Inspiring Motion
Since 1988

Maestro Developer Studio (MDS) User Guide

April 2016 (Ver. 2.0)

www.elmomc.com

Elmo
Motion Control

Notice

This guide is delivered subject to the following conditions and restrictions:

- This guide contains proprietary information belonging to Elmo Motion Control Ltd. Such information is supplied solely for the purpose of assisting users of Elmo's servo drive(s) in their installation.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based are subject to change without notice.
- Elmo Motion Control and the Elmo Motion Control logo are trademarks of Elmo Motion Control Ltd.
- Information in this document is subject to change without notice.

Elmo Motion Control	Elmo Motion Control and the Elmo Motion Control logo are registered trademarks of Elmo Motion Control Ltd.
EtherCAT Technology Group	EtherCAT Conformance Tested. EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
CANopen	CANopen compliant. CANopen® is a registered trademark and patented technology, licensed by CAN in Automation (CiA) GmbH, Kontumazgarten 3, DE-90429 Nuremberg, Germany.

Document no. GDS_UM3 (Ver. 2.0)

Revision History

Version	Date	Details
Ver. 1.0		Initial Release
Ver. 1.1		Updated to include detailed explanation of: • Copy files via remote system browser • Log files • User defined path • Clear auto exe • Inline documentation • Download executable tool in the menu
Ver. 1.101		Small changes and template update
Ver. 1.2		

Ver. 2.0	Updated to include detailed explanation of: • Multiple platform support - extended to support both Gold and Platinum Maestro • Ability to choose what toolchain to use • Ability to choose whether to link the API library statically or dynamically. • New MDS Installer • General user interface updates

Chapter 1: Before You Begin

1.1 How to Use this Manual

This manual is part of the Elmo documentation set. It should be used in conjunction with the other manuals that describe Elmo products to be used in conjunction with Maestro Developer Studio (MDS).

Section 1.4 Installing the MDS of this manual guides you through the process of installing the MDS and first getting started. At the end of this chapter, you will have a Maestro (Gold or Platinum) connected to your PC with a simple “Hello World” example program running on it in debug mode.

To learn more about more advanced features of the Studio, and how to perform other tasks refer read the other chapters.

Important: All operation processes, configurations, and so forth, that apply to Gold Maestro also apply to Platinum Maestro.

1.2 What is Maestro Developer Studio (MDS)?

Maestro Developer Studio is an integrated development and debug environment that is customized to work with Elmo's Gold and Platinum Maestro multi-axis controllers. The Maestro Development studio includes all the standard familiar features of a development environment that allow you to create C/C++ projects, edit files and debug software.

Additional features were added to make a single, seamless working environment for the Maestro:

- Establish communication with the Maestro
- Choose what toolchain to use, according to the Maestro hardware
- Choose whether to link the API library statically or dynamically
- Download a program to the Maestro independent of size
- Run and debug your Maestro software
- Use a terminal
- Attach to Maestro to debug a running program

1.3 Terminology

Term	Explanation
Eclipse	A main developing application used for programming in C/C++.
MDS	Maestro Development Studio – A composite of several applications including Eclipse, Cygwin, Java and so on, in conjunction with Gold Maestro and Platinum Maestro supported plugins as a single component.
Gold Maestro	Gold Maestro is an advanced network based, multi-axis machine motion controller. The Gold Maestro controls any multi-axis scenario, from simple point-to-point motion to complete multi-axis coordinated or synchronized motion.
Platinum Maestro	Platinum Maestro is the advanced version of the Gold Maestro and is designed to significantly extend and enhance the power of the motion control, connectivity, and performance, with intensive focus on the ease of use.
RSE	Remote System Explorer

1.4 Installing the MDS

1.4.1 System Requirements

To install Maestro Developer Studio you need a PC with the following requirements:

- One of the following:
  - Windows 7
  - Windows 8
  - Windows 10
- At least 1 GB RAM
- Supports 64 bit processing
- Microsoft .NET Framework 4 Extended version

1.4.2 Procedure

The Maestro Developer Studio installation wizard installs the following:

- Maestro Developer Studio
- WinSCP (transparent)
- Elmo sample program libraries or library updates
- CygWin (transparent)

The installation directory includes the following file:

- MaestroDeveloperStudioSetup.exe.

To install or update MDS and all its components:

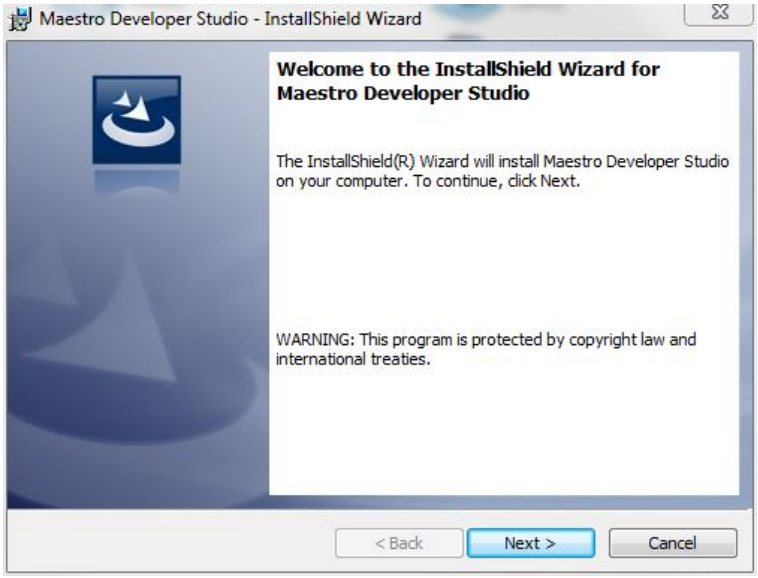
Notes:

- When installing in a system with Windows 7, 8, or 10, make sure to right-click the **MaestroDeveloperStudioSetup.exe** and select **Install as Administrator**.
- Before installing MDS, be sure to uninstall any previous version of GDS from your system. If a previous version of GDS is still installed on your computer while trying to install MDS, the following message appears:

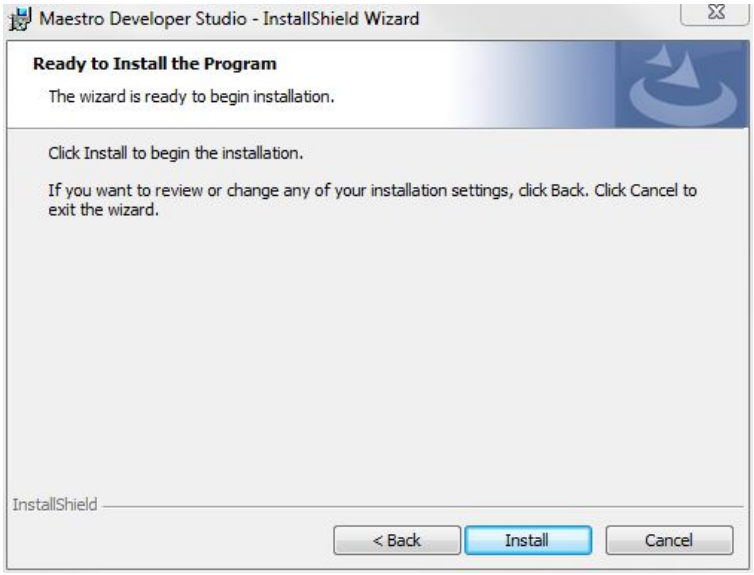
1. Right-click the **MaestroDeveloperStudioSetup.exe** file and select **Install as Administrator**.

   Note: If you only want to update the samples and the Library files you can install by double-clicking the **MaestroLibraryUpdateSetup.exe** file.

The InstallShield Wizard opens.

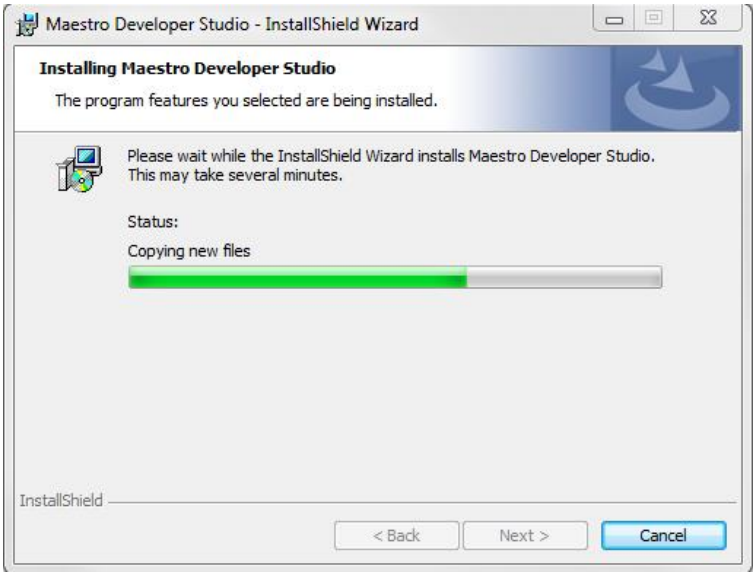


2. In the InstallShield Wizard, click **Next** to continue the installation. The **Ready to Install the Program** page appears:



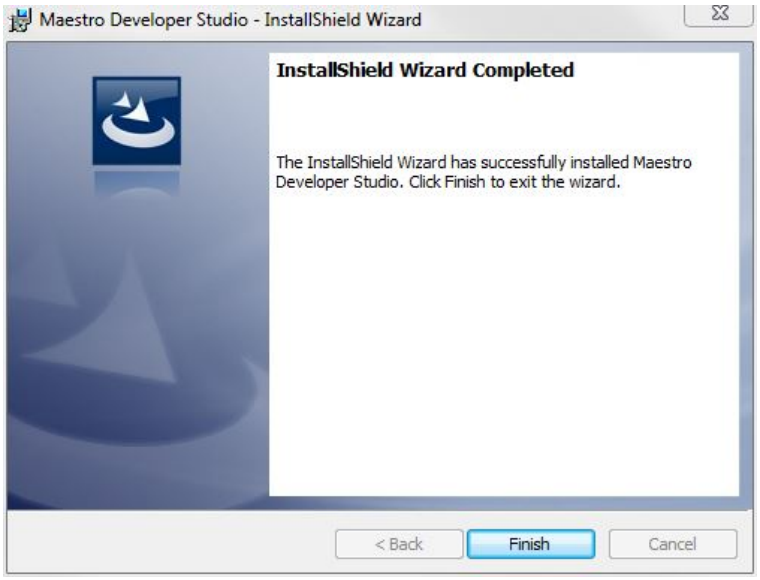
3. Click **Install**.

The installation progresses, installing each section of the application in turn.



At any time, you can click **Cancel** to cancel the installation. However, this is not recommended.

When the MDS is successfully completed, the **InstallShield Wizard Completed** page appears:



4. Click **Finish** to exit the wizard.
5. At the end of the new installation, the following icon is created on the desktop.

1.5 Starting MDS for the First Time

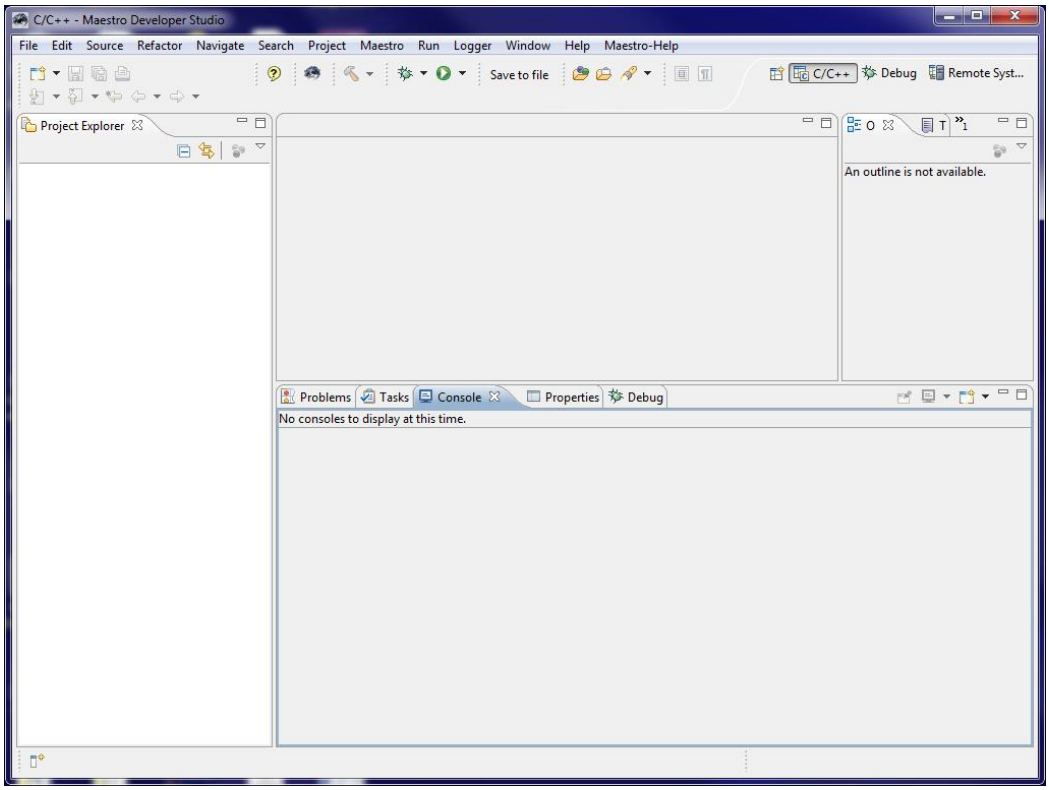
This chapter explains the basic process of the MDS and includes some basic required settings. In addition, it demonstrates some of the most commonly used features of MDS.

To run MDS:

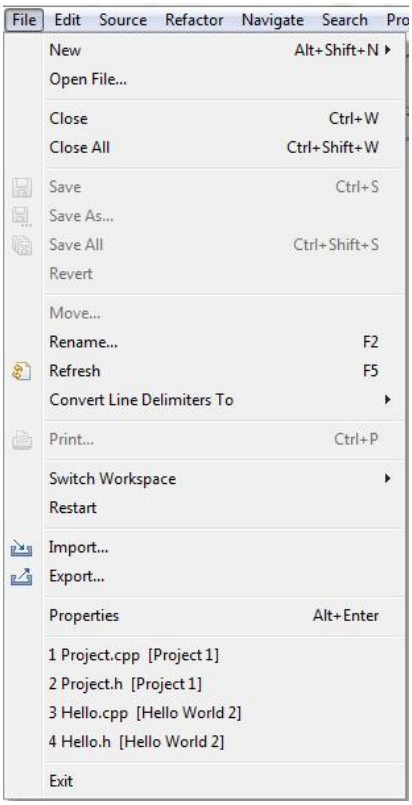
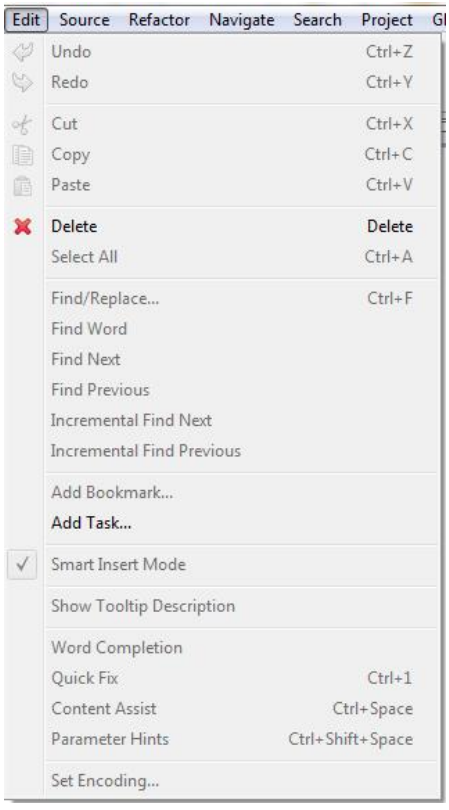
1. Connect a Maestro to your PC.
2. Double-click the **Maestro Developer Studio** icon 

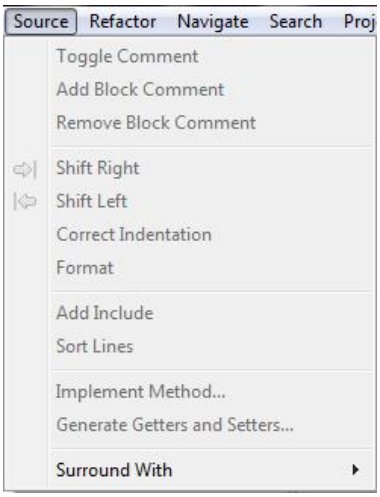
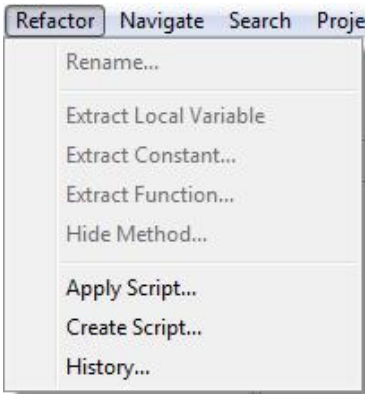
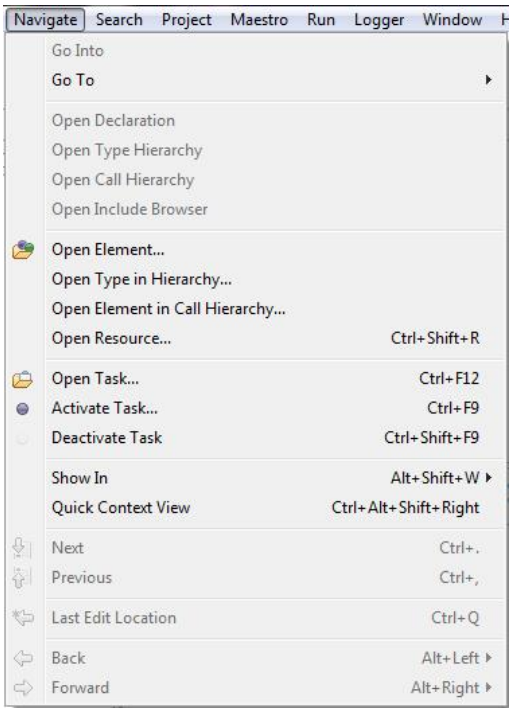
on the desktop.

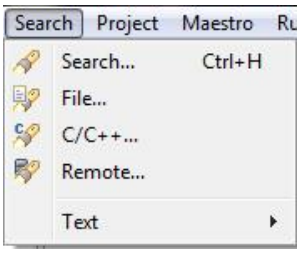
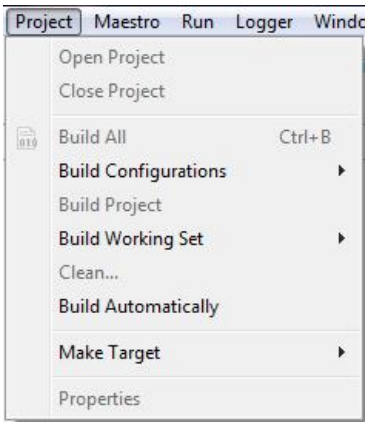
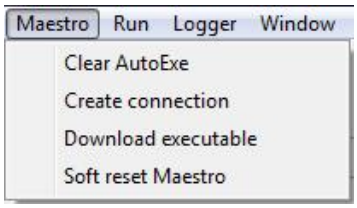
   The Maestro Developer Studio opens.

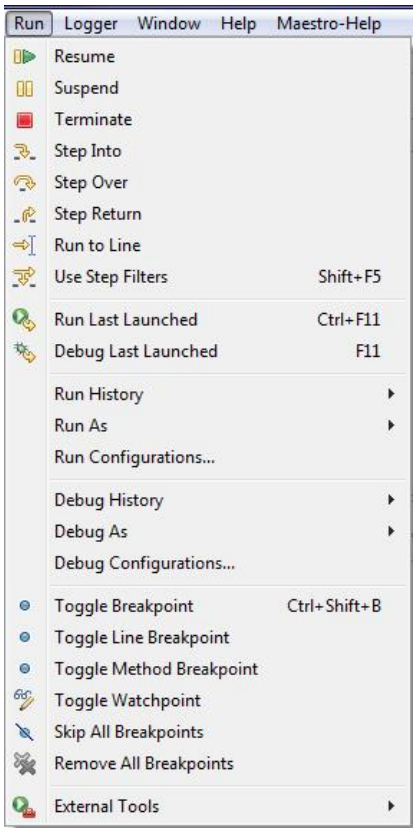
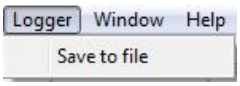
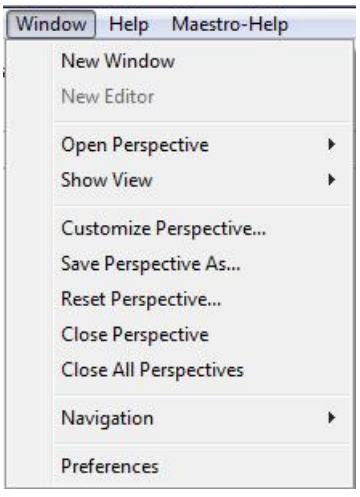


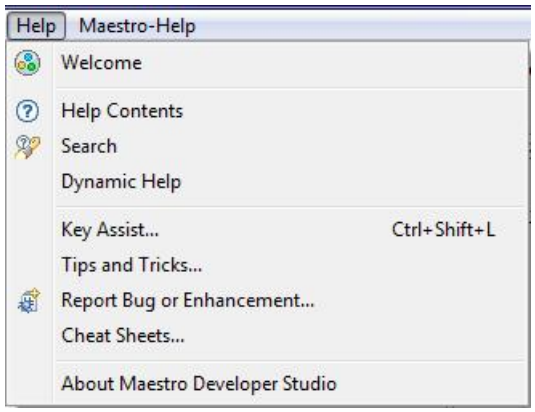
The main menu consists of the following options and icons. The Eclipse On Line Help provides explanations for most of the Eclipse options. The following table offers a short explanation of the extra options provided for the Maestro setup and operation.

Menu Item	Menu Options	Maestro Options Explained
File		Normal File options, fully described in the Eclipse On-Line Help.
Edit		Edit and set Perspectives as fully described in the Eclipse On-Line Help.

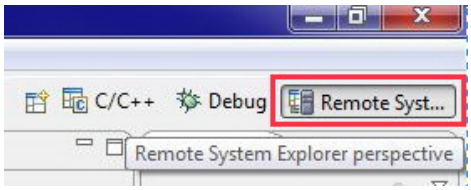
Menu Item	Menu Options	Maestro Options Explained
Source		Edit and add source files relevant to the Project. For details, refer to the full description in the Eclipse On-Line Help.
Refactor		Extract specific variables from a Project. For details, refer to the full description in the Eclipse On-Line Help.
Navigate		Navigation throughout a Project. For details, refer to the full description in the Eclipse On-Line Help.

Menu Item	Menu Options	Maestro Options Explained
Search		Search throughout a project for any function, variable, and so on, in C/C++ or a remote file. For details, refer to the full description in the Eclipse On-Line Help.
Project		Build, rebuild a target project or build all projects in the Explorer list. For details, refer to the full description in the Eclipse On-Line Help.
Maestro		• **Clear AutoExe** Clears any autoexec files installed in the Maestro. • **Create connection** Creates a connection with the Maestro hardware. • **Download executable** Downloads an autoexecutable file to the Maestro for Maestro operations. • **Soft reset Maestro** Uses a software reset to restart the Maestro.

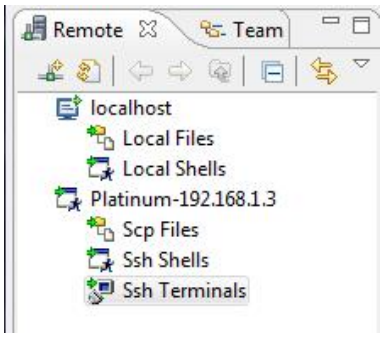
Menu Item	Menu Options	Maestro Options Explained
Run		Test the project by running the complied project. Debug using Steps or Breakpoints. For details, refer to the full description in the Eclipse On-Line Help.
Logger		**Logger** Saves a log file of all Maestro output operations, e.g. Console, Debug, using Eclipse.
Window		Open and change perspective windows. For details, refer to the full description in the Eclipse On-Line Help.

