phyBOARD®-Wega-AM335x

Application Guide

Document No.:	**L-792e_0**
SBC Prod. No..:	**PB-00802-xxx**
CB PCB No.:	**1405.0**
SOM PCB No.:	**1397.0**

Preliminary

Edition: November 2013

	EUROPE	NORTH AMERICA
Address:	PHYTEC Messtechnik GmbH Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Ordering Information:	+49 (6131) 9221-32 sales@phytec.de	1 (800) 278-9913 sales@phytec.com
Technical Support:	+49 (6131) 9221-31 support@phytec.de	1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	1 (206) 780-9135
Web Site:	http://www.phytec.de http://www.phytec.eu	http://www.phytec.com

Preliminary Edition November 2013

List of Figures

List of Tables

Conventions, Abbreviations and Acronyms

This hardware manual describes the PB-00802-xxx Singe Board Computer (SBC) in the following referred to as phyBOARD-Wega-AM335x. The manual specifies the phyBOARD-Wega-AM335x's design and function. Precise specifications for the Texas Instruments AM335x microcontrollers can be found in the Texas Instrumenten's AM335x Data Sheet and Technical Reference Manual.

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.: $\overline{\text{RD}}$) 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^{2}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.

Abbreviations and Acronyms

Many acronyms and abbreviations are used throughout this manual. Use the table below to navigate unfamiliar terms used in this document.

Abbreviation	**Definition**
A/V	Audio/Video
BSP	Board Support Package (Software delivered with the Development Kit including an operating system (Windows, or Linux) preinstalled on the module and Development Tools).
CB	Carrier Board; used in reference to the phyBOARD-Wega Development Kit Carrier Board.
DFF	D flip-flop.
DSC	Direct Solder Connect
EMB	External memory bus.
EMI	Electromagnetic Interference.
GPI	General purpose input.
GPIO	General purpose input and output.
GPO	General purpose output.
IRAM	Internal RAM; the internal static RAM on the Texas Instruments AM335x microcontroller.
J	Solder jumper; these types of jumpers require solder equipment to remove and place.
JP	Solderless jumper; these types of jumpers can be removed and placed by hand with no special tools.
NC	Not Connected
PCB	Printed circuit board.
PDI	PHYTEC Display Interface; defined to connect PHYTEC display adapter boards, or custom adapters
PEB	PHYTEC Extension Board
PMIC	Power management IC
PoE	Power over Ethernet
PoP	Package on Package
POR	Power-on reset
RTC	Real-time clock.
SBC	Single Board Computer; used in reference to the PBA-CD-02 /phyBOARD-Wega-AM335x module
SMT	Surface mount technology.
SOM	System on Module; used in reference to the PCL-051 /phyCORE-AM335x module
Sx	User button Sx (e.g. S1, S2) used in reference to the available user buttons, or DIP-Switches on the CB.
Sx_y	Switch y of DIP-Switch Sx; used in reference to the DIP-Switch on the carrier board.
VSTBY	SOM standby voltage input

Table 1: Abbreviations and Acronyms used in this Manual

🪧	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.

Note: The BSP delivered with the phyBOARD-Wega-AM335x usually includes drivers and/or software for controlling all components such as interfaces, memory, etc. Therefore programming close to hardware at register level is not necessary in most cases. For this reason, this manual contains no detailed description of the controller's registers. Please refer to the AM335x Technical Reference Manual, if such information is needed to connect customer designed applications.

Preface

As a member of PHYTEC's new phyBOARD® product family the phyBOARD-Wega-AM335x is one of a series of PHYTEC System on Modules (SBCs) that offers various functions and configurations. PHYTEC supports a variety of 8-/16- and 32-bit controllers in two ways:

(1) as the basis for Rapid Development Kits which serve as a reference and evaluation platform

(2) as insert-ready, fully functional phyBOARD® OEM modules, which can be embedded directly into the user's peripheral hardware design.

Implementation of an OEM-able SBC subassembly as the "core" of your embedded design allows you to focus on hardware peripherals and firmware without expending resources to "re-invent" microcontroller circuitry. Furthermore, much of the value of the phyBOARD® SBC lies in its layout and test.

PHYTEC's new phyBOARD® product family consists of a series of extremely compact embedded control engines featuring various processing performance classes.

Production-ready Board Support Packages (BSPs) and Design Services for our hardware will further reduce your development time and risk and allow you to focus on your product expertise. Take advantage of PHYTEC products to shorten time-to-market, reduce development costs, and avoid substantial design issues and risks. With this new innovative full system solution you will be able to bring your new ideas to market in the most timely and cost-efficient manner.

For more information go to:

http://www.phytec.de/de/leistungen/entwicklungsunterstuetzung.html or www.phytec.eu/europe/oem-integration/evaluation-start-up.html

Ordering Information

Ordering numbers:

phyBOARD-Wega-AM335x Development Kit:	**KPB-00802-xxx**
phyBOARD-Wega-AM335x SBC:	**PB-00802-xxx**

In order to receive product specific information on changes and updates in the best way also in the future, we recommend to register at

http://www.phytec.de/de/support/registrierung.html or
http://www.phytec.eu/europe/support/registration.html

For technical support and additional information concerning your product, please visit the support section of our web site which provides product specific information, such as errata sheets, application notes, FAQs, etc.

http://www.phytec.de/de/support/faq/faq-phyBOARD-Wega-AM335x.html or
http://www.phytec.eu/europe/support/faq/faq-phyBOARD-Wega-AM335x.html

Declaration of Electro Magnetic Conformity of the PHYTEC phyBOARD-Wega-AM335x

PHYTEC Single Board Computers (henceforth products) are designed for installation in electrical appliances or as dedicated Evaluation Boards (i.e.: for use as a test and prototype platform for hardware/software development) in laboratory environments.

Caution:
PHYTEC products lacking protective enclosures are subject to damage by ESD and, hence, may only be unpacked, handled or operated in environments in which sufficient precautionary measures have been taken in respect to ESD-dangers. It is also necessary that only appropriately trained personnel (such as electricians, technicians and engineers) handle and/or operate these products. Moreover, PHYTEC products should not be operated without protection circuitry if connections to the product's pin header rows are longer than 3 m.

PHYTEC products fulfill the norms of the European Union’s Directive for Electro Magnetic Conformity only in accordance to the descriptions and rules of usage indicated in this hardware manual (particularly in respect to the pin header row connectors, power connector and serial interface to a host-PC).

Implementation of PHYTEC products into target devices, as well as user modifications and extensions of PHYTEC products, is subject to renewed establishment of conformity to, and certification of, Electro Magnetic Directives. Users should ensure conformance following any modifications to the products as well as implementation of the products into target systems.

Product Change Management and information in this manual on parts populated on the SOM / SBC

When buying a PHYTEC SOM / SBC, you will, in addition to our HW and SW offerings, receive a free obsolescence maintenance service for the HW we provide.

Our PCM (Product Change Management) Team of developers, is continuously processing, all incoming PCN's (Product Change Notifications) from vendors and distributors concerning parts which are being used in our products.

Possible impacts to the functionality of our products, due to changes of functionality or obsolesce of a certain part, are being evaluated in order to take the right masseurs in purchasing or within our HW/SW design.

Our general philosophy here is: **We never discontinue a product as long as there is demand for it.**

Therefore we have established a set of methods to fulfill our philosophy:

Avoiding strategies

- Avoid changes by evaluating long-livety of parts during design in phase.
- Ensure availability of equivalent second source parts.
- Stay in close contact with part vendors to be aware of roadmap strategies.

Change management in case of functional changes

- Avoid impacts on product functionality by choosing equivalent replacement parts.
- Avoid impacts on product functionality by compensating changes through HW redesign or backward compatible SW maintenance.
- Provide early change notifications concerning functional relevant changes of our products.

Change management in rare event of an obsolete and non replaceable part

- Ensure long term availability by stocking parts through last time buy management according to product forecasts.
- Offer long term frame contract to customers.

Therefore we refrain from providing detailed part specific information within this manual, which can be subject to continuous changes, due to part maintenance for our products.

In order to receive reliable, up to date and detailed information concerning parts used for our product, please contact our support team through the contact information given within this manual.

1 Introduction

1.1 Hardware Overview

The phyBOARD-Wega for phyCORE-AM335x is a low-cost, feature-rich software development platform supporting the Texas Instruments AM335x microcontroller. Moreover, due to the numerous standard interfaces the phyBOARD-Wega-AM335x can serve as bedrock for your application. At the core of the phyBOARD-Wega is the PCL-051/phyCORE-AM335x System On Module (SOM) in a direct solder form factor, containing the processor, DRAM, NAND Flash, power regulation, supervision, transceivers, and other core functions required to support the AM335x processor. Surrounding the SOM is the PBA-CD-002/phyBOARD-Wega carrier board, adding power input, buttons, connectors, signal breakout, and Ethernet connectivity amongst other things.

The PCL-051 System On Module is a connector-less, BGA style variant of the PCM-051/phyCORE-AM335x SOM. Unlike traditional PHYTEC SOM products that support high density connectors, the PCL-051 SOM is directly soldered down to the phyBOARD-Wega using PHYTEC's Direct Solder Connect technology. This solution offers an ultra-low cost Single Board Computer for the AM335x processor, while maintaining most of the advantages of the SOM concept.

Adding the phyCORE-AM335x SOM into your own design is as simple as ordering the connectored version (PCM-051) and making use of our phyCORE Carrier Board (PCM-953) reference schematics.

1.1.1 Features of the phyBOARD-Wega-AM335x

The phyBOARD-Wega-AM335x supports the following features :

- PHYTEC’s phyCORE-AM335x SOM with Direct Solder Connect (DSC)
- Pico ITX standard dimensions (100 mm × 72 mm)
- Boot from MMC or NAND Flash
- Max. 1 GHz core clock frequency
- Three different power supply options (5 V only with 3.5 mm combicon or micro USB connector; external power module e.g. 12 V – 24 V input voltage)
- Two RJ45 jacks for 10/100 Mbps Ethernet
- One USB Host interface brought out to an upright USB Standard-A connector or at the expansion connector
- One USB OTG interface available at an USB Micro-AB connector at the back side
- One Secure Digital / Multi Media Memory Card interface brought out to a Micro-SD connector at the back side
- CAN interface at 5×2 pin header 2.54 mm

- Audiocodec with Stereo Line In and Line Out (3×2 pin header 2.54 mm) and mono speaker (2-pole Molex)
- RS-232 transceiver supporting UART1 incl. handshake signals with data rates of up to 1 Mbps (5×2 pin header 2.54 mm)
- Reset-Button
- Audio/Video (A/V) connectors
- Expansion connector
- Backup battery supply for RTC with Goldcap (lasts approx. 17 ½ days)

1.1.2 View of the phyBOARD-Wega-AM335x

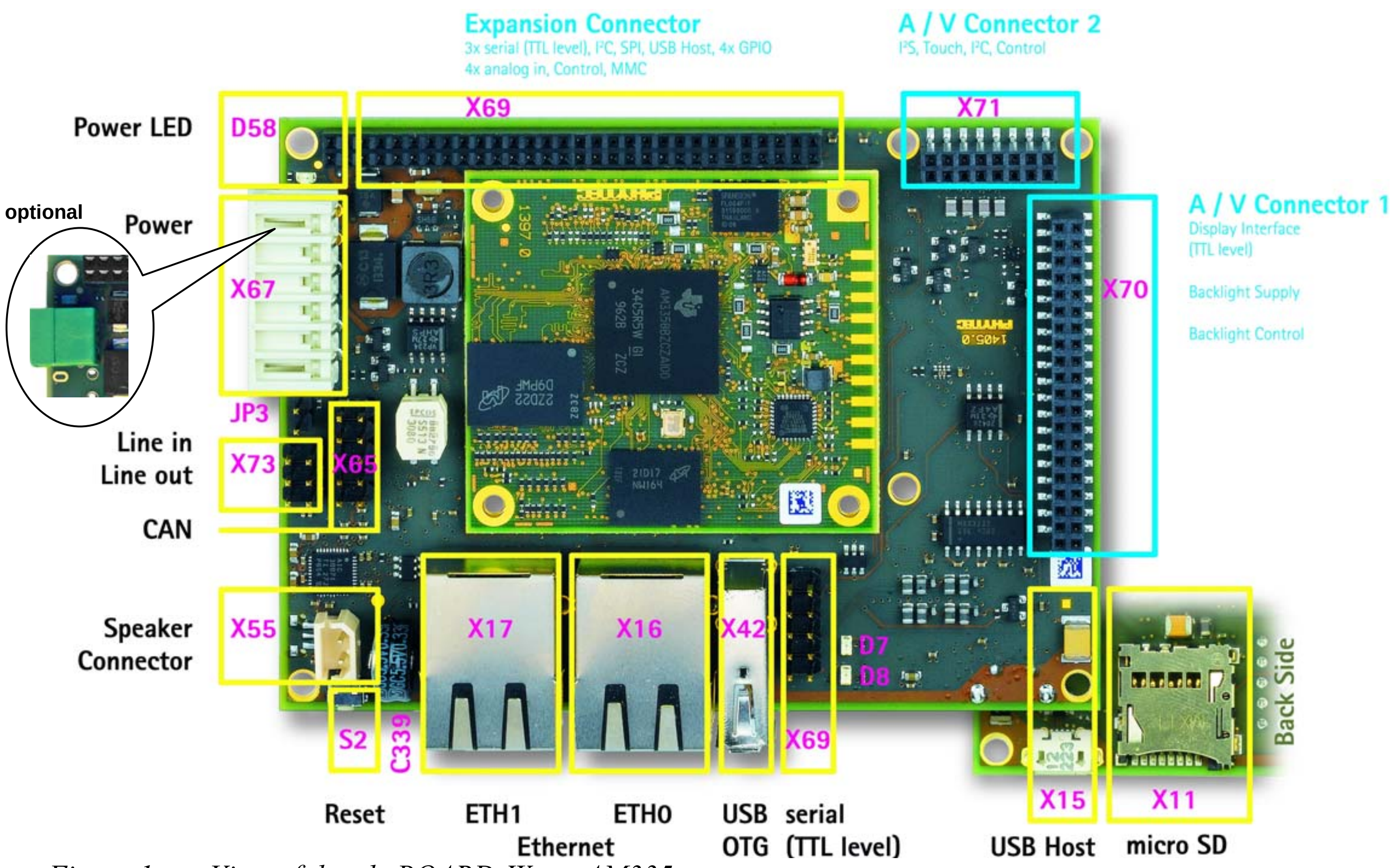


Figure 1: View of the phyBOARD-Wega-AM335x

1.1.3 Block Diagram

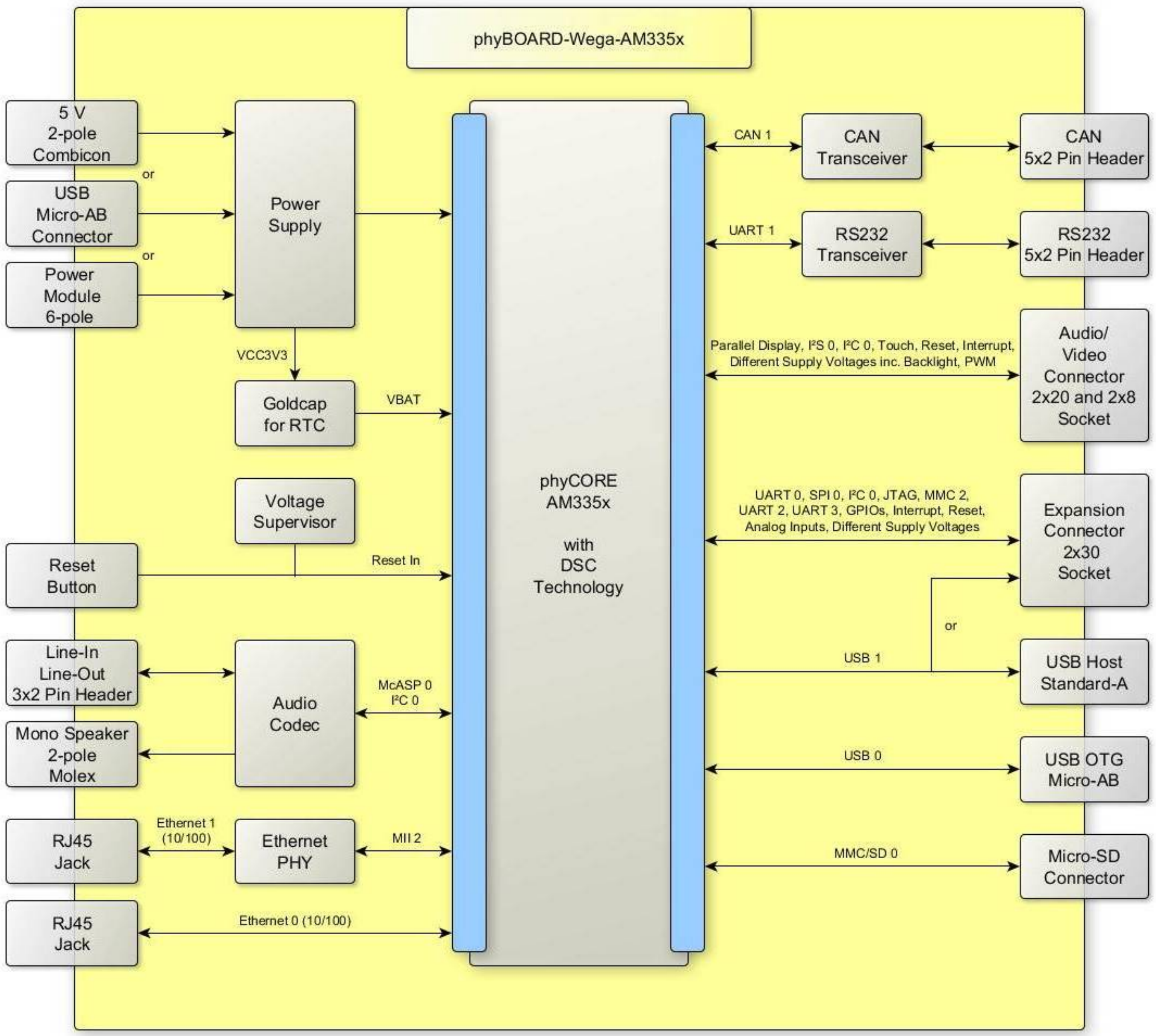


Figure 2: Block Diagram of the phyBOARD-Wega-AM335x

1.2 Software Overview

1.2.1 Ubuntu

Ubuntu - which you can find on the phyBOARD-Wega-AM335x Kit-DVD - 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 *12.04.3* and was released on 15. February 2013. *Ubuntu* 12.04 code name "*Precise Pangolin*" is designated as a **Long Term Support (LTS)** release and the first stable release was on 26 April 2012. 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*.

1.2.2 Eclipse

The Eclipse platform provides support for C/C++ development. Because the Eclipse platform is only a framework for developer tools, it doesn't support C/C++ directly; instead it uses external plug-ins. This Application Guide shows you how to make use of the *CDT,* a set of plug-ins for C/C++ development in conjunction with the GCC C/C++ tool chain.
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.
Due to its complexity, the CDT is broken down into several components, which take the form of separate plug-ins. Each component operates as an autonomous project, with its own set of committers, bug categories, and mailing lists. However, all plug-ins are required for the CDT to work properly. Here is a list of the plug-ins/components:

- **Primary CDT plug-in** is the “framework” for the CDT plug-ins.
- **CDT Feature Eclipse** is the CDT Feature Component.
- **CDT Core** provides Core Model, CDOM, and Core Components.
- **CDT UI** is the Core UI, views, editors, and wizards.
- **CDT Launch** provides the launch mechanism for external tools such as the compiler and debugger.

- **CDT Debug Core** provides debugging functions.
- **CDT Debug UI** provides the user interface for the CDT debugging editors, views, and wizards.
- **CDT Debug MI** is the application connector for MI-compatible debuggers.

1.2.3 The GNU Cross Development Tool Chain

Cross development in general refers to the overall software development process that produces a single application or a complete system running on a platform that is different from the development platform. This is an important concept when the target system doesn't have a native set of compilation tools, or when the host system is faster and has greater resources.

The platform where the actual development takes place is called the *host platform.* The platform where the final application is tested and run is called the *target platform.* In this Quick Start we are using an x86-based Linux as the host platform. As the target platform we are using the ARM®Cortex™-A8 architecture on the phyBOARD-Wega-Am335x SBC.

Building a program for a CPU architecture different from the one used on the machine where the compilation is done is accomplished using a cross compiler tool chain and cross-compiled libraries. In this Application Guide we are using the GNU C/C++ cross development tool chain.

2 Getting Started

During this chapter you will learn how to build your own C/C++ applications for the target with the help of Eclipse.

We establish that you have first played through our Quickstart Guide.

	As all changes on the example projects will be lost if you proceed using the live environment we recommend to now install our modified Ubuntu LiveDVD. If you only want to make your own fast experience with our phyBOARD-Wega-AM335x-Kit you can go on with section *2.2 "Working with Eclipse*".

⚠	To ensure successful introduction to the development with the phyBOARD-Wega-AM335x we strongly recommend continuing with the modified Ubuntu, either in a live environment, or completely installed on your PC as described in the next section. Nontheless; if you want to use your already existing environment we explain how to modify your system to get the same experience like our LiveDVD in the System Guide in section "Setup your own host PC"

2.1 Installing our modified Ubuntu LiveDVD

As described above, this step is not needed to successfully finish this chapter but for more in-depth development it is better to install our LiveDVD on your computer.

If another operating system is installed on your computer, you should first make a backup of your important files. Before we can start, make sure that your computer is set to boot from DVD before it boots from a hard disk drive.

- Insert the Linux- phyBOARD-Wega-AM335x-Kit-DVD into your DVD drive.
- Start or restart your computer. Your system finds a bootable DVD and starts from it. After a while a language screen appears.
- Select your desired language and click **Install PHYLiveDVD**
- The **Preparing to install PHYLiveDVD** window appears. From Ubuntu it's adviced that you select "*Download updates while installing*" and "*Install this third-party software*" now. Click on **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 choose one click on **Continue**.
- The **Install PHYLiveDVD...** window appears. After you have checked the settings you can click on **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. How to do that was shown in our Quickstart Guide.

2.2 Working with Eclipse

Now we start developing our own applications with the help of Eclipse. First we take a look on the C programming language before we go on with programming a QT project. At the end of this chapter we explain how to execute your written programs automatically when booting the target.

