How-to: Set Up Yocto Plugin for Eclipse

Targeted Hardware	phyCORE-i.MX7 SOM phyBOARD-Zeta SBC
Targeted Software	BSP Yocto FSL iMX7 PD18.1.0
Date	06 Jul 2018

PHYTEC recommends Eclipse as a platform for building and debugging applications. This guide will show you how to set up the Yocto plugin for Eclipse to cross-develop applications for Yocto-based PHYTEC BSPs.

The instructions are based on a Host PC with a Virtual Machine running Ubuntu 16.04 LTS and Eclipse Photon installed.

Preparing your BSP and Building the SDK

1. Before following the rest of this guide, make sure the BSP is built as described in the Quickstart: Quickstart - Building BSP from Source.
2. Once the BSP is built the following command needs to be run in the BSP directory to allow IDE support.

```
bitbake meta-ide-support
```

3. Navigate to the BSP build directory and execute the following bitbake command to install the sdk. This will take a little while to complete.
   The specific MACHINE configuration can be found in our release notes: BSP-Yocto-FSL-iMX7-PD18.1.0 Release Notes

```
MACHINE=imx7d-phyboard-zeta-001 bitbake fsl-image-gui -c populate_sdk
```

4. The "populate_sdk" command creates a .sh file in the <BSP DIRECTORY>/build/tmp/deploy/sdk directory. Execute the created script to install the sdk on the host machine.
   After the script starts it will ask for a directory to install the sdk too. Hit enter for the default location.

```
./<BSP DIRECTORY>/build/tmp/deploy/sdk/fsl-imx-x11-glibc-x86_64-fsl-image-gui-cortexa7hf-neon-toolchain-
4.9.11-1.0.0.sh
```

Eclipse Setup

The instructions are applicable for Eclipse Oxygen and Photon running on Ubuntu 16.04 LTS. Older versions of Eclipse (ex. Luna or Neon) have similar steps but should be run on a Ubuntu 14.04 LTS system.

1. Download and install the Eclipse Installer: https://wiki.eclipse.org/Eclipse_Installer
2. Unpack the Eclipse Installer to a desired location. This will automatically create a directory called "eclipse-installer".

```
tar -xvzf eclipse-inst-linux64.tar.gz -C /<ECLIPSE DIRECTORY>/
```

3. Download the latest Java 10 JDK: http://www.oracle.com/technetwork/java/javase/downloads/index.html
4. Unpack the JDK into the "eclipse-installer" directory that was created in the previous step.

```
tar -xvzf jdk-10.0.1_linux-x64_bin.tar.gz -C /<ECLIPSE DIRECTORY>/eclipse-installer/
```

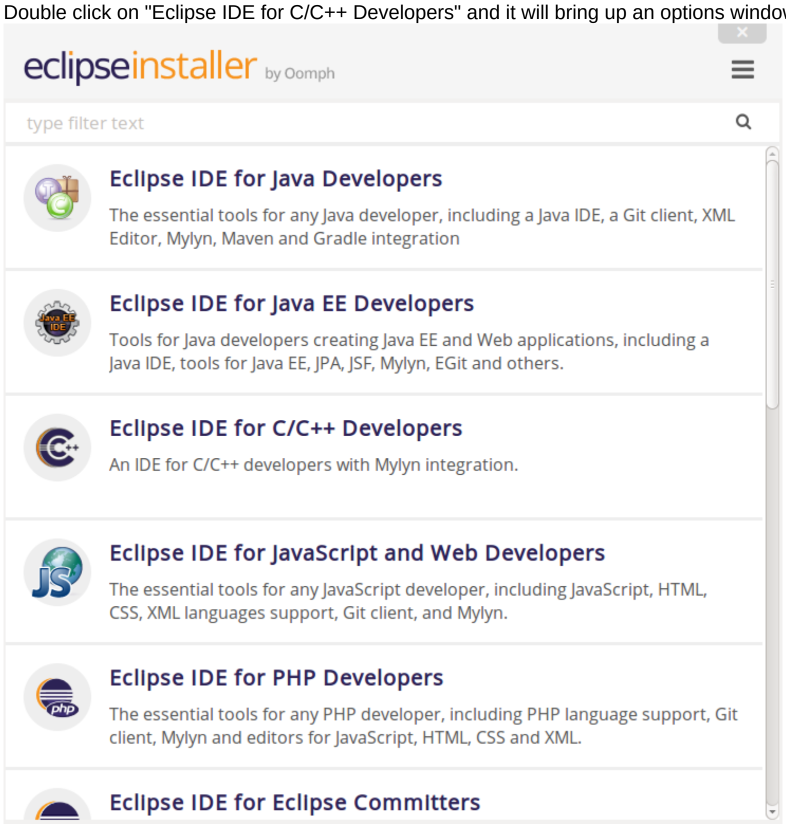
5. In the "eclipse-installer" directory edit the "eclipse-inst.ini" file. Add the following lines at the top of the file:

```
-vm
/<ECLIPSE DIRECTORY>/eclipse-installer/jdk-10.0.1/bin/java
```

6. The eclipse installer can now be run from the "eclipse-installer" directory.

```
./eclipse-inst
```

7. Double click on "Eclipse IDE for C/C++ Developers" and it will bring up an options window.



Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

Eclipse IDE for Java EE Developers
Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.

