROJECT File	14 KB
.project	5/25/2016 11:33 AM	PROJECT File	1 KB
main.c	5/25/2016 11:33 AM	C File	7 KB
Makefile	5/25/2016 11:33 AM	File	4 KB
YourWatchdogExample.launch	5/25/2016 11:33 AM	LAUNCH File	6 KB

Figure 15. YourWatchdogExample.launch file.

7. Open the .cproject, .project, and YourWatchdogExample.launch files in a text editing tool, and change every occurrence of the Watchdog project name to YourWatchdogExample. **Figure 16** shows the only occurrence of the Watchdog project name in the .project file.



Find what: Watchdog

Replace with: YourWatchdogExample

□ Match whole word only

□ Match case

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>Watchdog</name>
	<comment></comment>
	<projects>
	</projects>
	<buildSpec>
		<buildCommand>
			<name>
org.eclipse.cdt.managedbuilder.core.genmakebuilder</name>
			<triggers>clean,full,incremental,</triggers>
			<arguments>
			</arguments>
		</buildCommand>
		<buildCommand>
			<name>
org.eclipse.cdt.managedbuilder.core.ScannerConfigBuilder</name>
			<triggers>full,incremental,</triggers>
			<arguments>
			</arguments>
		</buildCommand>
	</buildSpec>
	<natures>
		<nature>org.eclipse.cdt.core.cnature</nature>
		<nature>
org.eclipse.cdt.managedbuilder.core.managedBuildNature</nature>
		<nature>
org.eclipse.cdt.managedbuilder.core.ScannerConfigNature</nature>
	</natures>
</projectDescription>
```

Figure 16. Finding and replacing the Watchdog project name in the .project file.

8. Select **File** > **Import** from the menu bar. The **Import** dialog box appears, as shown in **Figure 17**.

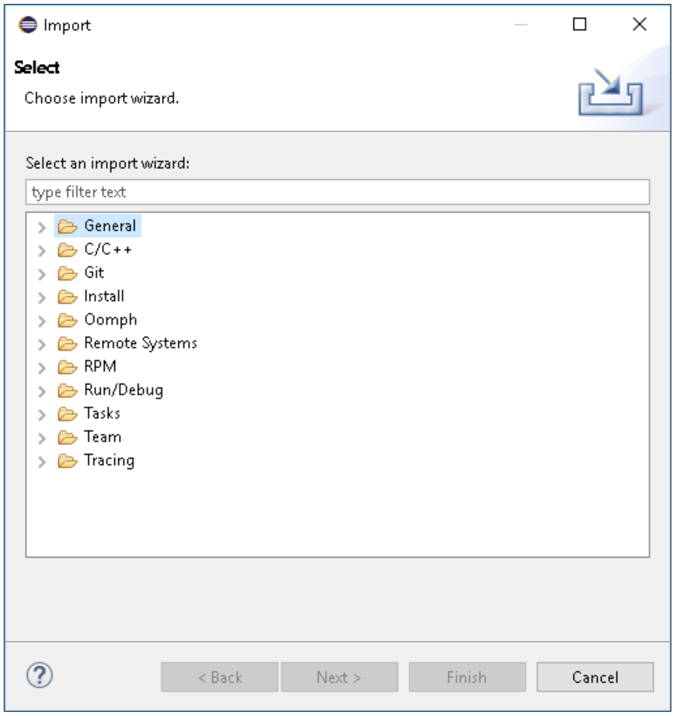
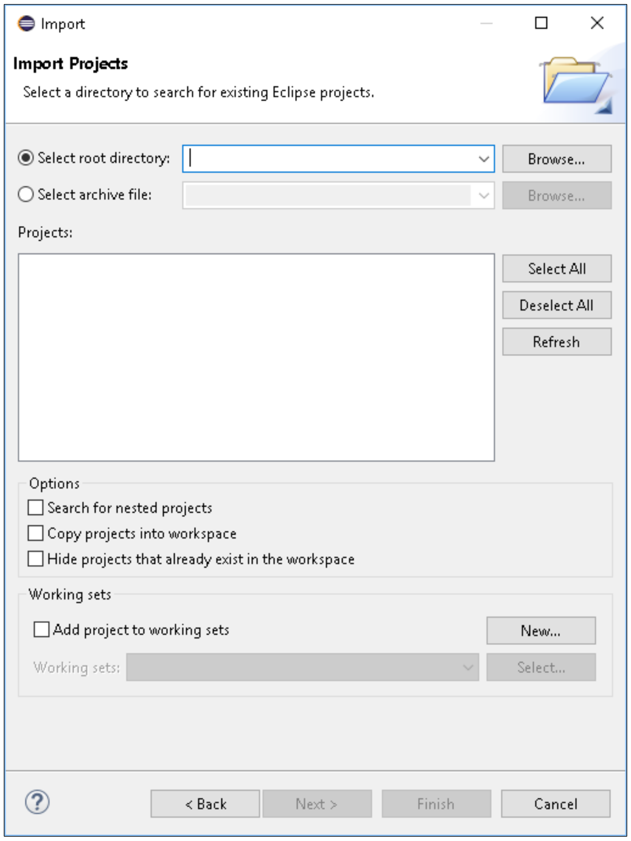


Figure 17. Import dialog box.

