ELEN90066 Embedded System Design

Workshop 2 - Embedded Development Tools

A quick poll

How many of you have:

- Used C before?
- Programmed FPGA before?
- Used Eclipse?
- Used LabVIEW?

Workshop Groups

- Make sure you are in groups of 3.

Workshop 2 - Embedded Development Tools

The goal is to gain some familiarity with the tools by creating a basic "hello world" program using each of them.

Learn how to deal with tools and frameworks may be confusing, awkward, brittle…

You will encounter this in practice; there is always "legacy."

Workshop 2 marking

- There are three tasks to complete in the workshop today, total of 25 marks
  - TASK 1 : Configure the myRIO board: 5 marks
  - TASK 2 : Program the myRIO Processor from Eclipse: 10 marks
  - TASK 3 : Program the myRIO Processor from LabVIEW: 10 marks

Please notify me every time you finish one of the assessed tasks, so I can check that off for you!

Download the code

Please download the workshop code from LMS and unzip it to an appropriate folder on your network drive.

Task 2 - Eclipse
1: Connect myRIO

- Connect a power supply to myRIO.
- When powered, the Power LED on the controller will illuminate.
- Connect myRIO to your computer using a standard USB cable.

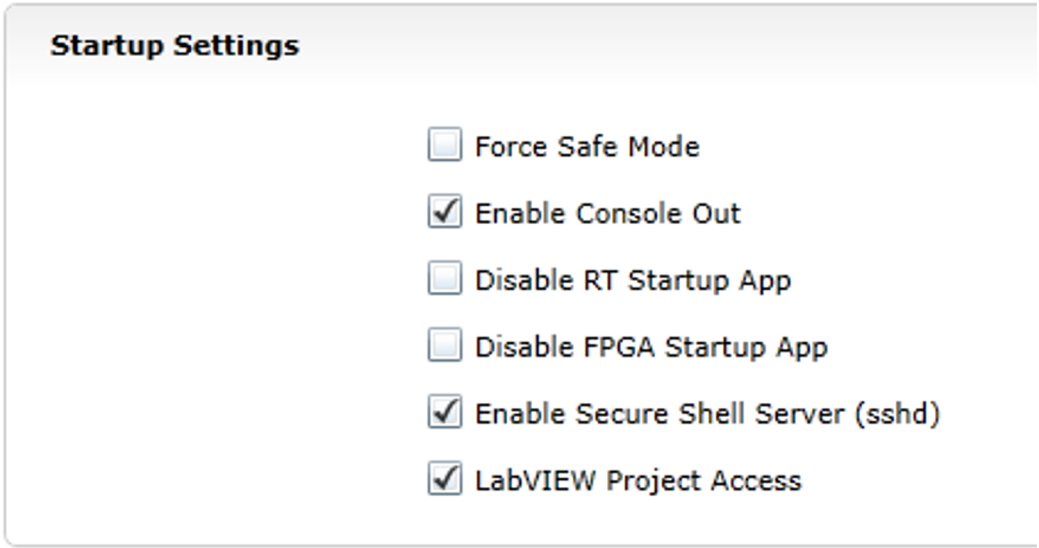
Task 2 - Eclipse

2 : Enable SSH

Your computer communicates with myRIO using the SSH protocol. This allows you to see printf() messages transmitted from myRIO to your computer, and to interact with myRIO using standard Linux shell commands.

These options need to be enabled in the myRIO settings.

You may use NI MAX to enable terminal access.



Task 2 - Eclipse

2 : Enable SSH

MAX may be launched from the Windows desktop or from the Windows Start menu under “National Instruments→NI MAX”.

Task 2 - Eclipse

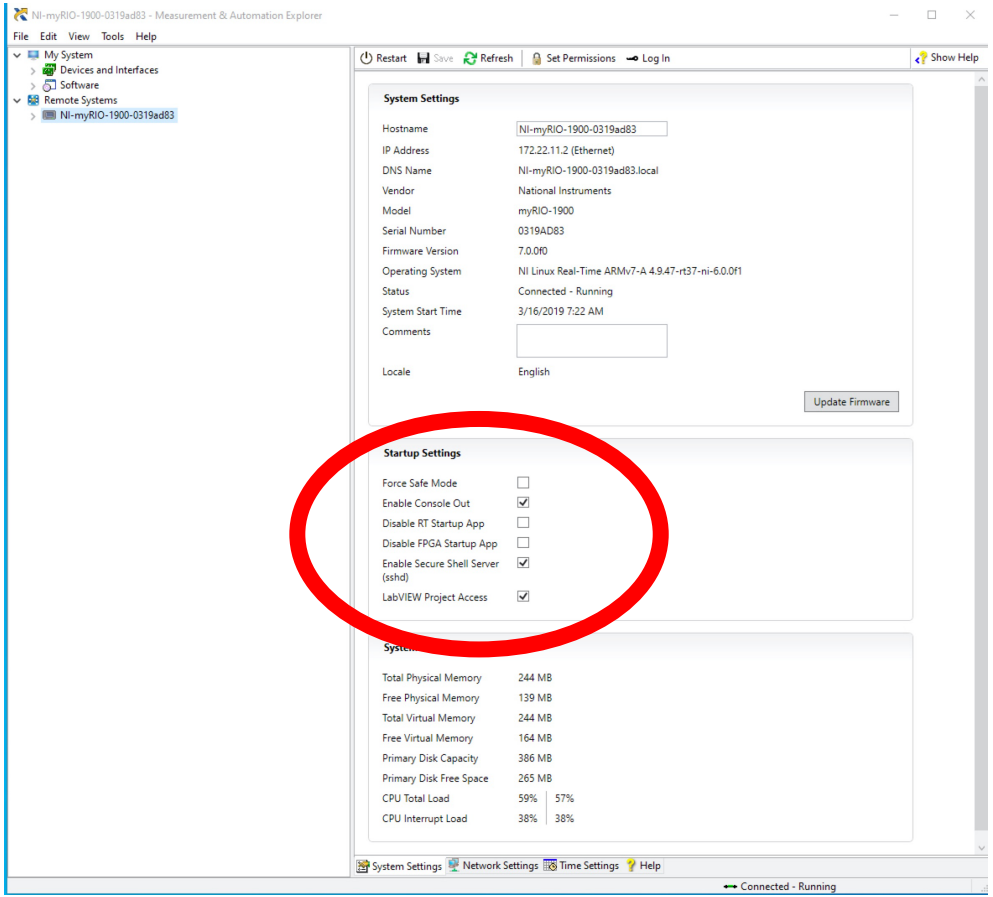
2 : Enable SSH

In the left navigation pane, expand Remote Systems to autodiscover National Instruments devices on the local network. Autodiscovery may take some time to complete.

If your myRIO appears under Remote Systems, it has been discovered.

Click on the myRIO device to see detailed information about the target, including the IP address of the Ethernet over USB adapter.

Click Save and then Restart if any changes are made to the configuration



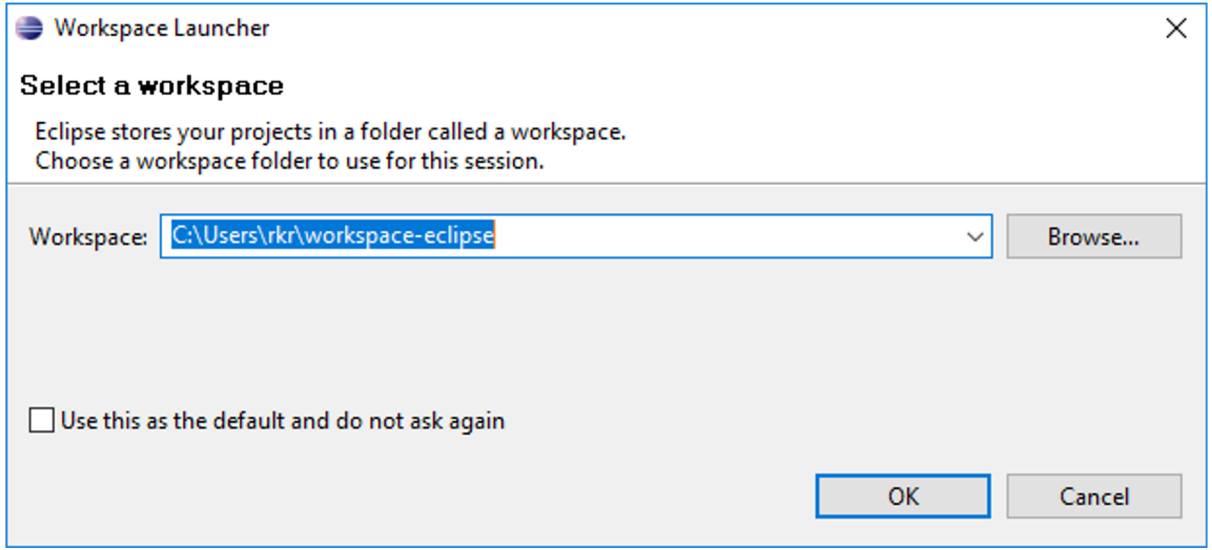
Ensure that the above options are selected

Task 2 - Eclipse

3 : Launch Eclipse

C & C++ Development Tools for NI Linux Real-Time, Eclipse Edition may be found in the Windows Start menu under “National Instruments→C & C++ Development Tools for NI Linux Real-Time 2017, Eclipse Edition”. Upon launching, you may be asked to select a workspace. A workspace organises projects, perspectives, and application preferences.

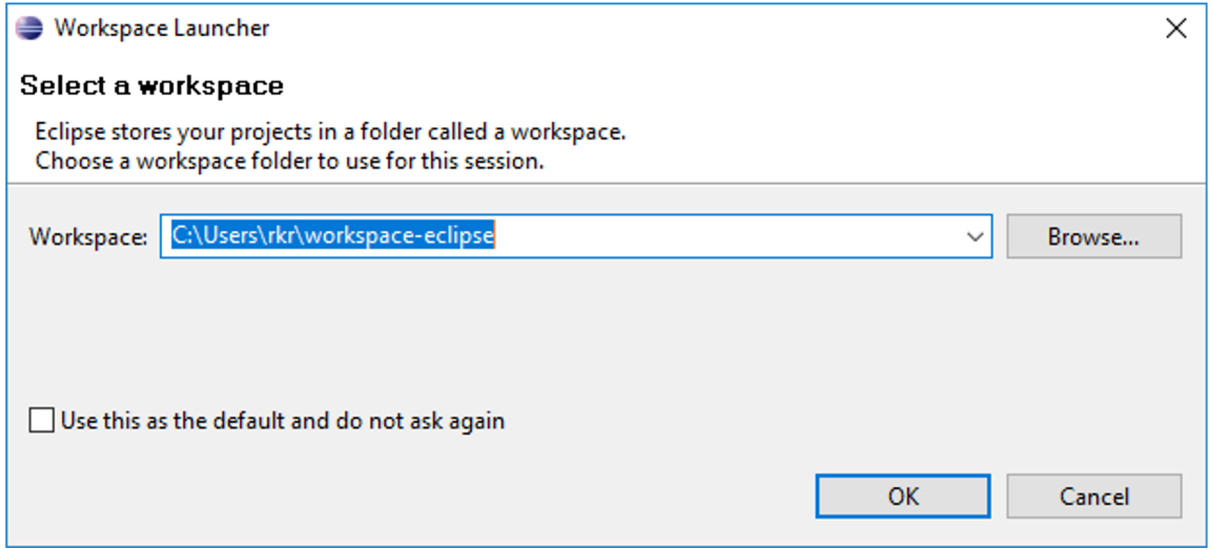
Choose a unique location for this workspace.



Task 2 - Eclipse

3 : Launch Eclipse

If you are not prompted to select a workspace, then Eclipse has been configured to automatically load a default workspace. If you need to change to a new location, then from the top menu bar select “File→Switch Workspace”.

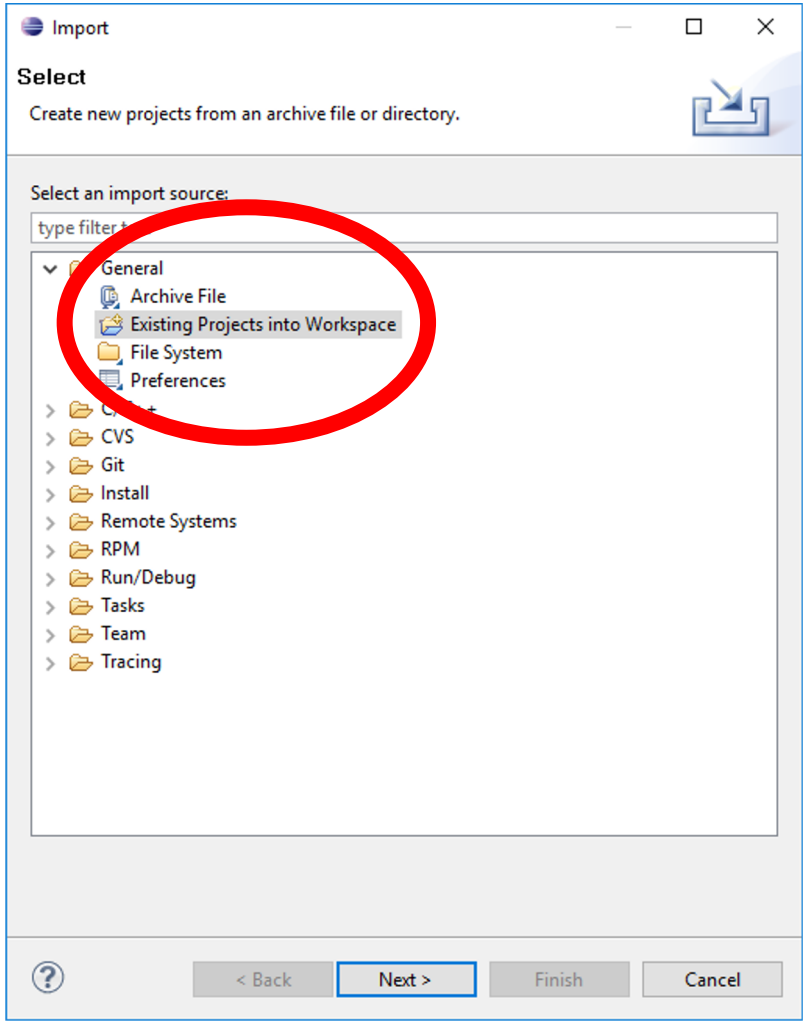


Task 2 - Eclipse

4 : Import Template Project

To import the myRIO project template, from the top menu bar select “File→Import...” to launch the Import wizard.