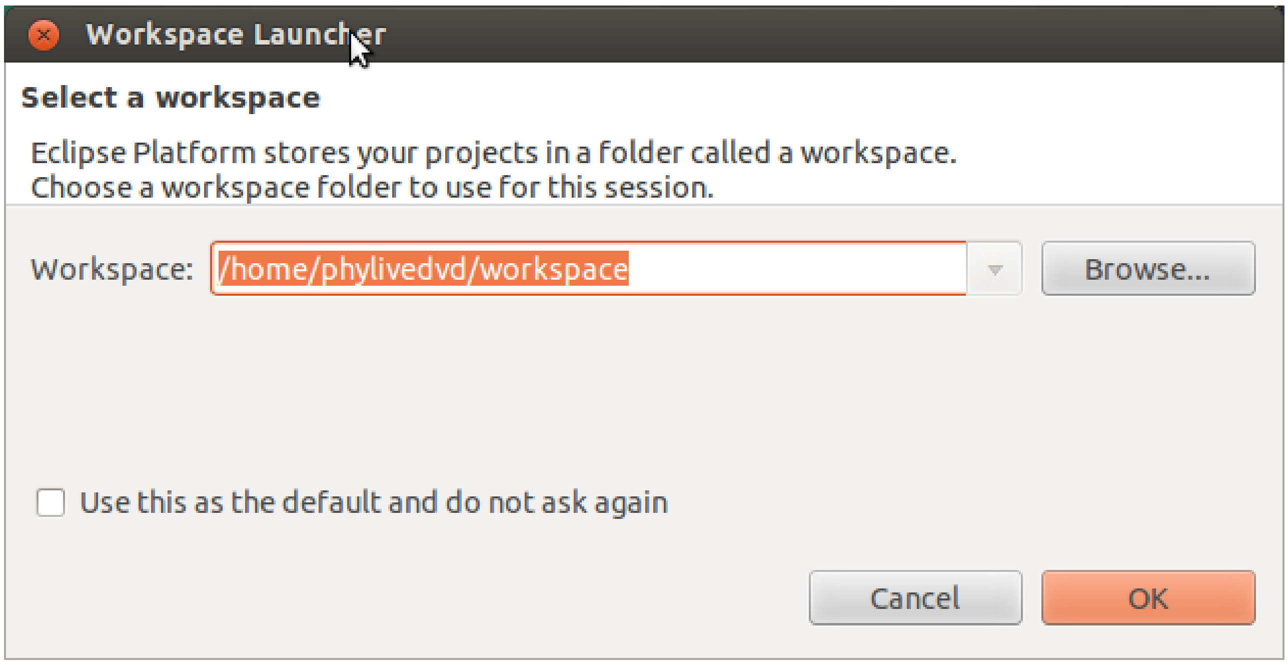
2.2.1 Programming in the C/C++ perspective

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 compiling the project, you will copy and execute the newly created program on the target.

2.2.1.1 Work with the demo project

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

- Confirm the workspace directory with OK

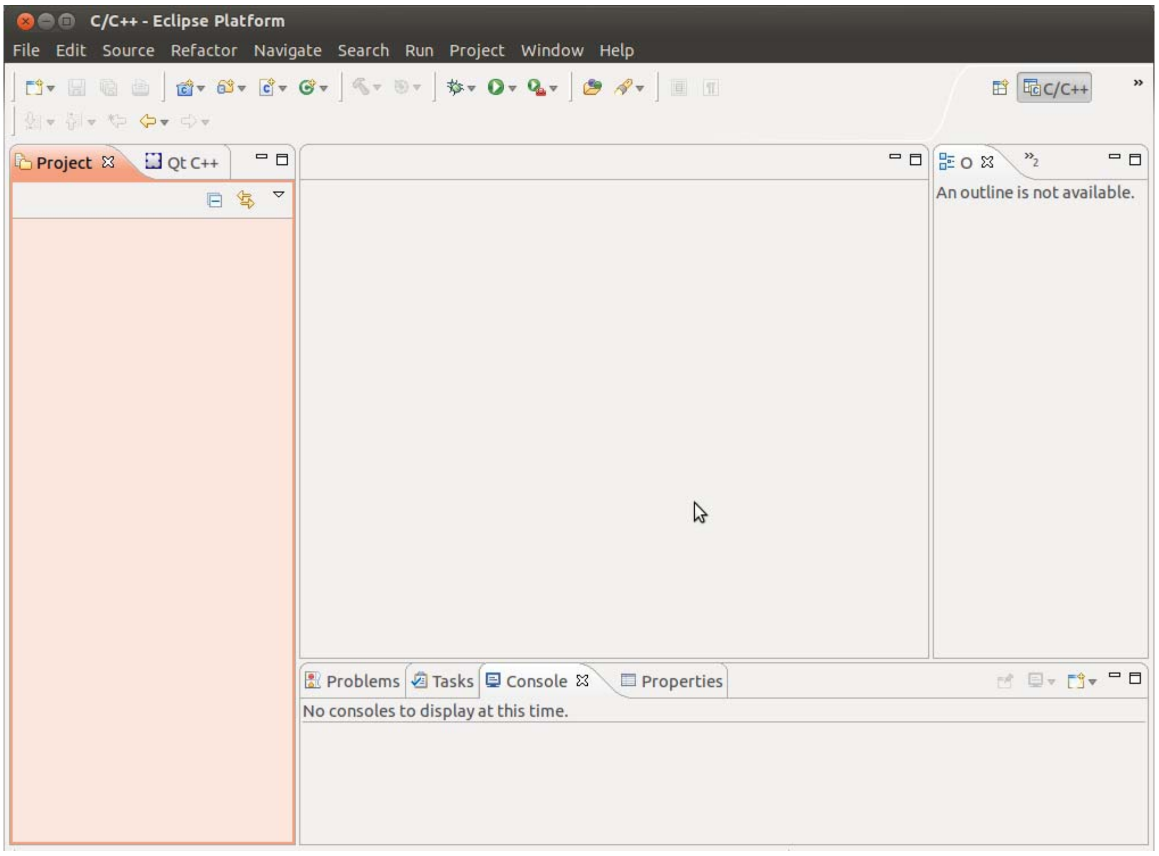

Workspace Launcher

Select a workspace

Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /home/phylivedvd/workspace

Now you can see the Eclipse workbench.

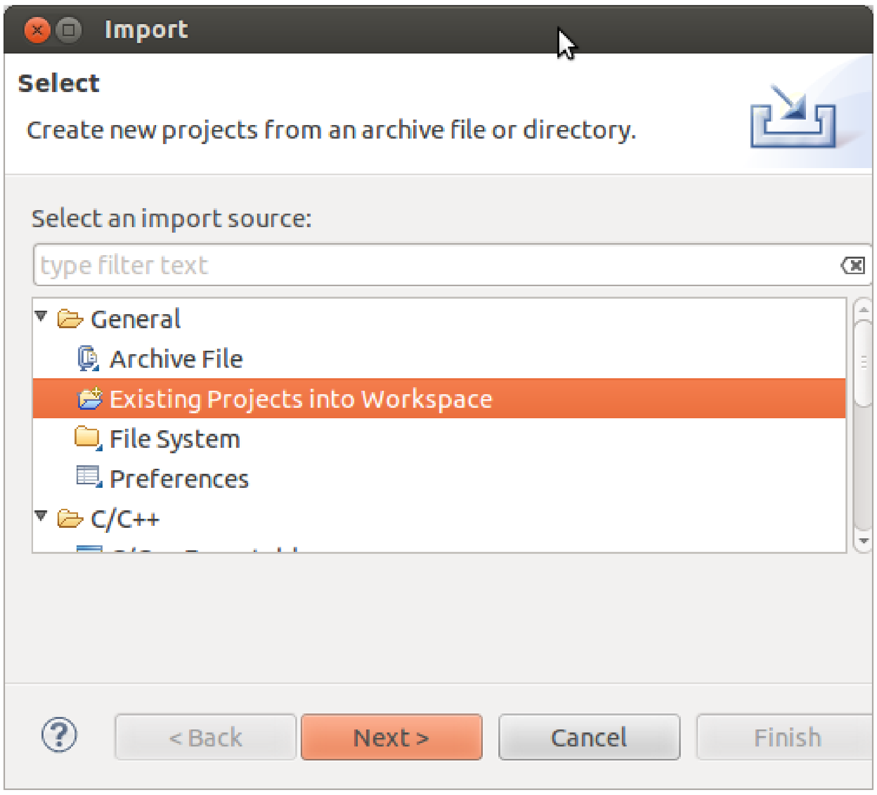


An outline is not available.

No consoles to display at this time.

First we will import an existing project.

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



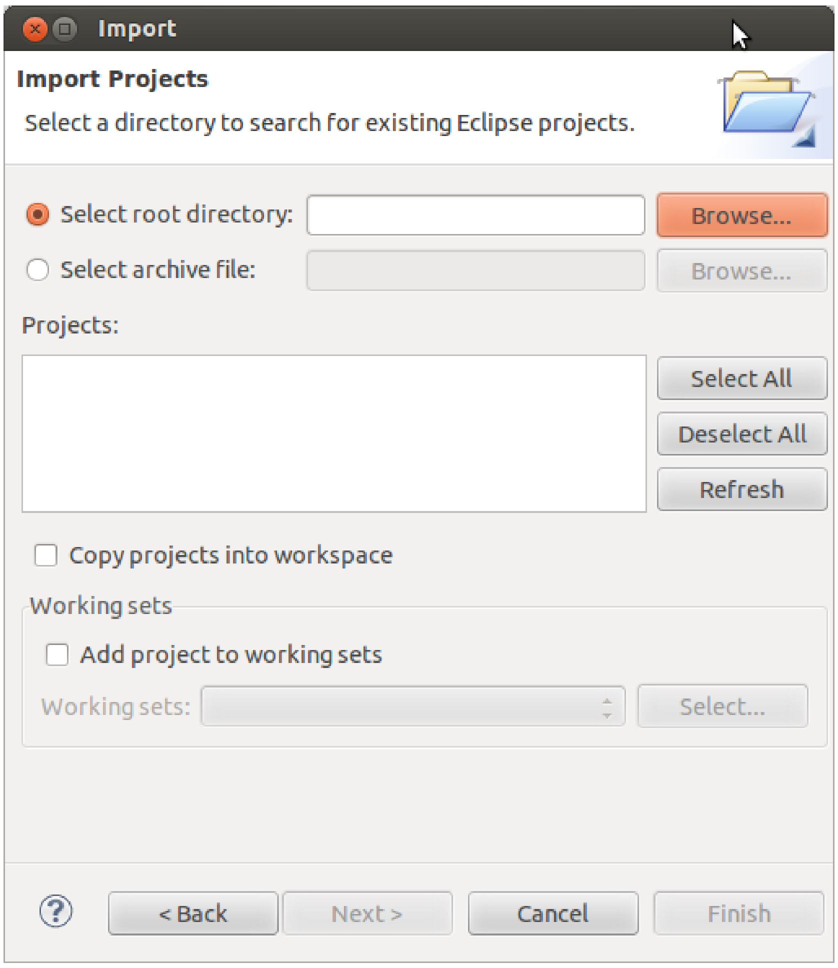
Select

Create new projects from an archive file or directory.

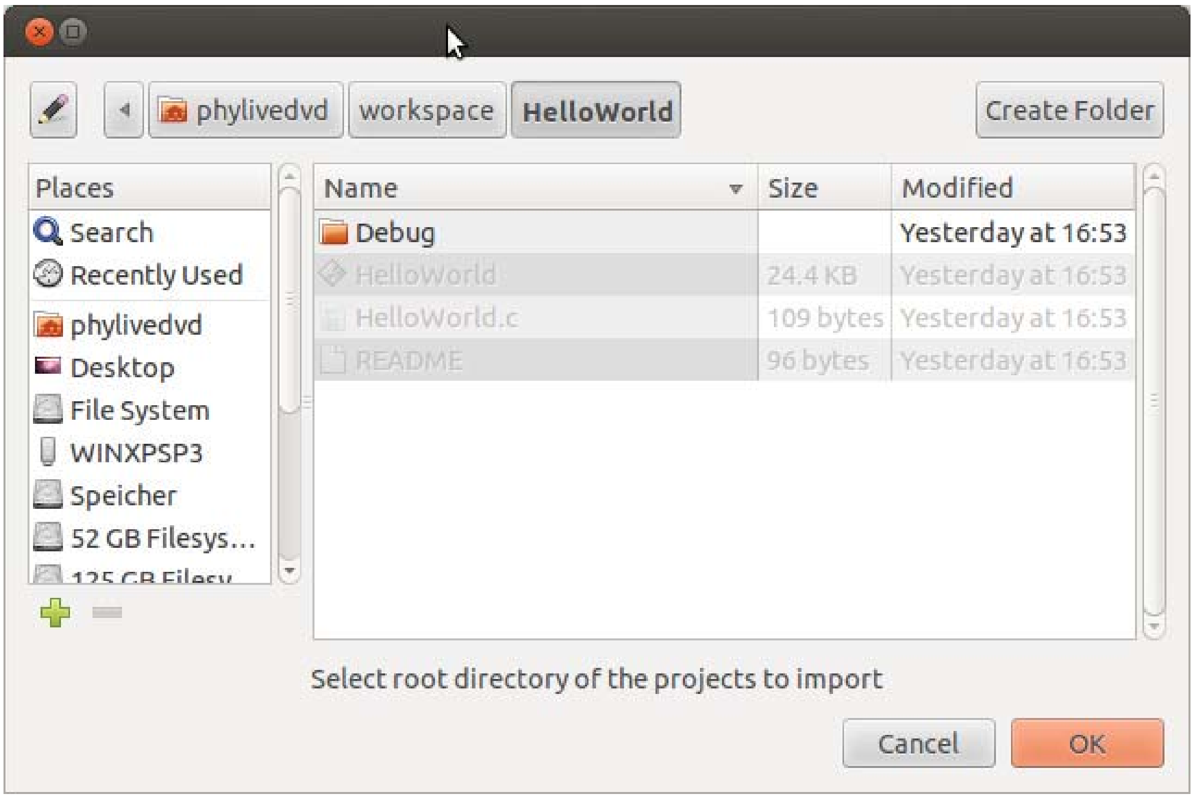
Select an import source:

type filter text

- Select *Browse*

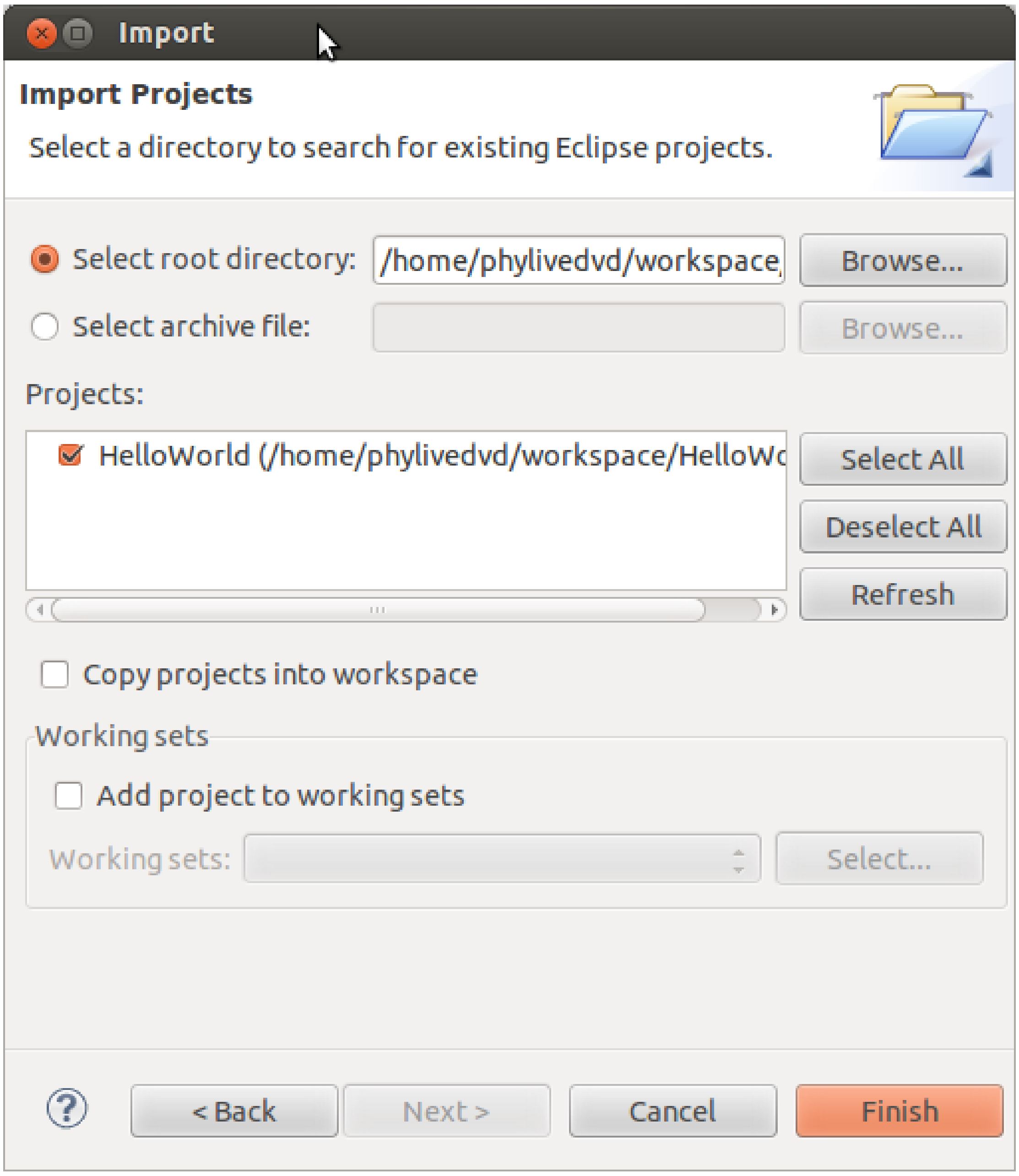


- Double-click the *HelloWorld* directory under */home/phylivedvd/workspace/*
- Click *OK*



Name	Size	Modified
Debug		Yesterday at 16:53
HelloWorld	24.4 KB	Yesterday at 16:53
HelloWorld.c	109 bytes	Yesterday at 16:53
README	96 bytes	Yesterday at 16:53

- Select *Finish* to import the project



Select root directory: /home/phylivedvd/workspace

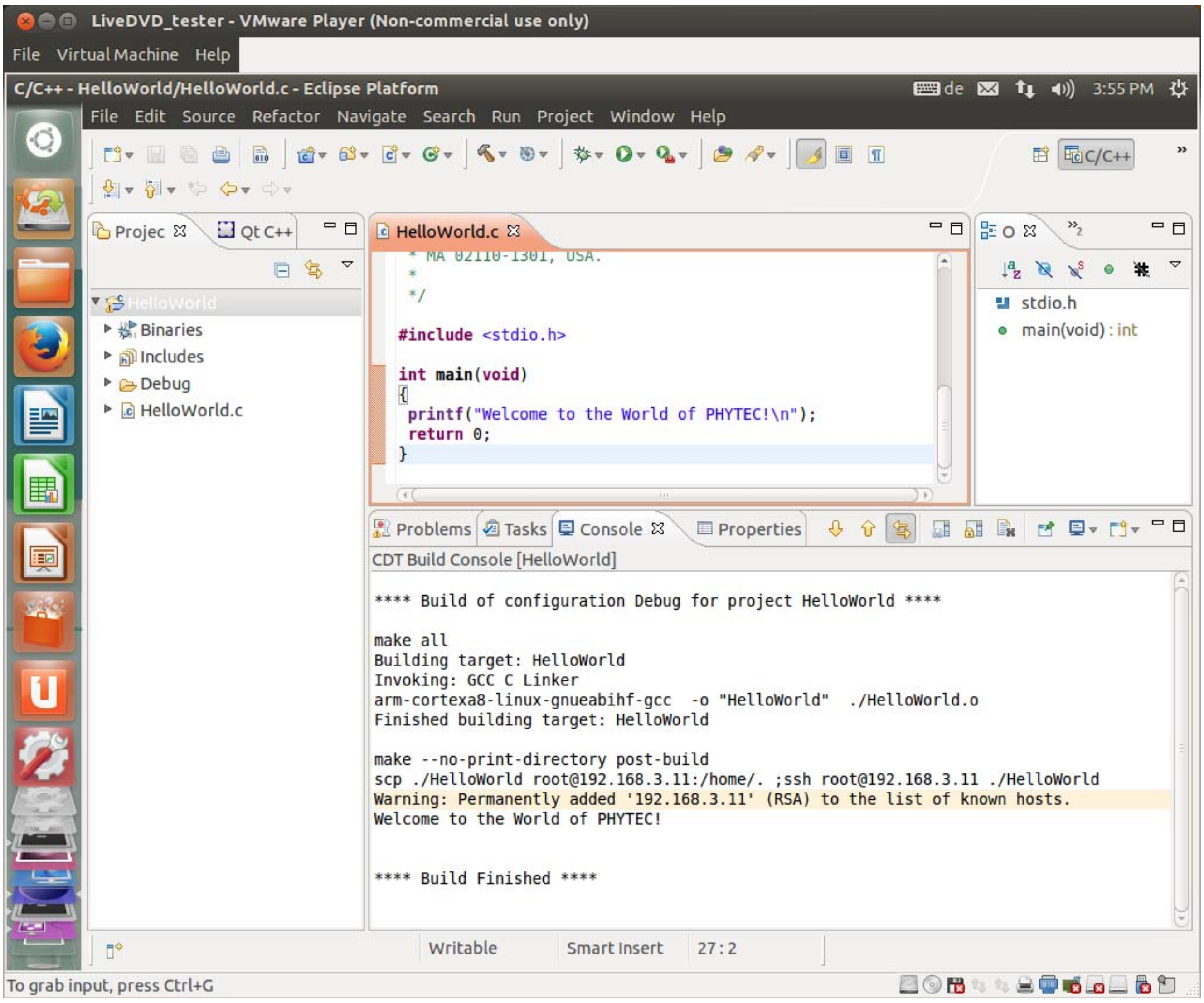
Projects: HelloWorld (/home/phylivedvd/workspace/HelloWo

Copy projects into workspace (unchecked)

Add project to working sets (unchecked)

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*. Because this is our first SSH connection we must accept the fingerprint with yes.

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



LiveDVD_tester - VMware Player (Non-commercial use only)
C/C++ - HelloWorld/HelloWorld.c - Eclipse Platform
File Edit Source Refactor Navigate Search Run Project Window Help
HelloWorld
Binaries
Includes
Debug
HelloWorld.c
stdio.h
main(void) : int
Problems Tasks Console Properties
CDT Build Console [HelloWorld]
Writable Smart Insert 27 : 2
To grab input, press Ctrl+G

```
 * MA 02110-1301, USA.
 *
 */

#include <stdio.h>

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

```
**** Build of configuration Debug for project HelloWorld ****

make all
Building target: HelloWorld
Invoking: GCC C Linker
arm-cortexa8-linux-gnueabihf-gcc  -o "HelloWorld"  ./HelloWorld.o
Finished building target: HelloWorld

make --no-print-directory post-build
scp ./HelloWorld root@192.168.3.11:/home/. ;ssh root@192.168.3.11 ./HelloWorld
Warning: Permanently added '192.168.3.11' (RSA) to the list of known hosts.
Welcome to the World of PHYTEC!


