

PetaLinux SDK User Guide

Eclipse Plugin Guide

UG979 (v2013.04) April 22, 2013



Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

© Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

Date	Version	Notes
2012-08-03	3.1	Initial public release for SDK 3.1 release
2012-09-03	12.9	Updated for PetaLinux SDK 12.9 release
2012-12-17	2012.12	Updated for PetaLinux SDK 2012.12 release
2013-04-22	2013.04	Updated for PetaLinux SDK 2013.04 release

Table of Contents

Revision History	1
Table of Contents	2
About this Guide	3
Prerequisites	3
Installation	4
Getting Started	7
Starting PetaLinux Eclipse SDK	7
Installing a BSP	8
Rebuilding a Reference Design Software Image	8
Test a Software Image in QEMU	10
Test a Software Image on the Board	12
Direct Kernel Image Download	12
Network Boot Kernel Image	13
Board Bringup	15
Preparation	15
Create and Configure New Software Platform	15
Build PetaLinux	17
Application Development	18
Application Creation	18
Application System Configuration	20
Building Applications	23
Testing Applications	24
Preparing the build system for debugging	25
Debugging Applications with GDB	28
Importing Existing PetaLinux Applications and Libraries	32
Additional Resources	36
References	36

About this Guide

The PetaLinux Eclipse SDK provides support to develop with PetaLinux from within the Eclipse IDE. This includes user application and library development and debugging, as well as building and configuring PetaLinux and bringup of hardware platforms. This guide describes how to use the functionality provided by the PetaLinux Eclipse SDK.

It is assumed that the readers of this document have basic Linux knowledge such as how to run Linux commands, basic familiarity with Eclipse and or Xilinx XSDK as well as basic PetaLinux familiarity having read and worked through the examples in the PetaLinux SDK Getting Started Guide (UG977) .

Prerequisites

This getting started document assumes that the following prerequisites have been satisfied:

- Eclipse or Xilinx SDK has been installed.
- PetaLinux SDK has been installed.
- PetaLinux setup script has been sourced in each command console which you work with PetaLinux. Run the following command to check whether the PetaLinux environment has been setup on the command console:

```
$ echo $PETALINUX
```

If the PetaLinux working environment has been setup, it should shows the path to the installed PetaLinux. If it shows nothing, please refer to section Environment Setup in the PetaLinux SDK Getting Started Guide (UG977) document to setup the environment.

Installation

This section describes the steps to install the PetaLinux Eclipse SDK into an Eclipse environment. Throughout this guide "Eclipse" can refer to either a standard Eclipse installation or a Xilinx SDK Eclipse installation.

1. Start Eclipse.



WARNING: If Eclipse or Xilinx SDK is installed in a directory owned by root, the plugin must also be installed as root. However, after installation is complete, your development should be done as a non-root user.

2. Select a workspace (or create one), this workspace is only required to launch Eclipse.
3. Open Install New Software from the menu **Help > Install New Software**.

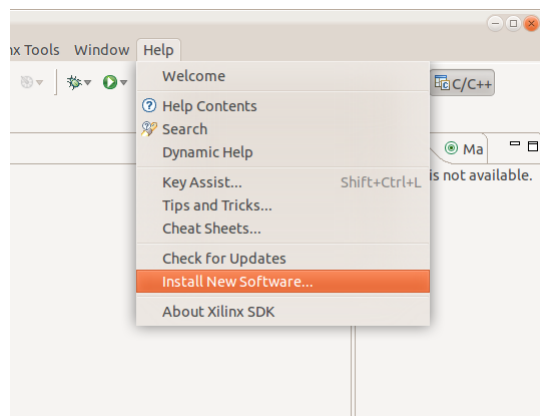


Figure 1: Install New Software Menu Item

4. Specify the location of the site as <http://www.xilinx.com/tools/petalinux/plugin> and press enter.
5. Please wait for Eclipse to contact the server.
6. Select the checkbox for the **PetaLinux SDK** and click **Next**.

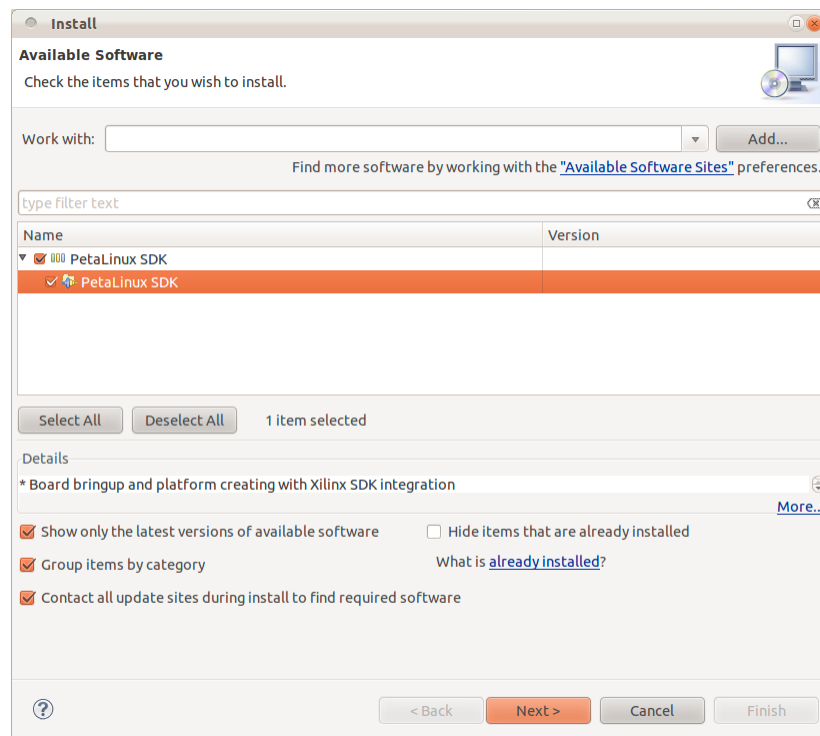


Figure 2: Install New Software Wizard

7. Click **Next**.
8. Please read the license agreement before accepting, once complete click **Finish** to install.
9. When prompted that you are installing "unsigned content" click **Ok** to continue.

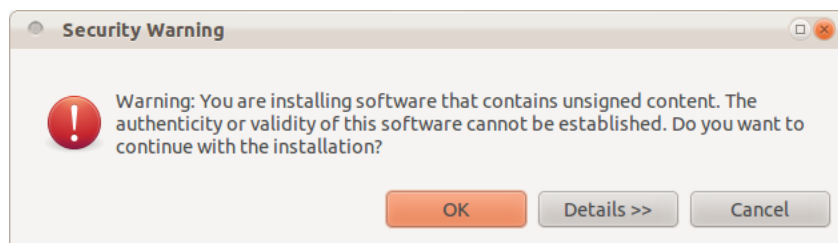


Figure 3: Unsigned Content Warning

10. When prompted to restart Eclipse select **Restart Now**.

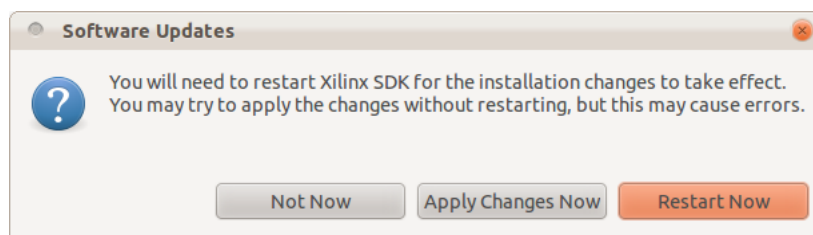


Figure 4: Restart Eclipse to Apply Software Updates

11. The PetaLinux Eclipse SDK is installed.

Getting Started

Starting PetaLinux Eclipse SDK

To launch Eclipse with the PetaLinux SDK the current PetaLinux environment must first be sourced (see prerequisites for more information).