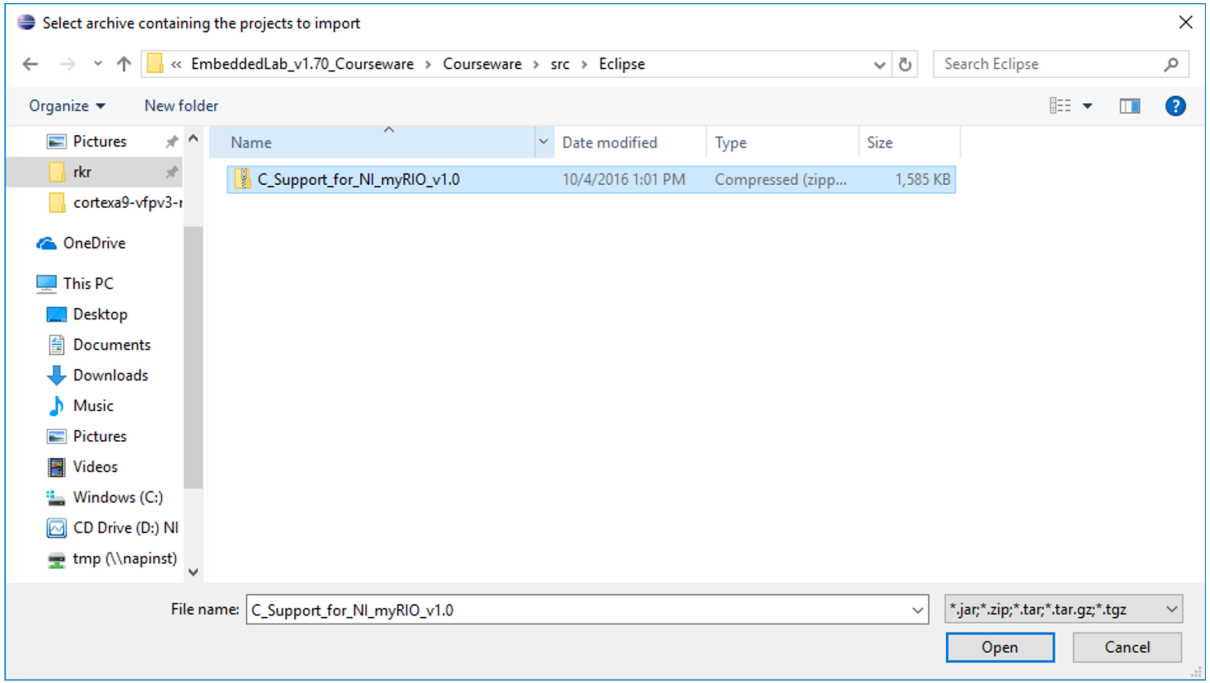
In the Import panel, select “General→Existing Projects into Workspace” and press [Next]



Task 2 - Eclipse

4 : Import Template Project

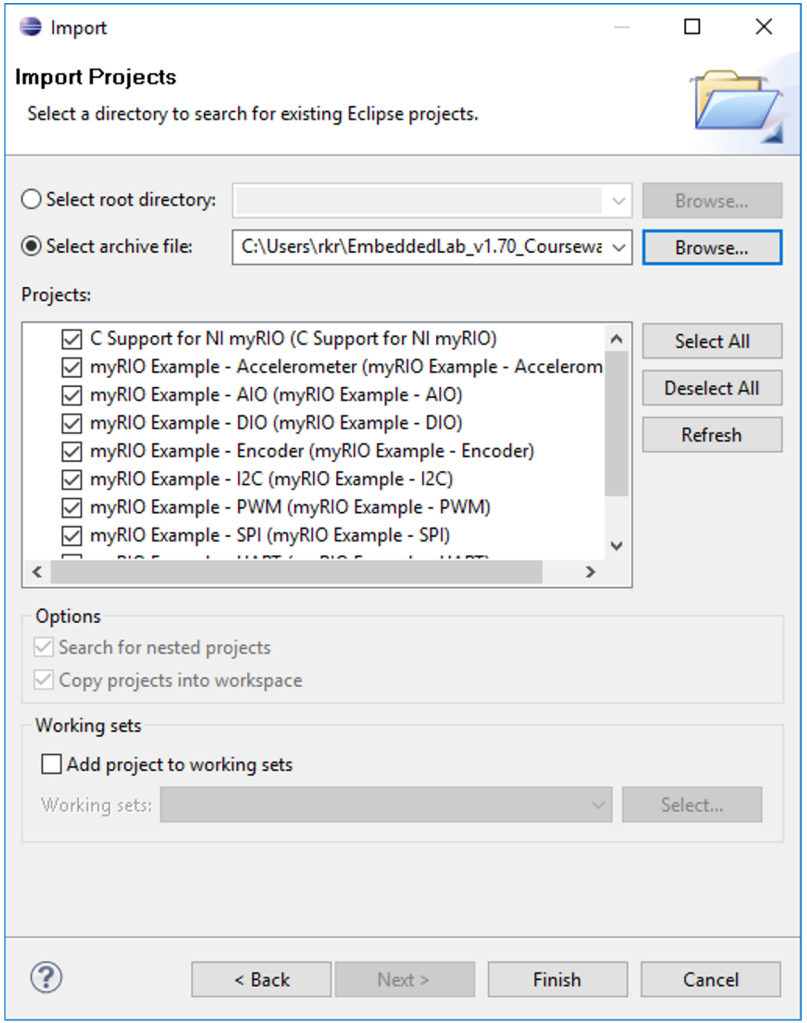
In the Import Projects pane, press “Select archive file” and browse to C_Support_for_NI_myRIO_v1.0.zip Several projects are available for import; the example projects are a useful resource but not needed for this lab, so deselect all example projects, leaving only C Support for NI myRIO and myRIO Template.



Task 2 - Eclipse

4 : Import Template Project

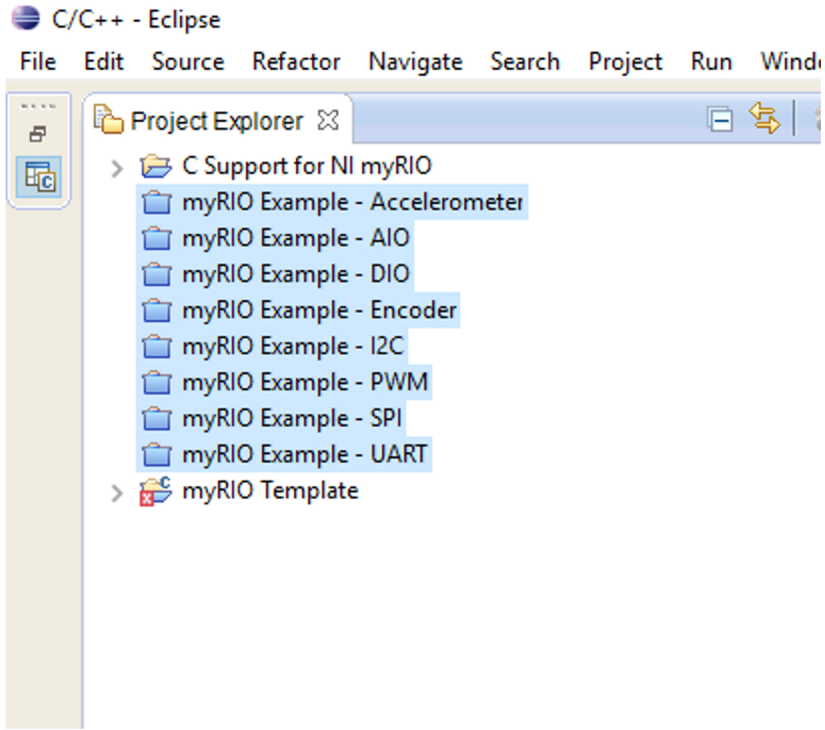
In the Import Projects pane, press “Select archive file” and browse to C_Support_for_NI_myRIO_v1.0.zip Several projects are available for import; the example projects are a useful resource but not needed for this lab, so deselect all example projects, leaving only C Support for NI myRIO and myRIO Template.



Task 2 - Eclipse

4 : Import Template Project

If you chose not to remove the example projects, close them within eclipse.



Task 2 - Eclipse

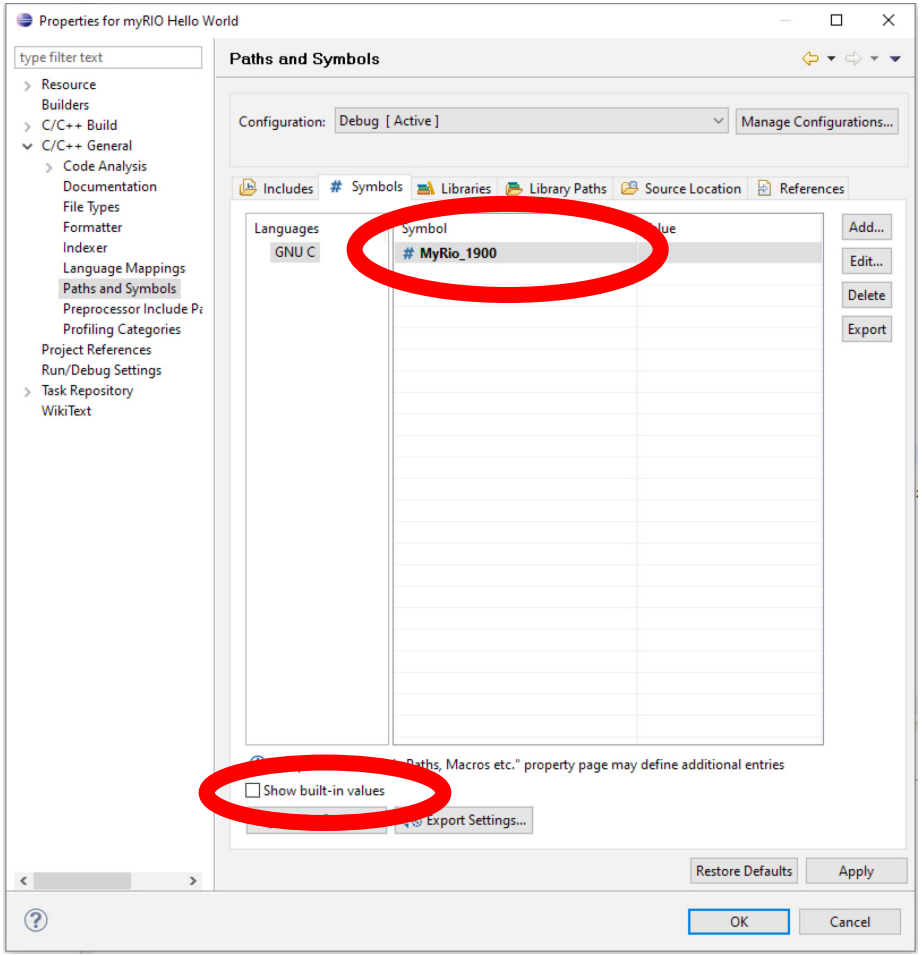
6 : Set Model Number

Set a project-level symbol to specify which model of myRIO you are using. This informs the generated C code about the pin configurations and peripherals.

Right-click on the myRIO Template project and select “Properties” to open project settings.

Navigate to “C/C++ General→Paths and Symbols” and open the #Symbols tab to see project-level symbols that are defined across all source files in the project.

The symbol **MyRio_1900** should be defined by default.



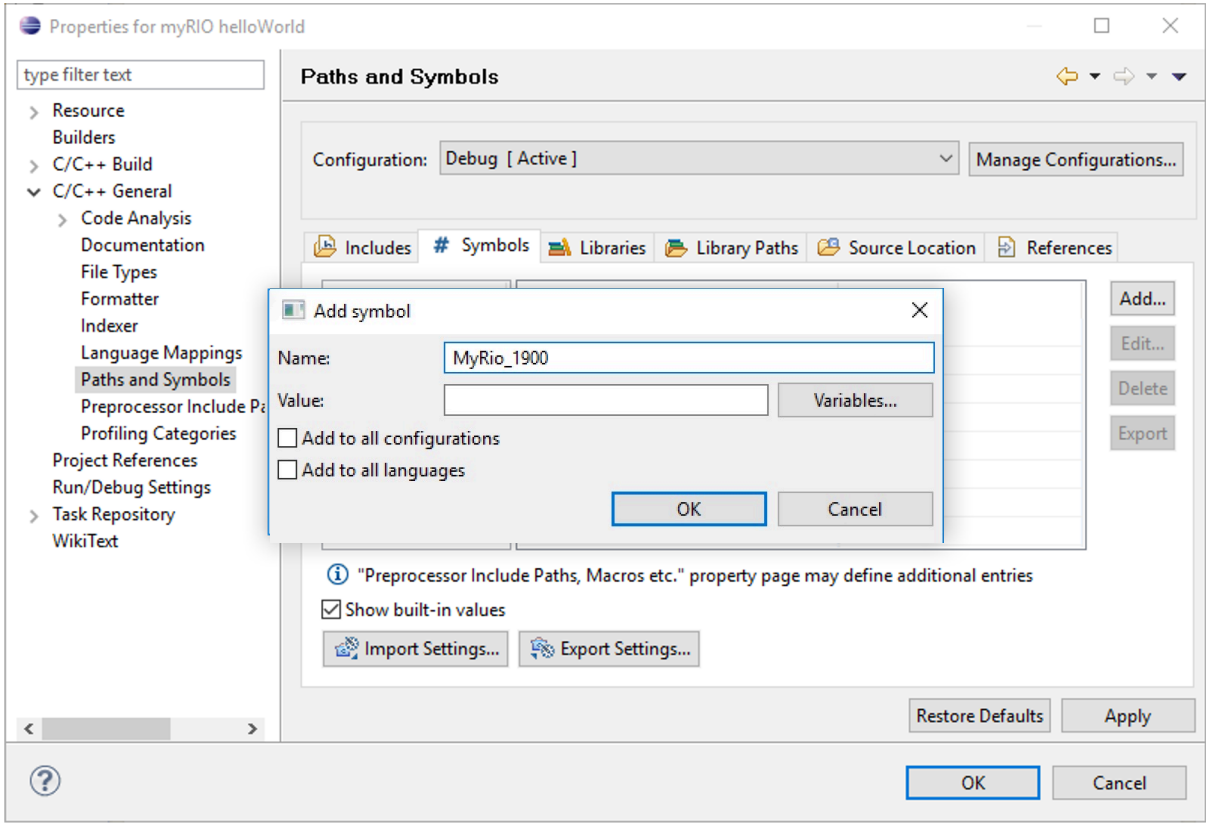
Uncheck “Show built-in values” to show only symbols you have defined

Task 2 - Eclipse

6 : Set Model Number

The symbol **MyRio_1900** should be defined by default; if not, make sure you define it by clicking the Add button.

Make sure to apply your changes on every edit.

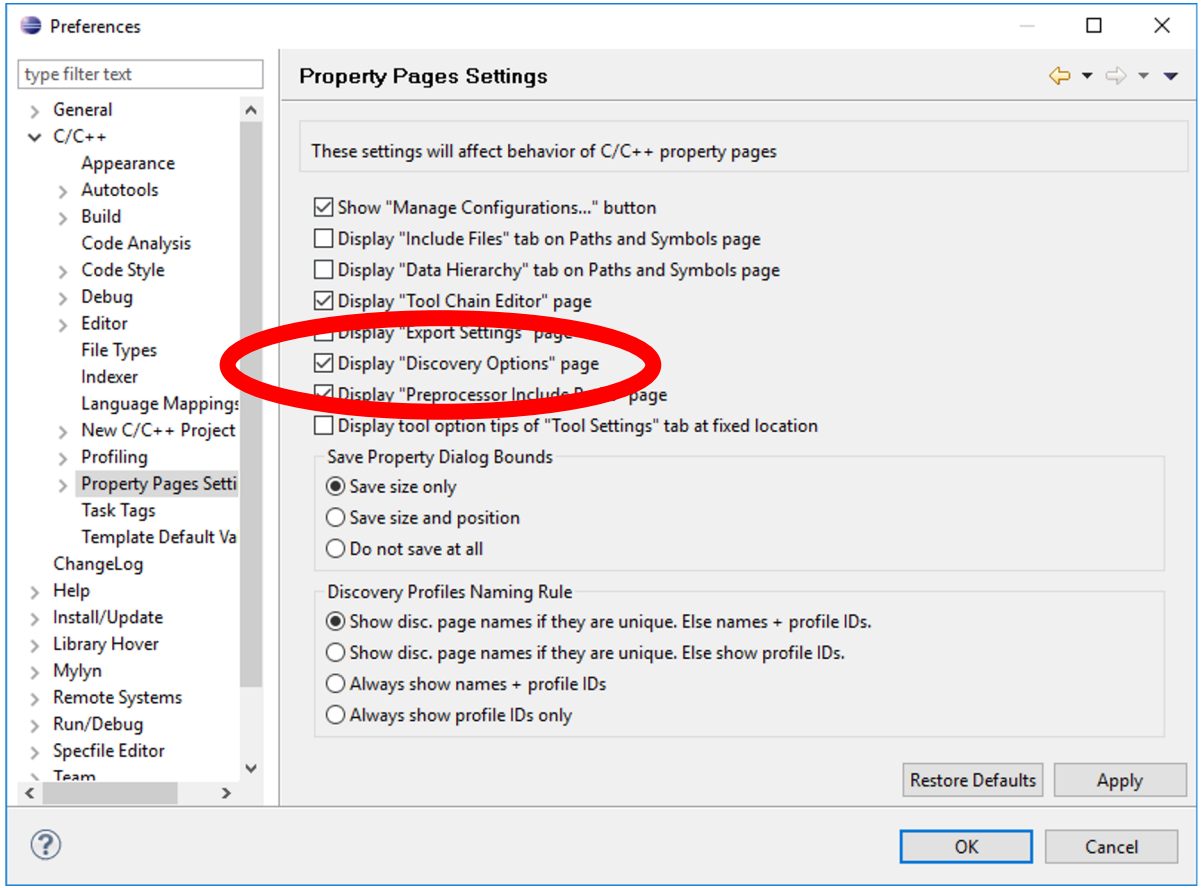


Task 2 - Eclipse

7 : Set Compiler Options

Unhide “Discovery options” by going to Window->Preferences-> C/C++ -> Property Pages Settings and make sure the “Display ‘Discovery Options’ page” is selected.

Make sure to apply your changes on every edit.



