

# **phyFLEX<sup>®</sup> -i.MX 6**

## **Quickstart Instructions**

Document No.: **L-781e\_7**

Release No.: **S0-525v9**

**Edition: September 2015**

Copyrighted products are not explicitly indicated in this manual. The absence of the trademark (™, or ®) and copyright (©) symbols does not imply that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual.

The information in this document has been carefully checked and is considered to be entirely reliable. However, PHYTEC Messtechnik GmbH assumes no responsibility for any inaccuracies. PHYTEC Messtechnik GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. PHYTEC Messtechnik GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages that might result.

Additionally, PHYTEC Messtechnik GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. PHYTEC Messtechnik GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

© Copyright 2015 PHYTEC Messtechnik GmbH, D-55129 Mainz.

Rights - including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part - are reserved. No reproduction may occur without the express written consent from PHYTEC Messtechnik GmbH.

	EUROPE	NORTH AMERICA	FRANCE
Address:	PHYTEC Messtechnik GmbH Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW Bainbridge Island, WA 98110 USA	PHYTEC France 17, place Saint-Etienne F-72140 Sillé-le-Guillaume FRANCE
Sales:	+49 6131 9221-32 <a href="mailto:sales@phytec.de">sales@phytec.de</a>	+1 800 278-9913 <a href="mailto:sales@phytec.com">sales@phytec.com</a>	+33 2 43 29 22 33 <a href="mailto:info@phytec.fr">info@phytec.fr</a>
Technical Support:	+49 6131 9221-31 <a href="mailto:support@phytec.de">support@phytec.de</a>	+1 206 780-9047 <a href="mailto:support@phytec.com">support@phytec.com</a>	<a href="mailto:support@phytec.fr">support@phytec.fr</a>
Fax:	+49 6131 9221-33	+1 206 780-9135	+33 2 43 29 22 34
Web Site:	<a href="http://www.phytec.de">http://www.phytec.de</a> <a href="http://www.phytec.eu">http://www.phytec.eu</a>	<a href="http://www.phytec.com">http://www.phytec.com</a>	<a href="http://www.phytec.fr">http://www.phytec.fr</a>

	INDIA	CHINA
Address:	PHYTEC Embedded Pvt. Ltd. #16/9C, 3rd Main, 3rd Floor, 8th Block, Opp. Police Station Koramangala, Bangalore-560095 INDIA	PHYTEC Information Technology (Shenzhen) Co. Ltd. Suite 2611, Floor 26, Anlian Plaza, 4018 Jin Tian Road Futian District, Shenzhen CHINA 518026
Sales:	+91-80-4086 7046/48 <a href="mailto:sales@phytec.in">sales@phytec.in</a>	+86-755-3395-5875 <a href="mailto:sales@phytec.cn">sales@phytec.cn</a>
Technical Support:	+91-80-4086 7047 <a href="mailto:support@phytec.in">support@phytec.in</a>	<a href="mailto:support@phytec.cn">support@phytec.cn</a>
Fax:		+86-755-3395-5999
Web Site:	<a href="http://www.phytec.in">http://www.phytec.in</a>	<a href="http://www.phytec.cn">http://www.phytec.cn</a>

<b>Conventions, Abbreviations and Acronyms .....</b>	<b>iii</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Rapid Development Kit Documentation .....	1
1.2 Professional Support Packages Available .....	1
1.3 Overview of these QuickStart Instructions .....	2
1.4 System Requirements .....	2
1.5 Software used in the virtual machine hard disk image.....	2
1.5.1 Ubuntu .....	2
1.5.2 Eclipse.....	3
1.5.3 Qt Creator .....	3
1.5.4 Yocto Project .....	3
<b>2 Getting Started .....</b>	<b>4</b>
2.1 Preparing and Starting the virtual machine .....	4
2.2 Starting the Hardware .....	5
<b>3 Working with Eclipse .....</b>	<b>10</b>
3.1 Programming in the C/C++ Perspective.....	10
3.1.1 Work with the Demo Project .....	10
3.1.2 Creating a New Project .....	15
3.1.3 Modifying the Demo Application .....	22
3.1.4 Starting a Program out of Eclipse on the Target.....	25
3.2 Debugging an Example Project .....	27
3.2.1 Starting the GDB Server on the Target .....	28
3.2.2 Configuring and Starting the Debugger in Eclipse.....	28
3.2.3 Setting a Breakpoint .....	33
3.2.4 Stepping through and Watching Variable Contents .....	34
3.2.5 Stepping through and Changing Variable Contents .....	36
3.2.6 Using the Memory Monitor.....	37
<b>4 Working with Qt Creator .....</b>	<b>40</b>
4.1 Stop the Running Qt Demo on the Target .....	40
4.2 Importing the Demo Application .....	40
4.3 Work with the Demo Application .....	42
4.4 Compile and Run the Demo Application on the Target.....	45
4.5 Compile and Run the Demo Application on the Host.....	48
4.6 Debugging the Demo Application .....	49
4.6.1 Using QDebug for simple Debugging Messages.....	49
4.6.2 Using the integrated Qt Creator Debugger .....	51
<b>5 Working with Yocto .....</b>	<b>55</b>
5.1 Add new packages to the Root Filesystem .....	55
5.2 Writing the Root Filesystem into the Target's Flash .....	59
<b>6 Setup your own Linux-Host-PC .....</b>	<b>62</b>
6.1 Essential Settings .....	62
6.1.1 Installing Ubuntu .....	62
6.1.2 Installation of Software Packages .....	64
6.1.3 Set the Git Configuration.....	65
6.1.4 Build the Board Support Package and Install the SDK.....	65
6.1.5 Set up Eclipse and Integrate Plug-ins.....	67
6.1.6 Install and Setup <i>Qt Creator</i> .....	69

6.1.7	Setting up a TFTP server.....	75
6.1.8	Optional Settings.....	76







## Conventions, Abbreviations and Acronyms

The following is a list of the typographical conventions used in this book:

### Conventions

The conventions used in this manual are as follows:

- Signals that are preceded by an "n", "/", or "#" character (e.g.: nRD, /RD, or #RD), or that have a dash on top of the signal name (e.g.: RD) are designated as active low signals. That is, their active state is when they are driven low, or are driving low.
- A "0" indicates a logic zero or low-level signal, while a "1" represents a logic one or high-level signal.
- The hex-numbers given for addresses of I<sup>2</sup>C devices always represent the 7 MSB of the address byte. The correct value of the LSB which depends on the desired command (read (1), or write (0)) must be added to get the complete address byte. E.g. given address in this manual 0x41 => complete address byte = 0x83 to read from the device and 0x82 to write to the device.
- Tables which describe jumper settings show the default position in **bold, blue text**.
- Text in *blue italic* indicates a hyperlink within, or external to the document. Click these links to quickly jump to the applicable URL, part, chapter, table, or figure.
- Text in **bold italic** indicates an interaction by the user, which is defined on the screen.
- Text in *Consolas* indicates an input by the user, without a premade text or button to click on.
- Text in *italic* indicates proper names of development tools and corresponding controls (windows, tabs, commands etc.) used within the development tool, no interaction takes place.
- **White Text on black background** shows the result of any user interaction (command, program execution, etc.)

	At this icon you might leave the path of this Application Guide.
	This is a warning. It helps you to avoid annoying problems.
	You can find useful supplementary information about the topic.
	At the beginning of each chapter you can find information about the time required to read the following chapter.
	You have successfully completed an important part of this Application Guide.
	You can find information to solve problems.

## 1 Introduction

*In this QuickStart Instructions Manual you will find general information about the PHYTEC phyFLEX-i.MX6-Kit and instructions how to start-up with the phyFLEX-i.MX6. You will learn how to...*

- ... use the prepared virtual machine hard disk image.
- ... connect to the target.
- ... handle with the Yocto Project and our Board Support Package to build your own images.
- ... working with Eclipse and Qt Creator .
- ... setup your own linux host system.

*This first Chapter gives a short introduction about the PHYTEC phyFLEX-i.MX6-Kit and it's Quickstart. Also you will find general requirements and information to successfully pass the Quickstart.*

### 1.1 Rapid Development Kit Documentation

This *Rapid Development Kit* (RDK) includes the following electronic documentation in the virtual machine hard disk under `/opt/PHYTEC_Tools/Documentation`.

- The PHYTEC phyFLEX-i.MX6 Hardware Manual
- Hardware schematics
- The Yocto Reference Manual
- The BSP Quickstart
- Image Processing manuals
- These Quickstart Instructions

### 1.2 Professional Support Packages Available

If you have any questions concerning installation and setup, you are welcome to contact our support department.

For more in-depth questions, we offer a variety of custom-tailored packages with different support options (e-mail, phone, direct contact to the developer) and different reaction times.

Please contact our sales team to discuss the appropriate support option if professional support beyond installation and setup is important for you. For more information please refer to the following sources:

<http://www.phytec.de>

<http://www.phytec.eu>

[support@phytec.de](mailto:support@phytec.de)

Also more contact information can be found on *page 2*.

## 1.3 Overview of these QuickStart Instructions

This QuickStart gives you an overview of the phyFLEX-i.MX6. It's content ranges from the first startup, to building your own kernel and file system, to the point of building your own program with *Eclipse* and *Qt Creator*. This Quickstart is structured as follows:

- The *Getting Started* section describes the basics such as configure your host platform and starting the phyFLEX-i.MX6 platform.
- The *Working with Eclipse* explains how to modify an example application, create, build, debug a new project, and copy programs to the phyFLEX-i.MX6 using Eclipse with the C/C++ CDT plugin.
- The *Working with Qt Creator* section explains how to modify an example application, create, build, debug a new project, and copy programs to the phyFLEX-i.MX6 using Qt Creator.
- The *Working with Yocto* section explains how to add new packages in our Board Support Package, build a new root filesystem and copy the root filesystem into the target's flash.
- The *Setup your own Linux-Host-PC* section explains how to setup your own Linux-Host-PC if you don't want to use our virtual machine hard disk image.

## 1.4 System Requirements

The following items will be needed to complete this Quickstart successfully:

- The PHYTEC phyFLEX-i.MX6
- The PHYTEC Development Board with the included DB-9 serial cable, Ethernet crossover cable and AC adapter supplying 12 VDC (min. 2 A)
- The PHYTEC virtual machine hard disk image (*S0-525v9*).
- An IBM-compatible host-PC (586 or higher) with 512 MB RAM (or more)

## 1.5 Software used in the virtual machine hard disk image

### 1.5.1 Ubuntu

*Ubuntu* - which is used as operating system for our virtual machine - is a free and open source operating system based on *Debian Linux*. Basically it is designed for desktop use. Web statistics suggest that *Ubuntu* is one of the most popular operating systems in the Linux desktop environment.

The *Ubuntu* release which we deliver is *14.04.3* and was released on 06. August 2015. *Ubuntu 14.04* code name "*Trusty Tahr*" is designated as a **Long Term Support (LTS)** release and the first stable release was on 17. April 2014. LTS means that it will be supported and updated for five years.



Our *Ubuntu* version comes with *Unity* as desktop environment, *dpkg* as package management system, the update method is based on *APT* (*Advanced Packaging Tool*) and the user space uses *GNU*.

### 1.5.2 Eclipse

The *Eclipse* platform provides support for *C/C++*. Because the *Eclipse* platform is only a framework for developer tools, it does not support *C/C++* directly, instead it uses external plug-ins. This Application Guide shows you how to make use of the *CDT plug-in*.

The *CDT* is an open source project (licensed under the Common Public License) implemented purely in *Java* as a set of plug-ins for the *Eclipse* SDK platform. These plug-ins add a *C/C++* perspective to the *Eclipse* Workbench that can now support *C/C++* development with a number of views and wizards, along with advanced editing and debugging support.

### 1.5.3 Qt Creator

*Qt Creator* is a cross-platform development environment for the Qt framework. Included are a code editor and a Qt Designer to build graphical user interfaces (GUI). It uses the *GNU C/C++* compiler.

### 1.5.4 Yocto Project

The *Yocto* Project is an open source collaboration to create custom Linux-based systems for embedded products regardless of the hardware architecture. We use the *Yocto* Project to create the Board Support Package (BSP) for our hardware.

## 2 Getting Started

### 2.1 Preparing and Starting the virtual machine



To ensure successful introduction to the development with the phyFLEX-i.MX6 we strongly recommend continuing with the prepared virtual machine hard disk image

Nonetheless, if you want to use your already existing environment we explain how to modify your system to get the same experience as with our virtual machine in [section 6](#).

To follow this QuickStart you need our phyFLEX-i.MX6 virtual machine hard disk image. The hard disk image includes all software installations and configurations (e.g. pre-built Yocto BSP, Eclipse, Qt Creator) necessary for a successful start-up of the phyFLEX-i.MX6. You can freely download the compressed virtual machine hard disk image from our website, if you did not purchase an USB flash drive with our image. The compressed image (S0-525v9.tar.xz) has a size of approx. 15 GB and includes a virtual machine disk file (.vmdk) which is suitable for most virtualization software, and which can be mounted to an appropriate Linux machine (tested with VMware Player and VirtualBox).

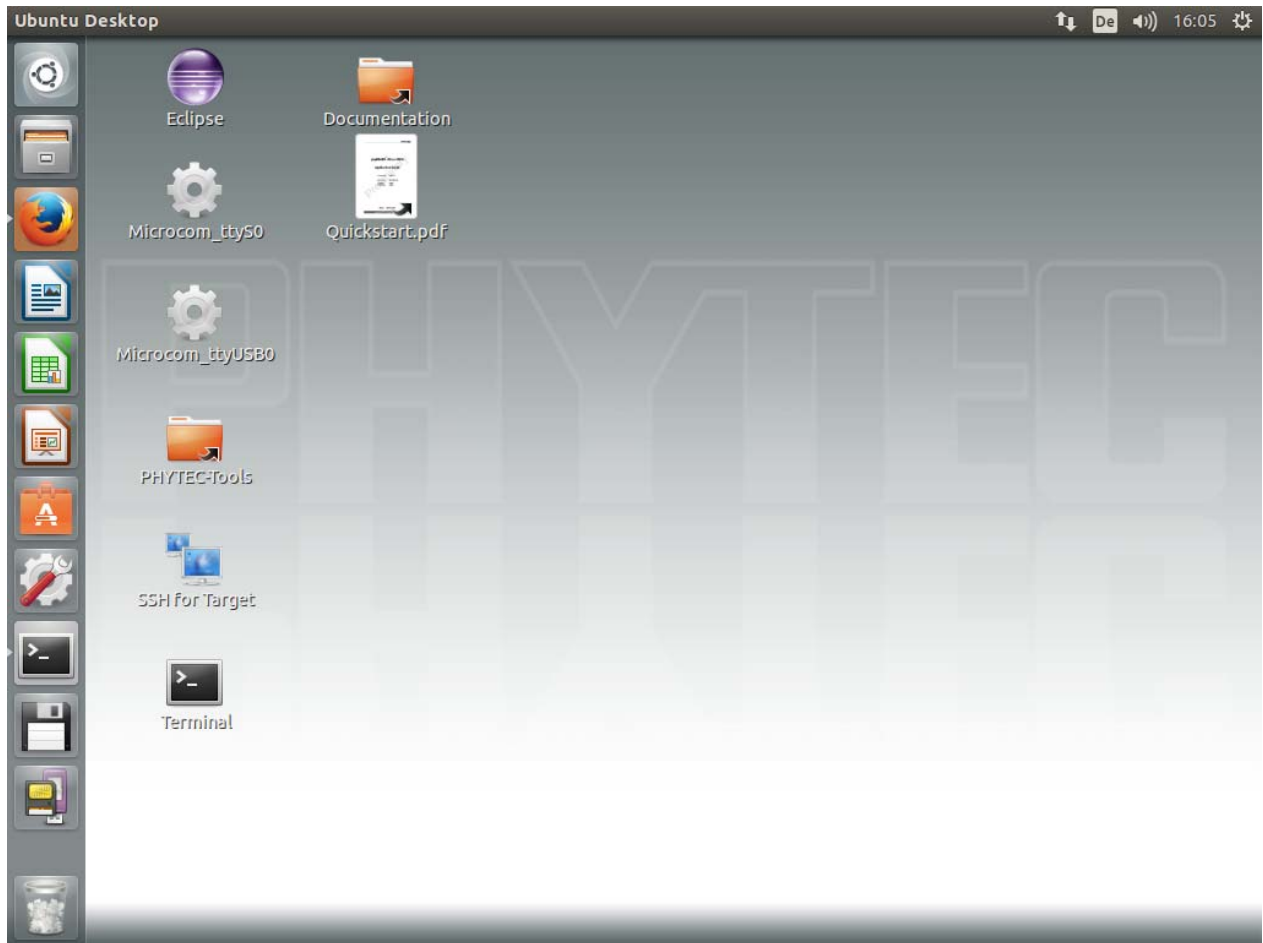
1. Plug-in the purchased USB flash drive or download the compressed VM hard disk image from <http://www.phytec.de/de/support/faq/faq-phyflex-imx6> .
2. Unpack the compressed file with an appropriate tool to get the included .vmdk file.
3. Follow the instructions of your virtualization software to create a new virtual machine. However, pay regard to the following settings required to obtain a correctly working virtual machine:
  - choose Linux-Ubuntu 64-bit version as operating system
  - use the unpacked file S0-525v9.vmdk as hard disk
  - an Ethernet adapter, as well as a serial interface, or the USB controller must be available to connect to the phyFLEX-i.MX6



In order to avoid any conflicts we recommend to have only the Ethernet adapter in the virtual machine which will be used for the peer to-peer connection between your PC and the phyFLEX-i.MX6.

4. Start the virtual machine with our hard disk image.

You will see the modified Ubuntu desktop :



For further use of the virtual machine the pre-created login data is:

**username: „phyvm“ — password: „phytec“.**



Now you are ready to connect your PC with the phyFLEX-i.MX6.

## 2.2 Starting the Hardware

In this section you will learn how to connect your host PC to the target. The connection will be done using a cross-over Ethernet cable and a serial one-to-one cable. You will start Linux from the flash memory on the target and you will be able to log in with the serial communication program *Microcom* as well as via a *SSH* session using a peer-to-peer network connection.



By default every input and output is transmitted over the serial connection.

- Connect the serial cable to the UART0 RS232 port on the target and the first serial interface on your host.



Be sure to use the one-to-one serial cable included in this Rapid Development Kit.

- Connect the cross-over Ethernet cable to the Ethernet connector on the target and to the appropriate network card of your host, which is passed on the virtual machine.



- Click the *Microcom\_ttyS0* icon on your desktop, if you have connected the target to your serial interface on your host. If you have connected the serial cable with an USB dongle to your PC click the *Microcom\_ttyUSB0* icon on your desktop.
- Connect the AC adapter to the power supply connector X12 (12V) on your board.

After connecting the board to the power supply, the target starts booting. When the target has finished loading the system, you should see a screen similar to the following:

```

Microcom_ttyUSB0
[ OK ] Started Getty on tty1.
        Starting Serial Getty on ttymxc3...
[ OK ] Started Serial Getty on ttymxc3.
[ OK ] Reached target Login Prompts.
[ OK ] Started Network Name Resolution.
[ OK ] Started Login Service.
[ OK ] Reached target Multi-User System.
        Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

Phytec
YOGURT

Yogurt (Phytec Example Distribution) iMX6-PD15.1.1 phyflex-imx6-2 ttymxc3
phyflex-imx6-2 login: 

```

- Enter **root** as the login name and press **Enter**.
- As a good example (and helpful for understanding the next step) enter **ifconfig**.

You will get information about the configuration of the Ethernet interface on the phyFLEX-i.MX6

```



Microcom_ttyUSB0
root@phyflex-imx6-2:~# ifconfig
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:31

eth0      Link encap:Ethernet  HWaddr 50:2D:F4:06:FE:0D
          inet addr:192.168.3.11  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::522d:f4ff:fe06:fe0d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4884 (4.7 KiB)  TX bytes:3524 (3.4 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:166 errors:0 dropped:0 overruns:0 frame:0
          TX packets:166 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:12712 (12.4 KiB)  TX bytes:12712 (12.4 KiB)

root@phyflex-imx6-2:~# 

```

	<p>When the target is connected to the power supply, the boot loader <i>Barebox</i> is first loaded from the flash memory. Then the boot loader is uncompressing and booting the Linux kernel from the flash. Then the kernel will install the root file system, which is also located in the target's flash.</p>
	<p>If you don't see any output in the <i>Microcom</i> window, check the serial connection between the target and your host. If you have more than one serial port try the others. By default <i>Microcom</i> uses <code>/dev/ttyS0</code>. If you want to use another port you can click on the <i>Microcom</i> Icon at your desktop with your right mouse button and select <i>Properties</i>. A window opens in which you can change the Properties of the <i>Microcom</i> Icon. Click in the <i>Command</i> field and search for <code>"-p /dev/ttyS0"</code>. Change it for example to <code>/dev/ttyS1</code> and so on if you have more than one serial port. If you are connecting the board via RS232 to USB change it to <code>/dev/ttyUSB0</code>. The <code>-p</code> Parameter defines the port which <i>Microcom</i> will use.</p> <p>If the <i>Microcom</i> window does not open, one reason could be a <i>lock</i>-file which was created when <i>Microcom</i> was not correctly terminated. Delete this file by opening a terminal and type: <b><code>sudo rm /var/lock/LCK..*</code></b></p>

In the virtual machine hard disk image the Ethernet adapter eth0 is preconfigured to the IP address `192.168.3.10` and netmask `255.255.255.0` to establish the peer-to-peer connection to the phyFLEX-i.MX6.

You are ready to test the Ethernet network connection.



- Click on the SSH for Target icon on your desktop.

A console window will open and you will see an authentication question.

- Type **yes** and press **Enter**.



If the connection is working you will see following output in the SSH connection:

```

ssh root@192.168.3.11
The authenticity of host '192.168.3.11 (192.168.3.11)' can't be established.
ECDSA key fingerprint is e4:69:36:0d:6b:8f:65:70:d6:10:07:7f:2b:a3:28:80.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.3.11' (ECDSA) to the list of known hosts.
Last login: Wed Sep 16 14:01:28 2015

PHYTEC
YOGURT

Yogurt (Phytec Example Distribution) iMX6-PD15,1.1 %h
root@phyflex-imx6-2:~#

```



If you don't see the user login, check the Ethernet connection between the target and the host. If you have installed more than one network card on your host, be sure to connect the cable to the network card which you have configured with the IP address **192.168.3.10**.  
If you do not see the login, you may not have set up the right IP address for your host. Open the **System Settings**, click on the **Network** icon and check the settings for the physical interface to which you have connected the Ethernet cable.



You have successfully set up all configurations to access your phyFLEX-i.MX6 from your host.

## 3 Working with Eclipse

Now we start developing our own applications with the help of *Eclipse*.

### 3.1 Programming in the C/C++ Perspective

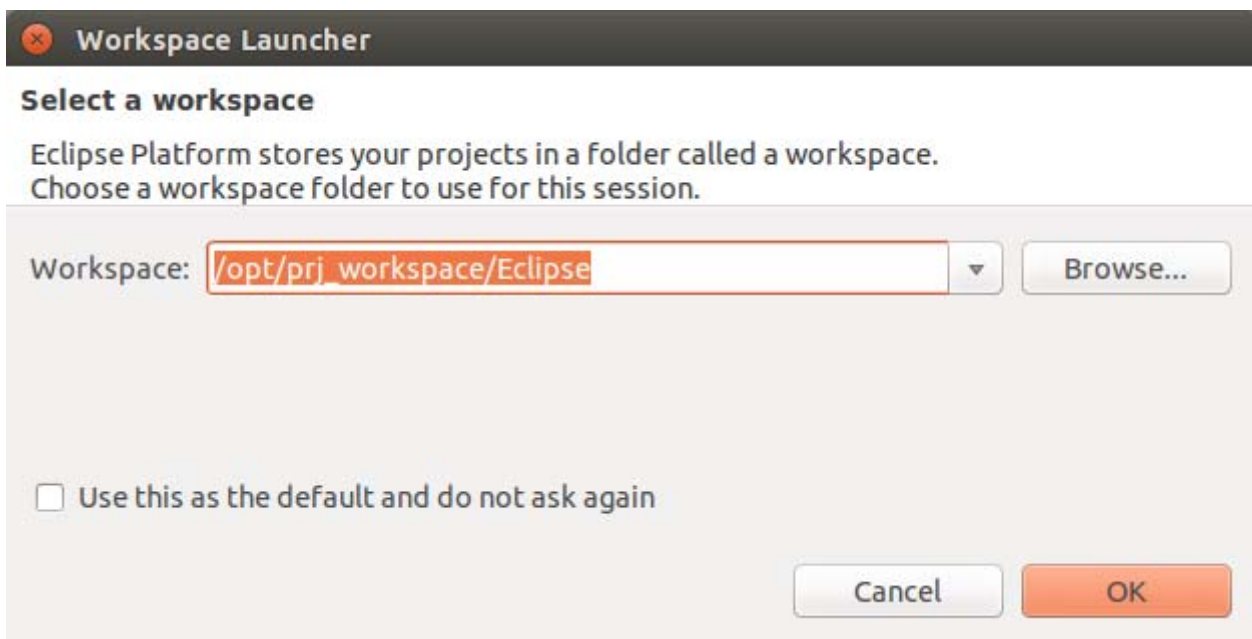
We are starting with the C/C++ workbench. Therefore you will import an existing Eclipse project into your workspace. The imported example project will be compiled with the cross compiler. After that, you will copy and execute the newly created program on the target.

#### 3.1.1 Work with the Demo Project

- Click the **Eclipse icon** to start the application. You can find this icon on your desktop.

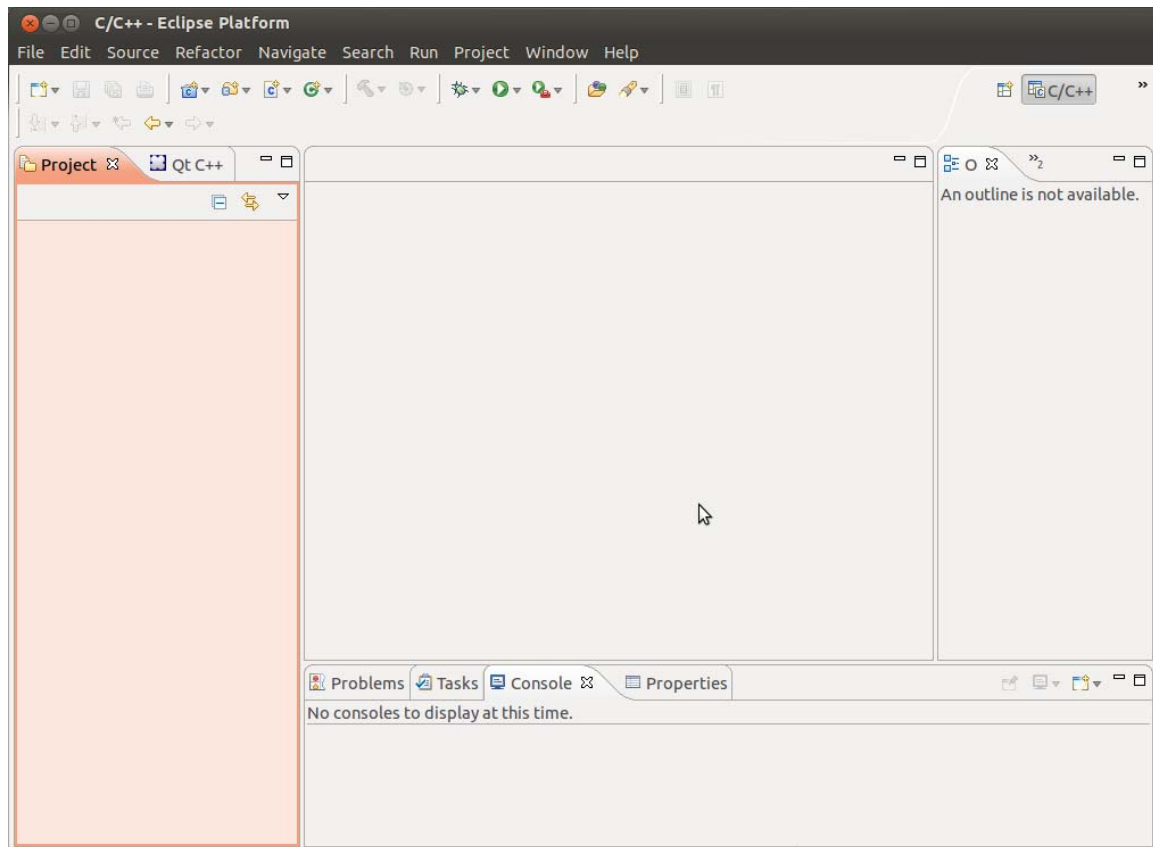


- Change the workspace directory to `/opt/prj_workspace/Eclipse` if necessary and confirm with **OK**.



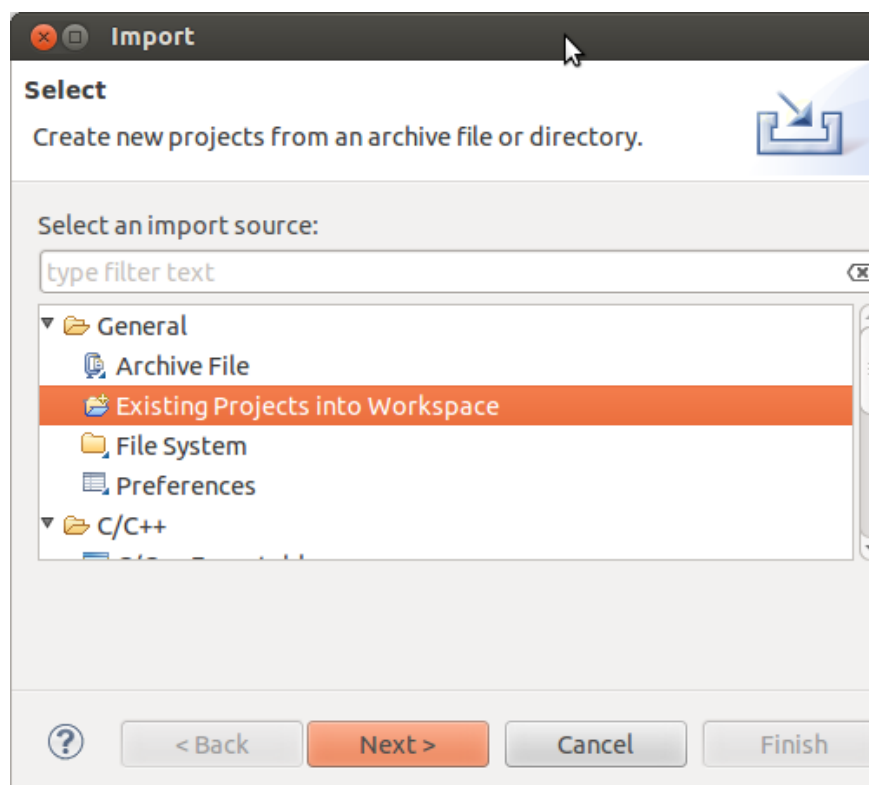


Now you can see the *Eclipse* workbench

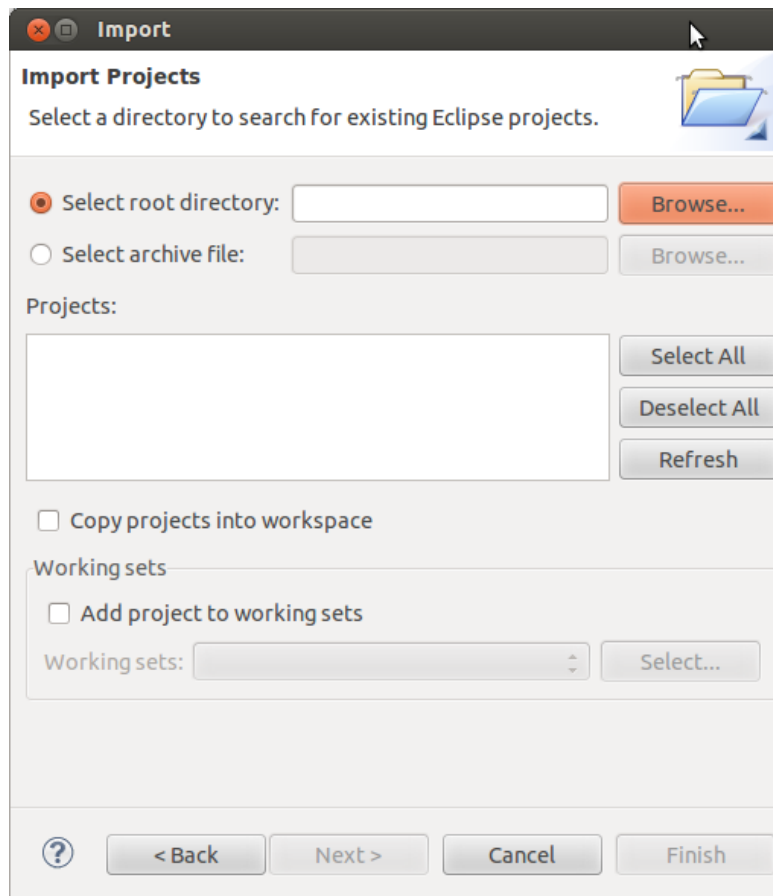


First we will import an existing project.