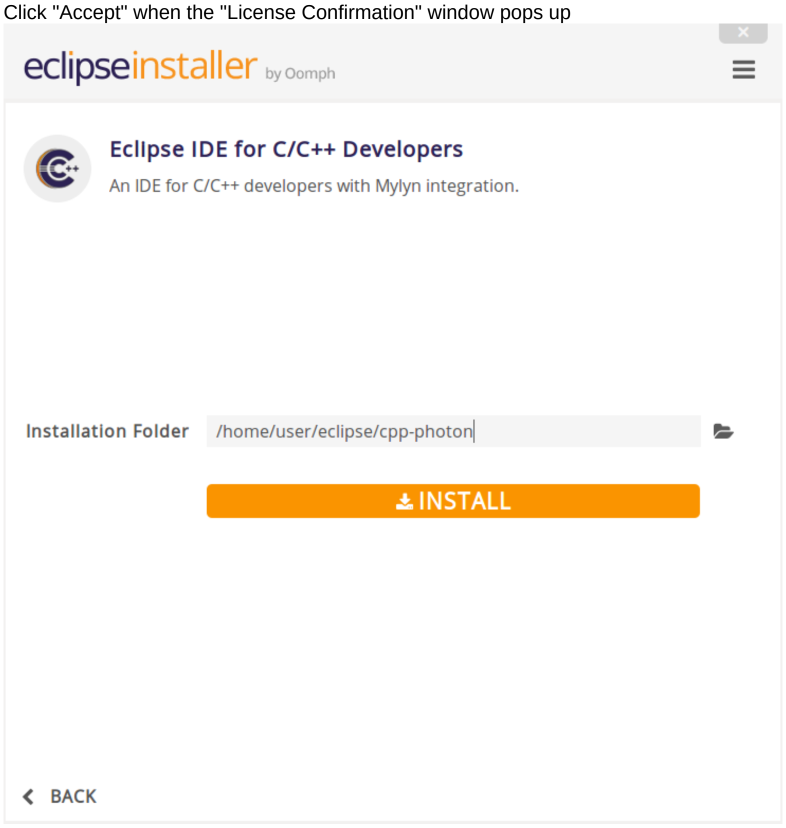
Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.

Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Eclipse IDE for Eclipse Committers

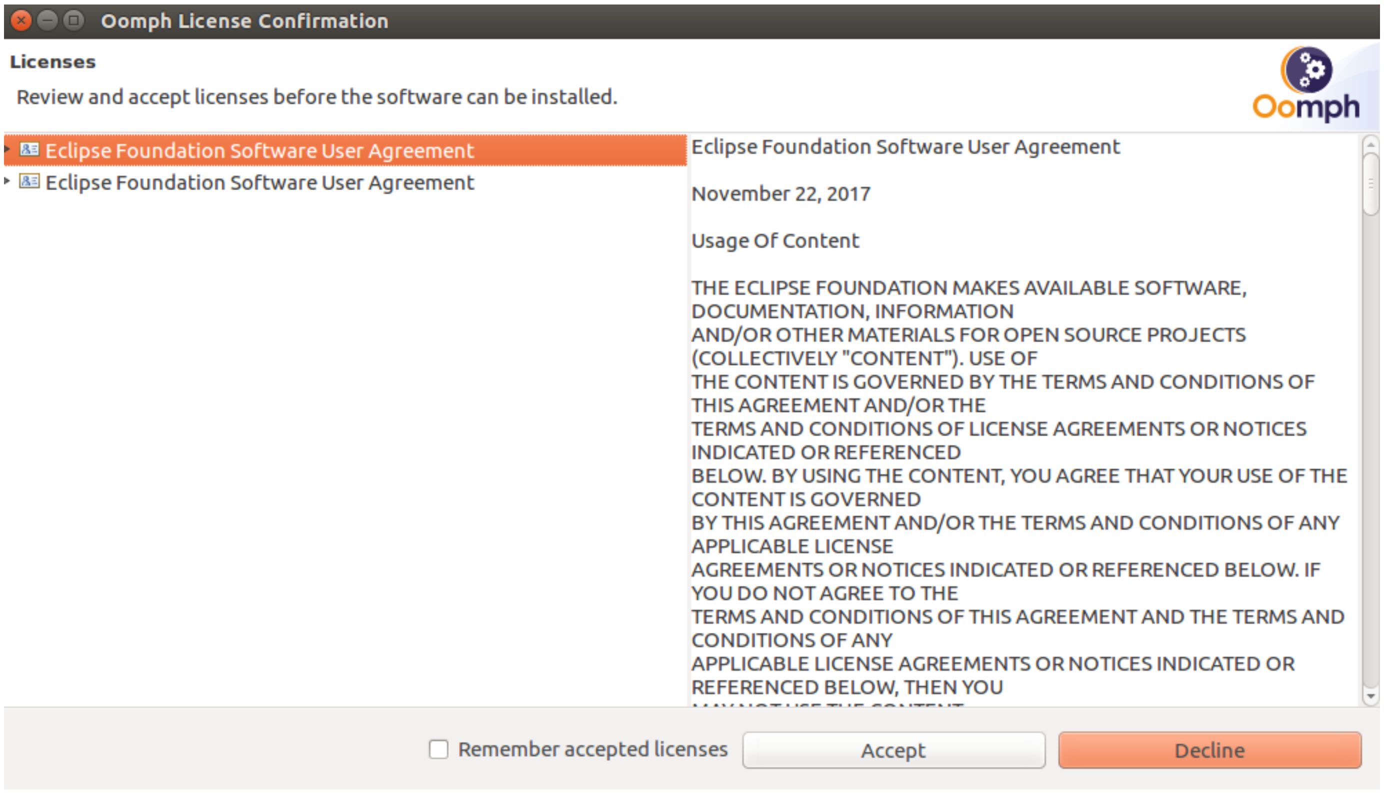
8. Change the "Installation Folder" to a desired location if needed and then click "Install" when ready

9. Click "Accept" when the "License Confirmation" window pops up


eclipseinstaller by Oomph
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.
INSTALL
BACK

Installation Folder /home/user/eclipse/cpp-photon

10. DO NOT LAUNCH Eclipse or close the program if it was launched. Eclipse will not function properly if launched before sourcing the correct environment. Please see the next section (Sourcing the Environment and Setting up Eclipse) on how to do t


Oomph License Confirmation
Licenses
Oomph
Eclipse Foundation Software User Agreement
Eclipse Foundation Software User Agreement
Remember accepted licenses
Accept
Decline

Review and accept licenses before the software can be installed.