Task 2 - Eclipse

7(a) : Set Compiler Settings

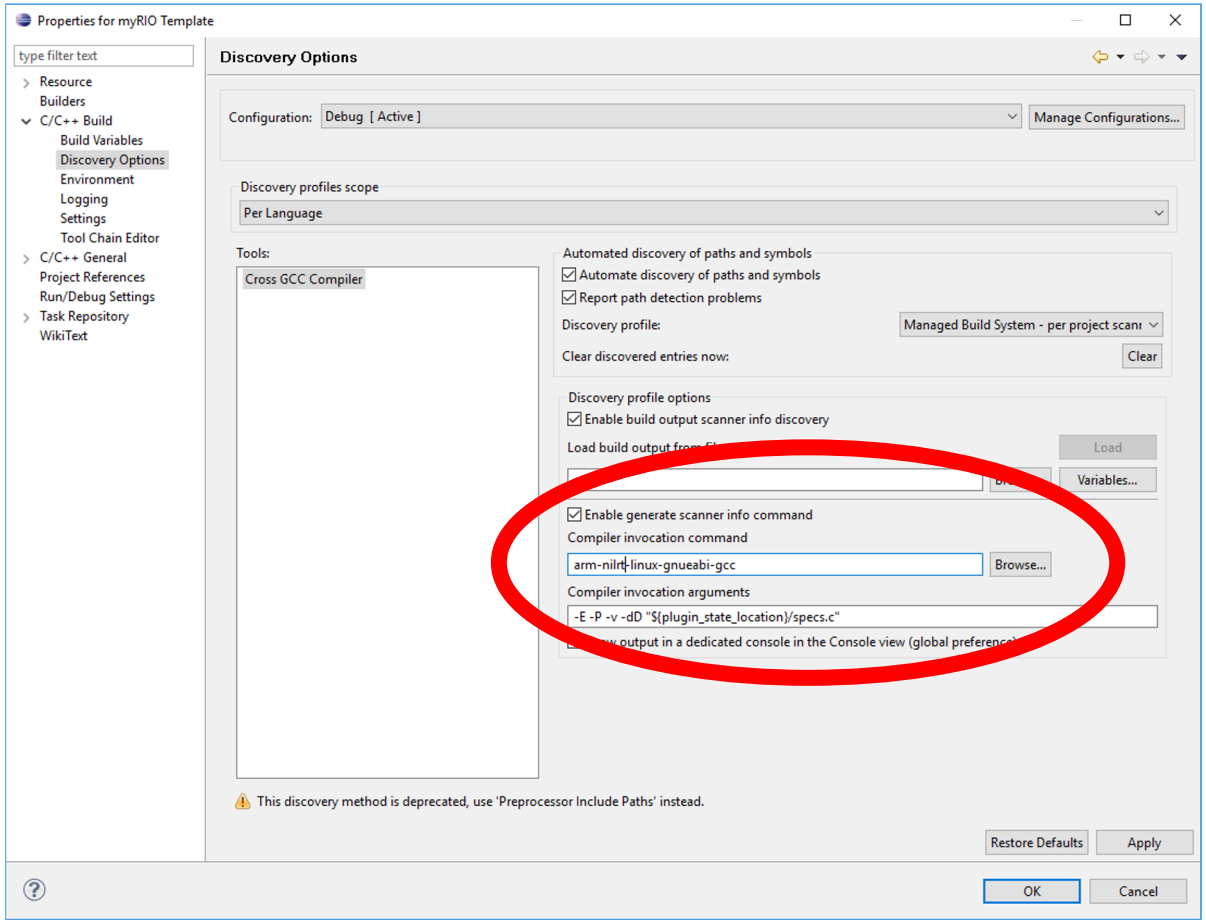
Right click the project and go to properties -> C/C++ Build -> Discovery Options. Change the Compiler invocation command from:

`arm-none-linux-gnueabi-gcc`

to:

`arm-**nilrt**-linux-gnueabi-gcc`

Note: If your screen doesn't look like this, click on "Settings" then click on "Discovery Options" again. Magic!

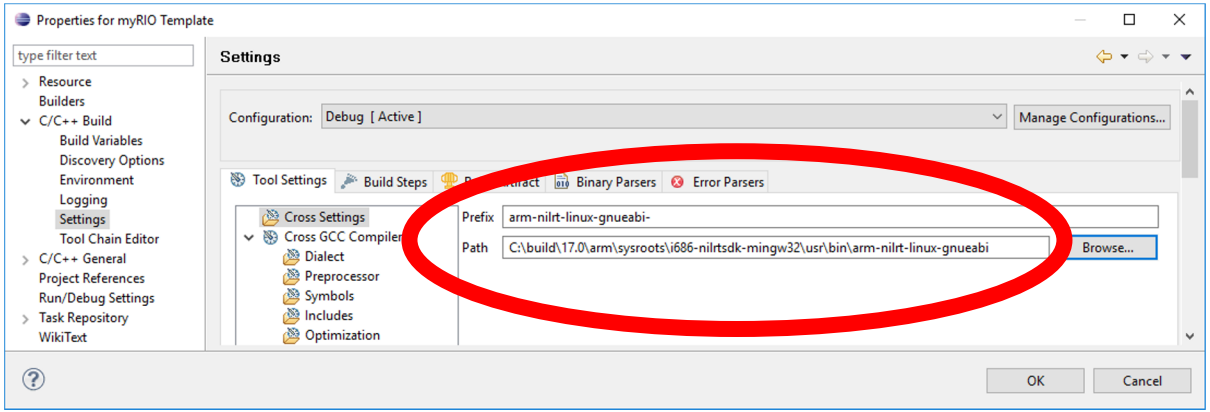


Task 2 - Eclipse

7(b) : Set Compiler Settings

Still under Project properties, go to C/C++ Build -> Settings and under the Tool Settings tab select Cross Settings. Change the prefix to `arm-nilrt-linux-gnueabi-`

and the path to

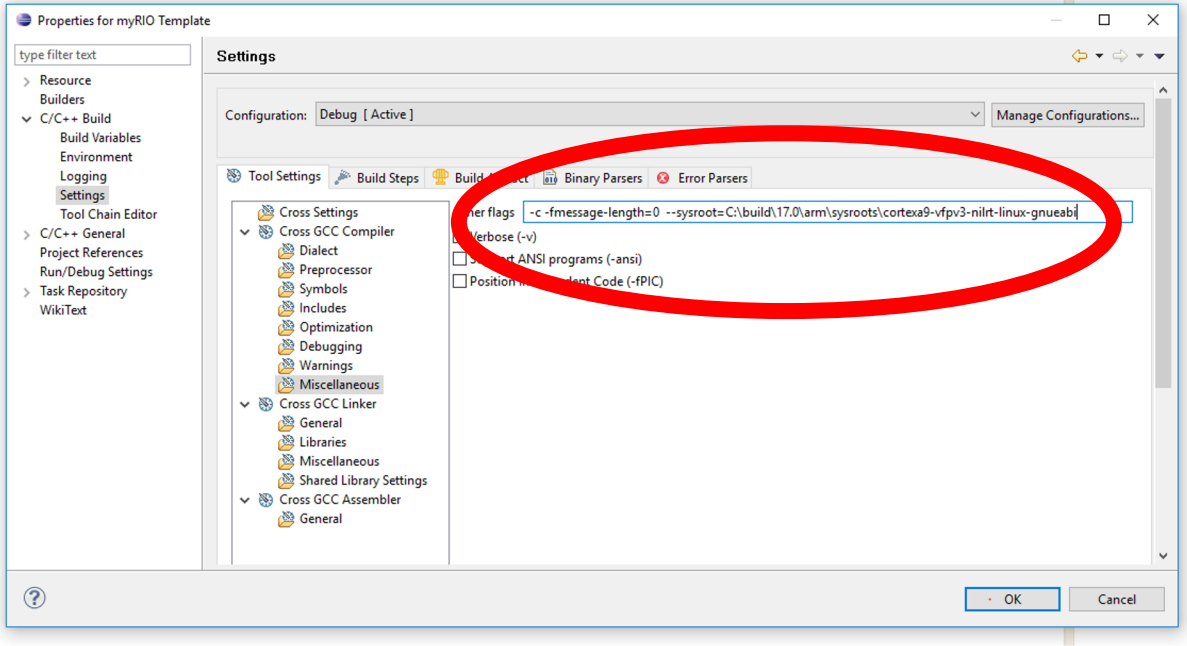



```
C:\build\17.0\arm\sysroots\i686-nilrtsdk-mingw32\usr\bin\arm-nilrt-linux-gnueabi
```

Task 2 - Eclipse

7(c) : Set Compiler Settings

Within the C/C++ Build -> Settings page of the same properties view, under the Tool Settings tab, select Cross GCC Compiler -> Miscellaneous and add the following to the list of other flags :

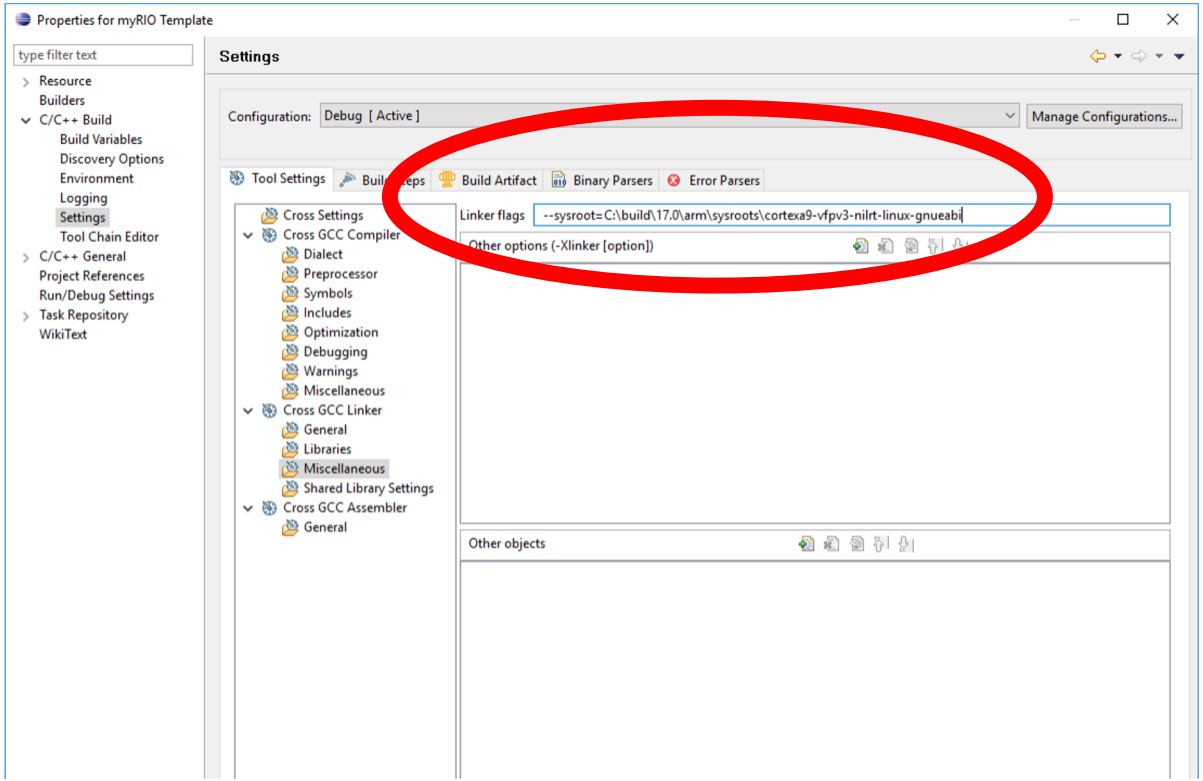



```
--sysroot=C:\build\17.0\arm\sysroots\cortexa9-vfpv3-nilrt-linux-gnueabi
```

Task 2 - Eclipse

7(d) : Set Compiler Settings

Within the C/C++ Build -> Settings page of the same properties view, under the Tool Settings tab, select Cross GCC Linker -> Miscellaneous and add the following to the list of linker flags :



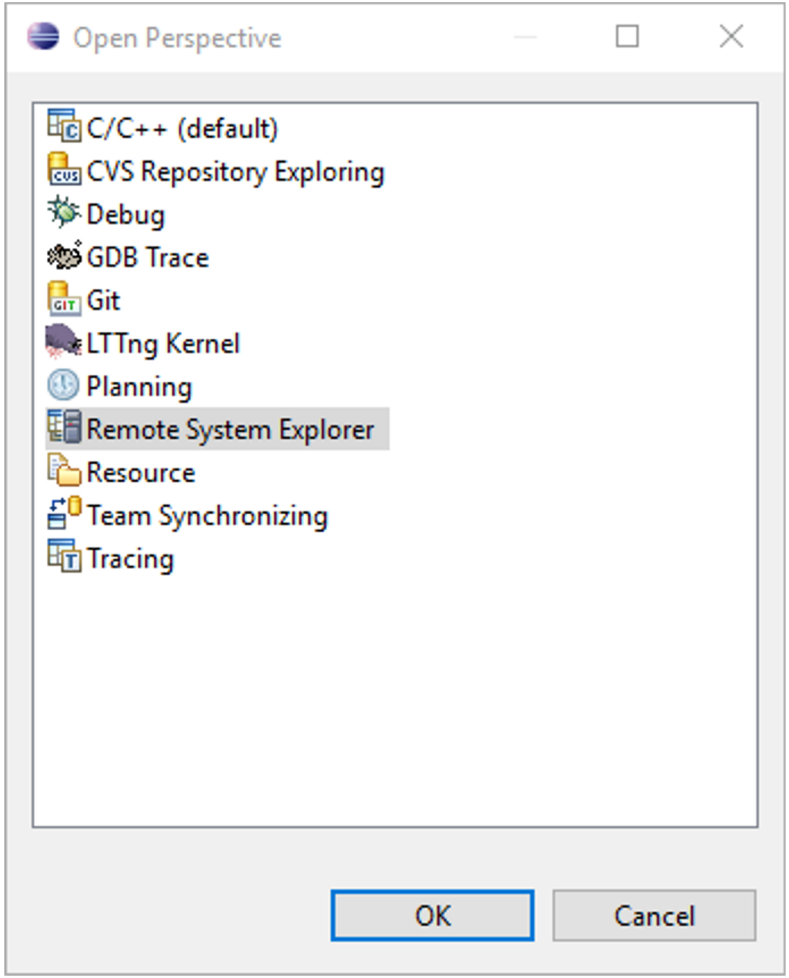

```
--sysroot=C:\build\17.0\arm\sysroots\cortexa9-vfpv3-nilrt-linux-gnueabi
```

Remember to apply your changes!

Task 2 - Eclipse

8 : Create Remote Target

To view connections to remote systems, use the Remote System Explorer (RSE) perspective. To open this perspective, from the top many bar select “Window→Open Perspective→Other” and choose “Remote System Explorer”.

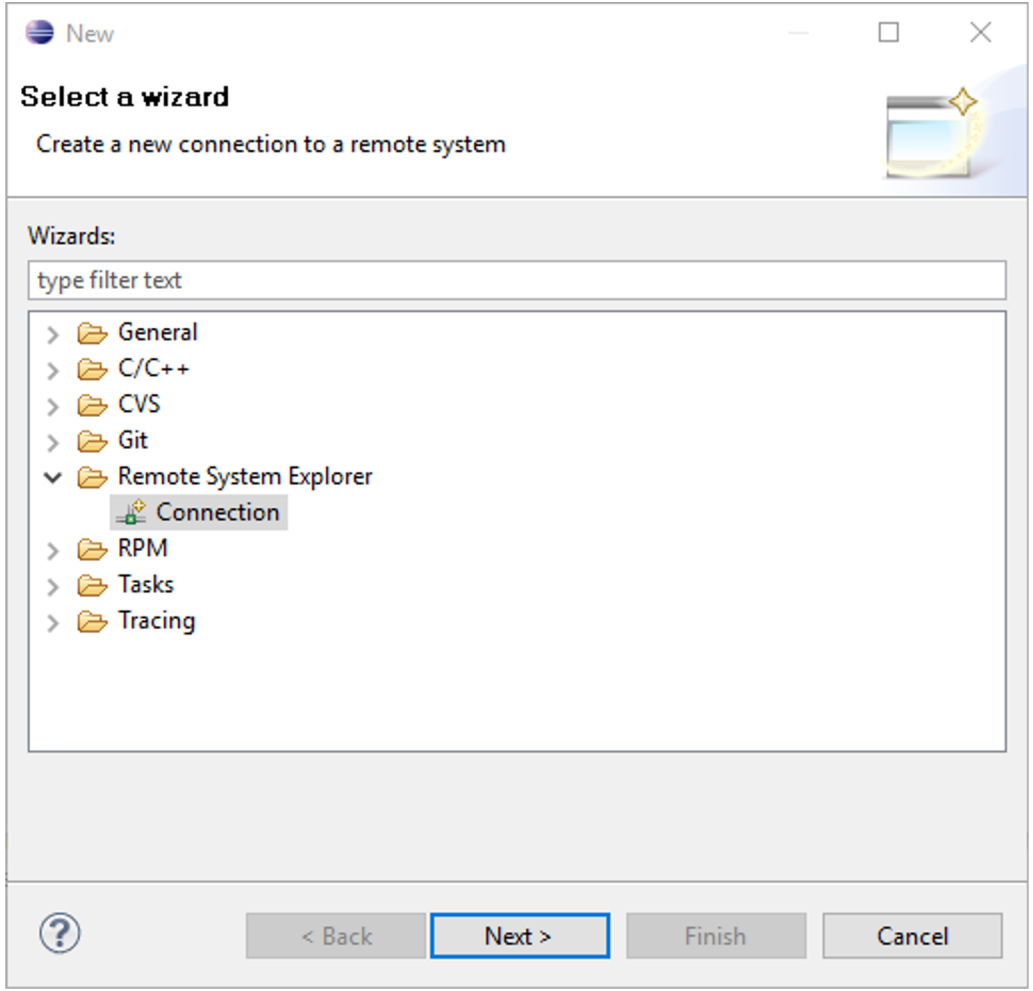


Task 2 - Eclipse

8(a) : Create Remote Target

From the top menu bar select “File→New→Other...” to launch the New Connection wizard