1. Launch Eclipse

```
$ eclipse
```

or Xilinx SDK.

```
$ xsdk
```

2. Select the Workspace.



RECOMMENDED: *It is recommended that you create one workspace for all PetaLinux work. The recommended location is "\$PETALINUX/workspace".*

3. The workspace will be opened/created and the environment will load the PetaLinux SDK.

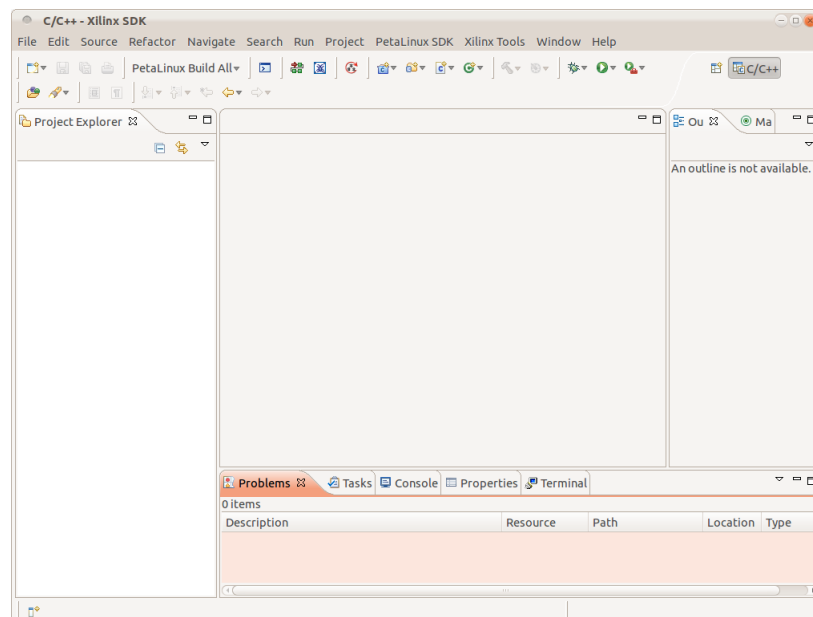


Figure 5: Environment with PetaLinux SDK

Installing a BSP

PetaLinux includes reference designs for you to start working with and customise for your own projects. These are provided in the form of installable BSP (Board Support Package) files, and include all the necessary design and configuration files, including pre-built and tested hardware and software images, ready for download to your board or for booting in the QEMU system simulation environment.

The PetaLinux Eclipse SDK provides the ability to install BSPs, and can be used in combination with the `petalinux-install-bsp` command which is documented in the *PetaLinux SDK Installation Guide (UG976)*.

To install a BSP using the PetaLinux Eclipse SDK follow the below steps:

1. Open the Install BSP Wizard via the **PetaLinux SDK > Install BSP** menu.

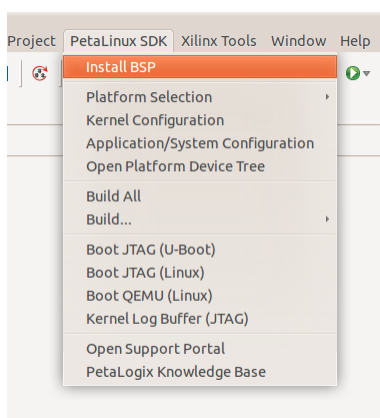


Figure 6: Install BSP Menu Item

2. Navigate and select the ".bsp" file, and click **Ok**.
3. Once complete the BSP(s) will be installed.

Rebuilding a Reference Design Software Image

So far you have tested the PetaLinux reference design pre-built software image both with QEMU and on hardware. You can also rebuild the software image of a reference design software platform. The following section covers the building of the image and the subsequent sections cover testing on QEMU and on hardware.

1. Select the platform to rebuild through the **PetaLinux SDK > Platform Selection** sub menu.

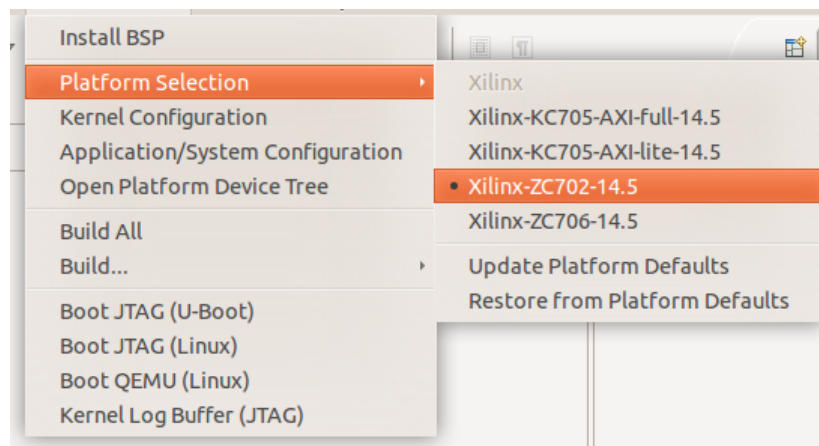


Figure 7: PetaLinux SDK Platform Selection Sub Menu

2. Select **All** from the Build Menu to rebuild the image.

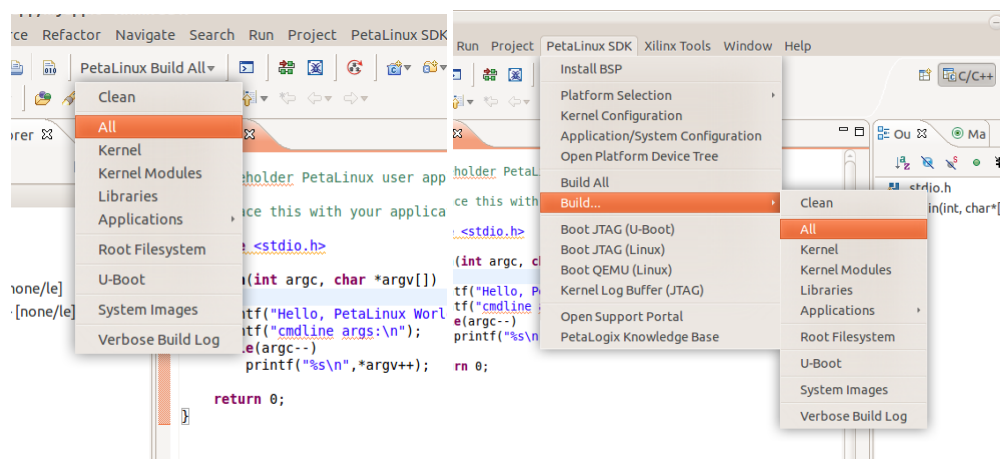


Figure 8: PetaLinux Build Menu

3. The build progress is displayed and the log is shown in a Console window.

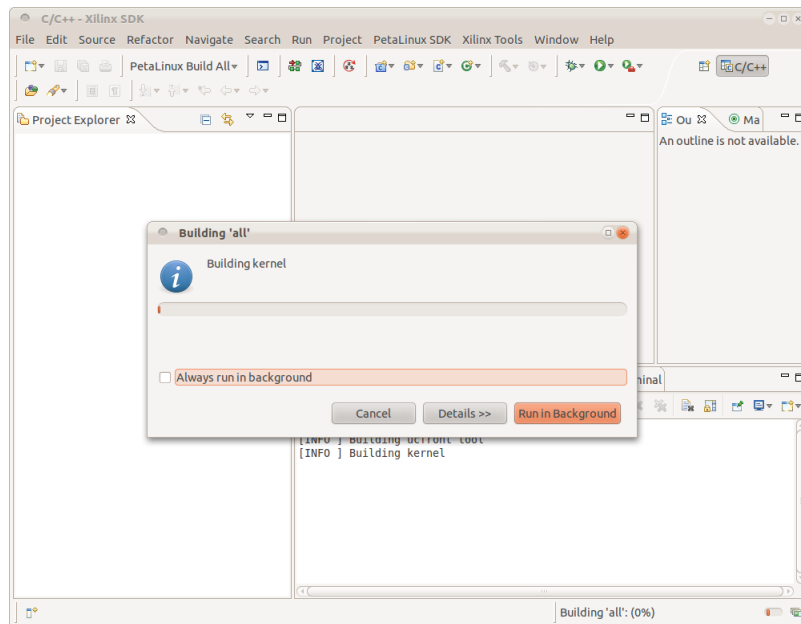


Figure 9: PetaLinux Build

Test a Software Image in QEMU

Now that you have a software system image built, its time to test it out.

1. Run QEMU using the **PetaLinux > Boot QEMU (Linux)** menu item. This will run the default configuration of QEMU for the current Platform.