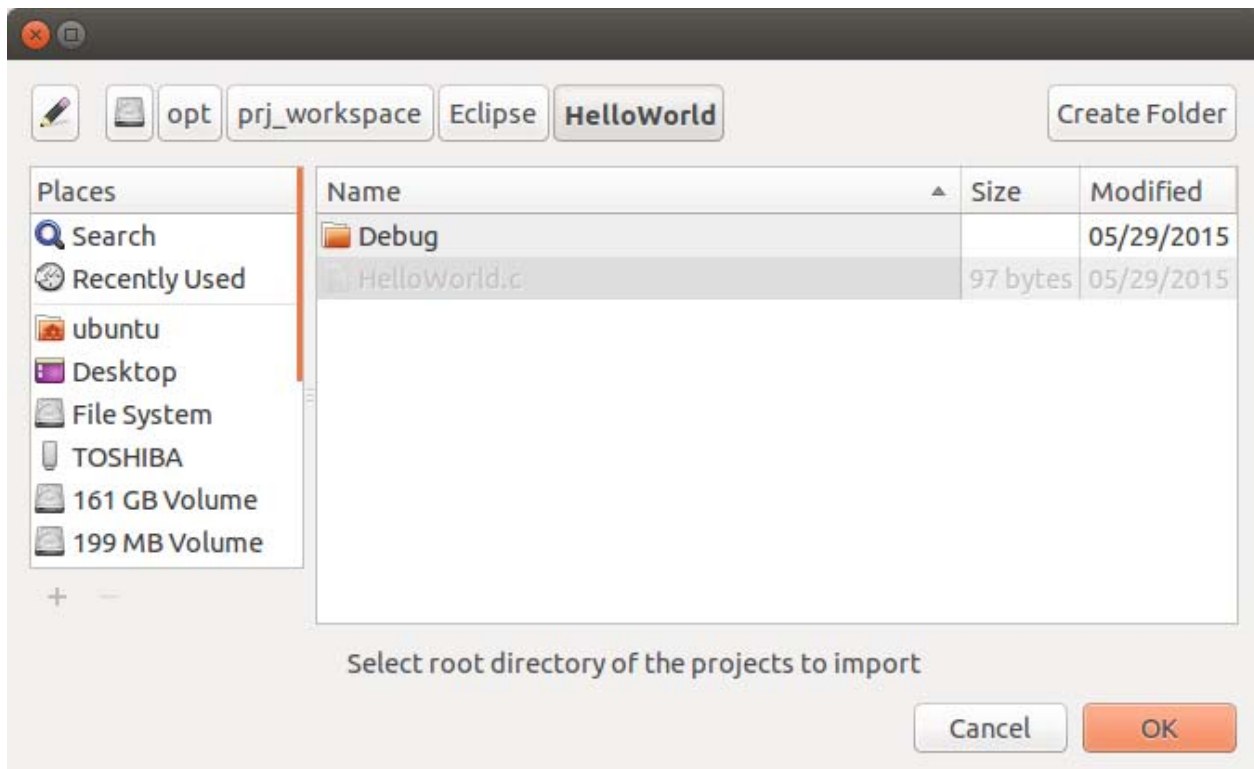
- Select **File ► Import** from the menu bar.
- Select **Existing Projects into Workspace** and click **Next**.



- Select **Browse**.

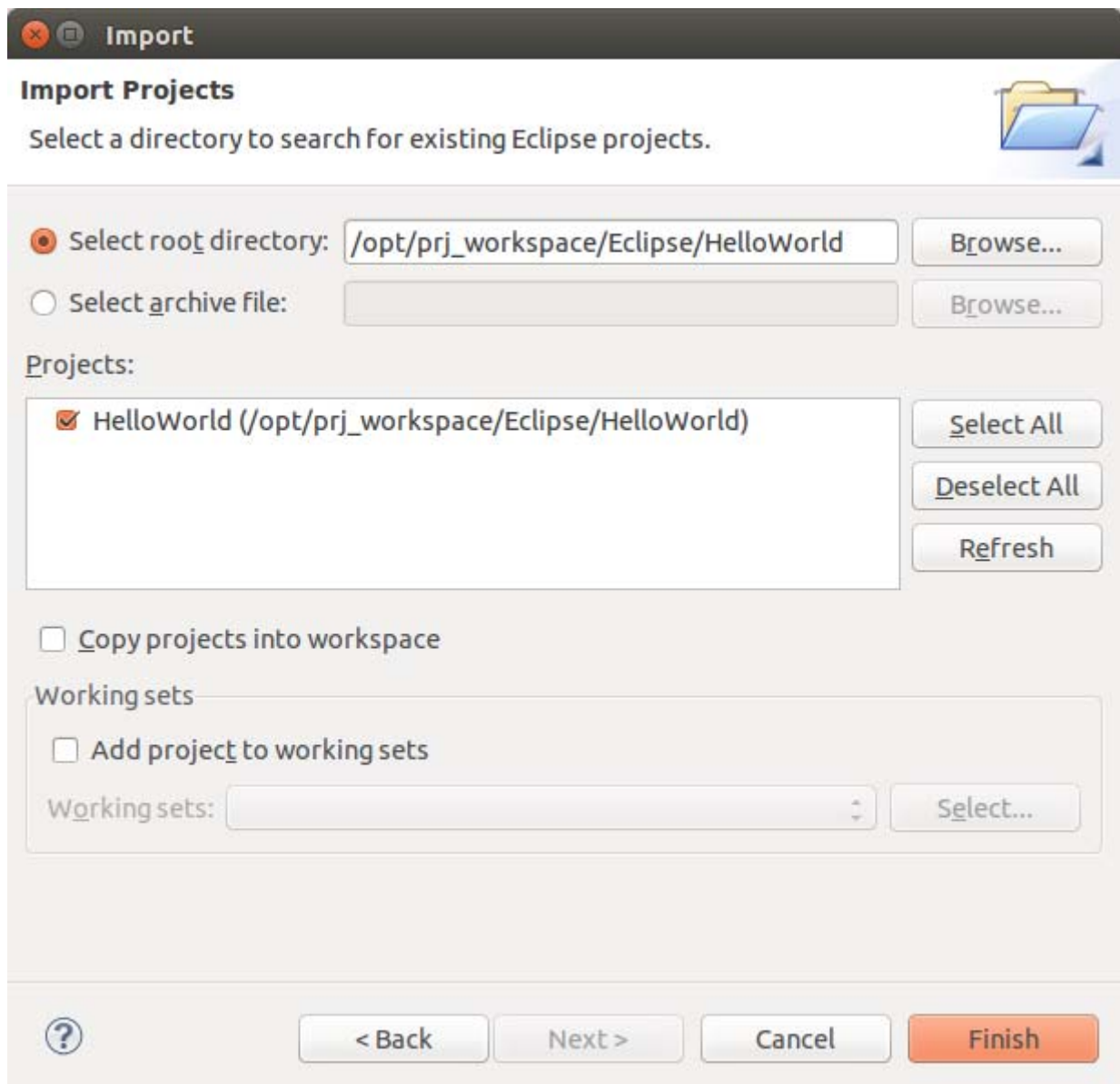


- Double-click the **HelloWorld** directory under `/opt/prj_workspace/Eclipse/`.



- Click **OK**.

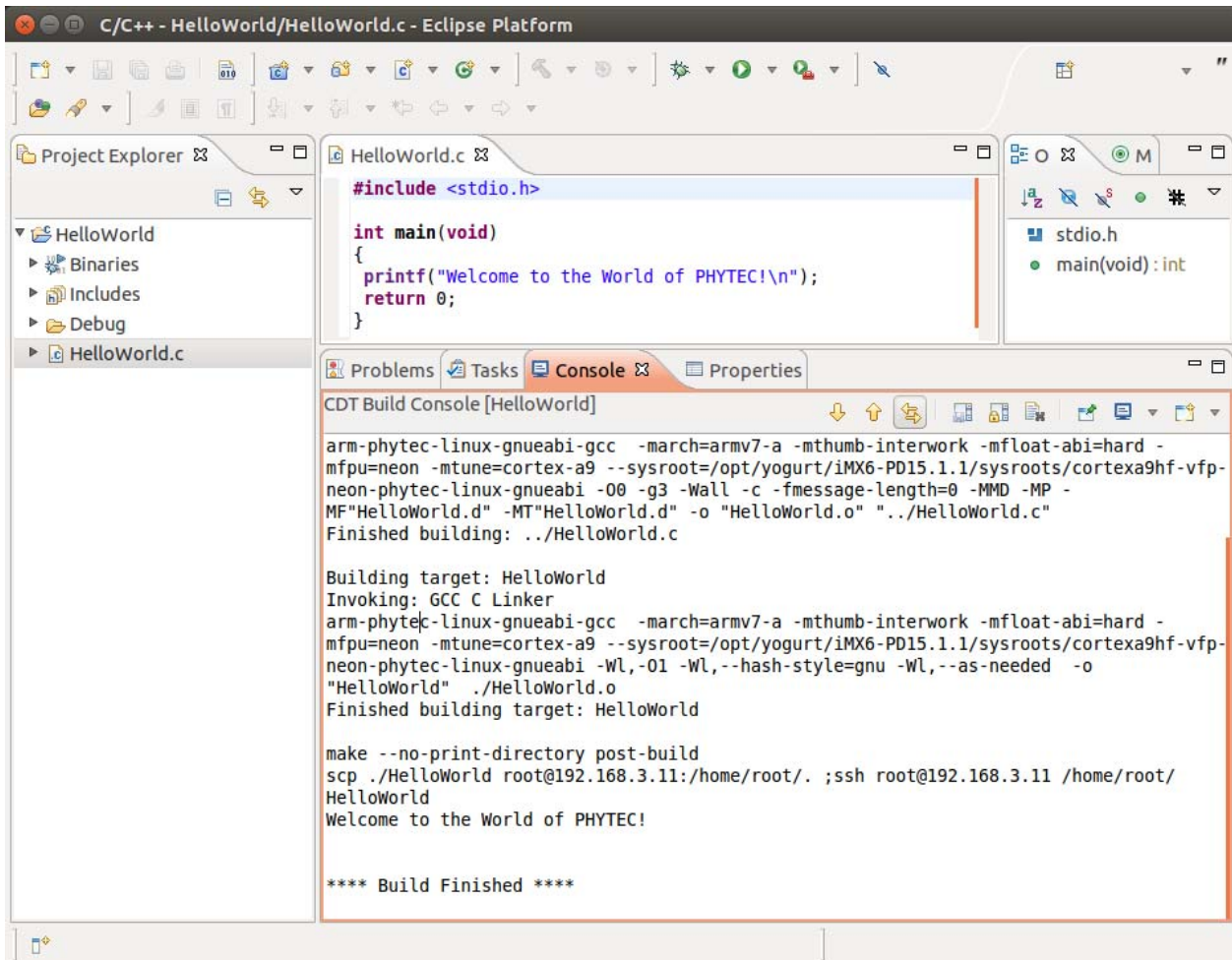
- Select **Finish** to import the project.



- Select **Project ► Build Project** from the menu bar.

The *HelloWorld* program will be compiled and the *HelloWorld* executable is built for the target. Then the *HelloWorld* file is copied to the target using *secure copy*. After the file has been copied to the target, the program is executed on the target using *SSH*.

You will see the following content in the *Console* window:



The screenshot shows the Eclipse IDE interface. The Project Explorer on the left shows a project named 'HelloWorld' with sub-items 'Binaries', 'Includes', 'Debug', and 'HelloWorld.c'. The main editor displays the 'HelloWorld.c' file with the following code:

```
#include <stdio.h>

int main(void)
{
    printf("Welcome to the World of PHYTEC!\n");
    return 0;
}
```

The CDT Build Console at the bottom shows the following output:

```
arm-phytec-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -
mfpu=neon -mtune=cortex-a9 --sysroot=/opt/yogurt/iMX6-PD15.1.1/sysroots/cortexa9hf-vfp-
neon-phytec-linux-gnueabi -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -
MF"HelloWorld.d" -MT"HelloWorld.d" -o "HelloWorld.o" "../HelloWorld.c"
Finished building: ../HelloWorld.c

Building target: HelloWorld
Invoking: GCC C Linker
arm-phytec-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -
mfpu=neon -mtune=cortex-a9 --sysroot=/opt/yogurt/iMX6-PD15.1.1/sysroots/cortexa9hf-vfp-
neon-phytec-linux-gnueabi -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed -o
"HelloWorld" ../HelloWorld.o
Finished building target: HelloWorld

make --no-print-directory post-build
scp ../HelloWorld root@192.168.3.11:/home/root/. ;ssh root@192.168.3.11 /home/root/
HelloWorld
Welcome to the World of PHYTEC!

**** Build Finished ****
```



If you do not get this result verify that you have the target connected to your host, and that the network has been configured as explained in this Quickstart.



You have successfully passed the first steps with the *Eclipse* IDE. You are now able to import existing projects into the *Eclipse* workspace. You can compile an existing project and execute the program on the target.

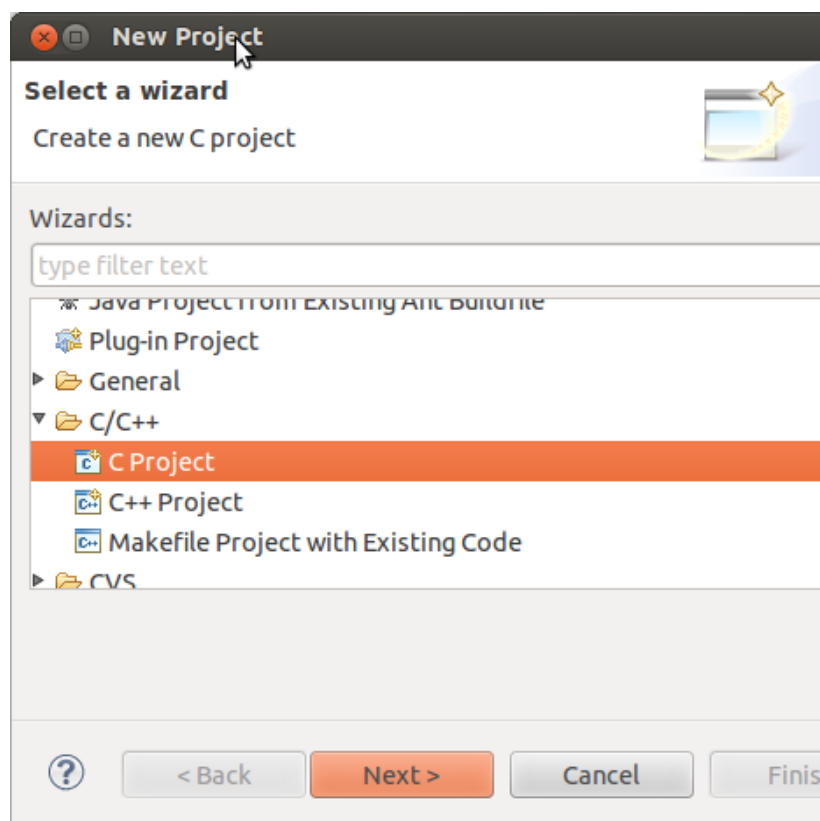
### 3.1.2 Creating a New Project

In this section you will learn how to create a new project with *Eclipse* and how to configure the project for use with the *GNU C/C++* cross development toolchain.

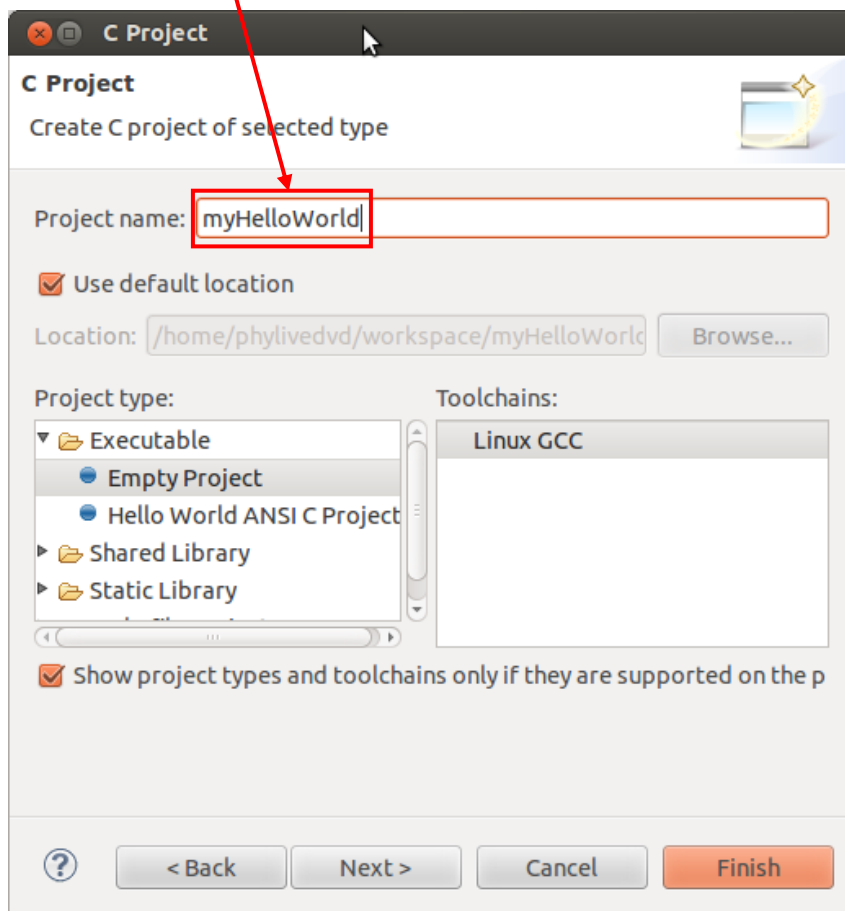
- Open *Eclipse* if it is not already opened.
- Select **File ► New ► Project** from the menu bar.

A new dialog opens.

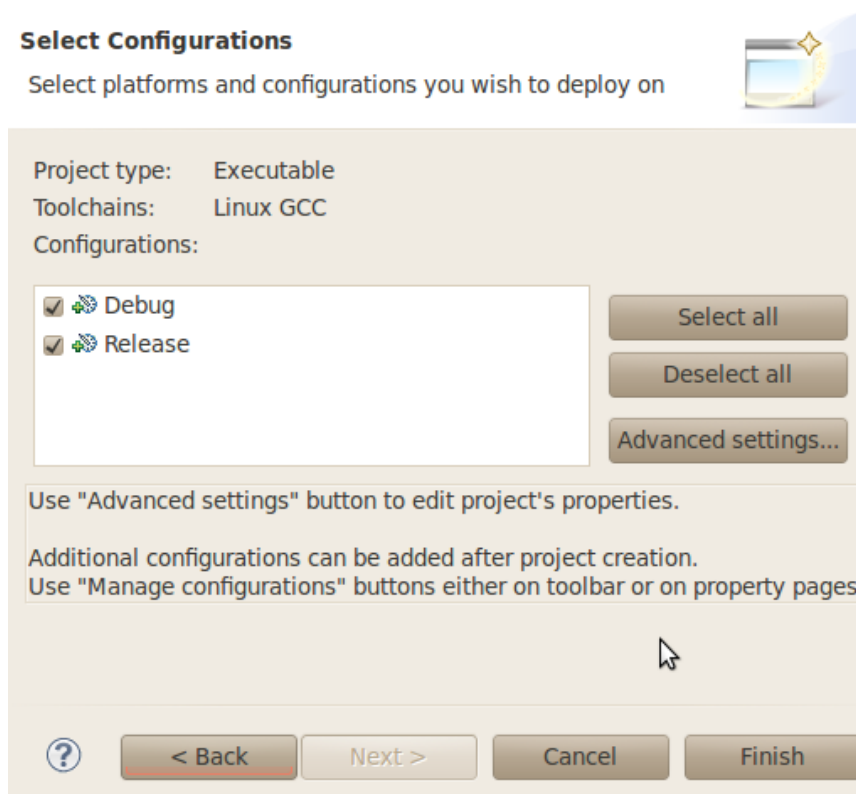
- Select **C Project** and click **Next**.



- Enter the project name **myHelloWorld** and click **Next**.

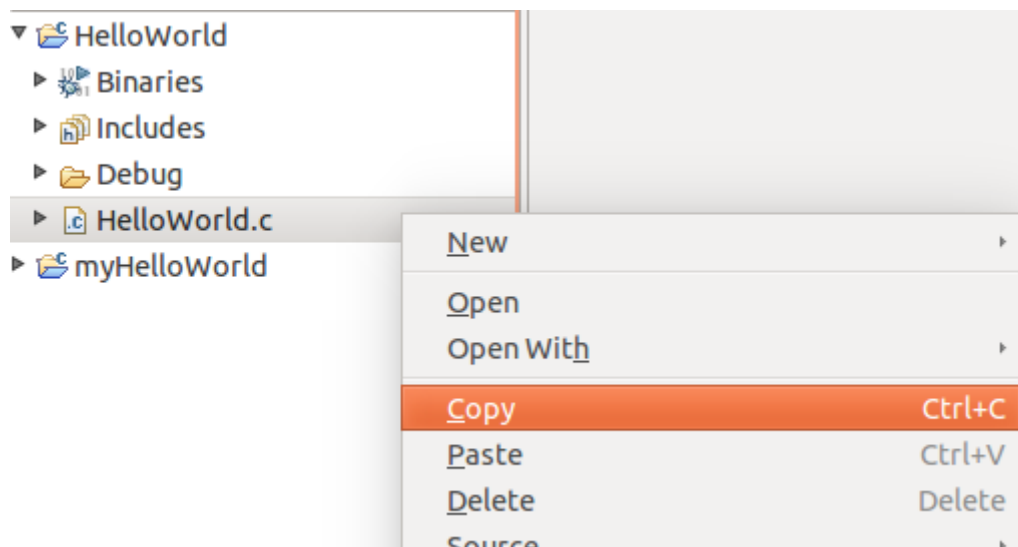


- Click **Finish**.



You will see the C/C++ IDE with the *myHelloWorld* project.

- If the HelloWorld Project is not expanded double-click the **HelloWorld** project which we have worked with previously.
- Right-click on **HelloWorld.c** in the *HelloWorld* project.
- Select **Copy**.



- Select the **myHelloWorld** project.
- Right-click the **myHelloWorld** project.
- Select **Paste**.
- Double-click on **HelloWorld.c** in the *myHelloWorld* project.

If *Build Automatically* from the *Project* menu is selected, the *HelloWorld* application will now be compiled and created with the standard *GCC C/C++* compiler suitable for your host machine. You will find the executable file, which can only run on your host system, in the *workspace/myHelloWorld/Debug* directory.

To compile your project for phyFLEX-i.MX6 instead, you will have to use the *GNU C/C++* cross compiler.

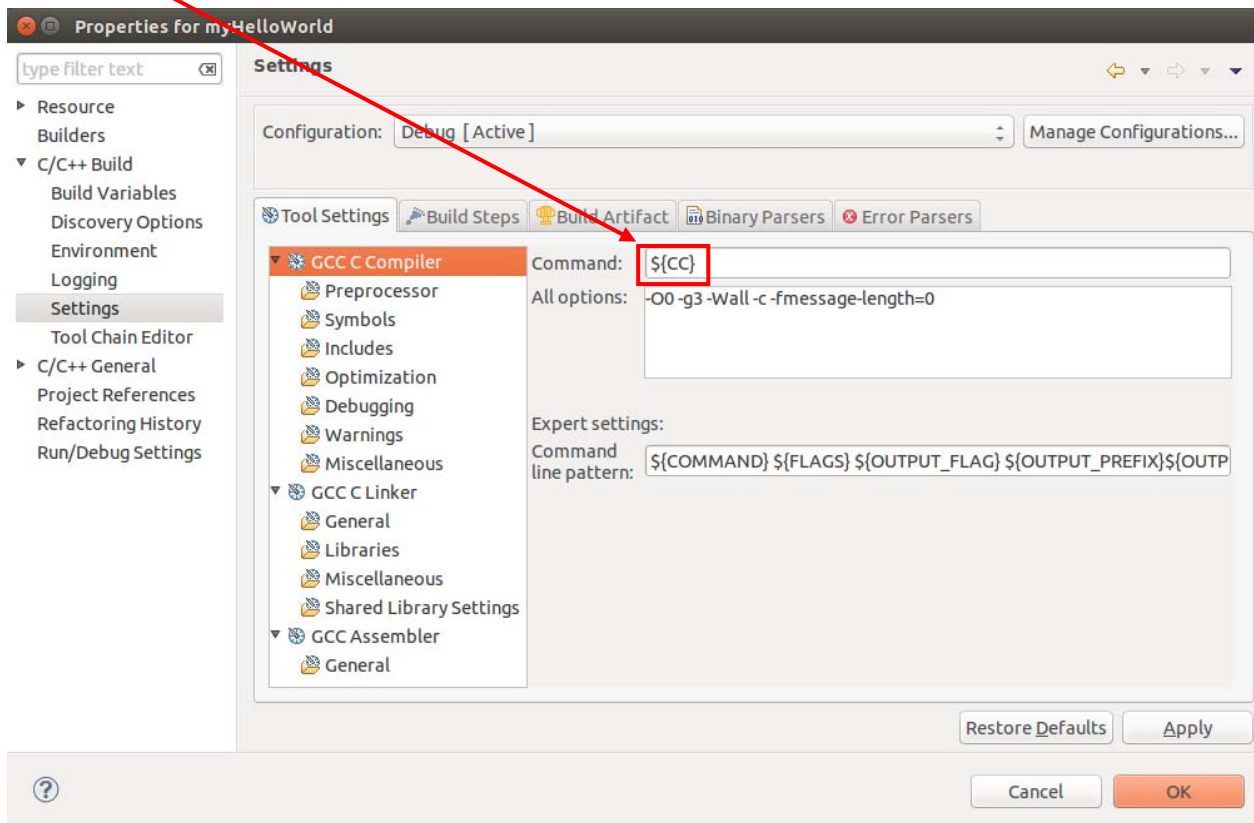
- Right-click the **myHelloWorld** project and choose **Properties**.

The *Properties* dialog appears.

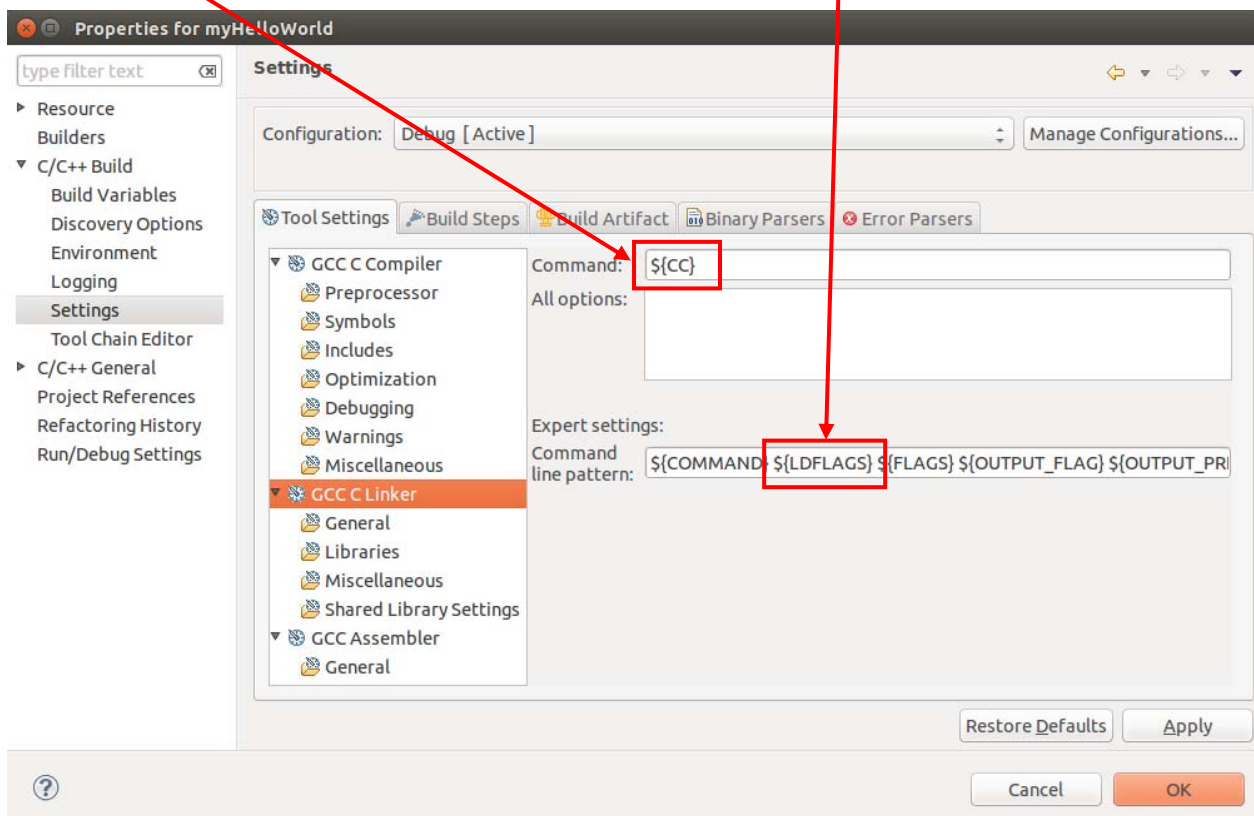
- Select **C/C++ Build ► Settings**.
- Select **GCC C Compiler**.



- Enter `${CC}` into the *Command* input field.

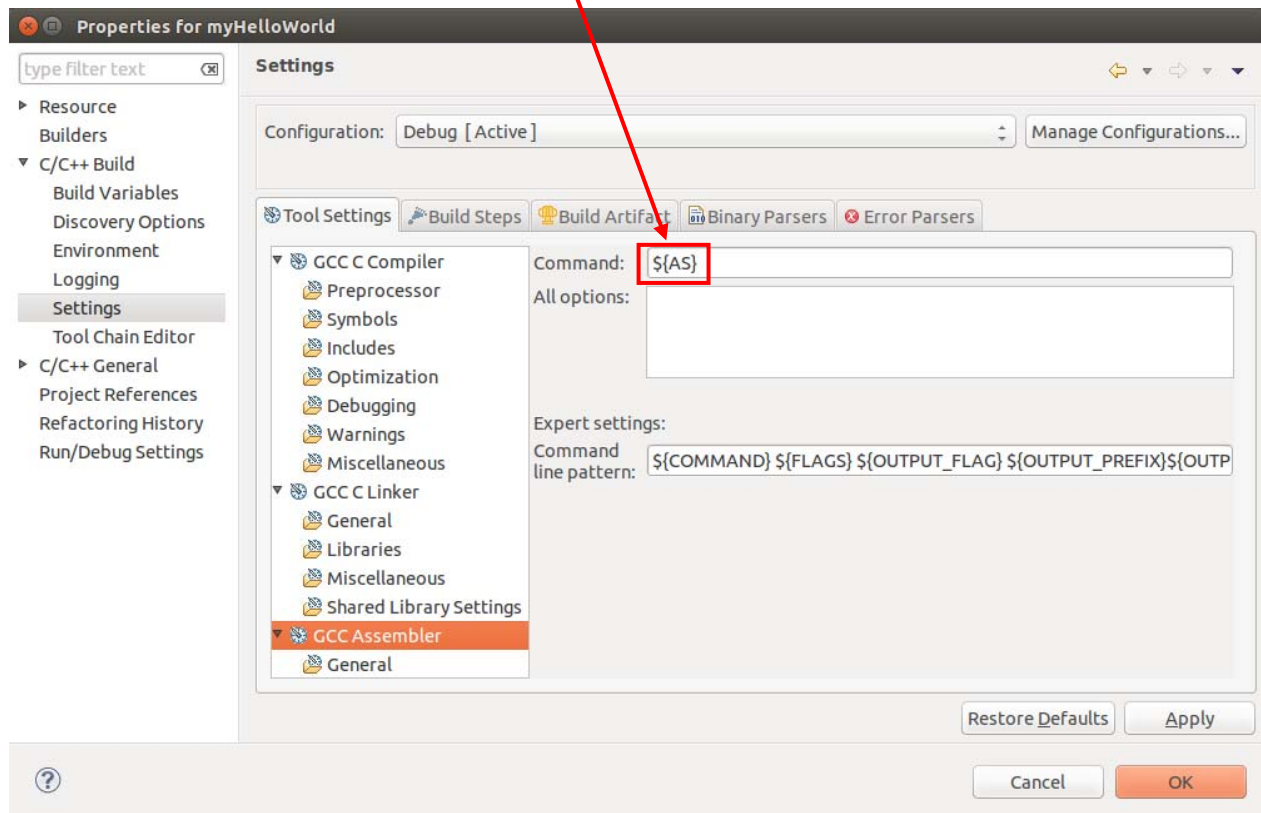


- Select ***GCC C Linker***.
- Enter `${CC}` into the *Command* input field and add `${LDLAGS}` in the *Command line pattern* after `${COMMAND}`.