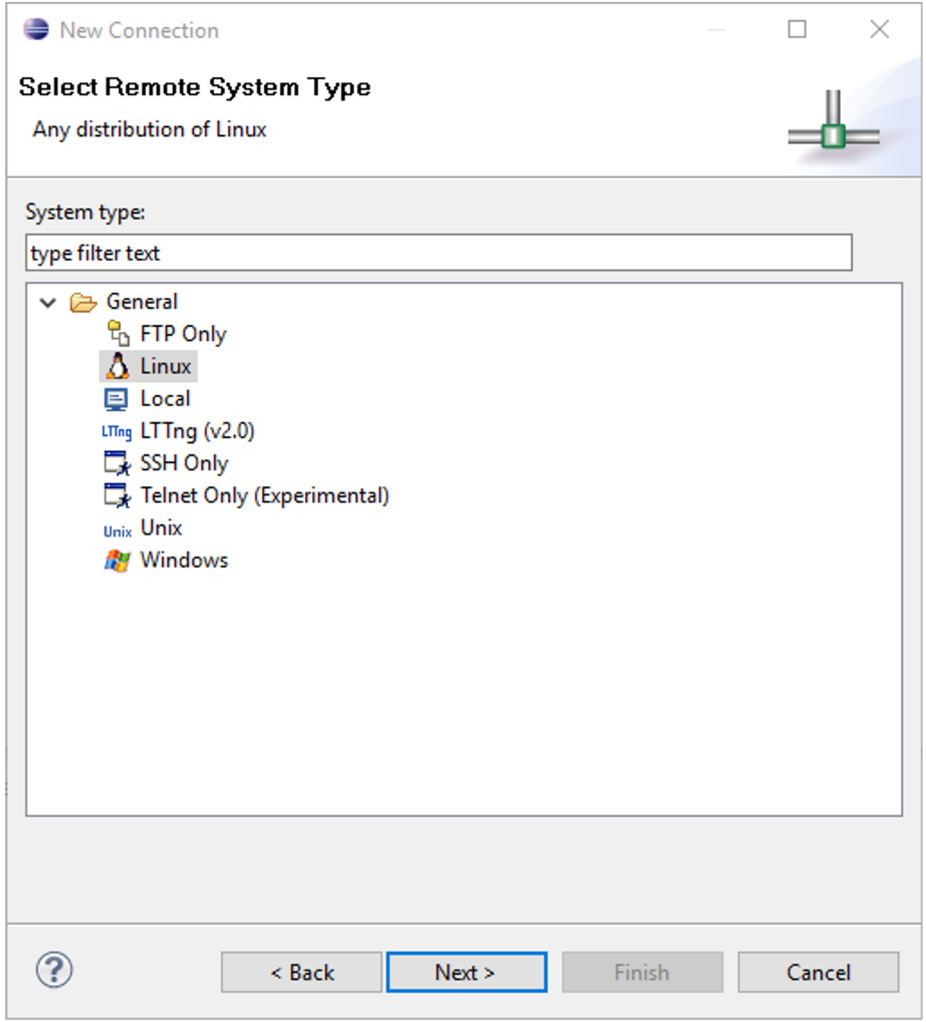
In the Select a wizard panel, select “Remote System Explorer→Connection” and press [Next >] .



Task 2 - Eclipse

8(b) : Create Remote Target

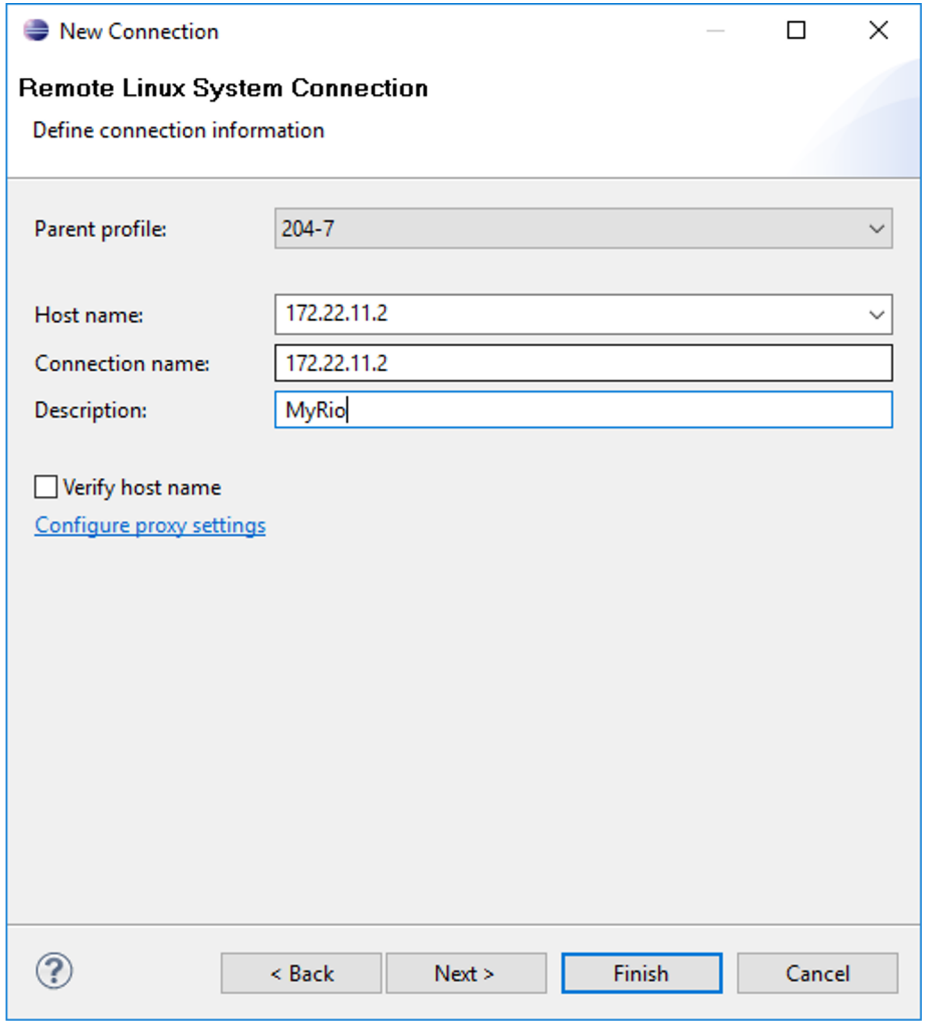
In the Select Remote System Type panel, select “Linux” and press [Next >] .



Task 2 - Eclipse

8(c) : Create Remote Target

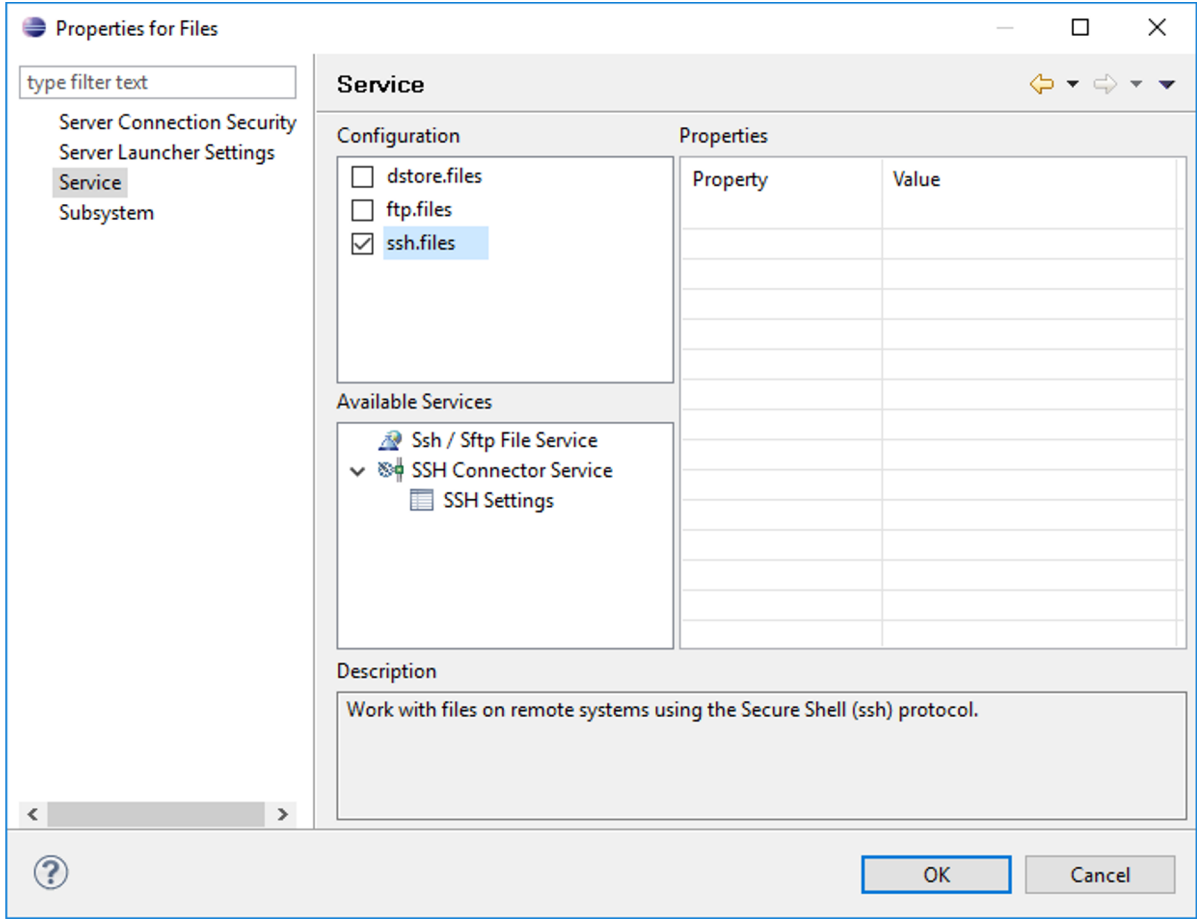
In the Remote Linux System Connection panel, enter the IP address of your myRIO into the Host name field and the name “myRIO” into the Connection name field and press [Next >] .



Task 2 - Eclipse

8(d) : Create Remote Target

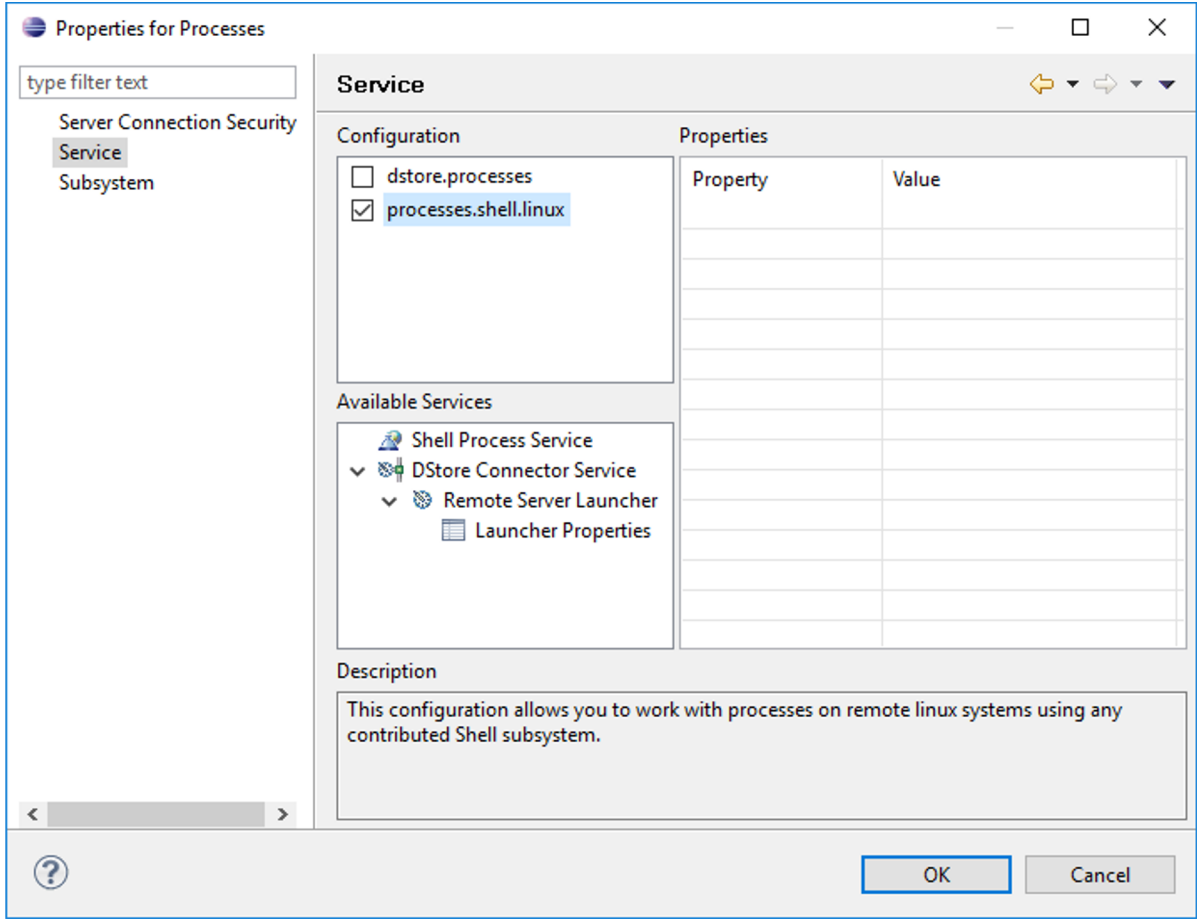
In the Files panel, choose the configuration “ssh.files” to indicate the file system connection protocol. Press [Next >]



Task 2 - Eclipse

8(e) : Create Remote Target

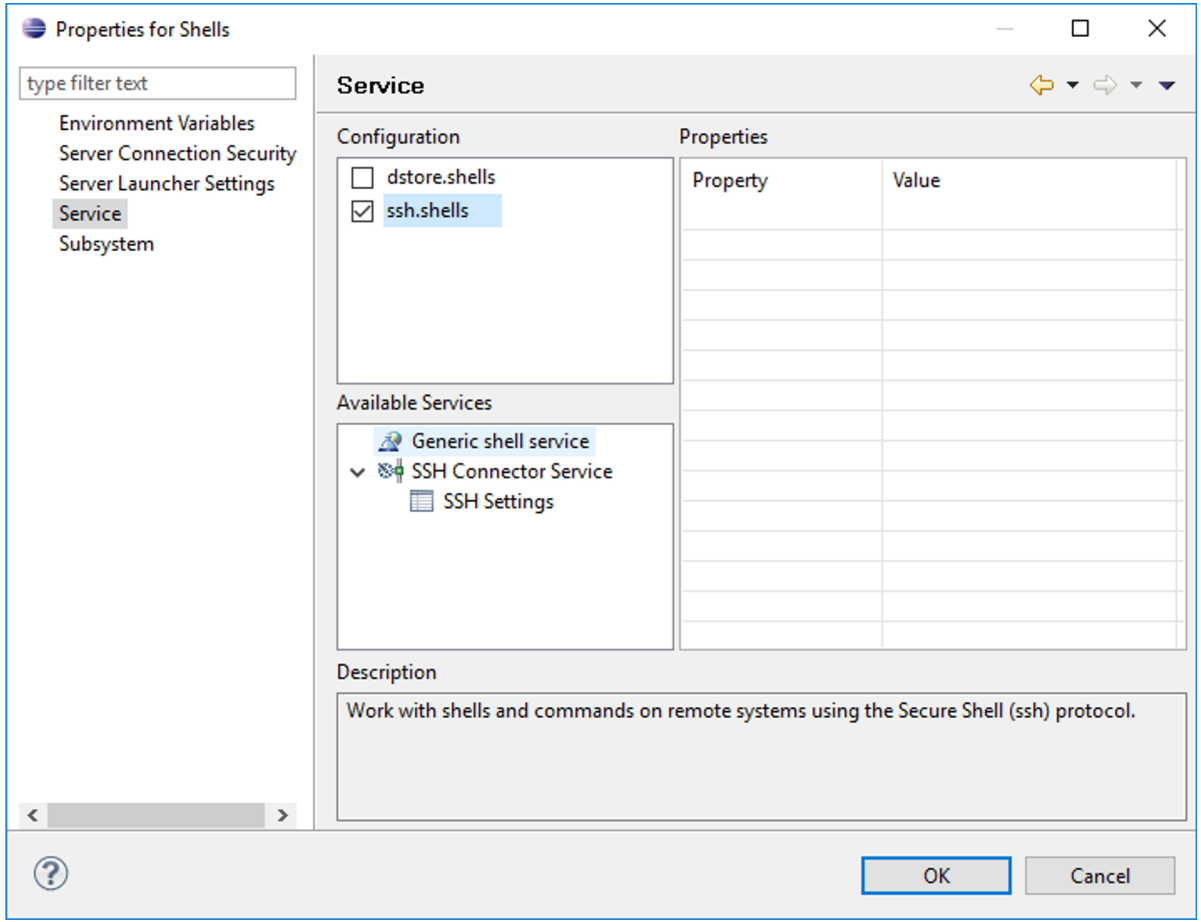
In the Processes panel, choose the configuration “processes.shell.linux” to indicate the Shell process subsystem for connecting to remote processes. Press [Next >].