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

If the project is not built automatically, you will have to check *Project* ► *Build automatically* from the menu bar. To build it new select *Project* ► *Clean...*

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.

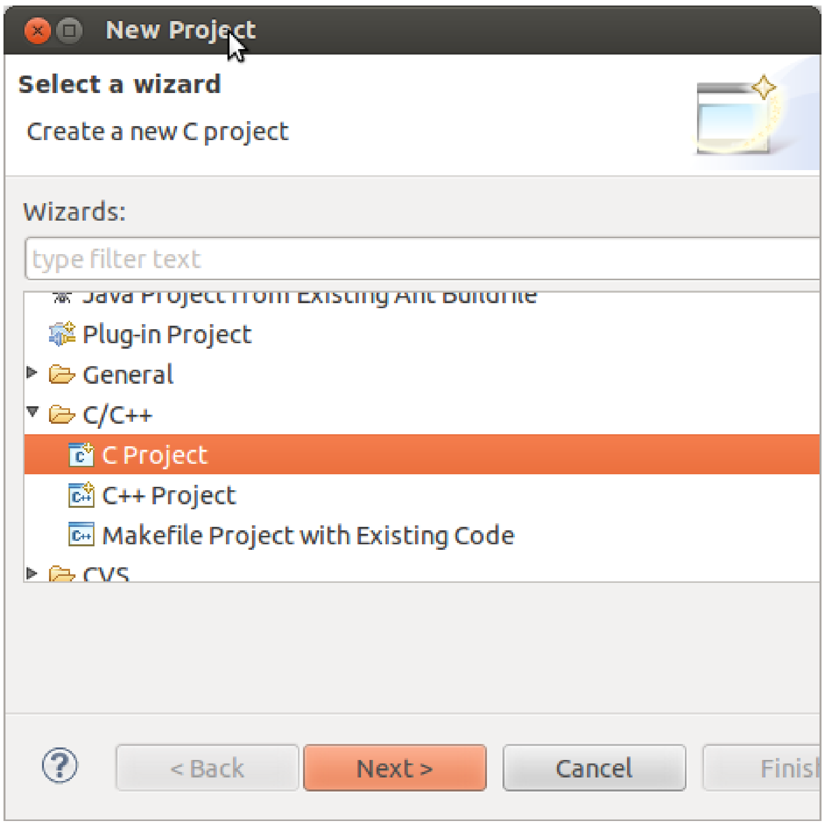
2.2.1.2 Creating a New Project

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.

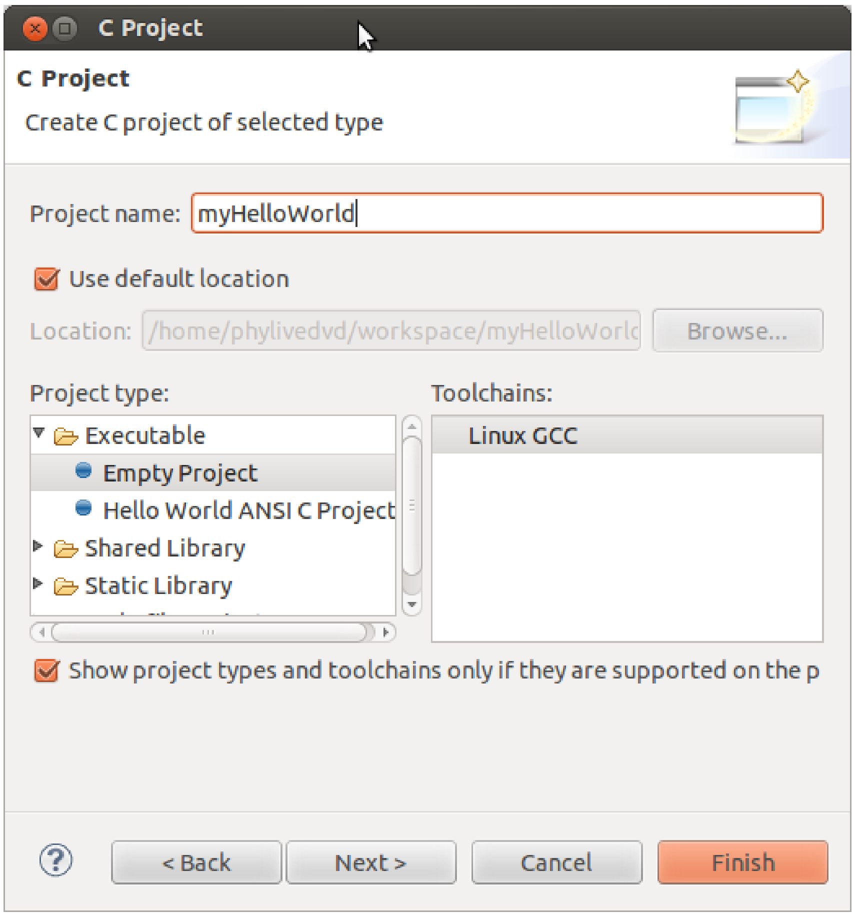
- Open Eclipse if it isn't already opened
- Select *File* ▶ *New* ▶ *Project* from the menu bar.

A new dialog opens

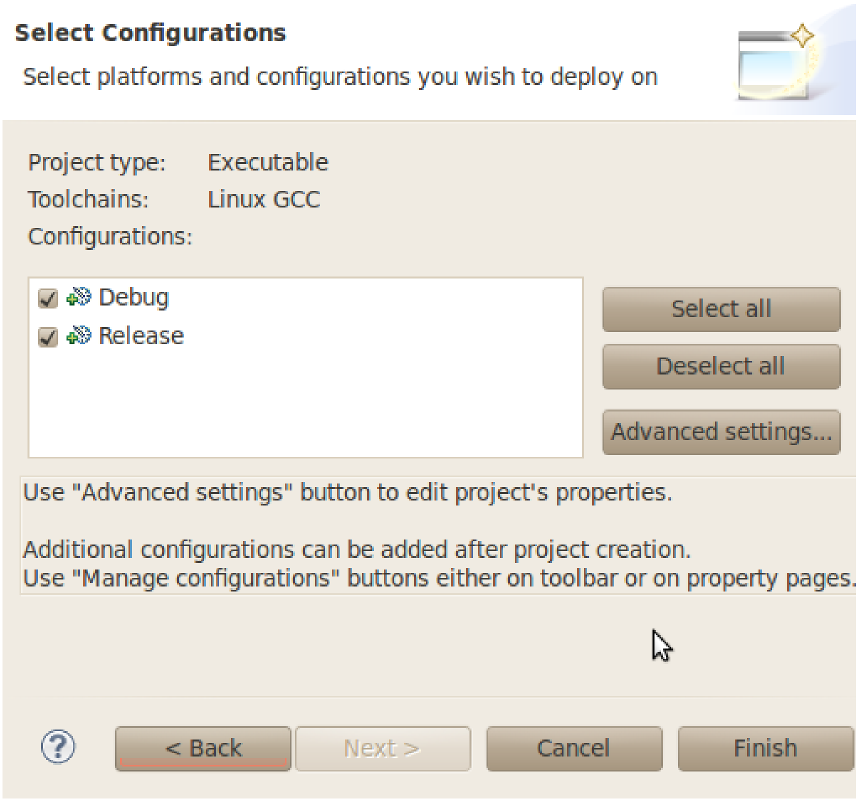
- Select *C Project* and click *Next*



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


C Project
C Project
Create C project of selected type
Project name: myHelloWorld
Use default location
Location: /home/phylivedvd/workspace/myHelloWorld
Browse...
Project type:
Toolchains:
Executable
Empty Project
Hello World ANSI C Project
Shared Library
Static Library
Linux GCC
Show project types and toolchains only if they are supported on the p
< Back
Next >
Cancel
Finish

- Click *Finish*

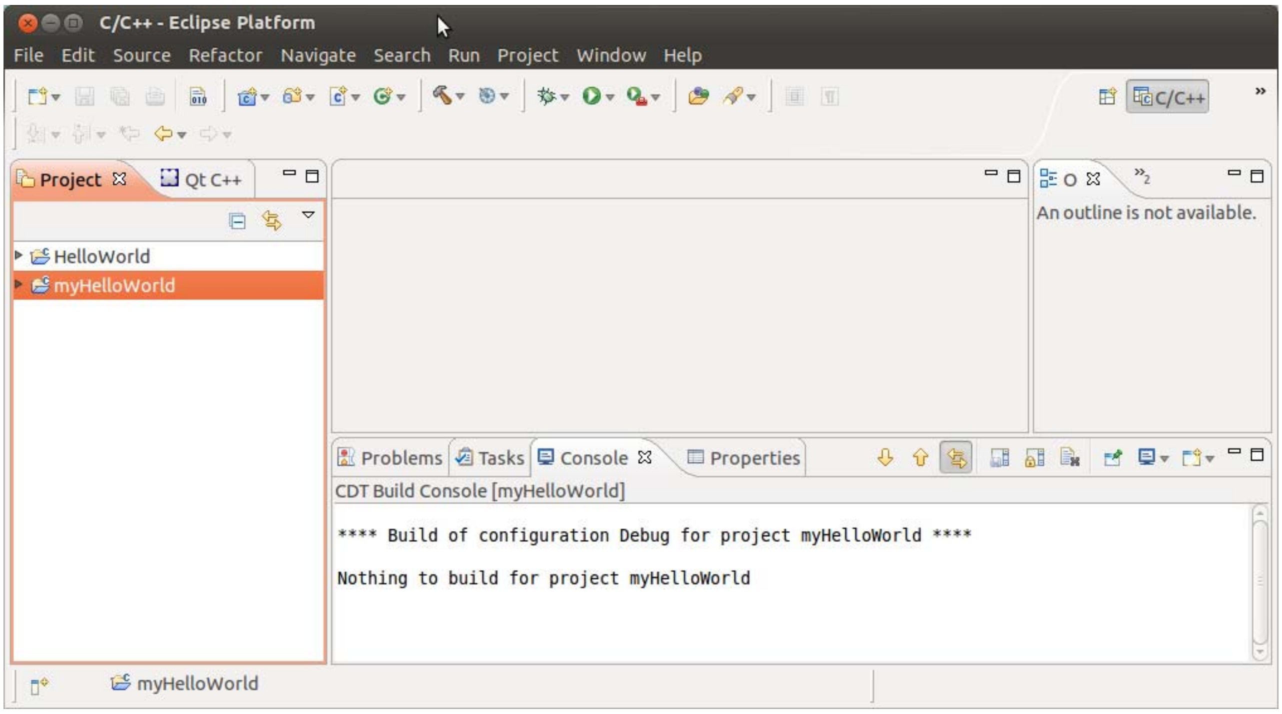


Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

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

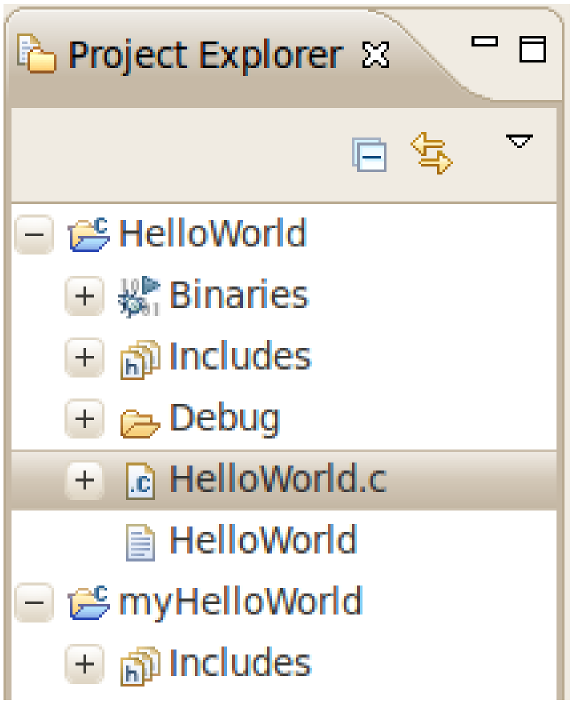
- Double-Click the *HelloWorld* project which we have worked with previously



CDT Build Console [myHelloWorld]

```
**** Build of configuration Debug for project myHelloWorld ****

Nothing to build for project myHelloWorld
```

- 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

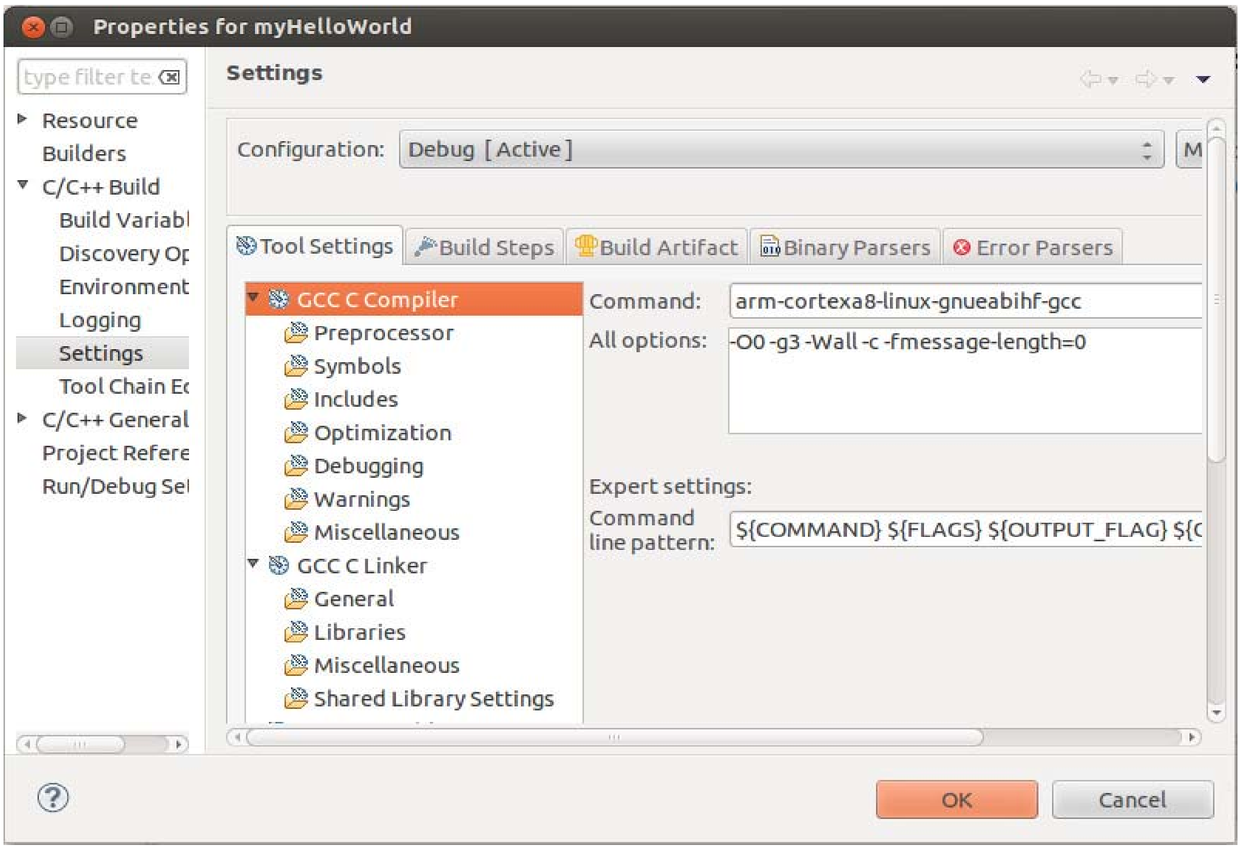
If *Build Automatically* from the *Project* menu is selected, the *HelloWorld* application will now be compiled and created with the standard GCC C/C++ compiler suitable for your host machine. You will find the executable file, which can only run on your host system, in the *workspace/myHelloWorld/Debug*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

To compile your project for the phyCORE-AM335x 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*
- Enter **arm-cortexa8-linux-gnueabihf-gcc** into the *Command* input field



Configuration: Debug [Active]

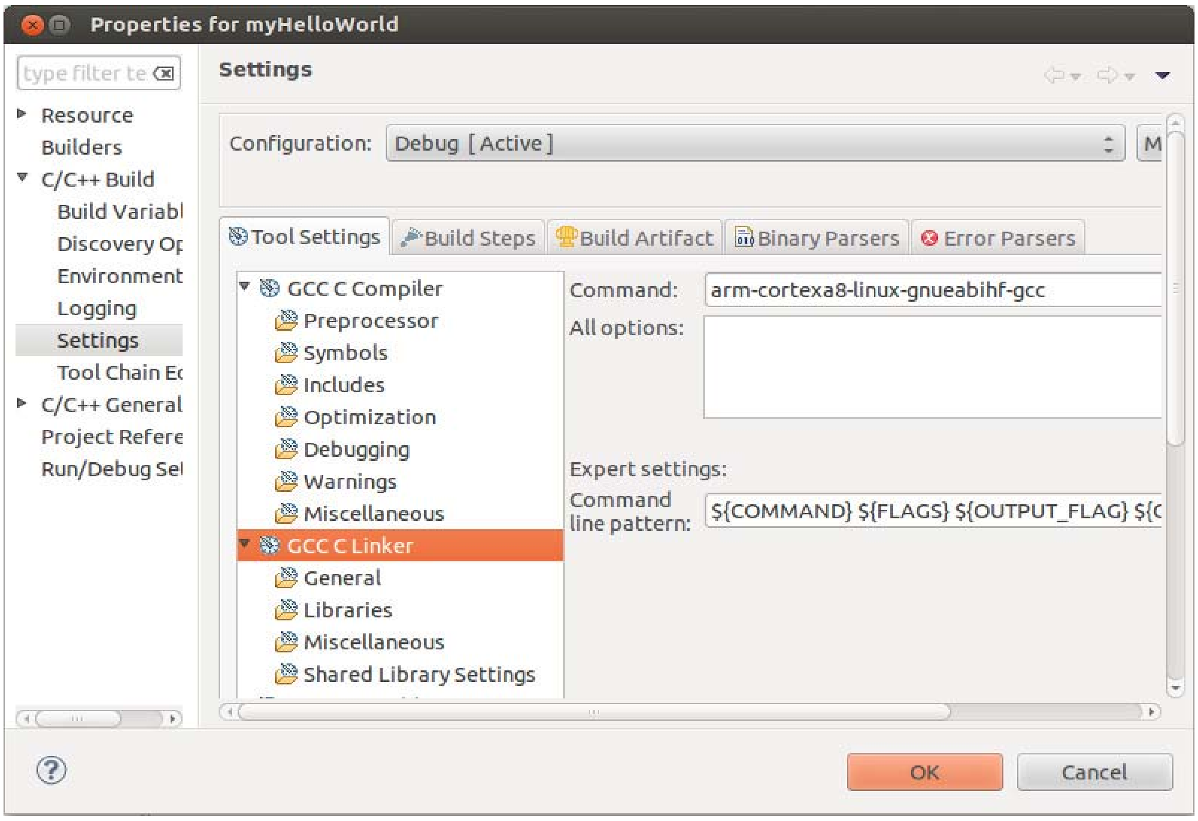
Command: arm-cortexa8-linux-gnueabihf-gcc

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

Expert settings:

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

- Select *GCC C Linker*
- Enter **arm-cortexa8-linux-gnueabihf-gcc** into the *Command* input field



Configuration: Debug [Active]

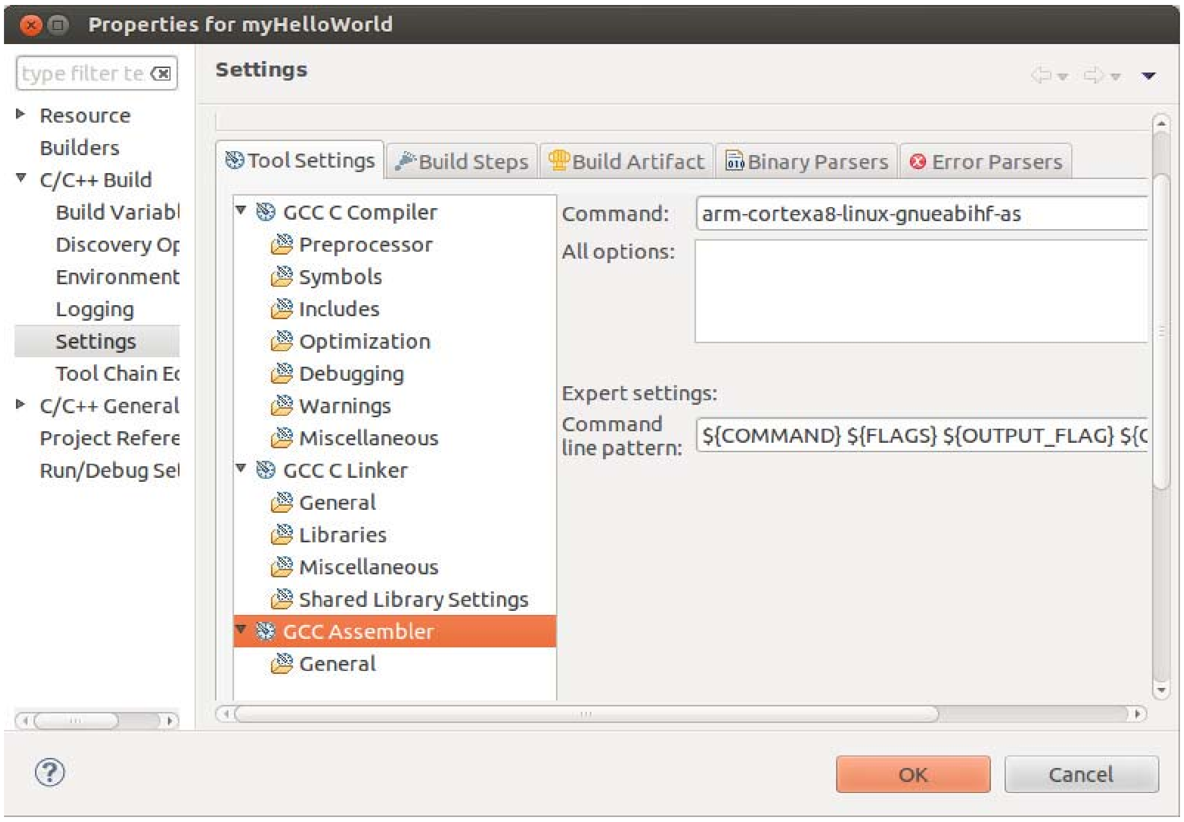
Command: arm-cortexa8-linux-gnueabihf-gcc