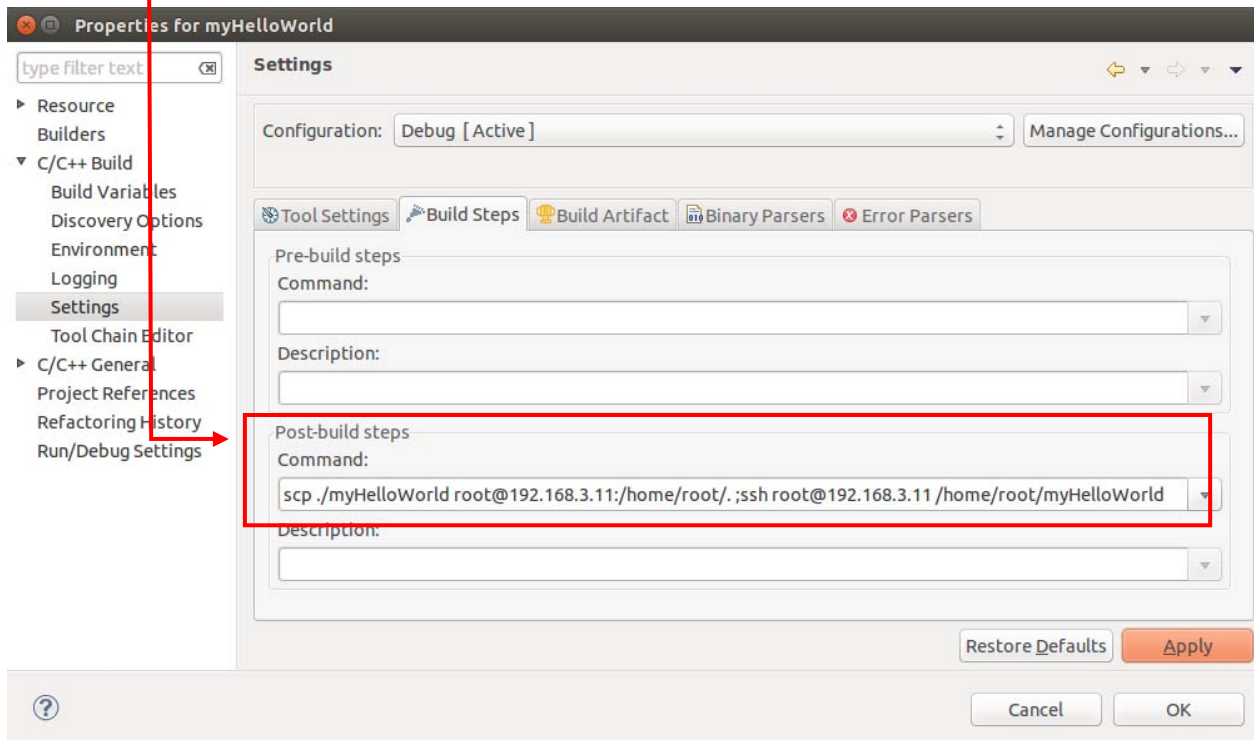
- Select ***GCC Assembler***.
- Change the *Command* input field to `${AS}` .



- Click ***Apply***.
- Select the *Build Steps* tab.

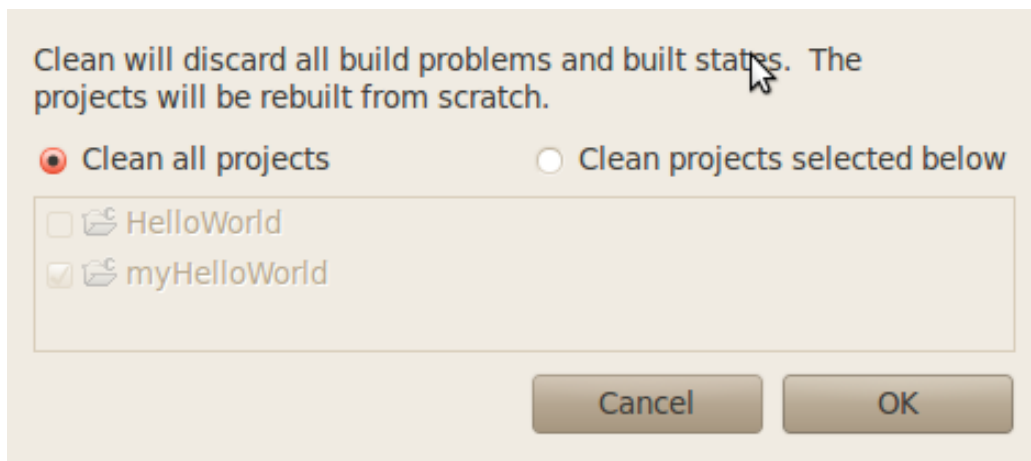
- Enter the following command in the *Post-build steps Command* input field:  

```
scp ./myHelloWorld root@192.168.3.11:/home/root/. ,ssh  
root@192.168.3.11 . /home/root/myHelloWorld
```



Be sure to enter the **semicolon** before the ssh command.  
 Ensure that the file *myHelloWorld* on the target will have execution rights, because otherwise ssh will fail.

- Click **Apply**.
- Click **OK**.
- Select **Project ► Clean** from the menu bar.



- Confirm with **OK**.

The project will be rebuilt.

- Select the *Console* tab.

If no errors occur while building the project, you will see the following output:

The screenshot shows the Eclipse IDE interface. The Project Explorer on the left shows a project named 'myHelloWorld' with sub-entries for 'Binaries', 'Debug', and 'HelloWorld.c'. The main editor displays the source code for 'HelloWorld.c', which includes `<stdio.h>` and a `main` function that prints 'Welcome to the World of PHYTEC!'. The Console tab at the bottom shows the build output for 'myHelloWorld', including the GCC compiler command, the linker command, and the final output 'Welcome to the World of PHYTEC!'. The console also shows the command to copy the binary to a remote server and the message '\*\*\*\* Build Finished \*\*\*\*'.

```

C/C++ - HelloWorld/HelloWorld.c - Eclipse Platform

Project Explorer
  HelloWorld
    Binaries
    Includes
    Debug
    HelloWorld.c
  myHelloWorld
    Binaries
    Debug
    HelloWorld.c

HelloWorld.c
#include <stdio.h>

int main(void)
{
    printf("Welcome to the World of PHYTEC!\n");
    return 0;
}

Problems Tasks Console Properties
CDT Build Console [myHelloWorld]

arm-phytec-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -
mfpu=neon -mtune=cortex-a9 --sysroot=/opt/yogurt/IMX6-PD15.1.1/sysroots/cortexa9hf-vfp-
neon-phytec-linux-gnueabi -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -
MF"HelloWorld.d" -MT"HelloWorld.d" -o "HelloWorld.o" "../HelloWorld.c"
Finished building: ../HelloWorld.c

Building target: myHelloWorld
Invoking: GCC C Linker
arm-phytec-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -
mfpu=neon -mtune=cortex-a9 --sysroot=/opt/yogurt/IMX6-PD15.1.1/sysroots/cortexa9hf-vfp-
neon-phytec-linux-gnueabi -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed -o
"myHelloWorld" ../HelloWorld.o
Finished building target: myHelloWorld

make --no-print-directory post-build
scp ./myHelloWorld root@192.168.3.11:/home/root/. ;ssh root@192.168.3.11 /home/root/
myHelloWorld
Welcome to the World of PHYTEC!

**** Build Finished ****
  
```



You have successfully created your first own project with the *Eclipse* IDE. You have configured the project to create an application for your target platform.

### 3.1.3 Modifying the Demo Application

Now we will extend the *myHelloWorld* application. The extended *myHelloWorld* application will write an output to the first serial interface as well as to the standard output.

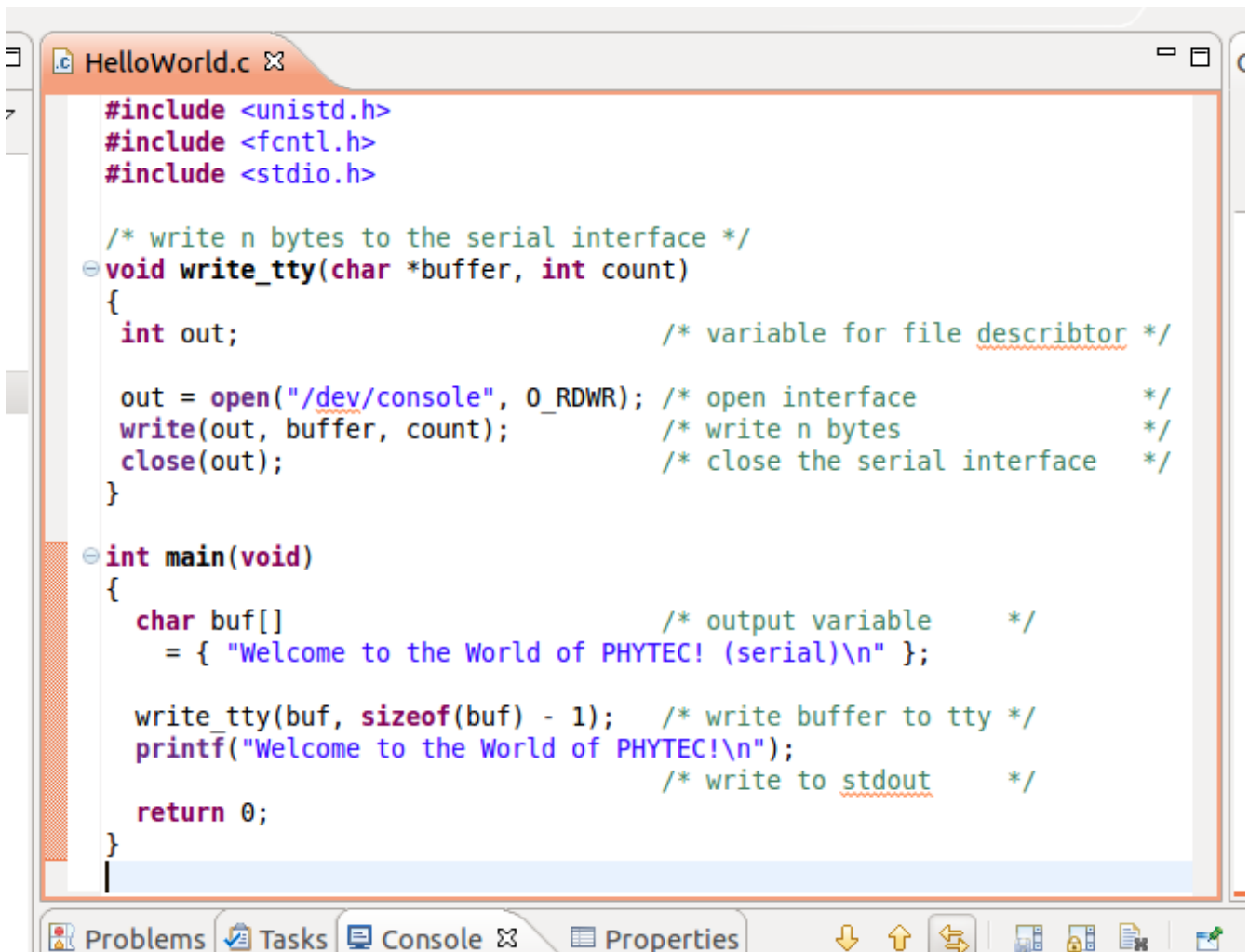
- Open *Eclipse* if it is not opened yet.
- Double-click **HelloWorld.c** in the *myHelloWorld* project.
- First include the following two additional header files:  

```
#include <unistd.h>
#include <fcntl.h>
```
- Then add the function *write\_tty()*, which writes *n* bytes to the first serial interface (which, on the phyFLEX-i.MX6, is connected to the system console */dev/console*):  

```
void write_tty (char *buffer, int count)
{
    int out,
    out = open ("/dev/console", O_RDWR),
    write(out, buffer, count),
    close(out),
}
```
- Enter the following two lines in the *main()* function to declare the buffer and call the *write\_tty()* function:  

```
char buf [] = { "Welcome to the World of PHYTEC! (serial)\n" },
write_tty(buf, sizeof (buf) - 1),
```

In the next screenshot you can see the complete program.



```
.c HelloWorld.c
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

/* write n bytes to the serial interface */
void write_tty(char *buffer, int count)
{
    int out; /* variable for file descriptor */

    out = open("/dev/console", O_RDWR); /* open interface */
    write(out, buffer, count); /* write n bytes */
    close(out); /* close the serial interface */
}

int main(void)
{
    char buf[] /* output variable */
        = { "Welcome to the World of PHYTEC! (serial)\n" };

    write_tty(buf, sizeof(buf) - 1); /* write buffer to tty */
    printf("Welcome to the World of PHYTEC!\n");
    /* write to stdout */

    return 0;
}
```

Problems Tasks Console Properties

- Save your program after changing the code.

The application will be compiled, built, copied to the target and executed.

- Click the **Microcom icon** on the desktop.



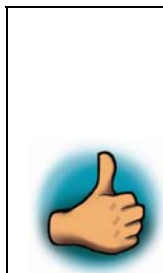
- If you are not logged in, enter **root** and press **Enter**.
- Type **./myHelloWorld** to start the application.
- You will see the following output:  

```
Welcome to the World of PHYTEC! (serial)
Welcome to the World of PHYTEC!
```
- Close *Microcom*.

When you start the application via an *SSH* session, you only see one output line. When you execute the program with *Microcom*, you see two output lines.



The first line is a direct output on the serial interface. You can not see this line in an *SSH* session, because you are connected over a TCP/IP connection to the target. With *Microcom*, however, you have direct access to the serial interface, so you can also see the line that is written to the serial console.



In this section you have changed an existing application. You also learned how to access the serial interface. First you called the function *open()* on the device */dev/console*. The return value of this function was a file descriptor. With the file descriptor you called the function *write()* to send *n* bytes to the device */dev/console*. After that, the file descriptor was closed with the function *close()*.

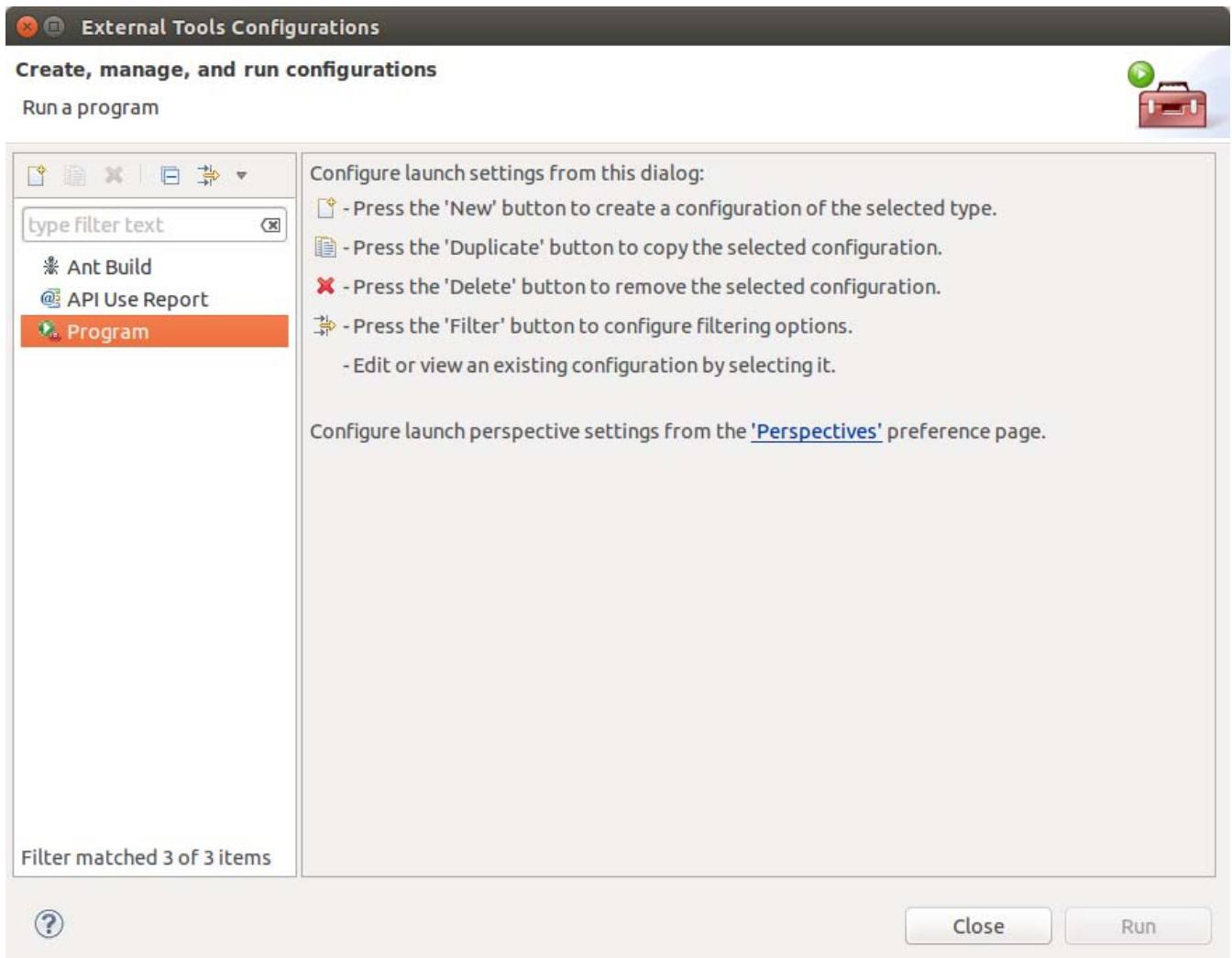
This procedure is quite typical for Linux, because Linux treats everything as a file.

### 3.1.4 Starting a Program out of Eclipse on the Target

In the following you will find another method to start an application out of Eclipse.

After compiling a project in *Eclipse*, the program is copied to the target and directly executed. A program can also be executed on the target without compiling a project. In the following section you will learn how to start a program on the target as an external tool.

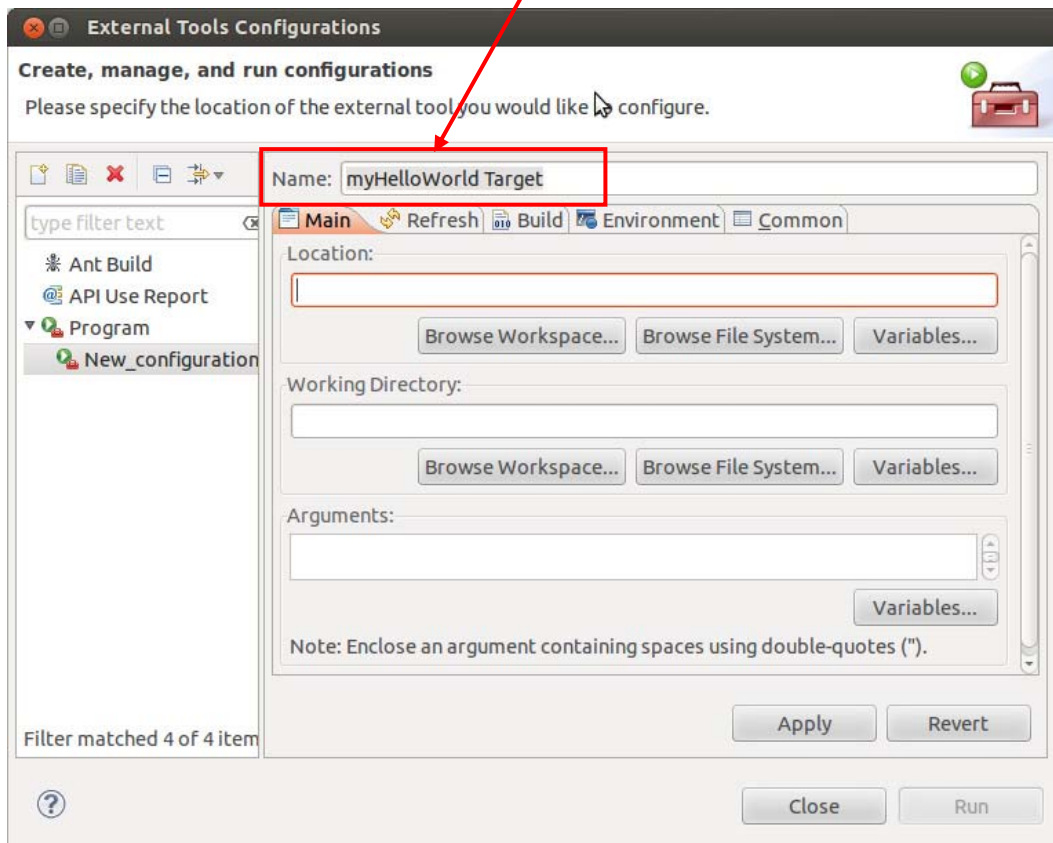
- Select **Run ► External Tools ► External Tools Configurations** from the menu bar.



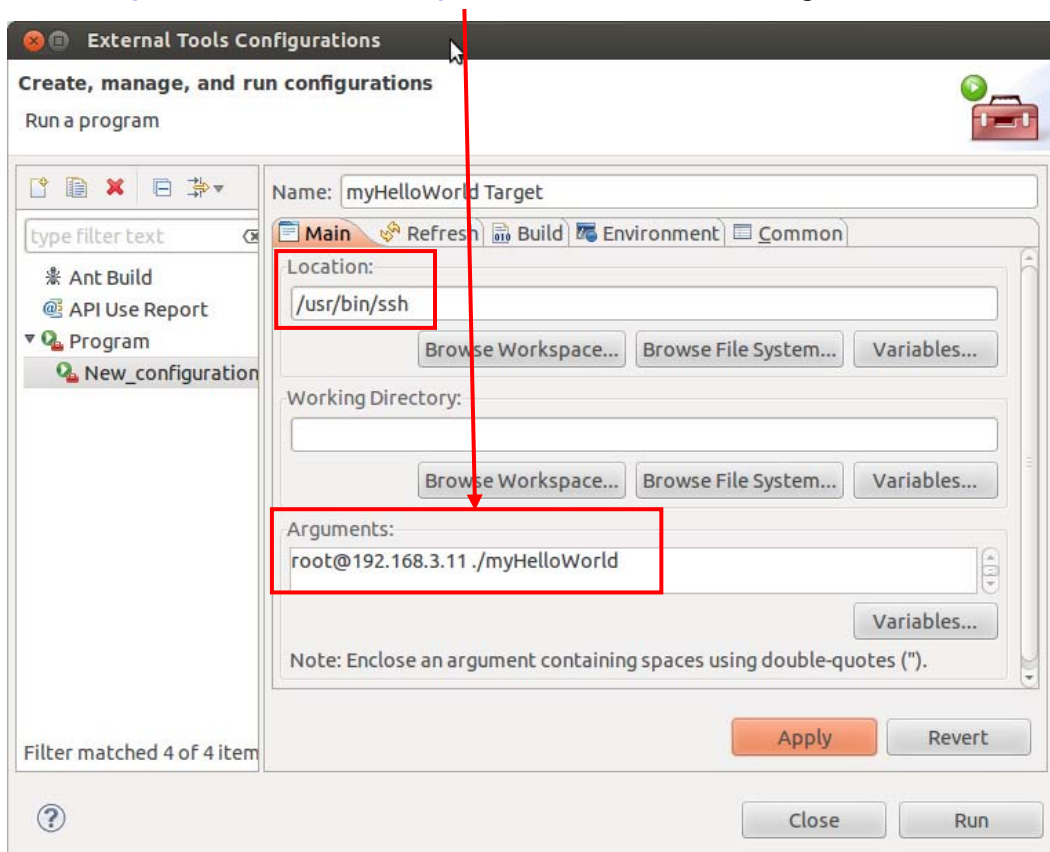
- Double-Click **Program** a new program configuration will be opened.



- In the *Name* input field, enter: **myHelloWorld Target**.



- Enter **/usr/bin/ssh** in the *Location* input field.
- Enter **root@192.168.3.11 ./myHelloWorld** into the *Arguments* field.



- Select **Apply**.



- Select **Run**.

Now the program is executed on the target and you will see the output **Welcome to the World of PHYTEC! (serial)** in the *Microcom* window.

If you want to execute the program the next time, you can use the *Run External Programs* button from the menu bar.



You have successfully created your own *Eclipse* project and you learned how to execute a program on the target.

### 3.2 Debugging an Example Project

In this chapter you will learn using the *GNU debugger GDB* on the host for remote debugging in conjunction with the *GDB* server on the target. *GDB* is the symbolic debugger of the *GNU* project and is arguably the most important debugging tool for any Linux system.

First you will start the *GDB* server on the target. Then you will configure the *Eclipse* platform and start the *GNU* debugger out of *Eclipse* using the *Debug* view.

The *CDT* extends the standard *Eclipse Debug* view with functions for debugging C/C++ code. The *Debug* view allows you to manage the debugging and running of a program in the workbench. Using the *Debug* view you will be able to set breakpoints/watchpoints in the code and trace variables and registers. The *Debug* view displays the stack frame for the threads of each target you are debugging. Each thread in your program appears as a node in the tree, and the *Debug* view displays the process for each target you are running.

The *GDB* client is running on the host and is used to control the *GDB* server on the target, which in turn controls the application running on the target. *GDB* client and *GDB* server can communicate over a TCP/IP network connection as well as via a serial interface. In this Application Guide we will only describe debugging via TCP/IP.

### 3.2.1 Starting the GDB Server on the Target

In this passage you will learn how to start the *GDB* server on the target. The *GDB* server will be used to start and control the *myHelloWorld* program.

To debug a program with *GDB*, the program needs extended debugging symbols. These have already been added while building the program.



- Open *Microcom*.
- Type `root` and press **Enter**.
- Start the *GDB* server:  
`gdbserver 192.168.3.11:10000 myHelloWorld`

You have started the *GDB* server on the target. The *GDB* server is now waiting for connections on TCP port 10000.

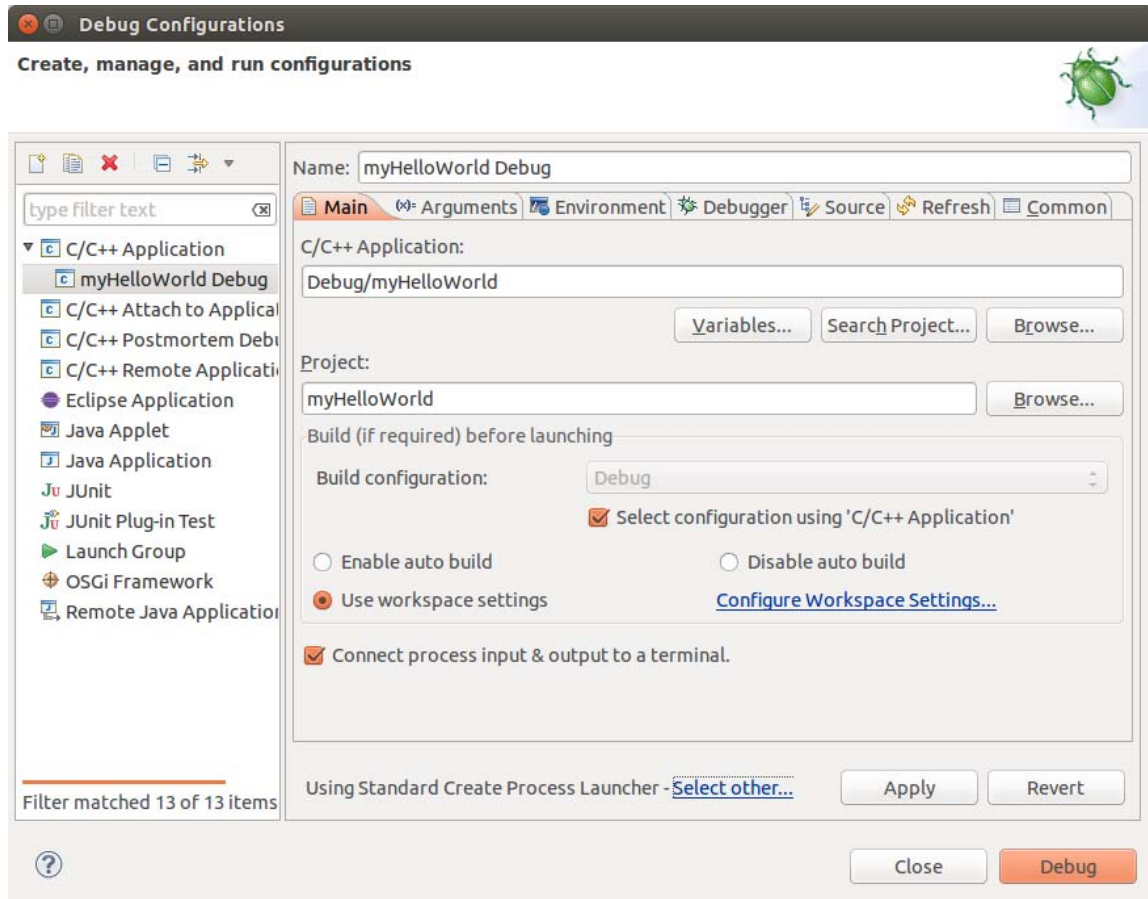
### 3.2.2 Configuring and Starting the Debugger in Eclipse

In this passage you will learn how to configure your project settings to use *Eclipse* with the *GNU* debugger. After the configuration of your project settings, the *GNU* debugger will start and connect to the *GDB* server on the target.

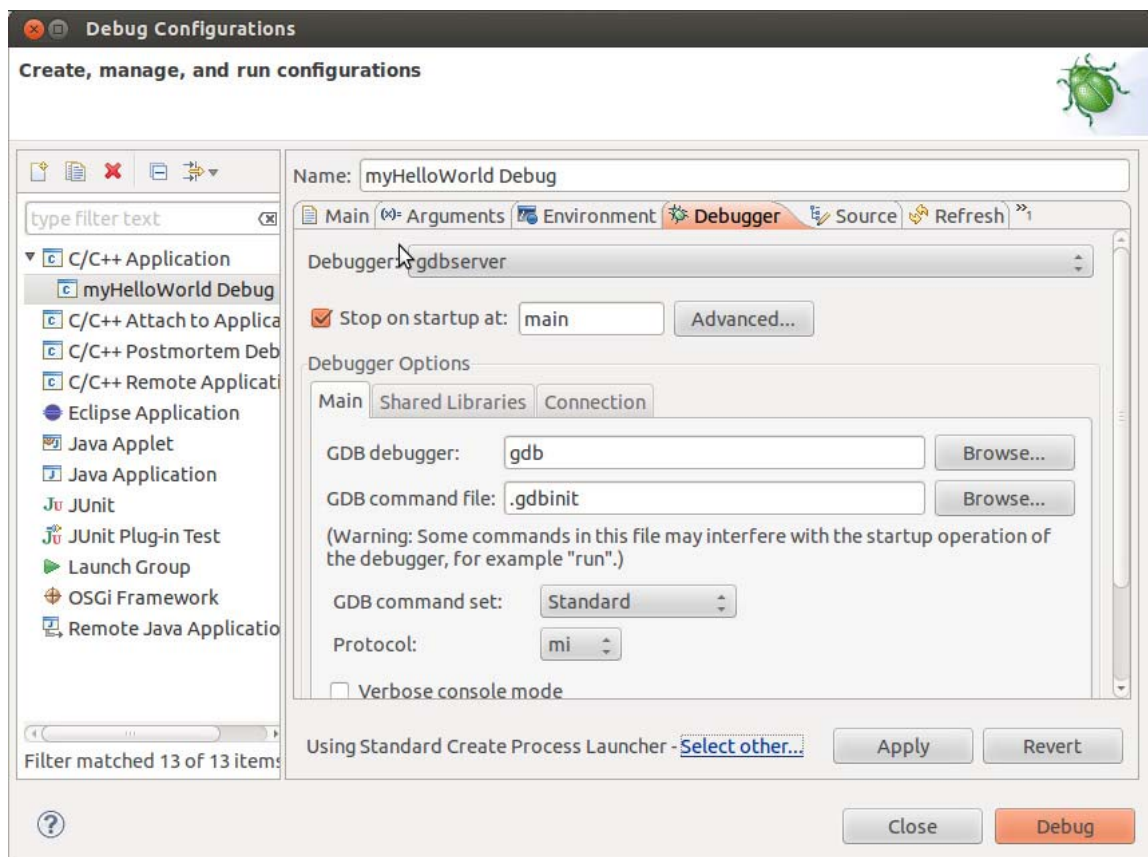
- Start *Eclipse* if the application is not started yet.
- Right-click on the *myHelloWorld* project in the *Navigator* window.
- Select **Debug As ► Debug Configurations**.