Task 2 - Eclipse

8(f) : Create Remote Target

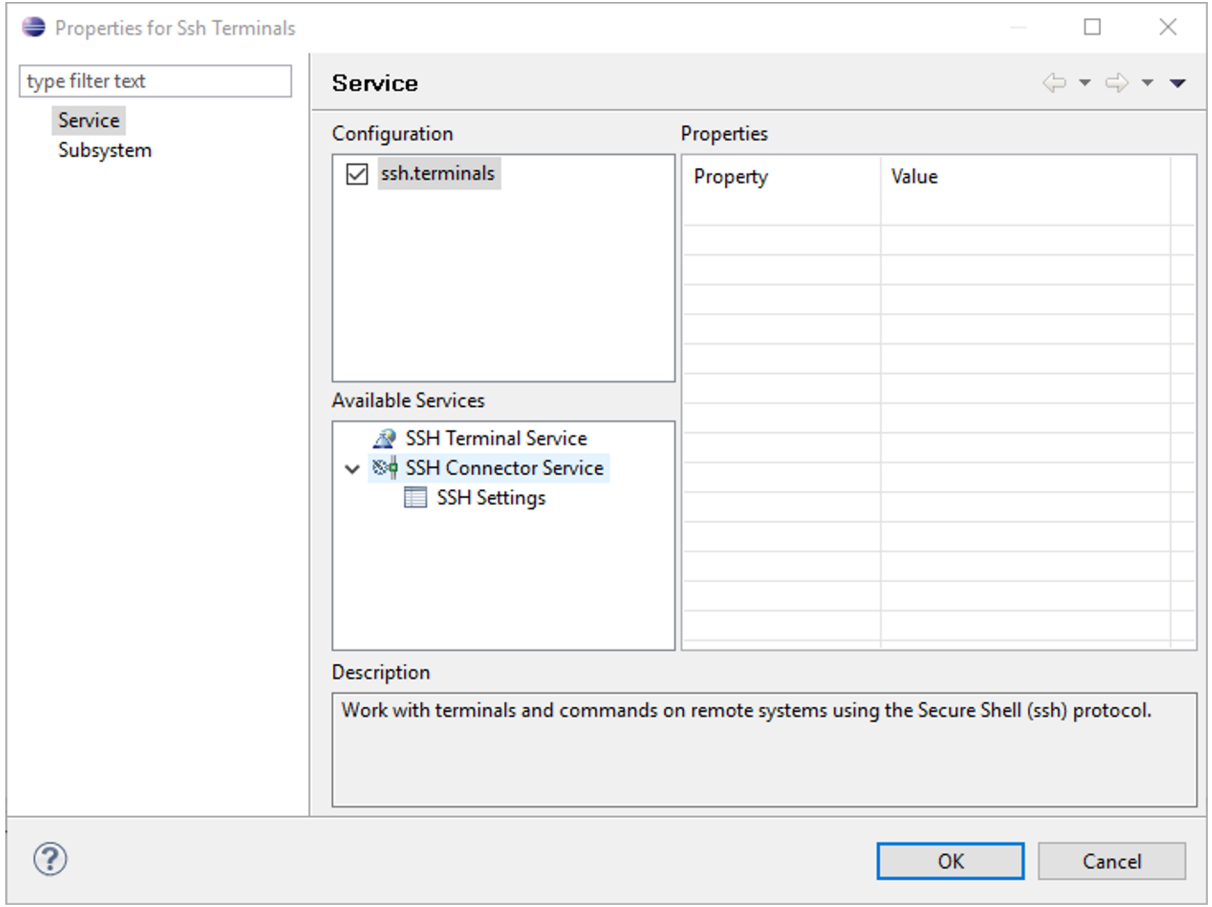
In the Shells panel, choose the configuration “ssh.shells” to indicate the shell and command protocol. Press [Next >].



Task 2 - Eclipse

8(g) : Create Remote Target

In the SSH Terminals panel, choose the configuration ssh.terminals to indicate the terminal protocol. [Next >].



Task 2 - Eclipse

8 : Create Remote Target

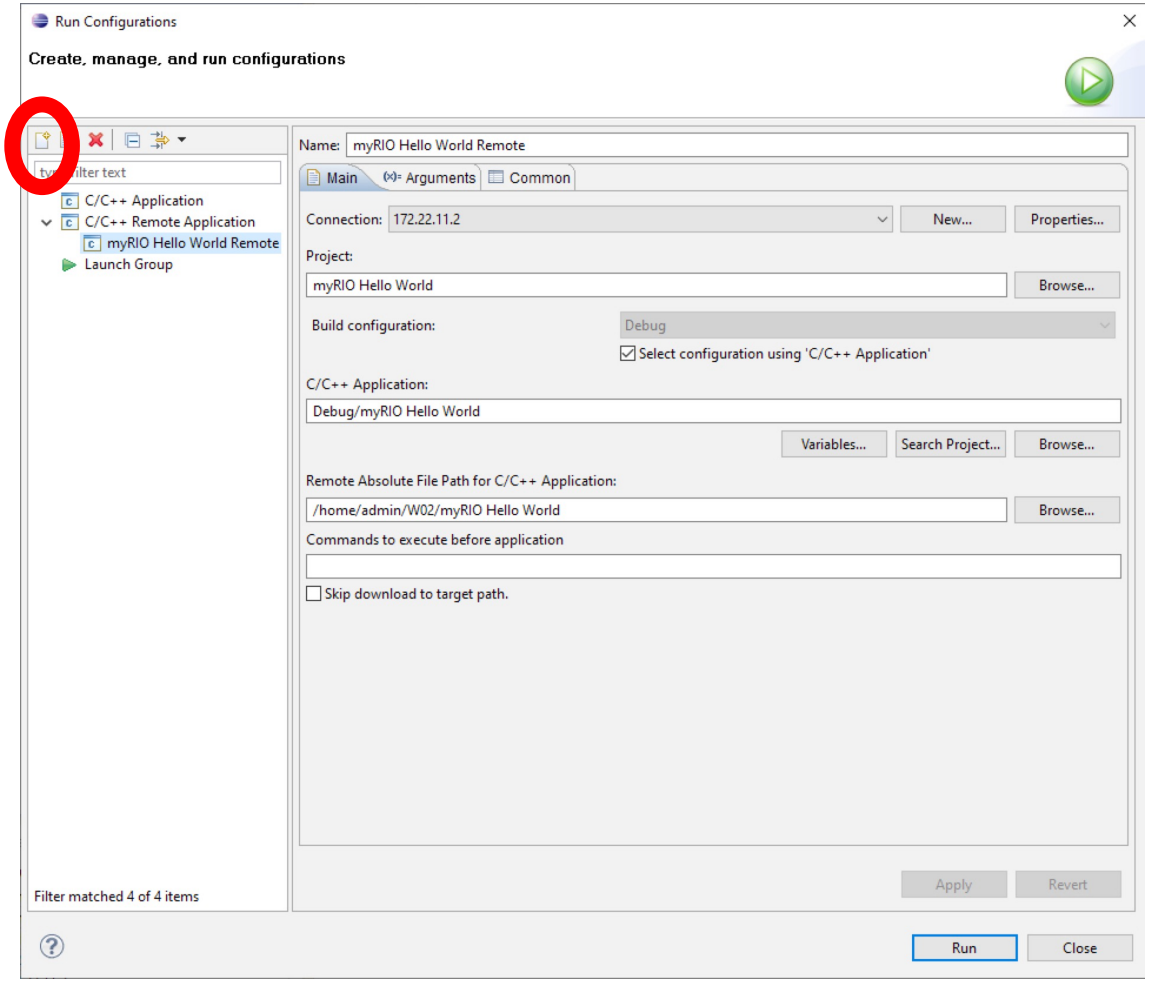
Select Run»Run Configurations to open the Run Configurations dialog box.

Select C/C++ Remote Application in the left pane.

Click the New launch configuration button, circled in the image, to specify settings for running an executable on your target.

Select your target from the Connection pull-down menu.

Click the Browse button beside the Remote Absolute File Path for C/C++ Applications text box to open the Select Remote C/C++ Application File dialog box.

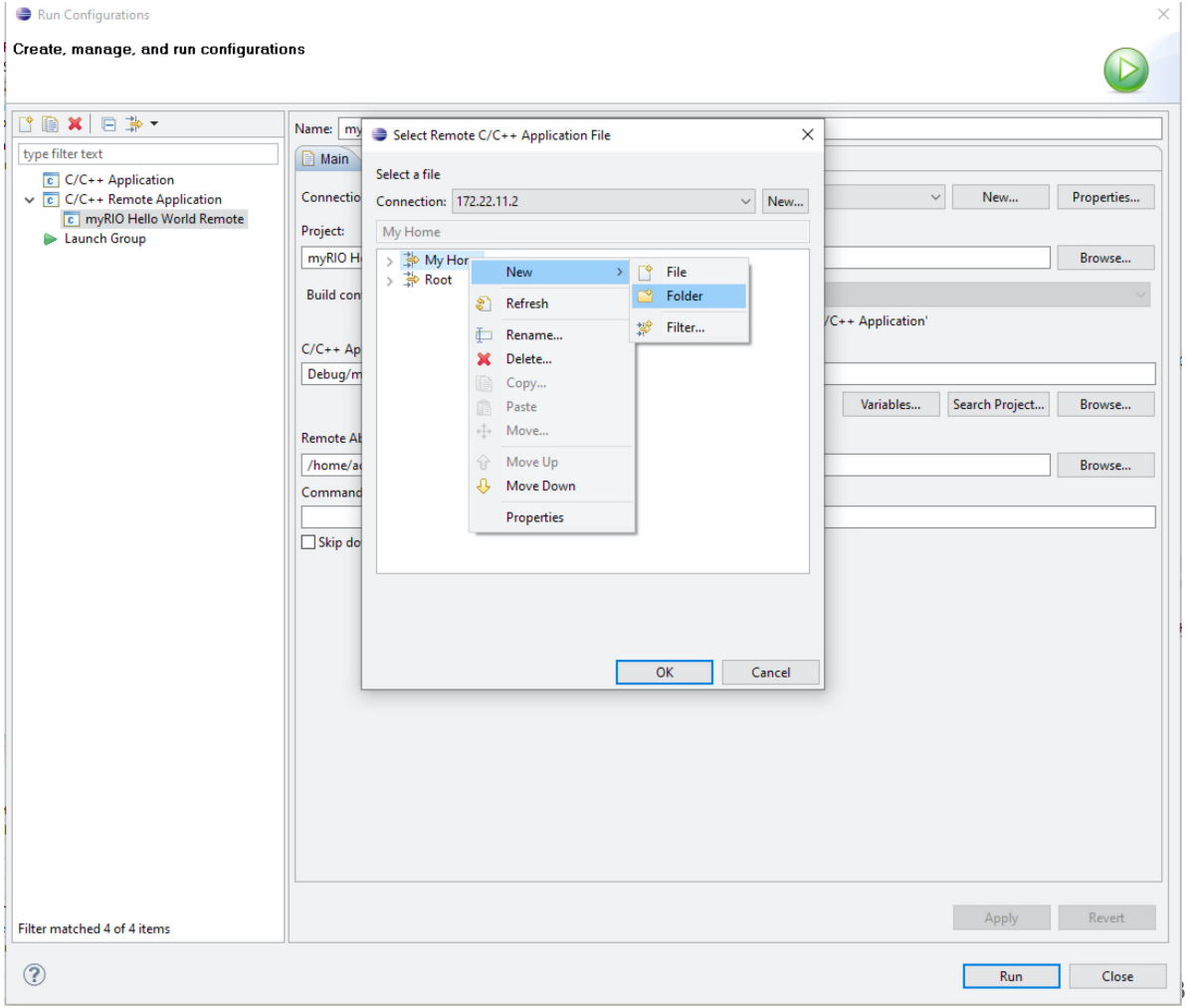


Task 2 - Eclipse

8 : Create Remote Target

Right-click the My Home directory in the list box and select New»Folder to create a folder on the target in which to place a copy of the executable.

Append your project name (or whatever name you wish to assign to the executable) to the file path populated in the Remote Absolute File Path for C/C++ Applications text box..

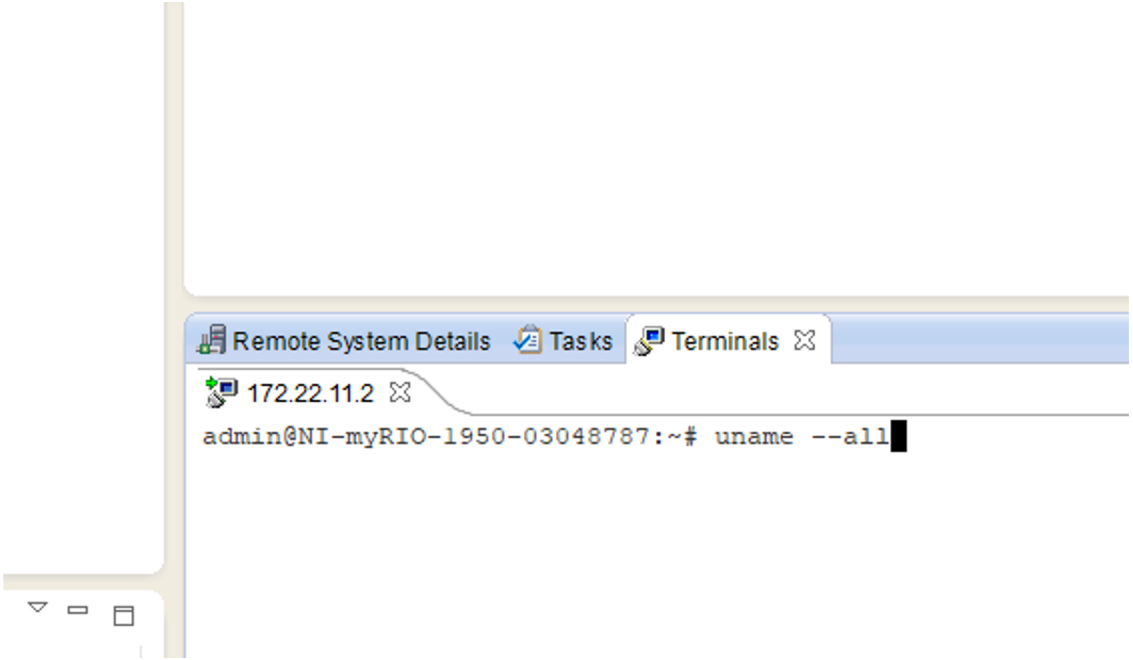


Task 2 - Eclipse

9 : Connect to MyRio

Next, launch a terminal window which will allow you to interact with the Linux shell on myRIO. Expand the myRIO target, right-click on Ssh Terminals, and select “Launch Terminal”. A terminal pane should appear in the bottom part of the perspective. Standard output messages are displayed on this terminal, and you may interact with the shell as with any Linux machine. Enter the command `uname --all`

and verify from the response that all is working.