9. Expand the **General** folder and select **Existing Projects into Workspace**.

10. Click **Next**. The **Import Projects** dialog box appears, as shown in **Figure 18**.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

Working sets

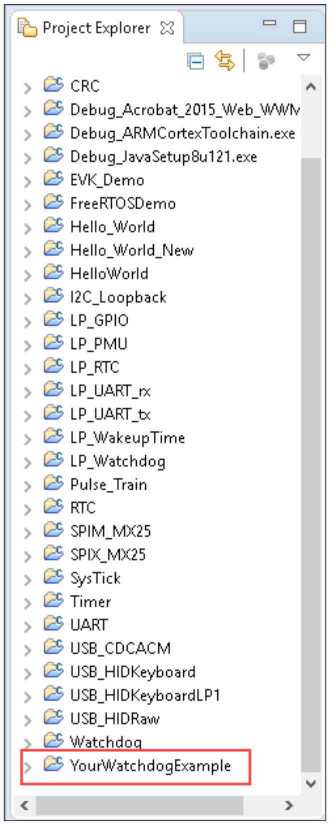
□ Add project to working sets

Working sets:

Figure 18. Import Projects dialog box.

11. Click **Browse**. The **Browse For Folder** dialog box appears.
12. Scroll the list, expand the workspace directory containing your new project, and select **YourWatchdogExample**.
13. Click **OK**. The **Import Projects** dialog box reappears, showing the root directory you selected in the **Select root directory** field and the YourWatchdogExample project in the **Projects** field.
14. (*Optional*) Select the **Copy projects into workspace** check box if the new project is located outside your workspace and you want to move it inside this directory.

15. Click **Finish**. The YourWatchdogExample project now appears on the **Project Explorer** tab, as shown in **Figure 19**.



CRC
Debug_Acrobat_2015_Web_WWM
Debug_ARMCortexToolchain.exe
Debug_JavaSetup8u121.exe
EVK_Demo
FreeRTOSDemo
Hello_World
Hello_World_New
HelloWorld
I2C_Loopback
LP_GPIO
LP_PMU
LP_RTC
LP_UART_rx
LP_UART_tx
LP_WakeupTime
LP_Watchdog
Pulse_Train
RTC
SPIM_MX25
SPIX_MX25
SysTick
Timer
UART
USB_CDCACM
USB_HIDKeyboard
USB_HIDKeyboardLP1
USB_HIDRaw
Watchdog
YourWatchdogExample

Figure 19. Project Explorer tab showing your project.