A dialog to create, manage and run applications appears.

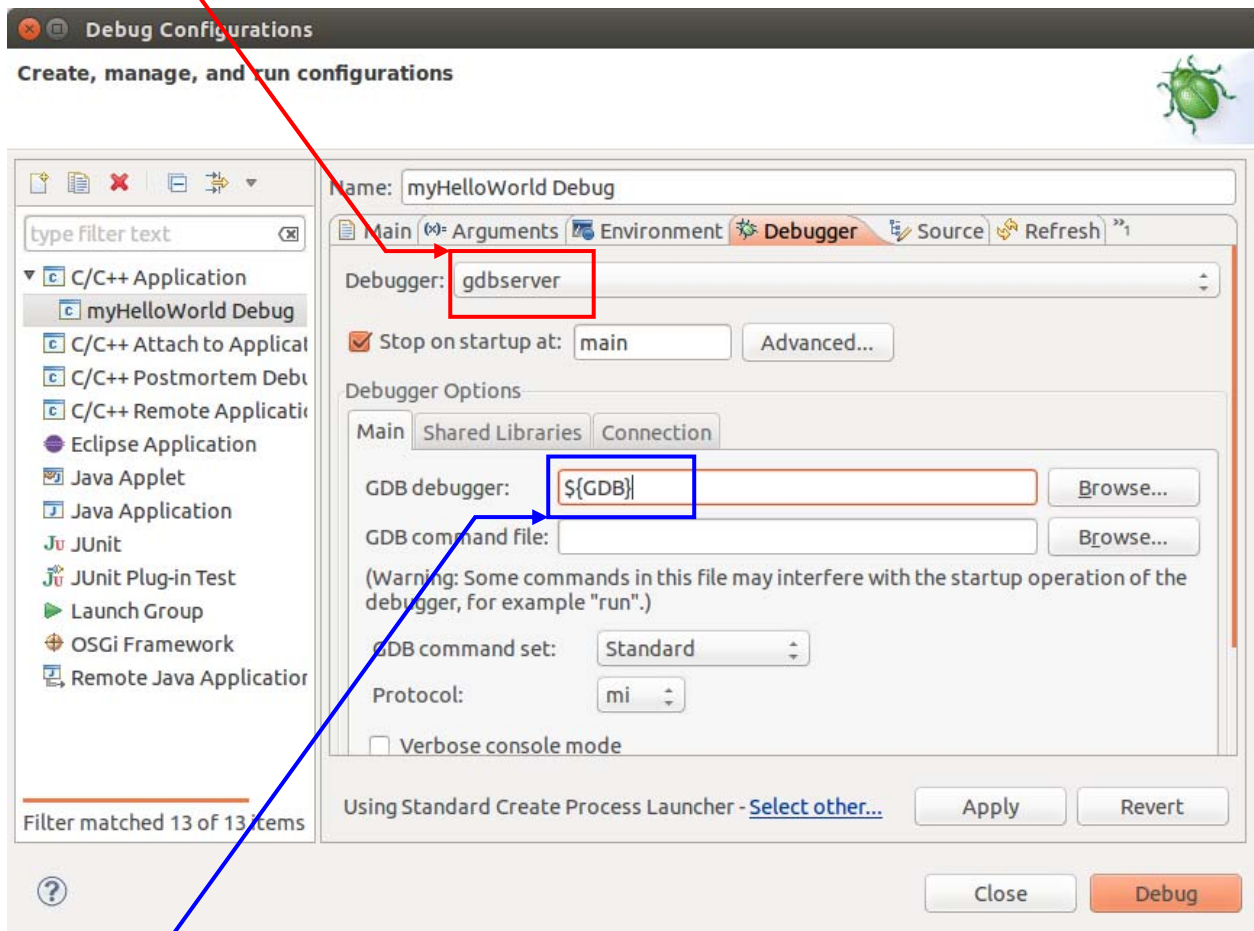
- Select **myHelloWorld** under *C/C++ Application* (to expand it double click on it).



- Select the **Debugger** tab.

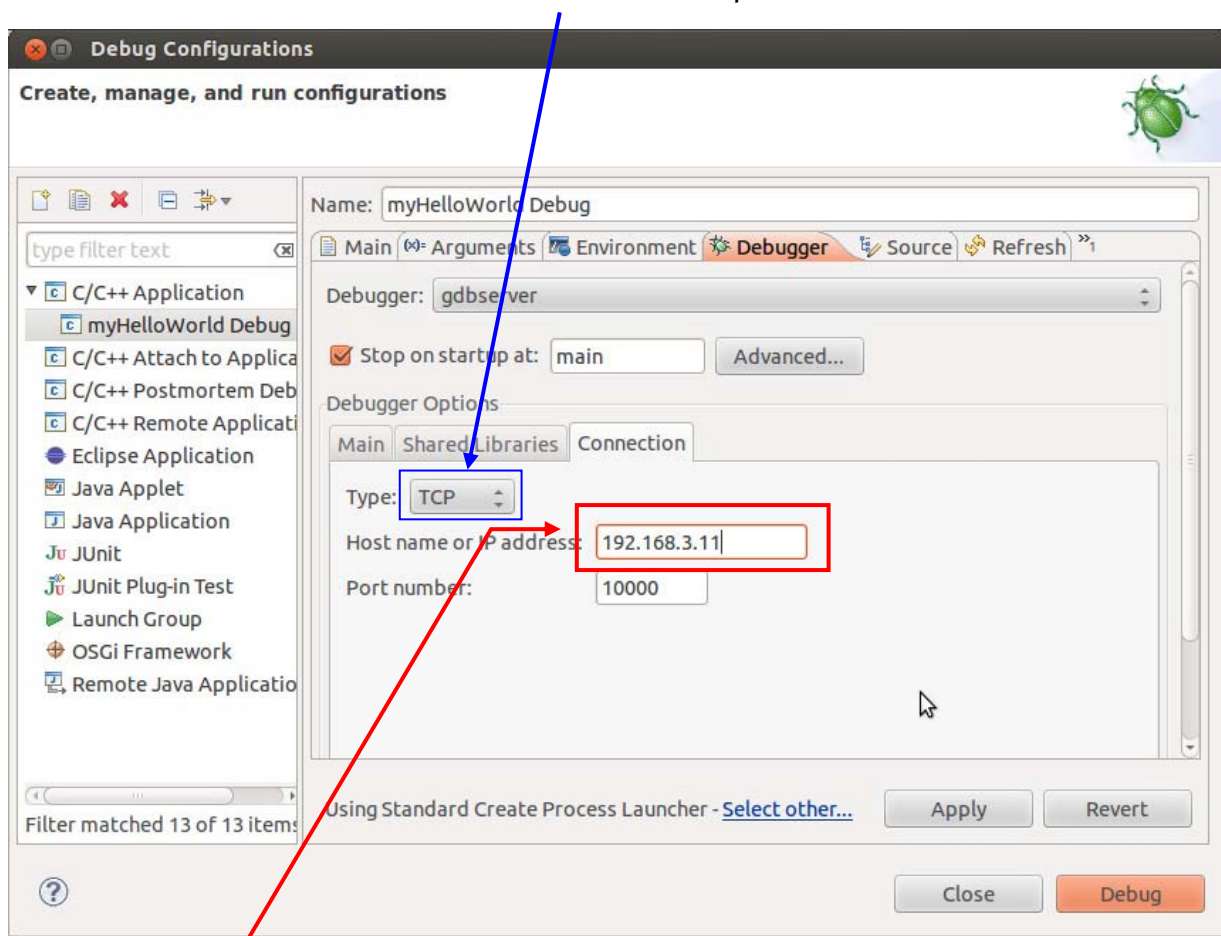


- Select ***gdbserver*** Debugger from the *Debugger* drop-down box.



- Enter ***\${GDB}*** in the *GDB Debugger* field
- Keep the *GDB command file* field empty.

- Select the **Connection** tab and select **TCP** in the drop-down box.



- Enter **192.168.3.11** (the target's IP address) in the *Host name* input field.

The host's *GDB* will connect to this IP address to communicate with the target's *GDB* server.

- Click **Apply**.
- Click **Debug**.

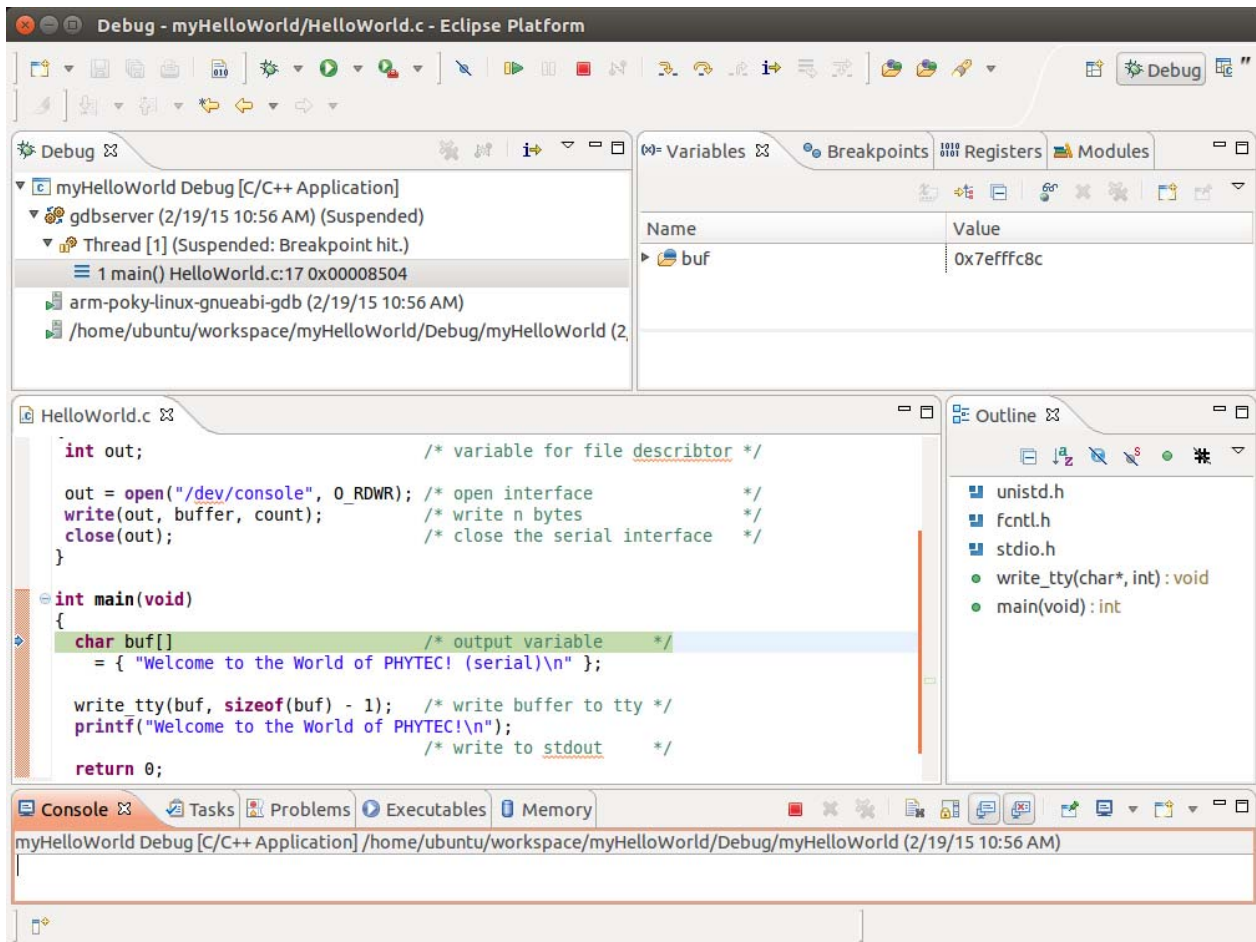
A new dialog appears.

- •Select **Yes** to switch to the *Debug* perspective.





The debug perspective opens and the debugger stops automatically at the first line. The host's *GDB* is now connected to the *GDB* server on the target.

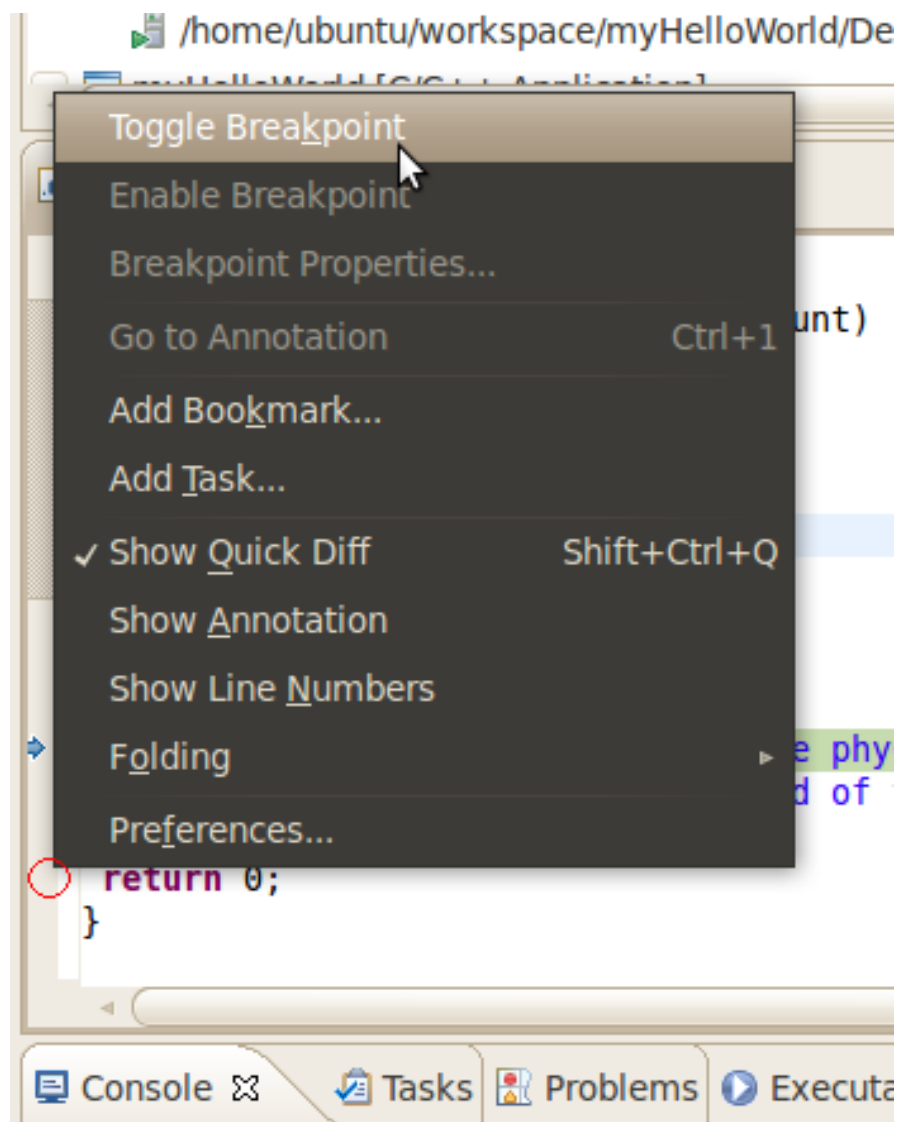


You have configured your project for remote debugging. You have started the *GNU* debugger in *Eclipse* and connected the host's *GDB* with the target's *GDB* server. You can now start to debug the project.

### 3.2.3 Setting a Breakpoint

Now you will set a breakpoint in your program. The breakpoint will be set on the last line of the function `main()`. If you resume the application, the debugger will stop at this line.

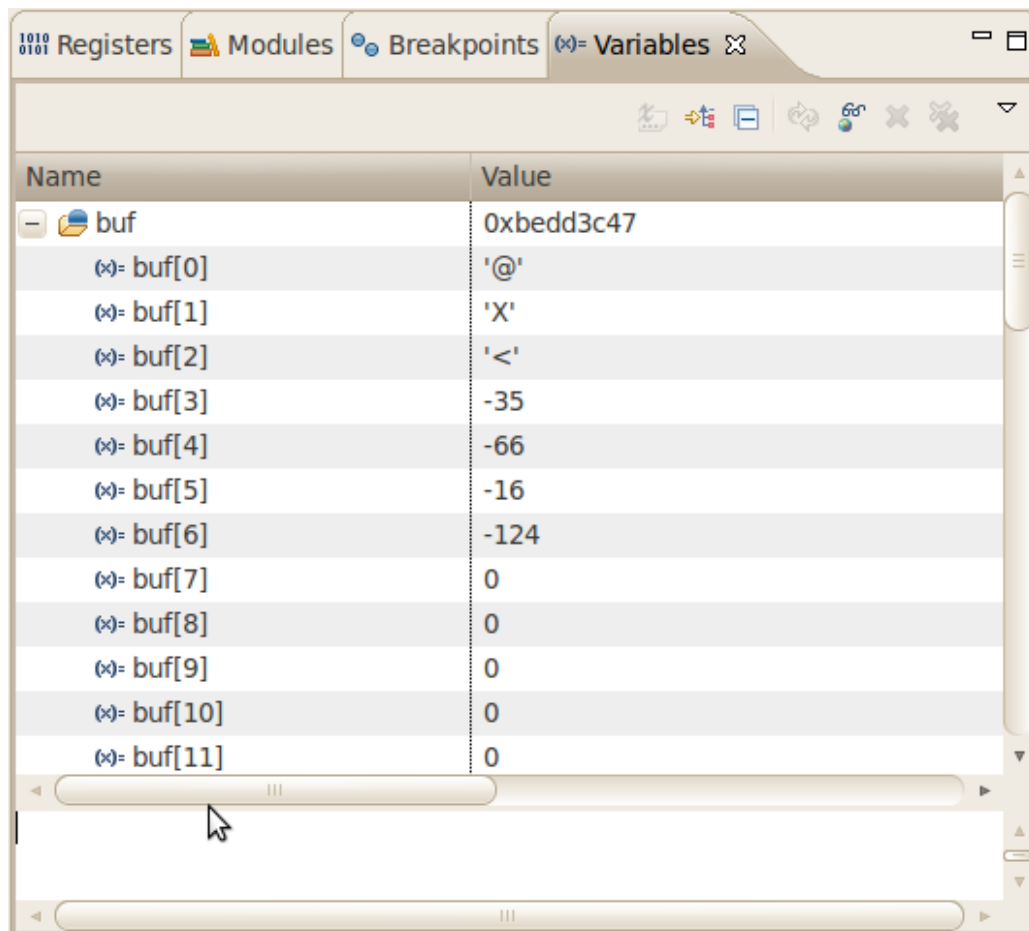
- Select the last line in `main()`.
- Right-click into the small grey border on the left-hand side and select **Toggle Breakpoint** to set a new breakpoint.



### 3.2.4 Stepping through and Watching Variable Contents

In this part you will step through the example project with the debugger. You will also learn how to check the content of a variable.

- Expand **buf** in the *Variables* window.

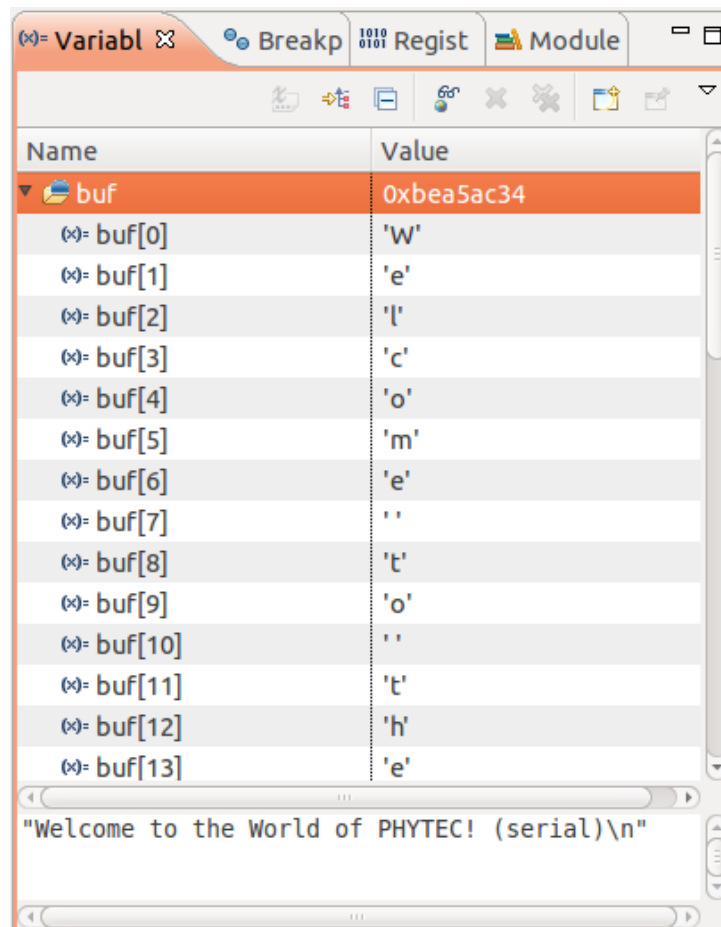


- Click the **Step Over** button in the *Debug* window to step to the next line. You will see the content of the *buf* variable in the *Variables* window.





- Click on the variable **buf**.

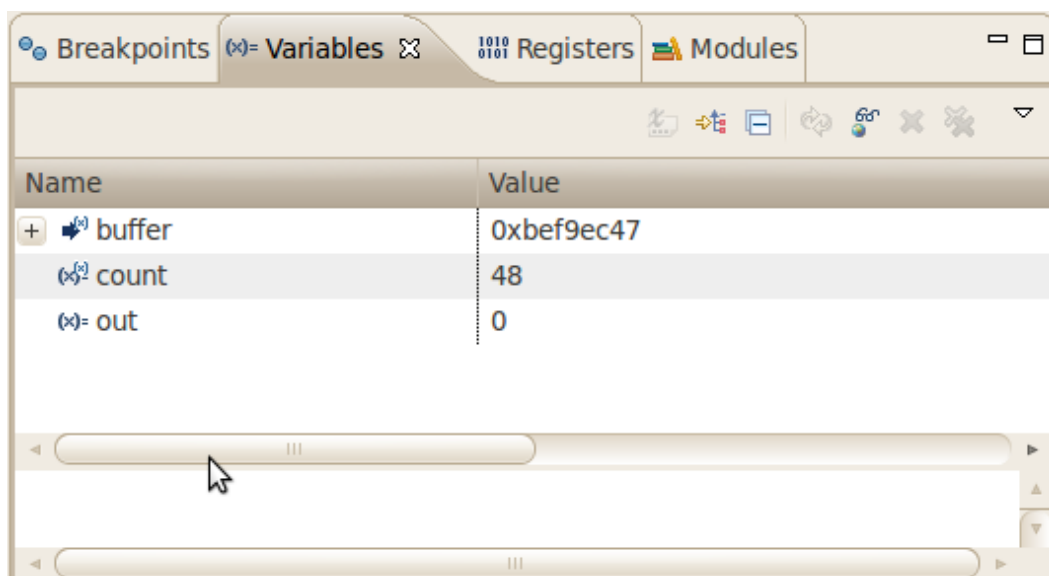


- Then click the button **Step into** to enter the function `write_tty()`.



- The debugger stops in `write_tty()`.

You will see the following variable window:



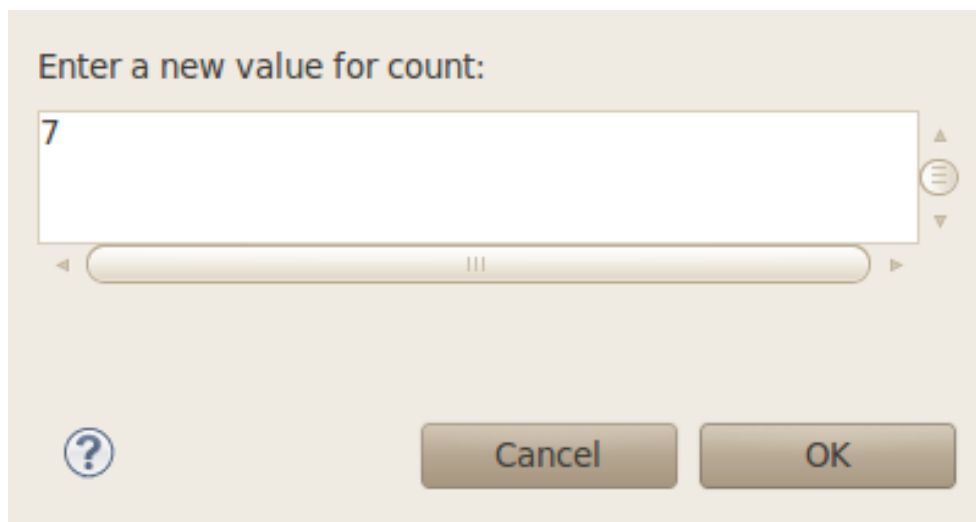
- Click on the variable **buffer**.

You will probably see a different address on the buffer pointer. Remember which address is shown in your case, you will need this address later.

### 3.2.5 Stepping through and Changing Variable Contents

In this section you will change the value of a variable. At the end of this part you will see the effect of this change.

- Select the **count** variable in the *Variables* window.
- Right-click on **count** and select **Change Value**.
- Change the value of count to **7** and click **OK**.



- Open *Microcom* if the application is not already opened.
- Go back to *Eclipse*.
- Click the **Step Over** button **twice**.



- Switch to *Microcom*.

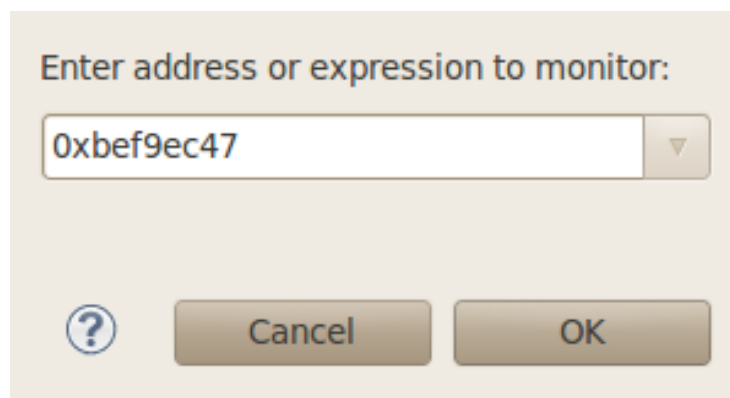
```
root@phyflex-imx6-2:~# gdbserver 192.168.3.11:10000 myHelloWorld
Process myHelloWorld created; pid = 827
Listening on port 10000
Welcome to the World of PHYTEC! (serial)
Remote debugging from host 192.168.3.10
Welcome
```

Because we changed the *count* variable to 7 only the first seven characters (*Welcome*) are displayed in the *Microcom* console.

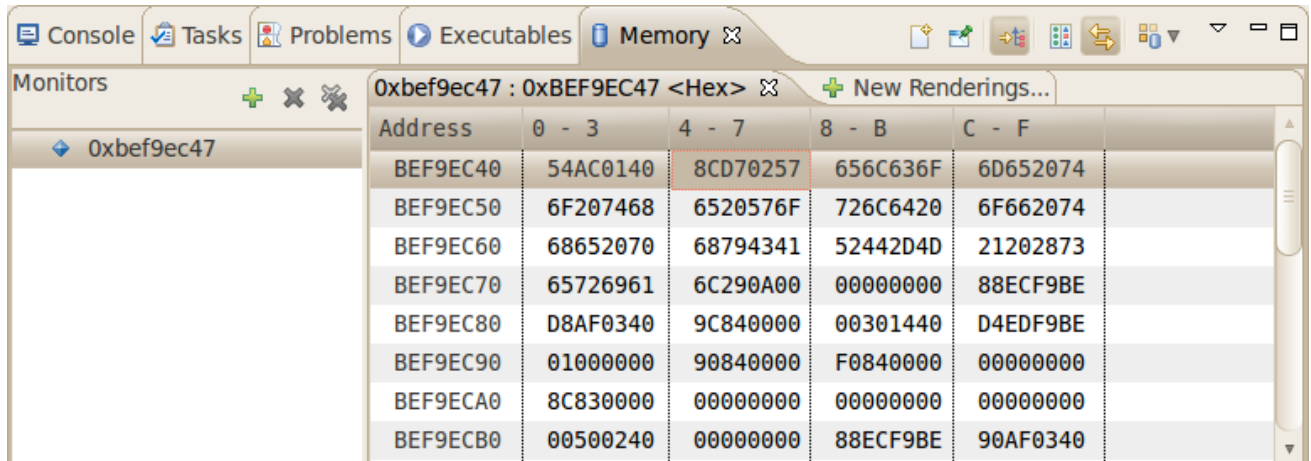
### 3.2.6 Using the Memory Monitor

In the last section of this chapter you will use the memory monitor to control the content at a memory address.

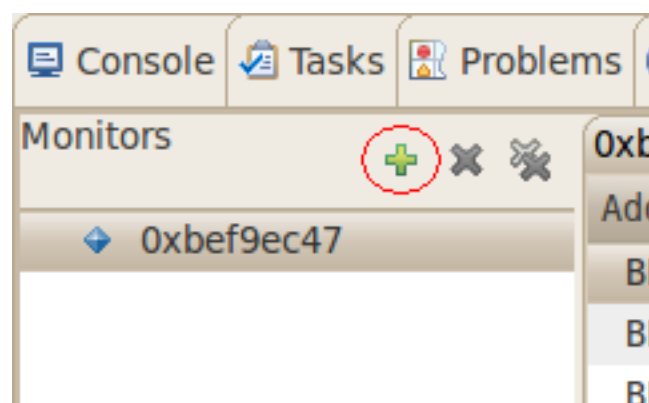
- Select the **Memory** tab in the frame where you can find the *Console*.
- Click **+ Add Memory Monitor**.
- Enter the address of the buffer and click **OK**. Remember that the variable's address might be different on your system.



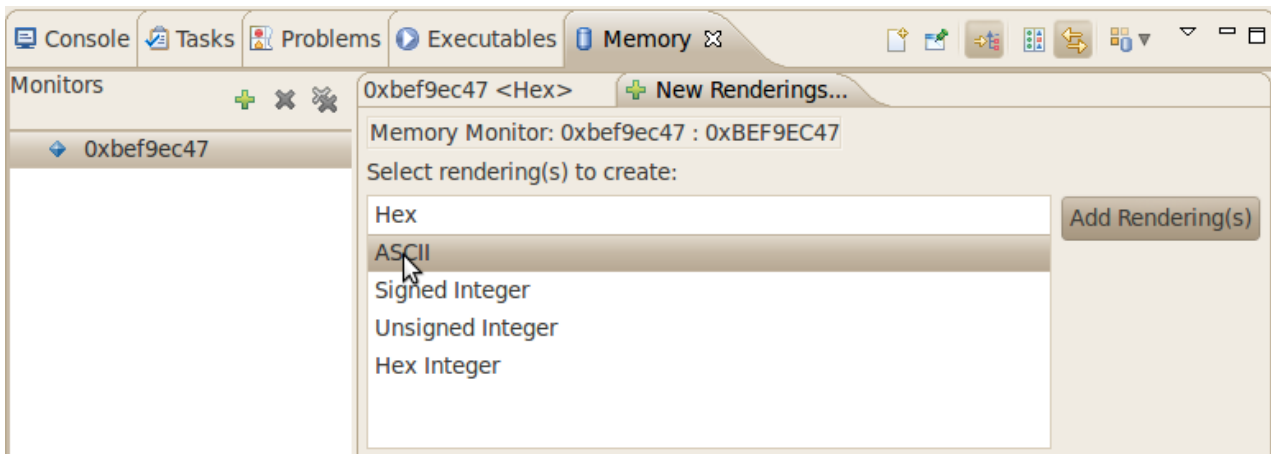
- Change the size of the window.