Eclipse Foundation Software User Agreement

November 22, 2017

Usage Of Content

THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION
AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF
THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE
TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED
BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED
BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE
AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE
TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS OF ANY
APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW, THEN YOU

Sourcing the Environment and Setting up Eclipse

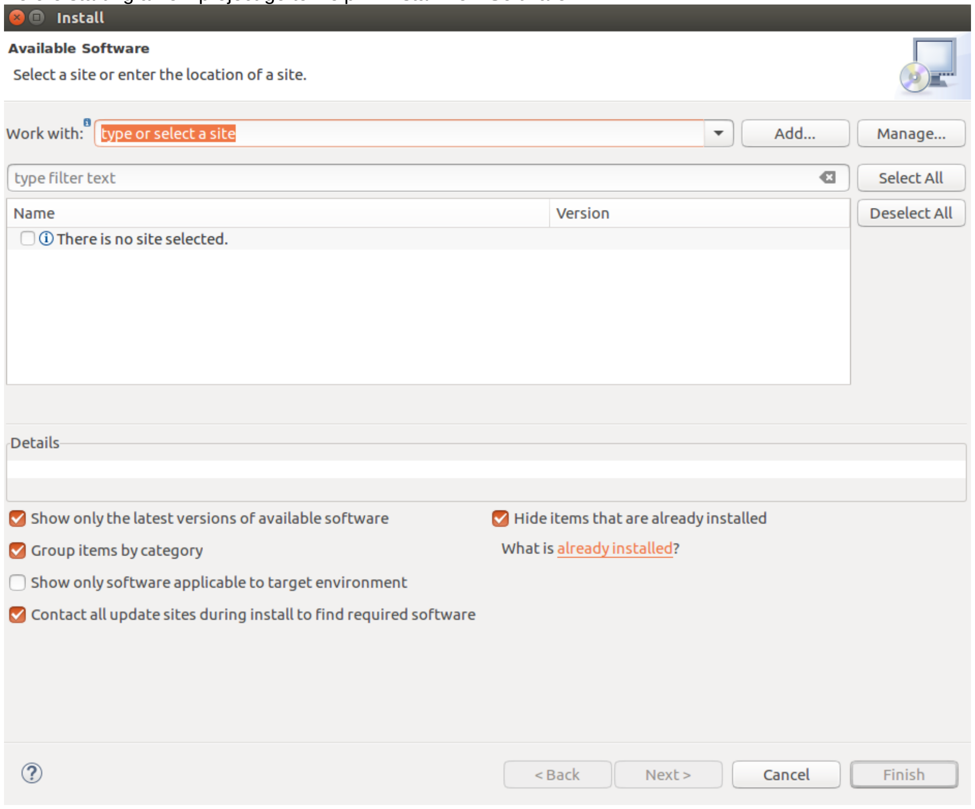
1. The correct environment needs to be sourced. In the terminal navigate to the BSP directory and source the following environment.

```
source build/tmp/work/imx7d_phyboard_zeta_001-poky-linux-gnueabi/fsl-image-gui/1.0-r0/sdk/image/opt/fsl-imx-x11/4.9.11-1.0.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

2. Eclipse **MUST** be started in the same terminal window the environment was sourced from.