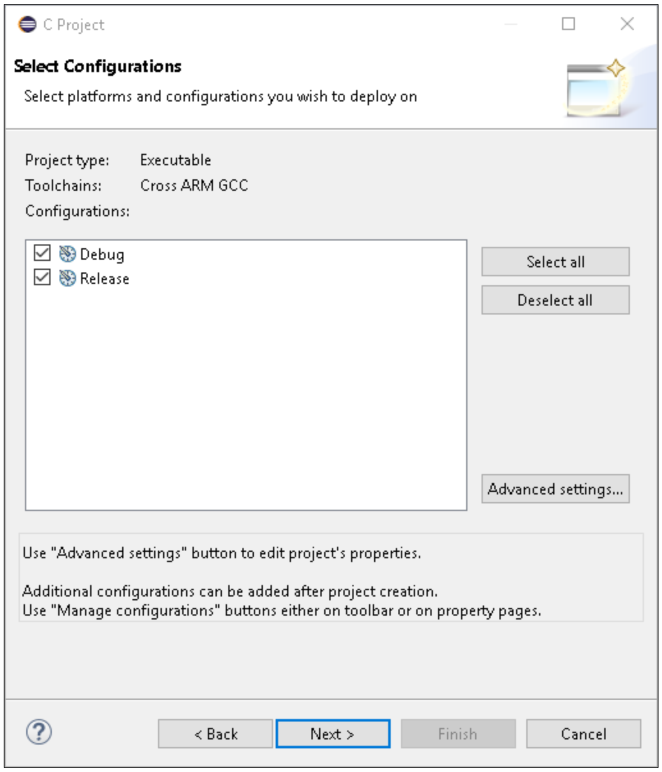
Creating a Project from Scratch

This section describes the procedure to create a new C project called Hello_World_New from scratch.

To create a new C project from scratch, complete the following steps:

1. Select **File** > **New** > **C Project** from the menu bar. The **C Project** dialog box appears.
2. Specify a name for the project in the **Project name** field (e.g., specify Hello_World_New).
3. Expand the **Executable** folder in the **Project type** field and select **Empty Project**. By default, only the project types that are buildable on the system on which Eclipse is running are listed.
4. Select **Cross ARM GCC** from the list in the **Toolchains** field.

5. Click **Next**. The **Select Configurations** dialog box appears, as shown in **Figure 20**.



C Project

Select Configurations

Select platforms and configurations you wish to deploy on

Project type: Executable
Toolchains: Cross ARM GCC
Configurations:

☑ Debug
☑ Release

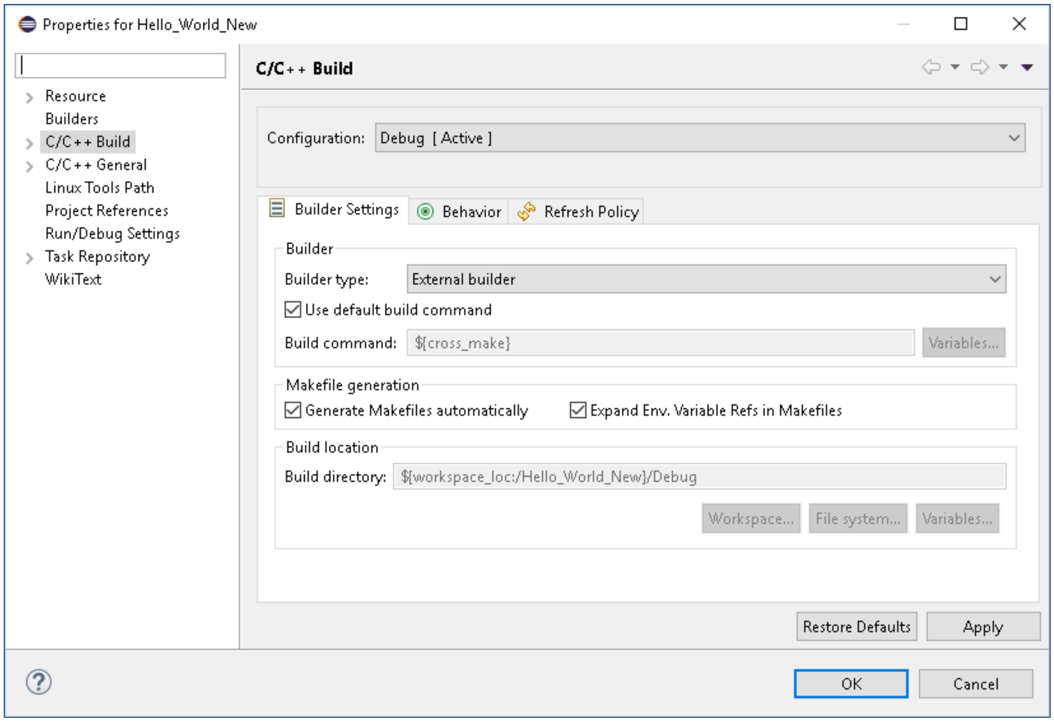
Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

Figure 20. Select Configurations dialog box.