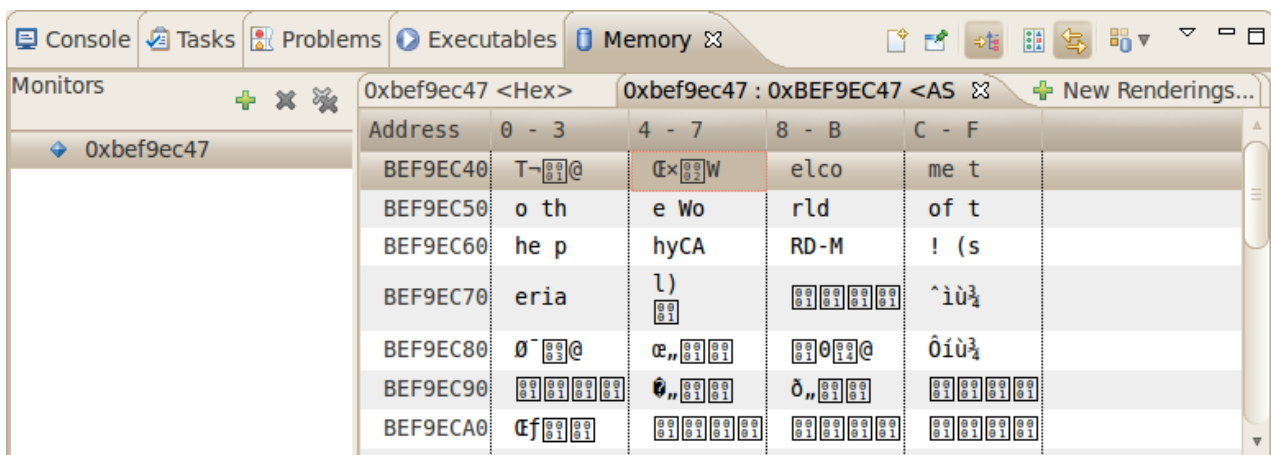
- Click **Add Rendering**.



- Select **ASCII** and click **Add Rendering(s)**.



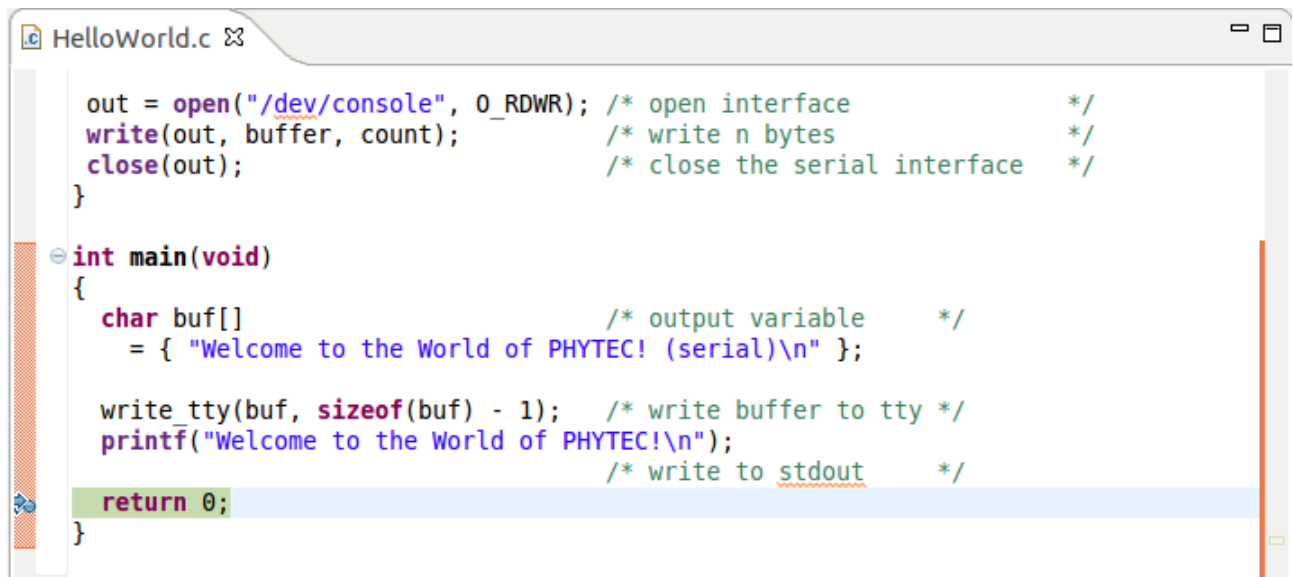
You can see the contents of the variable *buffer* at address *0xbef9ec47* (or at the specific address used on your system).



- Now click the **Resume** button from the menu bar.



The debugger stops at the breakpoint in the last line of *main()*.



```
out = open("/dev/console", O_RDWR); /* open interface */
write(out, buffer, count);          /* write n bytes */
close(out);                          /* close the serial interface */
}

int main(void)
{
    char buf[] /* output variable */
        = { "Welcome to the World of PHYTEC! (serial)\n" };

    write_tty(buf, sizeof(buf) - 1); /* write buffer to tty */
    printf("Welcome to the World of PHYTEC!\n");
    /* write to stdout */
    return 0;
}
```

- Click the **Resume** button to end the application.




You have successfully passed the debugging chapter. You are now able to configure and use *Eclipse* for remote debugging. You can step through a project, watch and change the content of variables, and you can use the memory monitor to view the content at a memory address.

## 4 Working with Qt Creator

In this section we learn how to work with *Qt Creator*. The *Qt* framework provides tools to develop graphical user interfaces. With the help of an example project we show how to compile your own *Qt* based programs and automatically transfer them to the target.

### 4.1 Stop the Running Qt Demo on the Target

A *Qt* demo application starts automatically by default after the target has booted completely.

	<p><i>Qt</i> provided by <i>Digia</i> is very commonly used for embedded systems and it is supported by this BSP. Please visit <a href="http://www.qt.io">http://www.qt.io</a> in order to get all the documentation that is available about this powerful cross-platform GUI toolkit.</p> <p>Some demos that show the capabilities of <i>Qt</i> version 5.3.2 are included in the BSP.</p>
---	---


Before we start compiling and running our example project out of *Qt Creator* we must first close this *Qt* application.

- Open a serial connection with *Microcom*.



- After the target is booted login with root and enter the following command to stop the *Qt* application:  
`systemctl stop qt5demo`

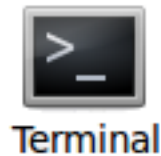
After the *Qt* demo is stopped with the command above we can start to use *Qt Creator*.

	<p>If you want to remove the <i>qt5demo</i> from the autostart enter the following command:  <code>systemctl disable qt5demo</code></p>
---	---

### 4.2 Importing the Demo Application

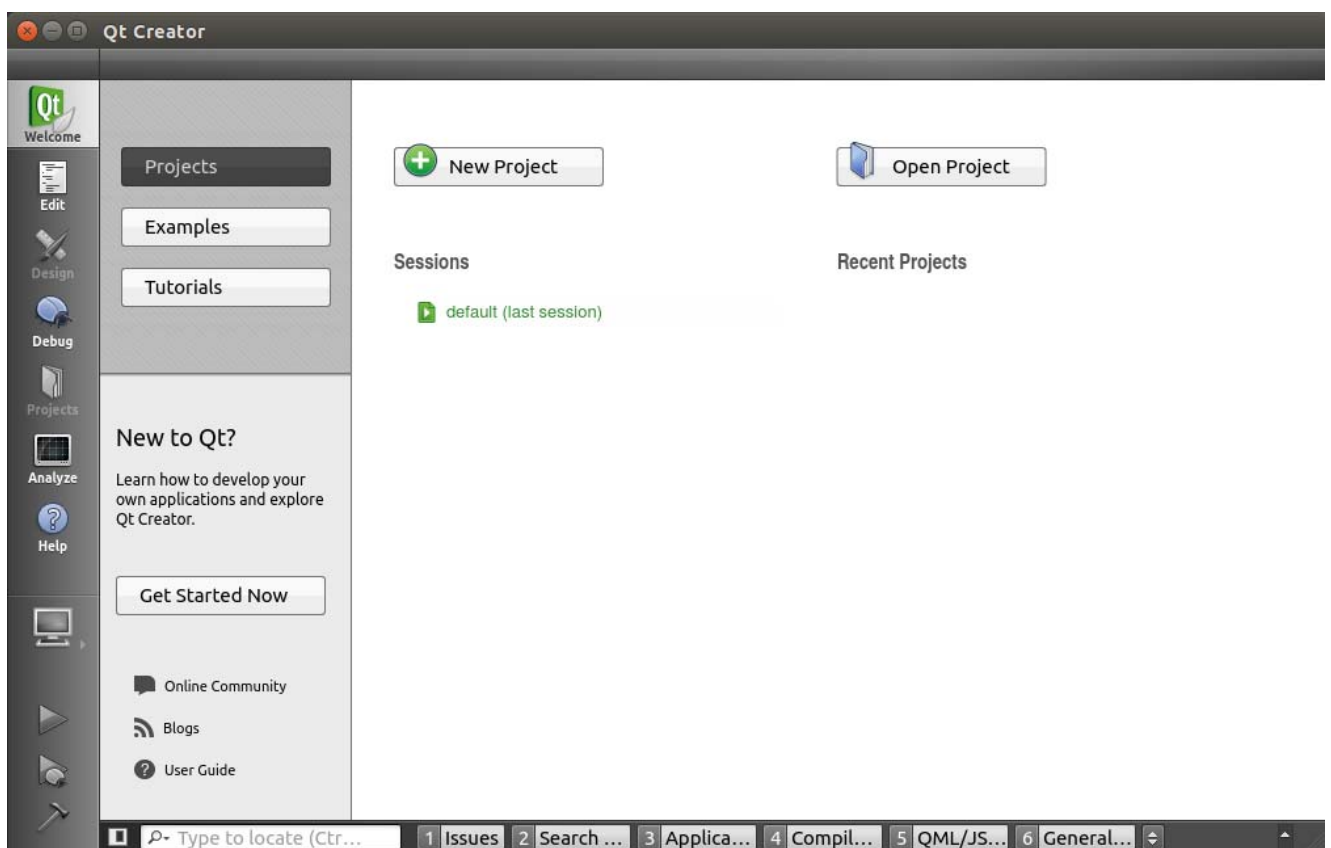
We start with opening the *Qt Creator* in a terminal, because otherwise the correct environment of the toolchain is not set.

- Open a terminal.



- Enter the following command in the terminal to start *Qt Creator*  
`/usr/bin/qtcreator.sh &`

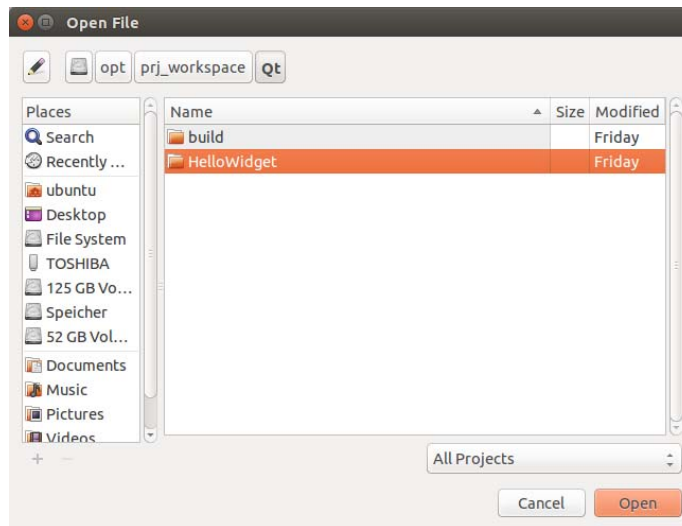
The user interface of *Qt Creator* appears:



- Now we import the example project by clicking ***Open Project***.

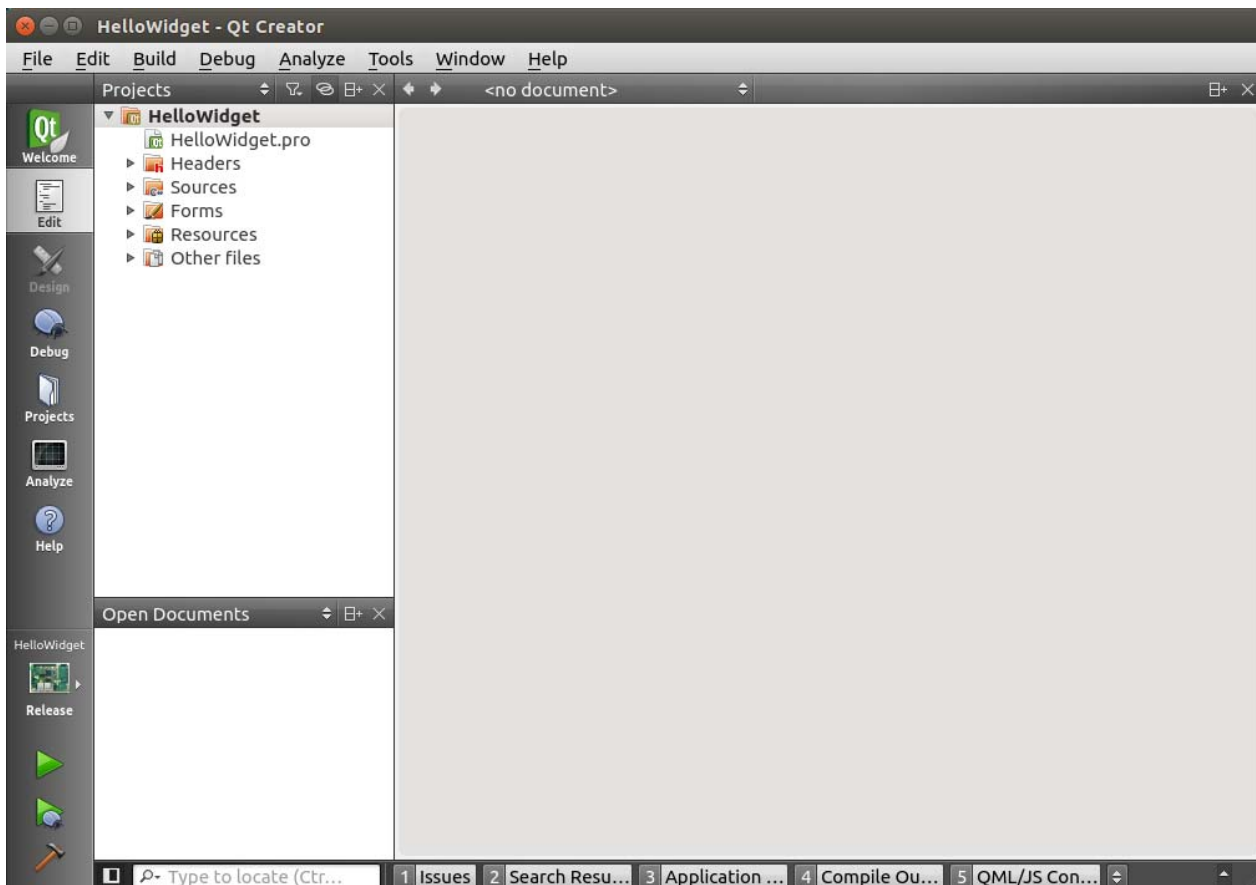
This opens a dialog in which the path to `/opt/prj_workspace/Qt` is set automatically.





- Double-click on the **HelloWidget** folder.
- Select **HelloWidget.pro** and click **Open**.

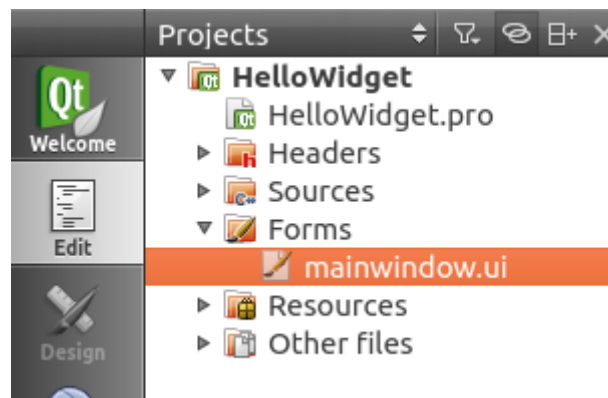
The *HelloWidget* project is now imported into your *Qt Creator* workspace.



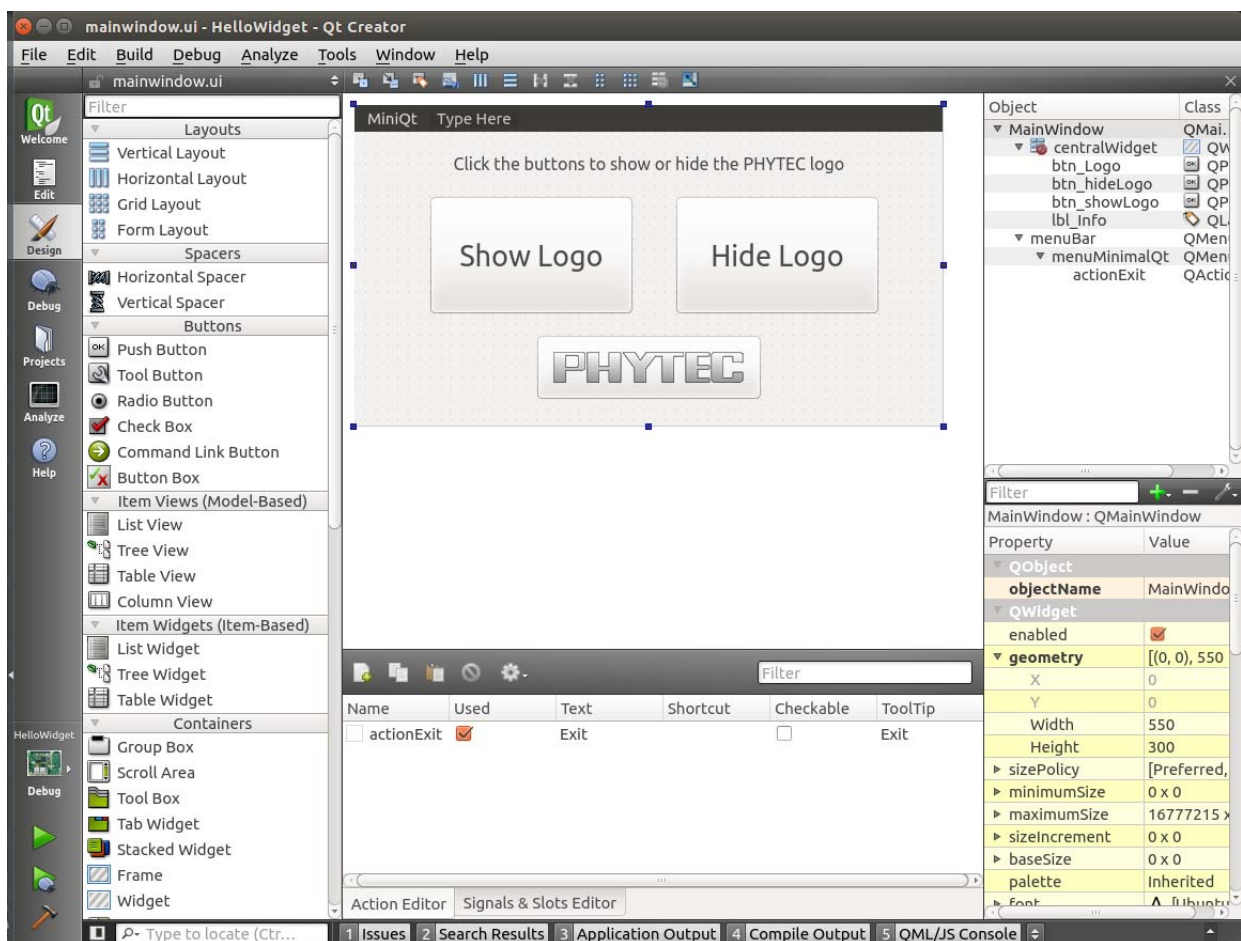
### 4.3 Work with the Demo Application

Our example project is a simple *Qt Widget Application*. First we take a look at the user interface of our example:

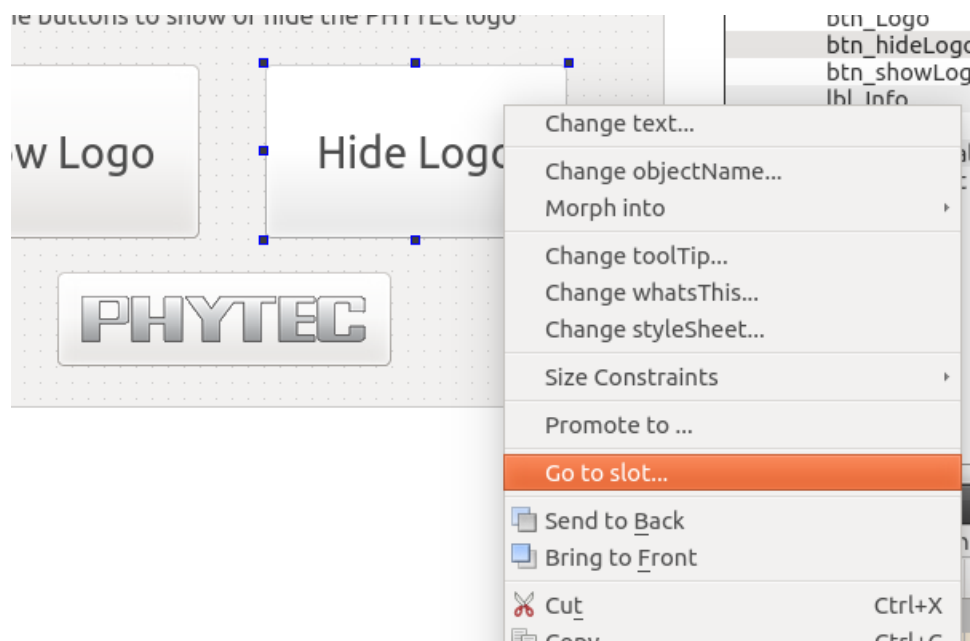
- Expand the folder **Forms** under the project sidebar on the left side and double-click **mainwindow.ui**.



Qt Creator opens the design mode and you can see the design of our project, which has a menu bar with an exit action under the menu item *miniQT*, one label and three buttons including one with the PHYTEC logo inside.



- Right-click the “Hide Logo” button and select **Go to slot ...** from the context menu.

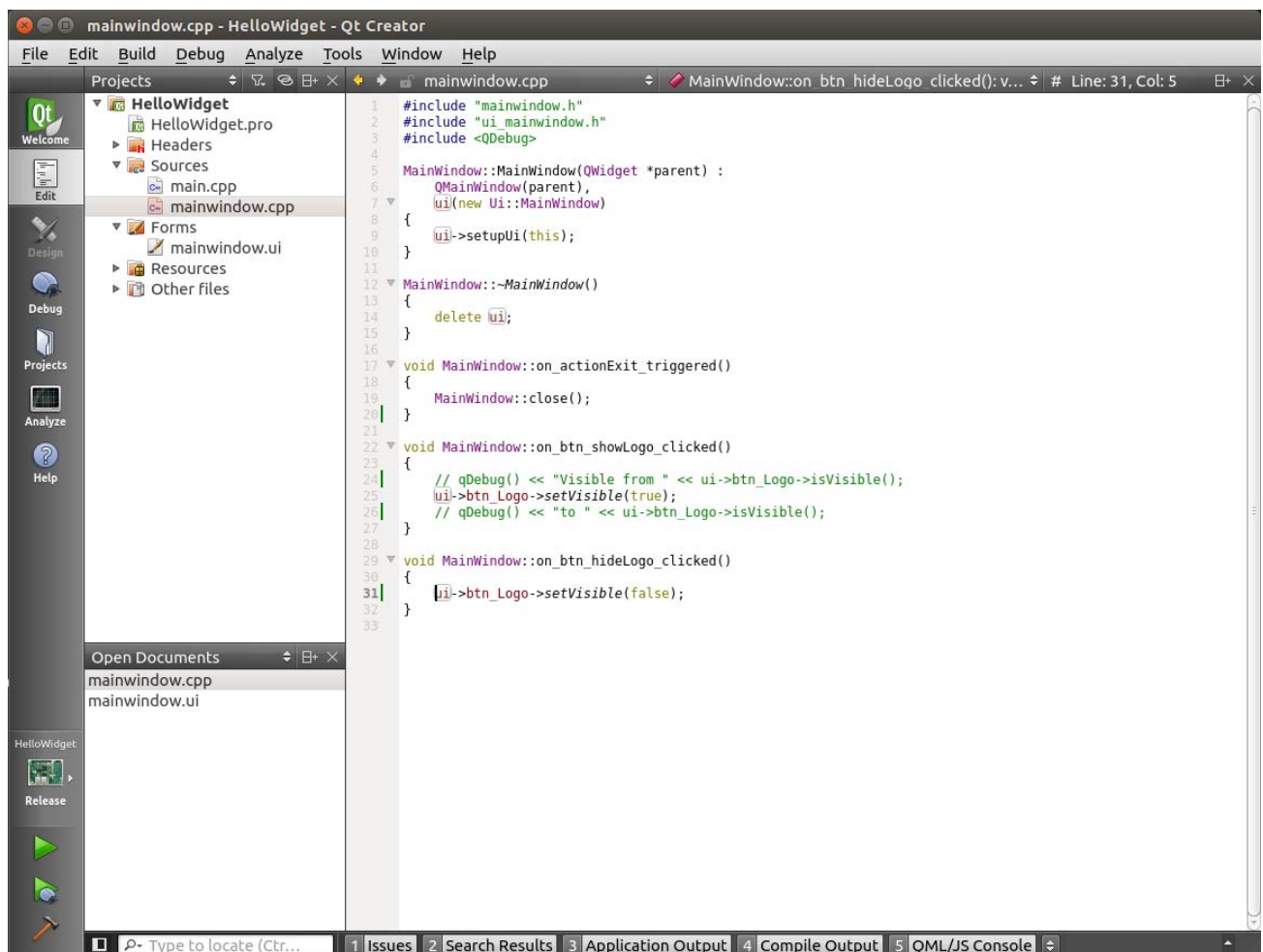


A window opens allowing you to select a signal.

- Select **clicked()** and click **OK**.



Now *Qt Creator* jumps into the *mainwindow.cpp* where you can find the definition of the function *on\_btn\_hideLogo\_clicked()*. You see that clicking this button changes the status of the *btn\_Logo* button from visible to hidden.



Next we build and run the example.

## 4.4 Compile and Run the Demo Application on the Target

Now we want to compile and run the demo application on the phyFLEX-i.MX6. The correct Qt settings are already preset in the virtual machine, so the project can be build directly.



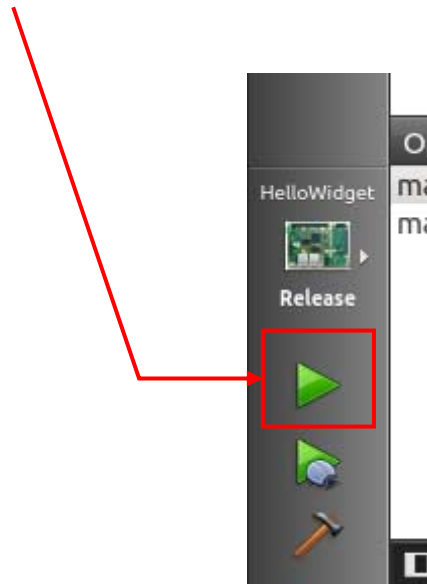
If you want to check the settings click **Tools ► Options...** in the menu bar. A new dialog appears. Click **Build & Run** if it is not already selected. Now, you can select different tabs to see the settings for the workspace, compiler, debugger, *Qt* version and a lot other options.



Be sure that the target is connected via Ethernet and is powered on. As described before do not forget to stop all actually running *Qt* applications on the target.

With only one click the project will be built, deployed to the target and executed.

- Click the **green filled triangle** near the bottom of the gray bar on the left.

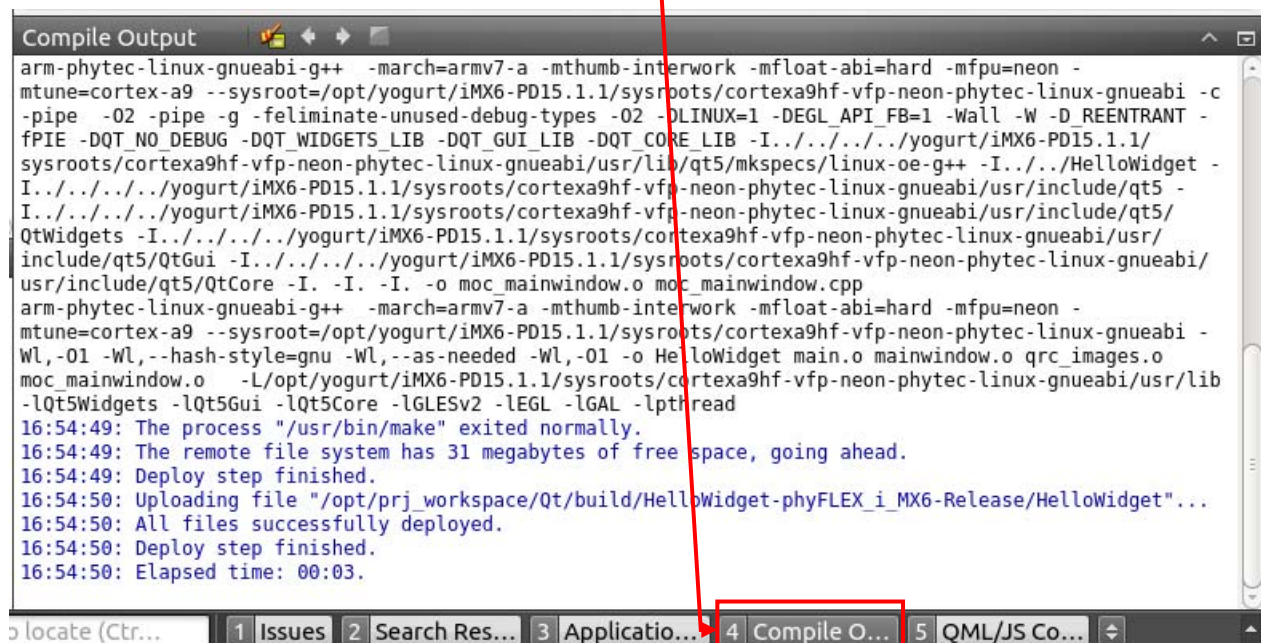


After the target is successfully deployed on to the target the *Application Output* frame is shown in the frame under the *mainwindow.cpp* source code where you can see the prompt from the phyFLEX-i.MX6.



On the connected display you will see the *HelloWidget* application. Click the big button with the PHYTEC logo to enable and disable the Welcome label.

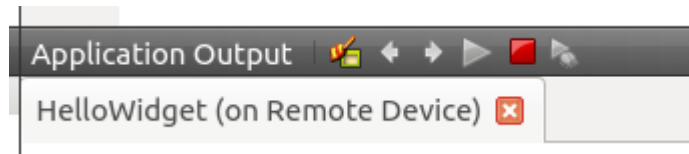
- To see the different build steps click **Compile output** which can be found in the bottom menu bar.



The *Compile output* frame is opened and you can scroll threwh through the compile steps.



If you want to stop the application on the target switch back to the **Application output** frame and click the **red rectangle** there.



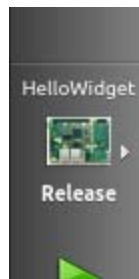
You have successfully imported and built a *Qt* project with *Qt Creator*. You have also learned how to download and execute your application on the target.