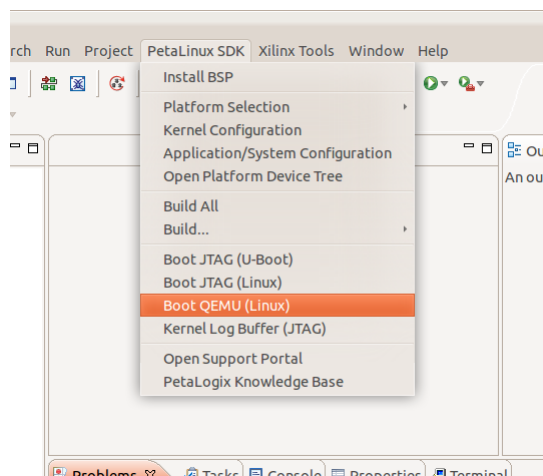


Figure 10: Boot QEMU (Linux) Menu Item

2. A QEMU terminal will appear in the foreground connected to the console of QEMU.

```
INIT: version 2.88 booting
Starting Bootlog daemon: bootlogd.
Creating /dev/flash/* device nodes
Configuring network interfaces... udhcpc (v1.20.2) started
Sending discover...
xemacps e000b000.ps7-ethernet: Set clk to 124999998 Hz
xemacps e000b000.ps7-ethernet: link up (1000/FULL)
Sending select for 192.168.11.3...
Lease of 192.168.11.3 obtained, lease time 864000
/etc/udhcpc.d/50default: Adding DNS 172.19.128.1
/etc/udhcpc.d/50default: Adding DNS 172.19.129.1
done.
starting Busybox inet Daemon: inetd... done.
Starting uWeb server:
INIT: Entering runlevel: 5
Starting network benchmark server: netserver.
Starting syslogd/klogd: done
Stopping Bootlog daemon: bootlogd.
```

```

  _ _ _ _ _
 | _ _ _ \   | |   | |   | |   ( _ )
 | | _ / / _ _ | | _ _ _ _ | | _ _ _ _ _
 | _ _ / / - \ | _ _ | / - ' | | | | ' - \ | | | \ \ / /
 | |   | _ _ / | | ( _ | | | _ _ _ | | | | | | _ | | > <
 \ _ |   \ _ _ | \ _ | \ _ , - \ _ _ _ _ / | | | | | | \ _ , - | / - \ \ \
```

PetaLinux v2013.04 (Yocto 1.3) Xilinx-ZC702-14_5 ttyPS0

Xilinx-ZC702-14_5 login:

Figure 11: QEMU Terminal

To run QEMU using custom parameters or with a custom image use the QEMU external tool configurations. Open **Run > External Tools > External Tools Configurations....**

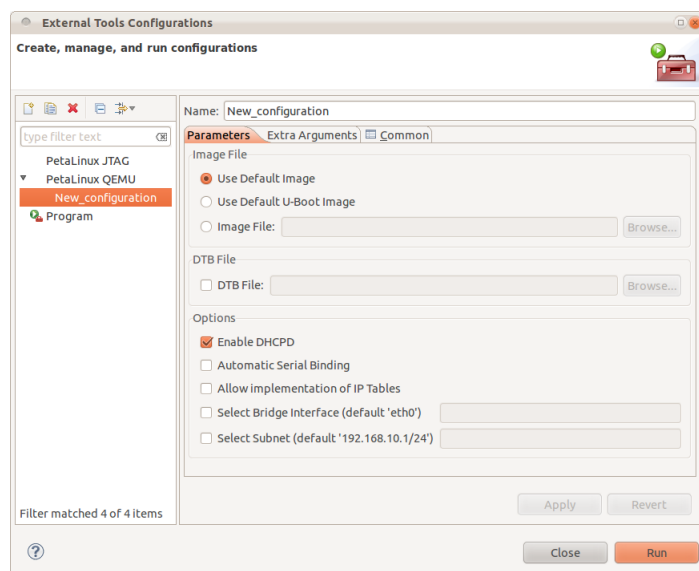


Figure 12: PetaLinux QEMU External Tool Configuration

Test a Software Image on the Board

Before downloading the images to the board, the board must first be programmed with the corresponding FPGA bitstream. The current release of the PetaLinux Eclipse SDK tools does not provide the ability to download FPGA bitstreams for pre-built or non-pre-built platforms.

Follow the instructions from the Test New Software Image on Hardware section of the *PetaLinux SDK Getting Started Guide (UG977)*, to download the bitstream, connect the board, serial and JTAG correctly.

Direct Kernel Image Download

1. JTAG boot the kernel image to the board using the **PetaLinux SDK > Boot JTAG (Linux)** menu item. This will load the current Linux Kernel image onto the board and execute it.

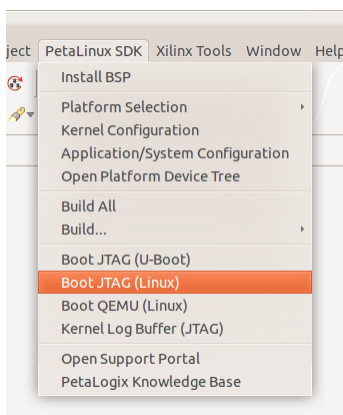


Figure 13: Boot JTAG (Linux) Menu Item

2. Watch the serial console, you should see the Linux booting messages shown on the serial console.

Network Boot Kernel Image

1. Make sure that the Linux image is in your TFTP directory "/tftpboot"
2. JTAG boot the u-boot image to the board using the **PetaLinux SDK > Boot JTAG (U-Boot)** menu item. This will load the current U-Boot image onto the board and execute it.

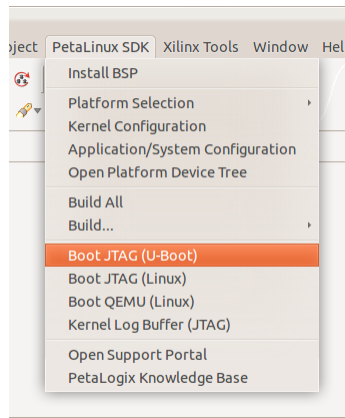


Figure 14: Boot JTAG (U-Boot) Menu Item

3. Watch the serial console. When you see "Hit any key to stop autoboot" on the console, press a key to stop auto boot, as shown below.

```
U-Boot 2013.01 (Apr 17 2013 - 10:58:28)

DRAM:  1 GiB
WARNING: Caches not enabled
MMC:   zynq_sdhci: 0
SF: Detected N25Q128 with page size 64 KiB, total 16 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
U-BOOT for Xilinx-ZC702-14.5

B00TP broadcast 1
DHCP client bound to address 192.168.0.2
Hit any key to stop autoboot:  0
U-Boot-PetaLinux>
```

Figure 15: U-boot

4. Check if the u-boot environment variable `serverip` is set your workstations IP by running u-boot print command on the serial console:

```
U-Boot-PetaLinux> print serverip
```

The `serverip` variable defines the IP address from which u-boot will attempt to load the kernel image, using the TFTP protocol. It should correspond to the IP address of your workstation.

When PetaLinux builds u-boot, it automatically set your workstations IP as the default u-boot `serverip`. However, any previously saved u-boot settings in flash will override this default.

5. Set the u-boot `serverip` to the IP of your workstation by running this command on the u-boot:

```
U-Boot-PetaLinux> set serverip <Your workstation IP>
```

6. Download the PetaLinux image with TFTP and then boot the image by running this command on u-boot:

```
U-Boot-PetaLinux> run netboot
```

You should be able to see the PetaLinux booting messages on the serial communication console.

7. Using u-boot and TFTP to load new kernel images is a great time-saver during the system development process, taking just seconds, instead of minutes that are required for a full JTAG boot of the Linux image.

Board Bringup

Preparation

Before using Eclipse to bringup a new board the sections up to Create New PetaLinux SDK Software Platform of the *PetaLinux SDK Board Bringup Guide (UG980)* document must be completed.

Once complete your workspace should include the PetaLinux BSP Software Project.



TIP: Use the workspace that is to be used for Eclipse work for the XSDK import sections of the bring up guide.

Create and Configure New Software Platform

The next step is to import the BSP configuration and settings into the PetaLinux environment.