Menu Item	Menu Options	Maestro Options Explained
Help		Eclipse built-in On-Line Help for standard operations.
Maestro-Help		Specific built-in Maestro On-Line Help and context sensitive Help for each function.

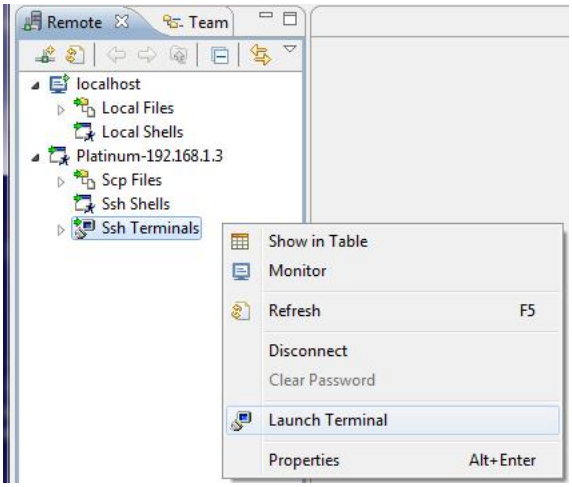
3. Create a connection between MDS and your Maestro.
   If your Maestro is connected peer to peer to your PC, follow the instructions in section 3.5.1 Peer to Peer Connection. If it is connected via a network follow the instructions in section 3.5.2 Creating a New Connection (first time only).
   Your Maestro is now connected to your PC.
4. To verify the connection using a terminal go to the Remote System perspective by clicking the **Remote System** button.



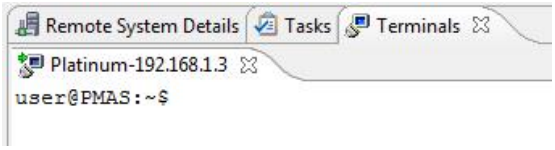
5. On the left hand side you can see a tree related to your Maestro.



6. To run the terminal right-click Ssh Terminals and select **Launch Terminal**.

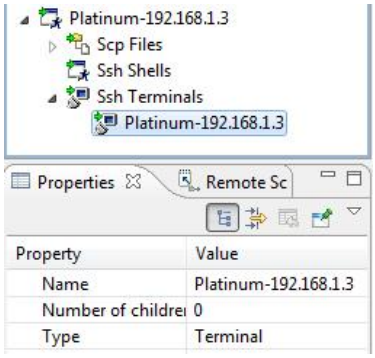


The terminal tab in the lower part of the window becomes active and displays the Maestro's message:



You are now connected to your Maestro.

In addition, a host connection is displayed in the left lower corner.



You can now use a simple example to become familiar with opening a project, building it and running the project on your Maestro.

7. Follow the procedure in section 3.1 Creating a New Maestro Project to create a new Maestro project.
8. Build your "HelloWorld" project as described in section 3.3 Building a Project.
   At this point, you have a file with the extension **.gexe** (for Gold Maestro) or **.pexe** (for Platinum Maestro). Now you can download this file to your Maestro for debug.
9. Follow the instructions in paragraph 3.8 Debugging a Project to perform this.
   Now you have a program running on the Maestro in debug mode. You can set a breakpoint, run the program systematically and use other debugging commands.

Chapter 2: Perspectives

Each Workbench window contains one or more perspectives. A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of functions oriented to accomplishing a specific task or work with specific resources. For example, the C/C++ perspective combines views that you would commonly use while editing C/C++ source files, while the Debug perspective contains the views that you would use while debugging C/C++ programs. As you work in the Workbench, you will probably switch perspectives frequently.

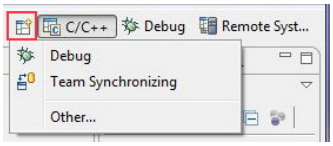
The perspective controls appear in certain menus and toolbars. They define visible *action sets*. The perspectives relevant for work with the Maestro are:

- C/C++
- Debug
- Remote System Explorer

To select which perspective to view:

1. Click the button indicating the perspective you want 

.

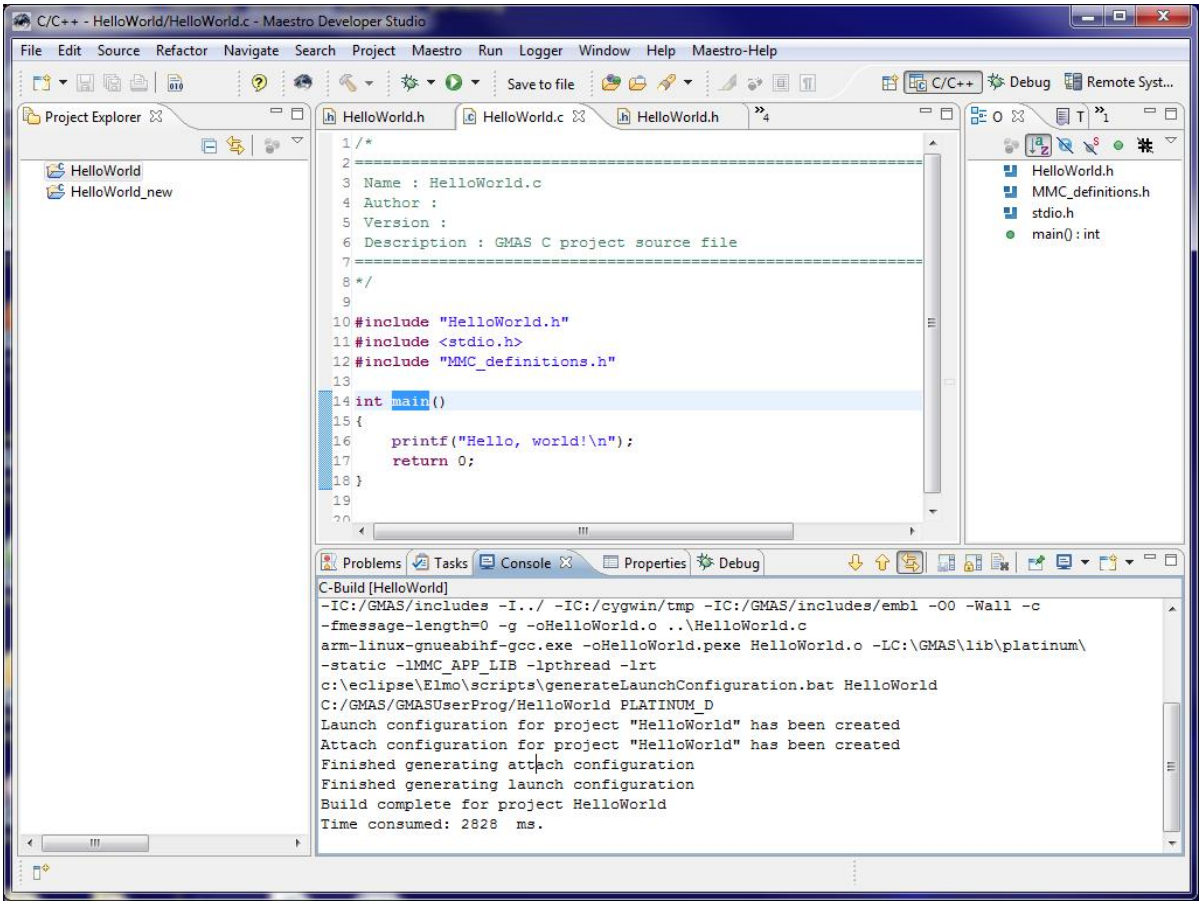
   Or, alternatively, click **Open Perspective** 

button to the left of the perspective indicator to open a selection of perspectives. Click **Other** to open the **Open Perspective** window.
2. Select the perspective you want.

   Note: The Open Perspective window can also be opened from the **Window** menu (**Window** -> **Open Perspective**).

2.1 The C/C++ Perspective

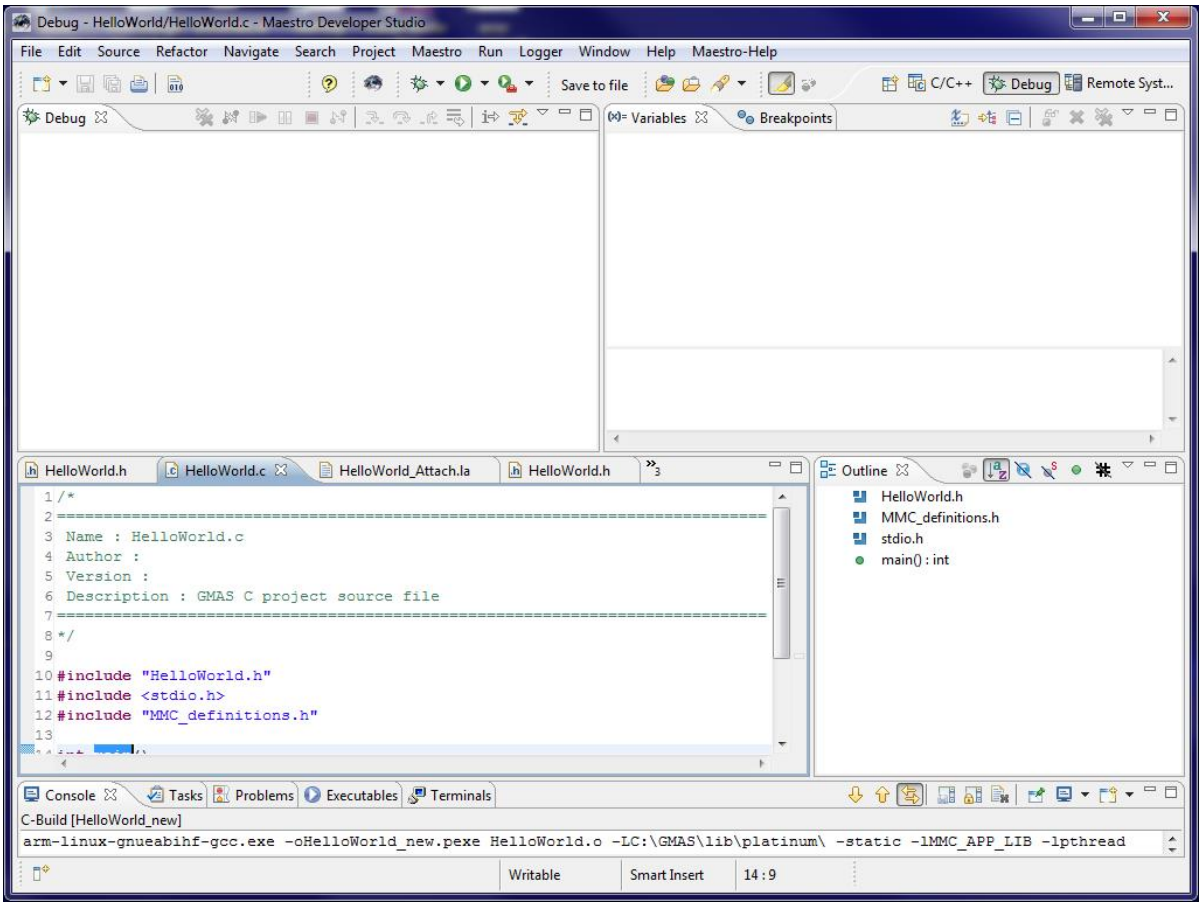
This is the perspective you use to write code, compile, fix errors, and so on.



This perspective is tuned for working with C/C++ projects. By default, it consists of an editor area and the following views:

- Project Explorer
- Problems
- Tasks
- Console
- Properties
- Debug

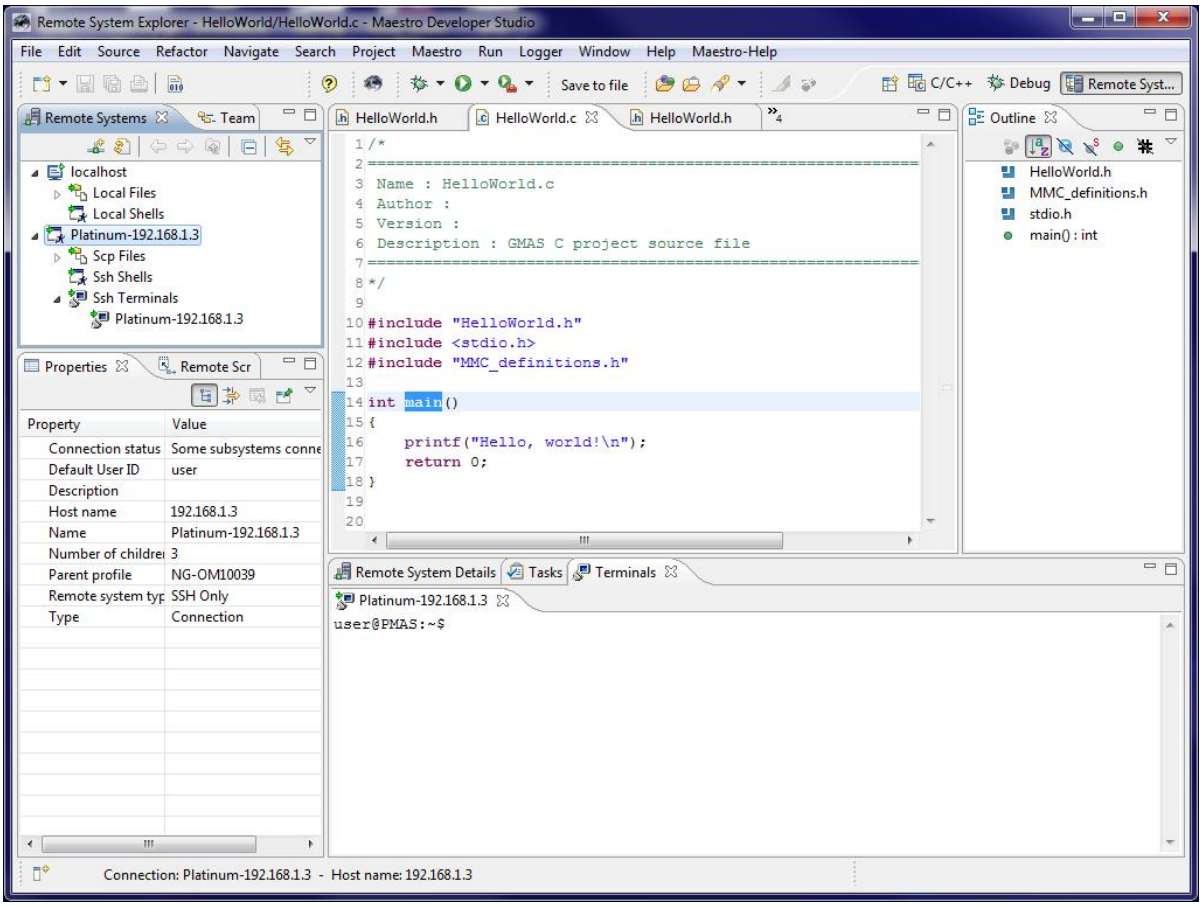
2.2 The Debug Perspective



This perspective is tuned for debugging your C/C++ program. By default, it includes an editor area and the following views:

- Debug
- Variables
- Breakpoints
- Expressions
- Outline
- Console
- Tasks
- Problems
- Executables
- Terminals

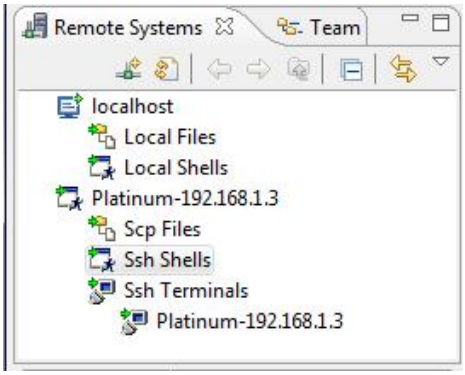
2.3 The Remote System Explorer Perspective



This perspective is tuned for working with the maestro. By default, it includes an editor area and the following views:

- Remote Systems explorer
- Outline
- Properties
- Remote scratchpad
- Remote system details
- Tasks
- Terminals

2.3.1 The Remote Systems Explorer

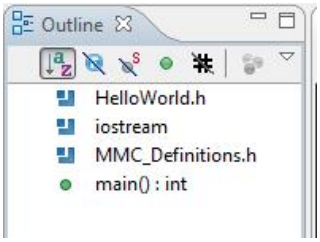


This view shows the connected Maestro devices with their files and folders and allows you to launch the Ssh terminal or shell.

1. Expand the **Scp Files** branch to see the files and folders that are in the Maestro.
2. Use right-click for shell and terminal activation.

2.3.2 Outline View

This view displays an outline of the structured file that is currently open in the editor area and lists structural elements.



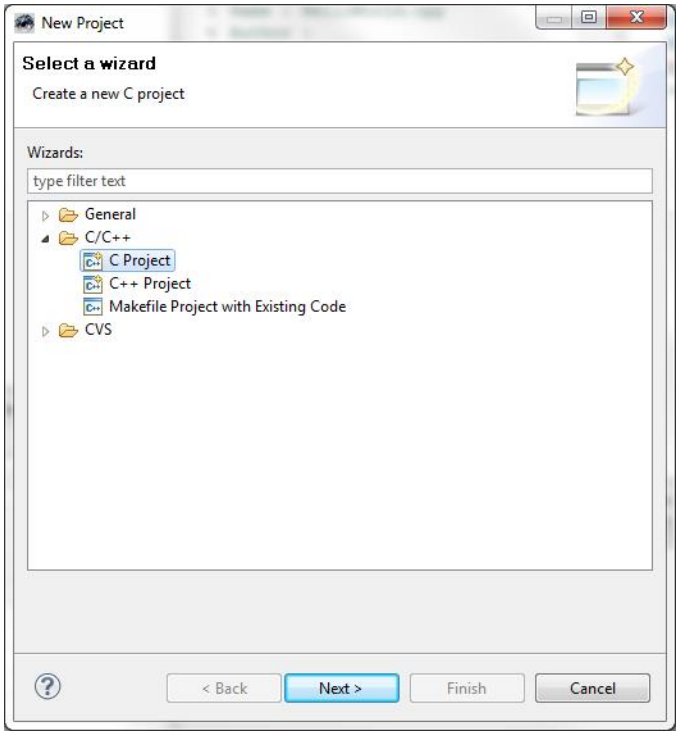
The example above is for the “HelloWorld” project. It shows the Include files and the Main function.

Chapter 3: Managing Projects

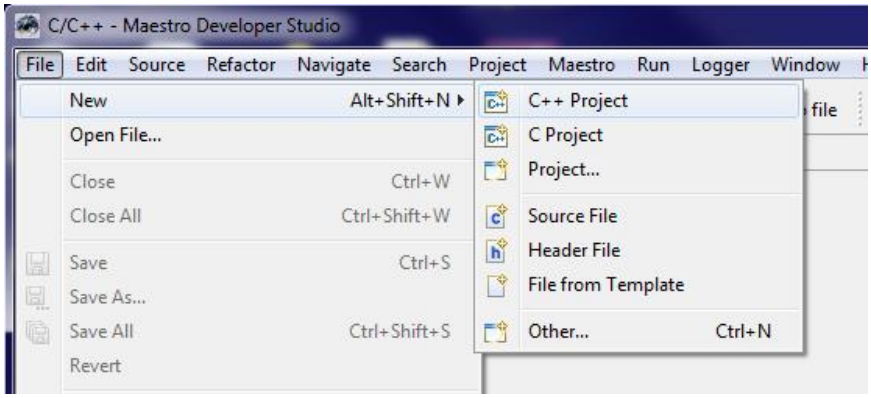
3.1 Creating a New Maestro Project

To create a new Maestro Project:

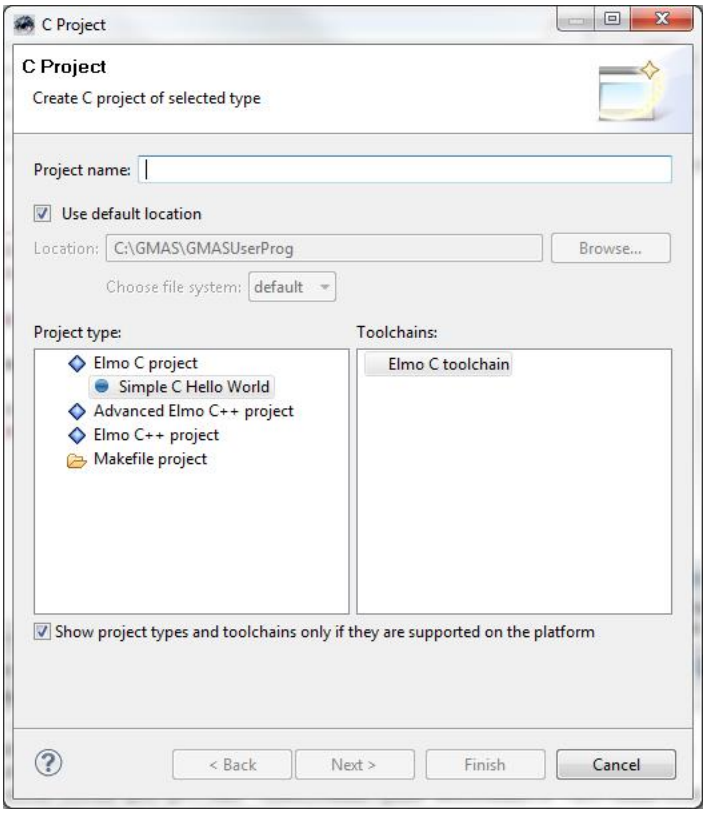
1. From the **File** menu select **New > Project**. The New Project window opens.



Alternatively in the **C/C++** perspective select **File-> New-> C/C++ Project**.



The following Project dialog opens.

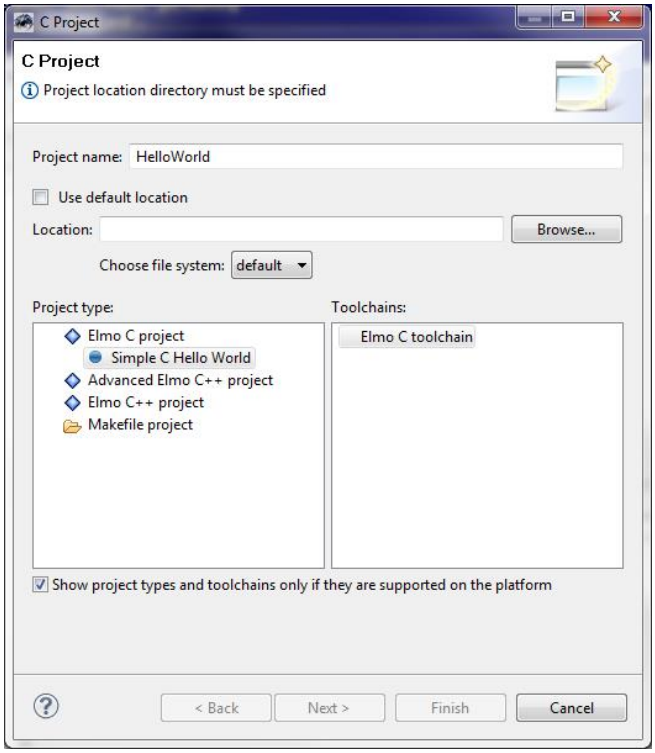


2. Select the required project type. If you want to base your project on an available template, select **Simple C Hello World**. These templates and useful examples include the necessary elements of a Maestro project. The toolchain is automatically selected according to the project type.
3. Enter the project name.

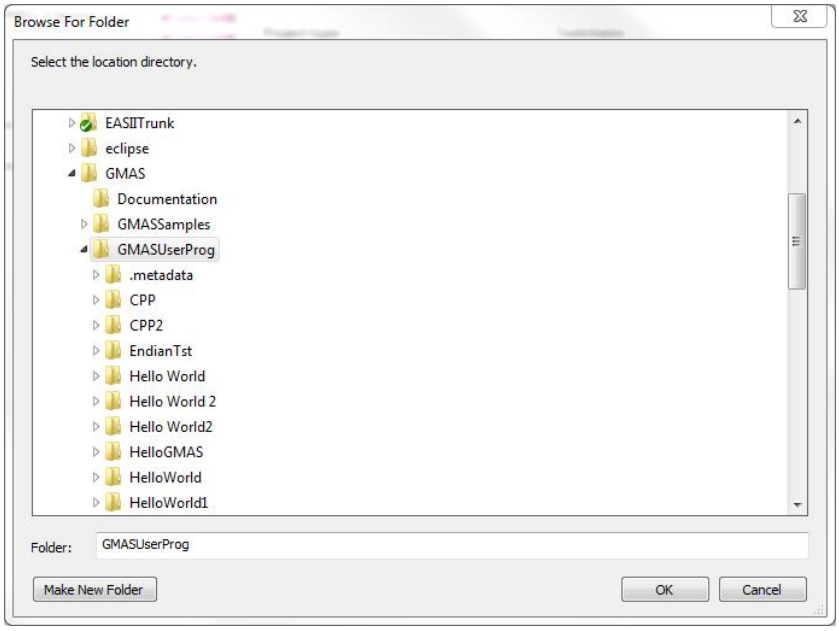
Note: Do not add spaces to the project name.