## 4.5 Compile and Run the Demo Application on the Host

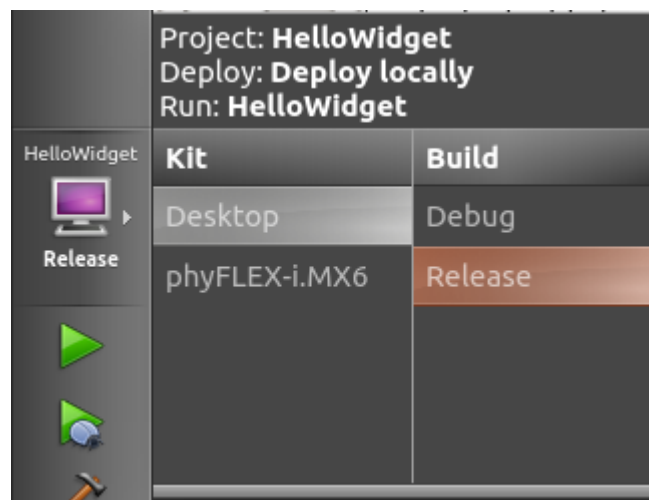
In some cases you do not want to compile and run the application on the target. For example if the phyFLEX-i.MX6 is not connected to the host or if you only changed some user interface relevant things and you do not want to copy the application and all the necessary resources to the target. Running the demo application on the host is faster, but features specific to the target do not work.

To change the target for downloading and running the application do following steps:

- In the gray bar on the left click the button with the small phyFLEX-i.MX6 picture.



- In the opened context menu select **Desktop** as **Kit** and **Release** as **Build**.





- Click outside of the context menu to close it.
- Now you can start the compilation and execution of the application by clicking the **green triangle**.
- After the application is compiled a window opens displaying the application running.



- Close the running application.

## 4.6 Debugging the Demo Application

We finish the *Qt Creator* chapter by showing how to debug the *HelloWidget* demo application.

### 4.6.1 Using QDebug for simple Debugging Messages

In our first debugging step we use the *QDebug* class. This class provides an output stream for debugging information. It is used whenever the developer needs to write out debugging or tracing information to a device, file, string or console.

To make use of the *QDebug* functions *QDebug* is already included in the header of the file *mainwindow.cpp*.

In this file you will also see two out-commented lines in the function *on\_btn\_showLogo\_clicked()* which start with a *QDebug()* command. This is an example how to make use of *QDebug()*.

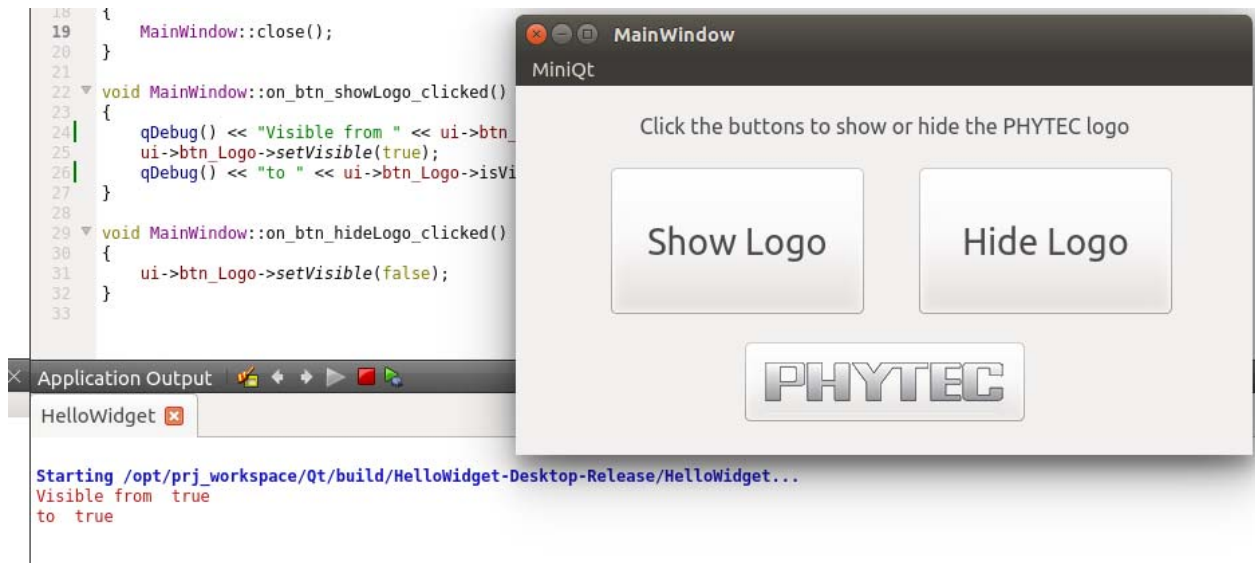
```

22 void MainWindow::on_btn_showLogo_clicked()
23 {
24     // qDebug() << "Visible from " << ui->btn_Logo->isVisible();
25     ui->btn_Logo->setVisible(true);
26     // qDebug() << "to " << ui->btn_Logo->isVisible();
27 }

```

- Remove the comment flags ( // ) from both "qDebug lines" to enable the two debug messages.
- Save the changes with **Ctrl + S**.
- Open the *Application Output* if it is not already opened.
- Build and run the application on the *Desktop* with the build configuration *Release*, as it was shown in the chapter before.

A window with the running application opens.

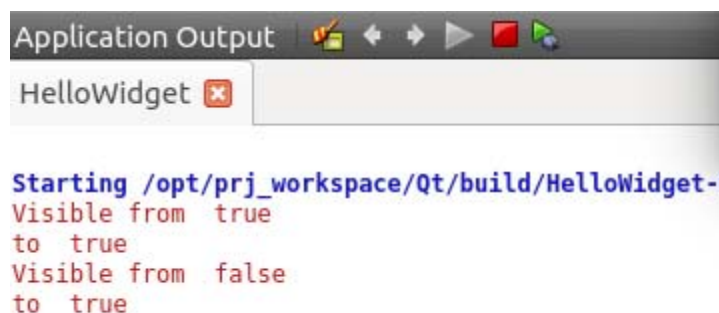


- Press the *Show Logo* button.

The logo was visible before we pressed the button so we see the appropriate message from `qDebug()` in the *Application Output*.

- Press the *Hide Logo* button and then the *Show Logo* button.

Now the logo was hidden before we pressed the *Show Logo* button again and we see the appropriate message from `qDebug()` in the *Application Output*.

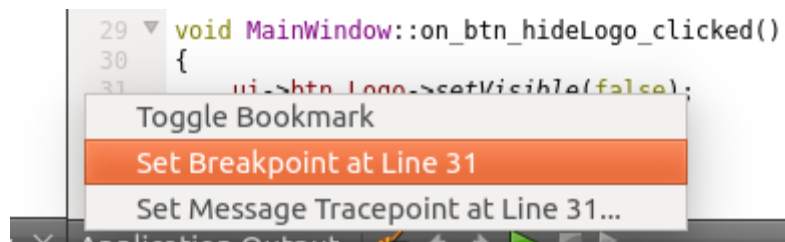


*QDebug* is a simple way to generate debug information. In the next chapter we use the debugger integrated in *Qt Creator*.

### 4.6.2 Using the integrated Qt Creator Debugger

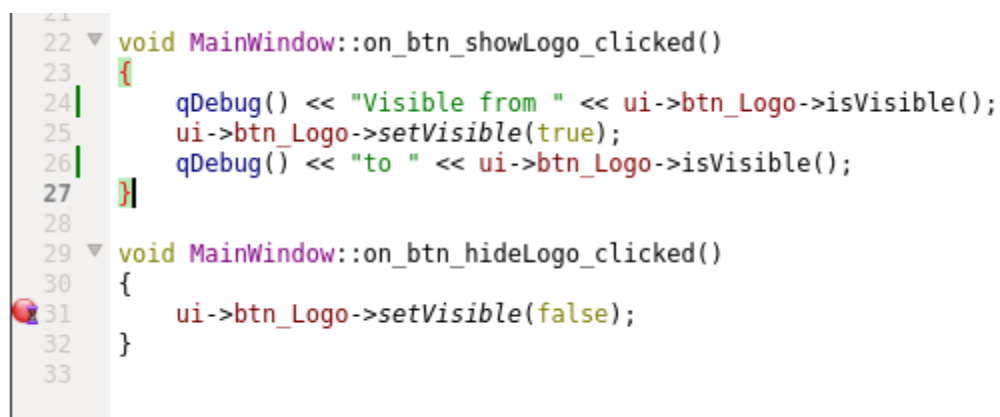
Before we start the integrated Debugger we first set a breakpoint where the demo application stops.

- Open the *mainwindow.cpp* if it is not already open.
- Right-click in front of line number 31.



- In the context menu click on *Set Breakpoint at Line 31* to add a breakpoint.

You now see a red filled circle with a small sand glass in front of this line.

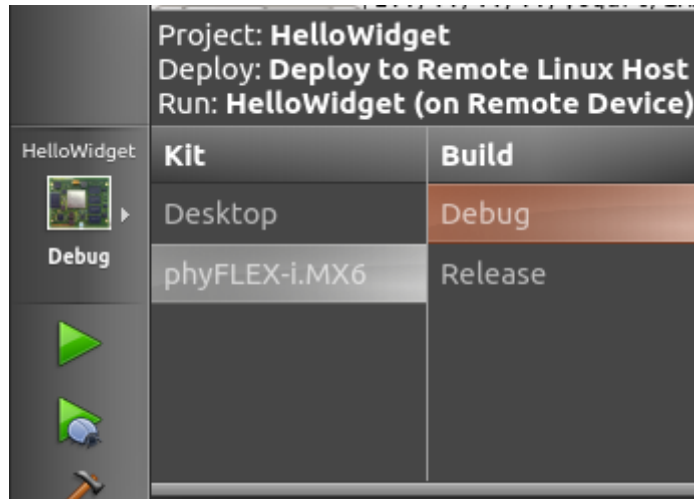


The next step is to change the build configuration from *Release* to *Debug*.

- In the gray bar on the left click the button with the small Desktop icon.



- In the opened context menu select **phyFLEX-i.MX6** as *Kit* and **Debug** as *Build*.

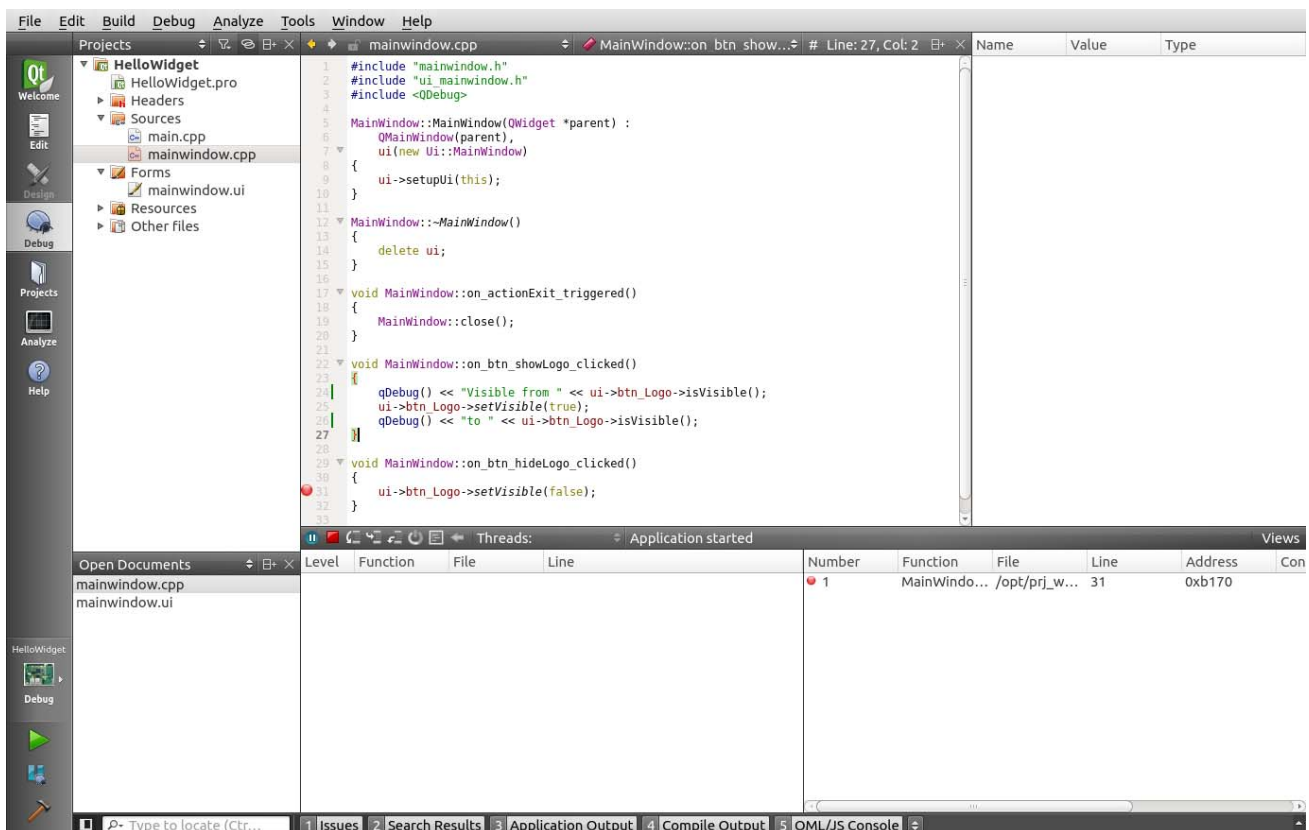


Be sure that the target is connected via Ethernet to the host.

- Start Debugging by clicking on the green filled triangle with the small magnifier.



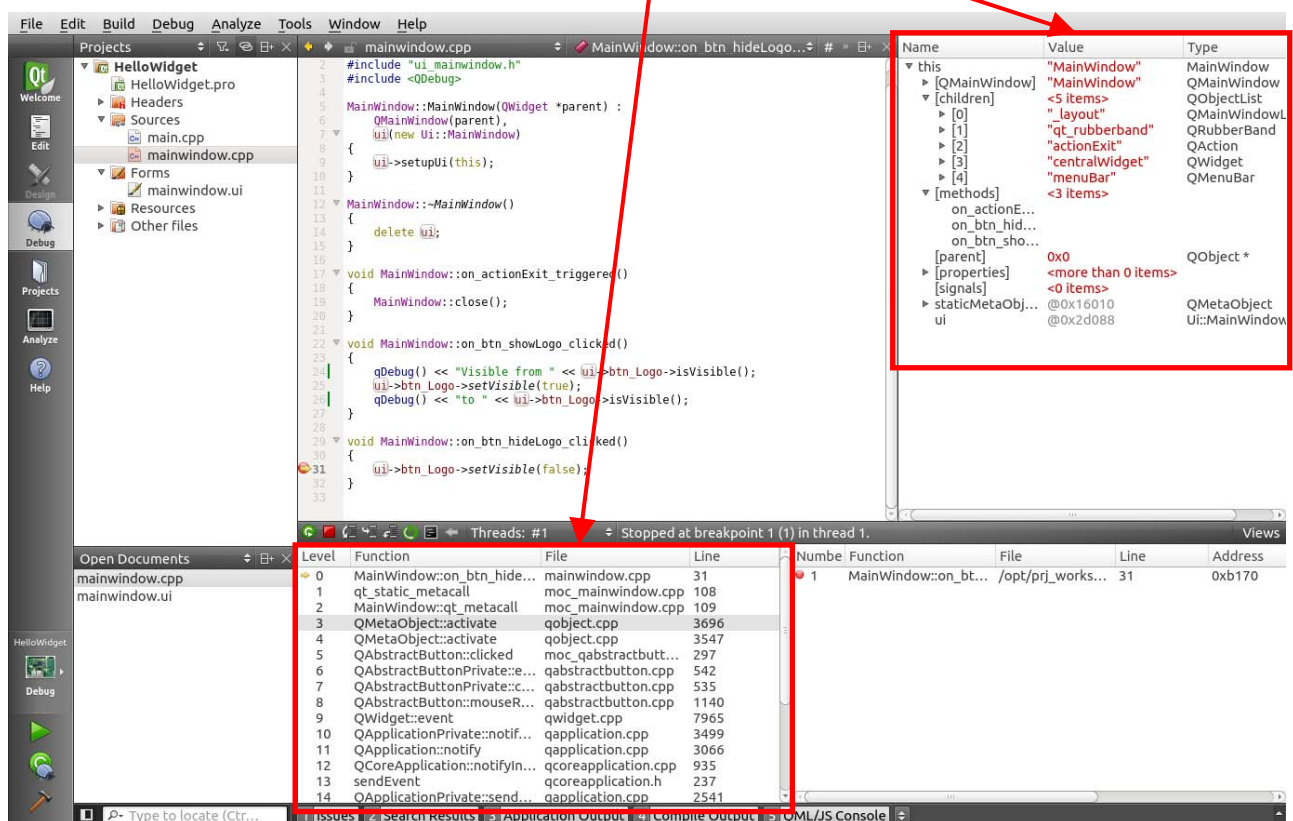
The debugger starts and *Qt Creator* changes his view to the Debug mode.



The demo application is shown on the connected display.

- In the running demo application click on the *Hide Logo* button.

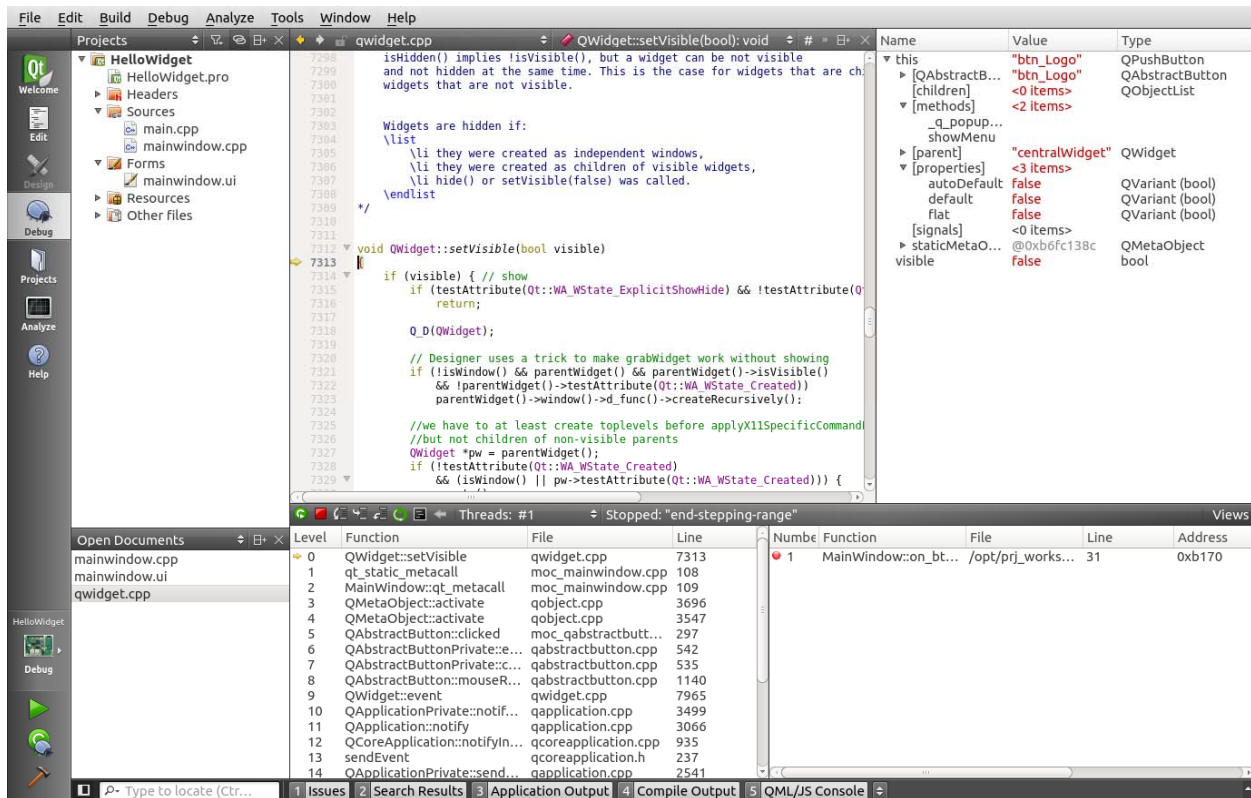
When the button is pressed the debugger stops at the created breakpoint, because the function is called. You can now watch the *stack* or the *Locals and Expressions*.



- Step into the `setVisible` function by pressing the *Step Into* button.

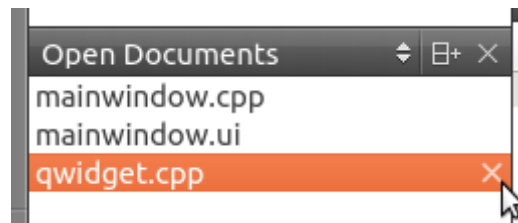


Now the definition of this function is opened.

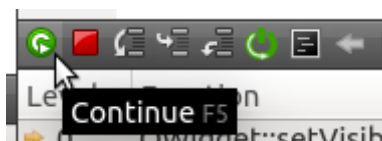


As many other calls are following it is recommended to close the opened `qwidget.cpp`.

- Close `qwidget.cpp` in the Open Documents frame by clicking the X on the right side.

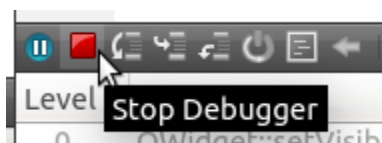


- Continue the stopped application by clicking on *Continue*.



Now the demo application is continued and waiting for user interactions.

- Stop the Debugger by pressing *Stop Debugger*.



You have successfully finished our short introduction to the Debugger from Qt Creator.



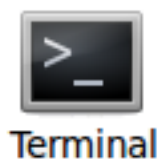
## 5 Working with Yocto



In this chapter of the Quickstart we describe only the basic *Yocto* usage. For in-depth explanations refer to the *Yocto* Reference Manual.

*In this part you will learn how to use the Yocto Project and the PHYTEC BSP.*

### 5.1 Add new packages to the Root Filesystem



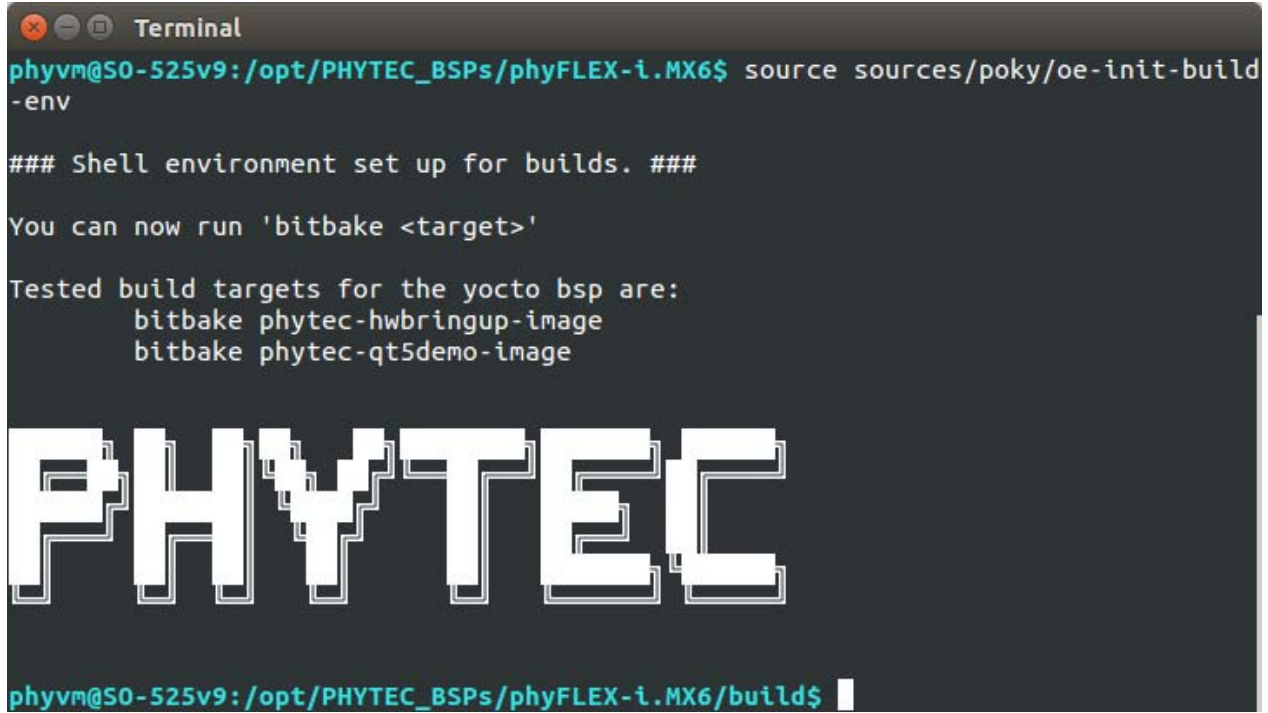
You can find the pre-build BSP in the virtual machine under `/opt/PHYTEC_BSPs/phyFLEX-i.MX6/`. All necessary tools and programs are already pre-installed to directly start with *Yocto*. Open a new terminal if you have not already done and change to the directory of the pre-installed BSP:

- Click the ***terminal*** icon on your desktop.
- Type the following command to change to the BSP-directory:  
`cd /opt/PHYTEC_BSPs/phyFLEX-i.MX6/`

This directory is the start point for our BSP. You will also find further documents in this directory. First we must set the correct environment.

- Enter the following command `source sources/poky/oe-init-build-env`





```
Terminal
phyvm@S0-525v9:/opt/PHYTEC_BSPs/phyFLEX-i.MX6$ source sources/poky/oe-init-build-env

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

Tested build targets for the yocto bsp are:
    bitbake phytec-hwbringup-image
    bitbake phytec-qt5demo-image

PHYTEC

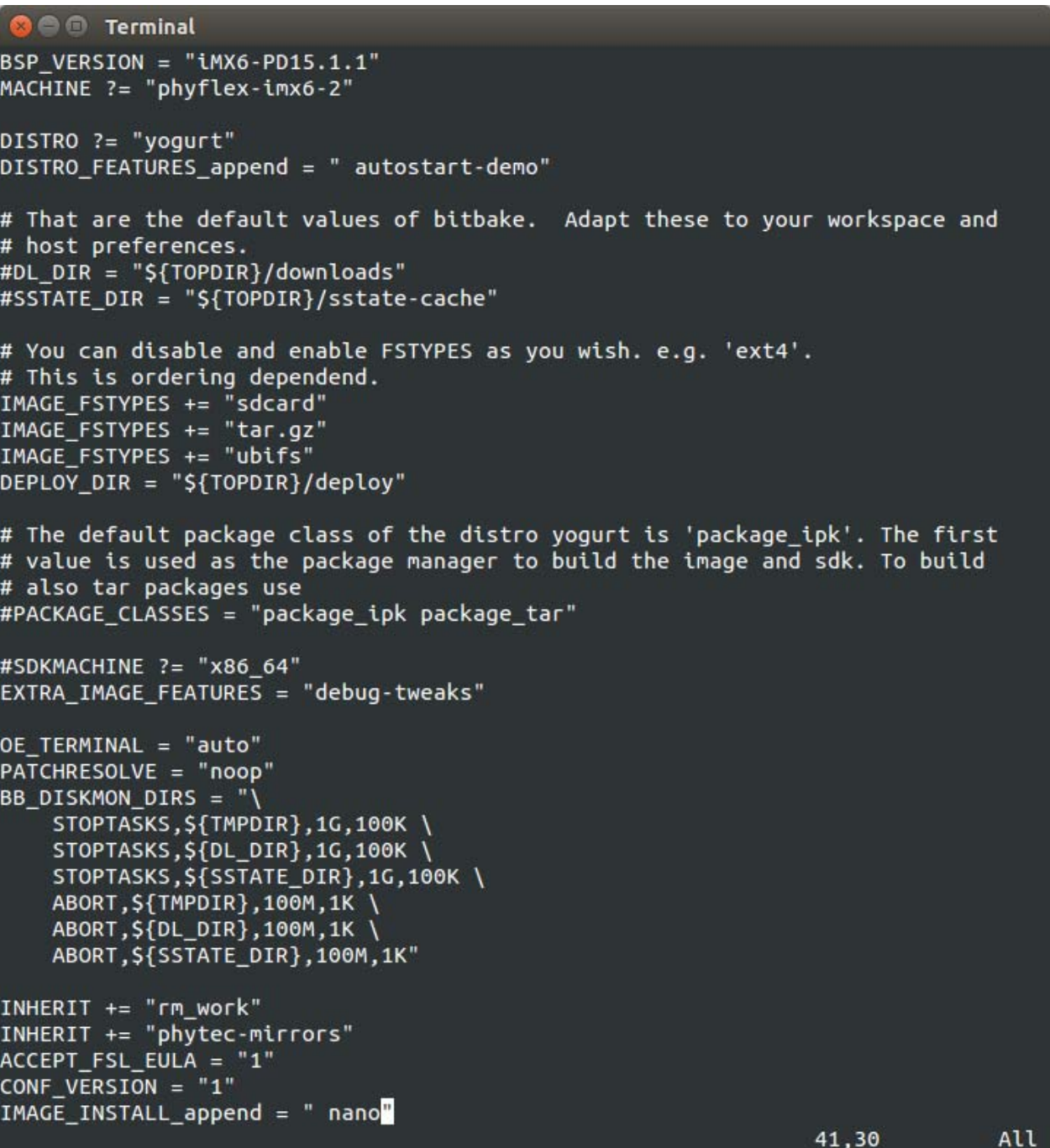
phyvm@S0-525v9:/opt/PHYTEC_BSPs/phyFLEX-i.MX6/build$
```

The environment is set and we are now in the build folder. In this example we want to add the *nano editor* to the image. Usually new packages are downloaded from an appropriate server prior to building the image unless they are stored locally. In the virtual machine the package for the *nano editor* is already pre-downloaded and available locally.

Before we add the package we check if the right configuration is set.

- Open the configuration file with any editor, e.g. *VI*, with the following command:  
`vi conf/local.conf`
- Check the *MACHINE* variable in this file.

- By default the variable is set to *phyflex-imx6-2*. This is the correct *MACHINE* for the PFL-A-02-13237E0 module..



```

Terminal
BSP_VERSION = "iMX6-PD15.1.1"
MACHINE ?= "phyflex-imx6-2"

DISTRO ?= "yogurt"
DISTRO_FEATURES_append = " autostart-demo"

# That are the default values of bitbake. Adapt these to your workspace and
# host preferences.
#DL_DIR = "${TOPDIR}/downloads"
#SSTATE_DIR = "${TOPDIR}/sstate-cache"

# You can disable and enable FSTYPES as you wish. e.g. 'ext4'.
# This is ordering dependend.
IMAGE_FSTYPES += "sdcard"
IMAGE_FSTYPES += "tar.gz"
IMAGE_FSTYPES += "ubifs"
DEPLOY_DIR = "${TOPDIR}/deploy"

# The default package class of the distro yogurt is 'package_ipk'. The first
# value is used as the package manager to build the image and sdk. To build
# also tar packages use
#PACKAGE_CLASSES = "package_ipk package_tar"


#SDKMACHINE ?= "x86_64"
EXTRA_IMAGE_FEATURES = "debug-tweaks"

OE_TERMINAL = "auto"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS = "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K"

INHERIT += "rm_work"
INHERIT += "phytec-mirrors"
ACCEPT_FSL_EULA = "1"
CONF_VERSION = "1"
IMAGE_INSTALL_append = " nano"
41,30 All

```


- In the same file we can add *nano* by adding the following line at the end of the file (as shown in the previous picture):  
`IMAGE_INSTALL_append = " nano"`
- Save the changes and close the file with **:x** and **Enter**.

	<p>You can search for more available packages at:  <a href="http://layers.openembedded.org/layerindex/branch/master/recipes/">http://layers.openembedded.org/layerindex/branch/master/recipes/</a></p> <p>Be sure that the layer on which the package depends is in our BSP. You can check our integrated layers in the file <i>conf/bblayers.conf</i>.          When your needed layer is not in our BSP you must first checkout the layer to the folder <i>/opt/PHYTEC_BSPs/phyFLEX-i.MX6/source/</i>.</p> <p>Please note that there is no a guarantee that the build process proceeds successfully with all new packages added.</p>
---	--

Now you can start building the configured BSP. There are two methods to create the images. You can use either *bitbake phytec-hwbringup-image*, or *bitbake phytec-qt5demo-image*

The first option leads to a faster compilation and smaller images, because all *Qt* relevant features are left out.