1. Select the BSP project to import (in this example **petalinux_bsp_0**).

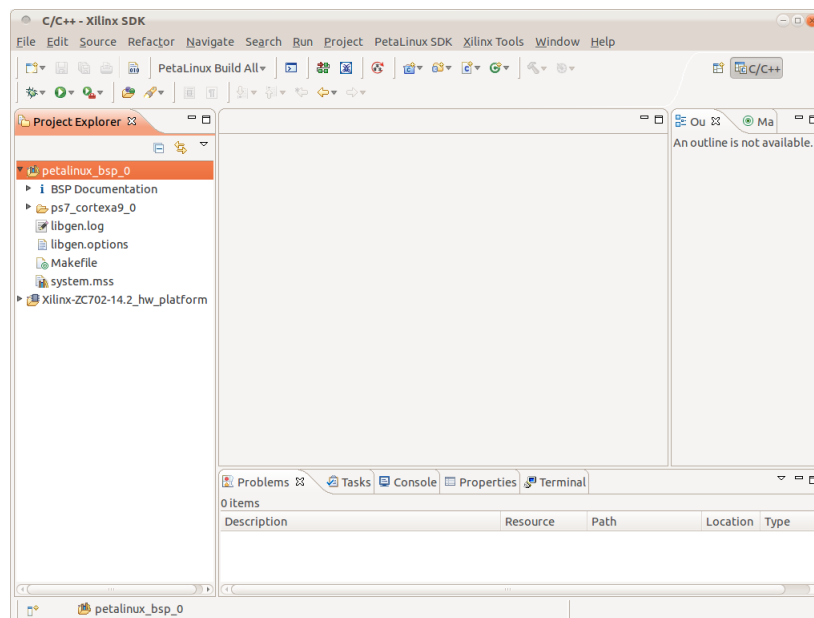


Figure 16: Workspace with BSP prepared

2. Right click the project and select the **Copy BSP to PetaLinux** menu option.

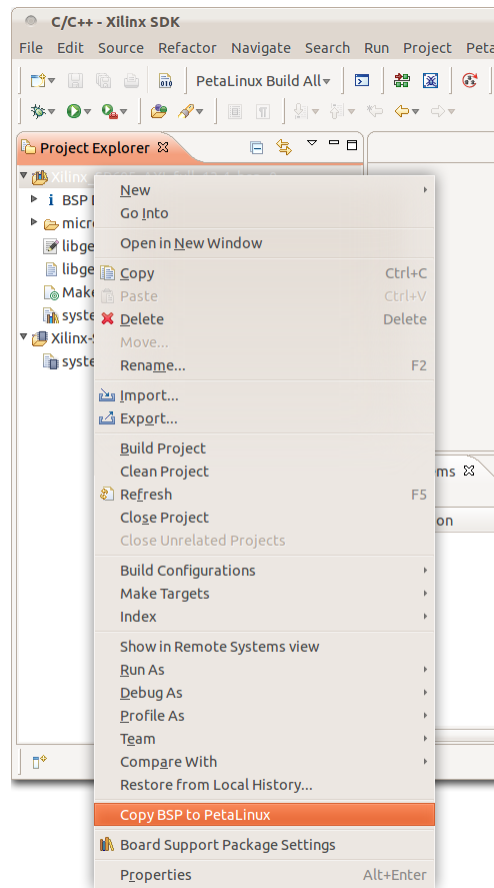


Figure 17: Copy BSP to PetaLinux Menu Item

3. Specify the name of the new Platform, and Vendor, and click **Finish**.
4. If the Platform does not already exist it will be created, its architecture will be determined by the BSP which is being imported.
5. Once the Platform is created the configuration will be imported.

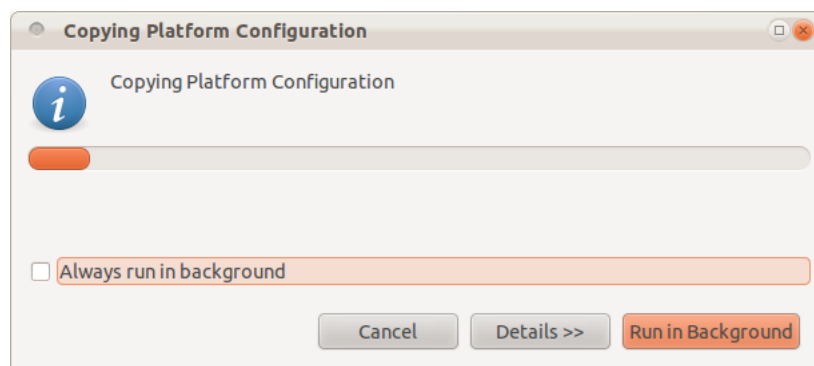


Figure 18: Copying Platform Configuration

6. Once complete the Platform is ready for use in the PetaLinux environment.

Build PetaLinux

The Platform is now ready to be built. Select **All** from the PetaLinux Build Menu.

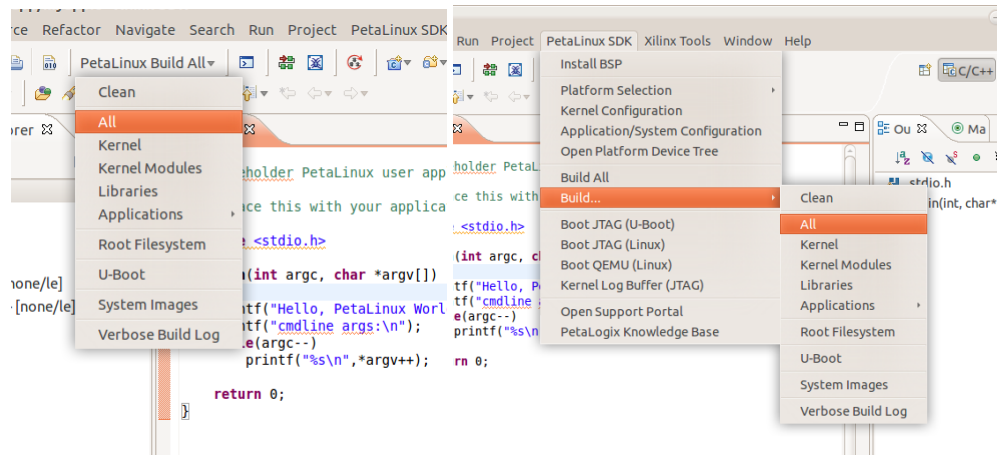


Figure 19: PetaLinux Build Menu

Once the platform is built it can be tested on hardware or in QEMU please refer to sections Test a Software Image on the Board and Test a Software Image in QEMU of this document.

Application Development

Application Creation

PetaLinux Eclipse SDK provides a tool to create PetaLinux user applications from either C or C++ templates.

These templates include application source code and Makefiles so that you can easily configure and compile applications for PetaLinux systems and install them into the root file system.

The basic steps are as follows:

1. Open the **New Project Wizard**.

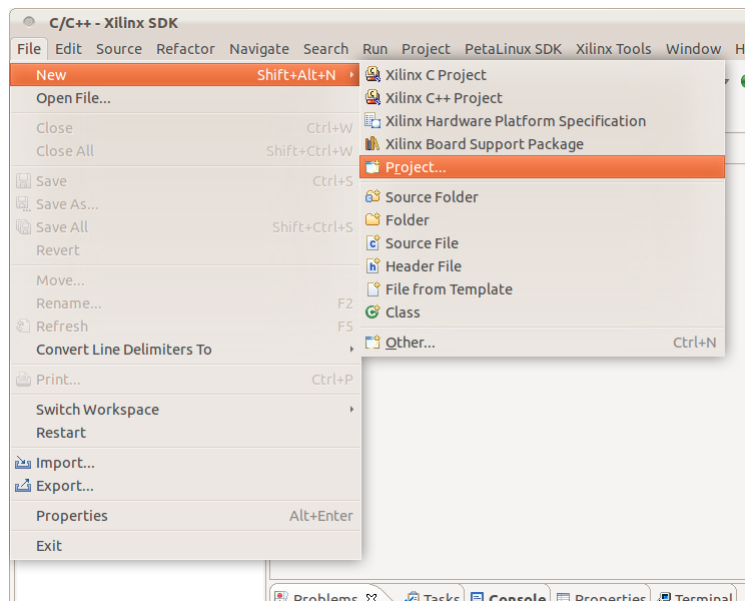


Figure 20: New Project Menu

2. In the New Project Wizard, select the **PetaLinux New Application** item and click **Next**.

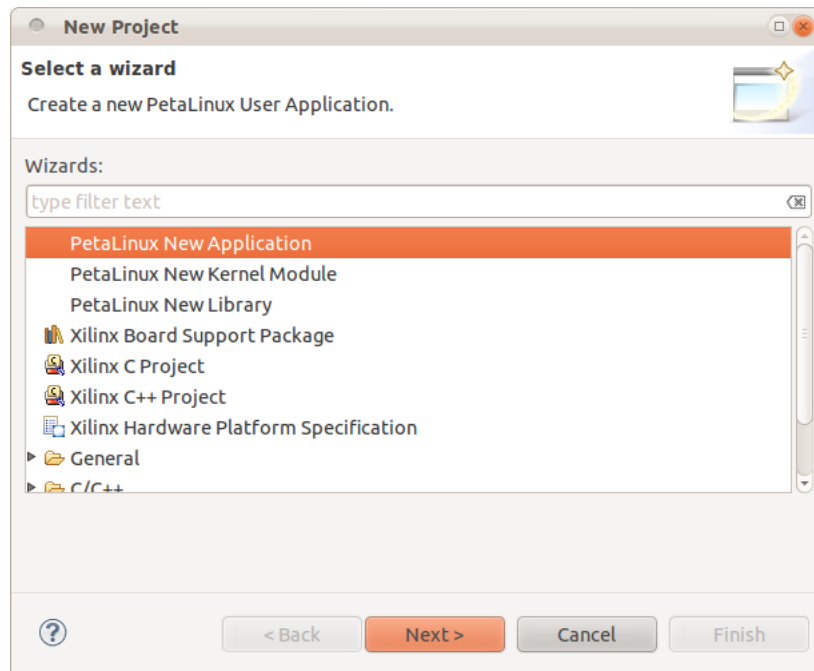


Figure 21: New Project Wizard

3. Enter the name of the Application myapp, and select the Style **C** for a C application. The **Enable the Application** check box will automatically select the application for the currently selected platform. Once complete click **Finish**.