4. (Optional) The default setting of this dialog box shows projects in the default location for Elmo's Maestro. It is recommended that you select one of the projects displayed under **GMASUserProg**. To select another location:

   a. Clear the **Use default location** checkbox.



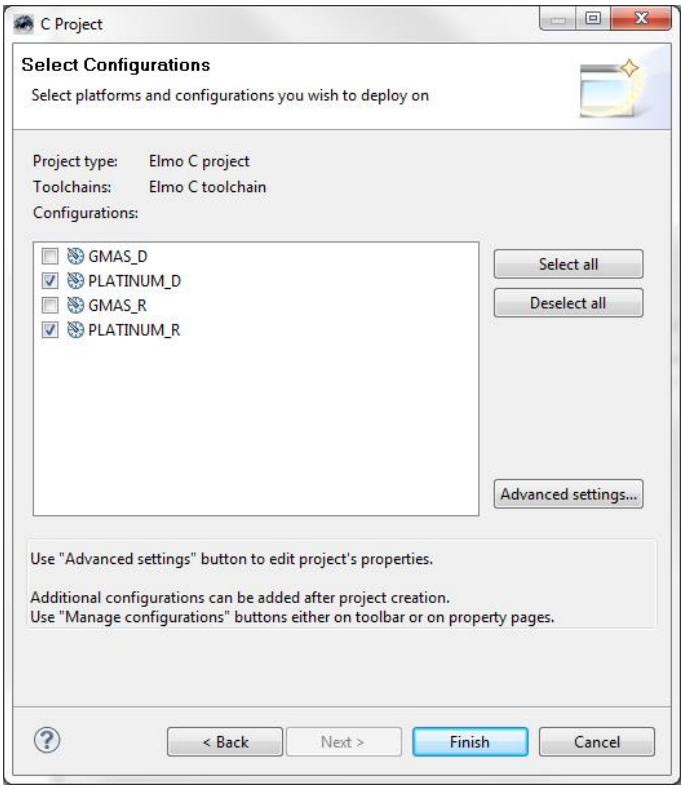
   b. Click **Browse** and browse to the appropriate folder containing the project.



   c. Click **OK**. The Browse window closes and returns you to the project window.

   Alternatively, select and copy a location from Windows Explorer and paste the location into the **Location** field.

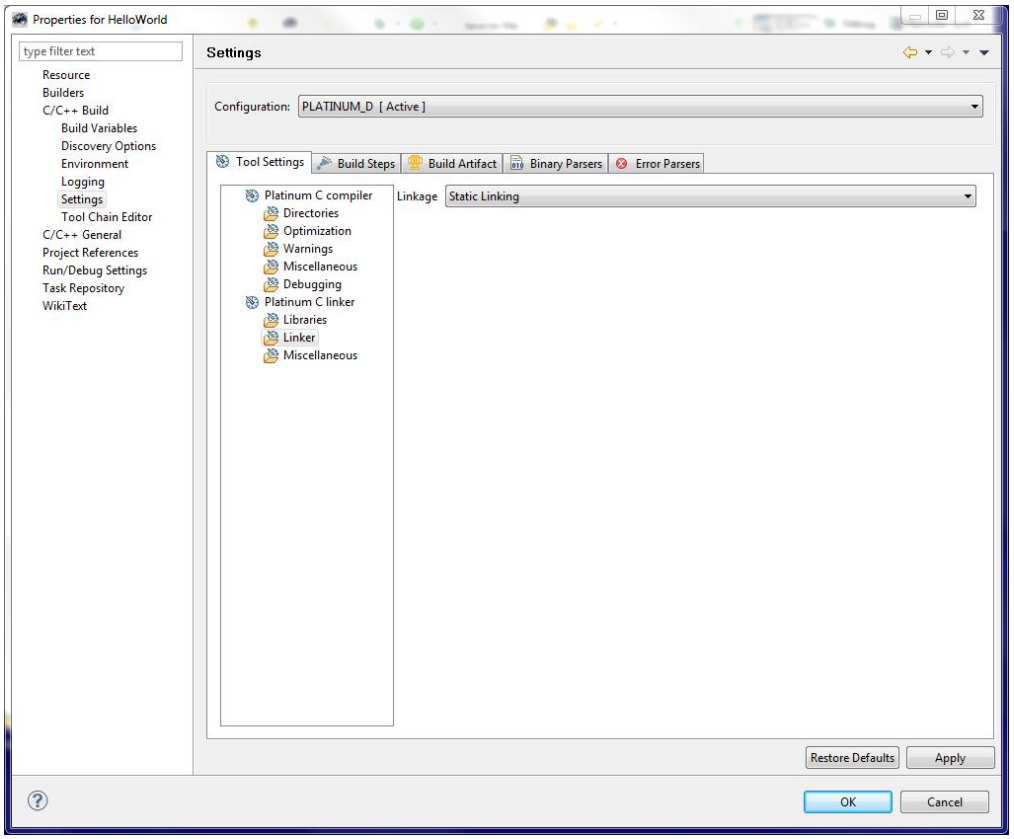
5. Click **Next**. The Select Configurations window opens.



6. Select the required configurations to apply.

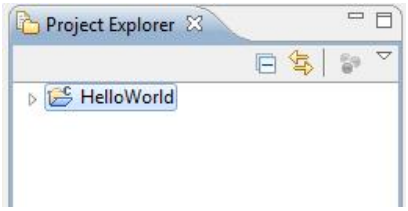
. **Note:** By default, Eclipse generates both debug and release versions when building a project.

7. Click **Advanced settings…** The Properties window opens.

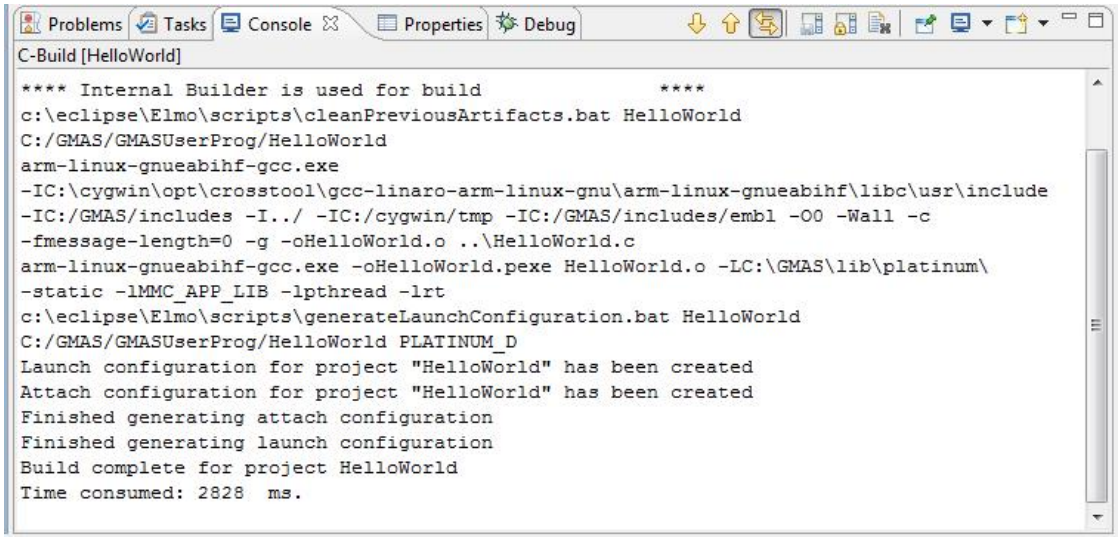


8. Edit the configuration properties as necessary. Then click **Apply** and **OK**.

9. To implement the sample project in this manual, click **Finish** in the Select Configurations dialog box. The new project now appears in the project explorer.



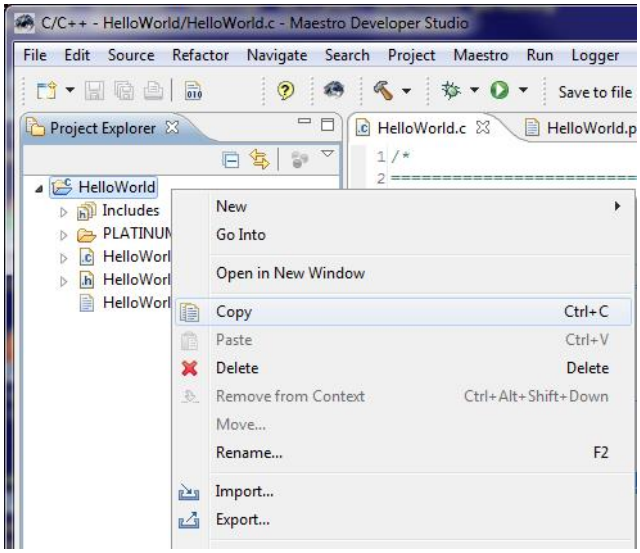
After clicking **Build** button, the **Console** tab in the bottom pane contains the build details.



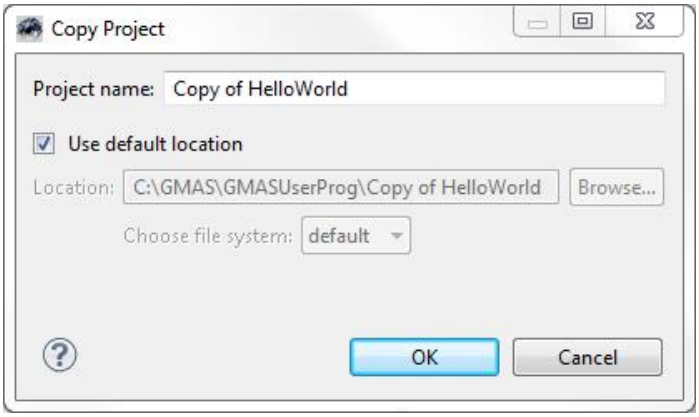
3.1.1 Duplicating a Maestro Project

To make a copy of an existing sample project or template without changing the original file:

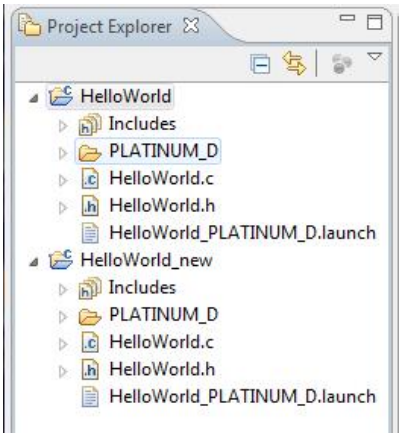
1. In the Project Explorer, right-click the project you want to duplicate and select **Copy**.



2. Paste the project using either **CTRL + V** or right-click the Project Explorer and select **Paste**. A dialog box opens with a default name for your new project.



3. Rename your project without using spaces in the new project name and click **OK**. In this example, we will change **HelloWorld** to **HelloWorld_new**. The new project now appears in the Project Explorer. Your project is automatically built.

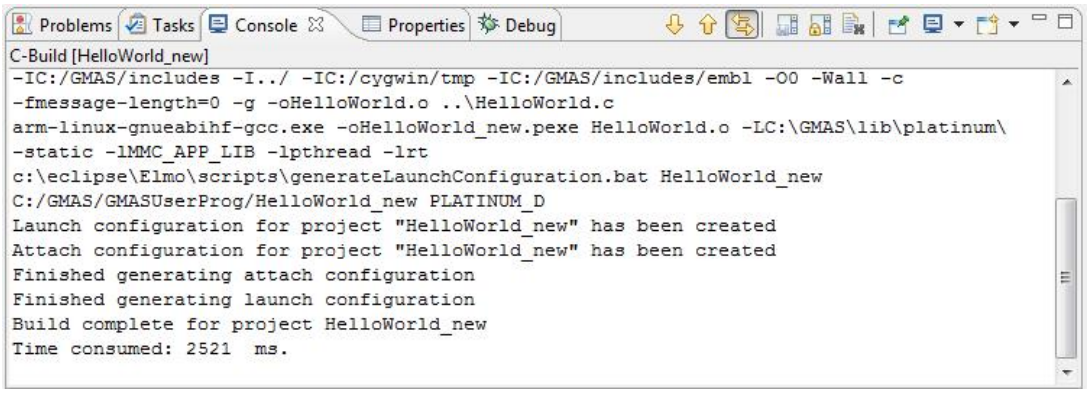


3.1.2 Branching From an Existing Project

If you want to create your own project, you should use one of the example projects or templates and add your own code to them. The projects that are provided with MDS include all the settings (environment variables, compilation flags etc.) that are required for your program to run properly on the Maestro.

To start your own project:

1. To start your own project based on a sample project, duplicate and rename an existing project as described in 3.1.1 Duplicating a Maestro Project. For this example, we are duplicating the project **HelloWorld** and renaming it **HelloWorld_new**.
2. In the **HelloWorld.c** file, make the required changes.
3. Click the **Build**  button to rebuild the project. Check the **Console** tab to see the output of the process.

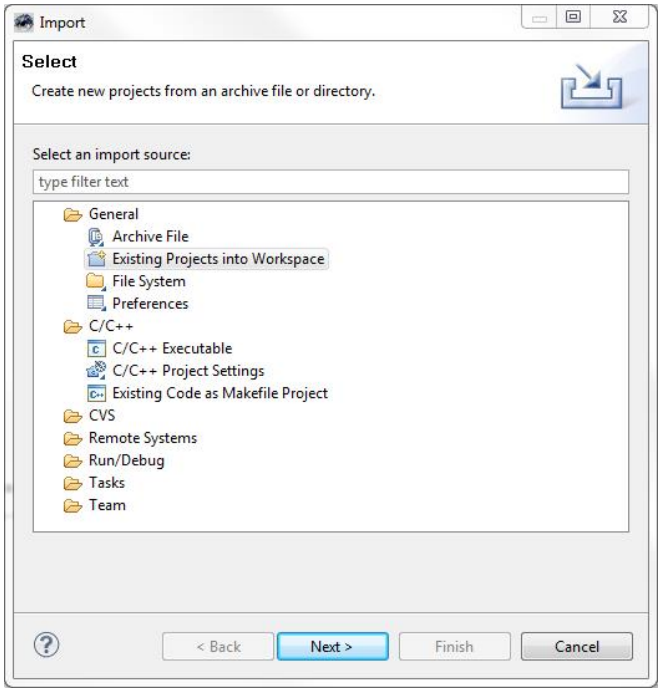


3.1.3 Importing a Project

You can work with a project that was received from another source only if it was created in Maestro Developer Studio. A project that was created in a different development environment will most likely result in errors.

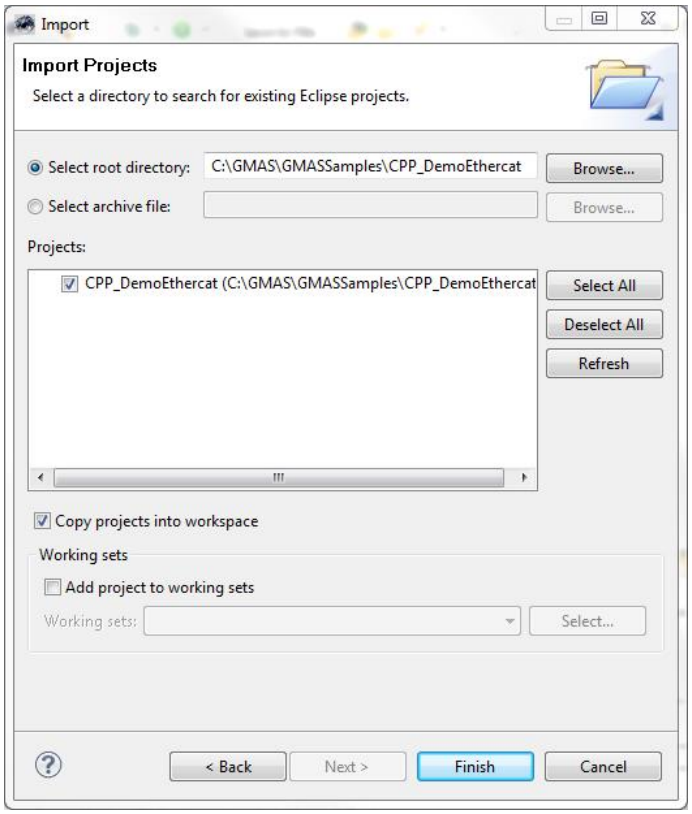
To open a project from a different source in Maestro Developer Studio:

1. In the **C/C++** perspective, right-click inside the **Project Explorer** tab and select **Import**. The Import window opens.



2. Select **Existing Projects into Workspace** and click **Next**.

3. Browse to the directory that includes the project to be imported.
4. Select the **Copy Project into Workspace** checkbox.

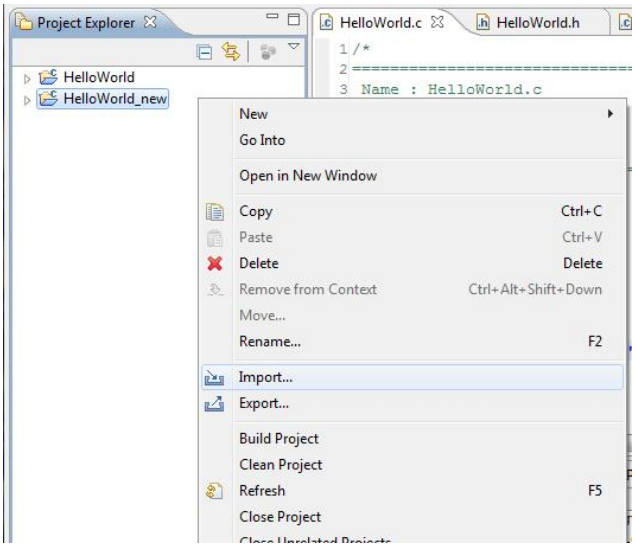


5. Click **Finish**.

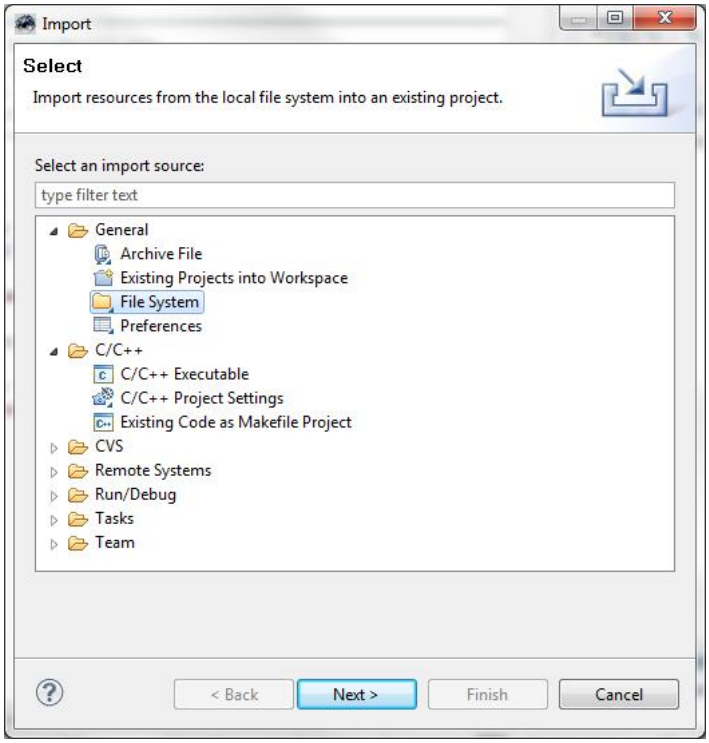
3.1.4 Importing a Source File

To import a source file into a project:

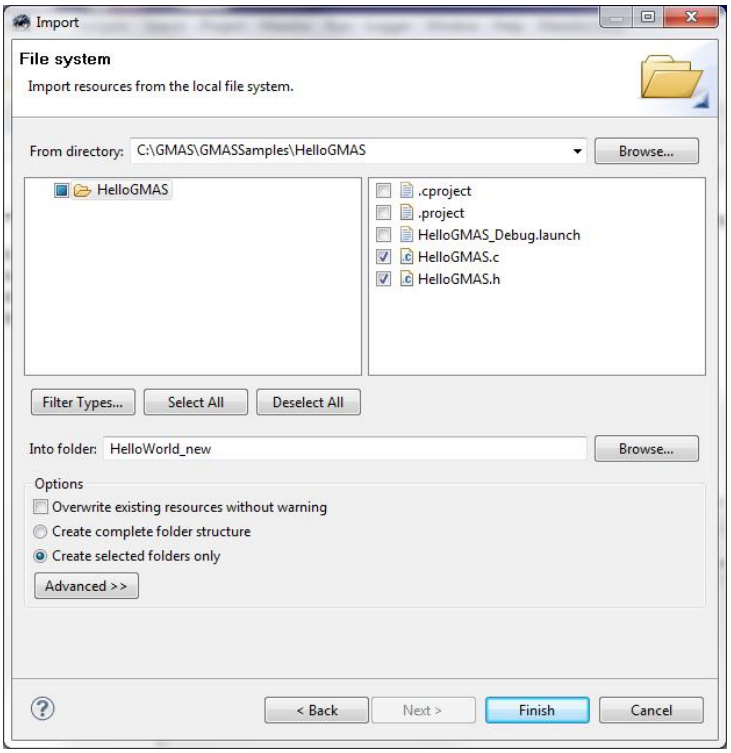
1. If you have a source file that you want to import into your project, right-click the name of the project in the **Project Explorer** and select **Import**.



2. In the window that opens select **File System** and click **Next**.



3. Browse to find the folder that includes your source files.



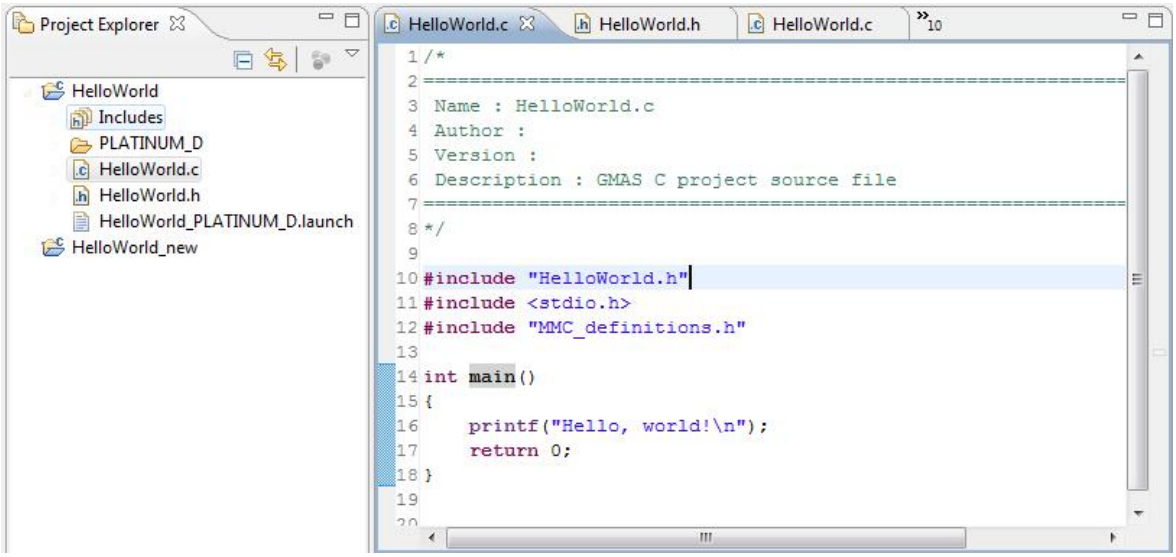
4. From the list of files select only files with the extensions **.c**, **.h** or **.cpp**. Files of other types may not open correctly.
5. Click **Finish**.