- In our example we want to keep the *Qt* relevant features, so we type following command:  
[bitbake phytec-qt5demo-image](#)

	<p>Because of dead links it is possible that an attempt to download a source package fails. PHYTEC cannot guarantee the accessibility of all the web pages that are needed to build a BSP. In case of a failure please check our ftp-server at <a href="ftp://ftp.phytec.de/pub/BSP_PACKAGES/EXTERNALS">ftp://ftp.phytec.de/pub/BSP_PACKAGES/EXTERNALS</a>, and download the source package from there into the source directory under <i>/opt/PHYTEC_BSPs/phyFLEX-i.MX6/build/downloads</i> manually. After that, please restart the build process by again calling <a href="#">bitbake phytec-qt5demo-image</a> or <a href="#">bitbake phytec-hwbringup-image</a>.</p> <p>If the package is also missing on our ftp-server, please send an email to <a href="mailto:support@phytec.de">support@phytec.de</a>. We will then take it from our build-server and upload it onto our ftp-server for you.</p>
---	---

Among other images you will find the kernel named *zImage*, the device tree named *zImage-imx6q-phytec-pbab01.dtb* and the root file system named *phytec-qt5demo-image-phyflex-imx6-2.ubifs* in the directory *deploy/images/phyflex-imx6-2/*.

The three files are only a symbolic link to the correct images with the timestamp of the build date.

## 5.2 Writing the Root Filesystem into the Target's Flash

In this section you will find a description on how to write the newly created root filesystem into the phyFLEX-i.MX6's flash memory. Before the images can be written into the flash, it must be uploaded to the phyFLEX-i.MX6 from a TFTP server. This will be done from the command line of the boot loader. The partition of the root filesystem will be formatted. Then the image will be downloaded over TFTP to the created flash partition.



You should never erase the *Barebox* partition. If this partition is erased, you will not be able to start your target anymore. In such a case, refer to our BSP Manual.



The versions of the *Barebox* and the Linux kernel must match. Therefore the following steps should only be done using the hardware that was shipped together with the virtual machine and thus already contains the same version of the BSP.

- First open a new terminal window if it is not opened yet. Then change to the directory `/opt/PHYTEC_BSPs/phyFLEX-i.MX6/build/deploy/images/phyflex-imx6-2/`



Terminal

- Copy** the root filesystem to the folder `/tftpboot`:  
`cp phytec-qt5demo-image-phyflex-imx6-2.ubifs /tftpboot/`



Microcom

- Open *Microcom* and press the **RESET** button on the target. You will see the output **Hit any key to stop autoboot**.
- Press **any key** to stop *autoboot*.

```
Microcom_ttyUSB0

barebox 2015.02.0-iMX6-PD15.1.1-dirty #1 Tue Sep 15 12:19:30 CEST 2015

Board: Phytec phyFLEX-i.MX6 Quad Carrier-Board
detected i.MX6 Quad revision 1.2
mdio_bus: miibus0: probed
eth0: got preset MAC address: 50:2d:f4:06:fe:0d
nand: NAND device: Manufacturer ID: 0xec, Chip ID: 0xd3 (Samsung NAND 1GiB 3,3V
8-bit), 1024MiB, page size: 2048, OOB size: 64
Bad block table found at page 524224, version 0x01
Bad block table found at page 524160, version 0x01
m25p80 m25p80@00: n25q128a13 (16384 Kbytes)
imx-esdhc 2194000.usdhc: registered as 2194000.usdhc
imx-esdhc 2198000.usdhc: registered as 2198000.usdhc
imx-ipuv3 2400000.ipu: IPUv3H probed
imx-ipuv3 2800000.ipu: IPUv3H probed
netconsole: registered as netconsole-1
malloc space: 0x2ff00000 -> 0x4fdfffff (size 511 MiB)
mmc2: detected SD card version 2.0
mmc2: registered mmc2
barebox-environment environment-sd.11: setting default environment path to /dev/
mmc2.barebox-environment
envfs: wrong magic
running /env/bin/init...

Hit m for menu or any other key to stop autoboot:  2

type exit to get to the menu
barebox@Phytec phyFLEX-i.MX6 Quad Carrier-Board:/
```

- Check the network settings with following command:

```
ifup eth0
devinfo eth0
```

The target should return these lines:

```
ipaddr=192.168.3.11
netmask=255.255.255.0
gateway=255.255.255.0
serverip=192.168.3.10
```

If you need to change something, type:

```
edit /env/network/eth0
```

edit the settings, save them by leaving the editor with **Strg-D**, then type `saveenv` and reboot the board.

- Now we flash the root filesystem. Type:

```
ubiformat /dev/nand0.root
ubiattach /dev/nand0.root
ubimkvol /dev/ubi0 root 0
cp /mnt/tftp/phytec-qt5demo-image-phyflex-imx6-2.ubifs /dev/ubi0.root
```

- Enter `boot` to boot the phyFLEX-i.MX6 with the new kernel and root file system.
- After the target has successfully finished booting, type `root` to log in.



- Now we can test the newly installed editor *nano* by trying to open a file with it.
- Enter `nano /etc/profile` .
- Close *nano* by pressing **CTRL+X**.
- Close *Microcom* after *nano* is closed.

**Troubleshooting:**

If any problem occurs after writing the kernel or the root filesystem into the flash memory, you can restore the original kernel (*zimage*) from the `/tftpboot` directory. The root file system is available in the following directory of the virtual machine:

`/opt/PHYTEC_BSPs/phyFLEX-iMX6/build/deploy/images/phyflex-imx6-2/phytec-qt5demo-image-phyflex-imx6-2-20150915092725.rootfs.ubifs`



All files are also downloadable from our ftp server. If you want other versions also check our ftp server:

<ftp://ftp.phytec.de/pub/Products/>



For more information about how to update the software on our target, please refer to our BSP Manual.



In this section you learned how to prepare the partition of the root filesystem and how to download the root filesystem from a TFTP server into the flash of the target.

## 6 Setup your own Linux-Host-PC

This chapter is for developers who want to use their own existing environment and not our modified Ubuntu version. It is not needed if you are satisfied with our virtual machine.

We will give an overview of the modifications which we made to the Ubuntu version on the phyFLEX-i.MX6 USB flash drive in comparison to the original Ubuntu in this chapter. In the following we distinguish between optional and essential modifications. So you can see faster which changes are important to execute this Quickstart in case you do not want to use our modified Ubuntu version. You can find a step-by-step instruction of the essential changes in order to modify your own distribution.



We can not guarantee that the presented changes are compatible to other distributions or versions. If you want to use another distribution, it might take a lot of individual initiative. We do not support other distributions. You should be sure about what you do.

### 6.1 Essential Settings

In the following section you get a short instruction about the important settings which are essential to guarantee the execution of the examples in this Application Guide.

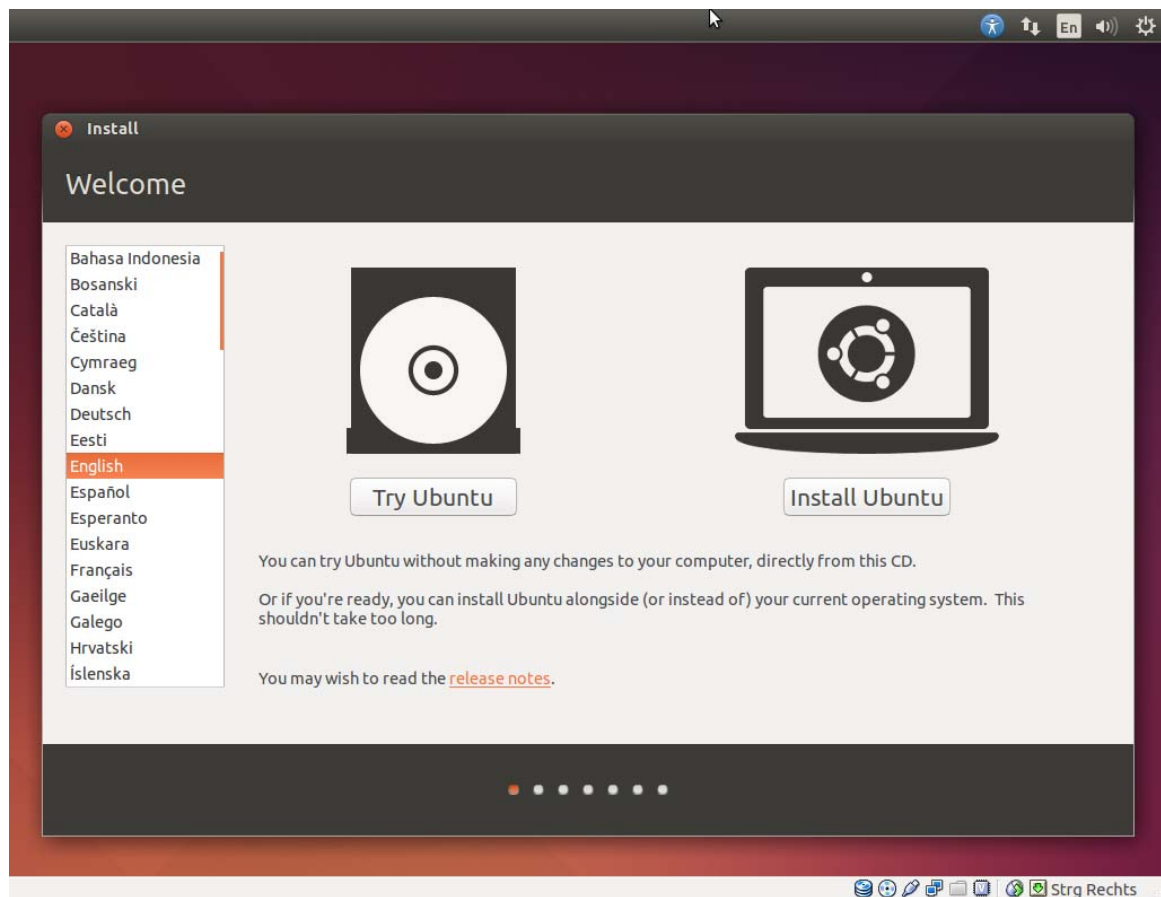
#### 6.1.1 Installing Ubuntu



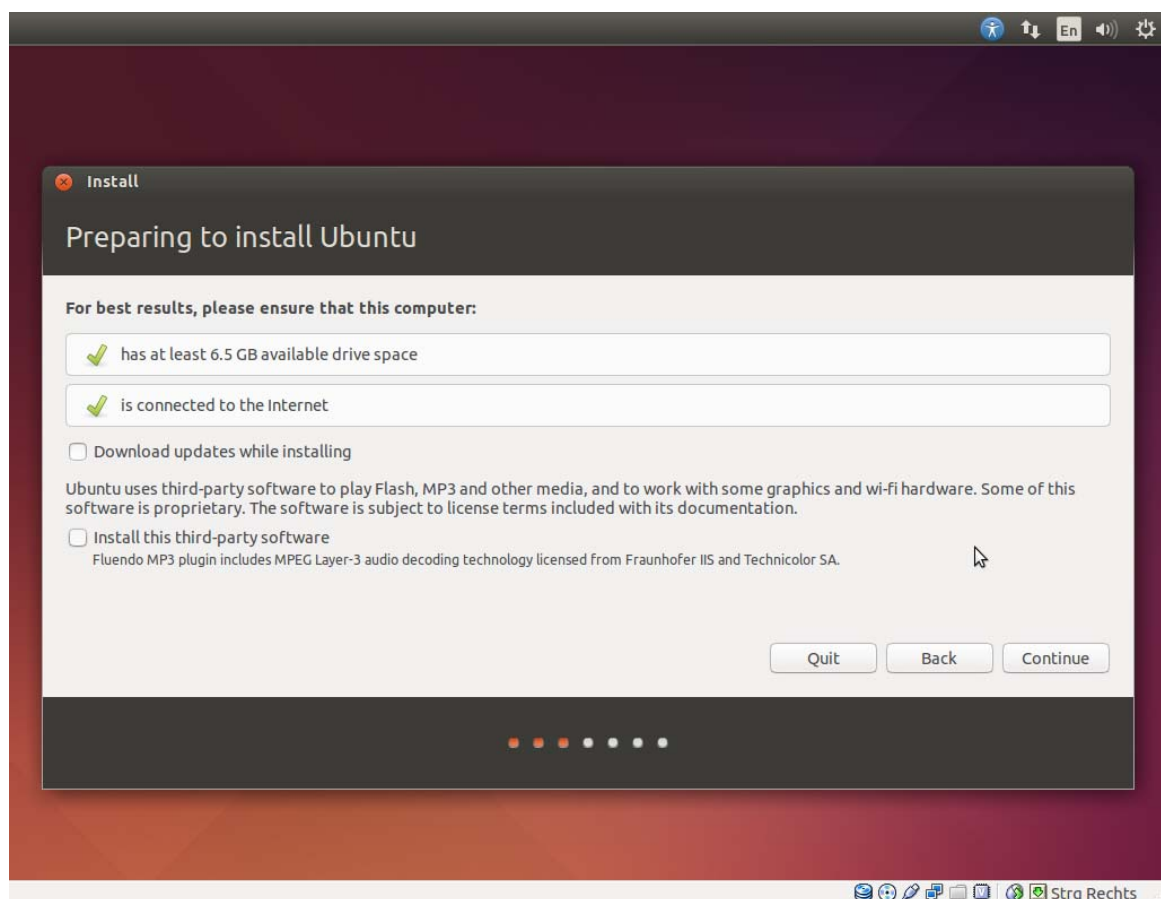
We recommend to have at least 80 GB of free disk space available.

- Download *Ubuntu 14.04.3 LTS* from [www.ubuntu.com](http://www.ubuntu.com) and create a bootable medium.
- Boot *Ubuntu* from your created boot medium.
- Select ***Install Ubuntu*** in the first *Welcome* window.





- The *Preparing to install Ubuntu...* window appears. From *Ubuntu* it is advised that you select **Download updates while installing** and **Install this third-party software now**.



- Click **Continue**.
- The *Installation type* window appears. You now have different options how to install *Ubuntu*. Depending on your system you have a number of possibilities that are shown in the dialog. After you have chosen one click **Continue**.
- The *Install Ubuntu...* window appears. After you have checked the settings you can click **Install now**.
- While the installation is started *Ubuntu* asks for your location, keyboard layout and login and password details. Please insert this information and wait until the installation is finished.
- Finally you must restart your system after the installation is finished.
- After that the system boots up and you can log into *Ubuntu*. Please configure your network connection now and connect the phyFLEX-i.MX6

### 6.1.2 Installation of Software Packages

First of all various software packages that are required must be installed using the package manager *APT*.

To gain a better understanding of the packages required, they are installed separately according to their function:

1. Packages which are needed for compiling and building the Board Support Package:  
`sudo apt-get -y install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev xterm`
2. Packages for development  
`sudo apt-get -y install vim eclipse`
3. Packages to set up the TFTP server:  
`sudo apt-get -y install tftpd-hpa`

During installation some programs may ask for a license agreement. Just go through the steps shown on the screen.

The first preparations are completed. In the next sections you will proceed with building the Board Support Package installation of *Eclipse* and *Qt Creator* and the set up of the TFTP server.

### 6.1.3 Set the Git Configuration

The Board Support Package is heavily based on *Git*. *Git* needs some information from you as a user to be able to identify which changes were done by whom. So at least set the *name* and *email* in your git configuration, otherwise building the BSP generates many warnings.

- Enter following two commands to directly set them

```
git config --global user.email "your_email@example.com"
git config --global user.name "name surname"
```

### 6.1.4 Build the Board Support Package and Install the SDK

In this chapter we build our Board Support Package with the help of the *Yocto Project*.

First we create a folder for the BSP in our example we use the path as it is in our virtual machine

- Enter the following commands:

```
sudo mkdir -p /opt/PHYTEC_BSPs/phyFLEX-i.MX6/
sudo chmod -R 777 /opt/PHYTEC_BSPs/
cd /opt/PHYTEC_BSPs/phyFLEX-i.MX6/
```
- Download the *phyLinux* script and set execution privileges:

```
wget ftp://ftp.phytec.de/pub/Software/Linux/Yocto/Tools/phyLinux
chmod +x ./phyLinux
```

The *phyLinux* script is a basic management tool for PHYTEC Yocto BSP releases. It is mainly a tool to get started with the BSP structure. You can get all the BSP sources without the need of interacting with repo or git.

- Start the phyLinux script

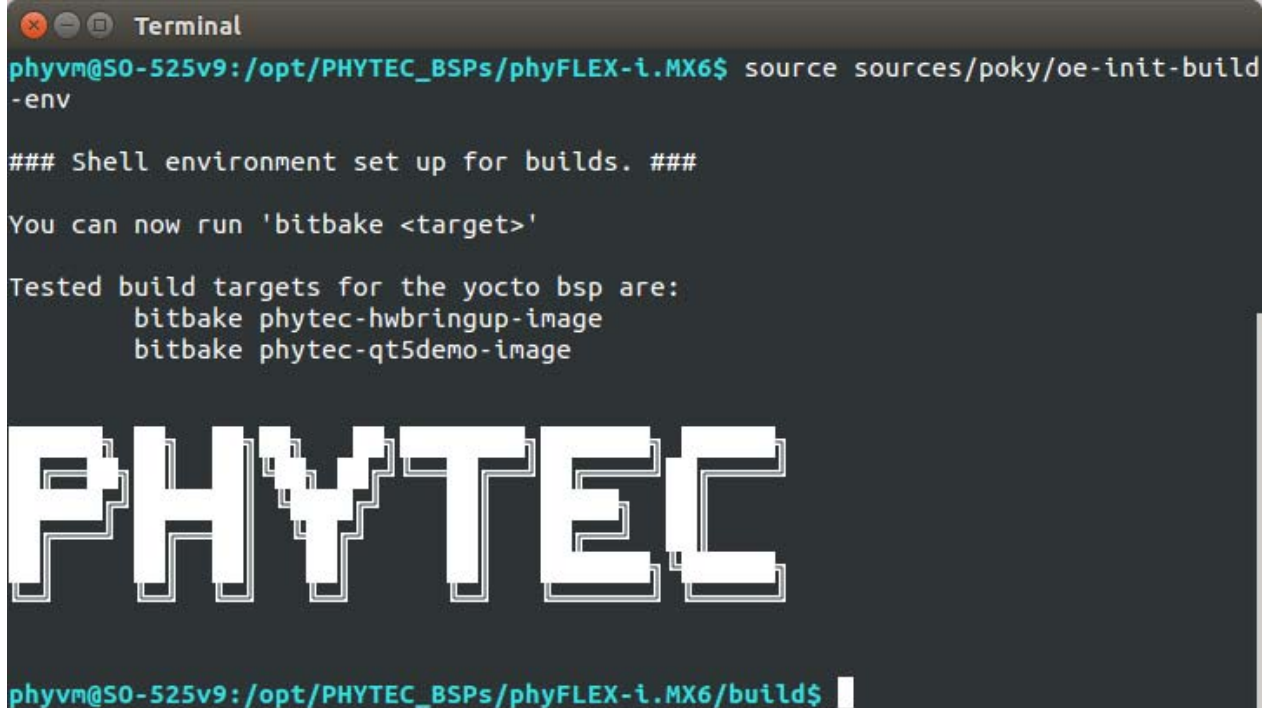
```
./phyLinux init
```

During the execution of the init command, you need to choose your processor platform (iMX6), PHYTEC's BSP release number (PD15.1.1) and the hardware you are working on (phyflex-imx6-2 ).

After you downloaded all the meta data with *phyLinux*, you have to set up the shell environment variables. This needs to be done every time you open a new shell for starting builds. We use the shell script provided by poky in its default configuration.

- Type:

```
source sources/poky/oe-init-build-env
```



```
Terminal
phyvm@S0-525v9:/opt/PHYTEC_BSPs/phyFLEX-i.MX6$ source sources/poky/oe-init-build-env

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

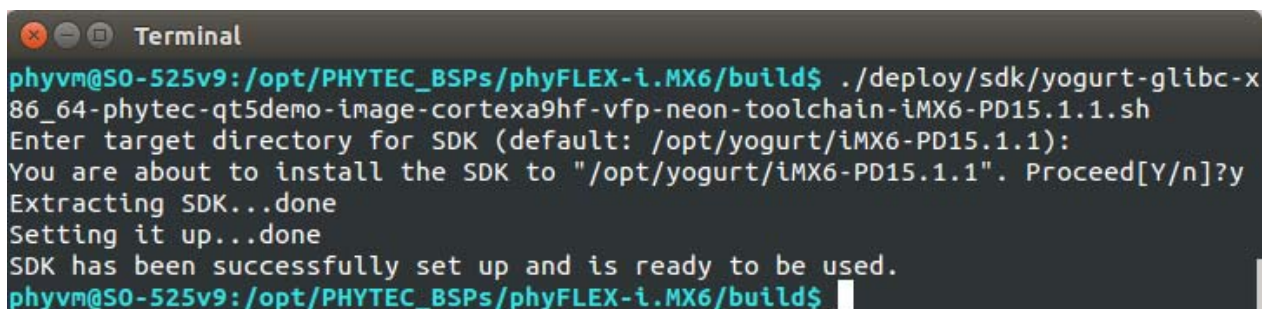
Tested build targets for the yocto bsp are:
    bitbake phytec-hwbringup-image
    bitbake phytec-qt5demo-image

PHYTEC

phyvm@S0-525v9:/opt/PHYTEC_BSPs/phyFLEX-i.MX6/build$
```

The current working directory of the shell should change to build/ and you are now ready to build your first images.

- Build the BSP:  
`bitbake phytec-qt5demo-image`
- Generate the SDK needed for *Eclipse* and *Qt Creator*:  
`bitbake phytec-qt5demo-image -c populate_sdk`
- Install the SDK into the recommended default directory:  
`./deploy/sdk/yogurt-glibc-*.sh`



```
Terminal
phyvm@S0-525v9:/opt/PHYTEC_BSPs/phyFLEX-i.MX6/build$ ./deploy/sdk/yogurt-glibc-x86_64-phytec-qt5demo-image-cortexa9hf-vfp-neon-toolchain-IMX6-PD15.1.1.sh
Enter target directory for SDK (default: /opt/yogurt/IMX6-PD15.1.1):
You are about to install the SDK to "/opt/yogurt/IMX6-PD15.1.1". Proceed[Y/n]?y
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
phyvm@S0-525v9:/opt/PHYTEC_BSPs/phyFLEX-i.MX6/build$
```

### 6.1.5 Set up Eclipse and Integrate Plug-ins

The instructions in this chapter show how to setup *Eclipse* and integrate the C/C++ plug-in. Thus you will be able to assign your own programs, written in *Eclipse*, to the target.

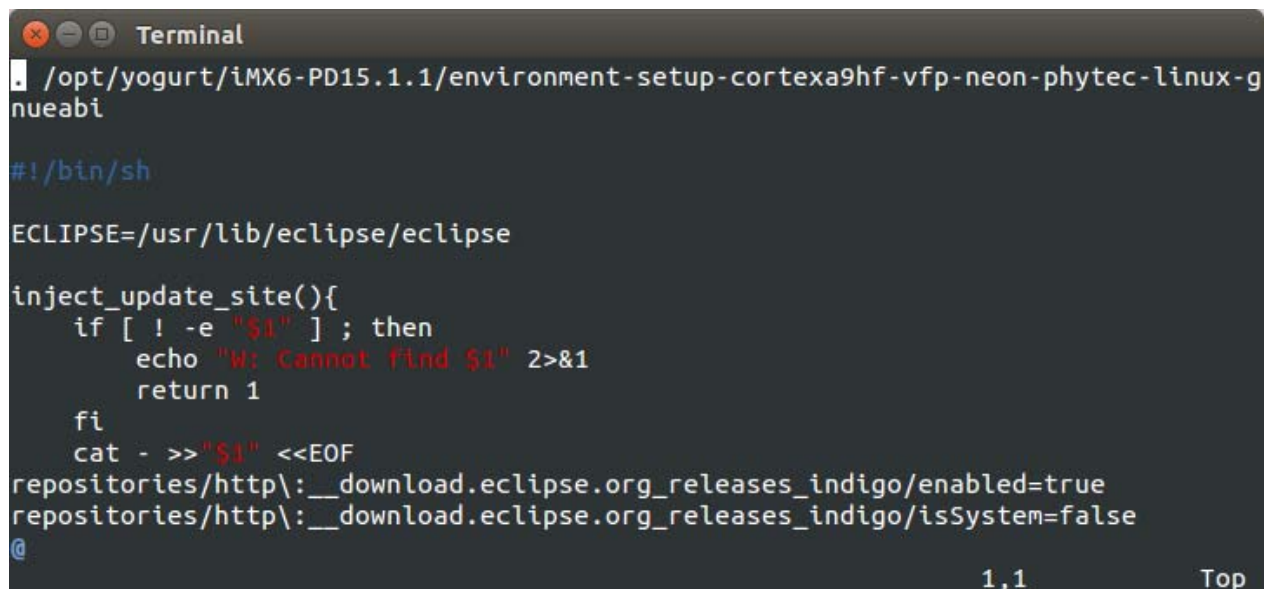
- First of all you have to create a workspace folder, in which you can save your Eclipse projects.

In this example we create the workspace in the `/opt/` directory as it is in the virtual machine:

```
mkdir -p /opt/prj_workspace/Eclipse
sudo chmod -R 777 /opt/prj_workspace/
```

- Before we start *Eclipse* we must set the correct environment to our SDK:

```
sudo vi /usr/bin/eclipse
```



```
Terminal
/opt/yogurt/iMX6-PD15.1.1/environment-setup-cortexa9hf-vfp-neon-phytec-linux-gnueabi

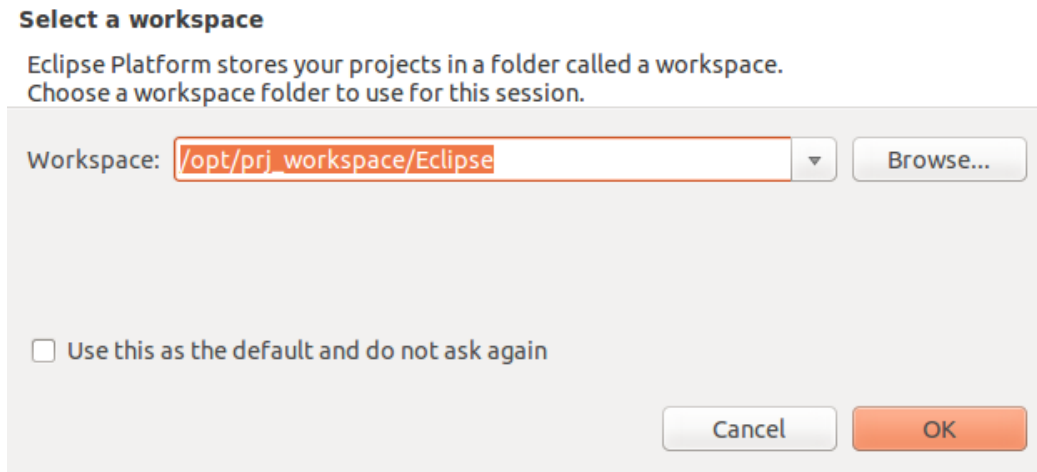
#!/bin/sh

ECLIPSE=/usr/lib/eclipse/eclipse

inject_update_site(){
    if [ ! -e "$1" ] ; then
        echo "W: Cannot find $1" 2>&1
        return 1
    fi
    cat - >>"$1" <<EOF
repositories/http\:__download.eclipse.org_releases_indigo/enabled=true
repositories/http\:__download.eclipse.org_releases_indigo/isSystem=false
@

1,1 Top
```

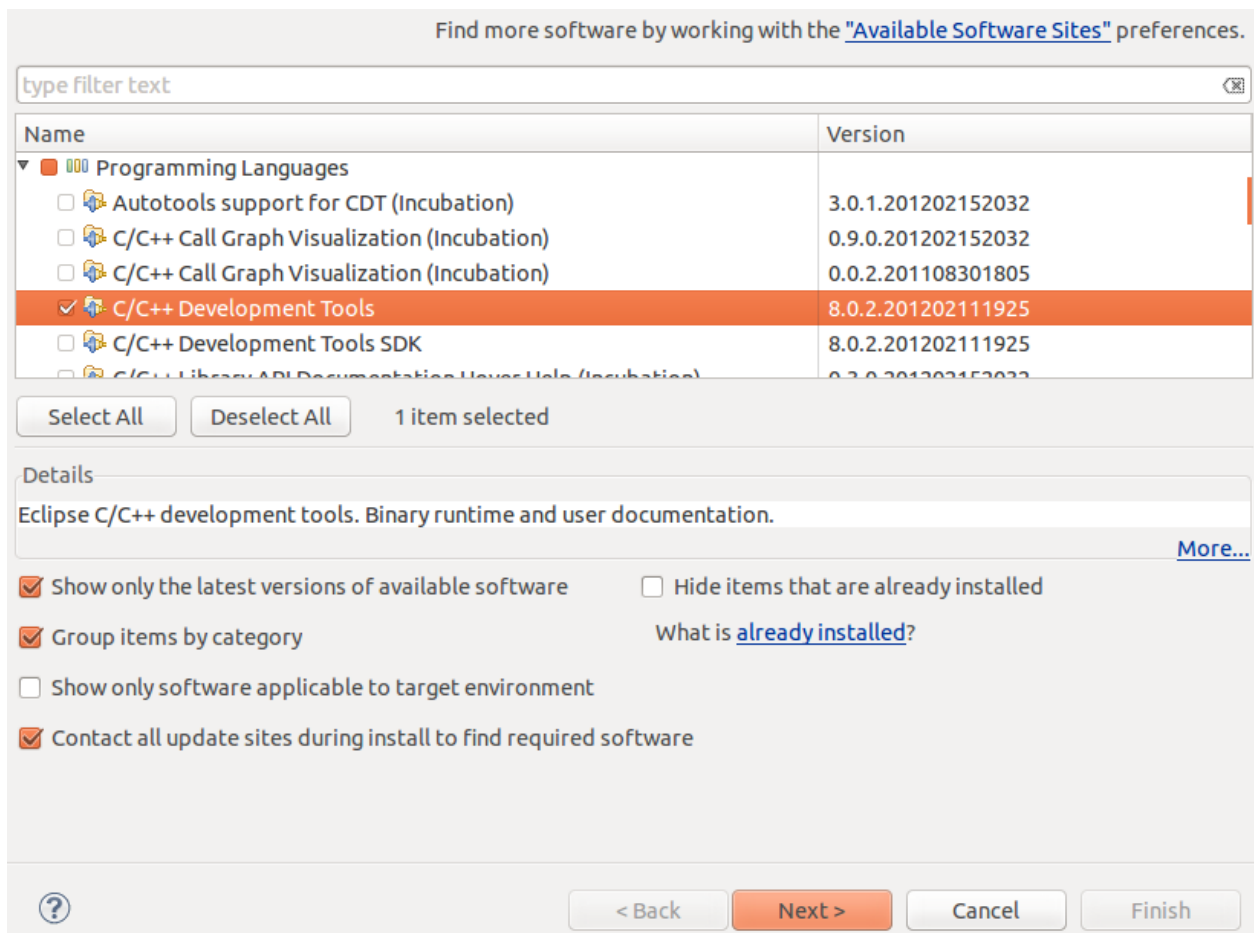
- Enter following line before `#!/bin/sh`:  
`. /opt/yogurt/iMX6-PD15.1.1/environment-setup-cortexa9hf-vfp-neon-phytec-linux-gnueabi`
- Afterwards open *Eclipse* and insert the path to the created workspace in the pop-up window:  
`eclipse`
- Enter the path to the created workspace:  
`/opt/prj_workspace/Eclipse`



- After *Eclipse* has started click **Help** in the menu bar and then **Install new Software**.

A window opens to add a plug-in to *Eclipse*.

- In the drop-down menu *Work with* select **Indigo** <http://download.eclipse.org/releases/indigo>
- The available plug-ins appear in the selection area below now. Expand **Programming Languages** and check **C/C++ Development Tools**. Click **Next**.



- Now the system displays an overview of the installation details. Click **Next** to proceed.