All options:

Expert settings:

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

- Select *GCC Assembler*
- In the *Command* input field, change the default **as** to **arm-cortexa8-linux-gnueabihf-as**



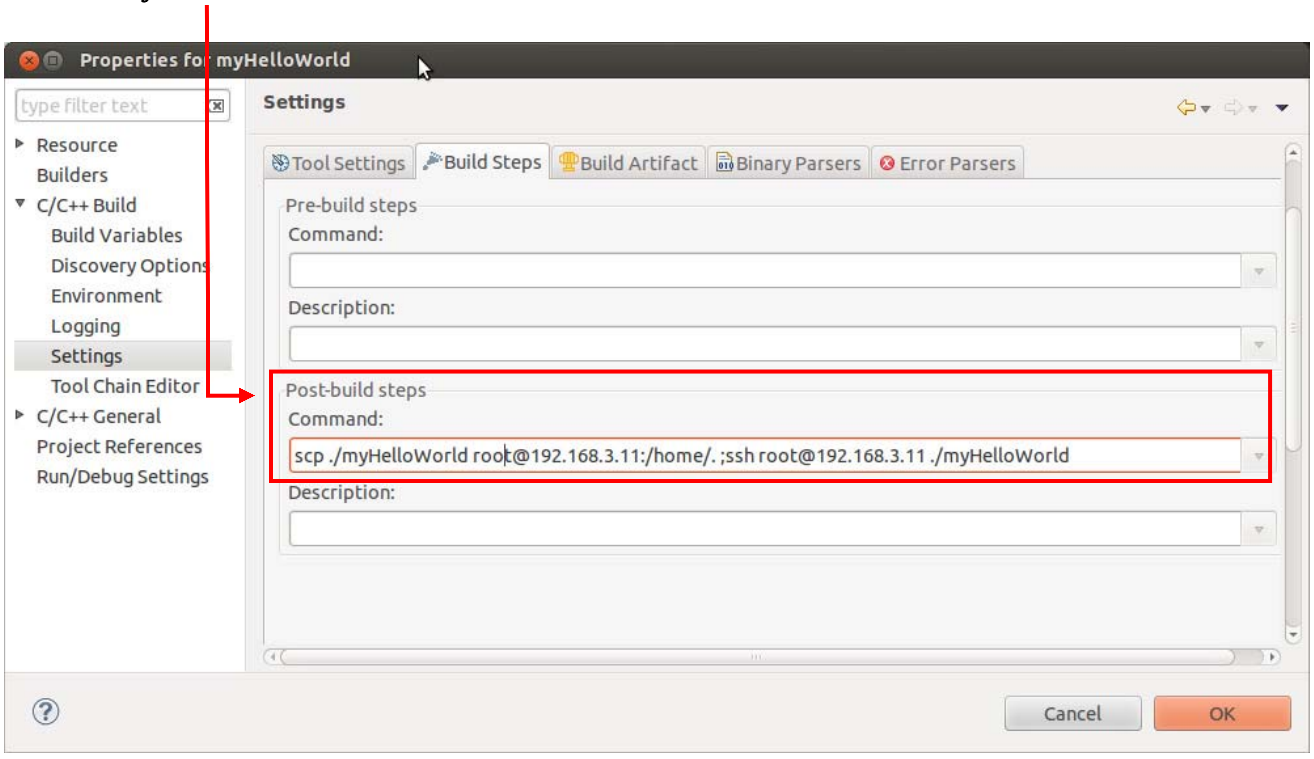
Command: arm-cortexa8-linux-gnueabihf-as

All options:

Expert settings:

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

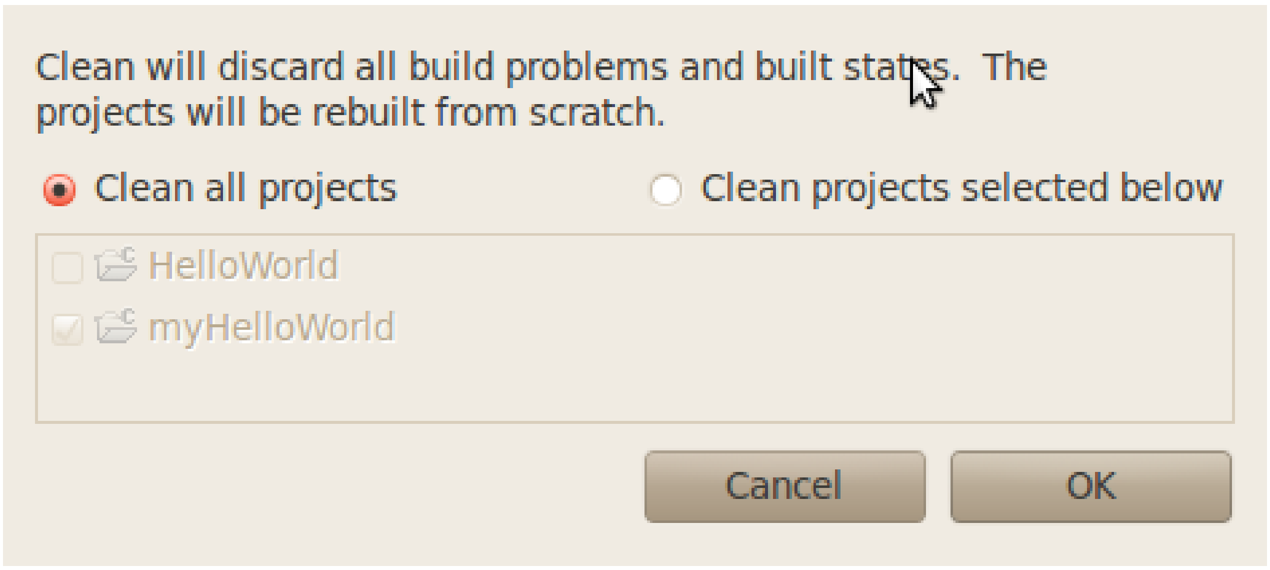
- Click *Apply*
- Select the *Build Steps* tab
- Enter the following command in the Post-build steps *Command* input field:
  scp ./myHelloWorld root@192.168.3.11:/home/. ;ssh root@192.168.3.11 ./ myHelloWorld



Pre-build steps
Command:
Description:
Post-build steps
Command: scp ./myHelloWorld root@192.168.3.11:/home/. ;ssh root@192.168.3.11 ./myHelloWorld
Description:

⚠	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


Clean will discard all build problems and built states. The projects will be rebuilt from scratch.

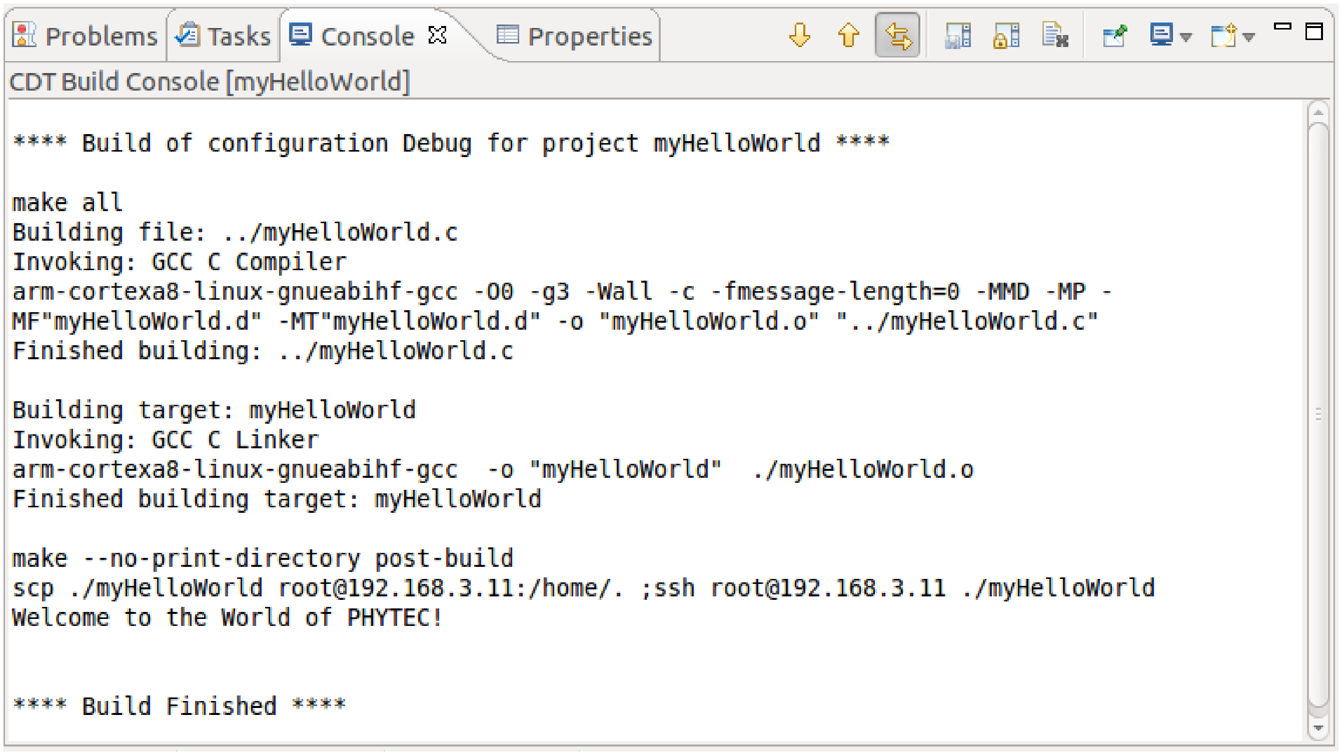
[✓ Clean all projects] Clean projects selected below

- 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:



```
**** Build of configuration Debug for project myHelloWorld ****

make all
Building file: ../myHelloWorld.c
Invoking: GCC C Compiler
arm-cortexa8-linux-gnueabihf-gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -
MF"myHelloWorld.d" -MT"myHelloWorld.d" -o "myHelloWorld.o" "../myHelloWorld.c"
Finished building: ../myHelloWorld.c

Building target: myHelloWorld
Invoking: GCC C Linker
arm-cortexa8-linux-gnueabihf-gcc  -o "myHelloWorld"  ./myHelloWorld.o
Finished building target: myHelloWorld

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


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

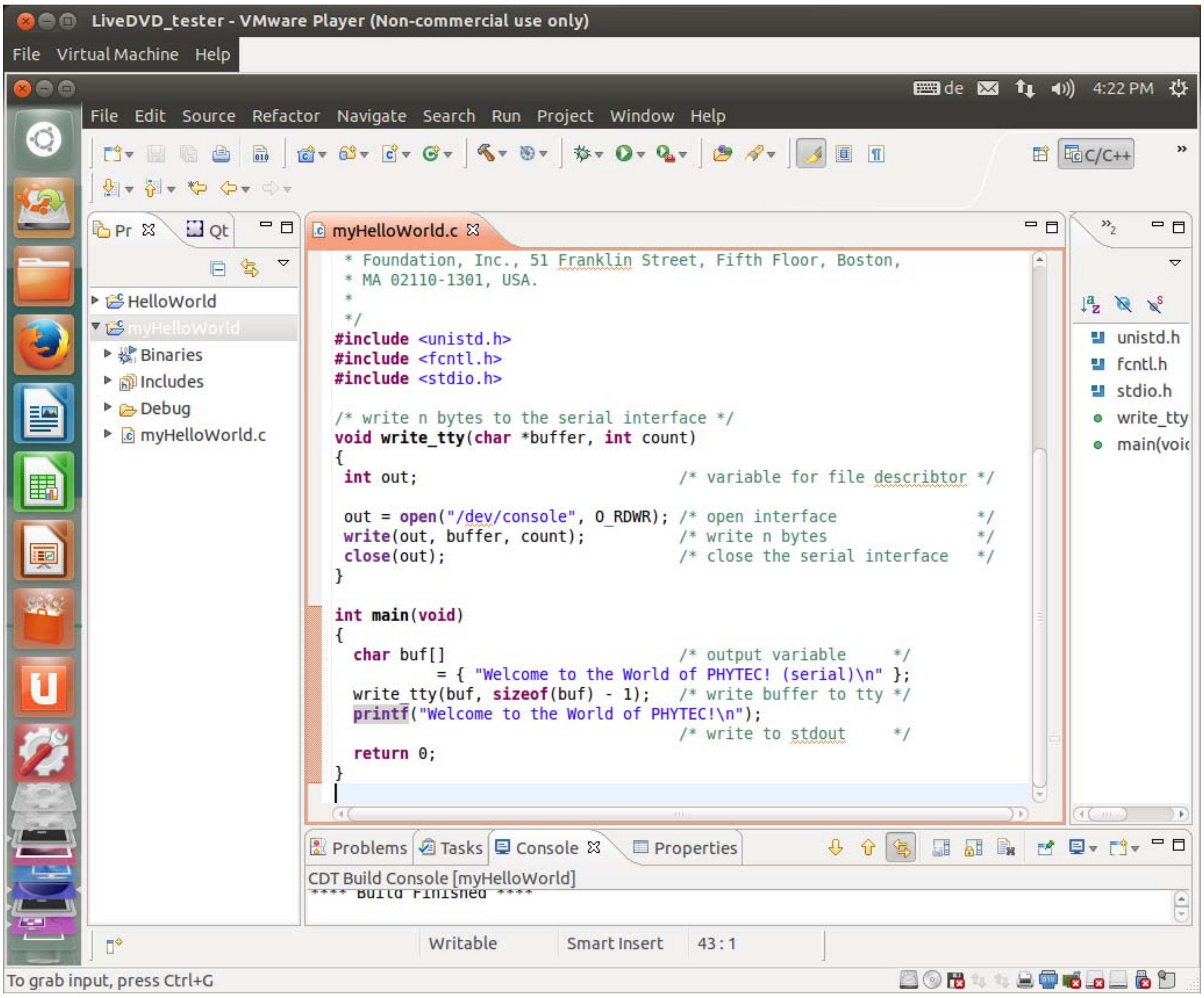
	You have successfully created your first own project with the Eclipse IDE. You have configured the project to create an application for your target platform.

2.2.1.3 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 phyBOARD-WEGA-AM335x, 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



```
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.
 *
 */
#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 describtor */