3.2 Editing Files

Maestro Development Studio supports all the common editing actions (Copy, paste, search and so forth).

To open a file in the editing area:

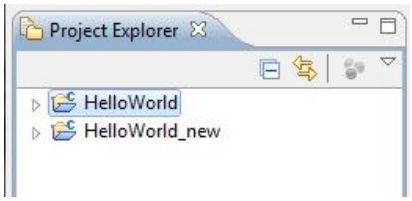
1. Expand the project in the Project Explorer and double-click the file name.
2. Open the **HelloWorld.c** file.



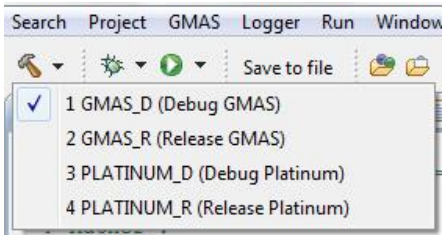
3. Make the required changes.

3.3 Building a Project

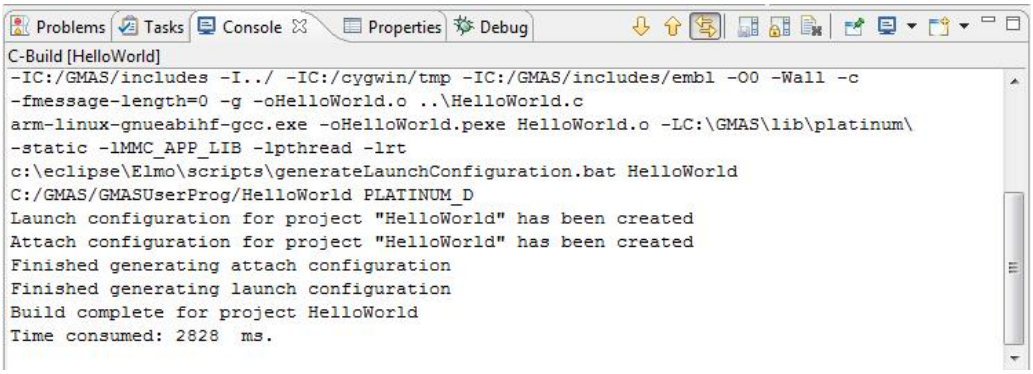
1. In the Project Explorer, select the project you want to build.



2. Click the arrow to the right of the **Build**  button and select the required build configuration.



Look at the **Console** tab to can see the output of the process. Scroll down in this tab to see that a file with the extension **.gexe** (for Gold Maestro) or **.pexe** (for Platinum Maestro) was generated and the selected configurations were created.



3. Make a small change in the example program to justify a new build. For example, change the text to be printed from “Hello, world” to “Hello, Maestro”.

4. Click the **Build** button  to build the project again and display the results in the **Console** tab.

3.4 Multiple Hardware Support

MDS supports the following hardware types:

- Gold Maestro
- Platinum Maestro

Projects from both Gold Maestro and Platinum Maestro are created from templates based on Elmo samples or imported to the workspace. Each project can be configured for either Gold Maestro or Platinum Maestro.

Note: You can connect to two different hardware targets at one time as long as each one has a different IP address.

3.5 Connecting to a Maestro Target

To run or debug an executable file:
To run or debug an executable file on a Maestro target, you first need to connect to the target. Make sure the Maestro is connected to your PC as directed in the Gold Maestro/Platinum Maestro Installation Guide.

Note: Working with MDS via proxy server is not supported. Make sure that your PC does not use a proxy server (for example, verify that your internet browser is not configured to use proxy).

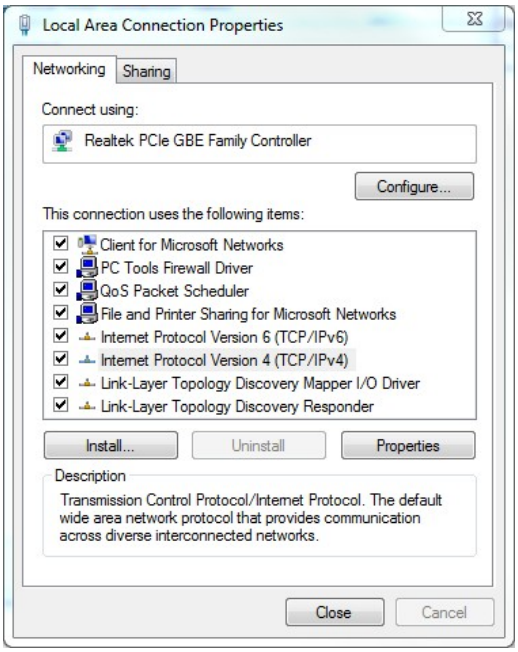
3.5.1 Peer to Peer Connection

Note: Skip this section if you are connected to the Maestro through a network. Continue directly with section 3.5.2 Creating a New Connection (first time only).

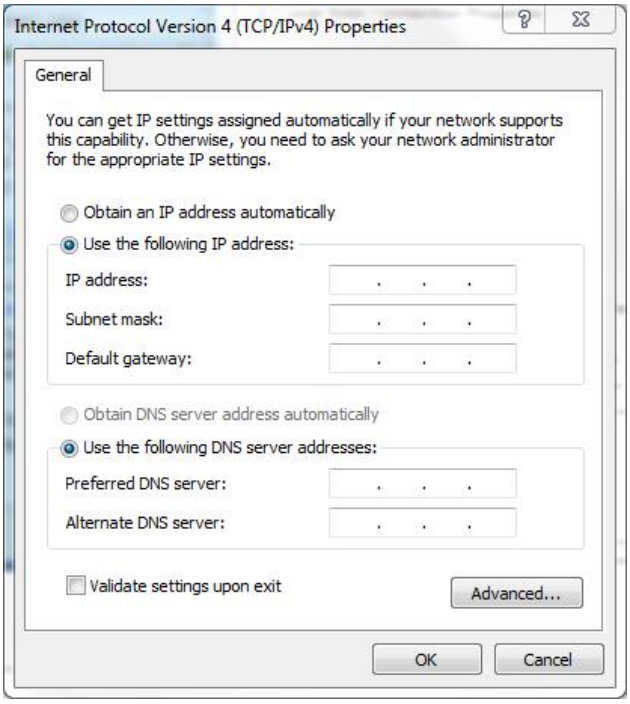
Peer to peer connection to Maestro is useful when the network presents problems. For example, if your network permissions are limited and some of your actions are limited by a firewall. The peer to peer connection does not apply to the MDS, but is also useful, for example, for the EAS application, Modbus connections, and so on.

To connect to a Maestro via a Peer to Peer network:

1. Connect the Maestro to your PC using either a standard or crossed network cable.
2. Select Start -> **Control Panel**.
3. From the window that opens, select **Network and Internet** -> **Network and Sharing Center** -> and then **Local Area Connection**.
4. In the **General** tab click **Properties**. Opens the Local Area Connection Properties dialog box.

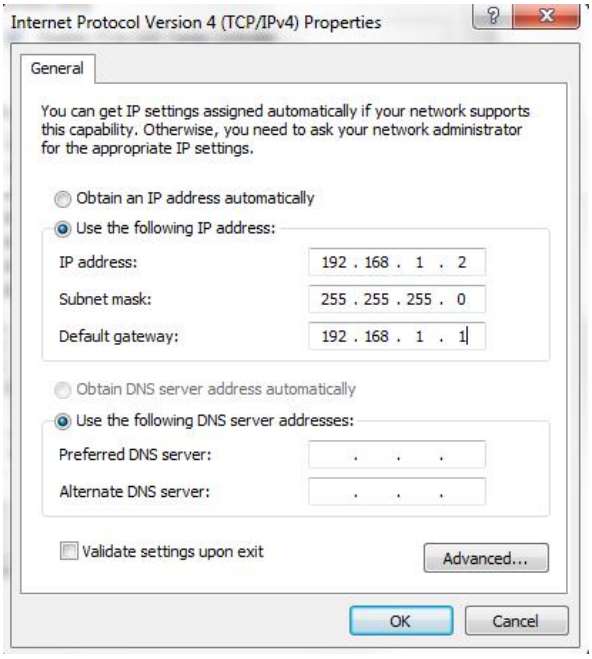


5. Select **Internet Protocol Version** (Internet Protocol Version 4 in Windows 7) and then click **Properties** to display the following dialog box (or similar, depending on your operating system).



6. Select **Use the following IP address** and enter the IP settings as described:

IP Address	The default Maestro IP address is set to 192.168.1.3. The IP address you set must be in the same segment but different (in the example it ends with 2 instead of 3). If you or someone else has changed the factory settings, apply the same logic to your new settings.
Subnet Mask	255.255.255.0
Default Gateway	Must start with 192. 168.1.X, the last digit is dependent on your network configuration.

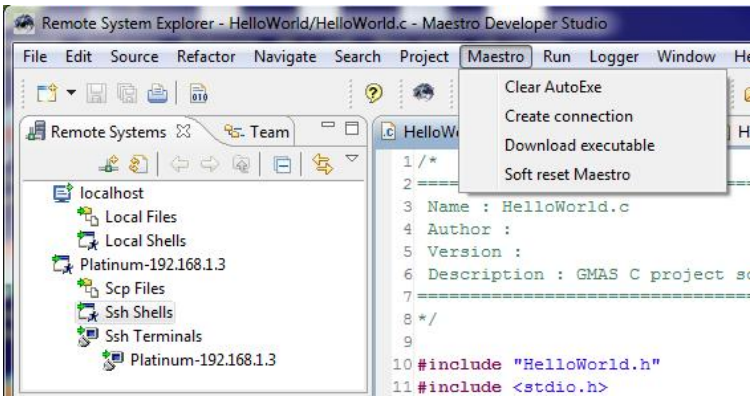


7. Click **OK**.

3.5.2 Creating a New Connection (first time only)

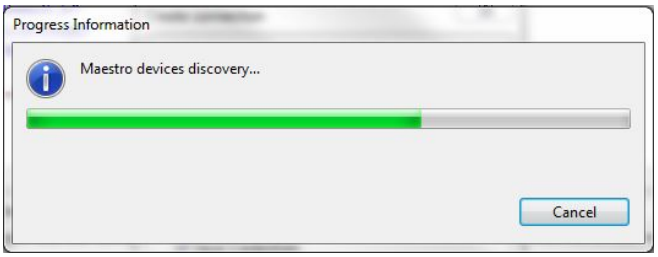
This section describes how to create a new Maestro connection.

1. From the main menu, select **Maestro-> Create connection.**



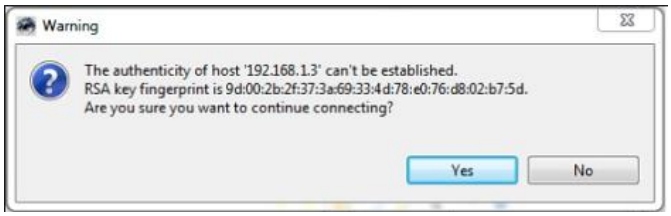
The Create connection dialog opens.

2. If you do not know the IP address of the Maestro you want to connect to, click **Discover** to let MDS find all the Maestro units connected to your PC directly or through a network.



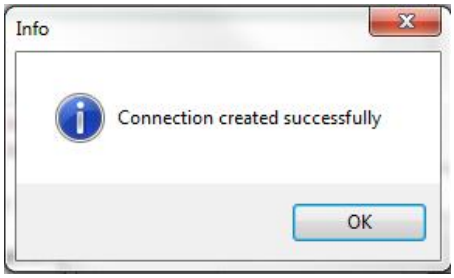
3. Select the Maestro you wish to connect from the list and click **Connect**. The **Use Default Credentials** and **Save Credentials** are checked by default. Do not change the defaults.

A warning may appear the first time you connect.



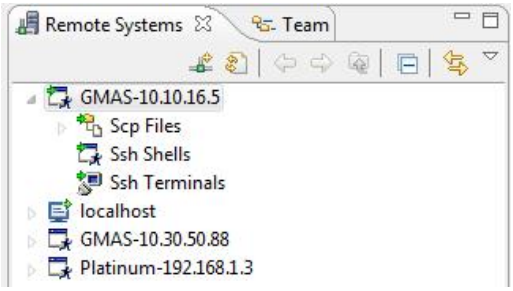
Click **Yes** to close the warning and continue.

4. When connected, the following window informs you that you are connected.



5. Click the **Remote System Explorer**  icon.

6. The **Remote Systems** perspective tab opens displaying the new entity related to the connection that is created. Maestro projects for both Gold and Platinum are supported in MDS. Both hardware types are displayed in the Remote System perspective.

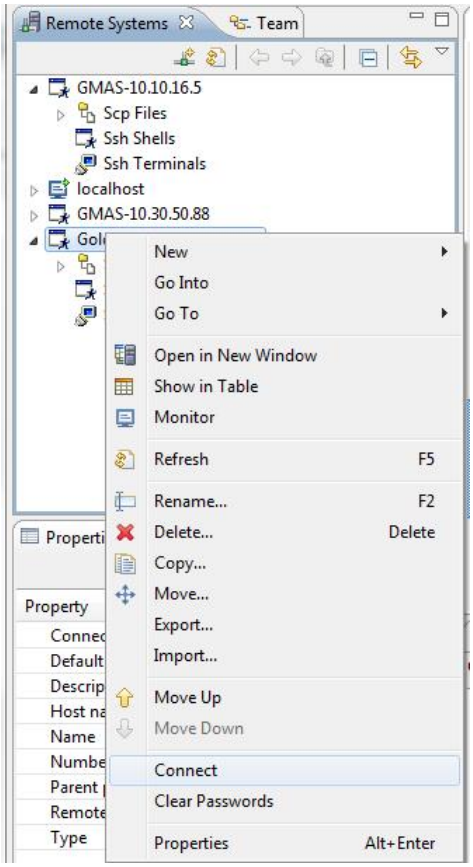


3.5.3 Starting a Communication Session

If the communication connection is created but is not active, then perform the following procedure.

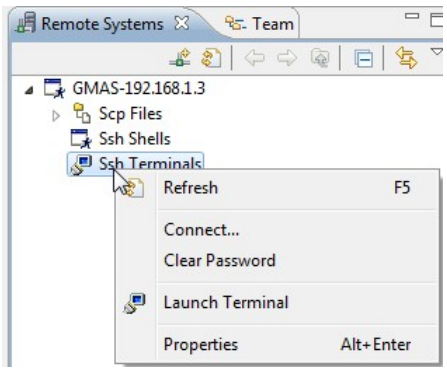
To activate the connection to the Maestro:

1. Right-click the name of the connection and select **Connect.** The icons next to the target and the items under it change to indicate that a connection is established.



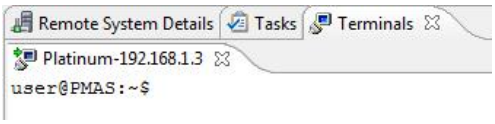
The terminal is still inactive.

2. To activate the terminal right click on the **Ssh Terminals** and select **Launch Terminal**.



3. If the password is locked, the SSH Terminal opens a window requesting the password. Enter the password and click **OK**.

The terminal is activated.



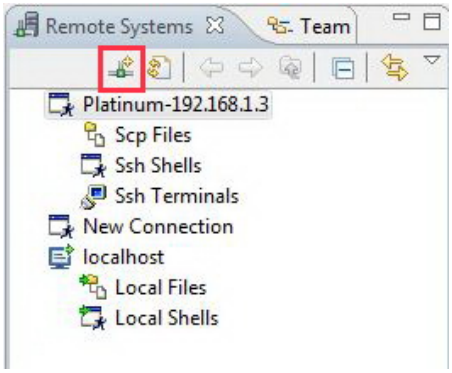
You can disconnect the Maestro using **Disconnect** from the same context menu. When the MDS session is terminated, the connection is automatically disconnected.

3.5.4 Connecting to a Remote System

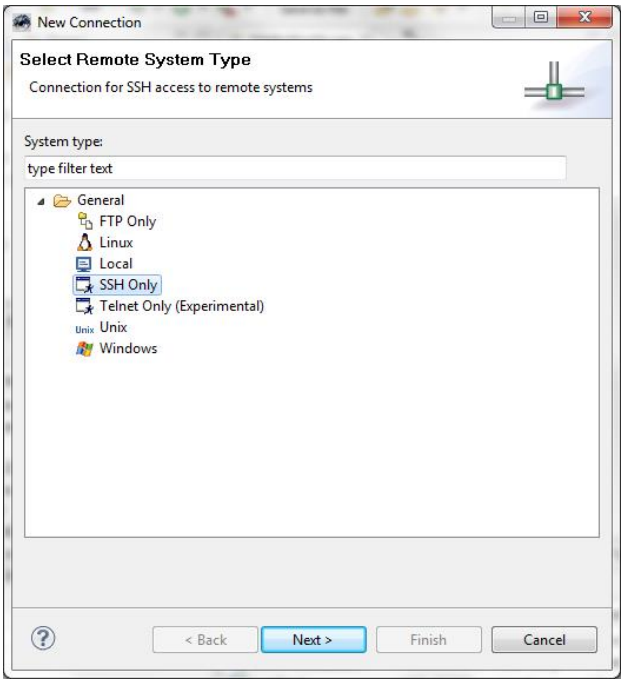
MDS offers the option to connect to a remote system, and to perform all the operations of Windows Explorer.

To connect to a remote system:

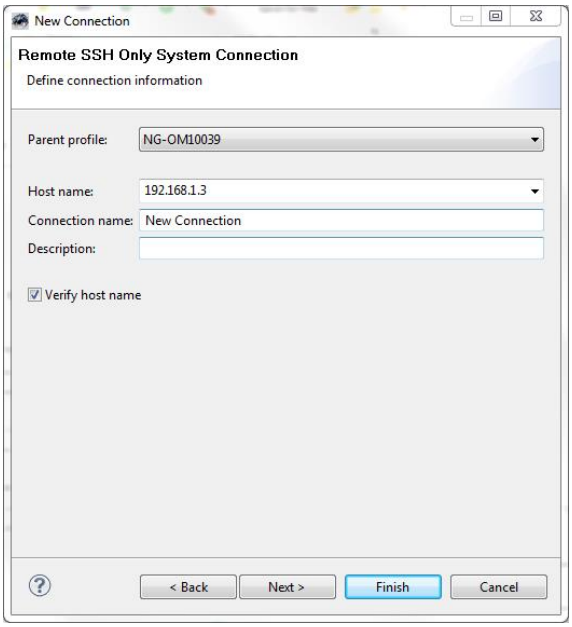
1. In the Remote Systems window, click the network icon.



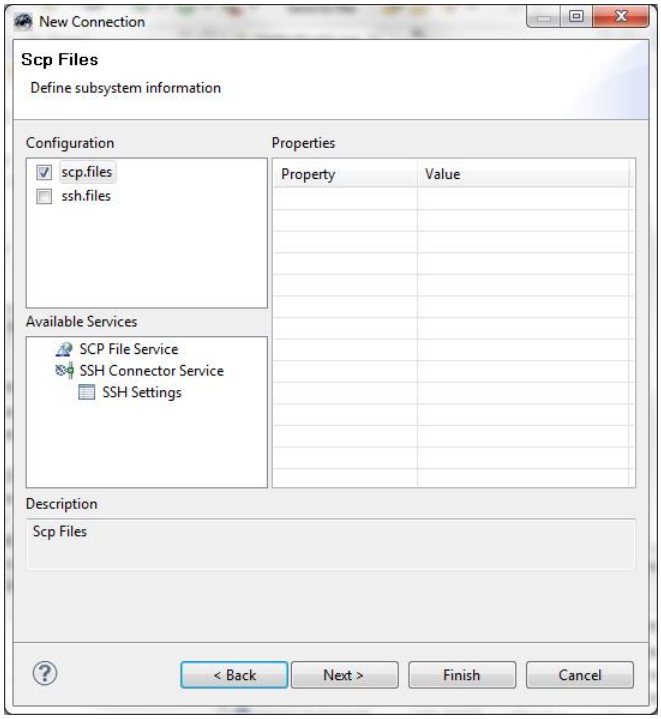
The New Connection window appears.



2. Select a remote connection, e.g. **SSH Only**, and click **Next**. In this example, the Remote SSH Only System Connection window opens.

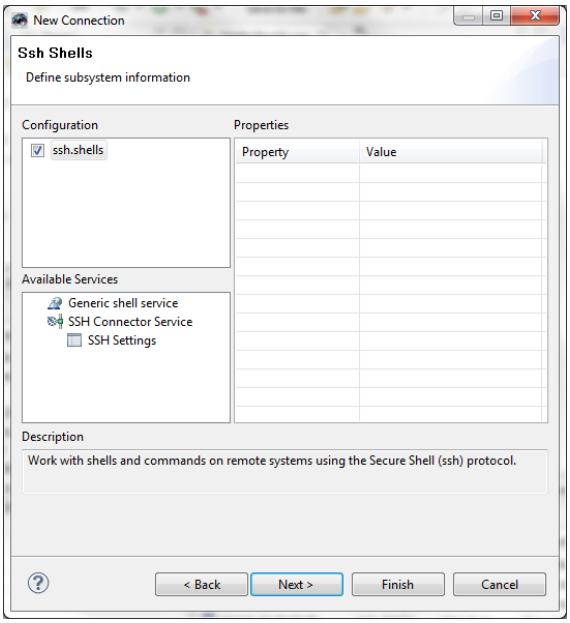


3. Enter the connection name, and click **Next**. The Scp Files window opens.

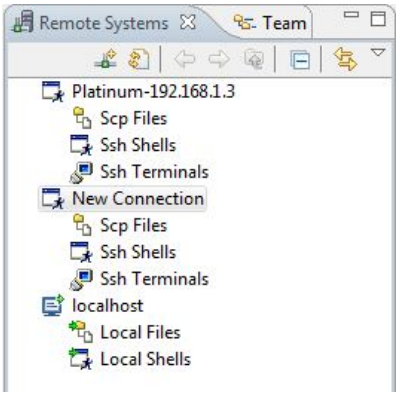


By default, no changes need to be made to this window unless more than one Configuration appears in the **Configuration** pane, or changes to the **Available Services** pane.

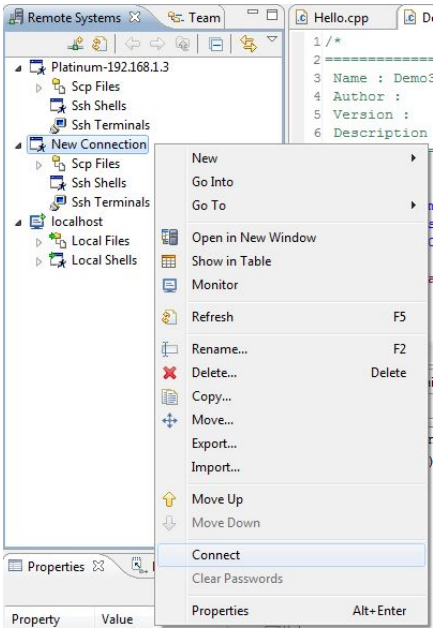
4. Click **Next**. The Ssh Shells window opens.



5. By default, no changes need to be made to this window unless other shells are available in the Configuration window. Click **Finish**. You are now connected to the local host systems.



6. In the Remote Systems window, right-click **New Connection** and click **Connect**.



7. The Enter Password dialog box opens.

8. (Optional) Enter the required password.
9. Click **OK**.

3.6 Working with Maestro

3.6.1 Terminal

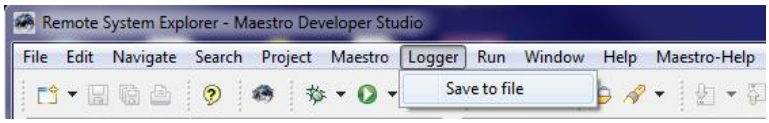
The terminal tab is a standard terminal that can be used to communicate with Maestro.

3.6.2 Logger

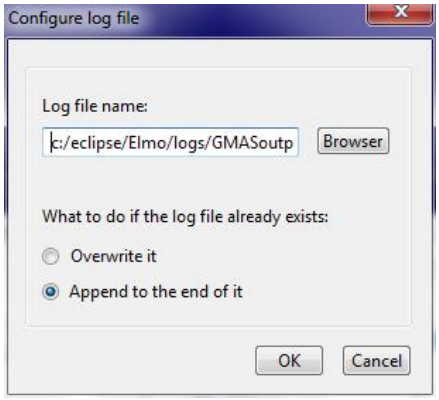
Use the logger to save the contents of the output Console window in a file. The log file will contain all the printf outputs created within the program. This will allow you to debug errors within the operation of the Maestro.