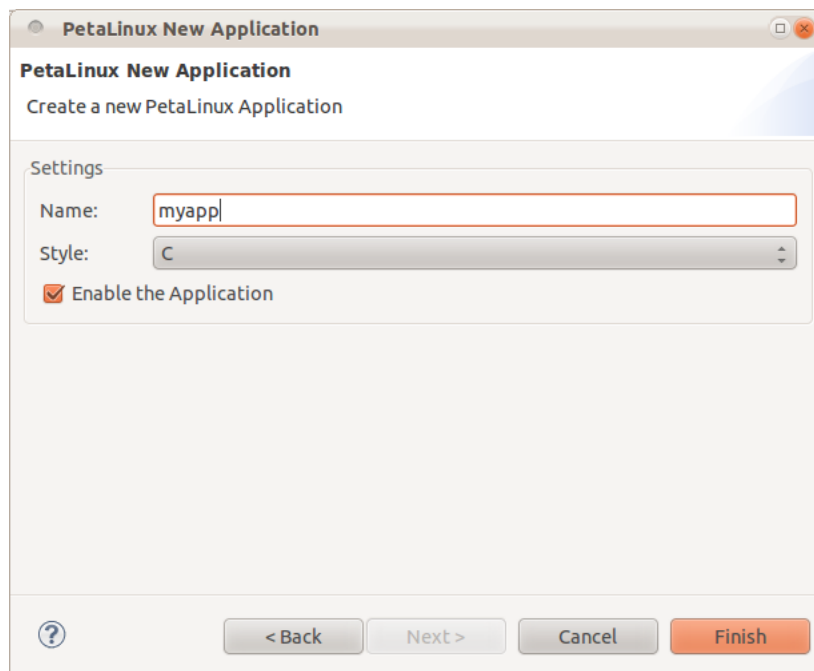


Figure 22: New Project Wizard

- The Wizard will complete the action and create the PetaLinux Application. Once complete the Application will appear as a project from within the Eclipse environment.

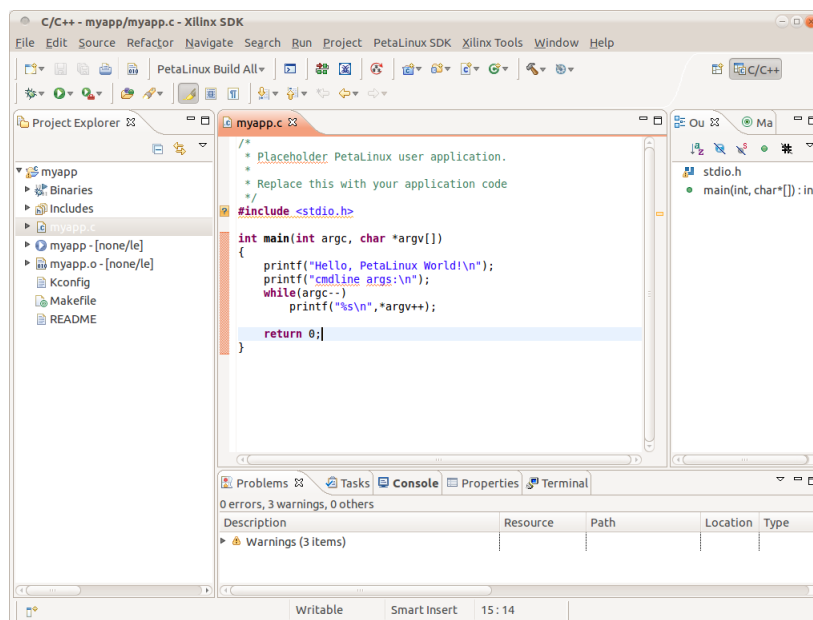


Figure 23: New User Application, myapp.c

Application System Configuration

In order for the application to be included in the system build and resulting romfs the application must be selected in the PetaLinux System Configuration.

- Open the PetaLinux System Configuration via the **PetaLinux Tools > Application/System Configuration** menu.

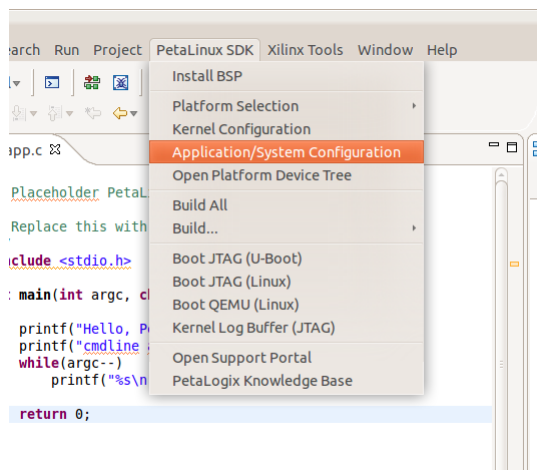


Figure 24: Application/System Configuration Menu

2. Navigate the Application/System Configuration Menuconfig using the Up/Down Arrow and the Enter keys.

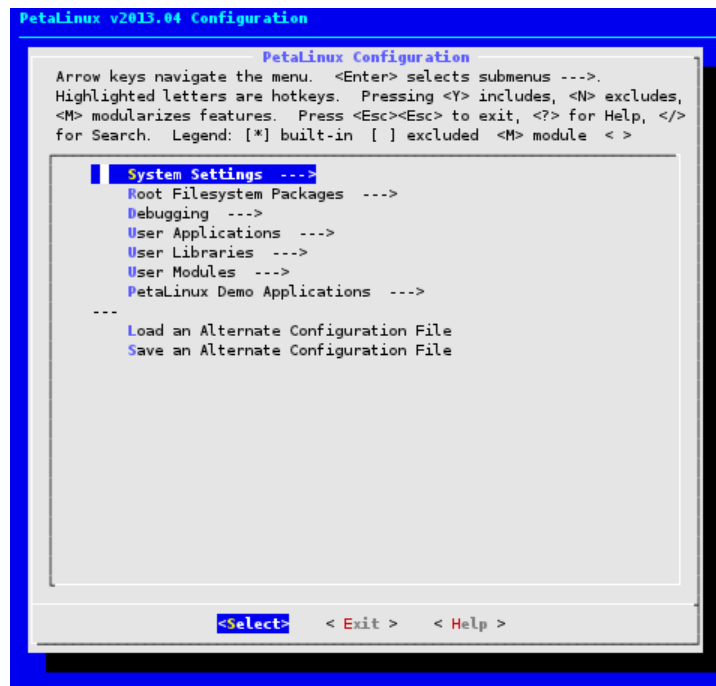


Figure 25: Application/System Configuration Menuconfig

3. Navigate to the **Custom User Applications** sub-menu.
4. You will see the newly created user application myapp listed in the menu.



Figure 26: Application/System Configuration Menuconfig, "Custom User Applications"

5. Select the **myapp** item and press the Space key to enable it.
6. Exit the PetaLinux System Configuration.
7. When prompted to save the configuration select **Yes** and press the Enter key.

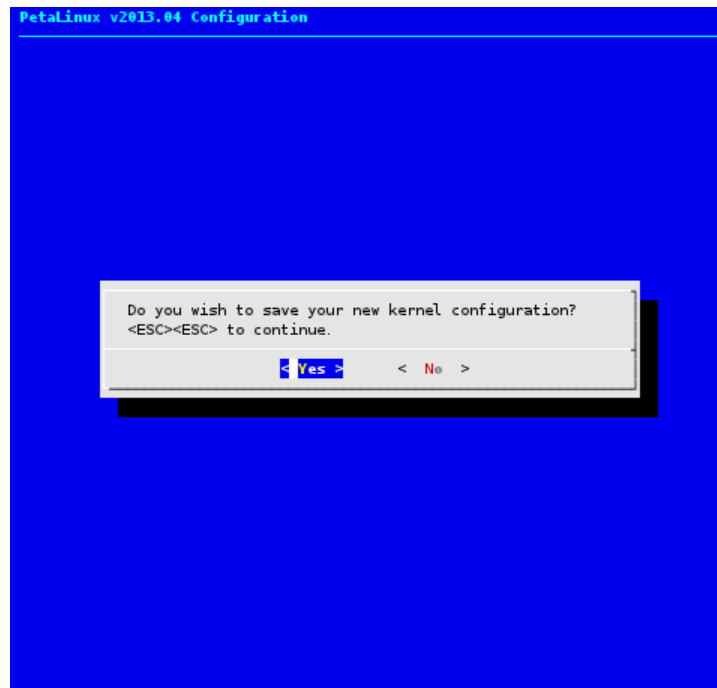


Figure 27: Application/System Configuration Menuconfig, Save Confirmation

8. It will take a few seconds for PetaLinux to apply the configuration change, please wait until the window is closed.

NOTE: While this is happening XSDK/Eclipse will be disabled.