 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;
}
```

```
CDT Build Console [myHelloWorld]
**** Build Finished ****
```

LiveDVD_tester - VMware Player (Non-commercial use only)
File Virtual Machine Help
File Edit Source Refactor Navigate Search Run Project Window Help
C/C++
myHelloWorld.c
HelloWorld
myHelloWorld
Binaries
Includes
Debug
myHelloWorld.c
unistd.h
fcntl.h
stdio.h
write_tty
main(voic
Problems Tasks Console Properties
Writable Smart Insert 43 : 1
To grab input, press Ctrl+G

- 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

Microcom

- 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 a 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.

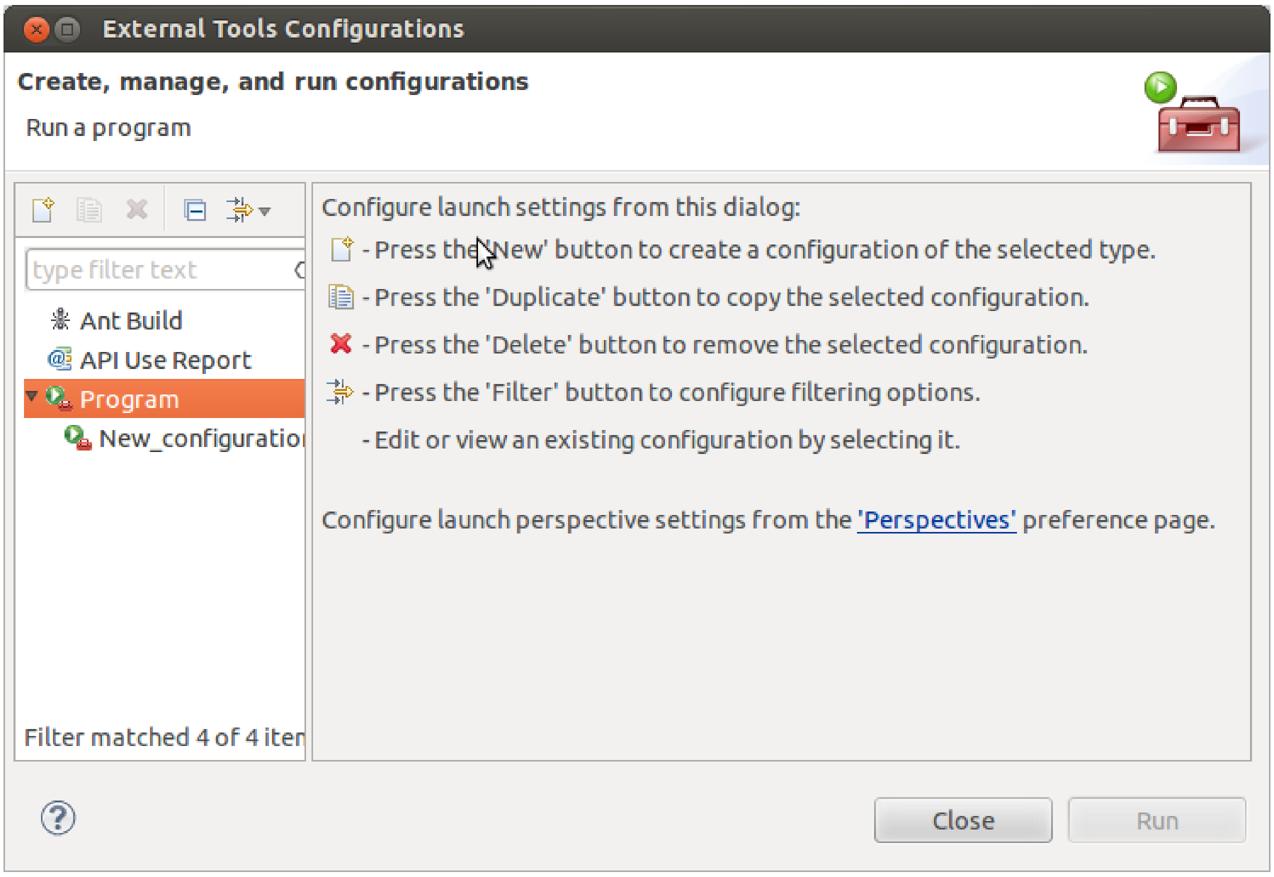
In this section you have changed an existing application. You also learned how to access the serial interface. First you called the function *open()* on the device */dev/console*. The return value of this function was a file descriptor. With the file descriptor you called the function *write()* to send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

This procedure is in principle quite typical for Linux, because Linux treats everything like a file.

2.2.1.4 Starting a Program out of Eclipse on the Target

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


External Tools Configurations

Create, manage, and run configurations

Run a program

type filter text

- Ant Build
- API Use Report
- Program
  - New_configuratio

Filter matched 4 of 4 iten

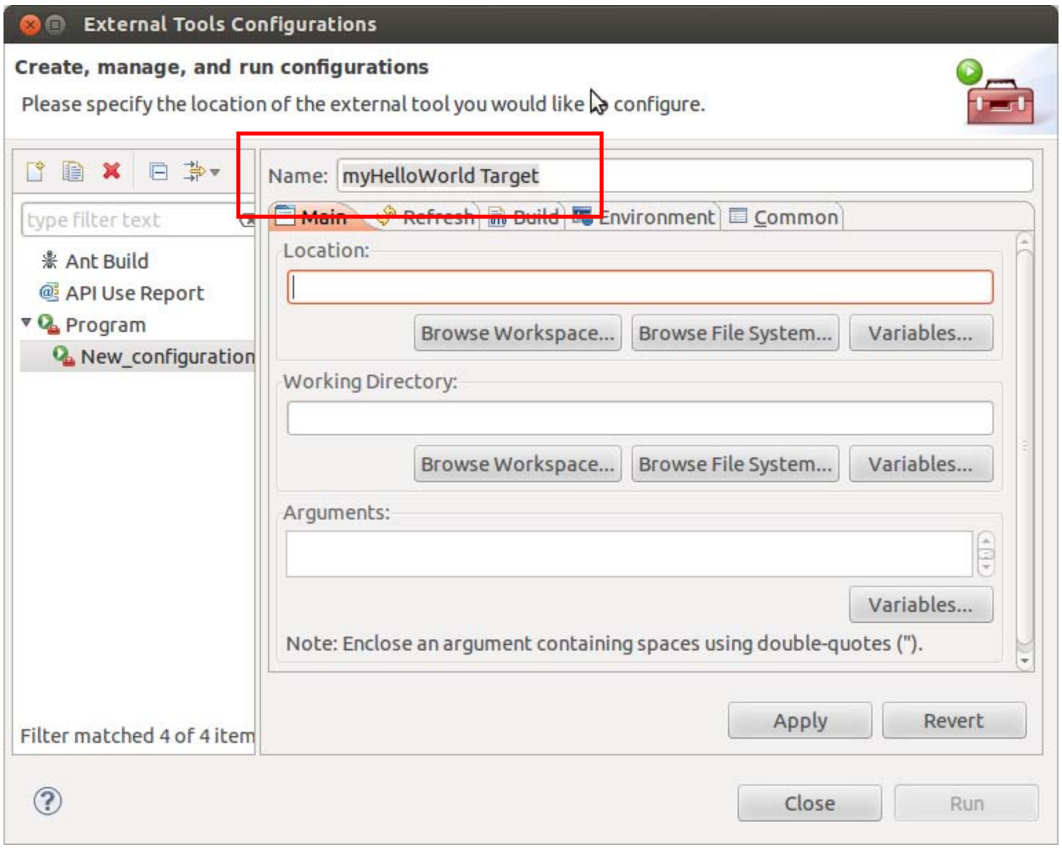
Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

- Under *Program* select *New_configuration (1)*

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


External Tools Configurations

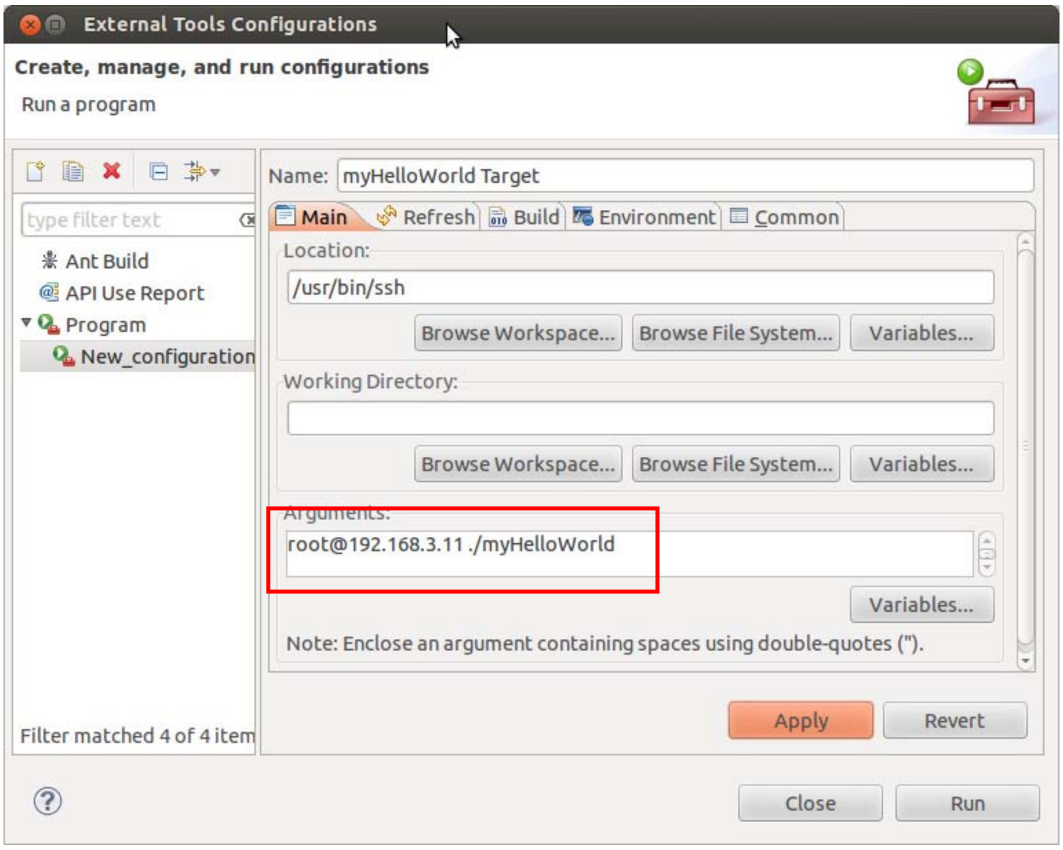
Create, manage, and run configurations

Please specify the location of the external tool you would like configure.

Name: myHelloWorld Target

Note: Enclose an argument containing spaces using double-quotes (").

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


External Tools Configurations

Create, manage, and run configurations

Run a program

Name: myHelloWorld Target

Location: /usr/bin/ssh

Arguments: root@192.168.3.11 ./myHelloWorld

Note: Enclose an argument containing spaces using double-quotes (").

- Select *Apply*

- Select *Run*

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.

2.2.2 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 Quickstart we will only describe debugging via TCP/IP.

2.2.2.1 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. This has already been added while building the program.

Microcom

- 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.

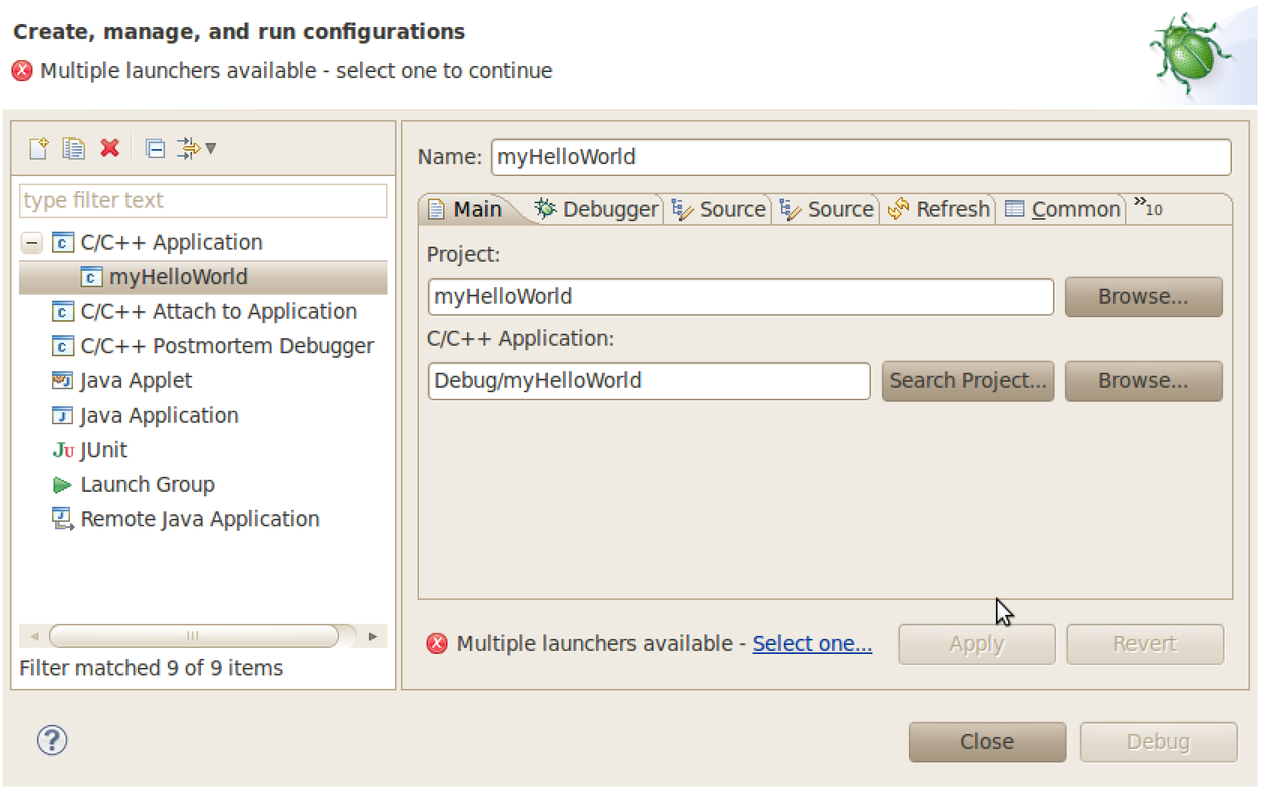
2.2.2.2 Configuring and starting the debugger in Eclipse

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 Debug* under *C/C++ Application* (to expand it *double click* on it)



Create, manage, and run configurations

Multiple launchers available - select one to continue

type filter text

- C/C++ Application
  - myHelloWorld
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- Java Applet
- Java Application
- JUnit
- Launch Group
- Remote Java Application

Filter matched 9 of 9 items

Name: myHelloWorld

Main | Debugger | Source | Source | Refresh | Common | »10

Project:

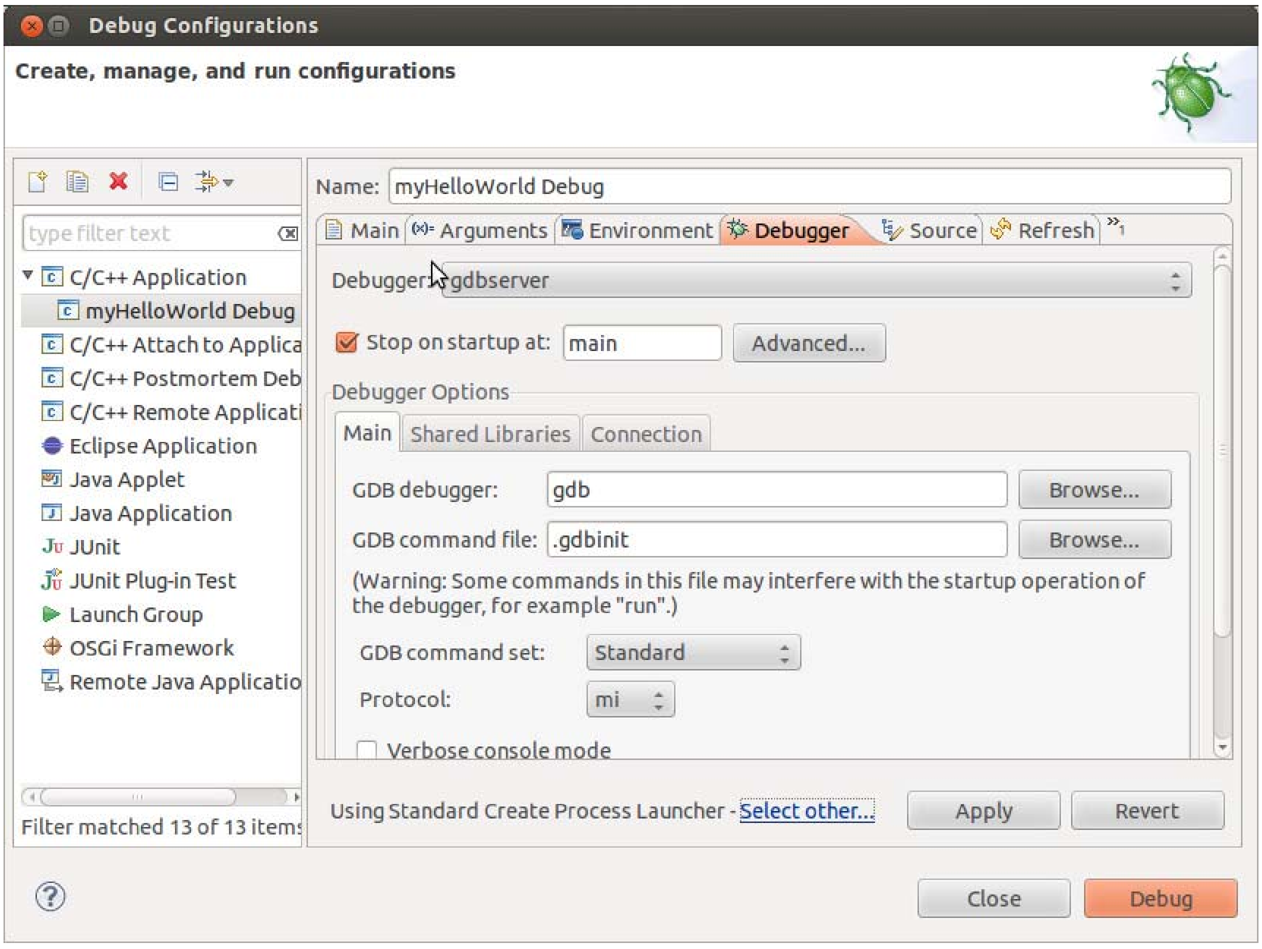
myHelloWorld

C/C++ Application:

Debug/myHelloWorld

Multiple launchers available - Select one...

- Select the *Debugger* tab



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
  - myHelloWorld Debug
- C/C++ Attach to Applica
- C/C++ Postmortem Deb
- C/C++ Remote Applicati
- Eclipse Application
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Launch Group
- OSGi Framework
- Remote Java Applicatio

Filter matched 13 of 13 items

Name: myHelloWorld Debug

Main | Arguments | Environment | Debugger | Source | Refresh | »1

Debugger: gdbserver

☑ Stop on startup at: main

Debugger Options

Main | Shared Libraries | Connection

GDB debugger: gdb

GDB command file: .gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

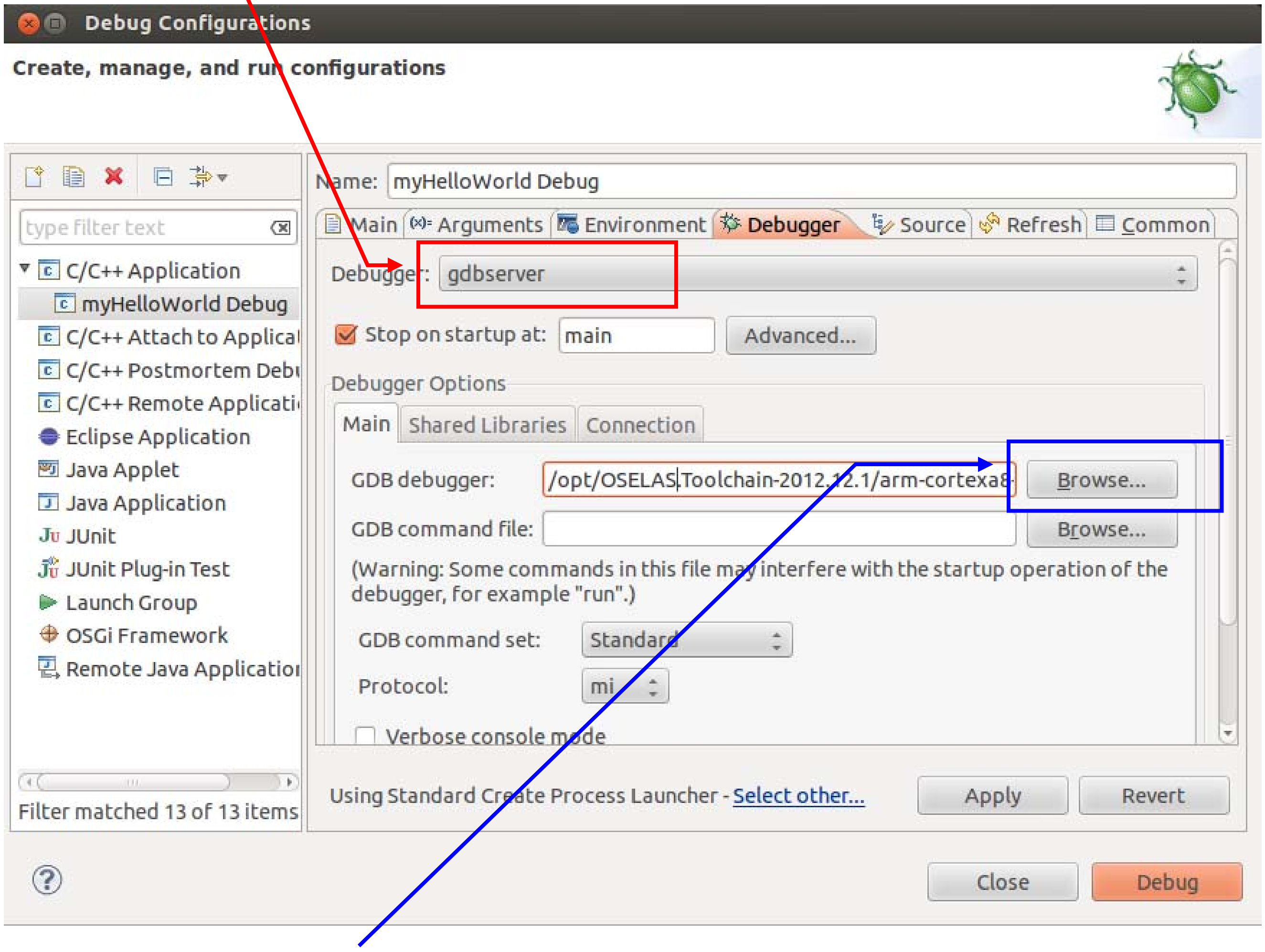
GDB command set: Standard

Protocol: mi

☐ Verbose console mode

Using Standard Create Process Launcher - Select other...

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



Name: myHelloWorld Debug

Debugger: gdbserver

Stop on startup at: main

GDB debugger: /opt/OSELAS.Toolchain-2012.12.1/arm-cortexa8

GDB command file:

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

GDB command set: Standard

Protocol: mi

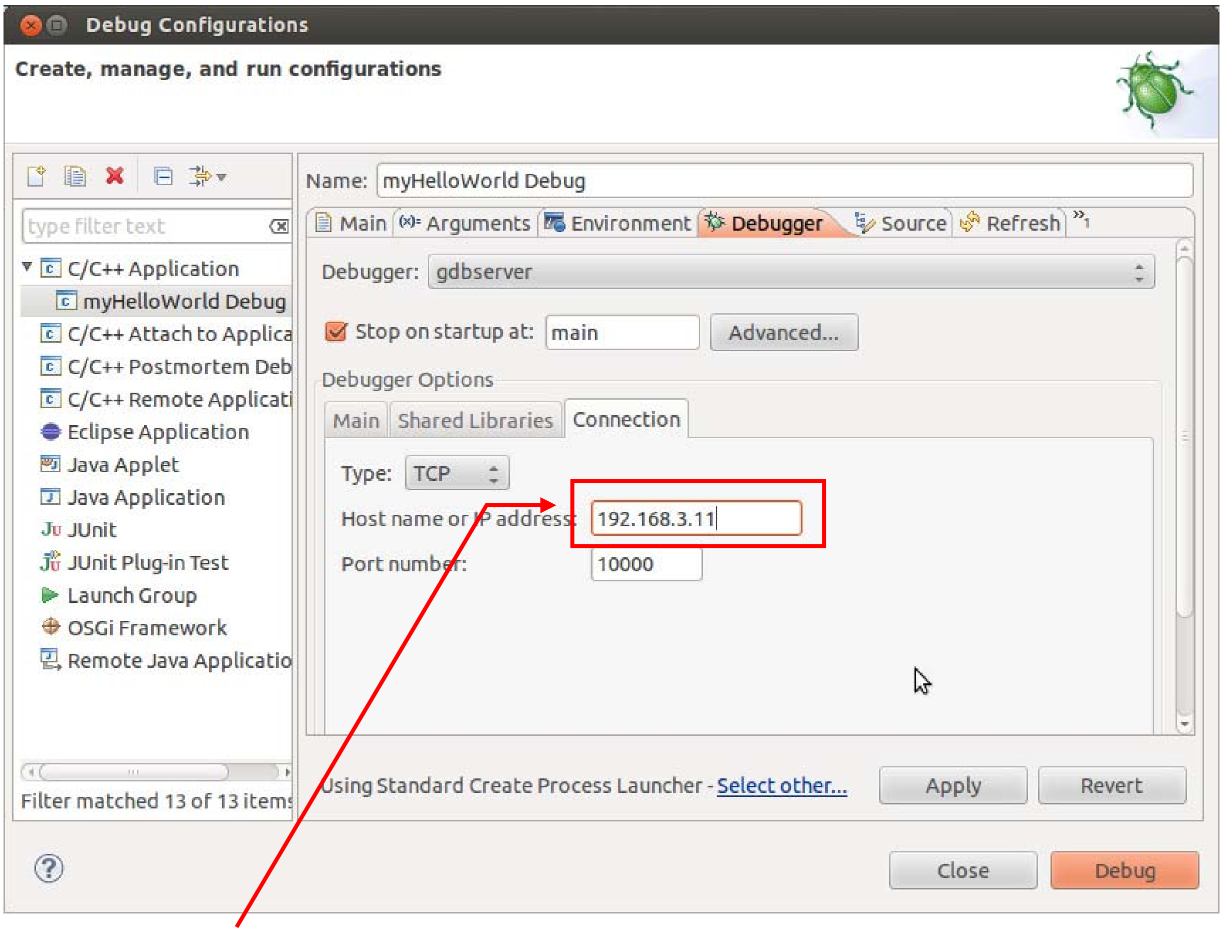
Verbose console mode

- Click the *Browse* button right beside the *GDB debugger* input field

A new dialog opens to choose the GDB executable.

- Click on *File System*
- Navigate to the directory */opt/OSELAS.Toolchain*/arm-cortexa8-linux-gnueabihf/gcc-4.7.3-glibc-2.16.0-binutils-2.22-kernel-3.6-sanitized/bin*.
- Select the file *arm-cortexa8-linux-gnueabihf-gdb*.
- Click *OK*
- Keep the *GDB command file* field empty

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



Name: myHelloWorld Debug
Debugger: gdbserver
Stop on startup at: main
Type: TCP
Host name or IP address: 192.168.3.11
Port number: 10000

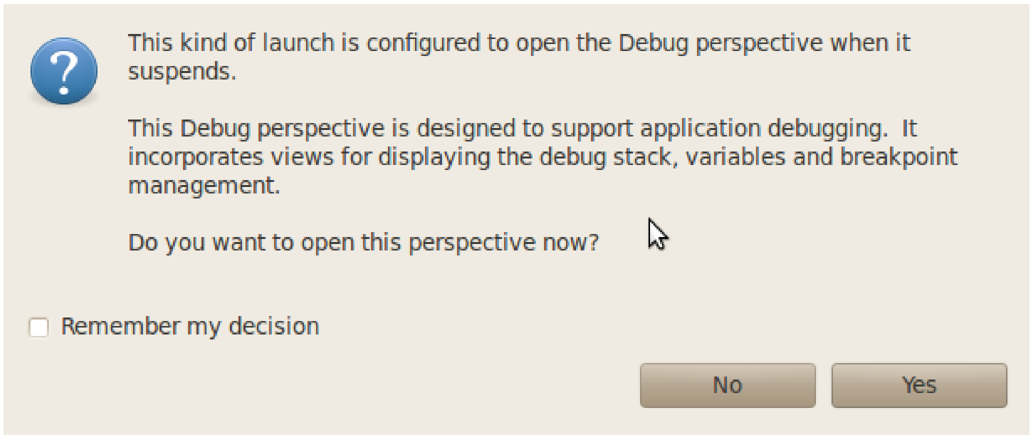
- Enter **192.168.3.11** (the target's IP address) in the *Host name* input field

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



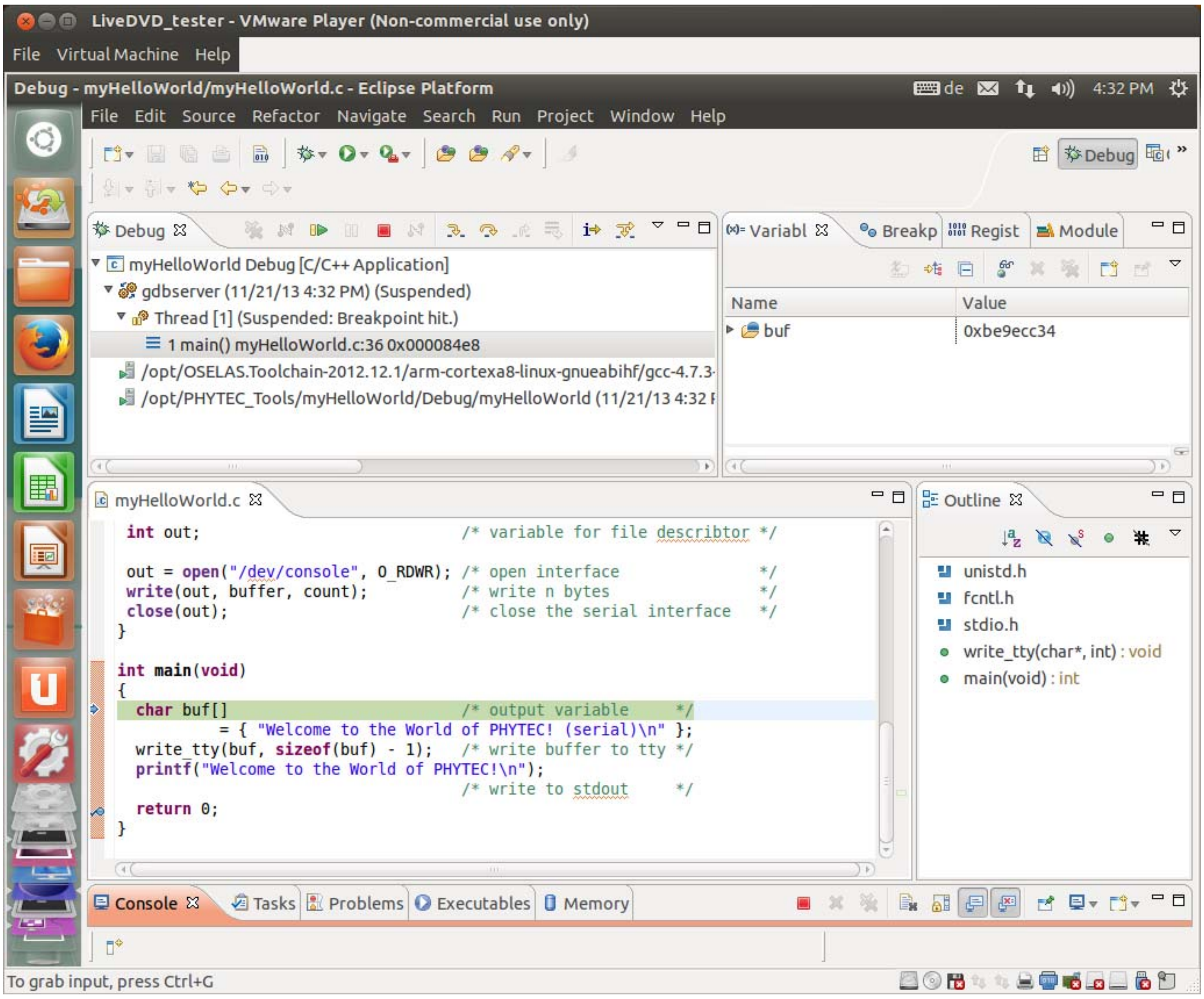
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

☐ Remember my decision

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.


LiveDVD_tester - VMware Player (Non-commercial use only)
File Virtual Machine Help
Debug - myHelloWorld/myHelloWorld.c - Eclipse Platform
de 4:32 PM
File Edit Source Refactor Navigate Search Run Project Window Help
Debug
Debug
Variabl Breakp Regist Module
myHelloWorld.c
Outline
Console Tasks Problems Executables Memory
To grab input, press Ctrl+G