To activate the logger (inactive by default)

1. Select from the menu **Logger-> Save to File.**



The Configure log file window opens.



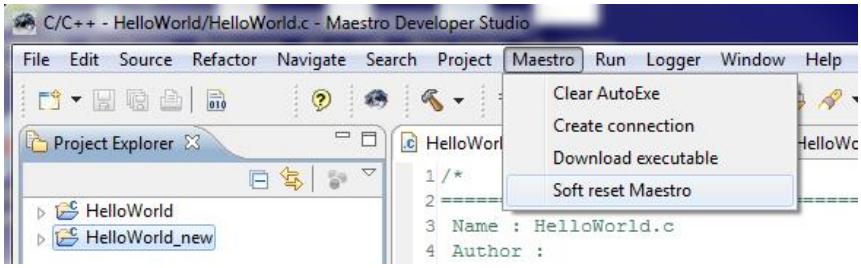
2. Either click **Browser** to select a log file name and its path or keep the default name and path of **C:\Eclipse\Elmo\logs**.
   If the log previously exits, select whether to overwrite the existing file or append to the end of it.

3.6.3 Software Reset

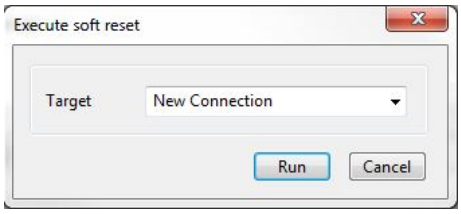
The Software reset facility, resets the Maestro, in a situation where the Maestro fails to respond, after a connection.

To perform a soft reset of the Maestro:

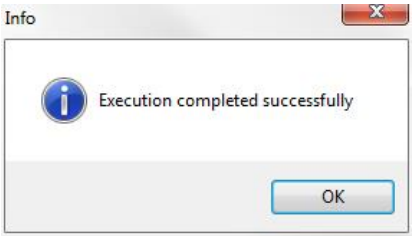
1. From the MDS main menu, select **Maestro** > **Soft reset Maestro**.



The Execute Soft reset dialog box opens.



2. If the target is not predefined, enter the Maestro IP address.
3. Click **Run**. The Info window appears displaying the following message: Execution completed successfully.



4. Click **OK** to close the window.

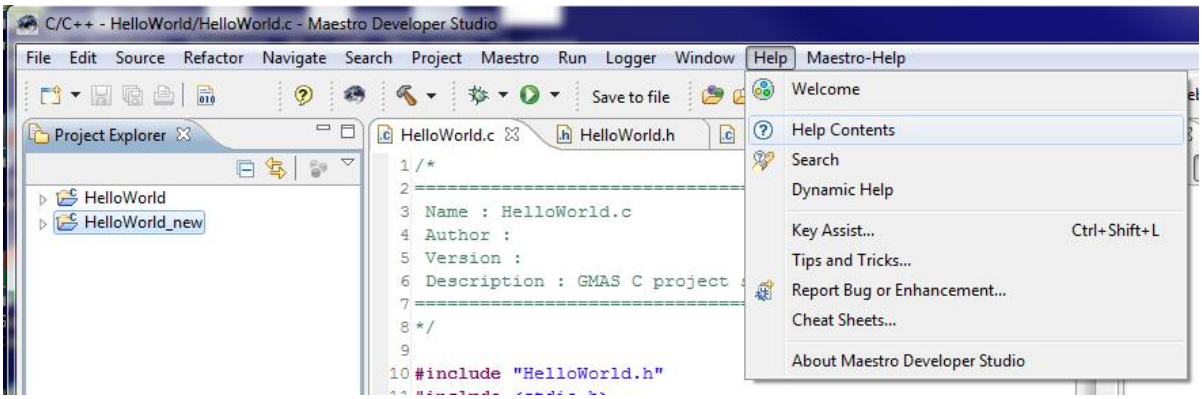
3.6.4 Help

There are two types of Help available in Eclipse:

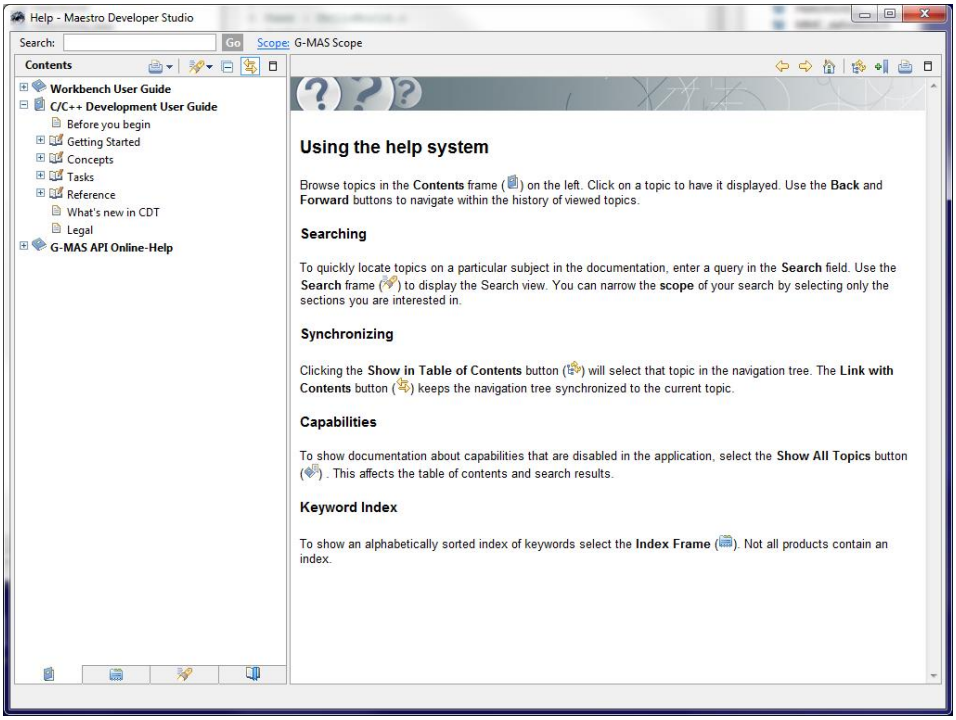
- Normal pull-down Help, which includes Maestro-Help
- Context sensitive Help which provides help on any function highlighted

To use the normal pull-down Help

1. From the **Help** menu, select **Help Contents**.

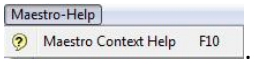


The Maestro Developer Studio Help opens.



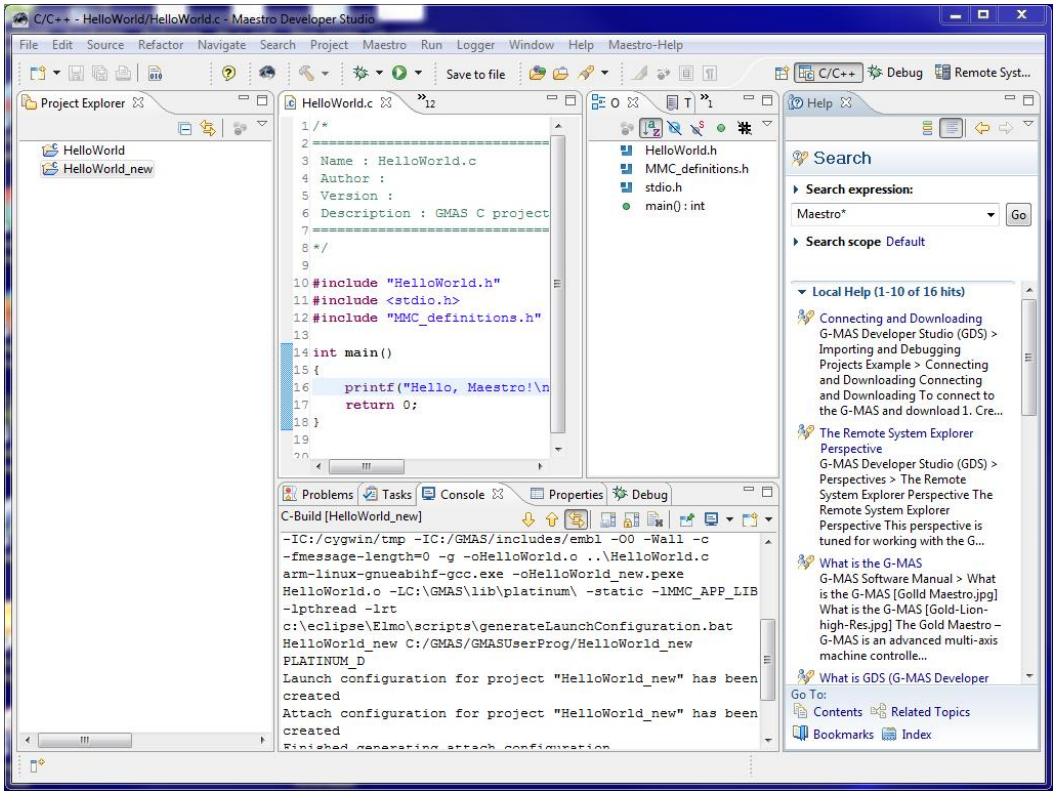
This Help form operates similar to all windows Net Help and clicking any detailed procedure, function, and so forth opens the details in the On-Line NetHelp.

To use the context sensitive Help as general Help:

1. From the **Maestro-Help** menu, select **Maestro Context Help** 

.

   Alternatively, press **F10** or click the  icon.



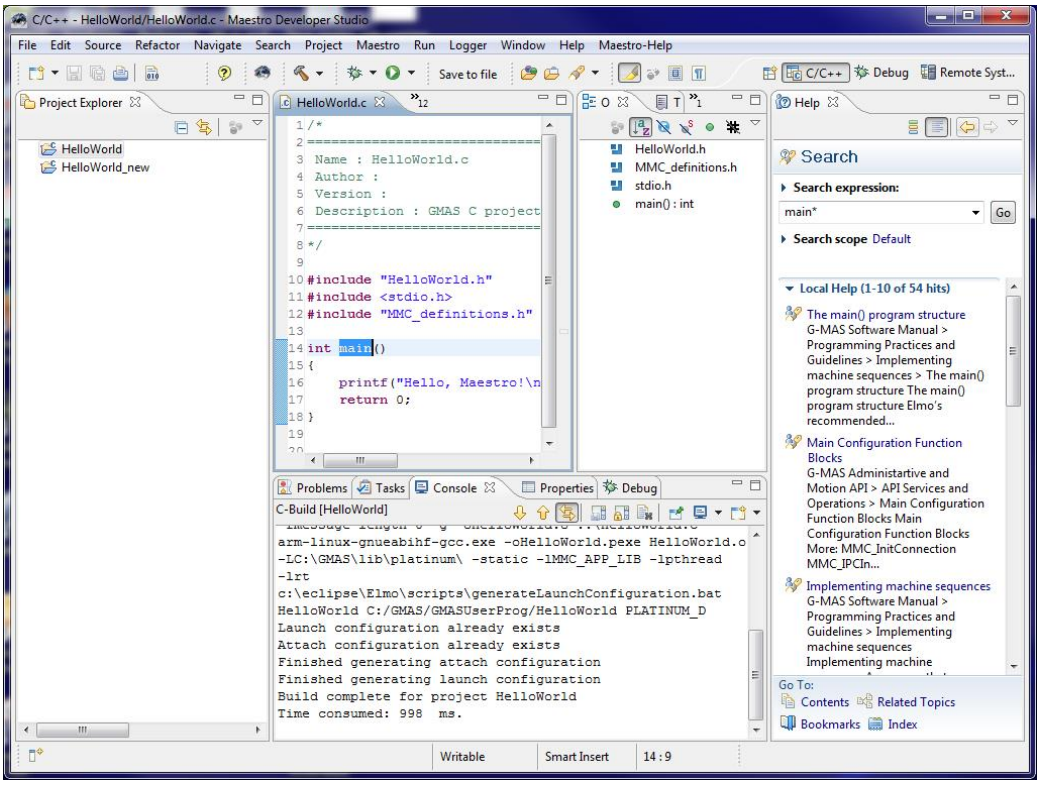
2. The Help **Search** opens. Use the **Search** to locate any procedure, function, and so on, by entering a search expression.
3. When located, click the required option heading (hyperlinked).

   The On-Line NetHelp opens with the option details selected and highlighted.

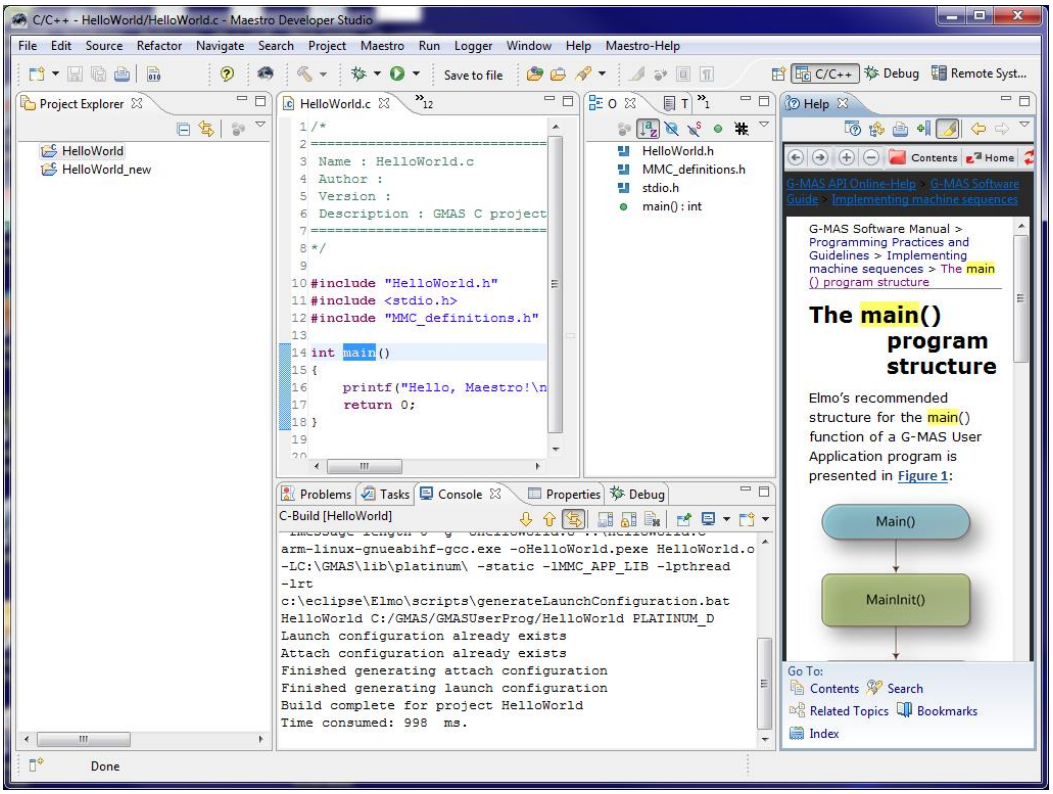
To use the context sensitive Help facility:

1. Highlight a particular function or variable to search for Help. If necessary, you can highlight part of a function or variable and insert an asterisk (*****) afterwards to locate help on all varieties of the function or variable.
2. Click the  icon, or press F10.

The Search expression opens at the right side and lists all possible options where the search expression appears.



3. Select the appropriate option and click the option heading.



The help opens with the function, variable, and other details highlighted in the Help.

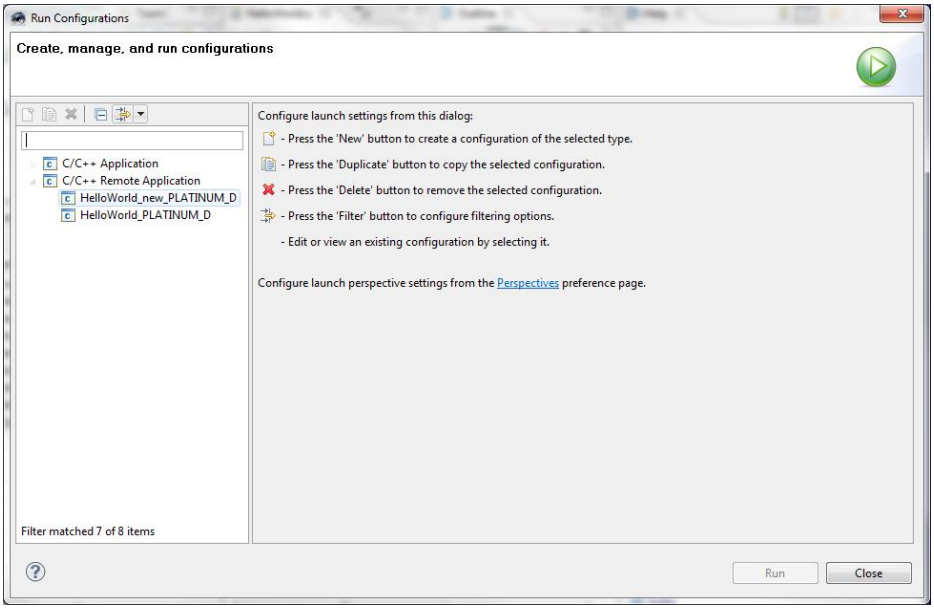
3.7 Running a Project

To run a project:

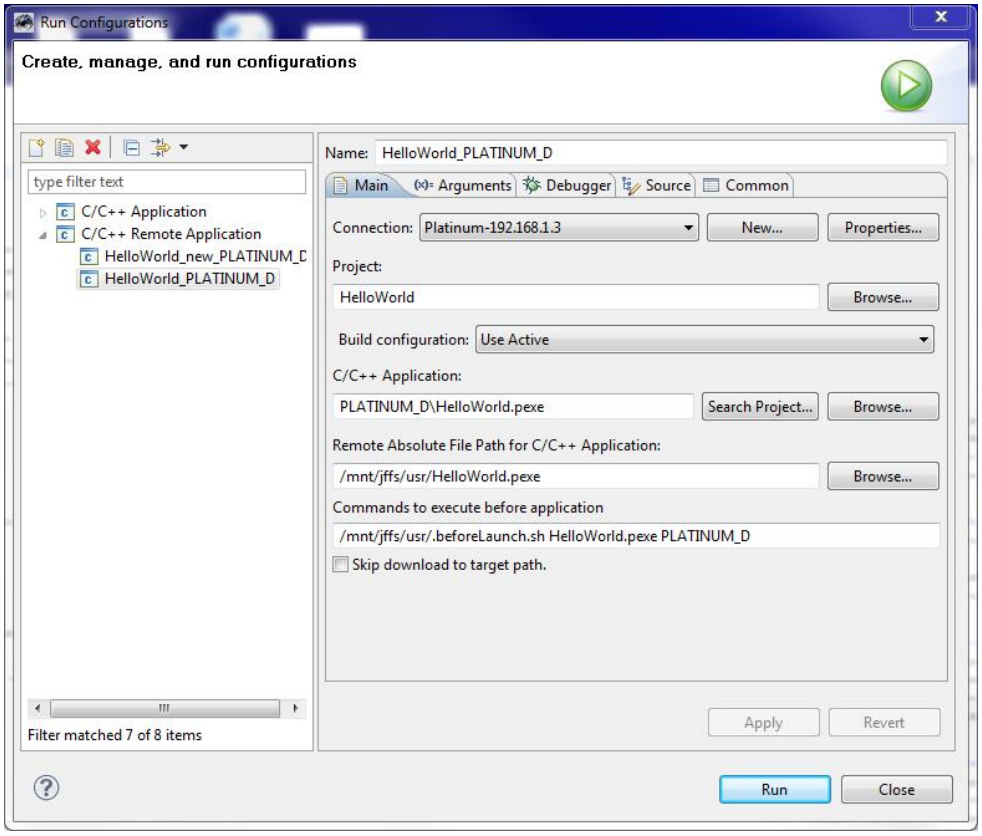
1. In the toolbar, click the **Run** button

   Or right-click the required project in the **Project Explorer** tab and select **Run As** -> **Run Configurations**.

   The Create, manager, and run configurations window appears.



2. In the left pane, select the required project.



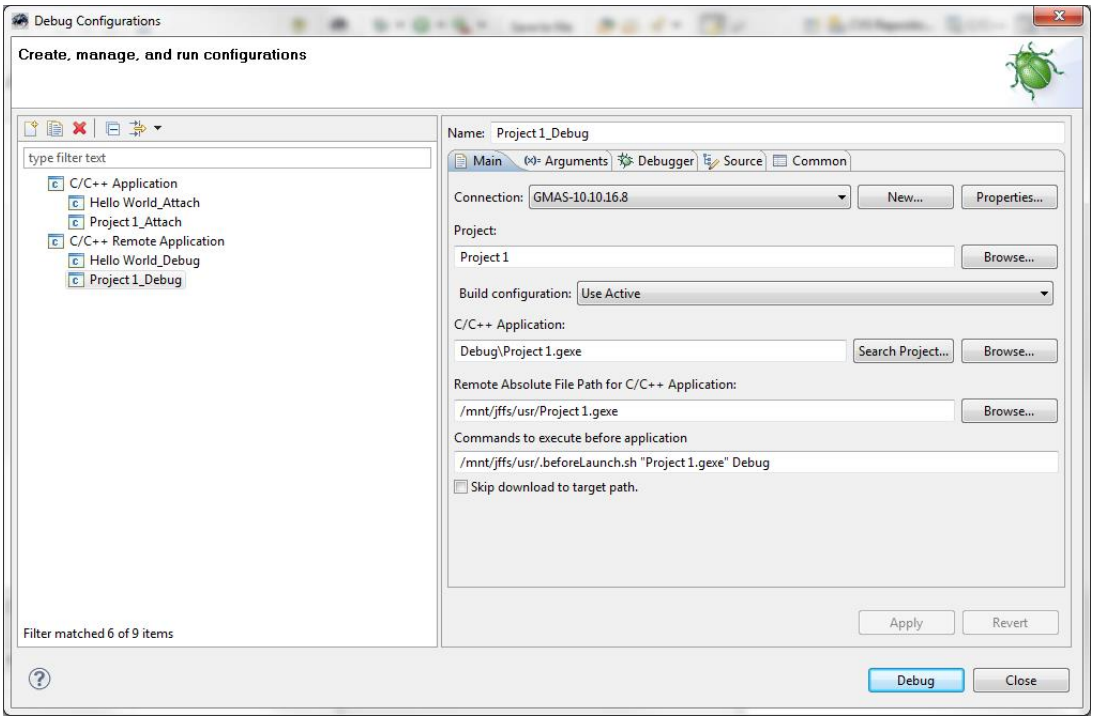
3. From the **Connection** list, make sure to select the correct hardware connection device.

Note: Before running the project, the program must be loaded to the maestro (compile the program and download the executable to the maestro (for details, see 3.8.1.1 Downloading an Executable).

The executable in the Maestro and the executable in Eclipse are the same. Both need to be loaded prior to running the project.

4. Click **Run**.

3.8 Debugging a Project



Note: You do not have to type any commands in Telnet to enable the debugger.