6. Uncheck the **Release** check box in the **Configurations** field.
7. Click **Advanced settings**. The **Properties for Hello_World_New** dialog box appears.

8. Select the **C/C++ Build** option, as shown in **Figure 21**.



Configuration: Debug [Active]

Builder type: External builder

☑ Use default build command

Build command: ${cross_make}

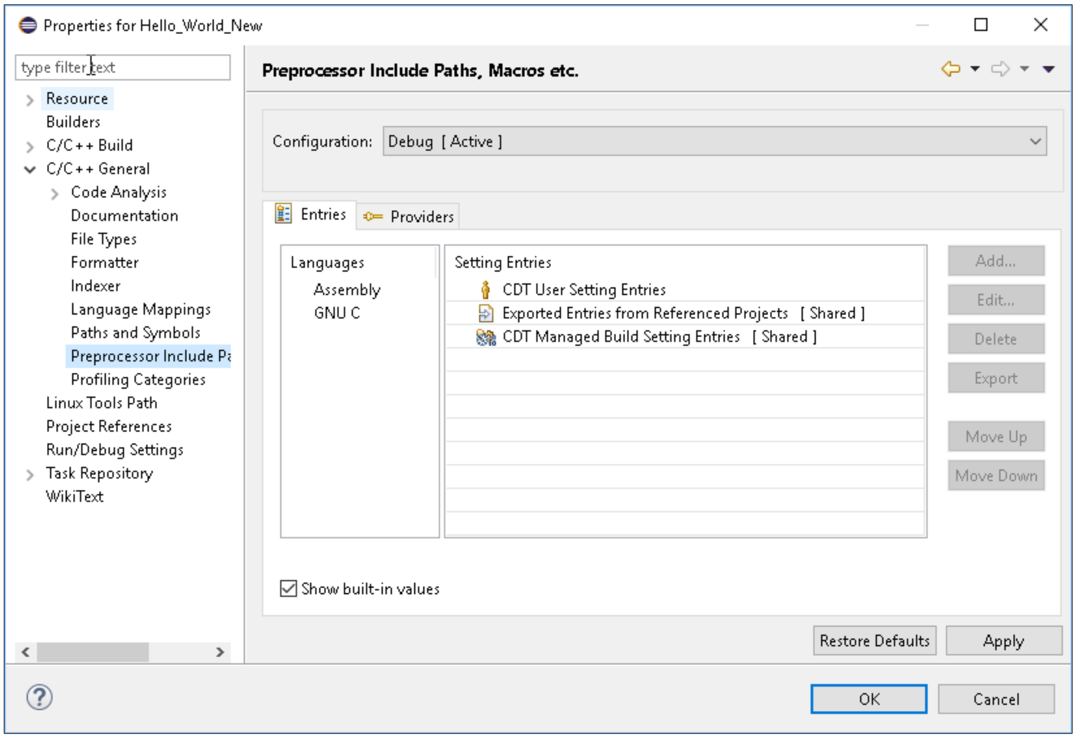
☑ Generate Makefiles automatically ☑ Expand Env. Variable Refs in Makefiles

Build directory: ${workspace_loc:/Hello_World_New}/Debug

Figure 21. Properties for Hello_World_New dialog box.