```
./eclipse/cpp-photon/eclipse/eclipse
```

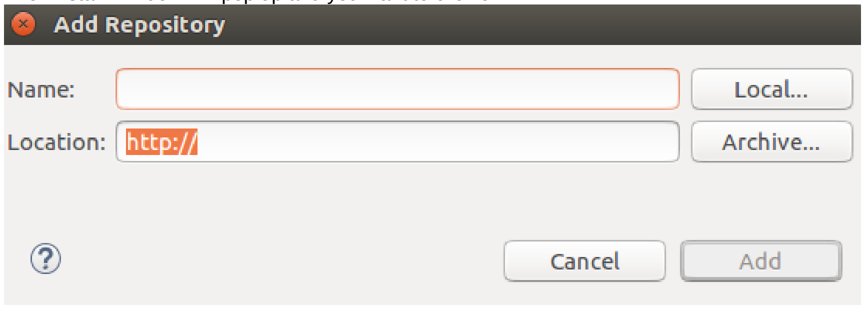
3. Before starting a new project go to Help -> Install New Software



There is no site selected.

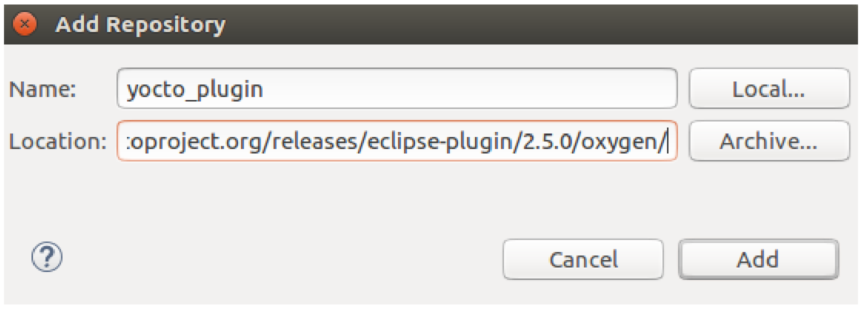
- [✓] Show only the latest versions of available software
- [✓] Group items by category
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software
- [✓] Hide items that are already installed

4. The "Install" window will pop up and you want to click on ADD



5. In the "Add Repository" window, name the plugin what you would like and then go to this link (http://downloads.yoctoproject.org/releases/eclipse-plugin/), find the latest plugin for oxygen or photon and then copy the address into the location field. Click OK when ready.

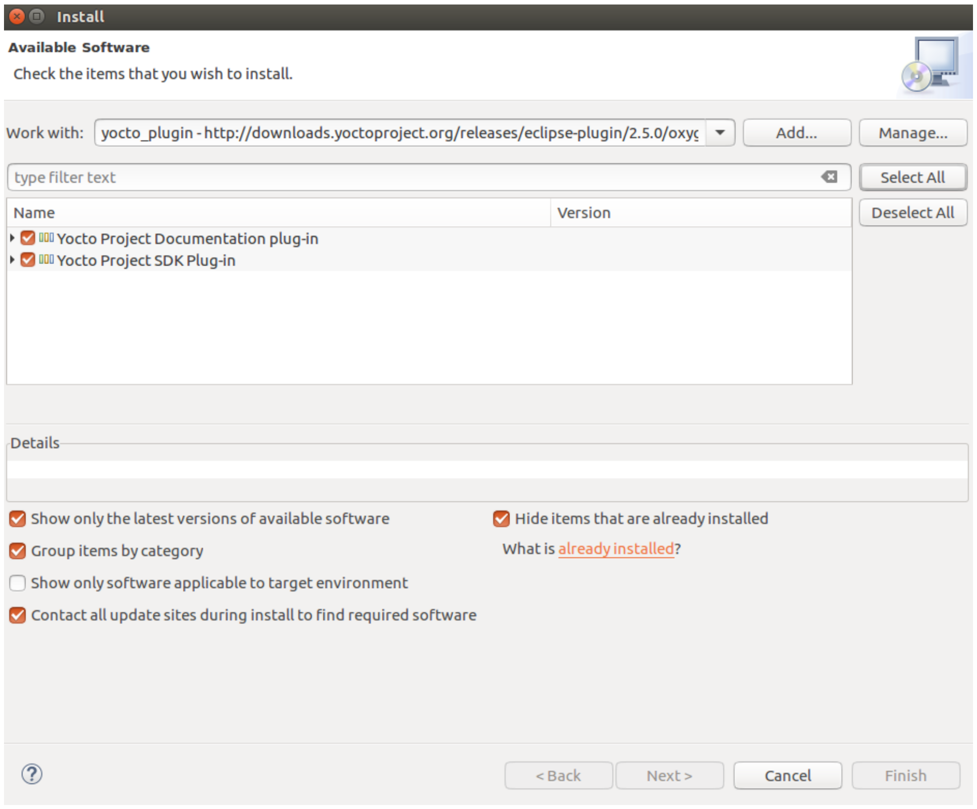
"Name": yocto_plugin
"Location": http://downloads.yoctoproject.org/releases/eclipse-plugin/2.5.0/oxygen/



Name: yocto_plugin

Location: :oproject.org/releases/eclipse-plugin/2.5.0/oxygen/

6. A few available plugins will show up in the window. Select all items and click Next to install.



Available Software

Check the items that you wish to install.

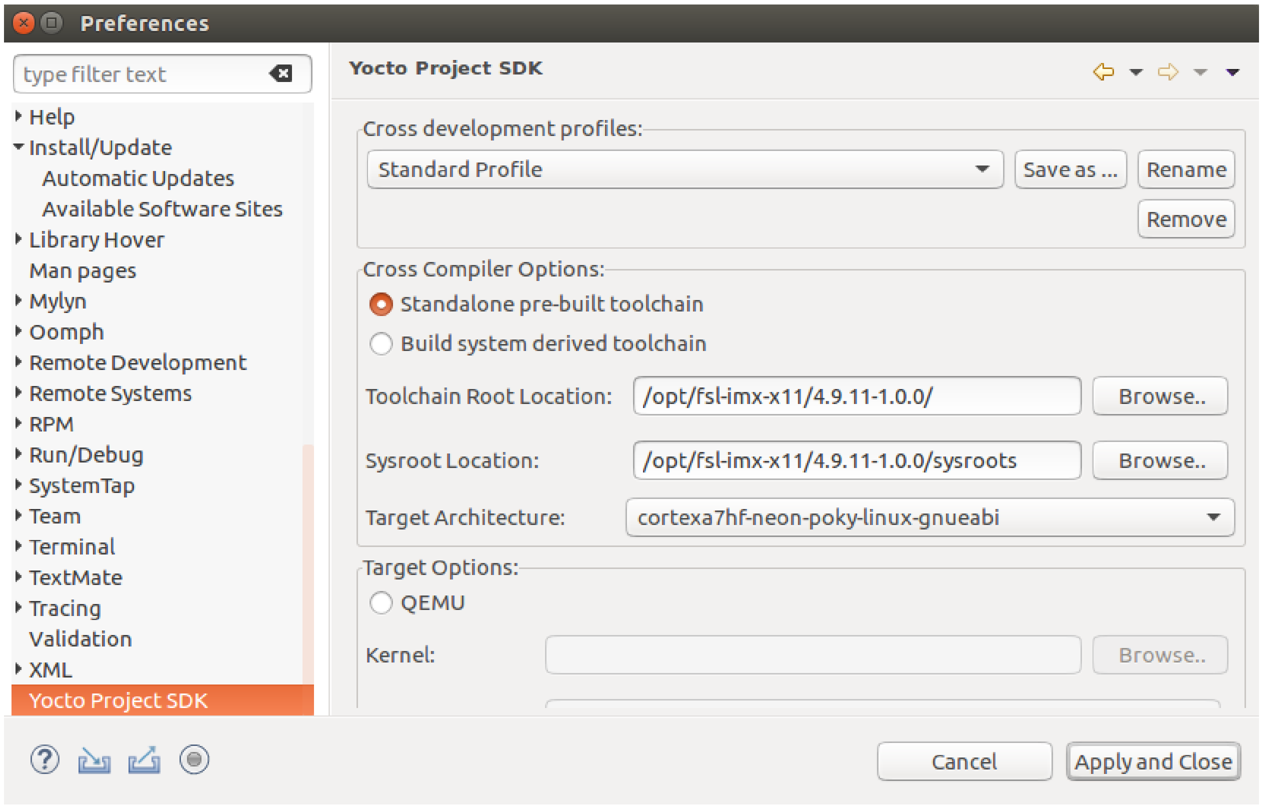
Work with: yocto_plugin - http://downloads.yoctoproject.org/releases/eclipse-plugin/2.5.0/oxyç

Name	Version
☑ Yocto Project Documentation plug-in	
☑ Yocto Project SDK Plug-in	

☑ Show only the latest versions of available software
☑ Group items by category
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software
☑ Hide items that are already installed
What is already installed?

7. In Eclipse go to Window -> Preferences.

   - Select the Yocto Project SDK option on the left.
   - Under "Cross Compiler Options" select the radial button for "Standalone pre-built toolchain".
   - Set the "Toolchain Root Location" to the SDK you just installed: /opt/fsl-imx-x11/4.9.11-1.0.0/
   - The "Target Architecture" field should automatically fill with "cortexa7hf-neon-poky-linux-gnueabi"
   - Set the "Sysroots Location": /opt/fsl-imx-x11/4.9.11-1.0.0/sysroots
   - Click "Apply and Close" to save settings.



Yocto Project SDK

Cross development profiles: Standard Profile

Cross Compiler Options:

[✓ Standalone pre-built toolchain] Build system derived toolchain

Toolchain Root Location: /opt/fsl-imx-x11/4.9.11-1.0.0/

Sysroot Location: /opt/fsl-imx-x11/4.9.11-1.0.0/sysroots

Target Architecture: cortexa7hf-neon-poky-linux-gnueabi