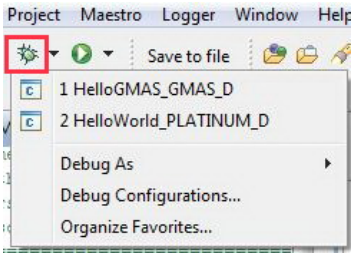
Once you click the **Debug** button, Eclipse does the following:

- Saves all files
- Cleans and compiles
- Copies binary to target (including killing any user processes running)
- Changes target binary privileges
- Executes GDBServer
- Connects the Eclipse IDE to the GDBServer on the remote target
- Enters a remote debug session on Eclipse
- Connects the output dialog

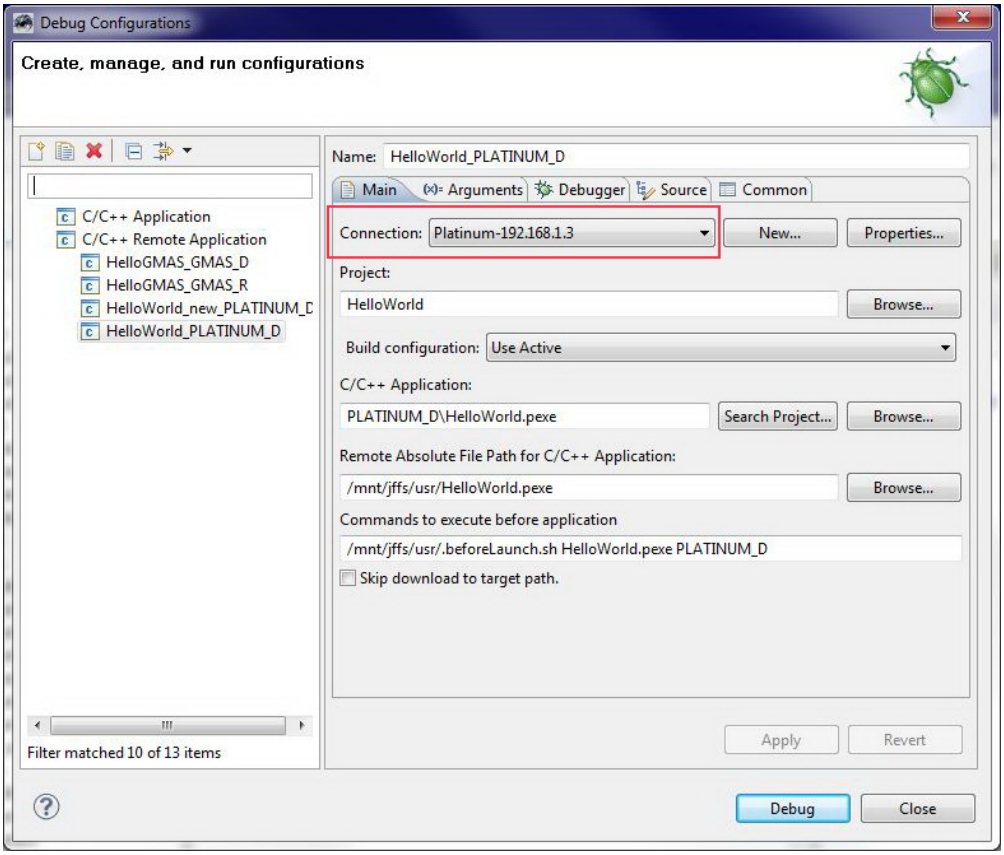
3.8.1 Downloading and Running an Executable File

To download a file and run an executable file in debug mode:

1. Click the ▾ button to the right of the **Debug** button and select **Debug Configurations.**

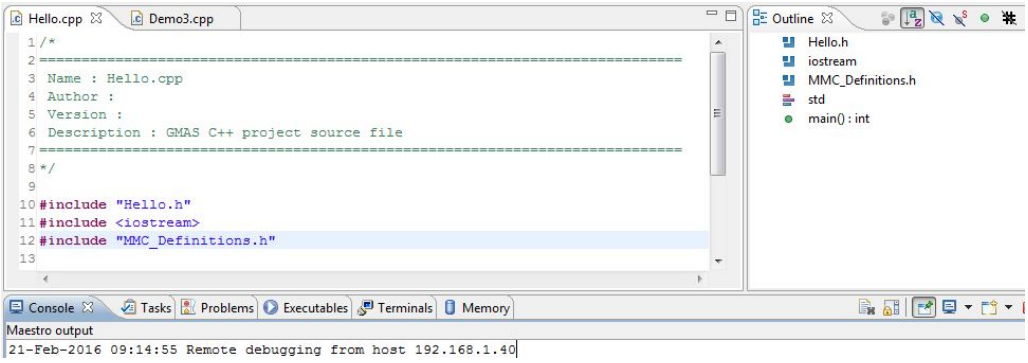


2. In the Create, manage, and run configurations dialog box, select the required project in the left pane.
3. From the **Connection** list, make sure to select the correct hardware connection device.

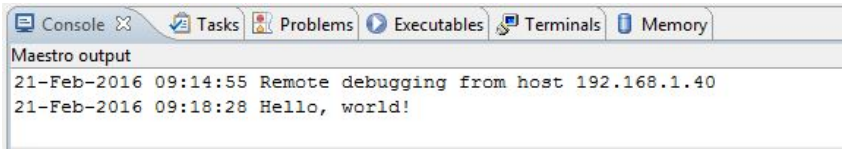


4. Click **Debug**.

The project is downloaded to the Maestro, execution starts and the program is stopped at the breakpoint you set (if you did) or on the first executable line. If other processes are currently running on the Maestro they are stopped (killed) and the downloaded file starts running.



5. Press **F6** to continue execution. The results of the program appear in the Console window.



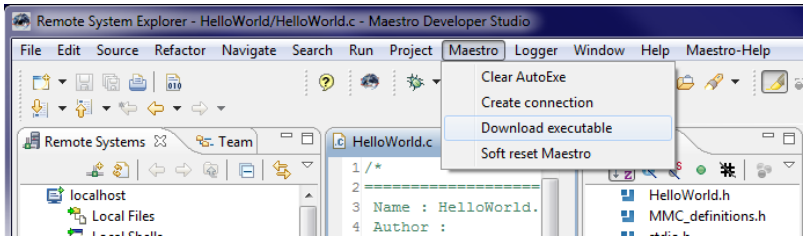
3.8.1.1 Downloading an Executable

This section explains how to download a file that automatically runs when the Maestro is powered on or to download a file and run it later on using the terminal.

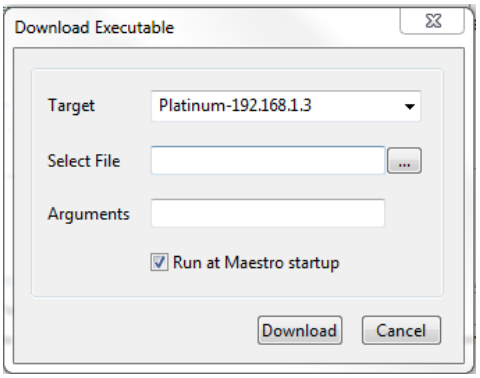
Note: The download destination is the same for both Gold Maestro and Platinum Maestro configurations. The file is saved by default to a location under: **GMAS\GMASUserProg**.

To download an executable:

1. From the **Maestro** dropdown menu, select **Download executable**.



The following dialog box opens.



2. From the **Target** list, select the target (Maestro) to where you want to download the file.
3. To the right of the **Select File** text box, click the browse button to locate the **.gexe** or **.pexe** file or enter its name.

Note: If you are using a different version of a certain hardware type, the download destination location must be the same as in the previous version. By default, the executable files are kept in **GMAS\GMASUserProg**.

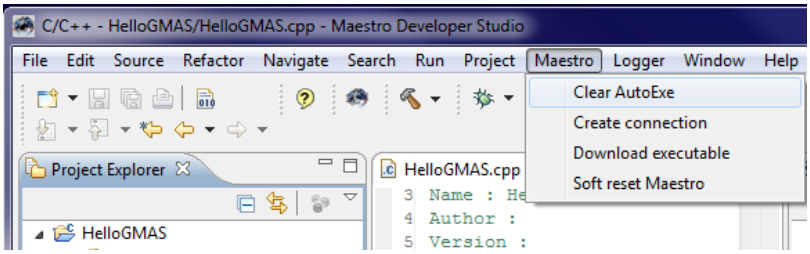
4. Enter the required Argument, for example: -1.
5. If you want this program to run whenever the Maestro is powered on, select the **Run at Maestro startup** checkbox.
6. Click **Download** to download the file to the target.

3.8.1.2 Clear Auto Execution

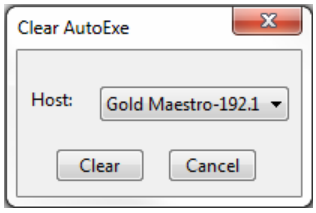
This command clears the Auto Execution of a file within the Maestro.

To clear the auto execution of a file in a Maestro:

1. From the **Maestro** menu, select **Clear AutoExe**.



The Clear AutoExe window opens.



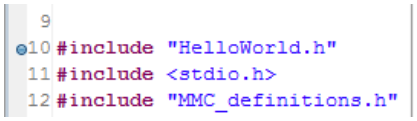
2. Click **Clear** to clear the default **UserStartup.h** from autoexecution. The Terminal shows that the file has been removed.
   If other Maestro's are connected then, select the correct Maestro Host from which to clear the autoexec file.

3.8.2 Adding Breakpoints

A breakpoint is set on an executable line of a program. If the breakpoint is enabled when you debug, the execution suspends before that line of code executes.

To add a breakpoint point

1. Double-click the marker bar located in the left margin of the **C/C++ Editor**, beside the line of code where you want to add a breakpoint.
   A dot is displayed in the marker bar and in the **Breakpoints** view, along with the name of the associated file.



3.8.2.1 Changing the Example Project

To demonstrate more complicated debug options we will use another example program that includes more features.

1. Follow the procedure described in section 5.1 Importing a Project to import a project using the files located by default in C:\GMAS\GMASSamples\HelloGMAS.
   If you selected a different location for your libraries during installation look for the **HelloGMAS** folder in that location.
2. Use the **Build** button to build the project.
   This project includes the example program that is described in more detail in the Gold Maestro/Platinum Maestro Software Manual. This program opens a connection to Maestro. Print the version number and close the connection. Now, we will not try to understand how it functions; only run it on the Maestro to demonstrate some debugging tools.

3.8.3 Watchpoint

A watchpoint is a special breakpoint that stops the execution of an application whenever the value of a given expression changes, without specifying where it might occur. Unlike breakpoints (which are line-specific), watchpoints are associated with files. They take effect whenever a specified condition is true, regardless of when or where it occurred. You can set a watchpoint on a global variable by highlighting the variable in the editor, or by selecting it in the Outline view.

To set a watchpoint on a global variable:

1. Highlight the variable in the editor, or select it in the Outline view.
2. Click **Run > Toggle Watchpoint**.
3. Do any of the following:

   - To stop execution when the watch expression is read, select the **Read** check box.
   - To stop execution when the watch expression is written to, select the **Write** check box.

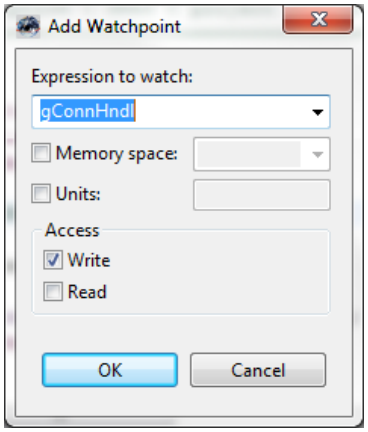
   The watchpoint appears in the Breakpoints view list.
   For example:

   a. Double-click the file name **HelloWorld.c** to open it in the editor.

   b. Find the global variable `gConnHndl` and highlight it:

```
10 #include "HelloWorld.h"
11 #include <stdio.h>
12 #include "MMC_definitions.h"
13
14 MMC_CONNECT_HNDL gConnHndl;
15
16 int main()
17 {
18     printf("Hello, MaestroGold!\n");
19     return 0;
20 }
21
```

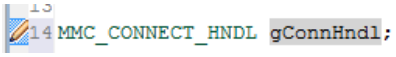
   c. Click **Run > Toggle Watchpoint**. The Add Watchpoint window appears.



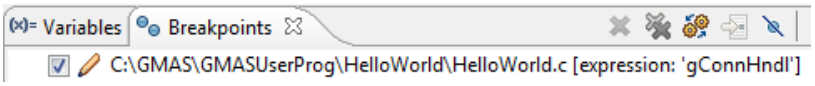
Since “Write” is selected, the program execution will halt when this variable is written.

d. Click **OK**.

Note the icon that now appears next to the marked line:



e. Go to the Debug perspective and look at the breakpoints tab. You will see that a breakpoint was added:



f. Click **Run**->**Run Configurations**. In the opened dialog, select HelloWorld_Platinum_D.

g. Verify that the IP address in the connection field is correct and click **Run**.

You can see the output of the program in the Console tab at the lower part of the screen.

3.8.4 Removing Breakpoints and Watchpoints

When you remove a breakpoint or watchpoint, the corresponding icon is removed from the marker bar, from where it was inserted, and from the **Breakpoints** tab.

To remove breakpoints or watchpoints:

In the **Breakpoints** tab, do one of the following:

1. Select the breakpoints and watchpoints you want to remove.

   Alternatively, right-click, and select **Select All**.
2. In the Breakpoints view, right-click and select **Remove** or **Remove All**.

3.8.5 Enabling and Disabling Breakpoints and Watchpoints

You can temporarily disable a breakpoint or watchpoint without losing the information it contains.

To enable or disable breakpoints or watchpoints

1. In the **Breakpoints** tab, do one of the following:

   a. Select the breakpoints and watchpoints that you want to remove by selecting **Edit > Select All**.

   Alternatively, right-click and select **Select All**.

   b. In the Breakpoints view, right-click the highlighted breakpoints and watchpoints and click **Disable** or **Enable**.

For example:

Set a breakpoint in the file HelloGMAS.c at the line reading:

"printf("MMC VERSION : <%c.%c.%c.%c> U-BOOT VERSION: <%d> \n" by double clicking the marker line next to it:

```
43    // Print Output:
44    printf("MMC VERSION : <%c.%c.%c.%c> U-BOOT VERSION: <%d> \n",
45                ver_out.cFirst, ver_out.cSecond, ver_out.cThird, ver_out.cFourth,
46                ver_out.uiUbootVer);
47    //
```

Now run the program in the debug mode. If the program is already running click the stop button first ( ). After starting, the program stops in the first line :

```
26    int iLibVer ;
27
28    //
29    // Open a connection to the GMAS. There is no callback function defined.
30    if (MMC_InitConnection(MMC_IPC_CONN_TYPE, conn_param, NULL, &gConnHndl) != 0) {
31        printf("ERROR:%s: MMC_InitConnection fail\n", __func__);
32        return -1 ;
33    }
34    //
35    // After we successfuly opened the connection - read the version.
36
37    if (MMC_GetVersionCmd(gConnHndl, &ver_out) != 0) {
38        printf("ERROR:%s: MMC_GetVersionCmd fail \n", __func__);
39        MMC_CloseConnection(gConnHndl);
40        exit(-1);
41    }
```

You can now use **F8** to resume running, **F5** to step into the function or **F6** to step over the function.

Alternately you can use the toolbar buttons:

Press the **F8** button. The program continues until the breakpoint that we have set. The line where the program is stopped is highlighted and an arrow in the marker bar points to it:

```
41    }
42    //
43    // Print Output:
44    printf("MMC VERSION : <%c.%c.%c.%c> U-BOOT VERSION: <%d> \n",
45                ver_out.cFirst, ver_out.cSecond, ver_out.cThird, ver_out.cFourth,
46                ver_out.uiUbootVer);
47    //
```

If you look at the console tab you can see that the connection was already started.
Pressing **F6** will cause the print command to be executed. The result of the print appears in the console window and the program is now stopped in the beginning of the next function.

Press **F8** to end the program.

3.8.6 Controlling Debug Execution

The debug execution controls are superseded by breakpoints. For example, if you attempt to step over a function and the program hits a breakpoint, it pauses, regardless of whether the function is completed. You can control your debug execution in various ways, but they all rely on a core set of debug controls.

To control a debug execution

1. In the **Debug** view, select a thread.
2. To control the debug session, select any of the following according to their actions:

   Run > Resume

   Run > Suspend

   Run > Terminate

   Run > Disconnect

   Run > Remove All Terminated Launches

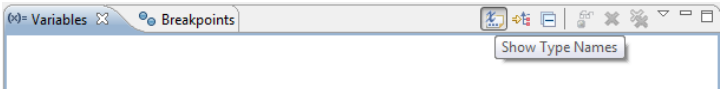
   Run > Restart

3.8.7 Working with Variables

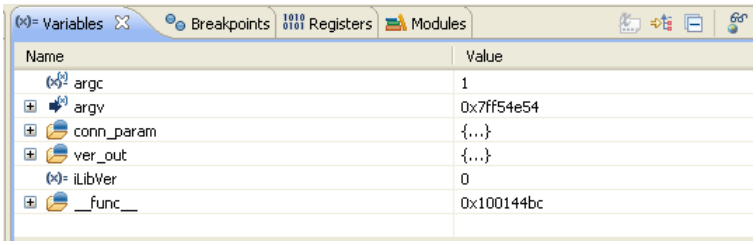
During a debug session, you can display variable types, and change or disable variable values.

To display variable type names:

1. In **Variables** view, click the **Show Type Names** toggle button.



When you start debugging the program the window displays the variables:



To change a variable value while debugging:

1. During a debug, you can change the value of a variable to test how your program handles a particular value or to speed through a loop.
2. In **Variables** view, right-click a variable, and select **Change Value**.
3. Type a value.

To disable a variable value while debugging:
You can disable a variable so that the debugger does not read the variable's value from the target. This is useful if the target is very sensitive or the variable is specified as volatile.

In **Variables** view, right-click a variable, and select **Disable**.

3.9 Dynamic Linking

MDS supports using both dynamically and statically linked libraries for C/C++ projects.

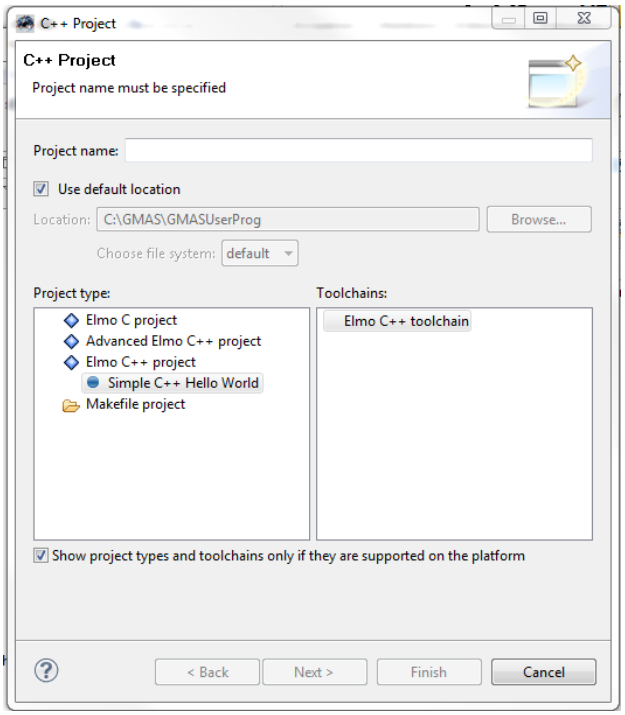
- If you choose to statically link the API library, the executable either includes all of the code together with the library (larger executable) or it is aligned to the library at run time.
- If you choose to dynamically link the API library, only a file pointer is included in the executable and not the file contents. A dynamic linker loads external shared libraries only when you run the executable file and then adds the shared libraries to the running process.

3.9.1 Defining a New C/C++ Project for Dynamic Linking

When you create a new C/C++ project, the project is linked statically by default. This section describes how to dynamically link an API library.

To define a new C/C++ project for dynamic linking:

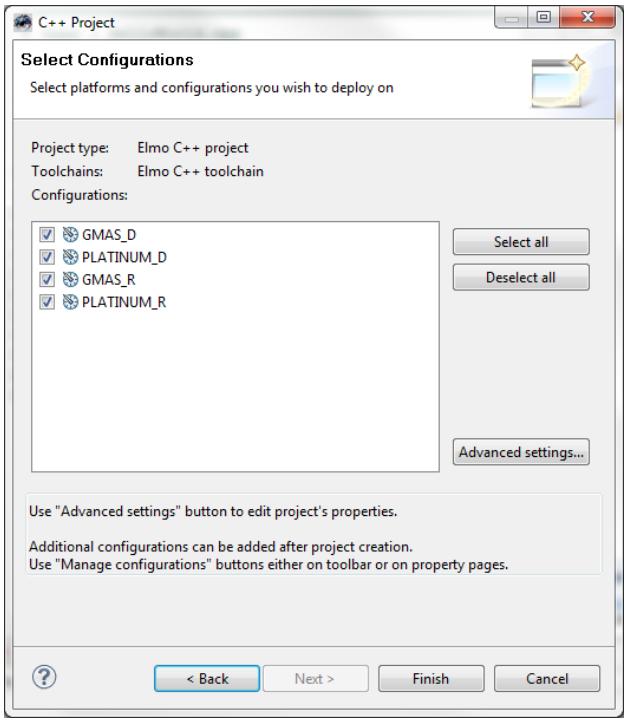
1. From the **File** menu select **New > C/C++ Project**. The C/C++ Project window opens.



2. Select the required project type. The toolchain is automatically selected according to the project type.
3. Enter the project name.

   Note: Do not add spaces to the project name.
4. (Optional) The default setting of this dialog shows projects in the default location for Elmo's Maestro. You can select one of the projects displayed under **GMASUserProg** or clear the **Use default location** checkbox and click the browse button to select another location.

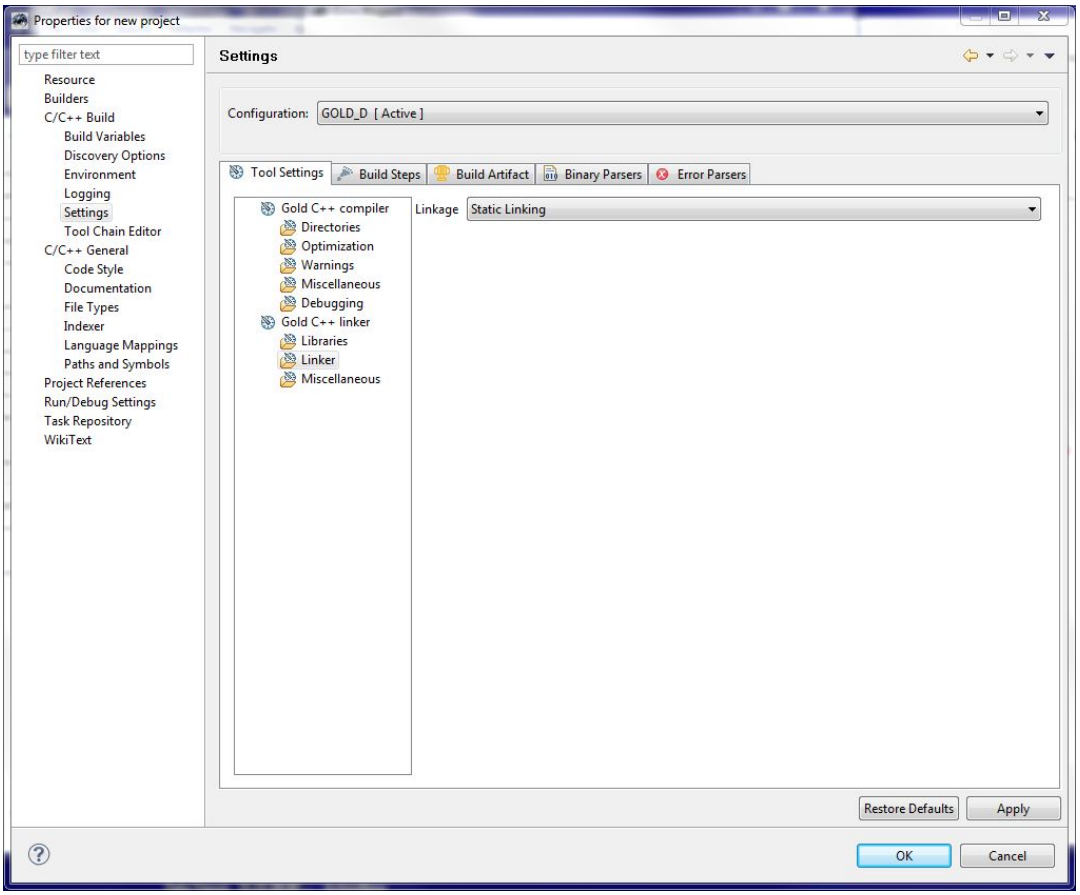
5. Click **Next**. The Select Configurations window opens.