9. Uncheck the **Use default build command** check box.
10. Replace the value in the **Build command** field with make Eclipse=1.
11. Uncheck the **Generate Makefiles automatically** check box.
12. Replace the value in the **Build directory** field with ${workspace_loc:/Hello_World_New}.

13. Expand the **C/C++ General** option, and select **Preprocessor Include Paths, Macros etc.** The **Preprocessor Include Paths, Macros etc.** dialog box appears, as shown in **Figure 22**.



Configuration: Debug [Active]

Languages	Setting Entries
Assembly	CDT User Setting Entries
GNU C	Exported Entries from Referenced Projects [Shared]
	CDT Managed Build Setting Entries [Shared]

☑ Show built-in values

Figure 22. Preprocessor Include Paths, Micros etc. dialog box.

14. Click the **Providers** tab.
15. Select the **CDT GCC Build Output Parser** check box.
16. Replace the value in the **Compiler command pattern** field with .*gcc.
17. Click **Move Up** to move the **CDT GCC Build Output Parser** field to second position on the list.
18. Select the **CDT GCC Built-in Compiler Settings Cross ARM** field and click **Move Up** to move it to third position on the list.
19. Click **OK**. The **Select Configurations** dialog box reappears.
20. Click **Next**. The **Cross GNU Arm Toolchain** dialog box appears.
21. Click **Finish**. The new C project appears on the **Project Explorer** tab.

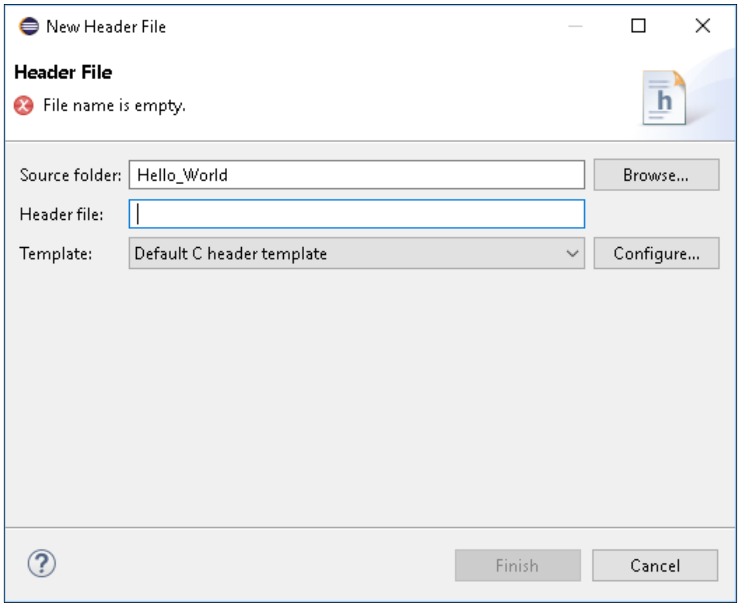
Adding Header and Source Files to Project

New projects require header and source files. The process to add header and source files is similar. This chapter describes the process for adding a new header file to the Hello_World project.

To add a new header file, complete the following steps:

1. Navigate to the **Project Explorer** tab.

2. Right-click the Hello_World project and select **New** > **Header File**. The **New Header File** dialog box appears, as shown in **Figure 23**.



Header File

File name is empty.

Source folder: Hello_World

Header file:

Template: Default C header template

Figure 23. New Header File dialog box.

3. Specify the name of the header file in the **Header file** field. Remember to add the .h extension to all header file names and the .c extension to all C source file names.
4. Click **Finish**. The new header file is added under the Hello_World project on the **Project Explorer** tab.
5. Open the new header file using a text editor.
6. Add the appropriate contents to the file.
7. Select **File > Save** to save your changes

Note: If you are adding a new C source file (such as Hello_World_New.c) and want to build this source file with the project, complete the following steps:

1. Navigate to the **Hello_World** project folder.
2. Open the project folder.
3. Open the Makefile using a text editor.
4. Add the following line to the source file section:

   ```
   SRCS += Hello_World_New.c
   ```

5. Click **File** > **Save** to save your changes.

Troubleshooting Unresolved Symbols

The Maxim EV kit examples are configured to discover the Maxim library include paths from the output of a project Makefile. After a successful build, the source files should not have unresolved include paths or symbols. If the source files are still showing unresolved errors, verify that the project is configured to discover the include paths by checking if the source file has a key symbol on its icon.