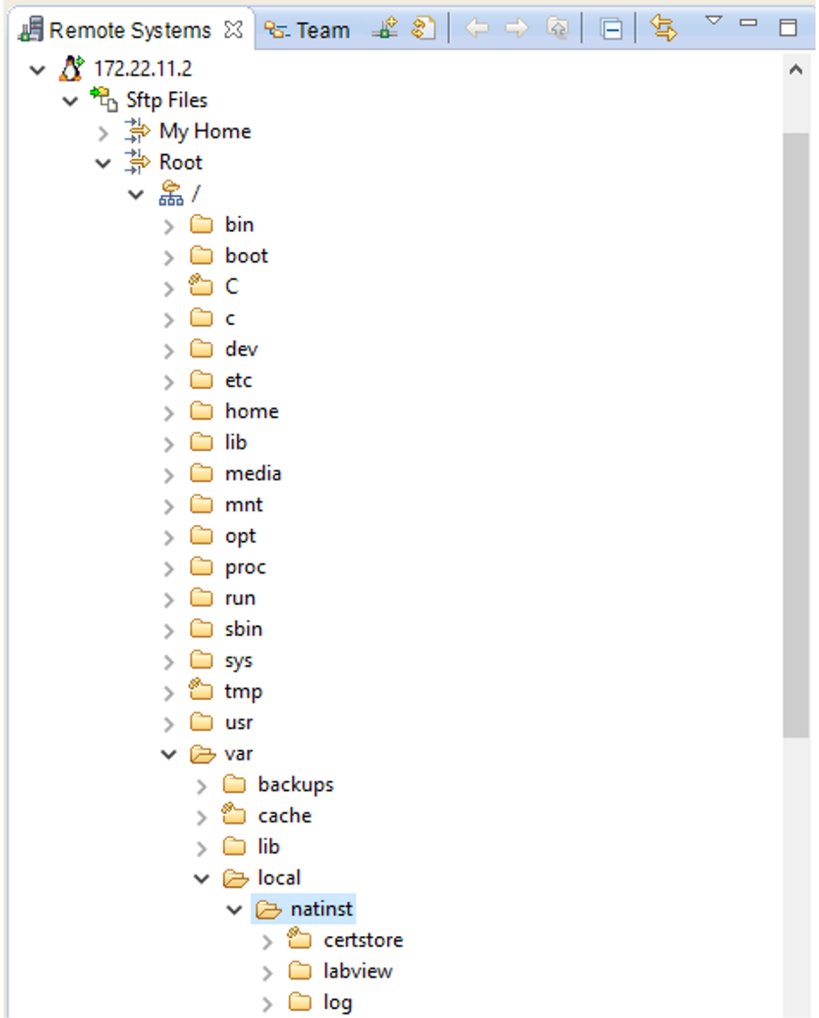


Task 2 - Eclipse

10 : Transfer FPGA Fixed Personality

The FPGA on myRIO may be configured with an FPGA fixed-personality stored in a LabVIEW bitfile (.lvbitx) file. In the Remote Systems pane, expand “myRIO→Sftp Files” to see the filesystem on myRIO. Navigate to the folder /var/local/natinst/

If there is already a subfolder called “bitfiles”, you do NOT need to create it and can skip to the next step



Task 2 - Eclipse

10 : Transfer FPGA Fixed Personality

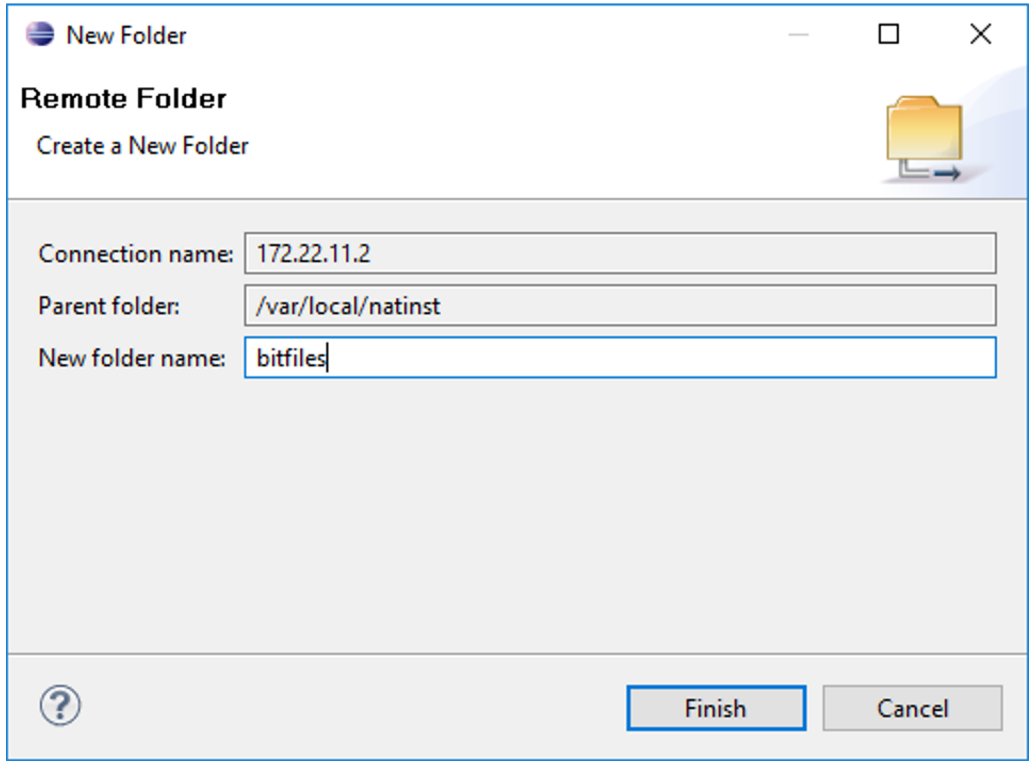
Right-click on the folder “natinst” and select “New → Folder”. Name the folder “bitfiles”.

Right-click on the newly created “bitfiles” folder and select “Export from Project...”.

Navigate to the source folder C Support for NI myRIO\source and check to include `NiFpga_myRio1900Fpga10.lvbitx`

No other files or folders should be checked. Verify that the destination folder indicates the “bitfiles” folder.

Click [Finish] to transfer the file, and verify the file now appears in the Remote Systems pane under the “bitfiles” folder.



Task 2 - Eclipse

11 : Write Hello World

Most programming tasks are performed in the C/C++ perspective. To open this perspective, from the top menu bar select “Window→ Open Perspective→Other” and choose “C/C++”.

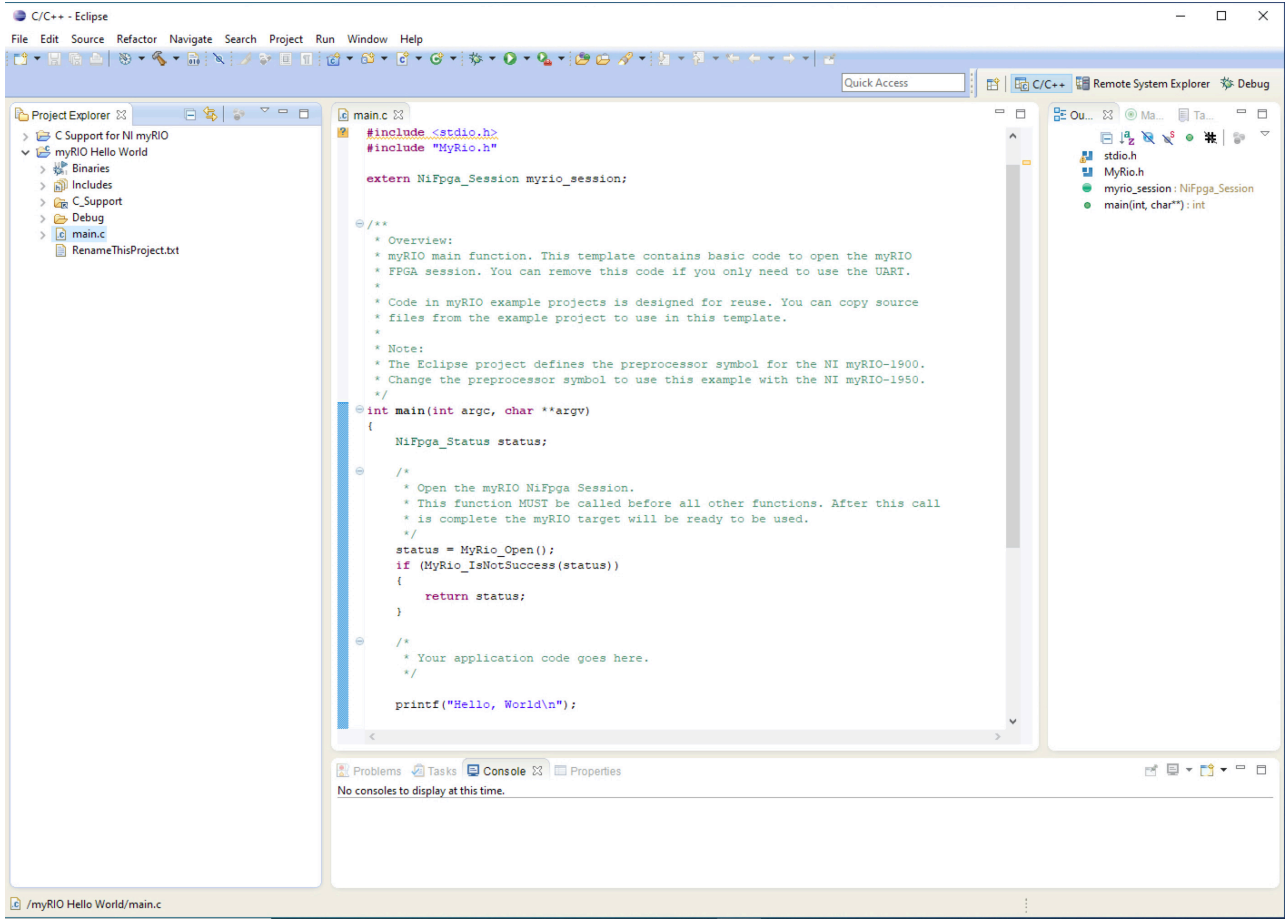
Expand the myRIO Hello World project and open the file main.c Add the statement

printf("Hello, World\n");

after the comment

/* Your application code goes here.*/.

To compile the source code, from the top menu bar select “Project→Build Project”. The Problems tab at the bottom of your workspace will show the results of the compilation – if there are errors (hopefully just typos), you must resolve them before continuing to the next step



Task 2 - Eclipse

13 (a,b) : Create a Debug Configuration

Set the Name to “myRIO Hello World”.

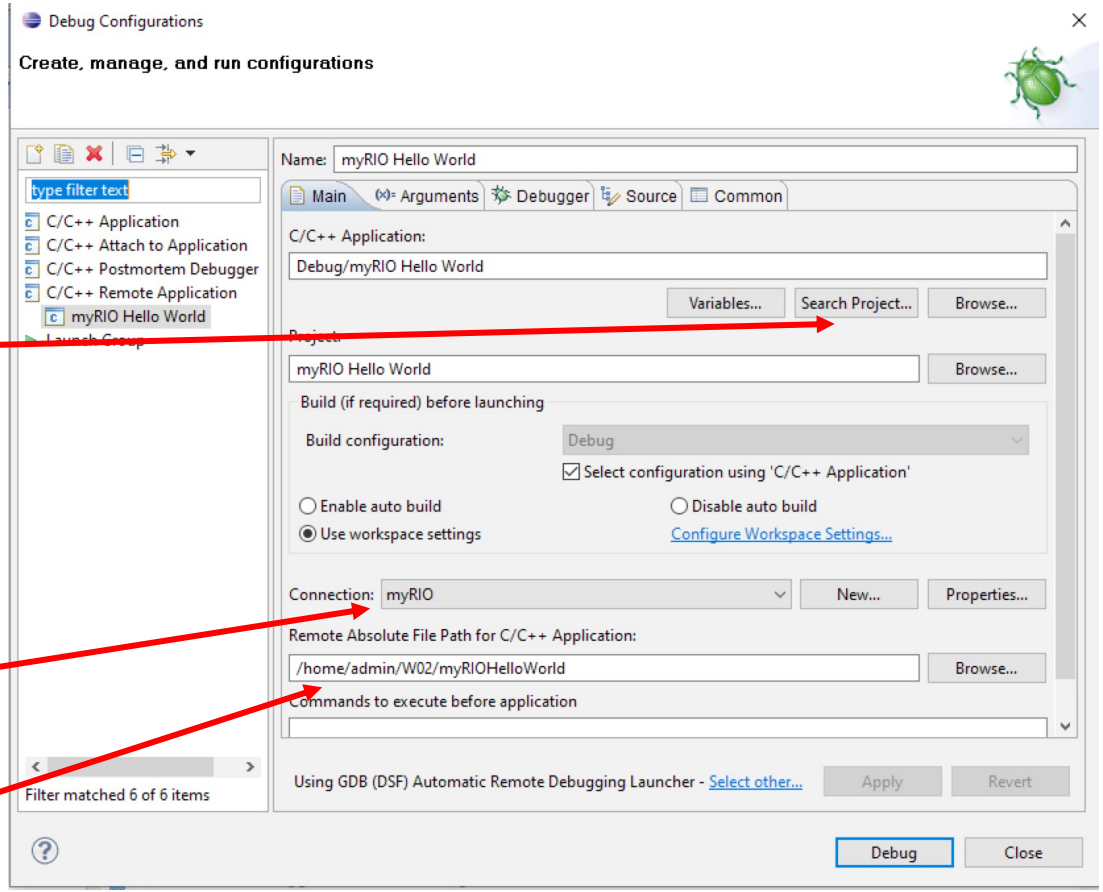
In the Main pane: Select the application to run by pressing [Search Project]... under C/C++ Application.

If your program compiled successfully, the project myRIO Hello World will appear in the Program Selection dialog. Select it and press [OK] .

For connection profile, ensure “myRIO” is selected.

Set Remote Absolute File Path to /home/admin/W02/myRIOHelloWorld

No spaces in filename!



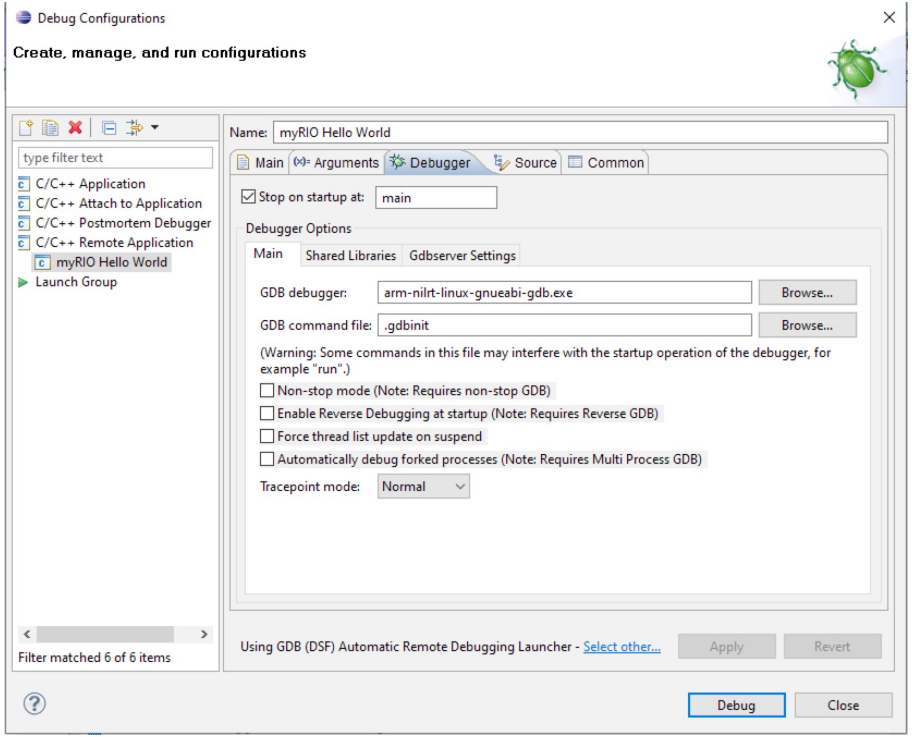
Task 2 - Eclipse

13 (c) : Create a Debug Configuration

In the Debugger tab, ensure GDB Debugger is set to:

C:\build\17.0\arm\sysroots\i686-nilrtsdk-mingw32\usr\bin\arm-nilrt-linux-gnueabi\arm-nilrt-linux-gnueabi-gdb.exe

Press [Apply] followed [Close] to save settings and close the configuration window

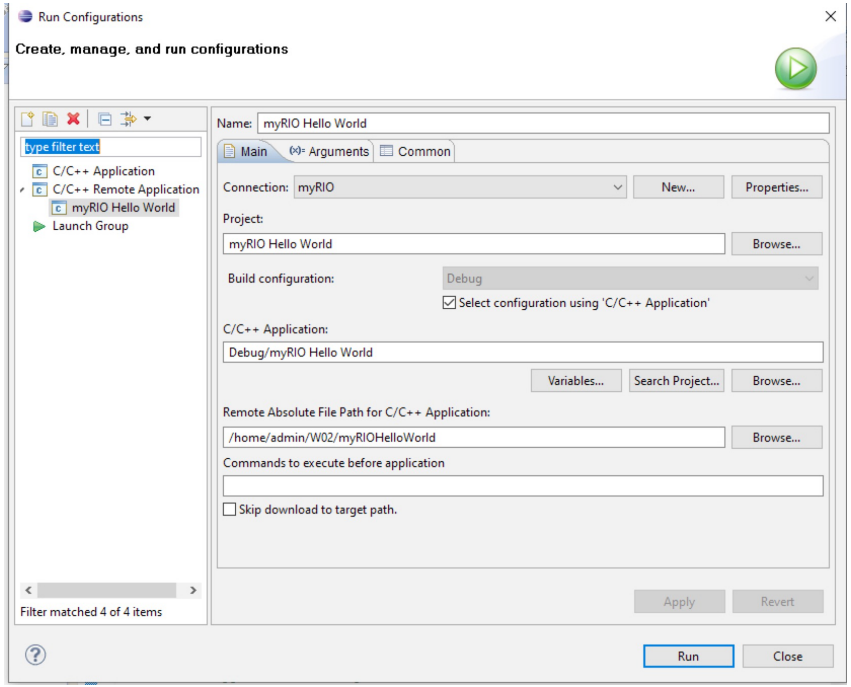


Task 2 - Eclipse

14 : Run your code

From the top menu bar, select “Run→ Run Configurations...” to open the Run Configurations window.

In the left navigation pane, select the configuration you created in the previous step, and press Run to upload and execute your code on the myRIO.