```
myHelloWorld Debug [C/C++ Application]
  gdbserver (11/21/13 4:32 PM) (Suspended)
    Thread [1] (Suspended: Breakpoint hit.)
      1 main() myHelloWorld.c:36 0x000084e8
  /opt/OSELAS.Toolchain-2012.12.1/arm-cortexa8-linux-gnueabihf/gcc-4.7.3-
  /opt/PHYTEC_Tools/myHelloWorld/Debug/myHelloWorld (11/21/13 4:32 P
```

Name	Value
buf	0xbe9ecc34

```
  int out;                           /* variable for file describtor */

  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;
}
```

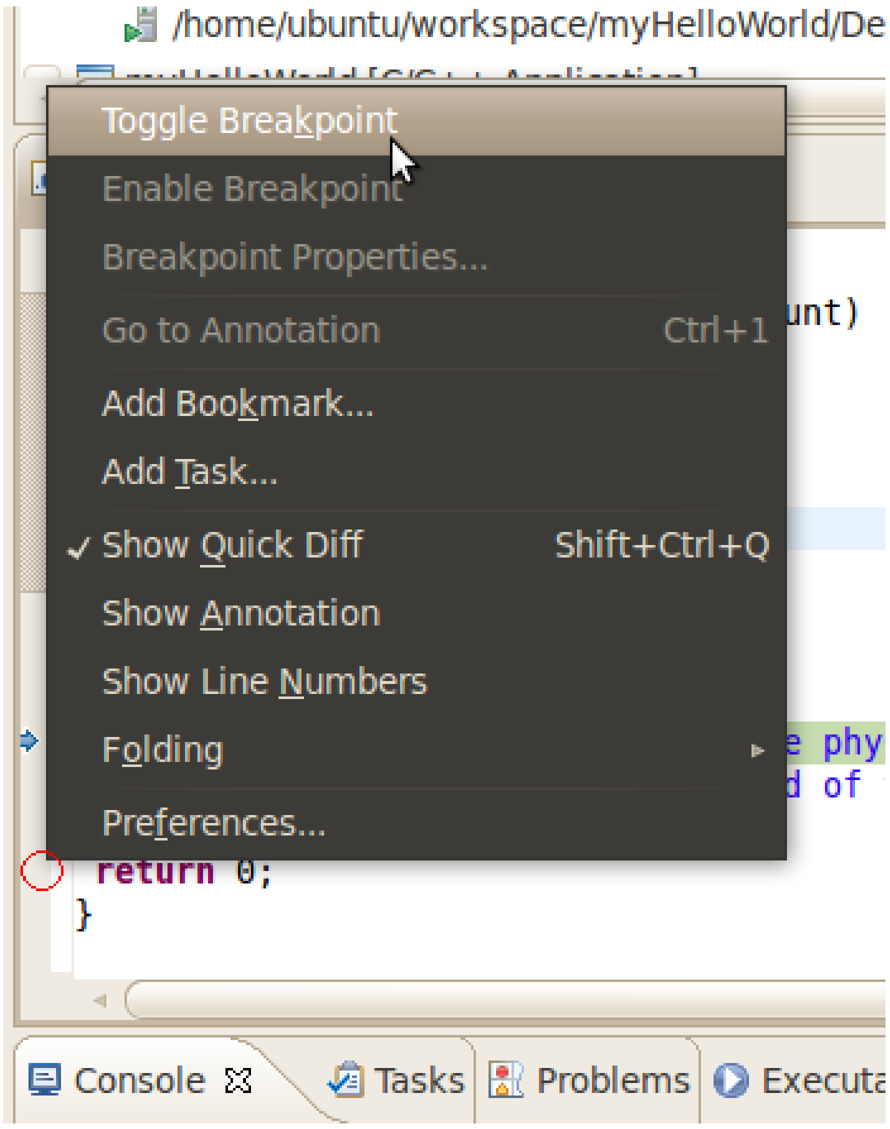
```
unistd.h
fcntl.h
stdio.h
write_tty(char*, int) : void
main(void) : int
```

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.

2.2.2.3 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 on 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

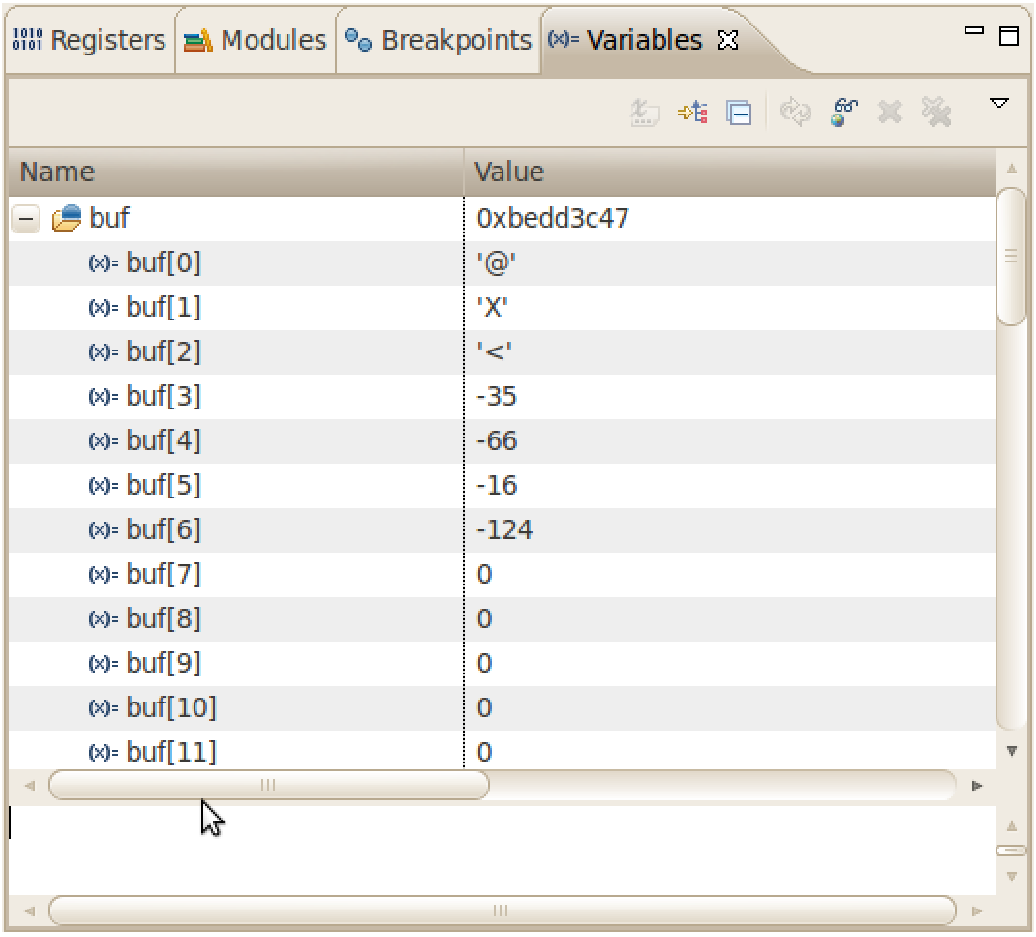


```
return 0;
}
```

2.2.2.4 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.

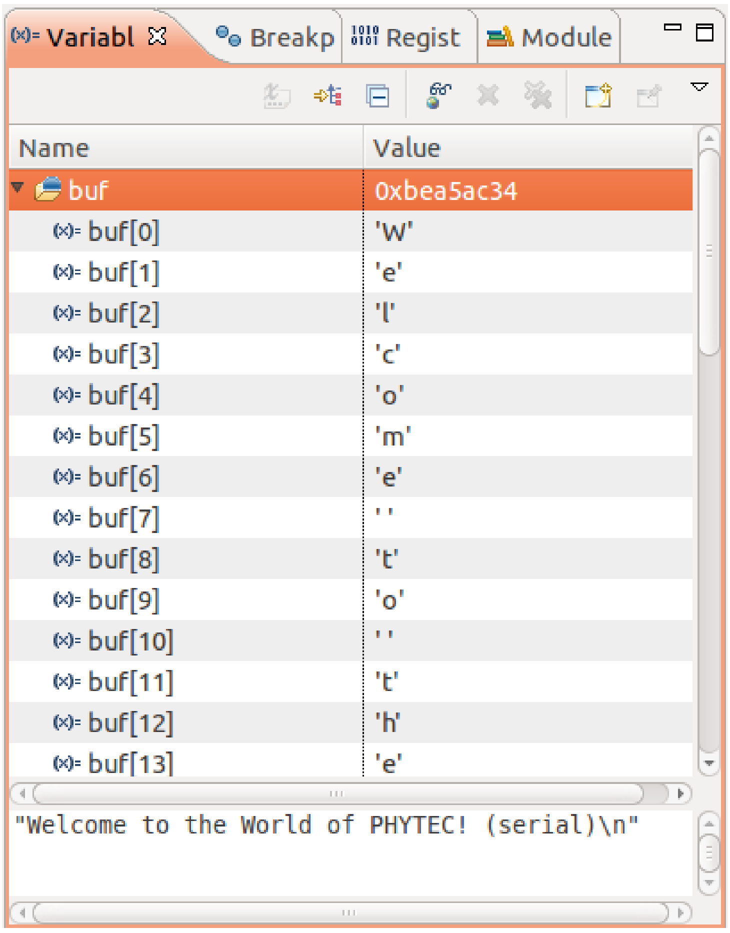
- Expand *buf* in the *Variables* window



Name	Value
buf	0xbedd3c47
buf[0]	'@'
buf[1]	'X'
buf[2]	'<'
buf[3]	-35
buf[4]	-66
buf[5]	-16
buf[6]	-124
buf[7]	0
buf[8]	0
buf[9]	0
buf[10]	0
buf[11]	0

- 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*



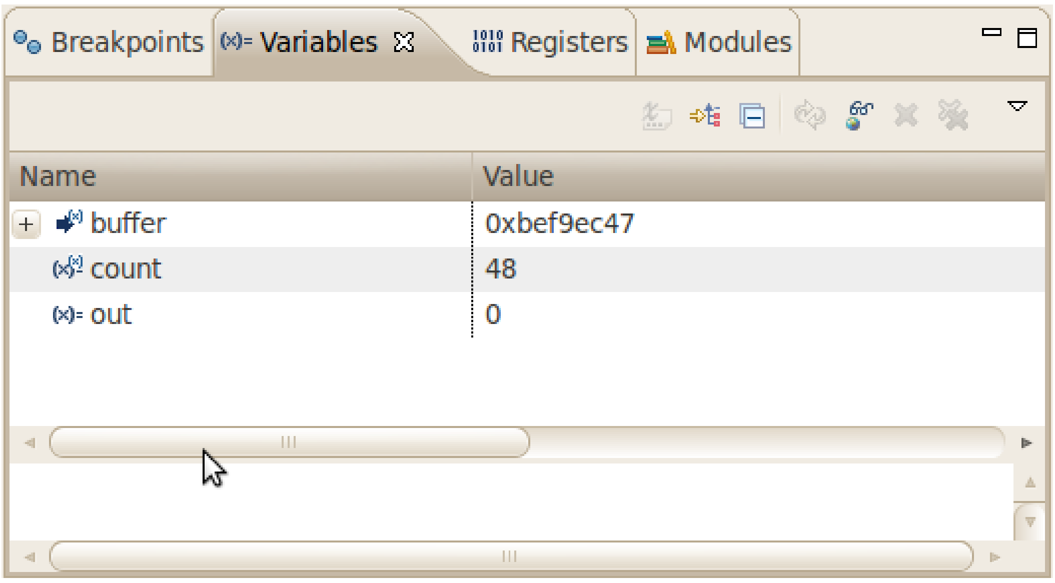
Name	Value
buf	0xbea5ac34
buf[0]	'W'
buf[1]	'e'
buf[2]	'l'
buf[3]	'c'
buf[4]	'o'
buf[5]	'm'
buf[6]	'e'
buf[7]	' '
buf[8]	't'
buf[9]	'o'
buf[10]	' '
buf[11]	't'
buf[12]	'h'
buf[13]	'e'

```
"Welcome to the World of PHYTEC! (serial)\n"
```

- 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:



Name	Value
buffer	0xbef9ec47
count	48
out	0

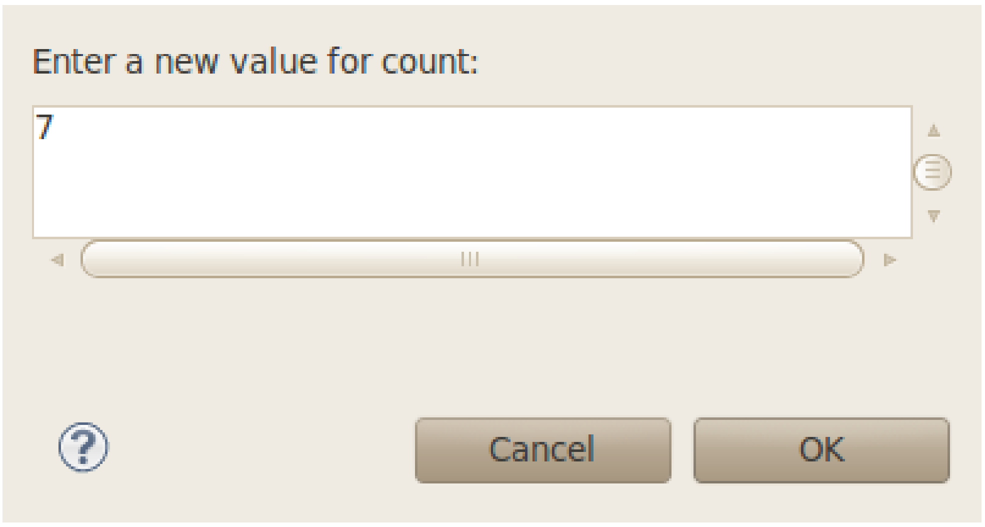
- Click on the variable *buffer*

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

2.2.2.5 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*



Enter a new value for count:

7

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

- Switch to *Microcom*

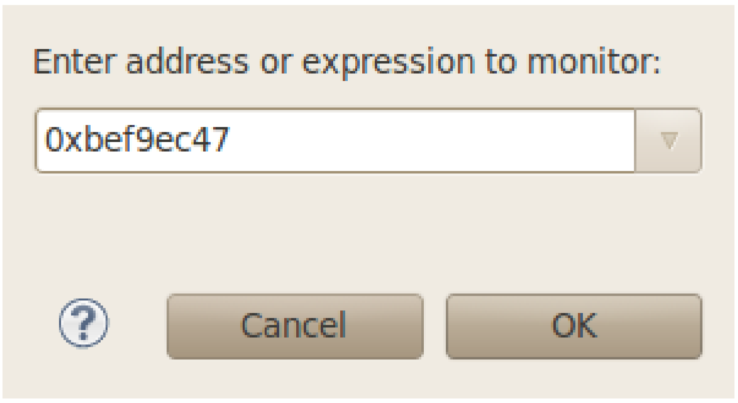
```
root@phyBOARD-WEGA-AM335x:~ gdbserver 192.168.3.11:10000 myHelloWorld
Process myHelloWorld created; pid = 827
Listening on port 10000
Welcome to the World of PHYTEC! (serial)
Remote debugging from host 192.168.3.10
Welcome
```

You will see the output *Welcome* in the Microcom window. This shows when changing the *counter* variable's value to 7 only the first seven characters of the buffer are displayed, instead of the whole sentence.

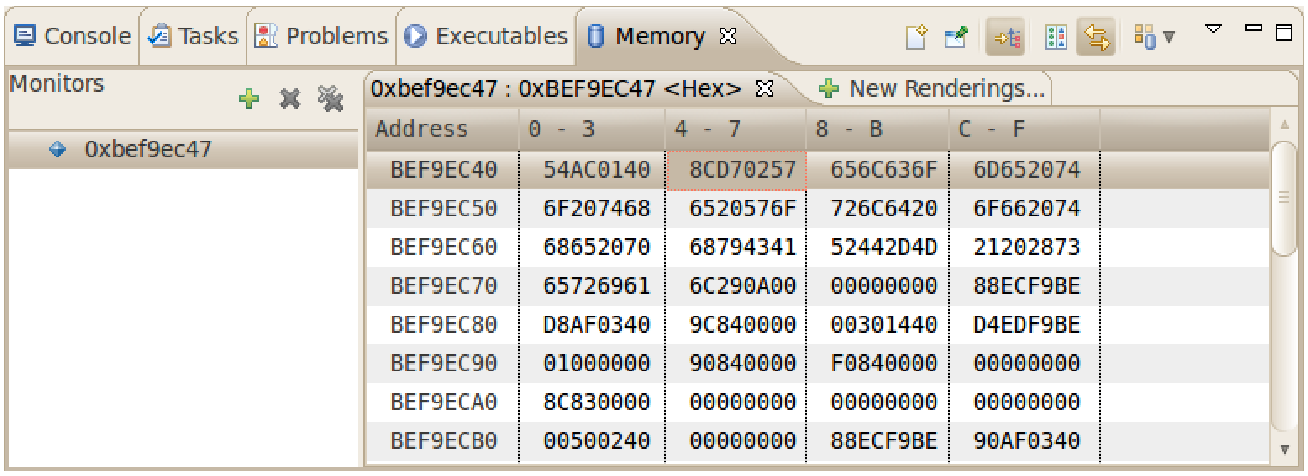
2.2.2.6 Using the Memory Monitor

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
- Click *Add Memory Monitor*
- Enter the address of buffer and click OK. Remember that the variable's address might be different on your system

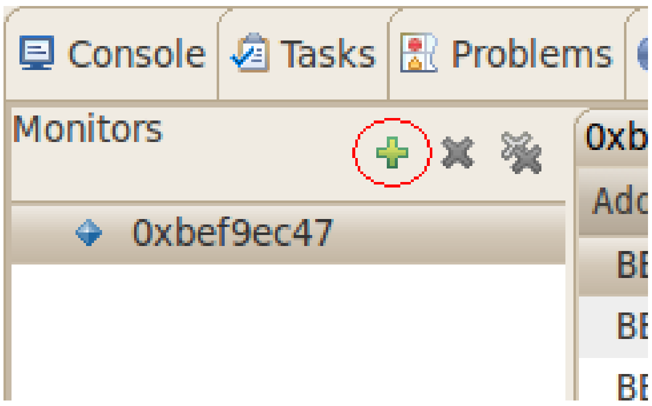


- Change the size of the window

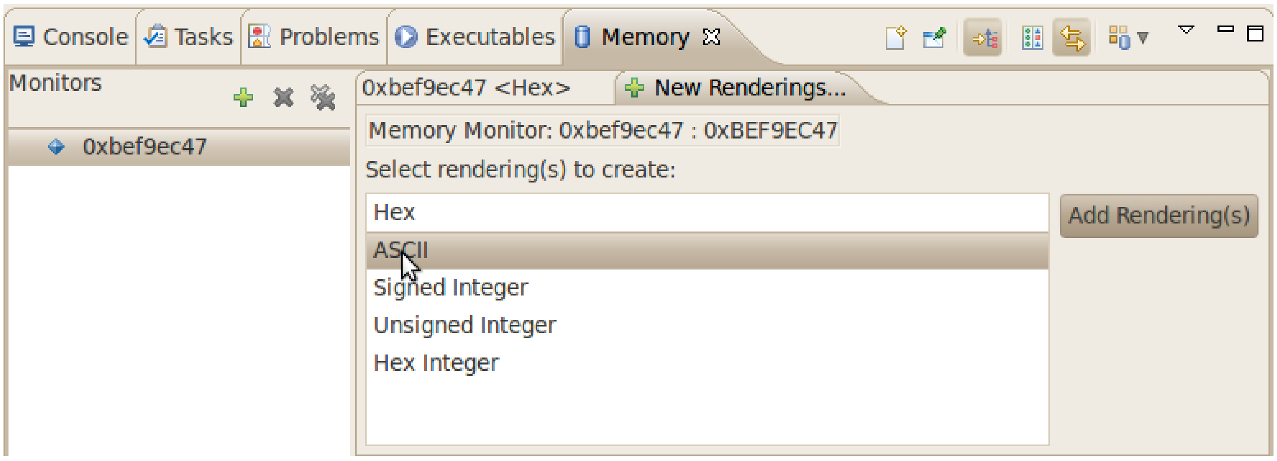


Address	0 - 3	4 - 7	8 - B	C - F
BEF9EC40	54AC0140	8CD70257	656C636F	6D652074
BEF9EC50	6F207468	6520576F	726C6420	6F662074
BEF9EC60	68652070	68794341	52442D4D	21202873
BEF9EC70	65726961	6C290A00	00000000	88ECF9BE
BEF9EC80	D8AF0340	9C840000	00301440	D4EDF9BE
BEF9EC90	01000000	90840000	F0840000	00000000
BEF9ECA0	8C830000	00000000	00000000	00000000
BEF9ECB0	00500240	00000000	88ECF9BE	90AF0340

- Click *Add Rendering*



- Select *ASCII* and click *OK*



Console | Tasks | Problems | Executables | Memory

Monitors

0xbef9ec47

0xbef9ec47 <Hex> | New Renderings...

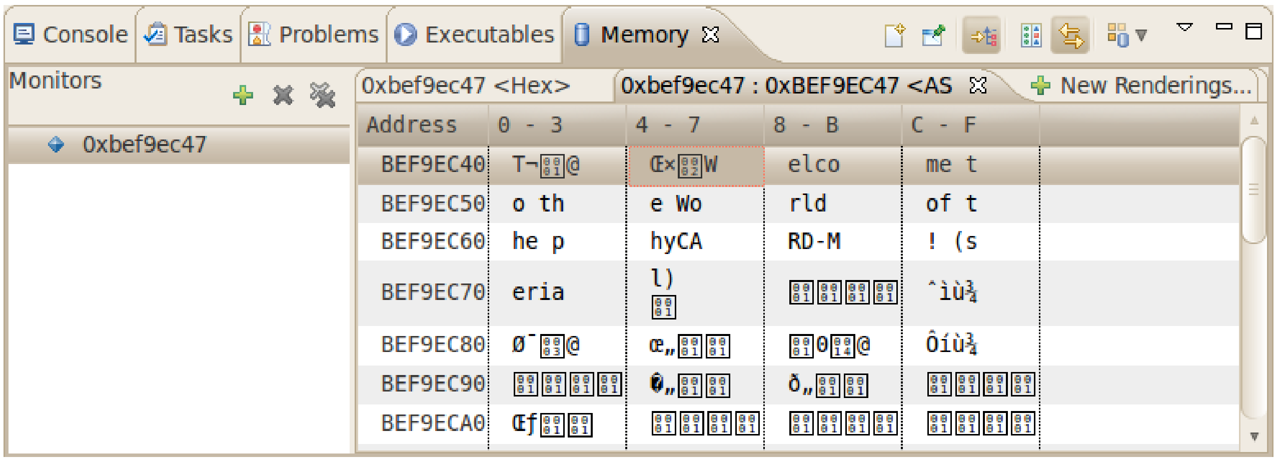
Memory Monitor: 0xbef9ec47 : 0xBEF9EC47

Select rendering(s) to create:

- Hex
- ASCII
- Signed Integer
- Unsigned Integer
- Hex Integer

Add Rendering(s)

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



Console | Tasks | Problems | Executables | Memory

Monitors

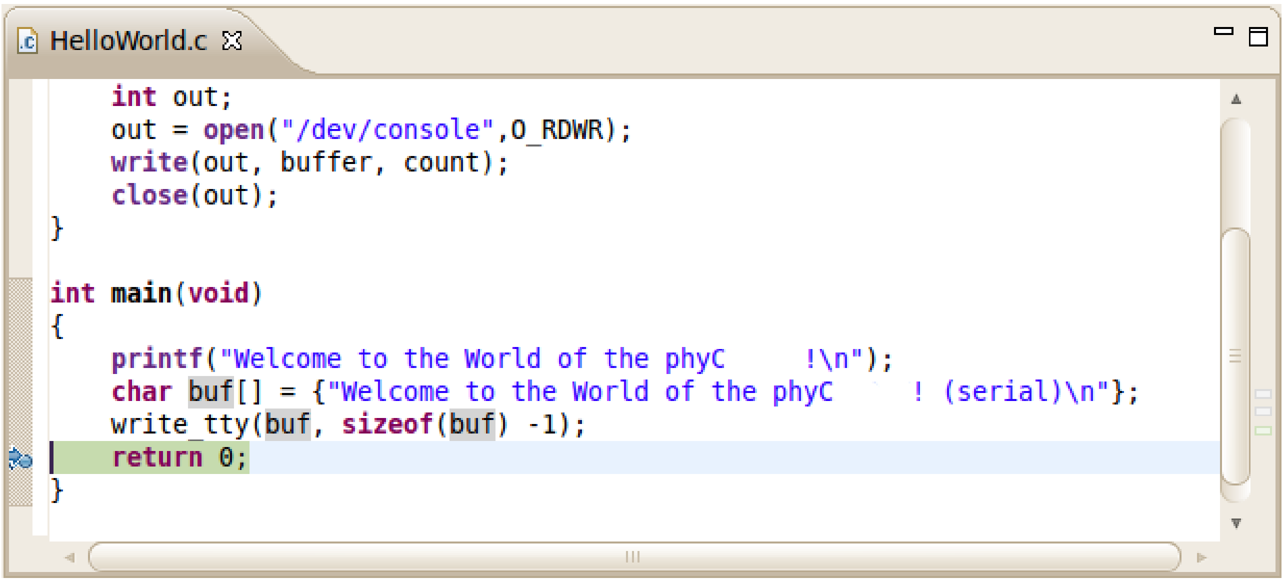
0xbef9ec47

0xbef9ec47 <Hex> | 0xbef9ec47 : 0xBEF9EC47 <AS | New Renderings...

Address	0 - 3	4 - 7	8 - B	C - F
BEF9EC40	T¬▯@	Œx▯W	elco	me t
BEF9EC50	o th	e Wo	rld	of t
BEF9EC60	he p	hyCA	RD-M	! (s
BEF9EC70	eria	l) ▯	▯▯▯▯	ˆìù¾
BEF9EC80	ø¯▯@	œ„▯▯	▯0▯@	Ôíù¾
BEF9EC90	▯▯▯▯	�„▯▯	ð„▯▯	▯▯▯▯
BEF9ECA0	Œf▯▯	▯▯▯▯	▯▯▯▯	▯▯▯▯

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

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