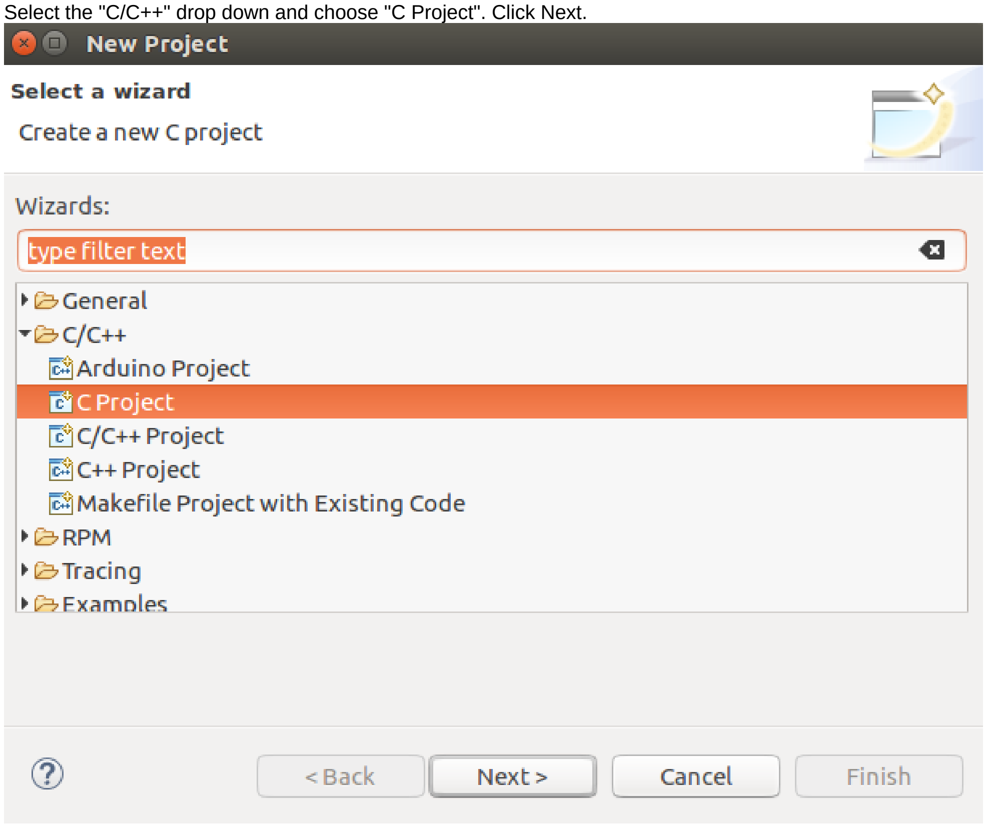
Target Options:

QEMU

Kernel:

Creating a Test Application

1. Now that Eclipse is set up a test project can be created to verify everything is set up correctly. Go to File New Project
2. Select the "C/C++" drop down and choose "C Project". Click Next.

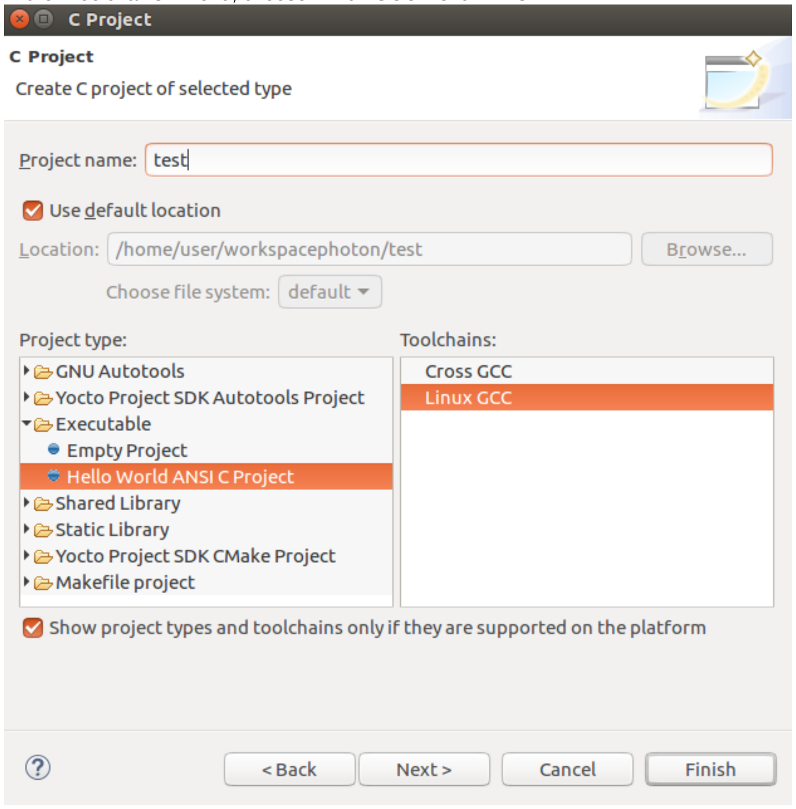


Select a wizard

Create a new C project

3. Name the project "test" and leave the default workspace box checked.
4. In the "Project type:" menu, under the "Executable" drop down choose "Hello World ANSI C Project".

5. In the "Toolchains:" menu, choose "Linux GCC". Click Finish.



C Project

C Project

Create C project of selected type

Project name: test

☑ Use default location

Location: /home/user/workspacephoton/test

Choose file system: default

Project type:
- GNU Autotools
- Yocto Project SDK Autotools Project
- Executable
  - Empty Project
  - Hello World ANSI C Project
- Shared Library
- Static Library
- Yocto Project SDK CMake Project
- Makefile project

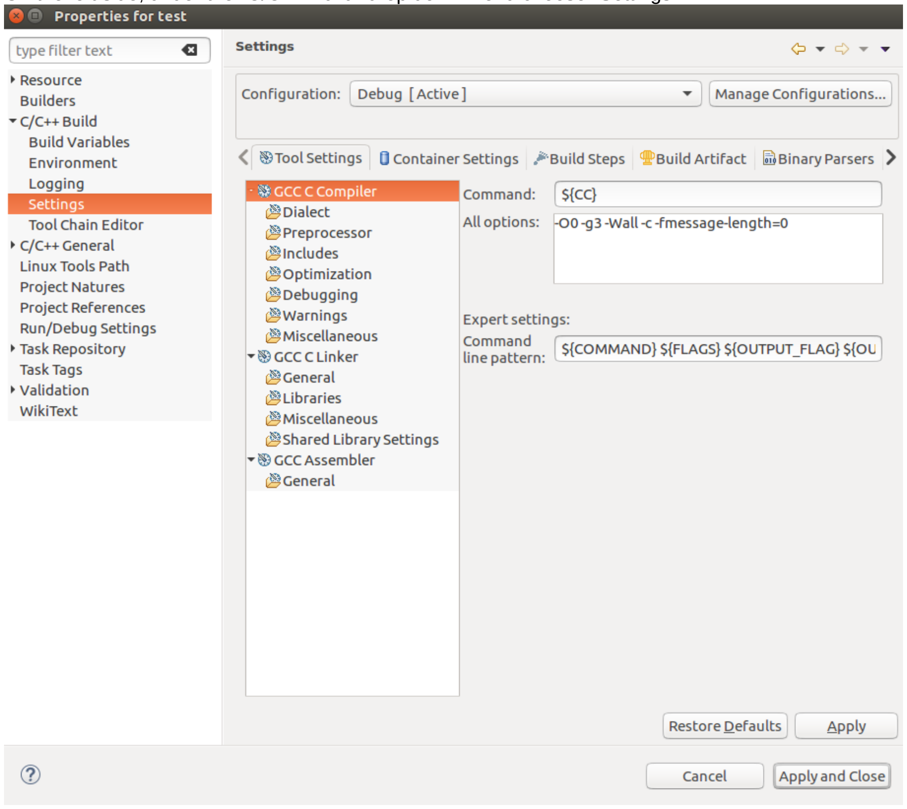
Toolchains:
- Cross GCC
- Linux GCC

☑ Show project types and toolchains only if they are supported on the platform

< Back | Next > | Cancel | Finish

Running an Application on Target Hardware

1. In the "Project Explorer" Right-click on the newly created project and choose "Properties".
2. On the left side, under the "C/C++ Build" drop down menu choose "Settings"



Properties for test

type filter text

- Resource
- Builders
- C/C++ Build
  - Build Variables
  - Environment
  - Logging
  - Settings
  - Tool Chain Editor
- C/C++ General
- Linux Tools Path
- Project Natures
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

Settings

Configuration: Debug [Active] | Manage Configurations...

Tool Settings | Container Settings | Build Steps | Build Artifact | Binary Parsers

- GCC C Compiler
  - Dialect
  - Preprocessor
  - Includes
  - Optimization
  - Debugging
  - Warnings
  - Miscellaneous
- GCC C Linker
  - General
  - Libraries
  - Miscellaneous
  - Shared Library Settings
- GCC Assembler
  - General

Command: ${CC}

All options: -O0 -g3 -Wall -c -fmessage-length=0

Expert settings:

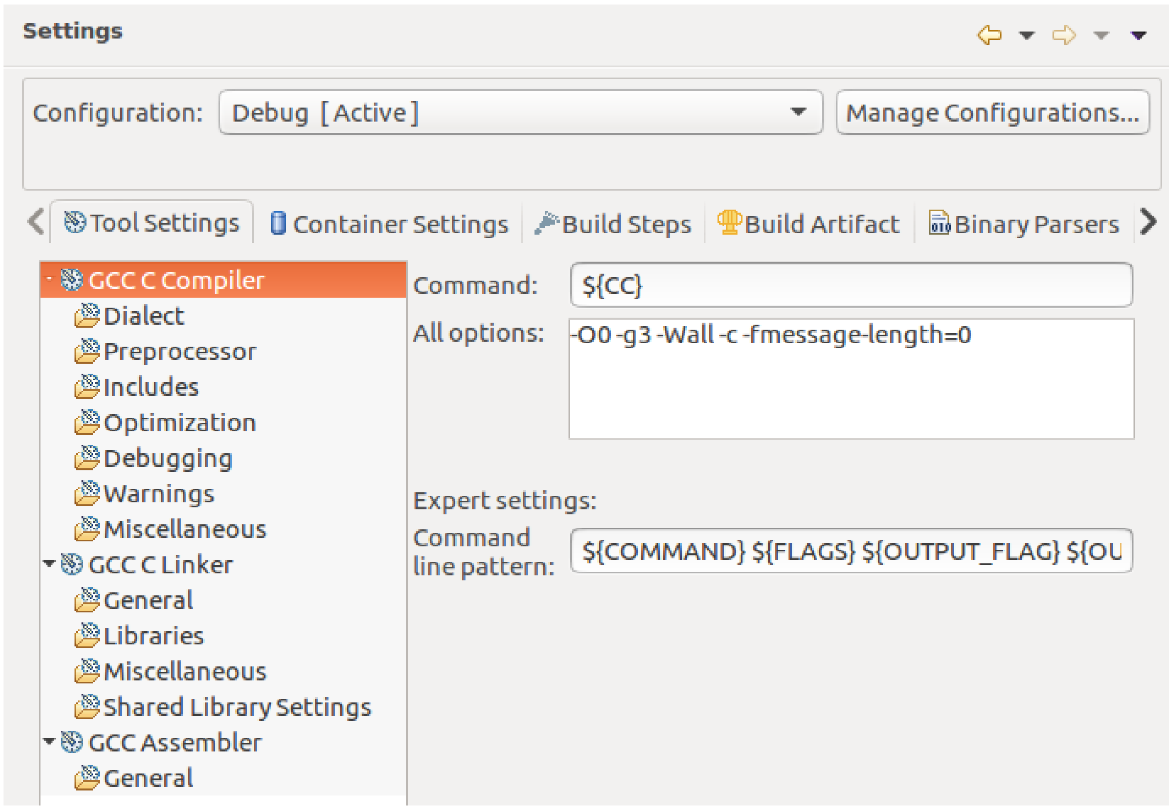
Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OU

Restore Defaults | Apply

Cancel | Apply and Close

3. In the main window choose "GCC C Compiler" and change "Command:" to the following:

```
${CC}
```



Command: ${CC}

All options: -O0 -g3 -Wall -c -fmessage-length=0

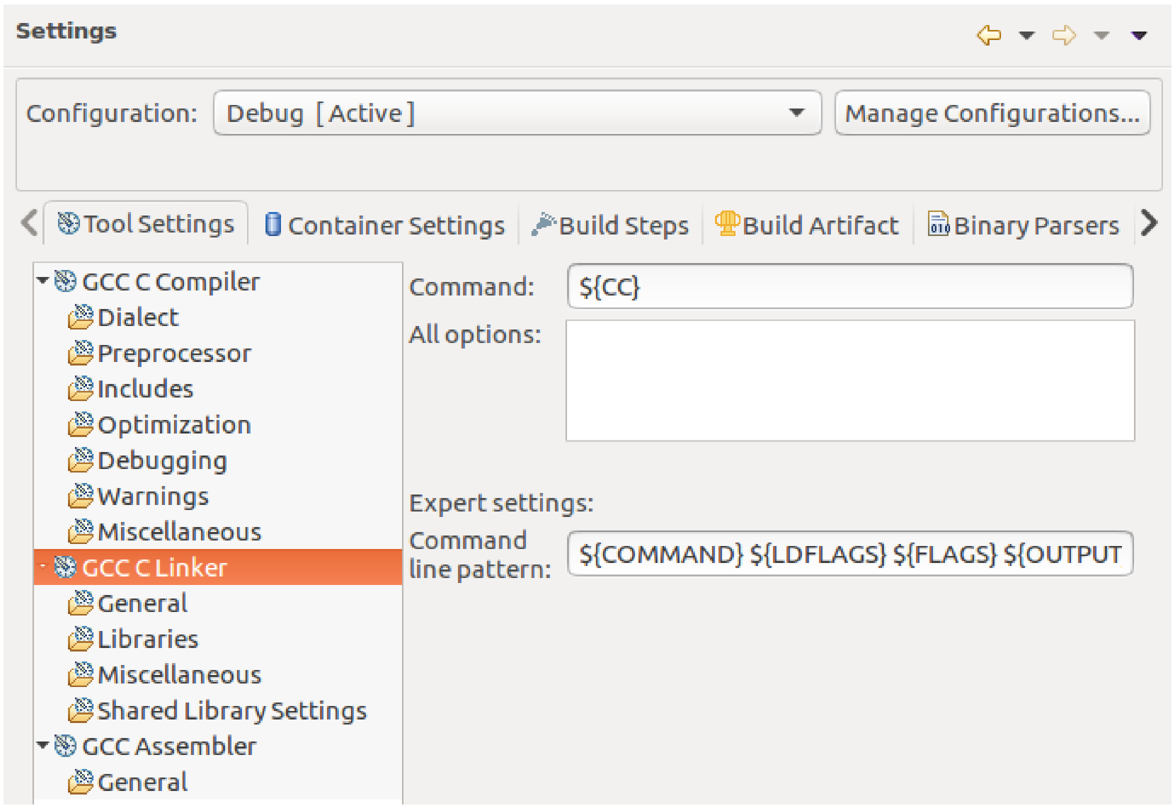
Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OU

4. Now choose "GCC C Linker"
   a. Change the "Command:" to the following:

      ```
      ${CC}
      ```

   b. Add in ${LDFLAGS} between ${COMMAND} and ${FLAGS} in the "Command line pattern:" field so it looks like the following:

      ```
      ${COMMAND} ${LDFLAGS} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}
      ```


Settings

Configuration: Debug [Active]

Command: ${CC}

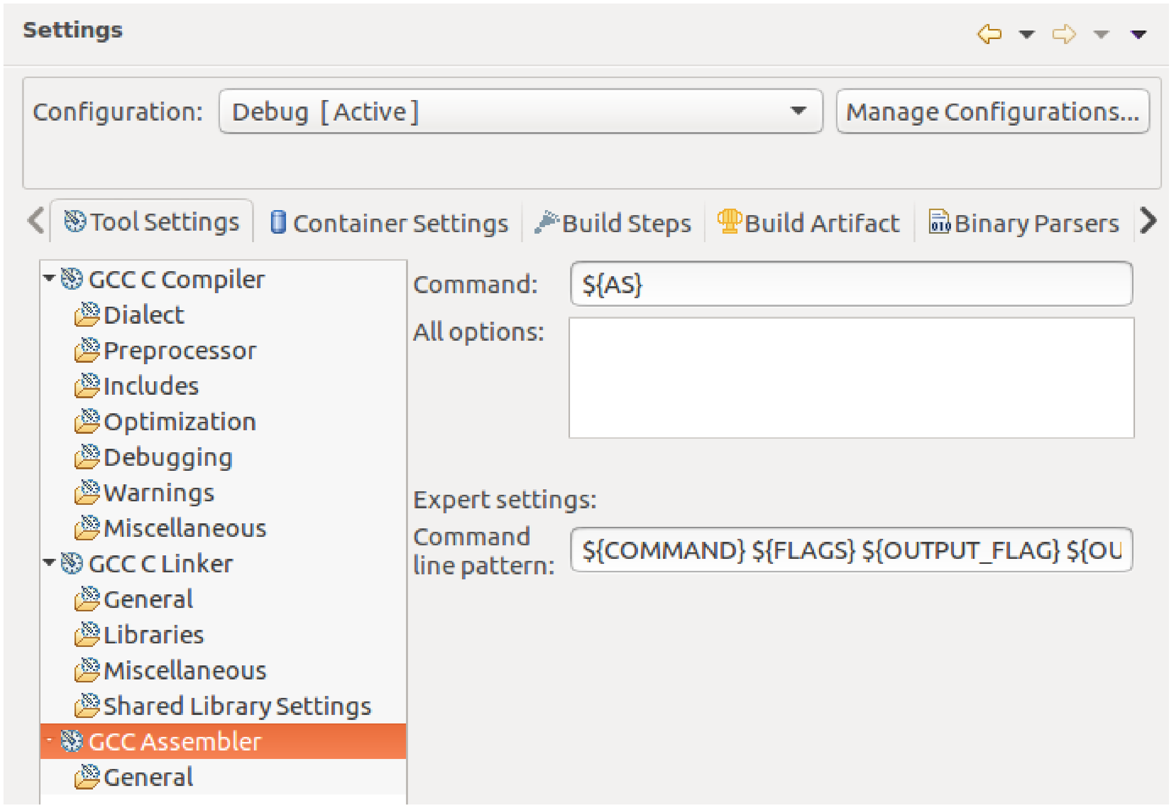
All options:

Expert settings:

Command line pattern: ${COMMAND} ${LDFLAGS} ${FLAGS} ${OUTPUT

5. Under "GCC Assembler", change the "Command:" to the following:

```
${AS}
```


Settings

Configuration: Debug [Active]

Command: ${AS}

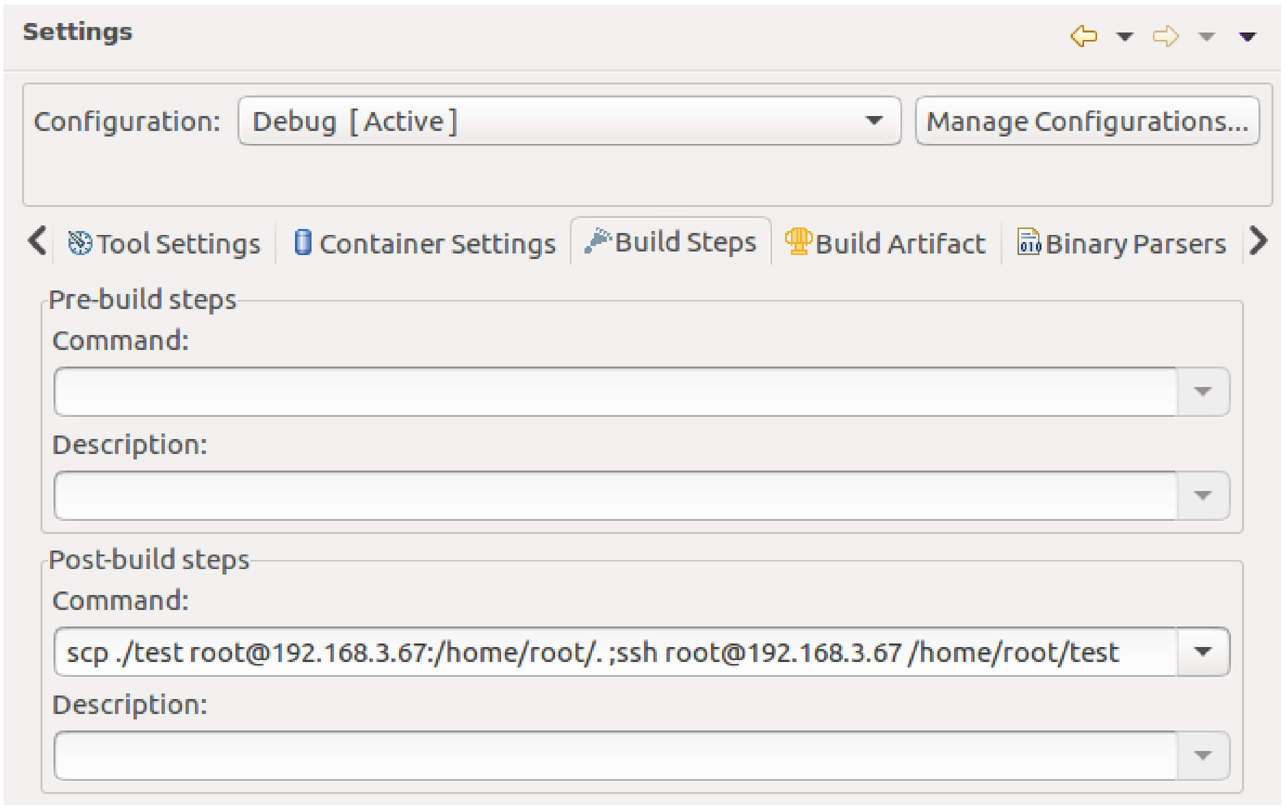
All options:

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OU

6. Now switch to the "Build Steps" tab and change the "Post-build steps" "Command:" field to the following. Make sure to change the ip address of the command to match the address of the target hardware.