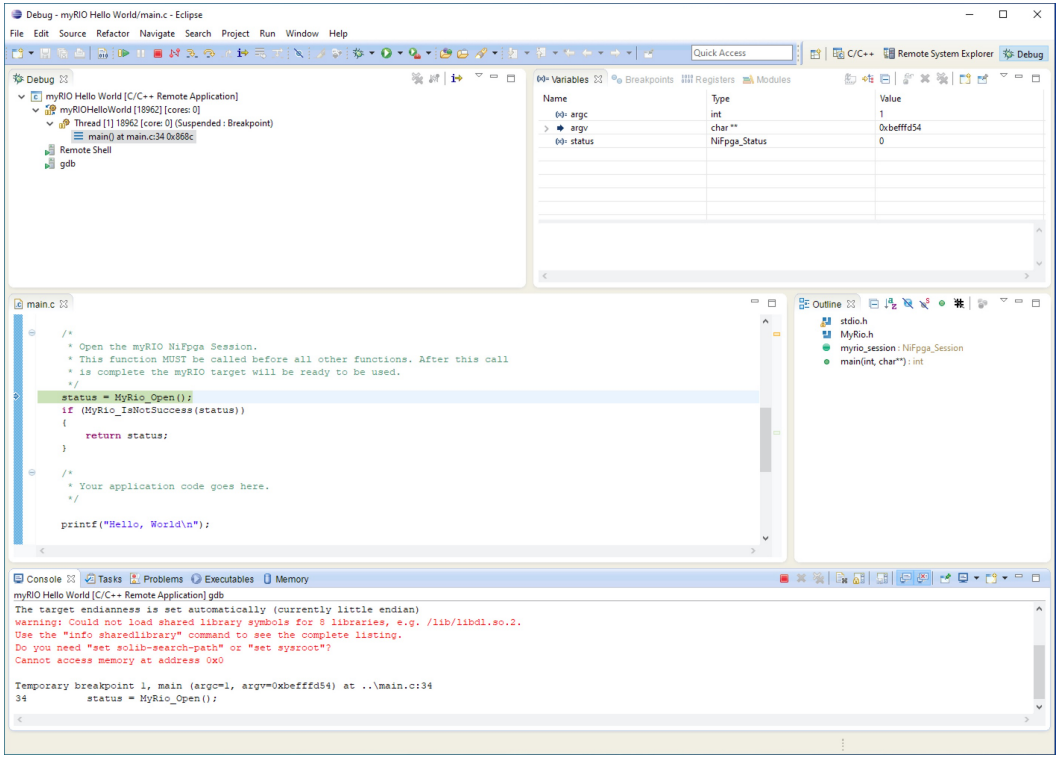
If everything is working you should see “Hello, World” appear in the Console in the Eclipse IDE. Congratulations, you have just executed your first piece of code on an embedded system!

Task 2 - Eclipse

15 : Launch a Debug session

From the top menu bar, select “Run→Debug Configurations...” to open the Debug Configurations window. In the left navigation pane, select the debug configuration you created in the previous step, and press [Debug] to launch a debug session. Click [Yes] to open the Debug perspective if asked

The debugger will stop at the first line of the main() function. To execute the first line, from the top menu bar select “Run→Step Over”. Keep stepping over the instructions until the console displays the message. The console should display the message, “Hello, World!”. To complete execution of your application, from the top menu bar select “Run→Resume”.



If the debugger exits successfully and you see the printed message on the console, then your code has been correctly cross-compiled, downloaded to the target, and remotely debugged.

Task 2 - Eclipse

16 : Write an I/O Application

Modify main.c to access the hardware peripherals on myRIO. At the top of the source file, include the myRIO header, and revise the main() function to open turn on an LED. Add the line

```
extern NiFpga_Session myrio_session;
```

immediately following the `#include` statements, and add the following line after your print statement:

```
printf("Hello, World\n");

/* Set the LED register to 1, turning on LED 1 */

status = NiFpga_WriteU8(myrio_session , DOLED30 , 0x01);
```

```
status = NiFpga_WriteU8(myrio_session, NiFpga_MyRio1900Fpga10_ControlU8_DOLED30, 0x01) ;
```

When you run this program you should see LED 0 flash dim for a fraction of a second.

Note: If the Eclipse Problems pane indicates an error that a symbol (such as DOLED30) cannot be resolved, this is actually a problem with the built-in Code Analyzer. Open the Console pane to see the output of the cross-compiler – if it indicates the program was built successfully, then you may ignore the unresolved symbol warning in the Problems pane.

TASK 2 CHECK!

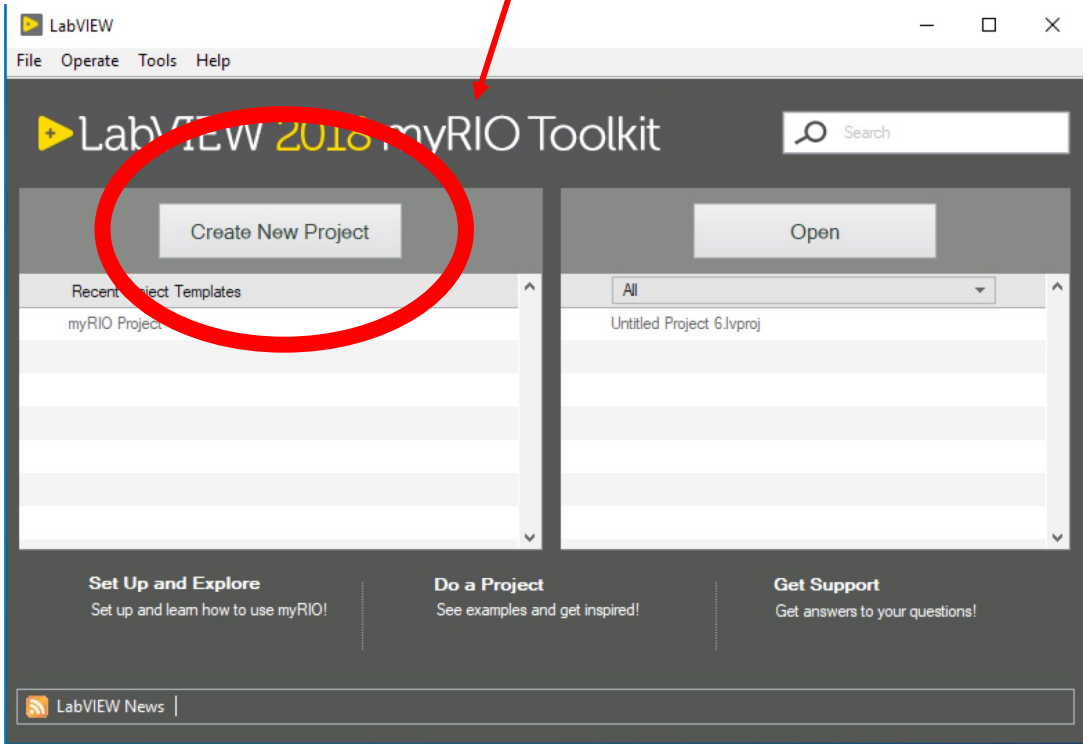
Task 3 - LabVIEW

May also be called “LabVIEW Robotics 2018”

3 : Create a Project

From the top menu of the Getting Started dialog, select “File→ Create Project...” to launch the Create Project Wizard. Navigate to “Templates → myRIO→myRIO Project” and select [Next] .

Or click the create project button on the LabView opening screen.

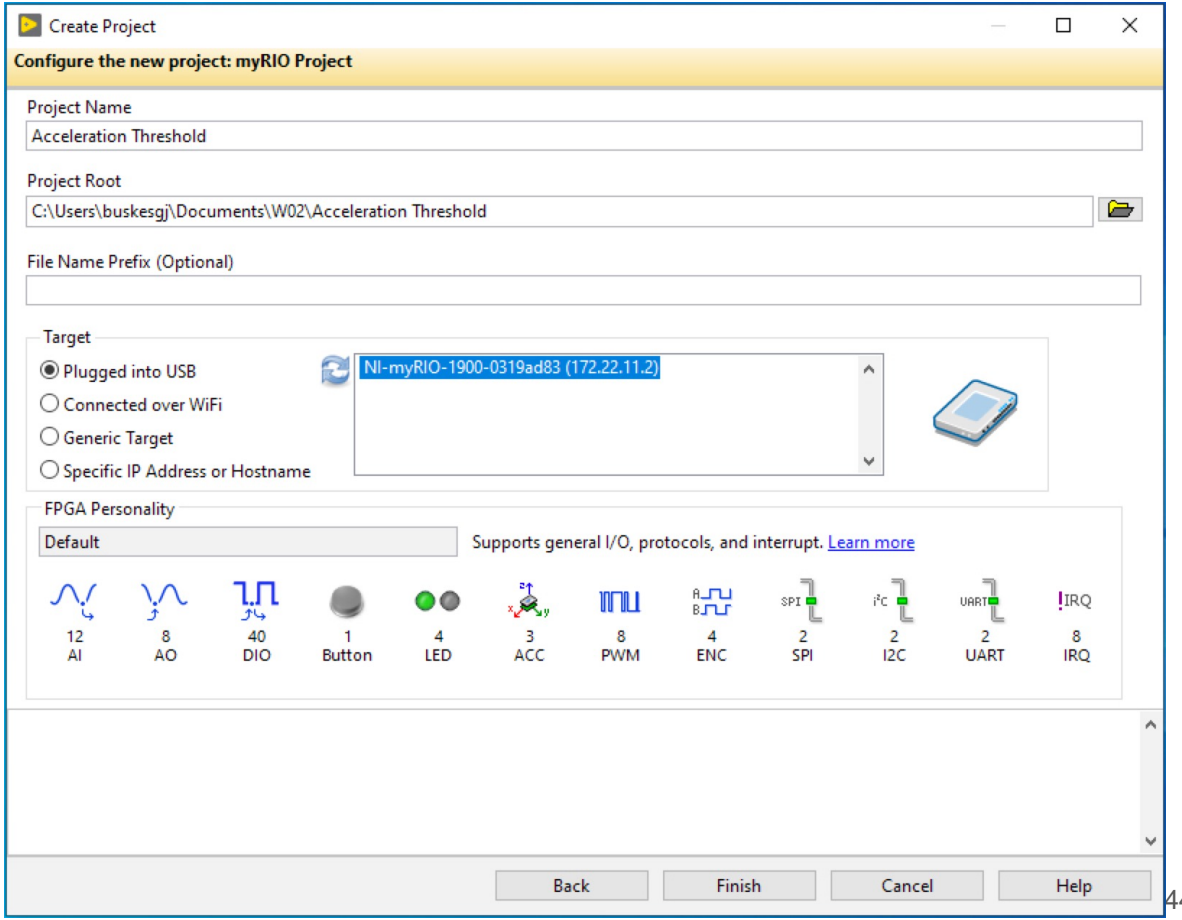


Task 3 - LabVIEW

3 : Create a Project

In the Configure the new project dialog, name your project HelloWorld and choose a location to save your project files. Your myRIO should automatically appear under the Select Target pane – if not, first verify that you can see your myRIO in NI MAX, and re-run the wizard.

Select [Finish] to complete the Create Project Wizard. The Project Explorer will automatically open with your new project. The Project Explorer is used to connect to myRIO and organize VIs that will run on the target processor.



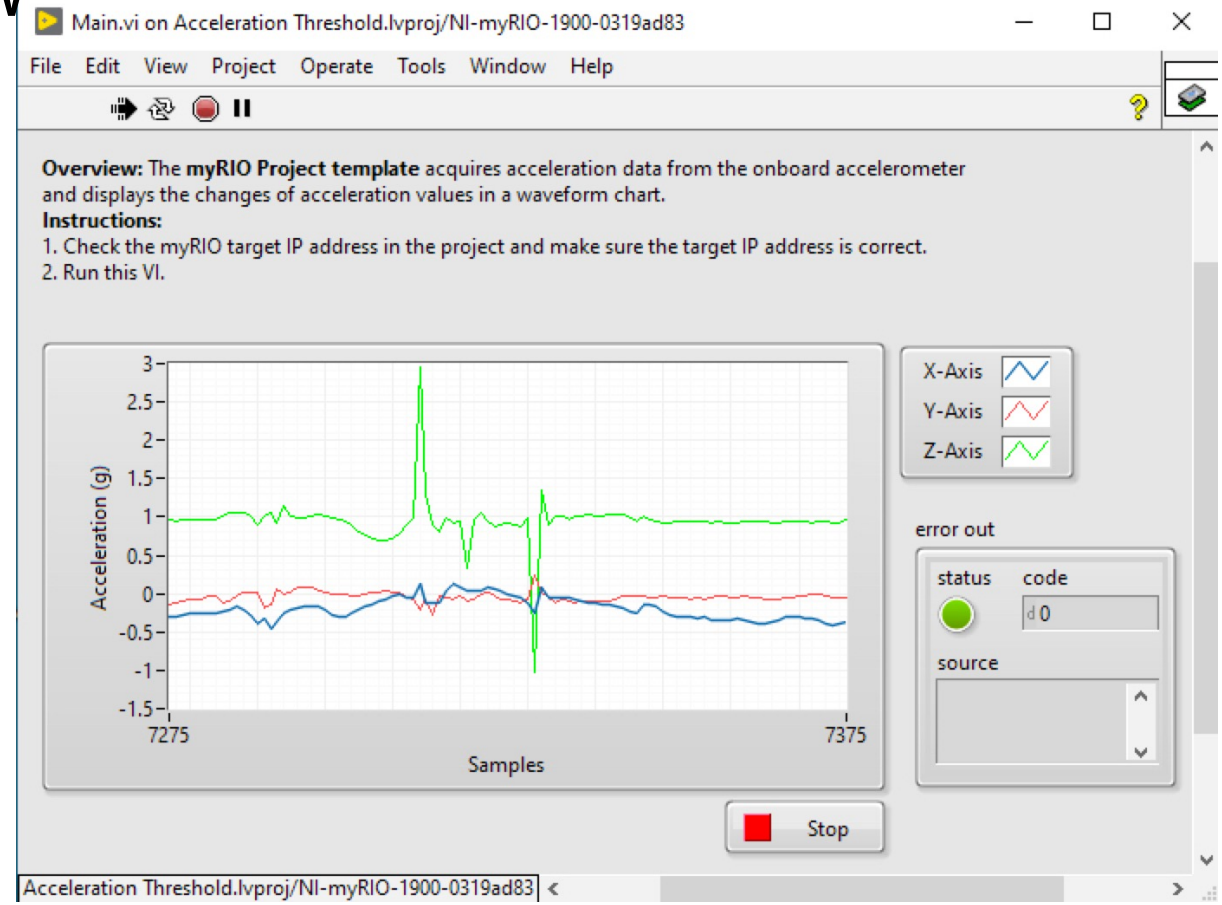
Task 3 - LabVIEW

4 : Run the Template VI

Open the top-level VI that will execute on the processor by navigating in the Project Explorer to “Project Acceleration Threshold.lvproj → (myRIO device name)→Main.vi”.

A template VI has been created that reads data from the onboard accelerometer and displays it on a graph.

On clicking the Run button, the VI will be deployed to the myRIO and run. Verify that the accelerometer data is shown on your screen.

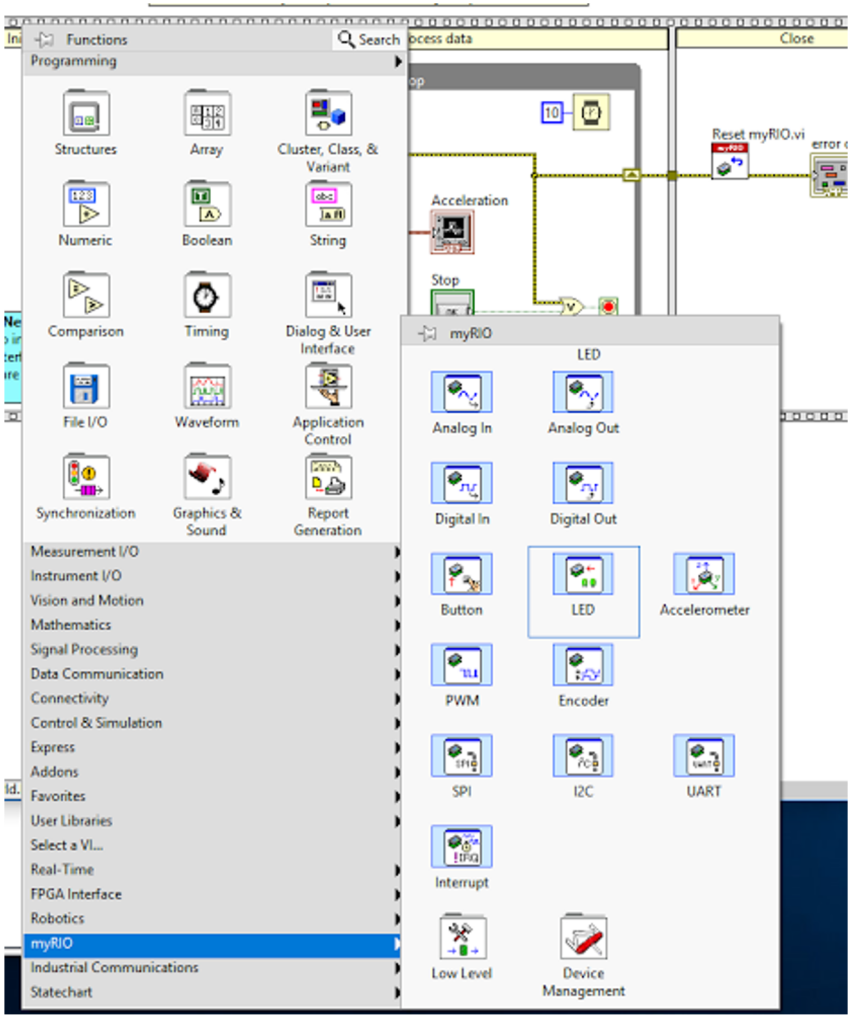


Task 3 - LabVIEW

5 (a) : Modify the Template VI

Add a LED Express VI (“myRIO→LED” in the palette). The Express VI will open a configuration dialog; uncheck LED3 and press [OK] to save the configuration.