```
    int out;
    out = open("/dev/console",O_RDWR);
    write(out, buffer, count);
    close(out);
}

int main(void)
{
    printf("Welcome to the World of the phyC     !\n");
    char buf[] = {"Welcome to the World of the phyC     ! (serial)\n"};
    write_tty(buf, sizeof(buf) -1);
    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.

	Knowing how to work with Eclipse and how to develop and execute an application on the phyBOARD-Wega-Am335x, you are now ready prepared to start your project. The following section will give you detailed information on the different features and interfaces of the phyBOARD-Wega and how to use them within your application. *If your project is more complex, or if you crave more information about working with the BSP, continue with chapter 4. Chapter 4ff inlcude step by step instructions on how to modify and download the BSP using PTXdist. They also include system level information on the phyBOARD-Wega-AM335x.*

3 Accessing the phyBOARD-Wega Interfaces

PHYTEC phyBOARD-Wega is fully equipped with all mechanical and electrical components necessary for the speedy and secure start-up and subsequent communication to and programming of applicable PHYTEC System on Module (SOM) modules. phyBOARD-Wega Boards are designed for evaluation, testing and prototyping of PHYTEC System on Module in laboratory environments prior to their use in customer designed applications.

3.1 Concept of the phyBOARD-Wega

The phyBOARD-Wega provides a flexible development platform enabling quick and easy start-up and subsequent programming of its soldered phyCORE-AM335x System on Module. The carrier board design allows easy connection of additional extension boards featuring various functions that support fast and convenient prototyping and software evaluation. The carrier board is compatible with phyCORE-AM335x only.

This modular development platform concept includes the following components:

- the **phyCORE-AM335x module** populated by default with the AM3354 processor and all applicable SOM circuitry such as DDR SDRAM, Flash, PHYs, and transceivers to name a few.

- the **phyBOARD-Wega** which offers all essential components and connectors for start-up including: A power socket which enables connection to an **external power adapter**, interface connectors such as **DB-9**, **USB** and **Ethernet** allowing for use of the SOM's interfaces with standard cable.

The following sections contain specific information relevant to the operation of the phyCORE-AM335x mounted on the phyBOARD-Wega Carrier Board.

3.2 Overview of the phyBOARD-Wega Peripherals

The phyBOARD-Wega is depicted in *Figure 1*. It features many different interfaces and is equipped with the components as listed in *Table 2*, and *Table 3*. For a more detailed description of each peripheral refer to the appropriate chapter listed in the applicable table.*Figure 1* highlights the location of each peripheral for easy identification.

3.2.1 Connectors and Pin Header

Table 2 lists all available connectors on the phyBOARD-Wega. *Figure 1* highlights the location of each connector for easy identification.

Reference Designator	Description	See Section
X11	Secure Digital / Multi Media Card (Micro-slot)	*3.3.8*
X15	USB Host connector (USB 2.0 Standard-A)	*3.3.4*
X16	Ethernet 0 connector (RJ45 with speed and link LED)	*3.3.3*
X17	Ethernet 1 connector (RJ45 with speed and link LED)	*3.3.3*
X42	USB On-The-Go connector (USB Micro-AB)	*3.3.4*
X55	Mono Speaker output (2-pole Molex)	*3.3.5*
X65	CAN connector (5×2 pin header)	*0*
X66	RS-232 with RTS and CTS (UART1 5×2 pin header)	*3.3.2*
X67	Power supply 5 V only (via 6-pole WAGO male header, or 2-pole PHOENIX base strip)	*3.3.1.1*
X69	Expansion connector (2×30 socket connector)	*3.3.7*
X70	A/V connector #1 (2×20 socket connector)	*3.3.6*
X71	A/V connector #2 (2×8 socket connector)	*3.3.6*
X72	Optional 5 V power supply via USB Micro-AB connector	*3.3.1.1*
X73	Stereo Line Out and Line In connector (2×3 pin header)	*3.3.5*

Table 2: phyBOARD-Wega Connectors and Pin Headers

Note:
Ensure that all module connections are not to exceed their expressed maximum voltage or current. Maximum signal input values are indicated in the corresponding controller User's Manual/Data Sheets. As damage from improper connections varies according to use and application, it is the user‘s responsibility to take appropriate safety measures to ensure that the module connections are protected from overloading through connected peripherals.

3.2.2 LEDs

The phyBOARD-Wega is populated with three LEDs to indicate the status of the USB VBUS voltages, as well as of the power supply voltage.

Figure 1 shows the location of the LEDs. Their function is listed in the table below:

LED	Color	Description	See Section
D7	green	Indicates presence of VBUS1 at the USB Host interface	*3.3.4*
D8	green	Indicates presence of VBUS0 at the USB OTG interface	*3.3.4*
D58	red	3.3 V voltage generation of the phyBOARD-Wega	*3.3.1.2*

Table 3: phyBOARD-Wega LEDs Descriptions

Note:
Detailed descriptions of the assembled connectors, jumpers and switches can be found in the following chapters.

3.3 Functional Components on the phyBOARD-Wega Board

This section describes the functional components of the phyBOARD-Wega. Each subsection details a particular connector/interface and associated jumpers for configuring that interface.

3.3.1 Power Supply

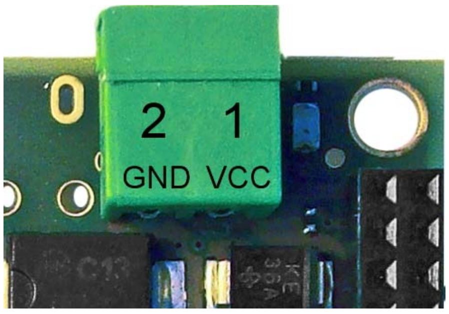
Caution:
Do not change modules or jumper settings while the phyBOARD-Wega is supplied with power!

3.3.1.1 Power Connectors (X67 and X72)

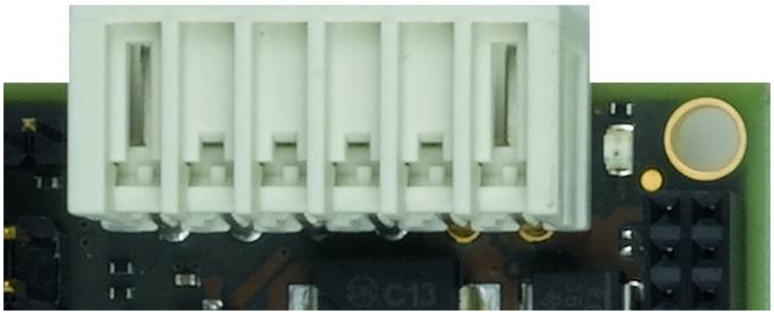
The phyBOARD-Wega is available with three different power supply connectors. Depending on your order you will find one of the following connectors on your SBC:

1. a 2-pole PHOENIX base strip 3.5 mm connector suitable for a single 5 V supply voltage, or
2. an USB Micro-AB connector to connect a standard 5 V USB power supply, or
3. a 6-pole WAGO male header to attach the Power Module for phyBOARD-Wega (PEB-POW-01) which provides connectivity for 12 V – 24 V

The required current load capacity for all power supply solutions depends on the specific configuration of the phyCORE mounted on the phyBOARD-Wega the particular interfaces enabled while executing software as well as whether an optional expansion board is connected to the carrier board. A 5 V adapter with a minimum supply of 1.5 A is recommended.



PHOENIX base strip



WAGO male header 6-pole

Figure 3: Power Supply Connectors

3.3.1.1.1 PHOENIX 2-pole Base Strip

The permissible input voltage is +5 V DC if your SBC is equipped with a 2-pole PHOENIX connector.

Figure 3 and the following table show the pin assignment.

Pin	Signal	Description
1	**VCC5V_IN**	**+5V power supply**
2	**GND**	**Ground**

Table 4: Pin Assignment of the 2-pole PHOENIX Connector at X67

3.3.1.1.2 USB Micro-AB

If your board provides an USB Micro-AB female connector at the upper side of the board a standard USB Micro power supply with +5 V DC can be used to supply the phyBOARD-Wega.

Caution!
Do not confuse the USB Micro connector on the upper side of the board with the one on the back side of the board which provides USB OTG connectivity. The USB Micro connector on the upper side is exclusively used for power supply and has no other USB functionality!

3.3.1.1.3 WAGO 6-pole Male Header

If a WAGO 6-pole male header is mounted on your board (see *Figure 1* and *Figure 3*) your board is prepared to connect to a phyBOARD-Wega Power Module (PEB-POW-01), or a custom power supply circuitry. The mating connector from WAGO has the EAN 4045454120610.

Use of the 6-pole connector has the following advantages:

- Higher and wider operate range of the input voltage
- External scaling potential to optimize the electrical output current
- 5 V, 3.3 V and backlight power supply

Pin assignment of the 6 –pole WAGO connector:

Pin	Signal	Description
1	**VCC5V_IN**	**+5 V power supply**
2	**GND**	**Ground**
3	**VCC3V3_PMOD**	**+3.3 V power supply**
4	**VCC_BL**	**Backlight power supply**
5	**PMOD_PWRGOOD**	**Power good signal (connected to reset nRESET_IN)**
6	**PMOD_PWRFAIL**	**Power fail signal**

Table 5: Pin Assignment of the 6-pole WAGO Connector at X67

A detailed description of the Power Module for phyCORE-Wega can be found in the Application Guide for phyBOARD-Wega-AM335x Expansion Boards (L-793e_0).

3.3.1.2 Power LED D58

The red LED D58 right next to the power connector (see *Figure 1*) indicates the presence of the 3.3 V supply voltage generated from the 5 V input voltage.

3.3.1.3 VBAT and RTC

On the phyBOARD-Wega the internal RTC of the AM335x is used for real-time or time-driven applications. To backup the RTC on the module, a goldcap (C339) is placed on the phyBOARD-Wega. This voltage source supplies the backup voltage domain VBAT of the AM335x which supplies the RTC and some critical registers when the primary system power, VCC5V_IN, is removed. The backup supply lasts approximately 17 ½ days.

3.3.2 RS-232 Connectivity (X66)

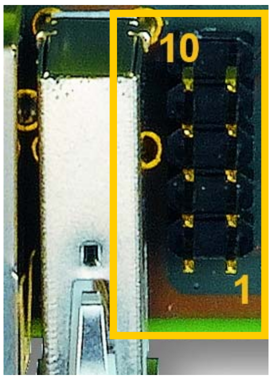


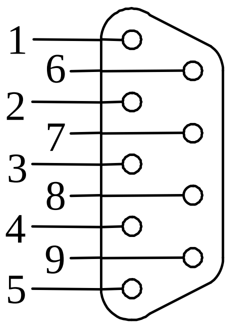
Figure 4: RS-232 Interface Connector X66

Pin header connector X66 located next to the USB host connector (see *Figure 1)* provides the UART1 signals of the AM335x at RS-232 level. The serial interface is intended to be used as data terminal equipment (DTE) and allows for a 5-wire connection including the signals RTS and CTS for hardware flow control. *Table 6* below shows the signal mapping of the RS-232 level signals at connector X66.

Pin	Signal	Pin	Signal
1	**NC**	2	**NC**
3	**UART1_RXD_RS232**	4	**UART1_RTS_RS232**
5	**UART1_TXD_RS232**	6	**UART1_CTS_RS232**
7	**NC**	8	**NC**
9	**GND**	10	**NC**

Table 6: Pin Assignment of RS-232 Connector X66

An adapter cable is included in the phyBOARD-Wega-Am335x Kit to facilitate the use of the UART1 interface. The following figure shows the signal mapping of the adapter.



Pin 2:	RxD-RS232
Pin 7:	RTS-RS232
Pin 3:	TxD-RS232
Pin 8:	CTS-RS232
Pin 5:	GND

Figure 5: RS-232 Connector Signal Mapping

Note:
The UART0 interface which is required for debugging is routed to expansion connector X69. The Evaluation Board (PEB-EVAL-01) delivered with the kit allows easy use of this interface. Please find additional information on the Evaluation Board in Application Guide for phyBOARD-Wega-AM335x Expansion Boards (L-793e_0)

3.3.2.1 Software Implementation

The AM335x SOM supports up to 6 so called UART units. On the phyBOARD-Wega UART1 is routed to pin header connector X66. UART0 and UART2 are routed to the expansion connector X69.

Only */dev/ttyO0* for UART0 and */dev/ttyO1* for UART1 have been implemented within the BSP. */dev/ttyO0* is the standard console, mainly used for debugging and control of software updates. */dev/ttyO1* is intended for use by customer's applications.

Usage of */dev/ttyO2* for UART2 */dev/ttyO5* for UART5 would need some development.

3.3.3 Ethernet Connectivity (X16 and X17)

The Ethernet interfaces of the phyBOARD-Wega are accessible at two RJ45 connectors (X16 and X17) on the board.

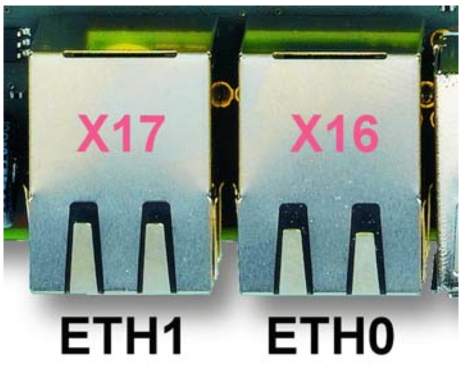


Figure 6: Ethernet Interfaces at Connectors X16 and X17

Ethernet 0 interface is available at X16, while Ethernet 1 interface is brought out at X17.

Both Ethernet interfaces are configured as 10/100Base-T networks. The LEDs for LINK (green) and SPEED (yellow) indication are integrated in the connector. Both LAN8710AI Ethernet transceivers support HP Auto-MDIX technology, eliminating the need for the consideration of a direct connect LAN cable, or a cross-over path cable. They detect the TX and RX pins of the connected device and automatically configure the PHY TX and RX pins accordingly.

3.3.3.1 Software Implementation

The two 10/100 Mbit Ethernet interfaces are being provided as network interfaces *eth0* with static IP 192.168.3.11 and *eth1* with static IP 192.168.4.11 by default.

Both interfaces offer a standard Linux network port which can be programmed using the BSD socket interface.

3.3.4 USB Connectivity (X15 and X42)

The phyBOARD-Wega provides one USB Host and one USB OTG interface.

USB0 is accessible at connector X42 (USB Micro-AB) located at the back side of the phyBOARD-Wega. It is configured as USB OTG. USB OTG devices are capable to initiate a session, control the connection and exchange host and peripheral roles between each other. This interface is compliant with USB revision 2.0.

USB1 is accessible on the top at connector X15 (USB Standard-A) and is configured as USB Host.

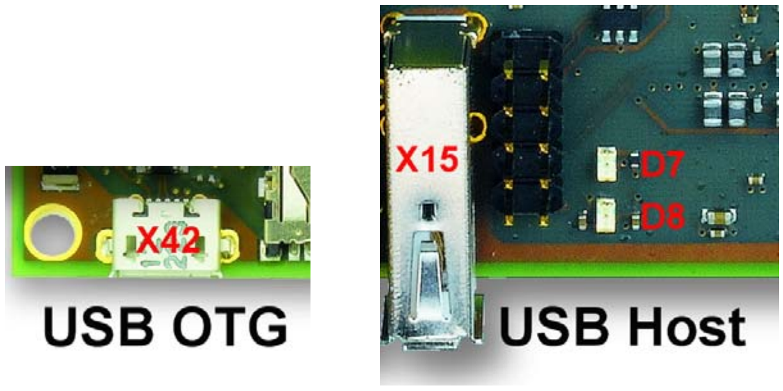


Figure 7: Components supporting the USB Interfaces

LED D8 displays the status of USB0_VBUS and LED D7 the status of USB1_VBUS.

For later expansion boards the USB1 interface can be routed to the expansion connector (X69) by populating J72 and J73 (2+3).

3.3.4.1 Software Implementation

3.3.4.1.1 USB Host

The AM335x CPU embeds a USB 2.0 EHCI controller that is also able to handle low and full speed devices (USB 1.1).

The BSP includes support for mass storage devices and keyboards. Other USB related device drivers must be enabled in the kernel configuration on demand.

Due to *udev*, connecting various mass storage devices get unique IDs and can be found in */dev/disk/by-id*. These IDs can be used in */etc/fstab* to mount different USB memory devices in a different way.

3.3.4.1.2 USB OTG

The BSP includes support for USB OTG. In order to activate it you need to load an appropriate module with *modprobe*, for example the mass storage gadget *g_file_storage*, which lets the phyBOARD-Wega behave like an USB stick:

```
dd if=/dev/zero of=/tmp/file.img bs=1M count=64
modprobe g_file_storage file=/tmp/file.img
```

After connecting the USB OTG port to the USB host port of any other Linux computer, use the following command in order to create a file system (replace *sdc1* by the device the computer really shows):

```
sudo mkfs.vfat /dev/sdc1
```

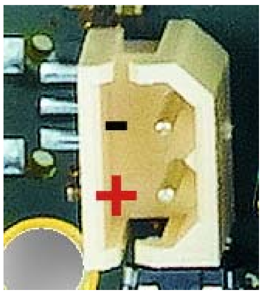
The BSP doesn’t provide support for using USB OTG as host.