6. Select one or more configuration to apply or click **Select all**.

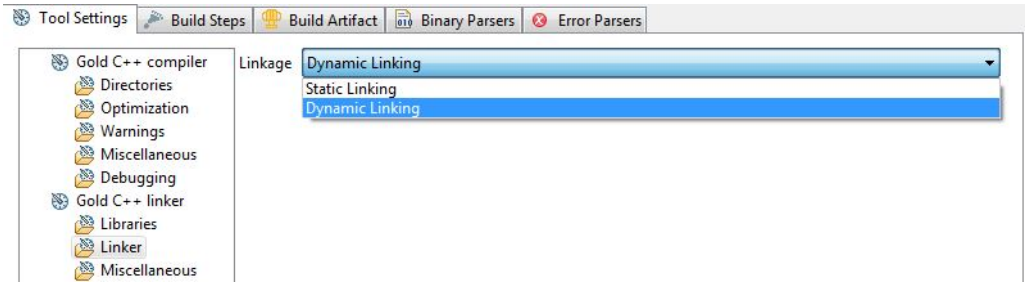
. **Note:** By default, Eclipse generates both debug and release versions when building a project.

7. Click **Advanced settings….** The Properties window opens.

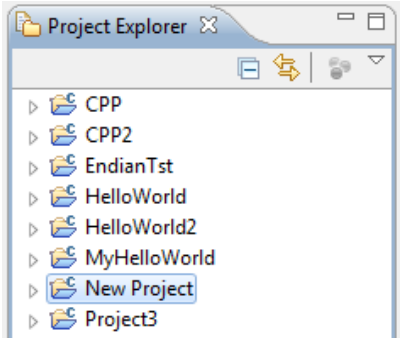


8. In the left pane, select **Settings**.

9. In the **Tool Settings** tab in the right pane select **Linker**.
10. From the **Linkage** list, select **Dynamic Linking**.



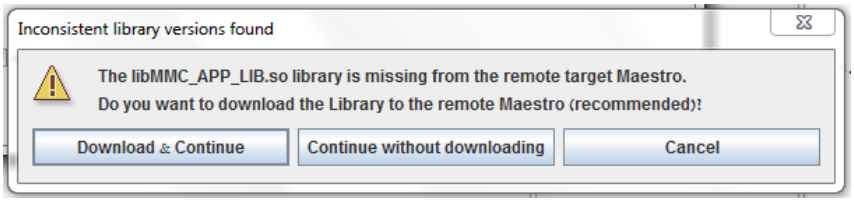
11. Click **Apply** and then **OK**.
12. In the Select Configurations dialog box, click **Finish**. The new project now appears in the project explorer and is linked to the API library dynamically.



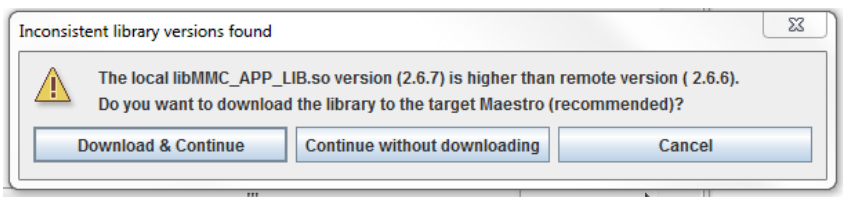
3.9.2 Managing Library Versions

When using dynamic linking, one of the following situations may occur:

- If the dynamically linked shared object **.so** library in the remote library is missing, you get the following message:

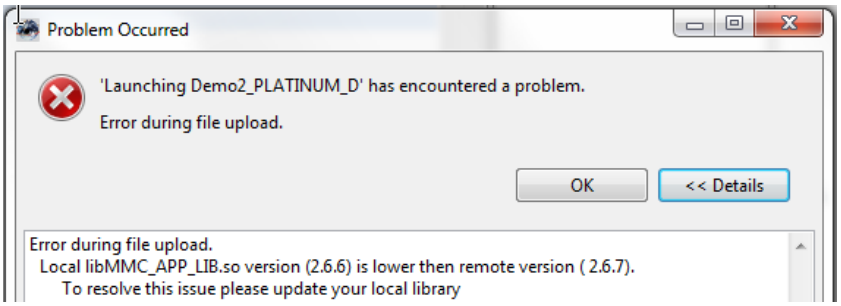


  - Click **Download & Continue** to copy the **.so** library to the target Maestro (recommended)
  - Click **Continue without downloading** to continue running the process without copying the **.so** library to the target Maestro
  - Click **Cancel** to stop running the process
- If the **.so** version in the local library is higher than the version in the remote library, you get the following message:



- Click **Download & Continue** to copy the **.so** library to the target Maestro (recommended)
- Click **Continue without downloading** to continue running the process without copying the **.so** library to the target Maestro
- Click **Cancel** to stop the running process

- If the **.so** version in the local library is the same as the version in the remote library, then MDS continues to run the executable file.
- If the **.so** version in the local library is lower than the version in the remote library, you need to update the local library. You get the following message:



3.9.3 Attach and Debug a Running Program

You can connect to the Maestro and debug a program that is already running on the target.

To enable attaching to the Maestro for debugging purposes:

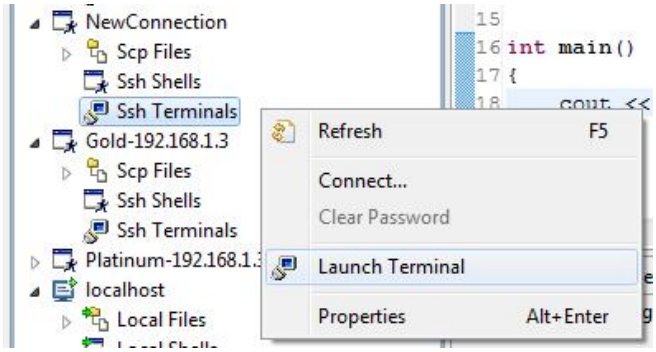
1. When the file is running on the Maestro, it must be built as a debug version, as it is not possible to attach to the released version.
2. First, we will make a program with a loop to demonstrate this feature.
3. Open the project HelloWorld in the C/C++ perspective and edit the file **HelloWorld.c** to look as follows, with a set breakpoint on the line containing “cout”.

```
10 #include "DemoCPP2.h"
11 #include <iostream>
12 #include "MMC_Definitions.h"
13
14 using namespace std;
15
16 int main()
17 {
18     while(1)
19     {
20         cout << "Hello, world!" << endl;
21         sleep(3);
22     }
23     return 0;
24 }
25
```

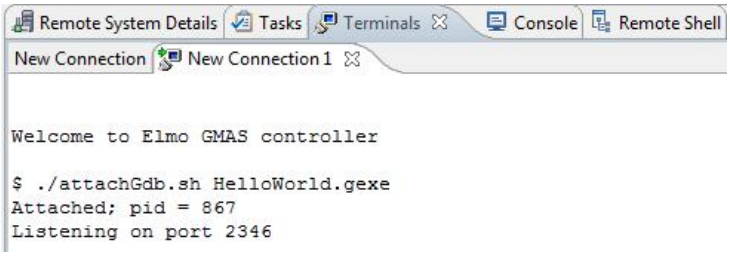
4. Run the program using the terminal command “./HelloWorld.gexe”.

The program starts running and repeatedly prints “Hello World”

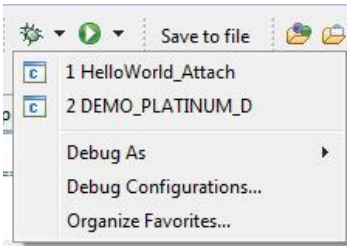
5. Open another terminal window using a right-click on **Ssh Terminals**.



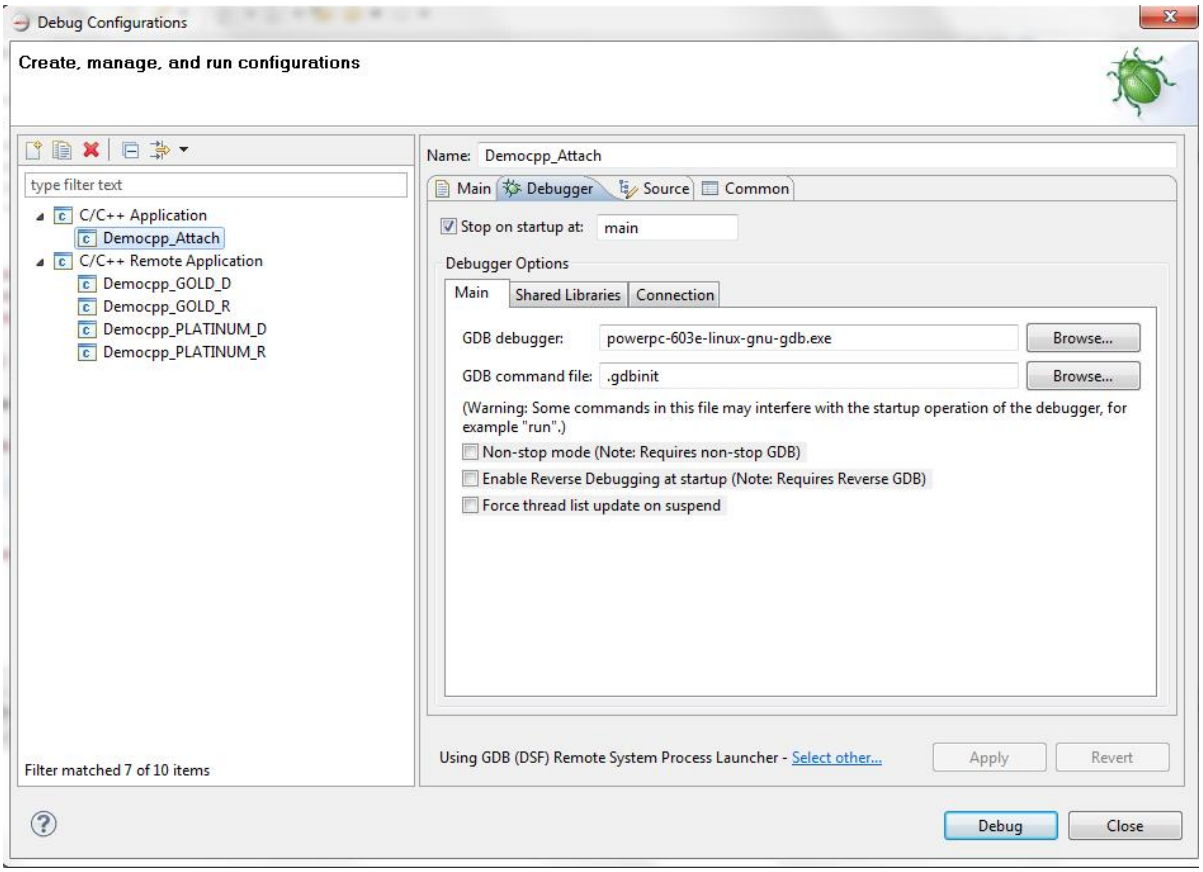
6. Open the new terminal and type in “./attachGdb.sh HelloWorld.gexe”.
   The program execution stops and Maestro is listening as the message indicates.



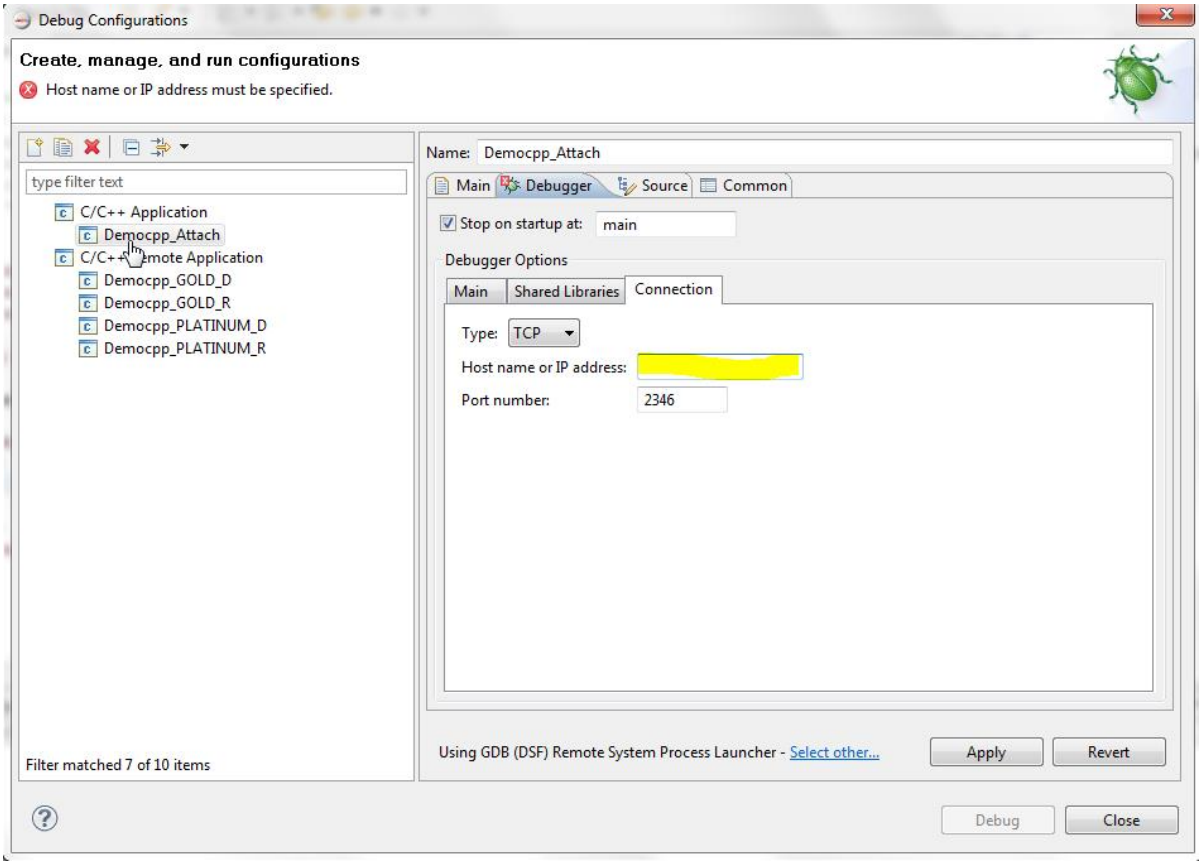
7. Expand the debug menu using the arrow next to the debug button.



8. Select **Debug Configurations** in the dialog that opens select the **Debugger** tab and from it select the **Connection** area:

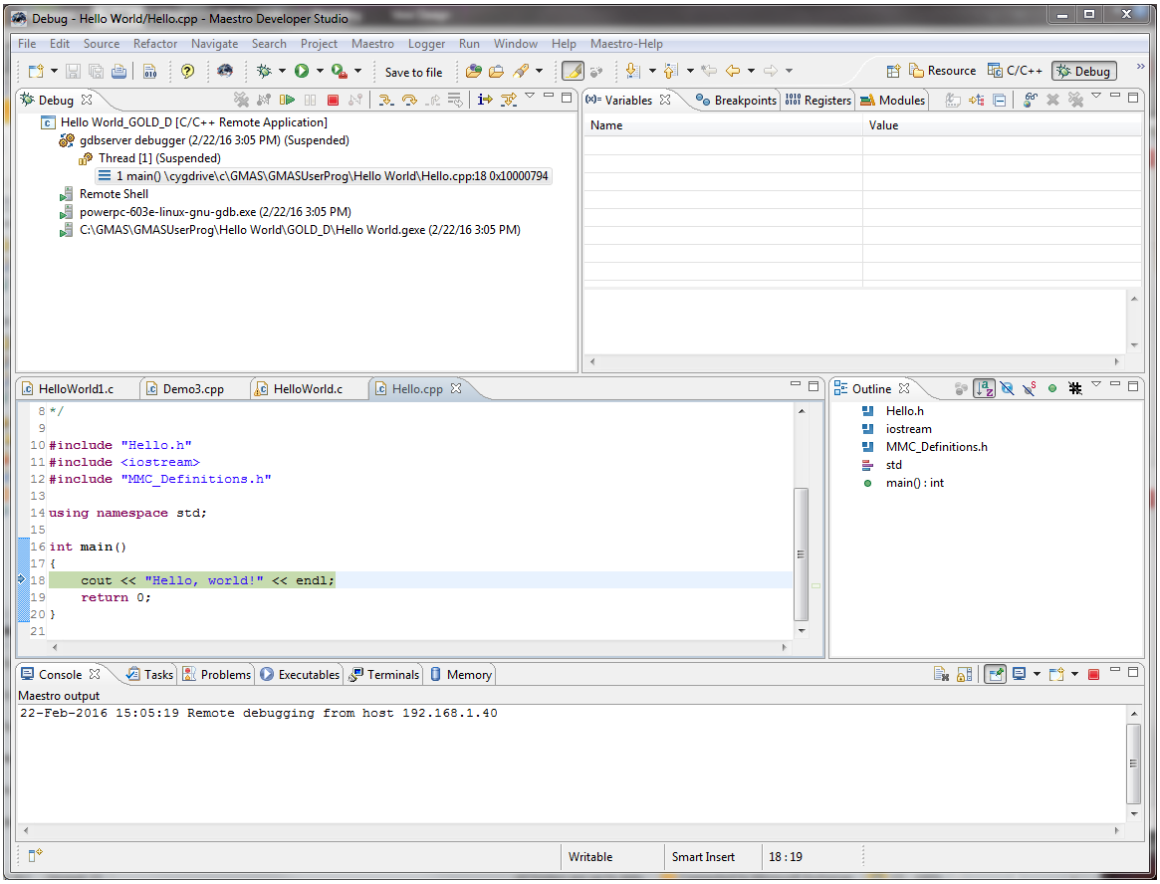


9. Enter the IP address of the target to which you want to connect.



10. Click **Apply** and then **Debug**.

The debugger is now attached to the Maestro and you can debug the program. The terminal indicates the debugging state.



The program is now stopped on the line we put the breakpoint on.

11. Click inside the program area.
12. Click **F6** to continue to the next line, **F5** to step into a function where applicable and use any other debugging actions required.

3.9.3.1 Detach – Exiting Debug Mode

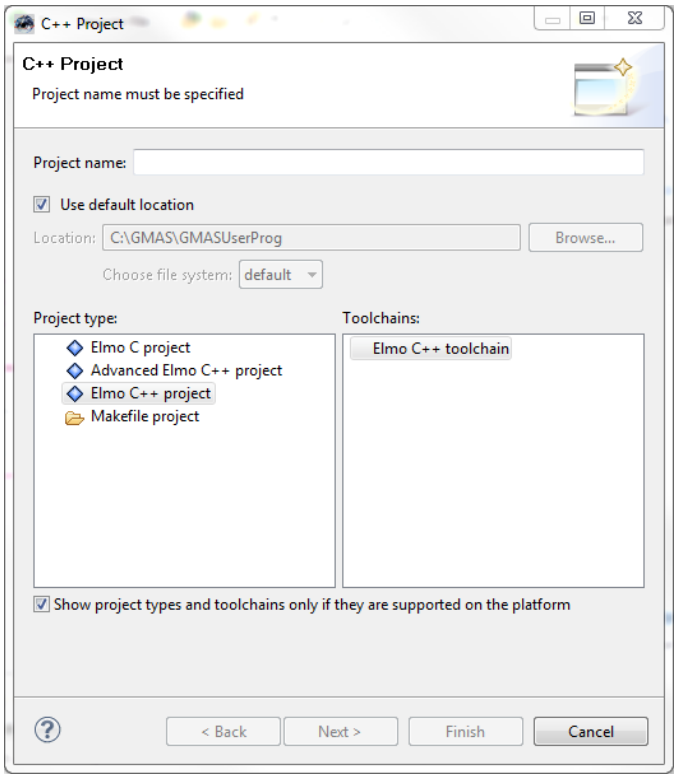
To exit debug mode

1. Double-click the breakpoint to cancel it.
2. Click the **Resume** button  to let the program continue running normally on Maestro.

Chapter 4: Converting a Gold Maestro Project to a Platinum Project

To convert a Gold Maestro project to a Platinum Maestro project:

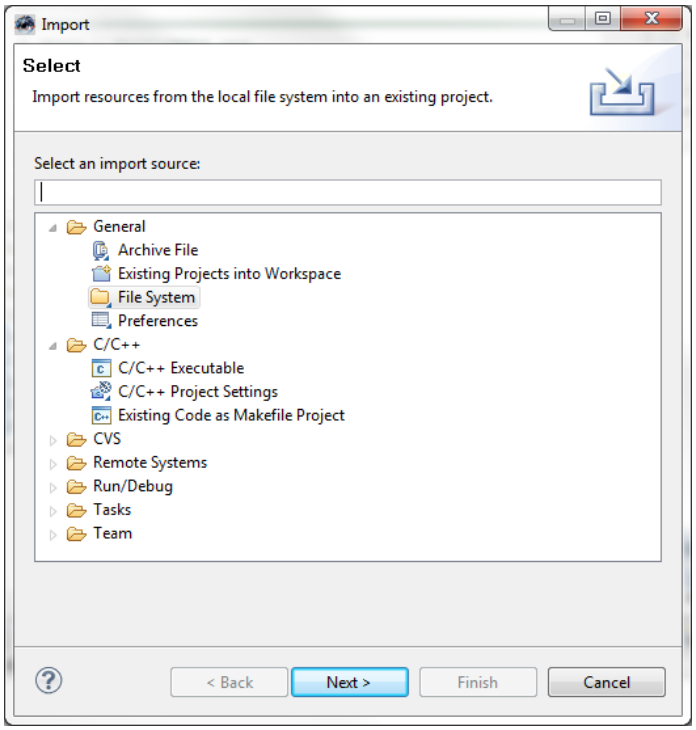
1. Click **File** > **New> C or C++** to create an empty project. The C++ Project dialog box opens.



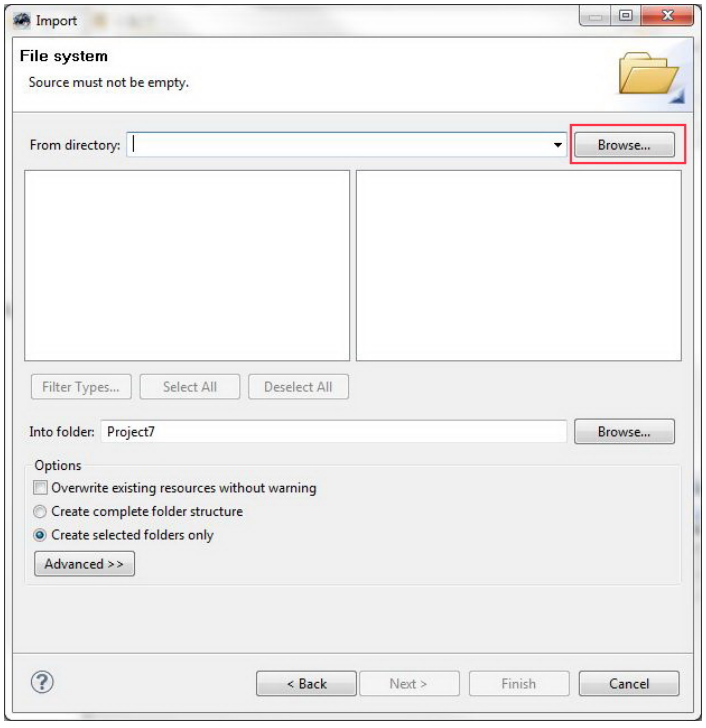
2. In the **Project name** field, enter the required project name and click **Finish**.

   Note: Do not use spaces in the project name.

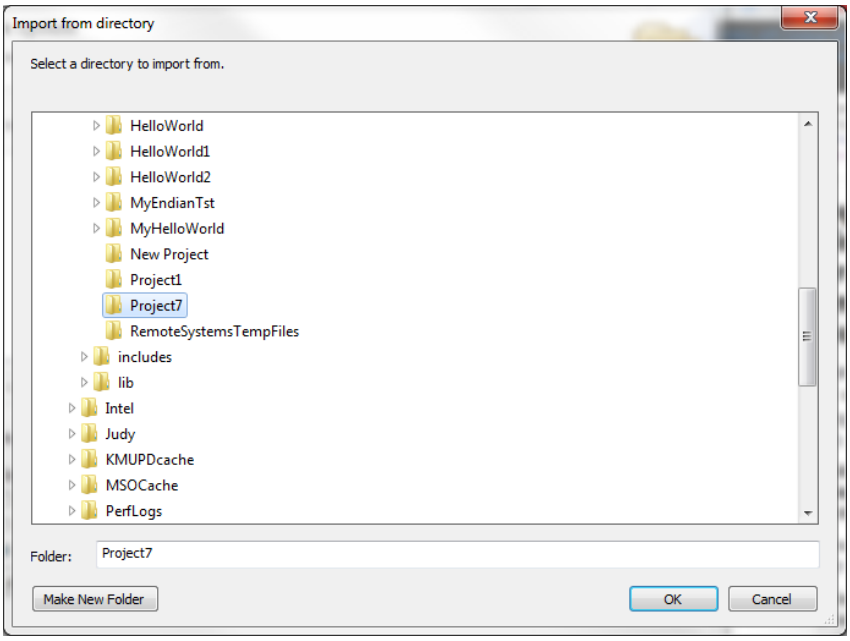
3. In the **Project Explorer** tab**,** right-click the project you created and click **Import**. The Import dialog box opens.