Building Applications

Once you have created the new application, the next step is to compile and build it. By default Eclipse will automatically compile your application for you when you change any source. To build the application into the PetaLinux System complete the following steps:

1. Select the **PetaLinux Build Menu**.
2. Select **All** to build all applications, libraries, kernel, u-boot and romfs.

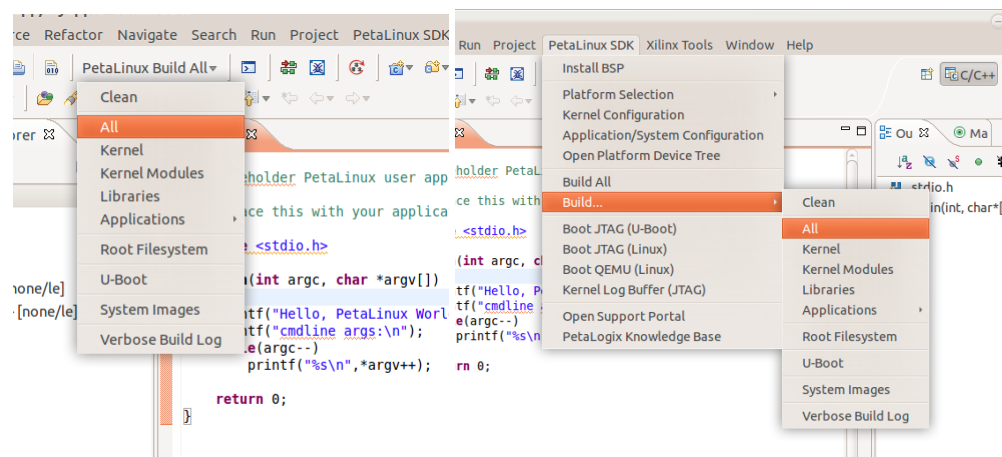


Figure 28: PetaLinux Build Menu

3. Whilst building the progress is shown and the output log of the build is shown in a Console window within Eclipse.

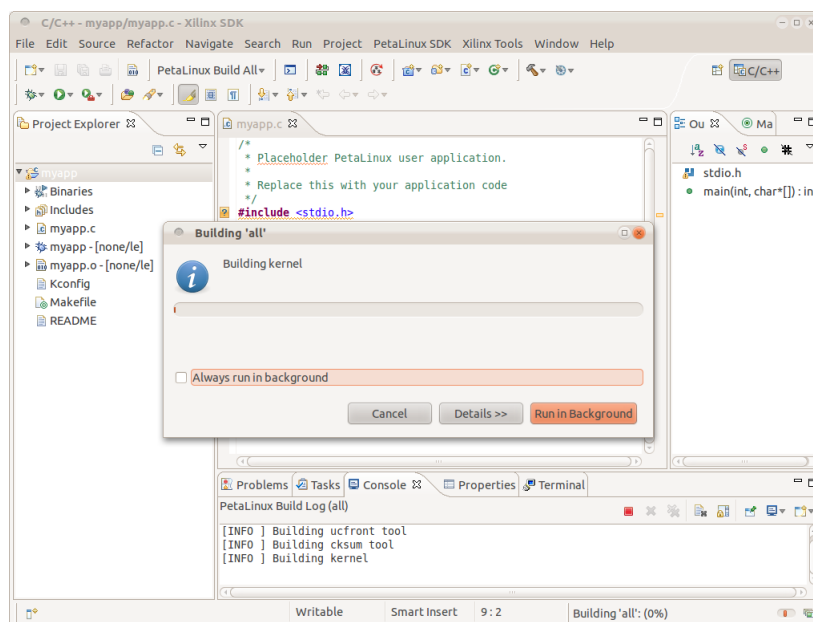


Figure 29: PetaLinux Build

Testing Applications

Having created and compiled the new application, it is time to test it.

1. Boot the newly created system image, either in QEMU or on your hardware board. (Refer to section Test New Software Image with QEMU and section Test New Software Image on Hardware for more information)

2. Confirm your user application is present on the PetaLinux system by running the following command on the system login console:

```
# ls /bin
```

Unless you have changed the user application's Makefile to put the user application to another location, the user application will be put into the "/bin" directory.

3. Run your user application on the PetaLinux system console:

```
# <user-application-executable-file-name>
```

e.g. to run user application myapp:

```
# myapp
```

4. Confirm the result of the application is as expected. If the new application is missing from the PetaLinux filesystem, double check to make sure that you completed the `romfs` and `image` Make targets from the PetaLinux Build. These ensure that your application binary is copied into the root filesystem staging area, and that the PetaLinux system image are updated with this new filesystem.

Preparing the build system for debugging

1. Change the user application Makefile to disable compiler optimizations. Compiler optimisations make debugging difficult because the compiler can re-order or remove instructions that do not impact the program result, making debugging difficult.

Here is an example taken from a changed Makefile:

```
...
ROMFSINST=$(ROOTDIR)/tools/romfs-inst.sh

CFLAGS += -O0

APP = myapp
...
```

2. Open the PetaLinux System Configuration via the **PetaLinux SDK > Application/System Configuration** menu.

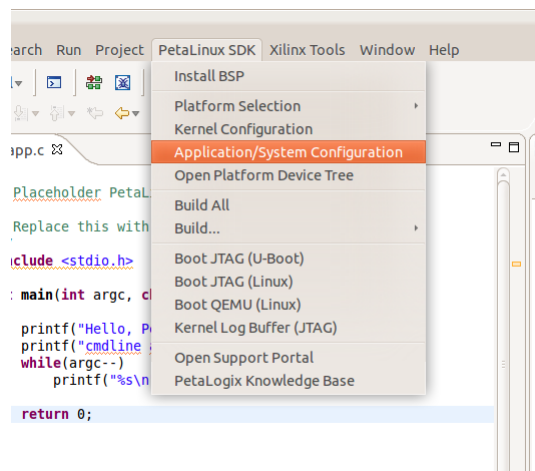


Figure 30: Application/System Configuration Menu

3. Scroll down the **PetaLinux Configuration** menu to **Debugging**:



Figure 31: PetaLinux Configuration - Scrolling down to Debugging option

4. Select the **Debugging** sub-menu:



Figure 32: Debugging Menu

5. Ensure that **build debugable libraries** and **build debugable applications** are all selected.
6. From the top level menu, navigate down to the **Root Filesystem Packages > base > external-csl-toolchain** sub-menu and enable the **gdbserver** package:



Figure 33: Root Filesystem Packages - GDB Server

7. Exit Application/System Configuration and select **Yes** to save the configuration.
8. Rebuild the PetaLinux image, select **All** to rebuild the image.

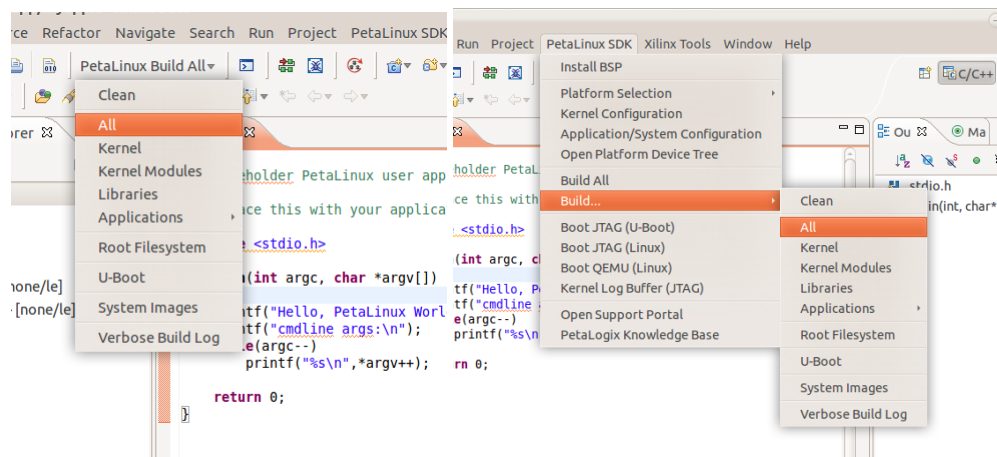


Figure 34: PetaLinux Build Menu

Debugging Applications with GDB

1. Boot your board (or QEMU) with the new image created previously.
2. Run gdbserver with the user application on the PetaLinux console:

```
# gdbserver host:1234 /bin/myapp
Process /bin/myapp created; pid = 73
Listening on port 1234
```

1234 is the "gdbserver" port - it can be any unused port number

3. Open the **Run > Debug Configurations** menu item.

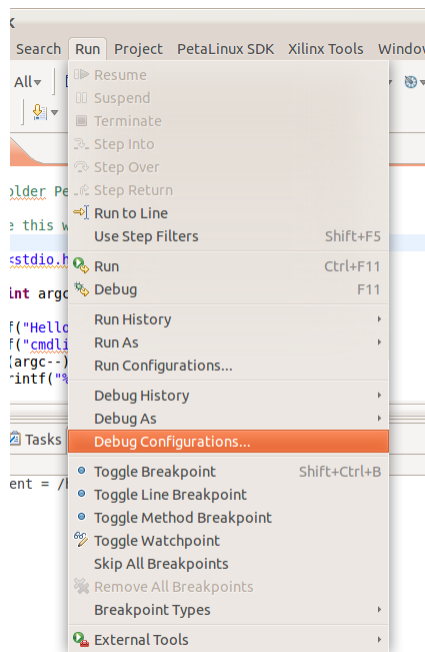


Figure 35: Debug Configurations... Menu

4. Create a new **PetaLinux Debug** configuration. Select the target project as **myapp** and the target executable as **myapp**.