- Finally accept the licensing agreement and click **Finish** to start the installation of the required software.
- After that start *Eclipse* with the option clean, which cleans any cached data:  
`eclipse --clean`



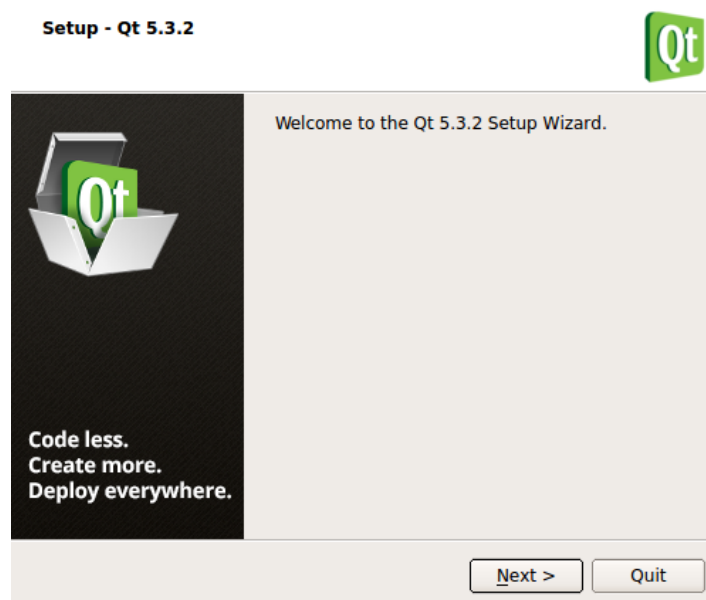
Congratulations! You have successfully integrated the *CDT* plug-in in *Eclipse* and now you can start programming in C/C++. In the next section you will find a short introduction on how to install and setup the *Qt Creator*.

### 6.1.6 Install and Setup *Qt Creator*

Because we want the same qmake version as in our BSP we install *Qt Creator* manually.

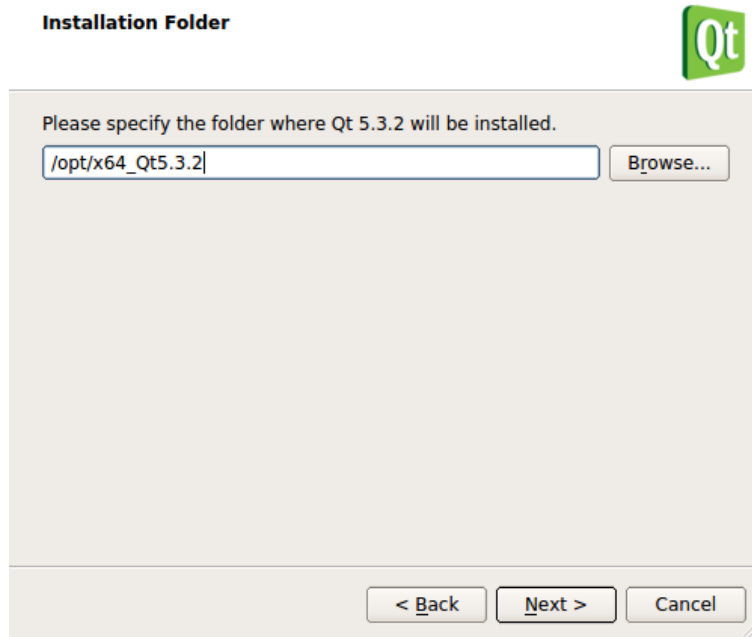
- Download *Qt Creator* with following command:  
`wget http://download.qt.io/official_releases/qt/5.3/5.3.2/qt-opensource-linux-x64-5.3.2.run`
- Set execute privileges:  
`sudo chmod +x qt-opensource-linux-x64-5.3.2.run`
- Run the *Qt Creator* installation routine  
`./qt-opensource-linux-x64-5.3.2.run`

This opens a window for the installation.

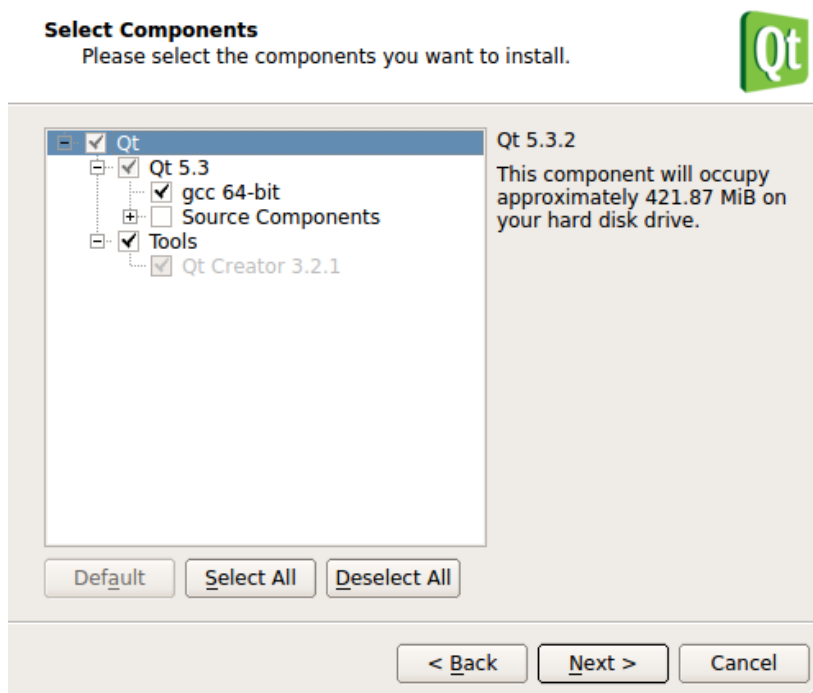


- Step through the installation and if you are asked set following installation directory:  
`/opt/x64_Qt5.3.2`





- In the *Select Components* window keep the default selection of the components and click **Next**.



- Proceed through the license agreement and start the installation.
- After the installation is finished make a symbolic link to easily start *Qt Creator*:  

```
sudo ln -s /opt/x64_Qt5.3.2/Tools/QtCreator/bin/qtcreator.sh /usr/bin/qtcreator.sh
```
- Set the correct environment to our SDK:  

```
sudo vi /usr/bin/qtcreator.sh
```
- Enter following line before `#!/bin/sh`:  

```
. /opt/yogurt/iMX6-PD15.1.1/environment-setup-cortexa9hf-vfp-neon-phytec-linux-gnueabi
```



```

/opt/yogurt/iMX6-PD15.1.1/environment-setup-cortexa9hf-vfp-neon-phytec-linux-g
nueabi

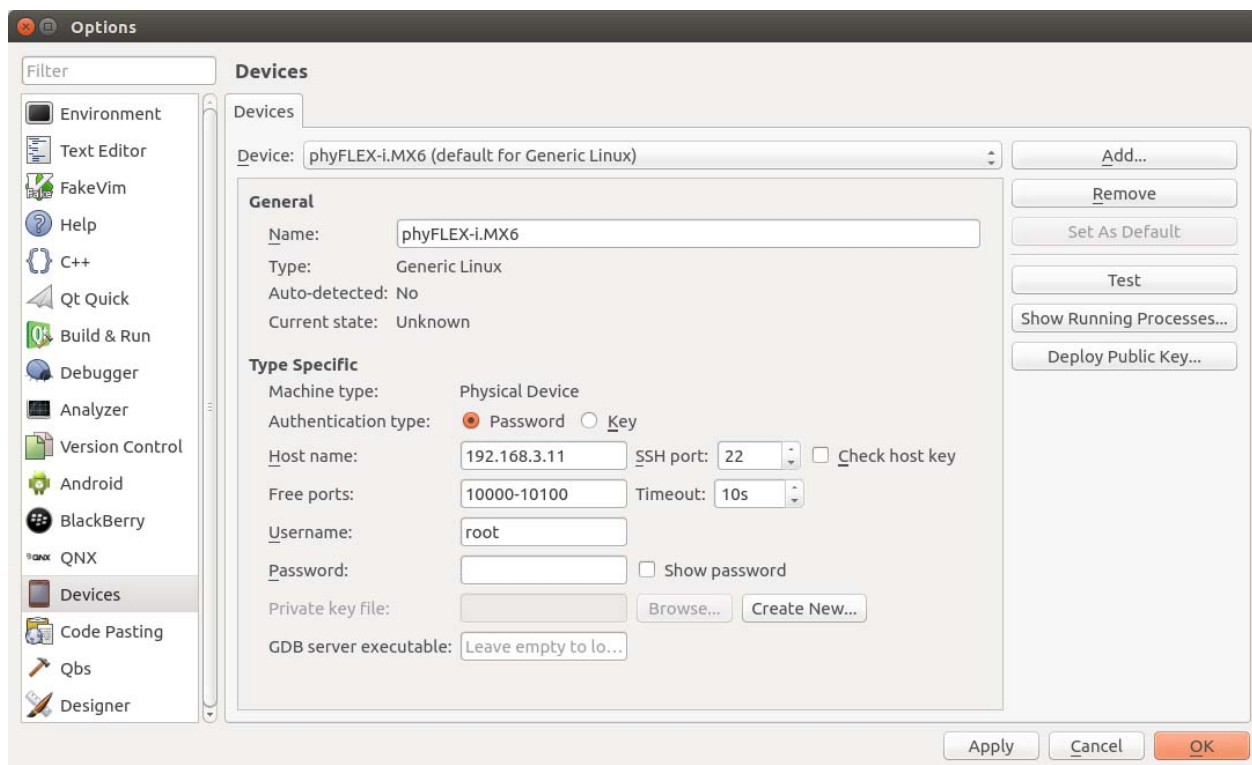
#!/bin/sh

makeAbsolute() {
    case $1 in
        /*)
            # already absolute, return it
            echo "$1"
            ;;
        *)
            # relative, prepend $2 made absolute
            echo `makeAbsolute "$2" "$PWD" `/"$1" | sed 's,/./,./,'
            ;;
    esac
}

"/usr/bin/qtcreator.sh" 37L, 1038C
1,1 Top

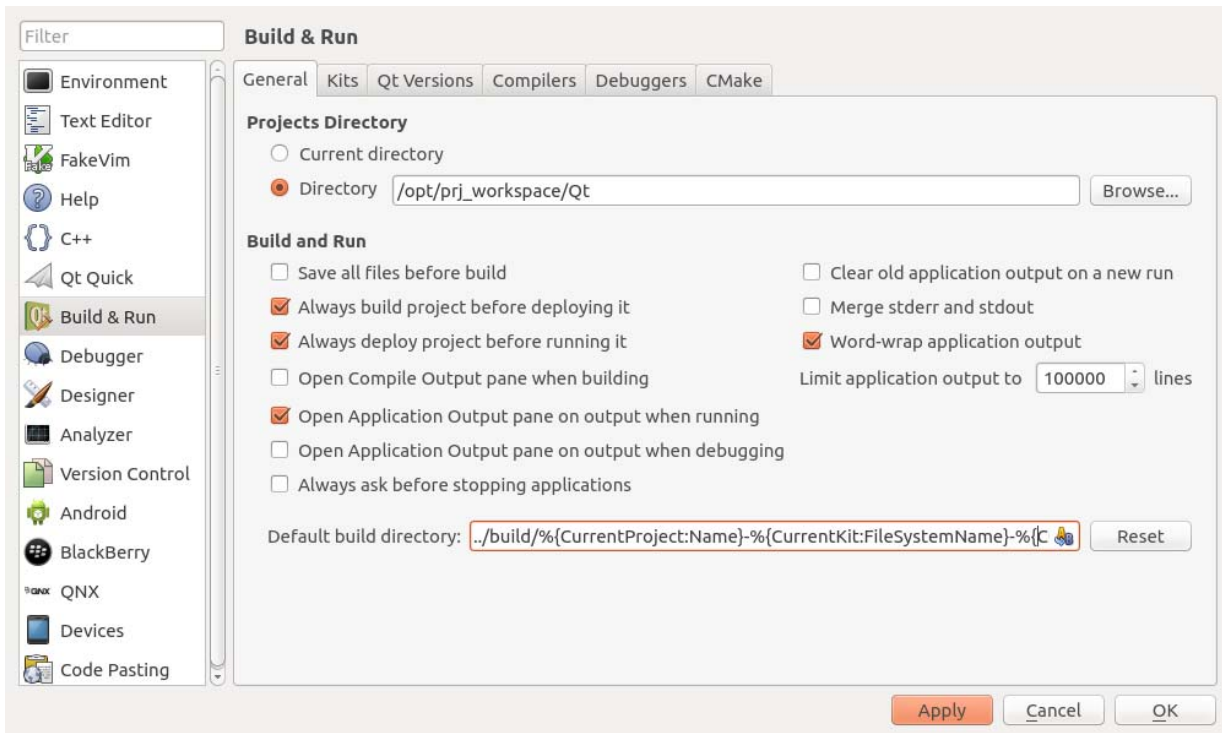
```

- Create a workspace for *QtCreator* e.g.  
`mkdir -p /opt/prj_workspace/Qt/build`
- Start *Qt Creator* with the following command in a terminal:  
`/usr/bin/qtcreator.sh`
- Add the phyFLEX-i.MX6 as device with following inputs by opening **Tools --> Options --> Devices --> Add --> Generic Linux Device:**  
 Name: `phyFLEX-i.MX6`  
 IP: `192.168.3.11`  
 Username: `root`  
 Keep the *password* empty



- Set the project directory and build directory  
**Tools --> Options --> Build & Run --> General.**  
*Projects Directory --> Directory: /opt/prj\_workspace/Qt*

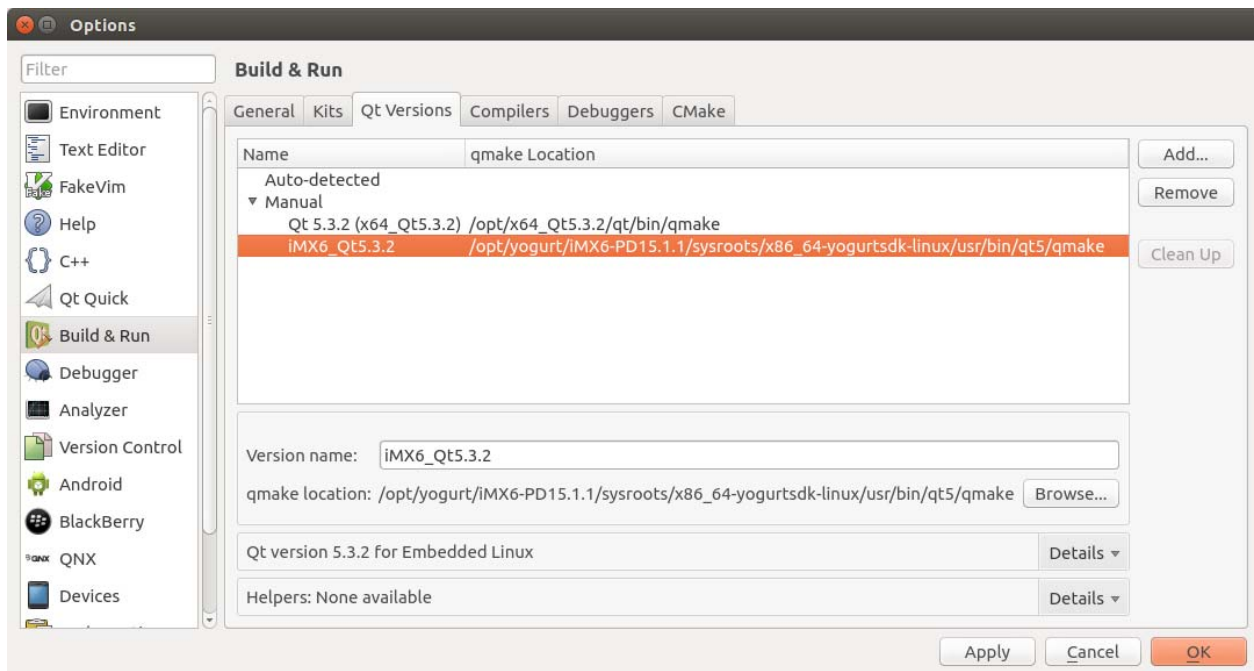
Change the beginning of the entry in the *Default build directory* field:  
`../build/%{CurrentProject:Name}`



- Select tab **Qt Versions** and click **Add...** to add a new one.

Name: `iMX6_Qt5.3.2`

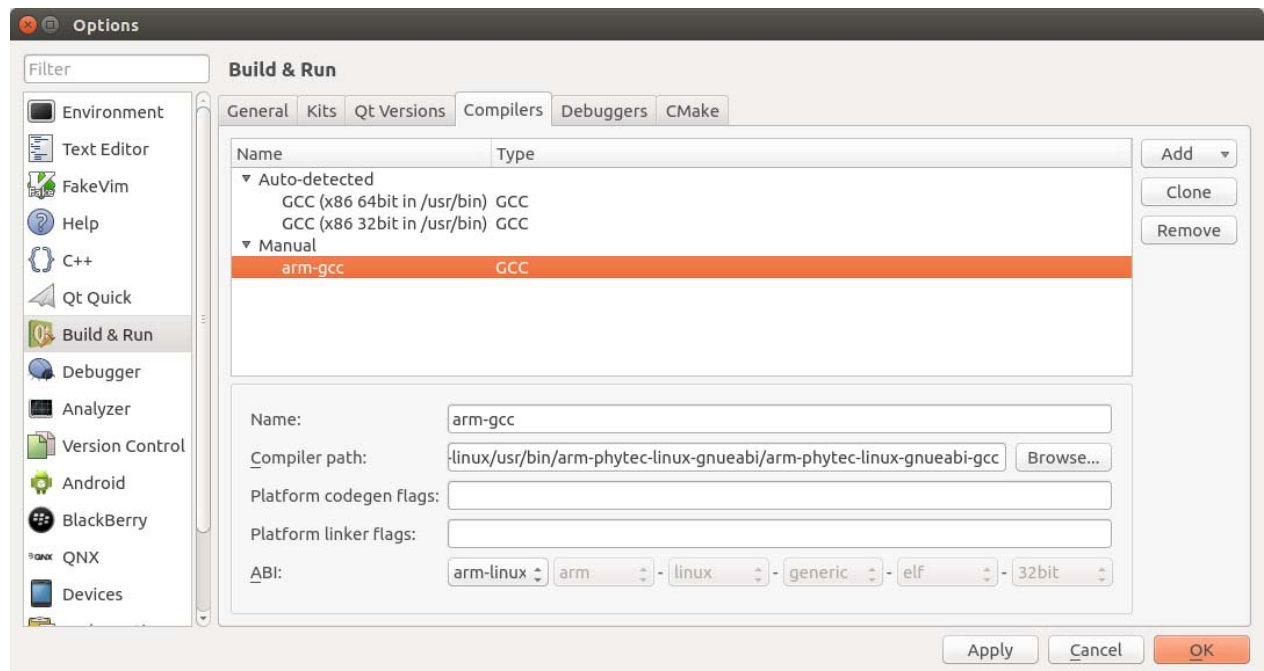
qmake Location: `/opt/yogurt/iMX6-PD15.1.1/sysroots/x86_64-yogurtsdk-linux/usr/bin/qt5/qmake`



- Select tab **Compilers** and click on **Add** select **GCC**

Name: `arm-gcc`

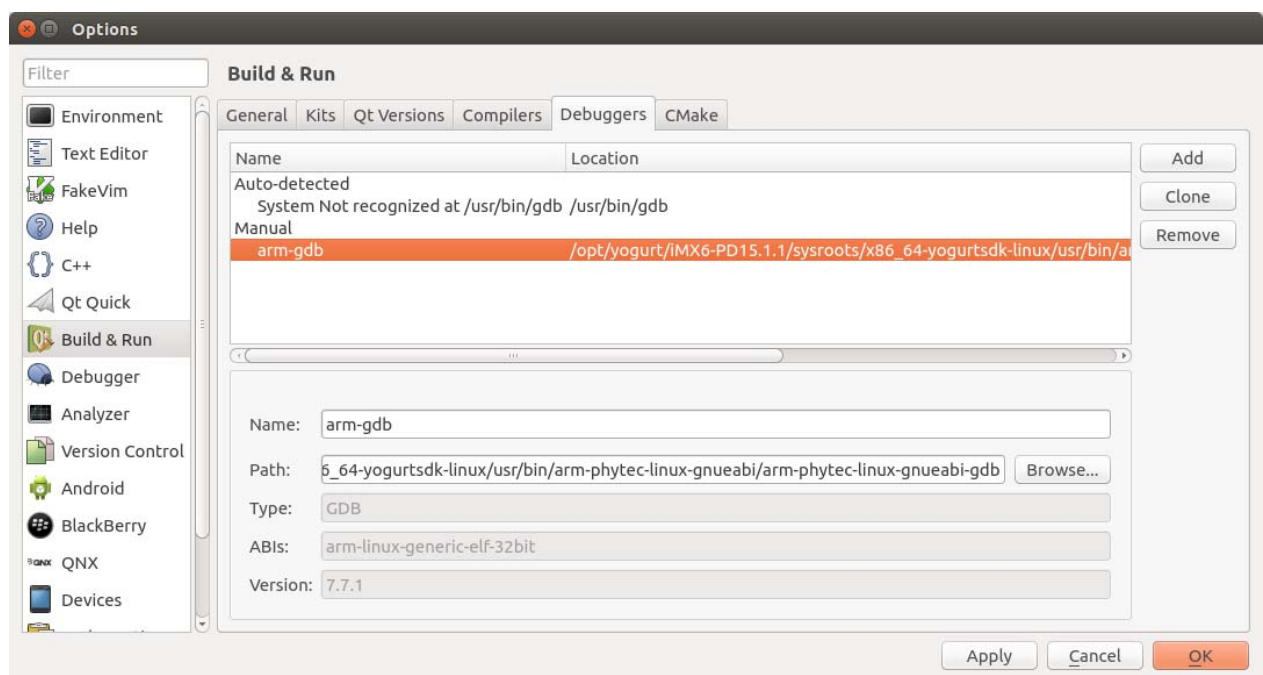
Compiler Path: `/opt/yogurt/iMX6-PD15.1.1/sysroots/x86_64-yogurtsdk-linux/usr/bin/arm-phytec-linux-gnueabi/arm-phytec-linux-gnueabi-gcc`



- Select tab **Debuggers** and click on **Add**

Name: `arm-gdb`

Path: `/opt/yogurt/iMX6-PD15.1.1/sysroots/x86_64-yogurtsdk-linux/usr/bin/arm-phytec-linux-gnueabi/arm-phytec-linux-gnueabi-gdb`



- Select tab **Kit** and click on **Adds**

Name: phyFLEX-i.MX6

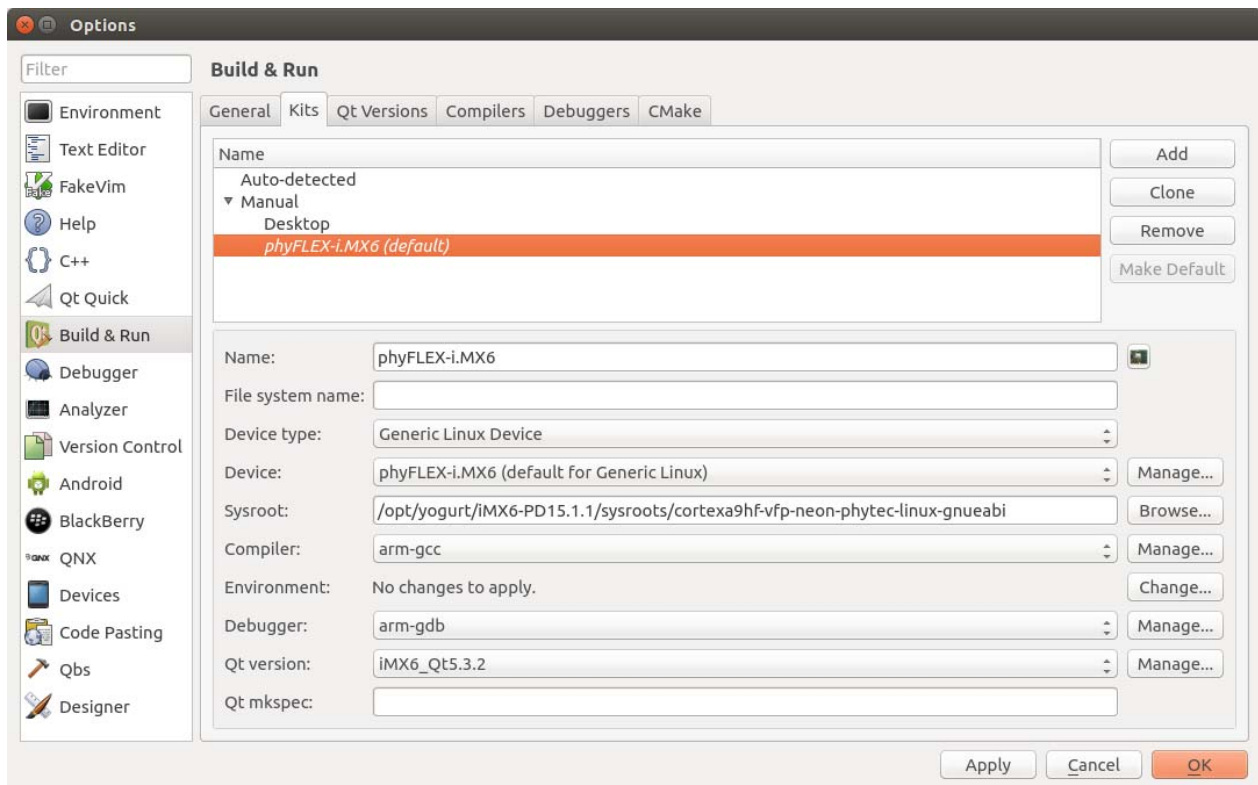
Device Type: Generic Linux Device

Sysroot: /opt/yogurt/iMX6-PD15.1.1/sysroots/cortexa9hf-vfp-neon-phytec-linux-gnueabi

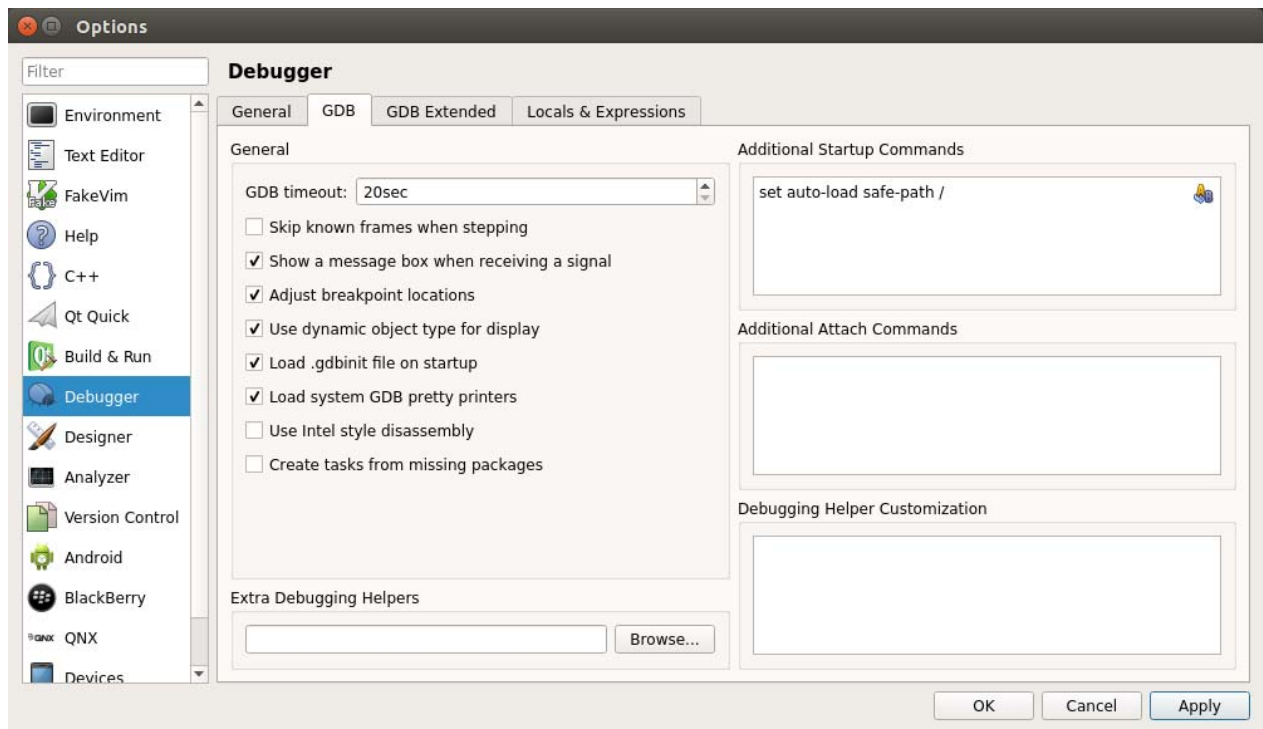
Compiler: arm-gcc

Debugger: arm-gdb

Qt version: iMX6\_Qt5.3.2



- Select **Debugger** on the left side of the window and click on the **GDB** tab.
- Enter following command in *the Addition Startup Commands*:  
`set auto-load safe-path /`



- Click on **Apply** and close the window with **OK**.

Now all configurations are done and you can use this Kit to cross compile your application also if you generate a new project.

### 6.1.7 Setting up a TFTP server

In the chapter *"Installation of software packages"* you have installed the required packages to set up a TFTP server. Now we have to change some short settings.


- First change the file `/etc/default/tftp-hpa` as follows:  


```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```
- Then create a folder called `/tftpboot`. The TFTP server will access this folder later.  

```
mkdir /tftpboot
```
- Finally we have to set the right permissions:  

```
chmod 777 /tftpboot
```




	<p>Due to a bug in the current version of the TFTP-hpa daemon the daemon does not start after the system is rebooted. You have to start it in the terminal with the following command:</p> <pre>restart tftpd-hpa</pre> <p>A permanent workaround is to create a file called <i>tftp-hpa</i> under <i>/etc/network/if-up.d/</i>. Insert the following commands in the file:</p> <pre>#!/bin/sh restart tftpd-hpa</pre> <p>After saving the file set the correct permission with</p> <pre>chmod 755 /etc/network/if-up.d/tftpd-hpa .</pre>
---	---

	<p>You have successfully set up the TFTP server. In the future the phyFLEX-i.MX6 can access to the <i>/tftpboot/</i> folder to load the images.</p>
---	---

### 6.1.8 Optional Settings

In the following section we list the optional settings. These settings are not needed for a successful operation of this Application Guide. They only simplify the handling and the look of the system. For this reason we show the modifications without big explanation.

- *vim*, the improved *vi* editor is installed.
- Desktop-icons for faster and easier start of required programs are created.
- The following modifications were made to the look of *Ubuntu*:
  - Other wallpaper and associated options are adjusted with the help of **gsettings**
  - The color of the *Gnome* terminal is changed.
  - The scrolling log for the *Gnome* terminal is changed.
- **History-search-backward** and **history-search-forward** in */etc/inputrc* is activated. This allows you to search through your history with the entered string.
- Also there are some scripts that will be executed at the first start after the installation. These scripts ensure that the right permissions are set for the created user.

	<p>Congratulations! You have successfully configured your <i>Ubuntu</i> to work with.</p>
---	---



## Revision History

Date	Version #	Changes in this manual
21.09.2012	Quickstart L-781e_1	First draft
10.10.2013	L-781e_4	Update Quickstart for working with Ubuntu 12.04 Update Quickstart for working with PD13.1.0 New barebox handling Fixed some issues
29.01.2014	L-781e_5	Update Quickstart for working with PD13.2.2 Fixed some issues
11.03.2014	L-781e_6	Change description for updating the bootloader
17.09.2015	L-781e_7	Major alteration Change LiveDVD to virtual machine hard disk image Change ptxdist to Yocto Update for working with Ubuntu 14.04.3 Update for working with PD15.1.1 Update for working with S0-525v9 Add Programming with Qt Creator Remove chapter Updating the Software (now in BSP Manual)

**Document:** phyFLEX-i.MX6  
**Document number:** L-7816e\_7, September 2015

**How would you improve this manual?**

---

---

---

---

**Did you find any mistakes in this manual?** **page**

---

---

---

---

**Submitted by:**

Customer number: 

---

Name: 

---

Company: 

---

Address: 

---

---

**Return to:**

PHYTEC Messtechnik GmbH  
Postfach 100403  
D-55135 Mainz, Germany  
Fax : +49 (6131) 9221-33