To check if the key symbol appears on the icons for the Hello_World and the Timer projects, complete the following steps:

1. Navigate to the **Project Explorer** tab.
2. Expand the **Hello_World** folder, as shown in **Figure 24**.

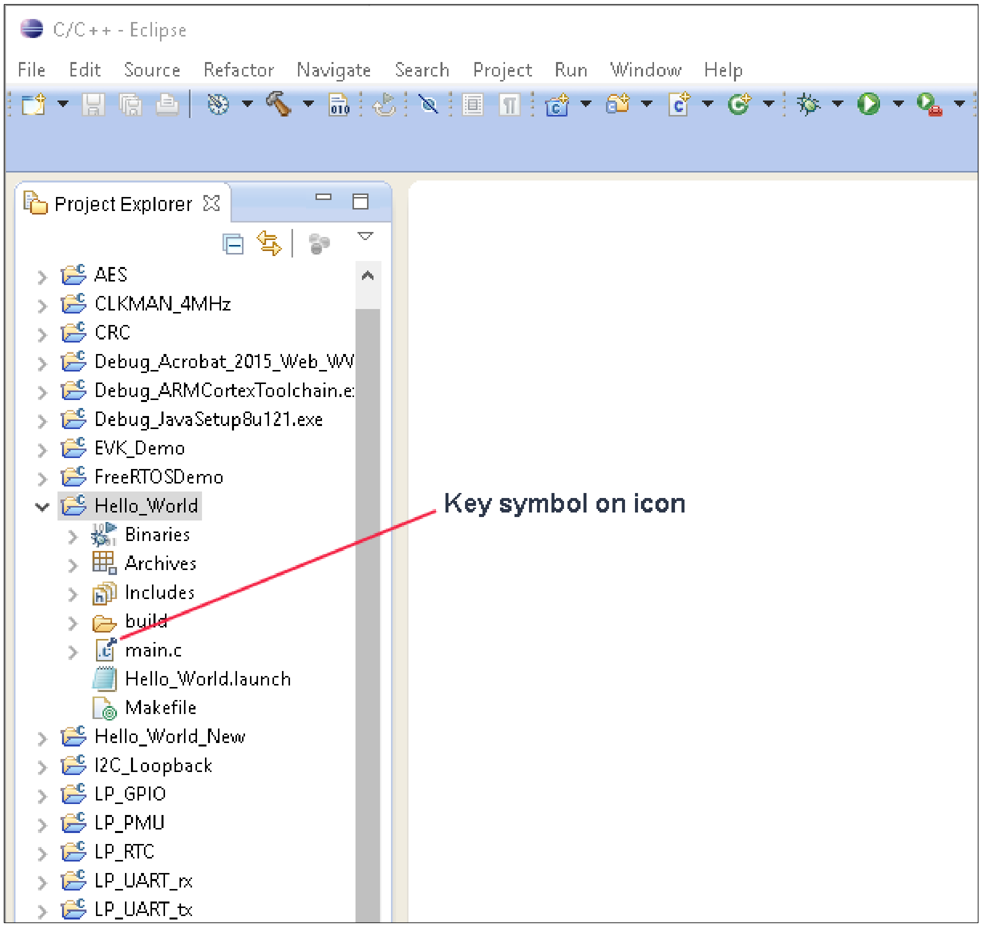
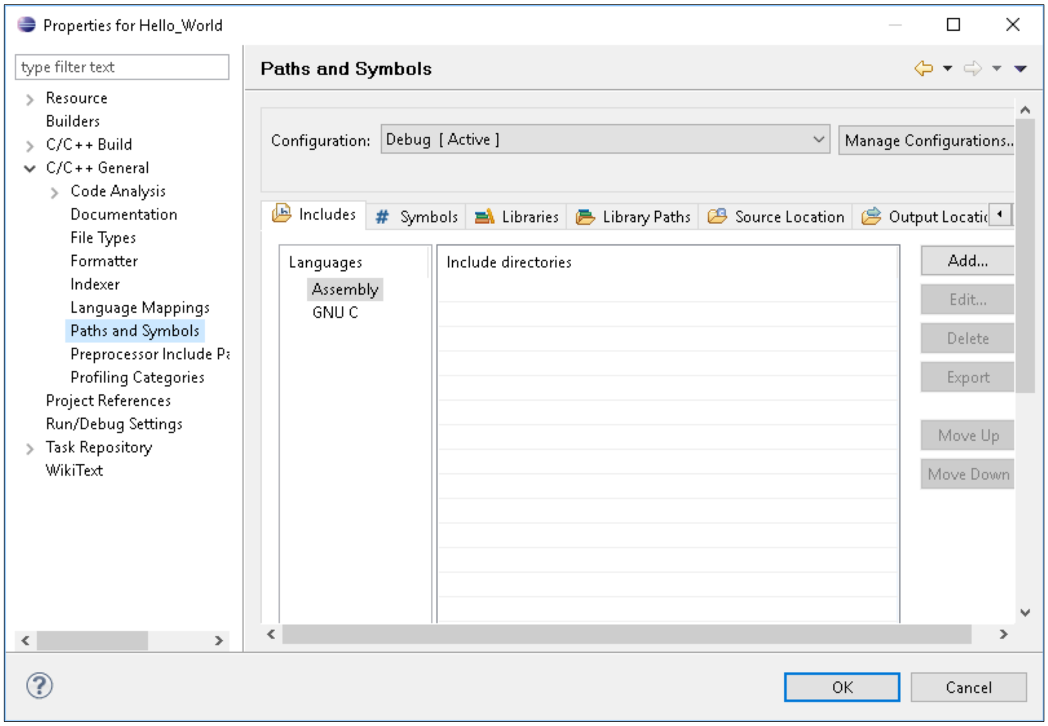


Figure 24. Hello_World main.c source file.

3. Complete one of the following actions to troubleshoot unresolved symbols:
    a. If the key symbol appears on the source file icon (as with the Hello_World project), complete the following steps:
        i. Select **Hello_World** > **Clean Project**.
        ii. Select **Hello_World** > **Build Project**.
        iii. Select **Hello_World** > **Index** > **Freshen All Files**.