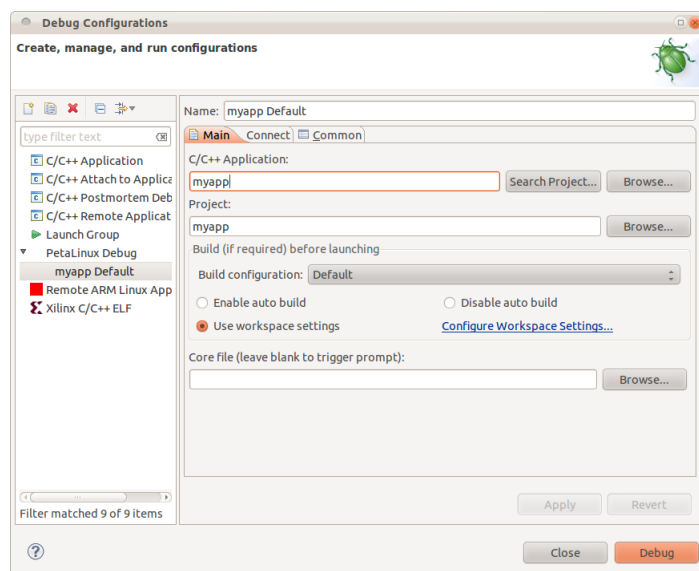


Figure 36: PetaLinux Debug Launch Configuration

5. Change to the **Connect** tab of the configuration and set the target IP address and Port.

- Use the IP address of the PetaLinux system, e.g.: 192.168.0.10. If you are not sure about the IP address, run ifconfig on the target console to check.
- Use the port 1234. If you chose a different gdbserver port number in the earlier step, use that value instead.

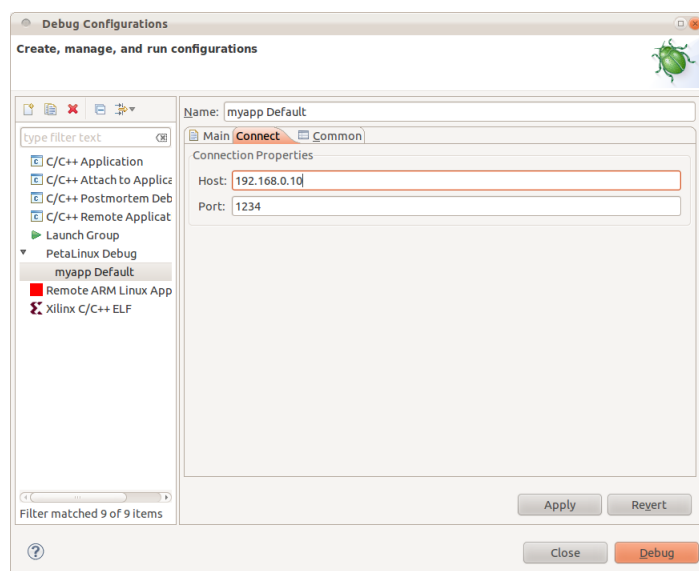


Figure 37: PetaLinux Debug Launch Configuration, Connect Tab

6. Click **Apply** to save the settings of the Launch Configuration, and click **Debug** to begin debugging.

TIP: You can access your Favorite/Recent Debug Launch Configurations and the Debug Configurations... dialog from the **Debug Toolbar Menu**,

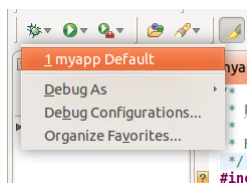


Figure 38: Debug Toolbar Menu

7. The Debugging Session will be launched, and Eclipse will switch to the Debug Perspective.

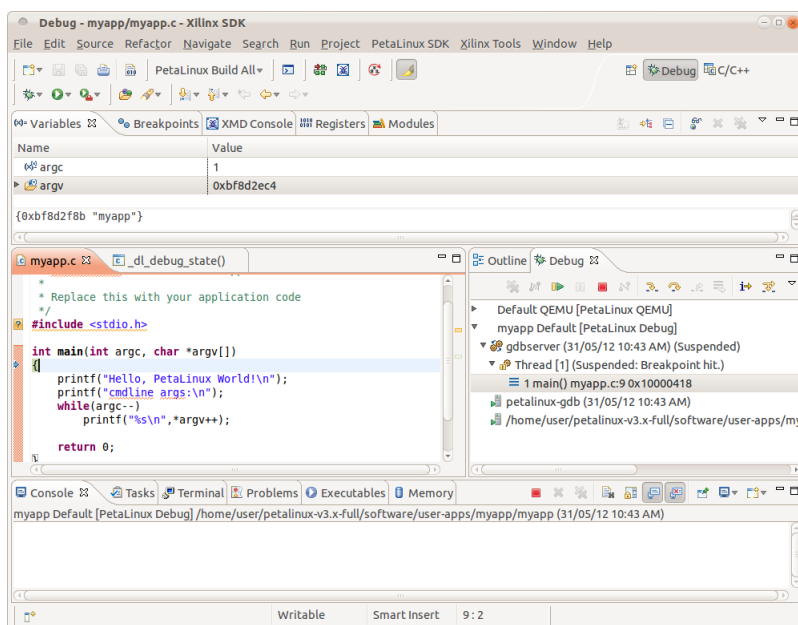


Figure 39: Debug Session

8. The session is now connected to the remote GDB Server, and has started the application, breaking at the function main.
9. Add a break point to the application by double clicking on the side strip in the Code window for the application code file "myapp.c". A small blue breakpoint icon should appear in the side strip for each breakpoint in the code file.



```

*/
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello, PetaLinux World!\n");
    printf("cmdline args:\n");
    while(argc--)
        printf("%s\n", *argv++);

    return 0;
}

```

Figure 40: Code Window with Breakpoints active

10. Run the program, using the **Resume** command via the Control Flow toolbar.

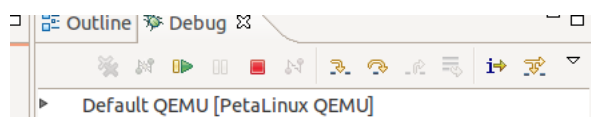


Figure 41: Control Flow Toolbar

11. Try the **Step into** and **Step over** commands. Try setting and removing breakpoints.
12. When the program finishes, the GDB server application on the target system will exit. Here is an example of messages shown on the console:

```

~ # gdbserver host:1234 /bin/myapp
Process /bin/myapp created; pid = 58
Listening on port 1234
Remote debugging from host 192.168.0.9
Hello, PetaLinux World!
cmdline args:
/bin/myapp

Child exited with status 0
GDBserver exiting
~ #

```

Importing Existing PetaLinux Applications and Libraries

To work with projects inside the Eclipse environment the projects must exist in the Eclipse workspace. To work with existing projects (applications, libraries and kernel modules) that were created by the command line tools `petalinux-new-...` they must first be imported into the Eclipse workspace using the Import Wizard.