```
scp ./test root@192.168.xx.xx:/home/root/. ;ssh root@192.168.xx.xx /home/root/test
```



```
scp ./test root@192.168.3.67:/home/root/. ;ssh root@192.168.3.67 /home/root/test
```

7. Click "Apply and Close" when you are ready to build.
8. Build the project and you should see a similar output in Eclipse to the following:

```
09:40:47 **** Build of configuration Debug for project test ****
make all
Building file: ../src/test.c
Invoking: GCC C Compiler
arm-poky-linux-gnueabi-gcc  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt
/fsl-imx-x11/4.9.11-1.0.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi -O0 -g3 -Wall -c -fmessage-
length=0 -MMD -MP -MF"src/test.d" -MT"src/test.o" -o "src/test.o" "../src/test.c"
Finished building: ../src/test.c

Building target: test
Invoking: GCC C Linker
arm-poky-linux-gnueabi-gcc  -march=armv7ve -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt
/fsl-imx-x11/4.9.11-1.0.0/sysroots/cortexa7hf-neon-poky-linux-gnueabi -Wl,-O1 -Wl,--hash-style=gnu -Wl,--
as-needed  -o "test"  ./src/test.o
Finished building target: test

make --no-print-directory post-build
scp ./test root@192.168.3.67:/home/root/. ;ssh root@192.168.3.67 /home/root/test
!!!Hello World!!!


09:40:48 Build Finished. 0 errors, 0 warnings. (took 1s.104ms)
```