        iv. Select **Hello_World** > **Properties**. The **Properties for Hello_World** dialog box appears.

v. Expand the **C/C++ General** folder and select **Paths and Symbols**. The Paths and Symbols controls appear in the right panel on the **Includes** tab, as shown in **Figure 25**.



Configuration: Debug [Active]

Figure 25. Includes tab on the Paths and Symbols dialog box.

vi. Select the **GNU C** option in the **Languages** field.

vii. Click **Add**. The **Add directory path** dialog box appears, as shown in **Figure 26**.

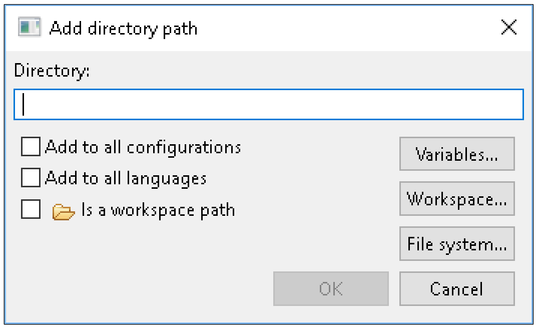


Figure 26. Add directory path dialog box.

viii. Add the following library include paths manually to the **Directory** field one at a time, followed by clicking **OK**. Replace <dev> with the part number for your device.