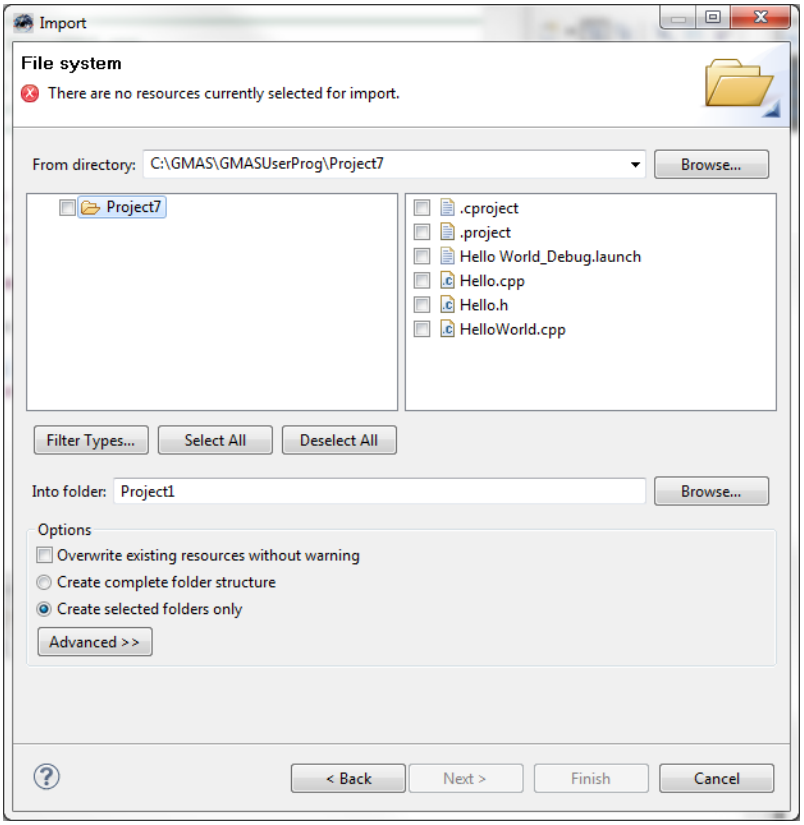
4. Select **General** > **File System** and then click **Next**. Opens the File System dialog box.



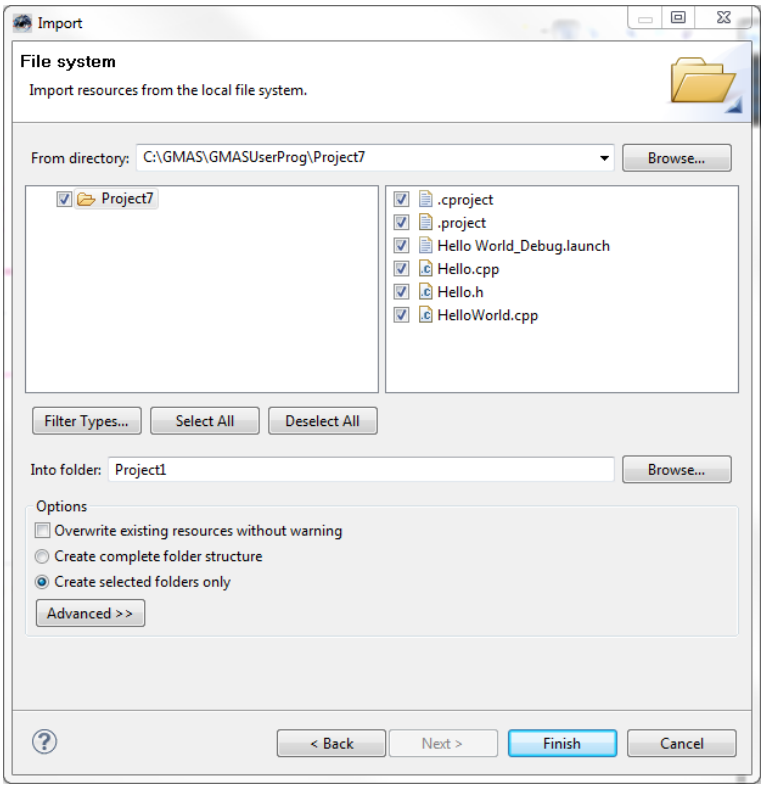
5. Click the **Browse** button to the right of the **From directory** field to open the Import from directory window.



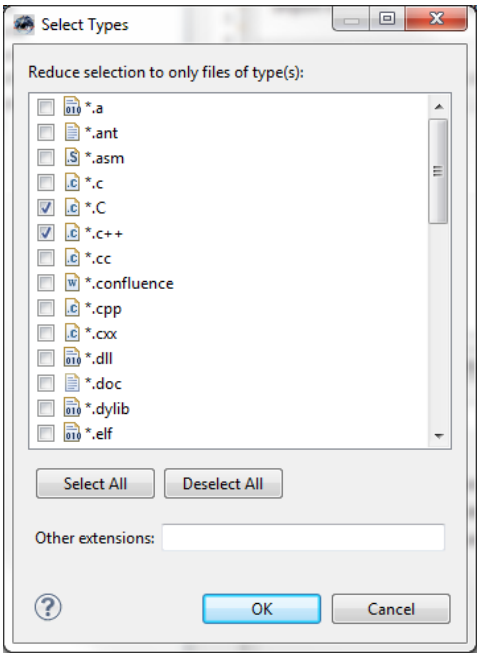
6. Find the required project and click **OK**. The File system dialog box opens.



7. In the right pane, select the required sources for import.



8. Click the **Filter Types** button to open the Select Types window and select the required file types to import.



9. Click **OK** and then in the File System dialog box, click **Finish**.

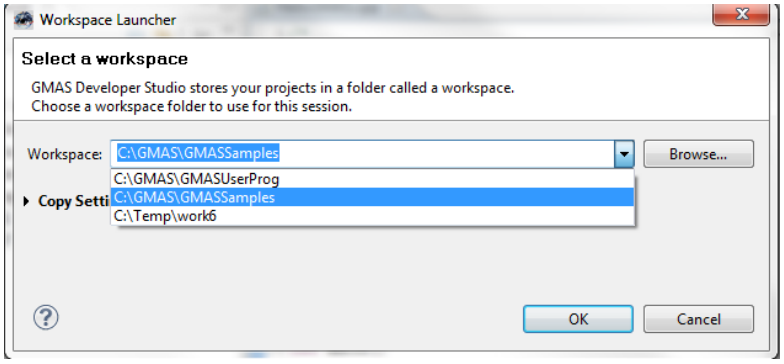
Chapter 5: Importing and Debugging Projects Example

This is an example of a real in-situ situation where a project is imported and debugged. It enhances your understanding of the procedures described in the previous chapters.

5.1 Importing a Project

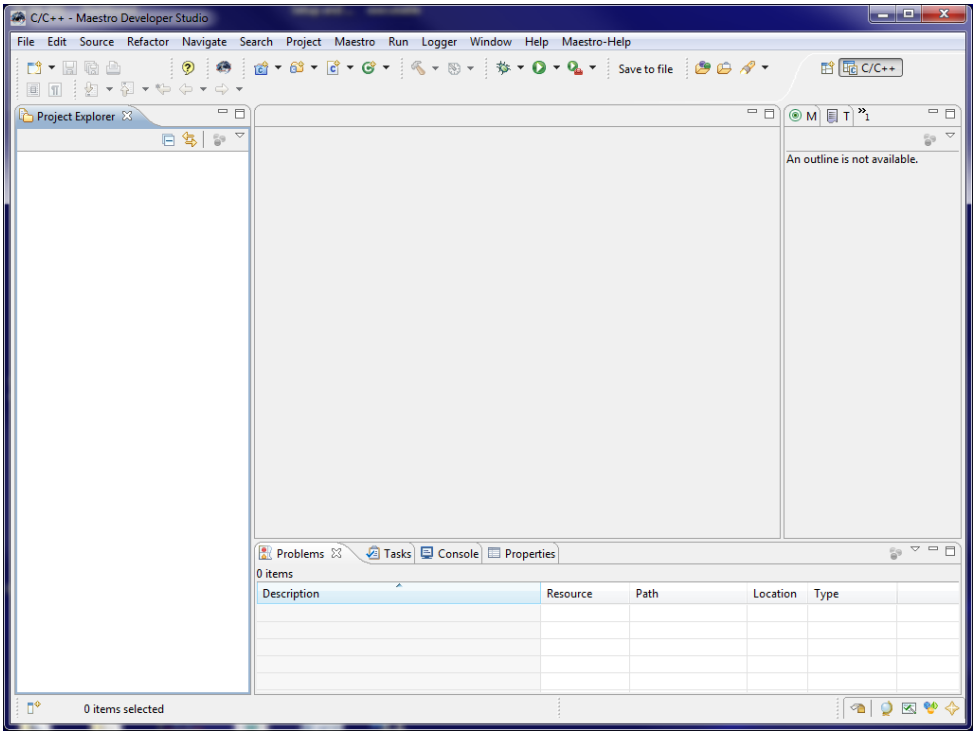
To import the project:

1. Run Maestro Developer Studio (MDS).
2. Select the required workspace with **File**->**Switch Workspace**.
   Select **Other** if no other choices are available or correct.
3. To begin, try the HelloGMAS project in the folder: C:\GMAS\GMASSamples.
   This is installed with the Maestro Developer Studio. When prompted for a workspace, select the folder: C:\GMAS\GMASSamples and click **OK**. Maestro Developer Studio restarts.

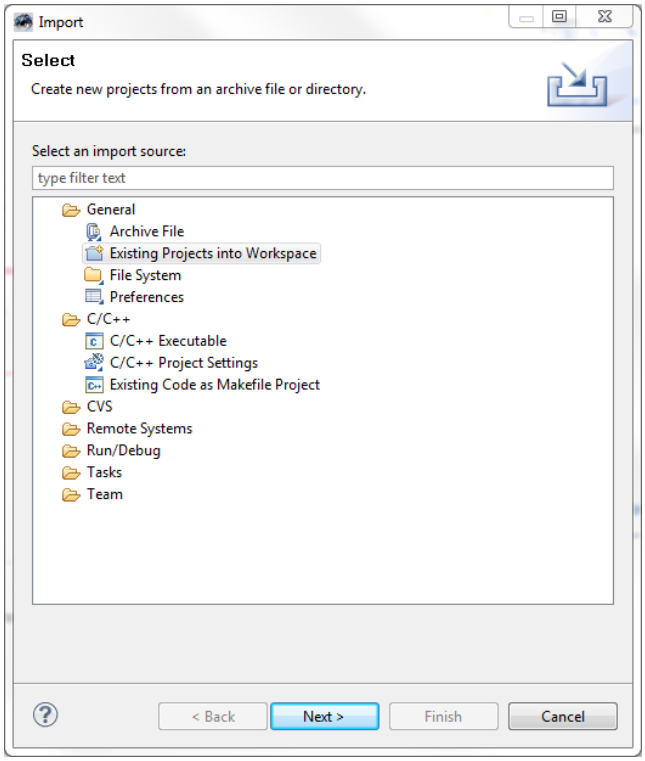


4. After MDS restarts, click the **Go to Workbench** link at the right-hand side of the Welcome screen.

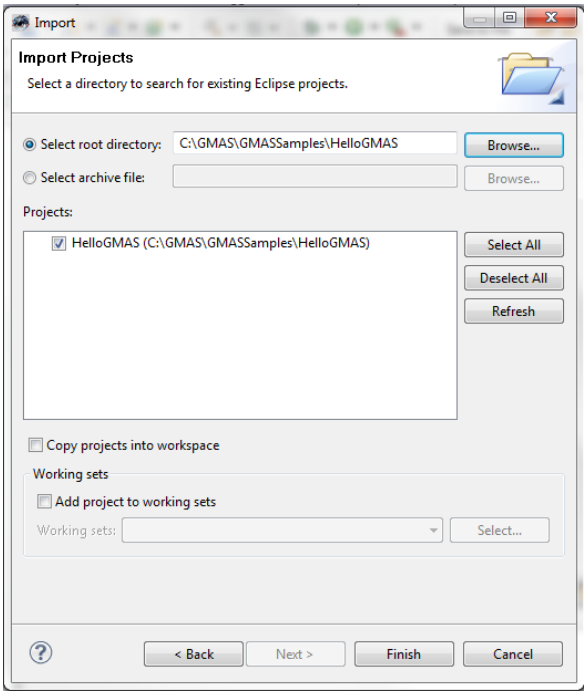
The workspace is blank and does not contain any projects.



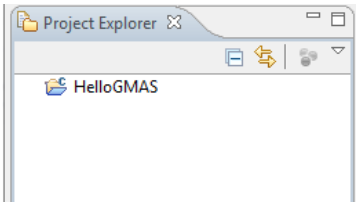
5. Import the HelloGMAS project from **File**->**Import**.
6. Select **General**->**Existing Projects into Workspace.**



7. Click **Next**.
8. Click **Browse** and select the C:\GMASGMASSamples folder as the root directory.
9. In the **Projects** area, select the HelloGMAS project.



10. Click **Finish**. The newly imported project is displayed in the Project Explorer



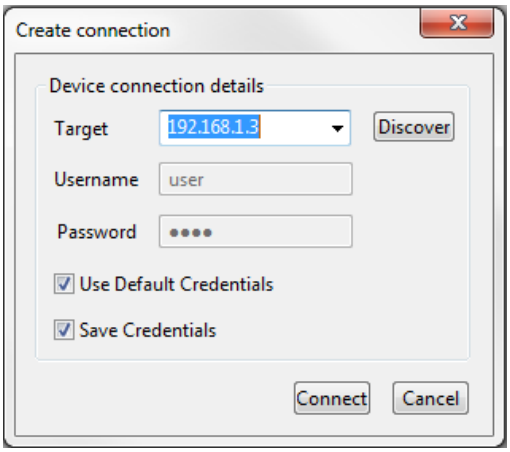
5.2 Connecting and Downloading a Project to the Maestro

To connect to the Maestro and download a project:

1. Create a connection to the Maestro, by selecting **Maestro** -> **Create Connection**.

   a. Click the **Discover** button to scan for available Maestros or enter the target Maestro IP address manually.

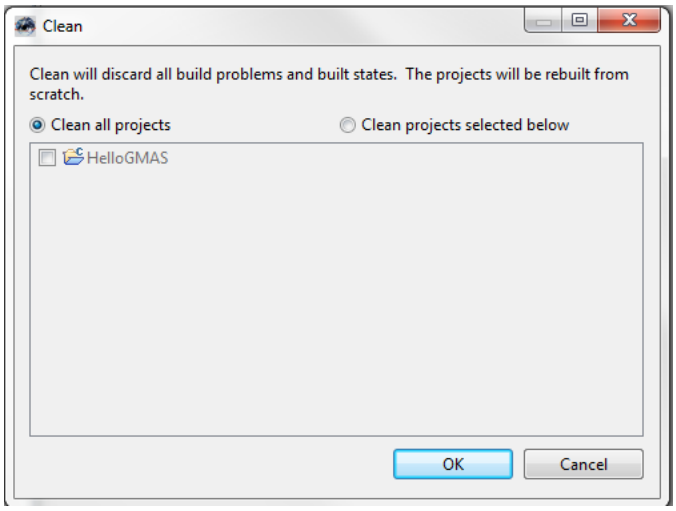
   The default IP address is 192.168.1.3.



   b. Use the default credentials.

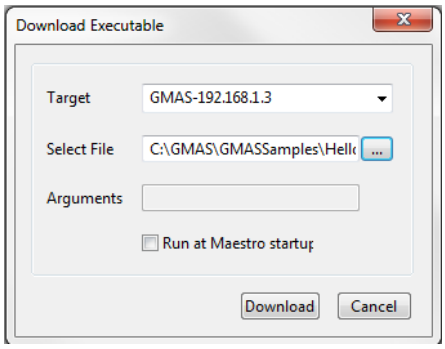
   c. Click **Connect**.

2. If the project did not automatically build during the import, clean and build the project by selecting **Project** -> **Clean**.

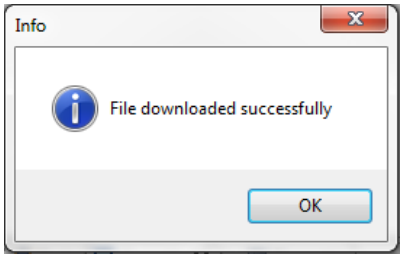


3. Select the required project.
4. Click **OK**.
5. Download the project to the Maestro by clicking the **Download Executable** button.

6. When prompted, select the file:
   C:\GMAS\GMASSamples\HelloGMAS\Debug**HelloGMAS.gexe** file.
7. For the purposes of this example, clear the **Run at Maestro startup** checkbox.



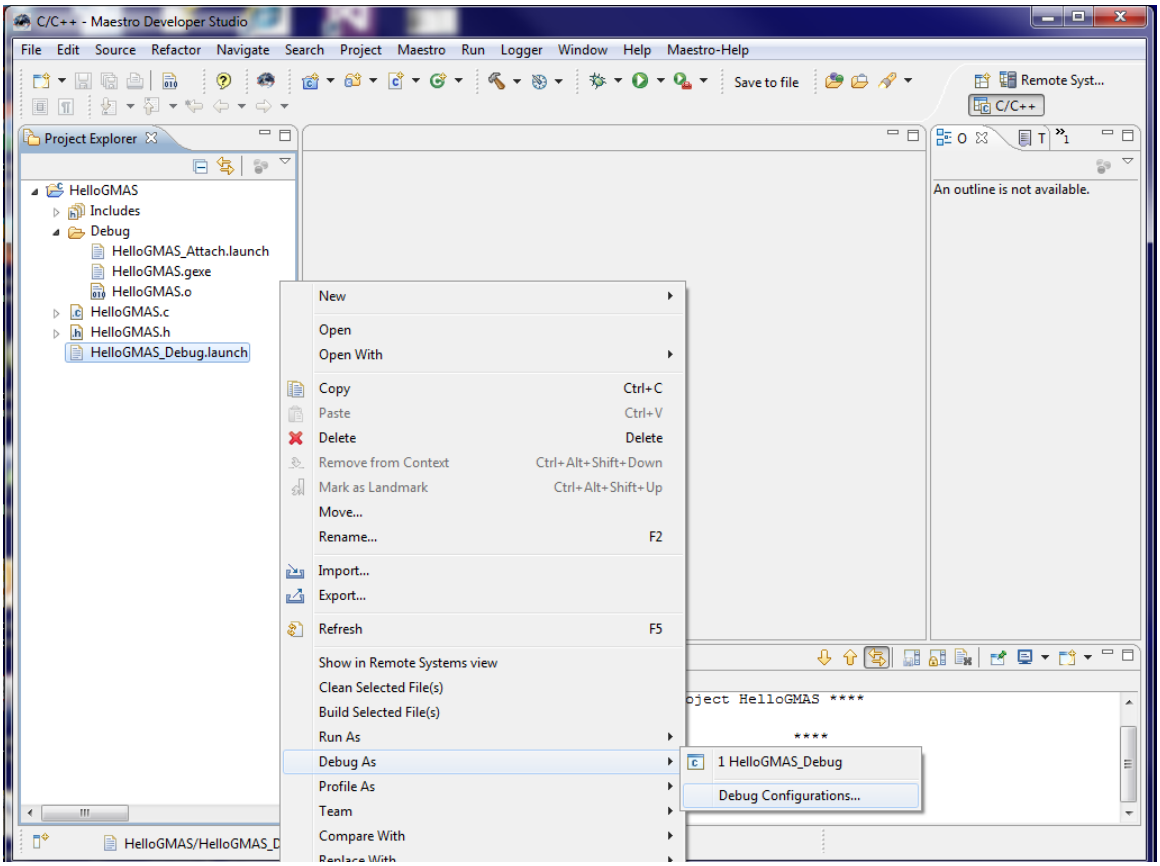
8. Click **Download**. If downloaded successfully, the following message appears.



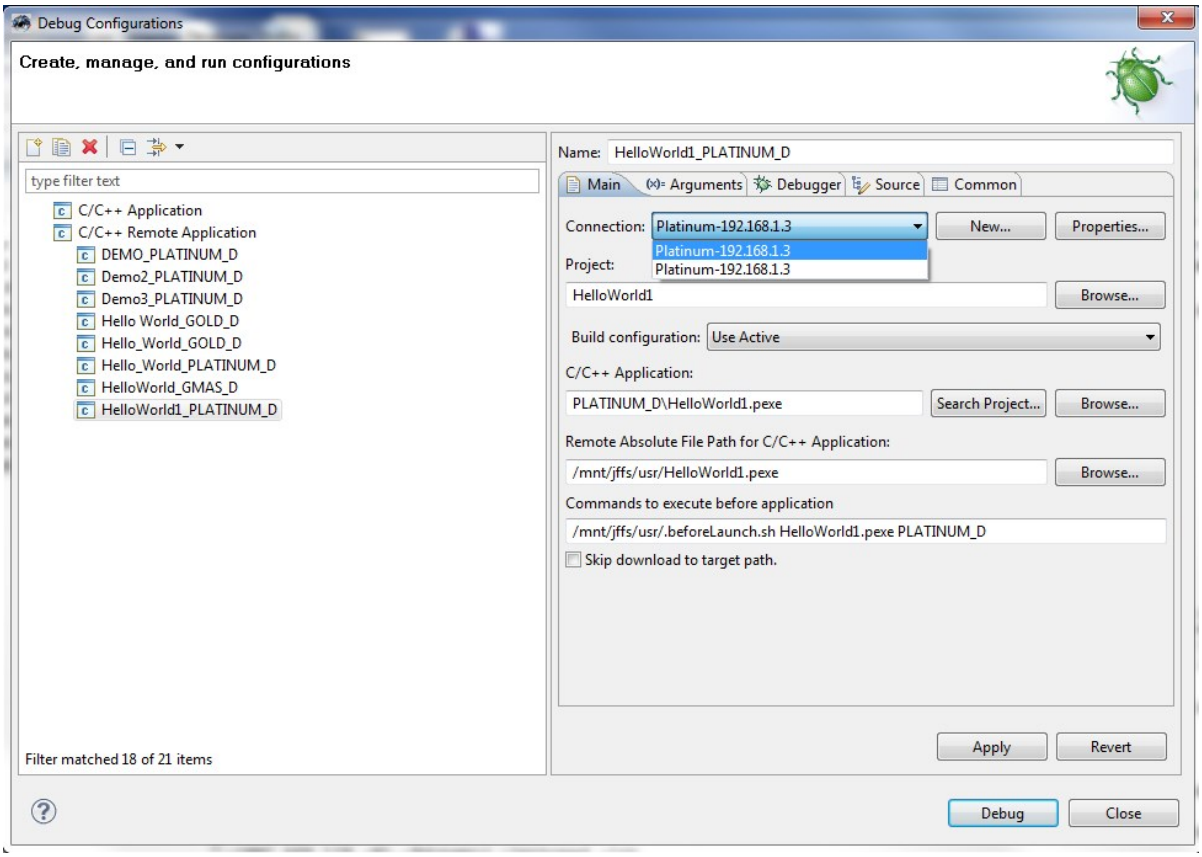
5.3 Debugging the Project

To debug the project:

1. In the Project Explorer, right-click **HelloGMAS_Debug.launch** and select **Debug As - > Debug Configurations**.



2. Make sure that the connection is set to the Maestro.



3. Click **Apply** if it is enabled.
4. Click **Debug** to start debugging immediately, or **Close** to debug later.


Inspiring Motion
Since 1988
For a list of Elmo's branches, and your local area office, refer to the Elmo site www.elmomc.com
Elmo
Motion Control