Hint : Click on “Customize->View this Palette As->Category (Icons and Text)” to bring up the textual description of the icons in the palette.

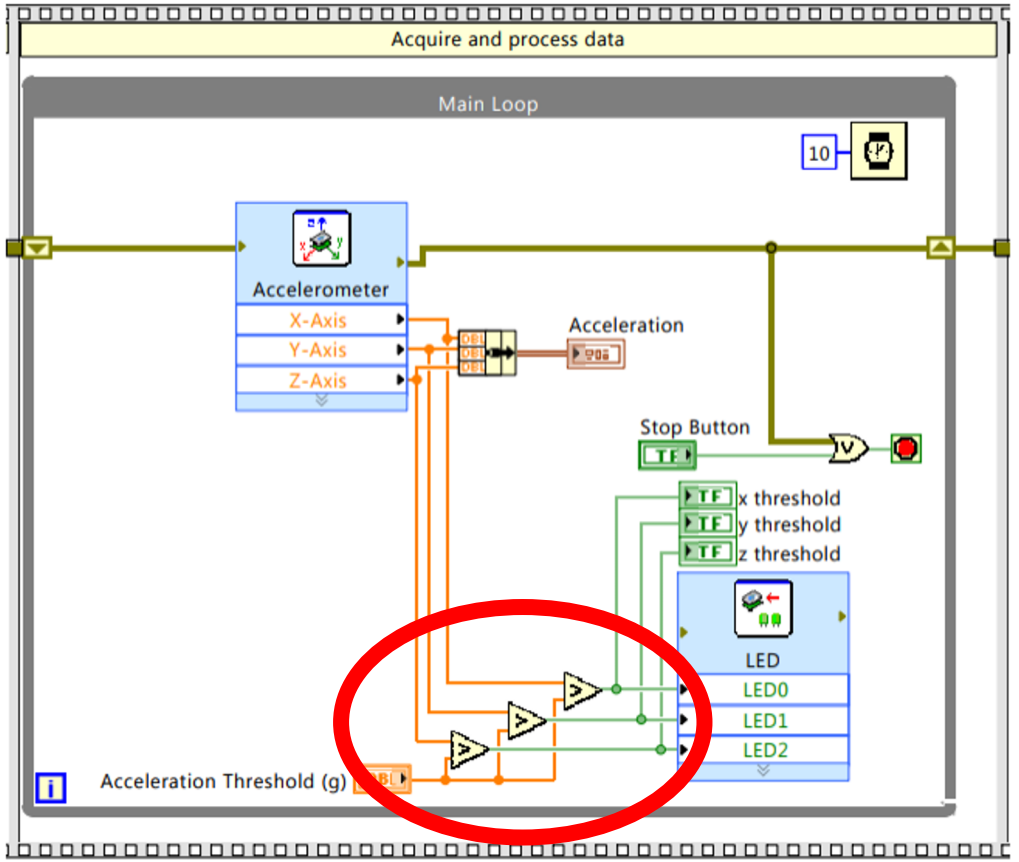


Task 3 - LabVIEW

5 (b,c) : Modify the Template VI

Three “Greater?” nodes that will be used to compare accelerometer values to a threshold.

Wire the ‘x’ (top) input of the first Greater? node to the x output of the Accelerometer VI, and similarly connect the remaining two Greater? nodes to the y and z axes

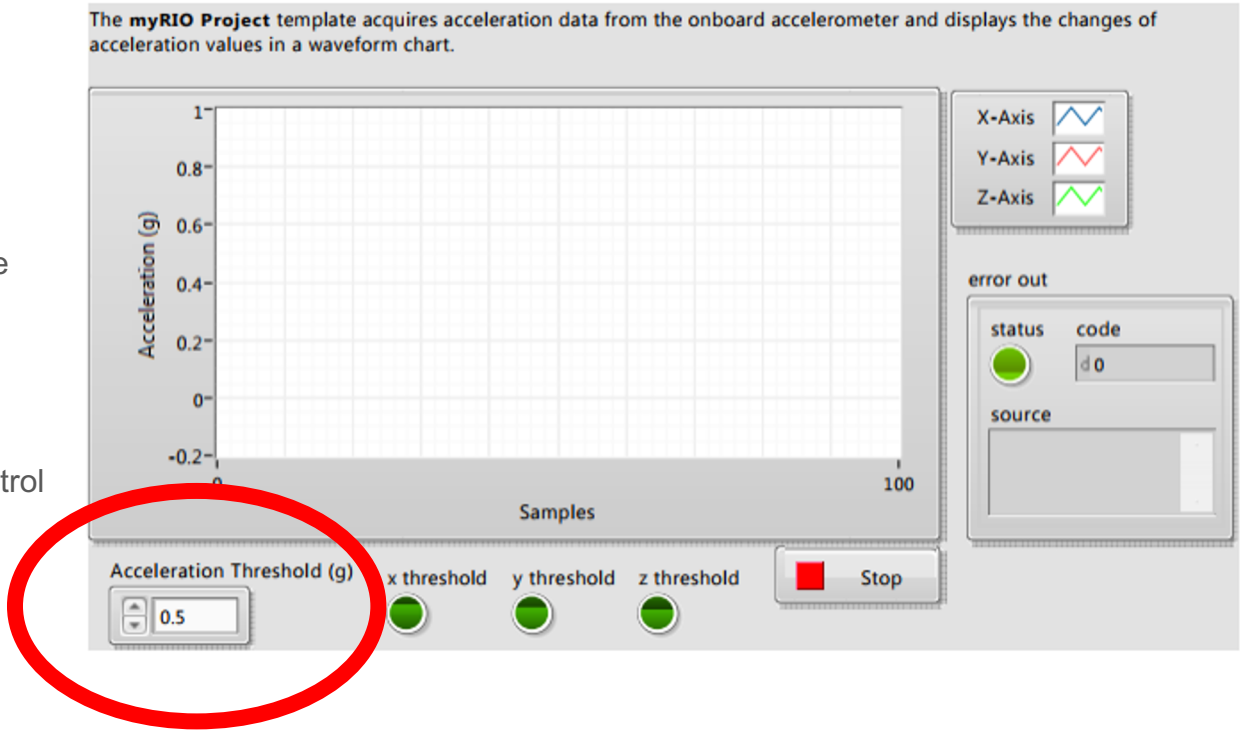


Task 3 - LabVIEW

5 (d) : Modify the Template VI

On the front panel, create a new Numeric Control and label it Acceleration Threshold (g) Set its initial value to 0.5. Right click on the control and from the context menu select “Data Operations→Make Current Value Default”.

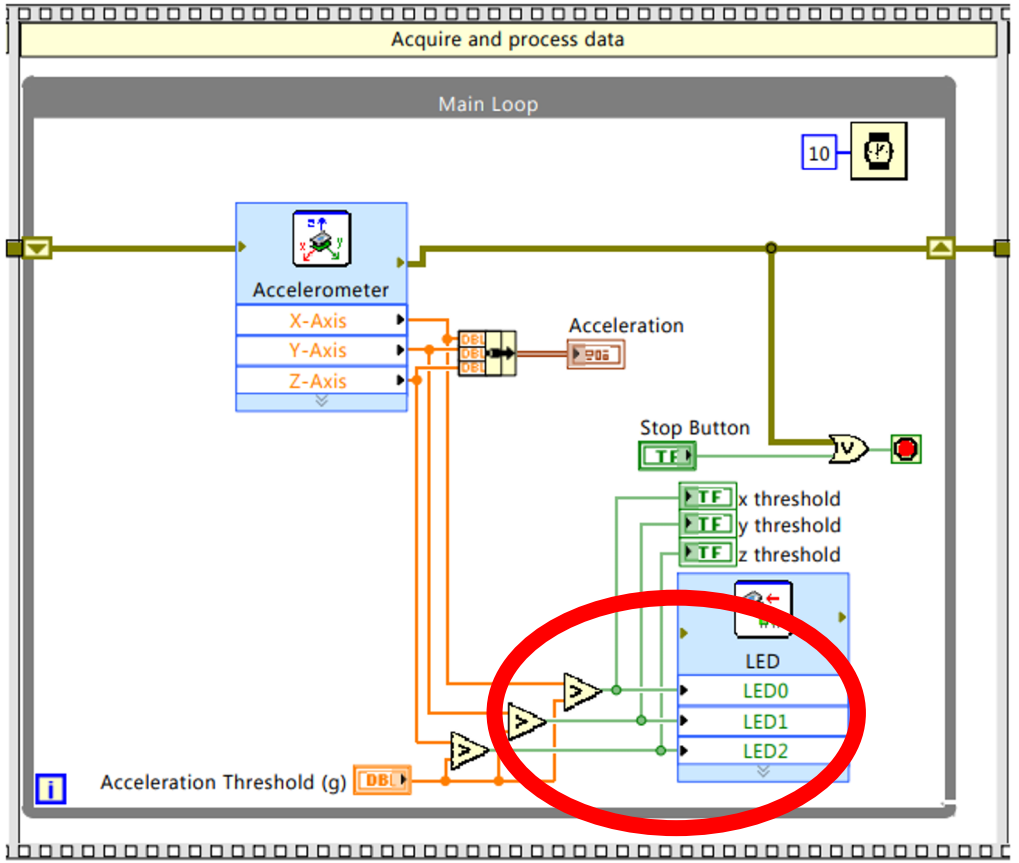
On the block diagram, wire this control to the ‘y’ (bottom) input of each Greater? node.



Task 3 - LabVIEW

5 (e) : Modify the Template VI

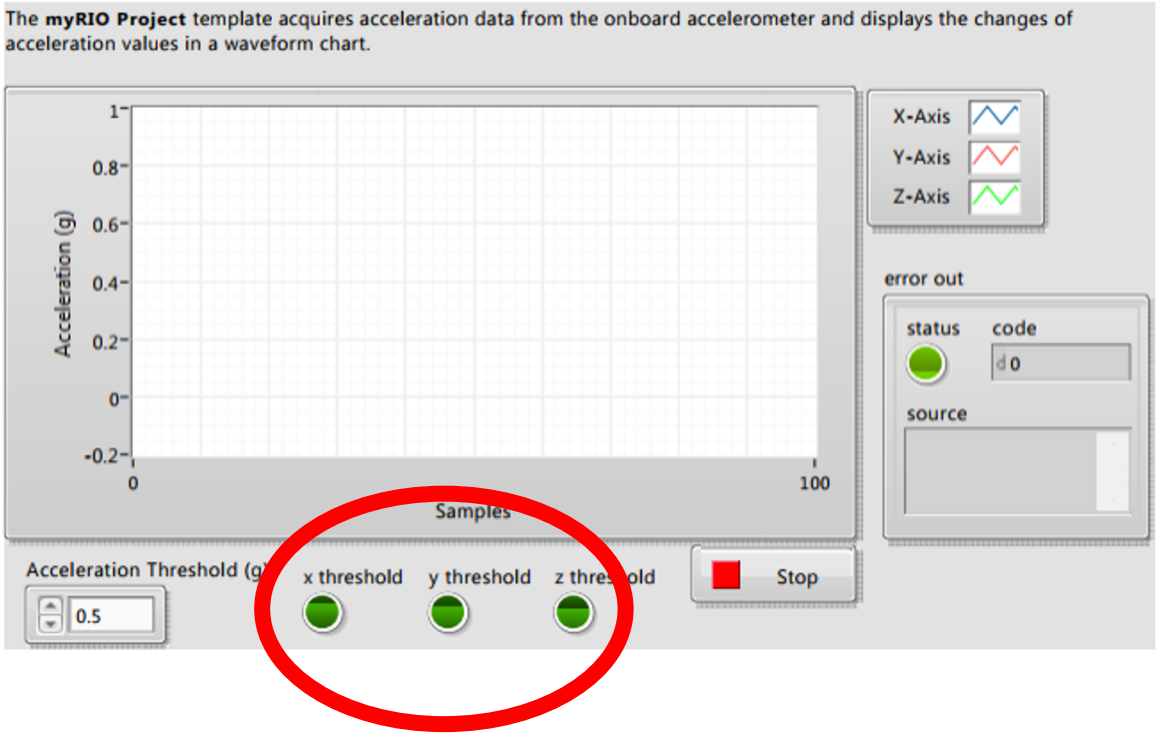
Wire the output of the x axis comparator to the LED0 input of LED VI, and similarly connect the output of the y and z comparators to LED1 and LED2.



Task 3 - LabVIEW

5 (f): Modify the Template VI

On the front panel, create three LED indicators. Label the first “x threshold”, and and similarly for the y and z axes. On the block diagram, wire these to the corresponding output of the comparators.



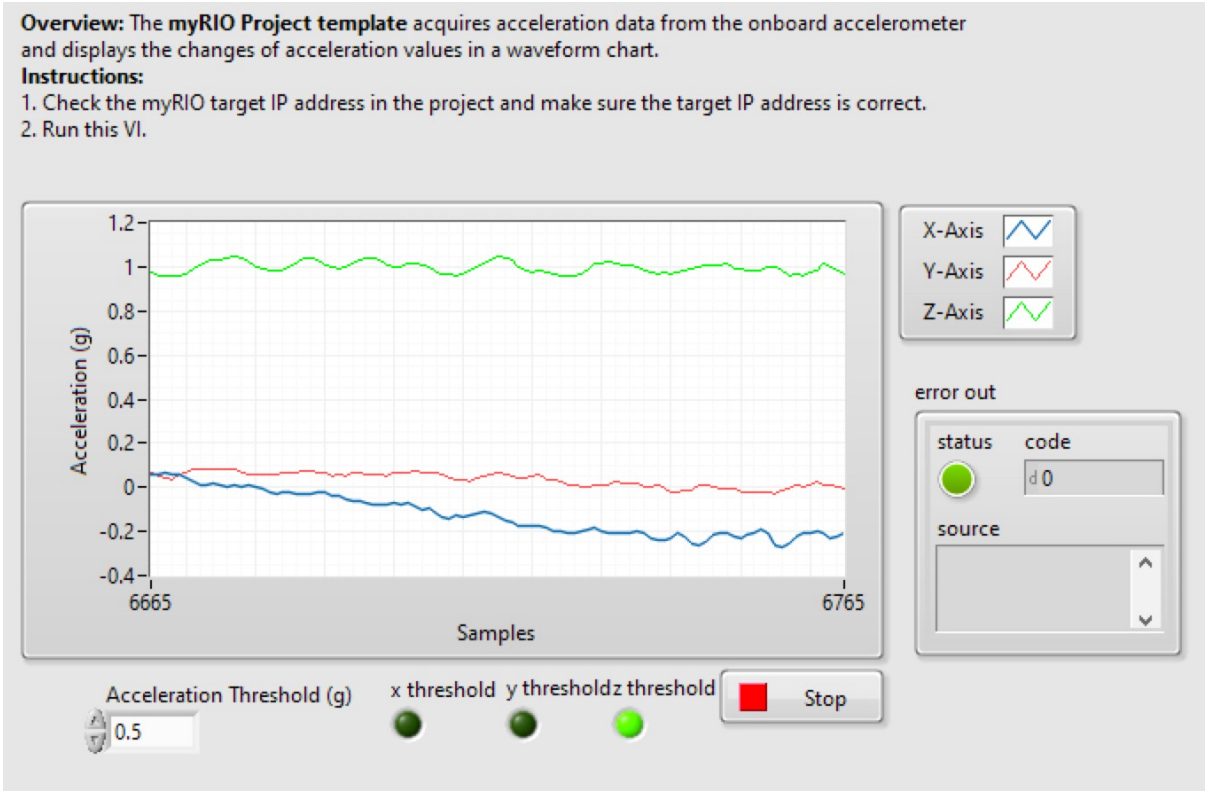
Task 3 - LabVIEW

6 : Run the VI

Run the VI to compile your application, download to myRIO, begin execution, and display debugging information to the front panel. You should see the onboard LEDs illuminate when acceleration on the corresponding axis of the onboard accelerometer exceeds the Acceleration Threshold (g).

TASK 3 CHECK!

Be sure that you can change the threshold and both the LEDs on the MyRio and the LEDs in LabVIEW light up.



Workshop Complete!