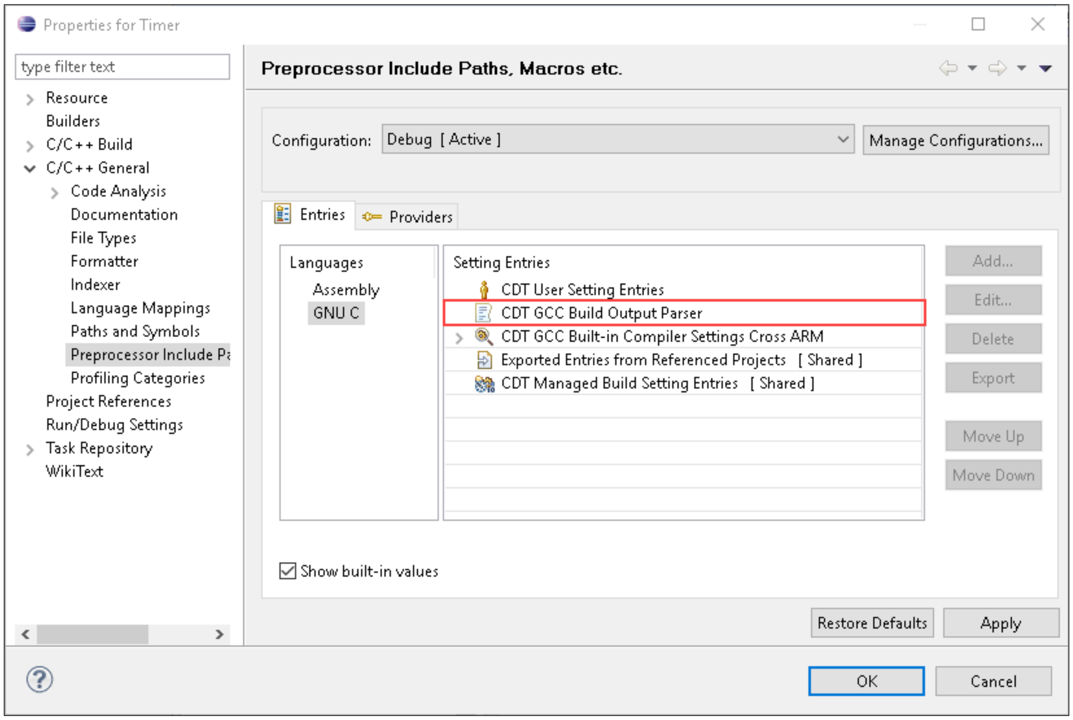
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/Boards/Include
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/Boards/EvKit_V1/Include
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/<dev>PeriphDriver/Include
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/FreeRTOS/Source/include
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/CMSIS/Device/Maxim/<dev>/Include
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/CMSIS/Include
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/MAXUSB/include/core/arm
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/MAXUSB/include/core
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/MAXUSB/include/enumerate
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/MAXUSB/include/devclass
- ${MAXIM_PATH}/Firmware/<dev>/Libraries/MAXUSB/include

ix. Click **OK**.

b. If the key symbol does not appear on the source file icon (as with the Timer project), complete the following steps:

i. Select **Timer** > **Properties**. The **Properties for Timer** dialog box appears.

ii. Expand the **C/C++ General** option, and select **Preprocessor Include Paths, Macros etc.** The Preprocessor Include Paths, Macros etc. controls appear in the right panel on the **Entries** tab, as shown in **Figure 27**.



Properties for Timer

Preprocessor Include Paths, Macros etc.

Configuration: Debug [Active]

Languages: Assembly, GNU C

Setting Entries
- CDT User Setting Entries
- CDT GCC Build Output Parser
- CDT GCC Built-in Compiler Settings Cross ARM
- Exported Entries from Referenced Projects [Shared]
- CDT Managed Build Setting Entries [Shared]

☑ Show built-in values

Figure 27. Entries tab on the Preprocessor Include Paths, Macros etc. dialog box.

iii. Verify that CDT GCC Build Output Parser is the second list item in the **Setting Entries** field.

iv. Repeat step iv through step ix for the Timer project to add the library include paths manually.

References/Other Resources

Arm SDK Download Links:

Windows: https://www.maximintegrated.com/en/design/software-description.html/swpart=SFW0001500A

Mac: https://www.maximintegrated.com/en/design/software-description.html/swpart=SFW0001660A

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	4/16	Initial release	—
1	4/19	Replaced document	all

©2019 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.