1. Open the Import Wizard via the **File > Import** menu.

Importing Existing PetaLinux Applications and Libraries

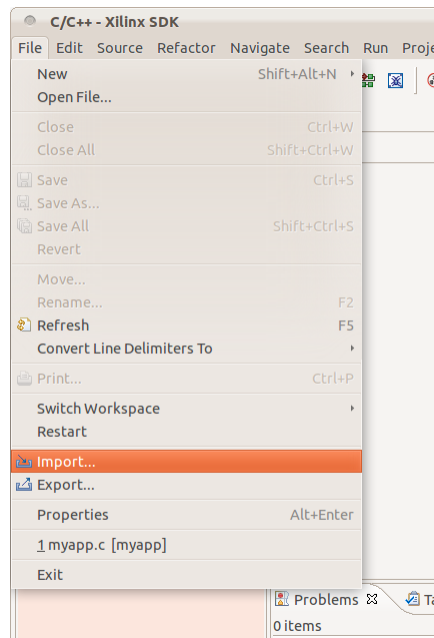


Figure 42: Import Wizard Menu

2. Select the **PetaLinux > Import Existing User Application/Library** wizard and click **Next**.

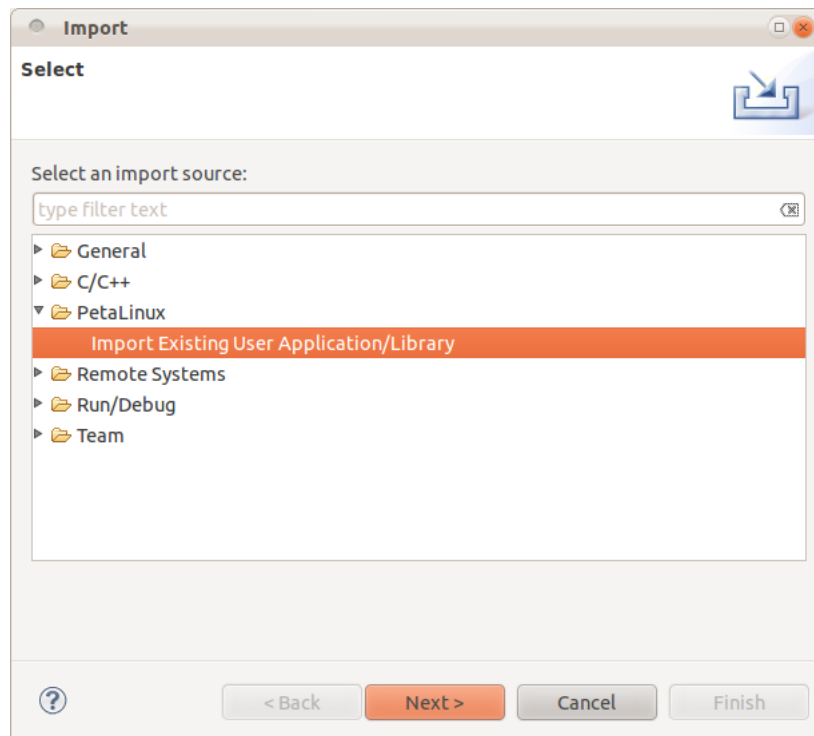


Figure 43: Import Wizard

3. Input the location of the application/library to import. This application/library must already be inside the PetaLinux working tree. Once entered click **Finish**.

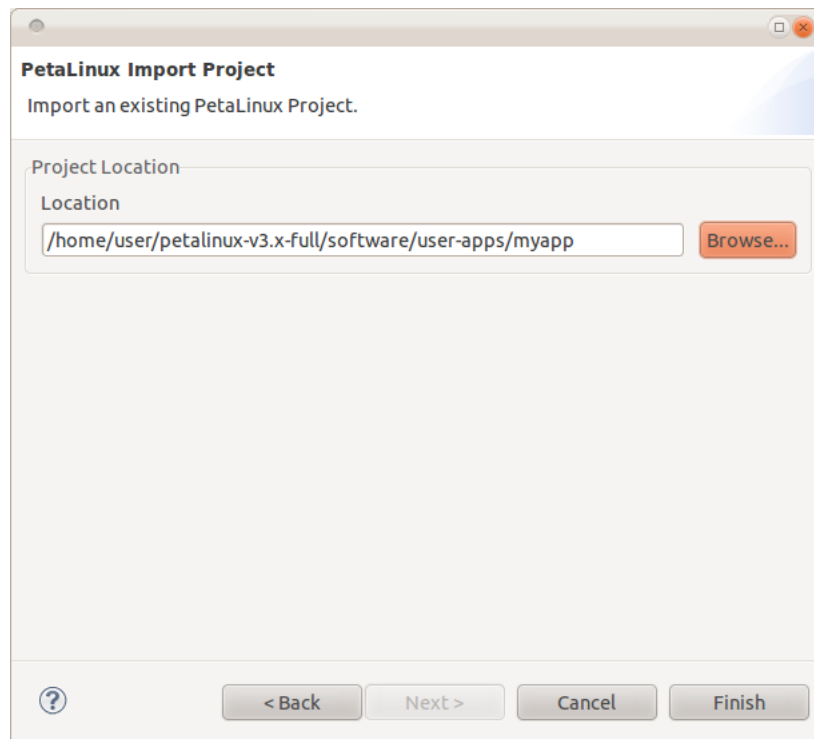


Figure 44: PetaLinux Import Project Wizard

4. The project will be imported.

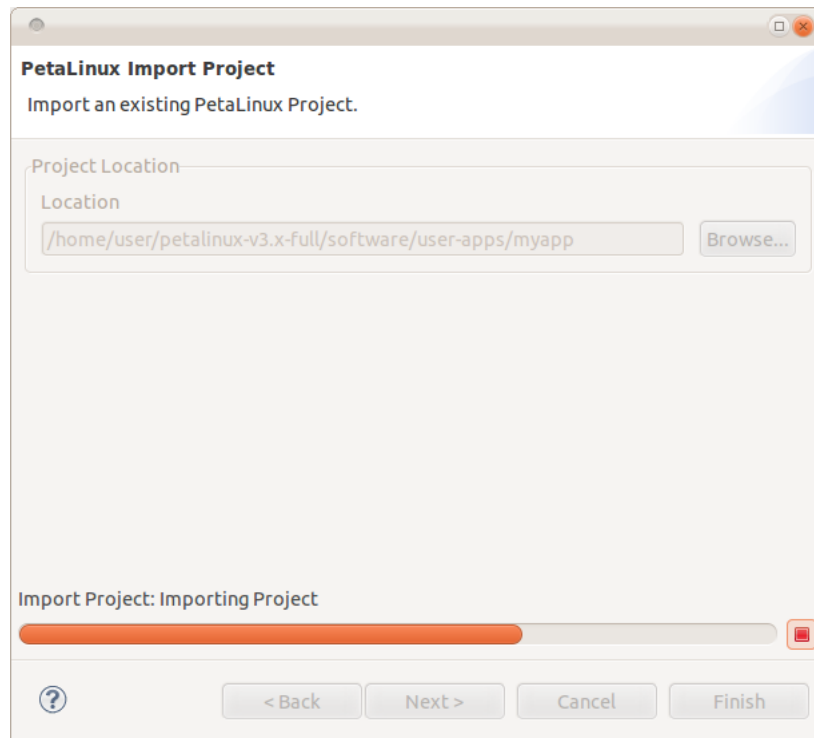


Figure 45: PetaLinux Import Project Wizard

5. The project is now accessible as an Eclipse project.

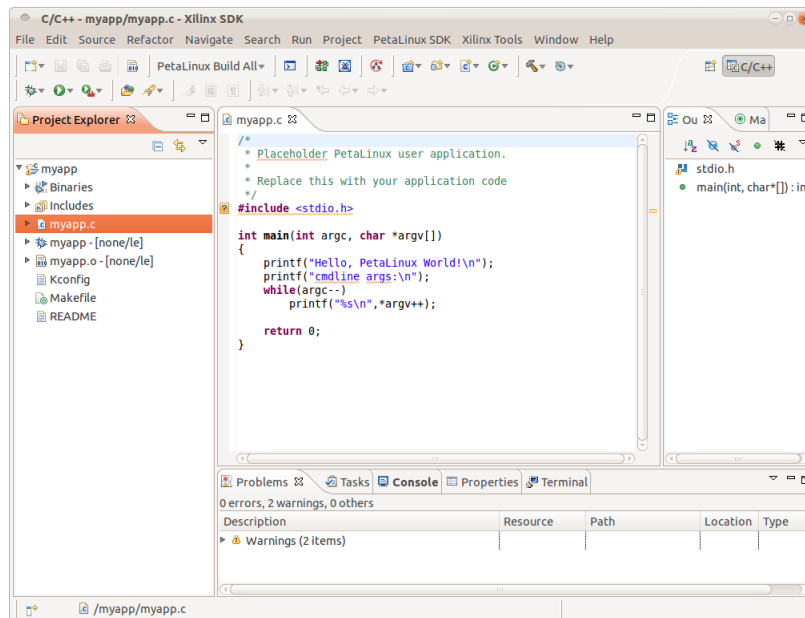


Figure 46: PetaLinux Project Imported

Additional Resources

References

- PetaLinux SDK Application Development Guide (UG981)
- PetaLinux SDK Board Bringup Guide (UG980)
- PetaLinux SDK Eclipse Plugin Guide (UG979)
- PetaLinux SDK Firmware Upgrade Guide (UG983)
- PetaLinux SDK Getting Started Guide (UG977)
- PetaLinux SDK Installation Guide (UG976)
- PetaLinux SDK QEMU System Simulation Guide (UG982)