3.3.5 Audio Interface (X55 and X73)

The audio interface provides a method of exploring AM335x's audio capabilities. The phyBOARD-Wega is populated with an audio codec at U35. The audio codec is connected to the AM335x's McASP0 interface to support stereo line input and stereo line output at connector X73. In addition to that the phyBOARD-Wega has one direct mono speaker output (2 × 1 W) at the Molex connector X55.

X55:

Speaker



X73:

Line In
Line Out

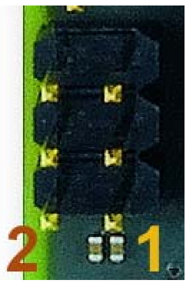


Figure 8: Audio Interfaces at Connectors X55 and X73

Pin assignment at X73

Pin	Signal	Pin	Signal
1	**LINE_IN_L**	2	**LINE_IN_R**
3	**AGND**	4	**AGND**
5	**LINE_OUT_L**	6	**LINE_OUT_R**

Pin assignment at X55

Pin	Signal	Description
1	**SPOP**	**Class-D positive differential output**
2	**SPOM**	**Class-D negative differential output**

For additional audio applications the McASP0 interface of the AM335x including the signals X_MCASP0_AHCLKX, X_I2S_CLK, X_I2S_FRM, X_I2S_ADC and X_I2S_DAC are routed to the A/V connector X71 (please refer to *section 4.6.1* for additional information on the A/V connector).

Please refer to the audio codec's reference manual for additional information regarding the special interface specification.

3.3.5.1 Software Implementation

Audio support on the module is done via the I2S interface and controlled via I2C.

On the phyBOARD-Wega the audio codec's registers can be accessed via the I2C0 interface at address 0x18 (7-bit MSB addressing).

As of the printing of this manual the BSP delivered with the phyCARD-Wega-Am335x does not support the audio interfaces. Please visit the PHYTEC website for a road map of the BSP.

CAN Connectivity

The CAN1 interface of the phyBOARD-Wega-AM335x is accessible at connector X65 (2×5 2.54 mm pin header).

Jumper JP3 can be installed to add a 120 Ohm termination resistor across the CAN data lines if needed.

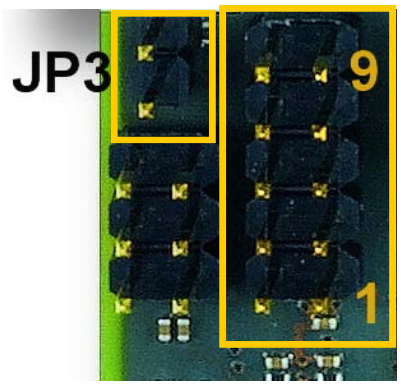


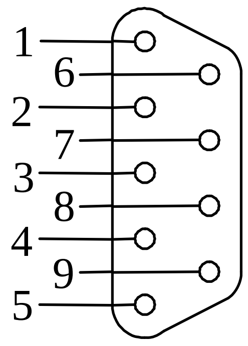
Figure 9: Components supporting the CAN Interface

Table 7 below shows the signal mapping of the CAN1 signals at connector X65.

Pin	Signal	Pin	Signal
1	**NC**	2	**GND**
3	**X_CANL**	4	**X_CANH**
5	**GND**	6	**NC**
7	**NC**	8	**NC**
9	**Shield**	10	**NC**

Table 7: Pin Assignment of CAN Connector X65

An adapter cable is included in the phyBOARD-Wega-Am335x Kit to facilitate the use of the CAN interface. The following figure shows the signal mapping of the adapter.



Pin 6:	GND
Pin 2:	X_CANL
Pin 7:	X_CANH
Pin 3:	GND
Pin 5:	Shield

Figure 10: CAN Connector Signal Mapping

3.3.5.2 Software Implementation

The phyCORE-AM335x provides a CAN feature, which is supported by drivers using the proposed Linux standard CAN framework ”Socket-CAN”. Using this framework, CAN interfaces can be programmed with the BSD socket API.

The CAN (Controller Area Network) bus offers a low-bandwidth, prioritized message fieldbus for communication between microcontrollers. Unfortunately, CAN was not designed with the ISO/OSI layer model in mind, so most CAN APIs available throughout the industry don’t support a clean separation between the different logical protocol layers, like for example known from ethernet.

The Socket-CAN framework for Linux extends the BSD socket API concept towards CAN bus. The Socket-CAN interface behaves like an ordinary Linux network device, with some additional features special to CAN. Thus for example you can use

```
ifconfig –a
```

in order to see if the interface is up or down, but the given MAC and IP addresses are arbitrary and obsolete.

Configuration happens within the script */etc/network/can-pre-up*. This script will be called when */etc/init.d/networking* is running at system start up. To change default used bitrates on the target change the variable *BITRATE* in */etc/network/can-pre-up*.

For a persistent change of the default bitrates change the local *projectroot/etc/network/can-pre-up* instead and rebuild the BSP.

The information for *can0* (which is used for AM335x's CAN1 routed to X65) looks like

```
root@phyCORE-AM335x:~ ifconfig can0
can0       Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-
00-00-00-00-00-00
          inet addr:127.42.23.180  Mask:255.255.255.0
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:1284643 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67450 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:5138560 (4.9 MiB)  TX bytes:269767 (263.4 KiB)
          Interrupt:211
```

The output contains a standard set of parameters also shown for Ethernet interfaces, so not all of these are necessarily relevant for CAN (for example the MAC address). The following output parameters contain useful information:

Field	**Description**
can0	Interface Name
NOARP	CAN cannot use ARP protocol
MTU	Maximum Transfer Unit
RX packets	Number of Received Packets
TX packets	Number of Transmitted Packets
RX bytes	Number of Received Bytes
TX bytes	Number of Transmitted Bytes
errors...	Bus Error Statistics

You can send messages with *cansend* or receive messages with *candump*:

```
cansend can0 0x03 0x12 0x06
candump can0
```

See *cansend --help* and *candump --help* help messages for further information about using and options.

3.3.6 Audio/Video connectors (X70 and X71)

The Audio/Video (A/V) connectors X70 and 71 provide an easy way to add typical A/V functions and features to the phyBOARD-Wega. Standard interfaces such as parallel display, I²S and I^2C as well as different supply voltages are available at the two A/V female dual entry connectors. Special feature of these connectors are their connectivity from the bottom or the top.

For further information of the A/V connectors see *chapter 4.6.1*. Information on the expansion boards available for the A/V Connecctors can be found in the Application Guide for phyBOARD-Wega Expansion Boards (L-793e_0).

3.3.7 Expansion connector (X69)

The expansion connector X69 provides an easy way to add other functions and features to the phyBOARD-Wega. Standard interfaces such as JATG, UART, MMC2, SPI and I^2C as well as different supply voltages and some GPIOs and Analog Inputs are available at the expansion female connector. The pinout of the expansion connector is shown in Secure Digital Memory Card/ MultiMedia Card (X11)
.

For further information of the expansion connector see see *chapter 4.6.5*. Information on the expansion boards available for the expansion conneccctor can be found in the Application Guide for phyBOARD-Wega Expansion Boards (L-793e_0)

3.3.8 Secure Digital Memory Card/ MultiMedia Card (X11)

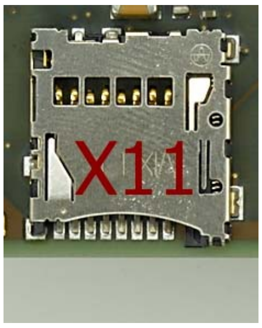


Figure 11: SD / MM Card interface at connector X11

The phyBOARD-Wega provides a standard microSDHC card slot at X11 for connection to SD/MMC interface cards. It allows easy and convenient connection to peripheral devices like SD- and MMC cards. Power to the SD interface is supplied by inserting the appropriate card into the SD/MMC connector, who features a card detection, a lock mechanism and a smooth extraction function by Push-in /-out of card.

The AM335x processor on the phyBAORD-Wega can boot from this interface.

3.3.8.1 Software Implementation

The phyBOARD-Wega supports a slot for Micro Secure Digital Cards and Micro Multi Media Cards to be used as general purpose blockdevices. These devices can be used in the same way as any other blockdevice.

⚠	These kind of devices are hot pluggable, so you must pay attention not to unplug the device while it's still mounted. This may result in data loss.

After inserting an MMC/SD card, the kernel will generate new device nodes in */dev*. The full device can be reached via its */dev/mmcblk0* device node, MMC/SD card partitions will occur in the following way:

```
/dev/mmcblk0p<Y>
```

<Y> counts as the partition number starting from 1 to the max count of partitions on this device.

⚠	These partition device nodes will only occur if the card contains a valid partition table (”hard disk” like handling). If it does not contain one, the whole device can be used for a filesystem (”floppy” like handling). In this case */dev/mmcblk0* must be used for formatting and mounting.

The partitions can be formatted with any kind of filesystem and also handled in a standard manner, e.g. the *mount* and *umount* command work as expected.

⚠	The cards are always mounted as being writable. Setting of write-protection of MMC/SD cards is not recognized.

3.3.8.2 Adressing the RTC

The RTC RV-4162-C7 on the phyCORE-AM335x can be accessed as */dev/rtc0*. It also has the entry */sys/class/rtc/rtc0* in the sysfs file system, where you can read for example the *name*.

Date and time can be manipulated with the *hwclock* tool, using the *-w* (systohc) and *-s* (hctosys) options. For more information about this tool refer to the manpage of *hwclock*.

To set the date first use *date* (see *man date* on the PC) and then run *hwclock -w -u* to store the new date into the RTC.

Please note that in case you need to use RTC's interrupt, you need to shortcut pin 37 *x_intr1* and pin 40 *x_intr_RTCn* of the expansion connector X69. But this can only be done if you don't need this interrupt line for a touch controller.

3.3.9 Boot Mode

The pyhBOARD-Wega has a defined boot sequence:

1. NAND
2. SD/MMC

The exact choosen boot mode in the processor is SYSBOOT[4:0] = 10011b : NAND, NANDI2C, MMC0, UART0

3.3.10 CPU core frequency Scaling

The phyCORE-AM335x supports dvfs (dynamic voltage and frequency scaling). Four different frequencies are supported. Type:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

and you'll get them all listed. In case you have an AM3354 or AM3359 with a maximum of 800MHz these are:

```
300000 600000 720000 800000
```

The voltages are scaled according to the setup of the frequencies.

You can decrease the maximum frequency (e.g. to 720000)

```
echo 720000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
```

or increase the minium frequency (e.g. to 600000)

```
echo 600000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
```

Asking for the current frequency will be done with:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

So called governors are selecting one of this frequencies in accordance to their goals, automatically. Available governors are:

performance Always selects the highest possible CPU core frequency.

powersave Always selects the lowest possible CPU core frequency.

ondemand Switches between possible CPU core frequencies in reference to the current system load. When the system load increases above a specific limit it increases the CPU core frequency immediately. This is the default governor when the system starts up.

userspace Allows the user or userspace program running as root to set a specific frequency (e.g. to 600000):

```
echo 600000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

So when you type

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
```

you’ll get the result:

```
userspace powersave ondemand performance
```

In order to ask for the current governor, type

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

and you’ll normally get:

```
ondemand
```

Switching over to another governor (e.g. *userspace*) will be done with:

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

For more detailed information about the govenors refer to the linux kernel documentation in:

Documentation/cpu-freq/governors.txt.

3.3.11 Using Qt

Nokia's Qtopia is very commonly used for embedded systems and it's supported by this BSP. Please visit http://qt.nokia.com in order to get all the documentation that are available about this powerful cross-platform GUI toolkit.

Within the BSP come some demos that show what is possible with Qt version 4.8.4. They're located in */usr/bin/qt4-demos*. In order to try them you need to connect an USB mouse to the board. Please go into this directory with

```
export QWS_MOUSE_PROTO=tslib:/dev/input/event0
cd /usr/bin/qt4-demos
```

and then start demos with commands like

```
spreadsheet/spreadsheet -qws
```

Each of the Qt applications shows a standardized little green Qt-icon at its upper left side that offers standard window functions like minimize, maximize and close.

Demo *fluidlauncher* will be started automatically after booting.

3.3.12 System Reset Button (S2)

The phyBOARD-Wega is equipped with a system reset button at S2. Pressing this button will toggle the X_nRESET_IN pin (X64A11) of the phyCORE SOM low, causing the module to reset. Additionally, a reset is generated on nRESET_OUT to reset peripherals.

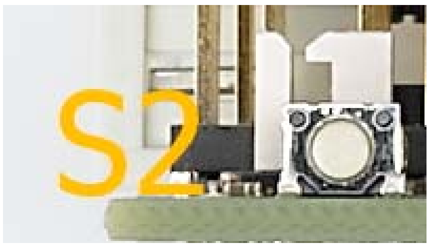


Figure 12: System Reset Button S2

3.3.12.1 Software Implementation

See chapter 3.3.9.1.

4 Advanced Information

4.1 About this Section

This Section addresses advanced developers who want to configure, build and install a new kernel and file system, design custom expansion boards, or display adapters. It includes the following information:

- Step by step instructions on how to configure, build and install a new kernel and file system using the Live DVD
- An explanation on how to set up PTXdist and the BSP on your Linux Host PC
- A description how to update the Bootloader and how to write the Kernel and Root File System into the Flash from within your own Linux system
- Detailed information on the different interfaces and features of the phyBOARD-Wega at a system level

4.2 Software Overview

In the chapter 2 you have learned how to work with Eclipse. The following section shows you how to work with PTXdist.

PTXdist is a GPL licensed tool for userlands, started by Pengutronix a close partner of PHYTEC. It is mainly a development environment to build your own embedded linux system. PTXdist creates a complete runable operating system with the root filesystem and all it's applications. After building the BSP on your host PC you can download it to and execute it on the target.

4.3 Getting Started with the BSP

In this chapter you will go through some software topics. First you will configure and compile your own kernel and root file system. With the help of PTXdist you can add additional features, or disable them if they are not needed. After compiling the new images, you will learn how to write the newly created kernel and root file system into target's flash memory and how to start from.

4.3.1 Working with the Kernel

In this part you will learn how to configure and build a new Linux kernel and a root file system. Then you will configure the kernel and the root file system with the help of PTXdist, a tool to build the Board Support Package and to create your own image. After the configuration you will create your own image.

To configure your images the following files are already pre-installed:

- ptxdist-XXXX.XX.X.tgz (Tool from our partner Pengutronix, that configures and compiles BSPs)
- arm-cortexa8-linux-gnueabihf (Toolchain)
- phyBOARD-WEGA-AM335x-PDXX.X.X.tar.gz (BSP for the phyBOARD)

4.3.2 Working with the filesystem

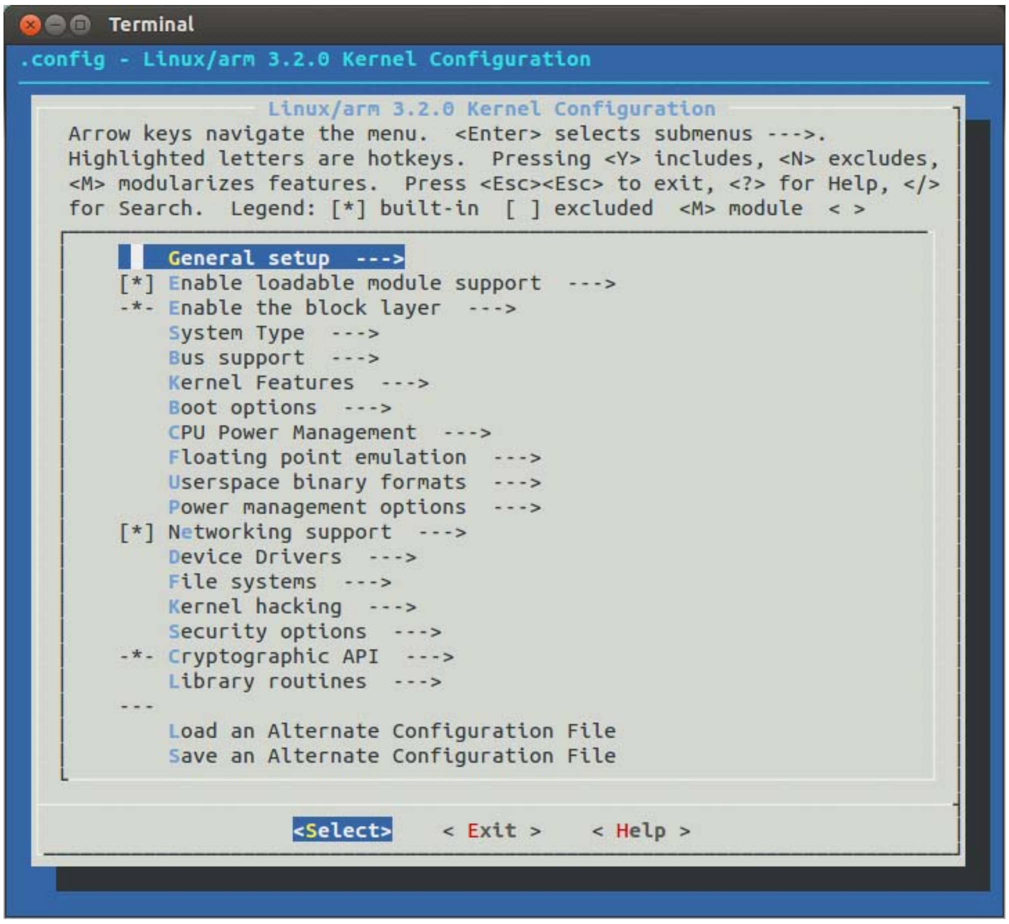
Let's start by opening a new terminal if you haven't already opened it and change into the directory of the pre-installed BSP:

Terminal

Click the terminal icon on your desktop

- Type the following command to change to the BSP-directory: **cd /opt/PHYTEC_BSPs/phyBOARD-WEGA-AM335x-PD13.0.0**
- Type the following to show the deselected/selected kernel features: **ptxdist kernelconfig**

You should see the following output on the terminal:

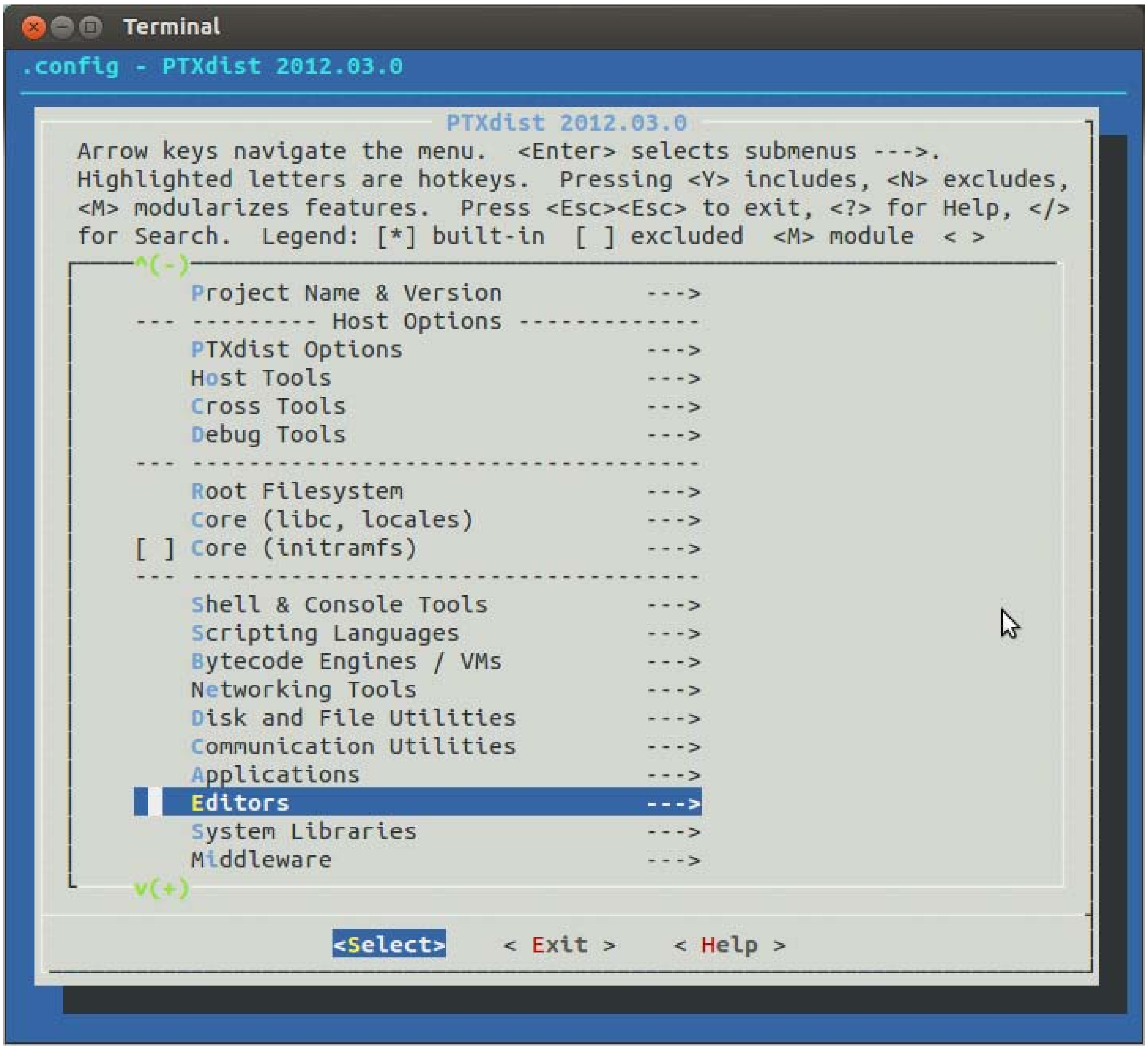


```
Terminal
.config - Linux/arm 3.2.0 Kernel Configuration
                    Linux/arm 3.2.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

          General setup  --->
      [*] Enable loadable module support  --->
      -*- Enable the block layer  --->
          System Type  --->
          Bus support  --->
          Kernel Features  --->
          Boot options  --->
          CPU Power Management  --->
          Floating point emulation  --->
          Userspace binary formats  --->
          Power management options  --->
      [*] Networking support  --->
          Device Drivers  --->
          File systems  --->
          Kernel hacking  --->
          Security options  --->
      -*- Cryptographic API  --->
          Library routines  --->
      ---
          Load an Alternate Configuration File
          Save an Alternate Configuration File

               <Select>    < Exit >    < Help >
```

- Leave the menu by selecting < *Exit* > at the bottom with the arrow right key and press **Enter**.

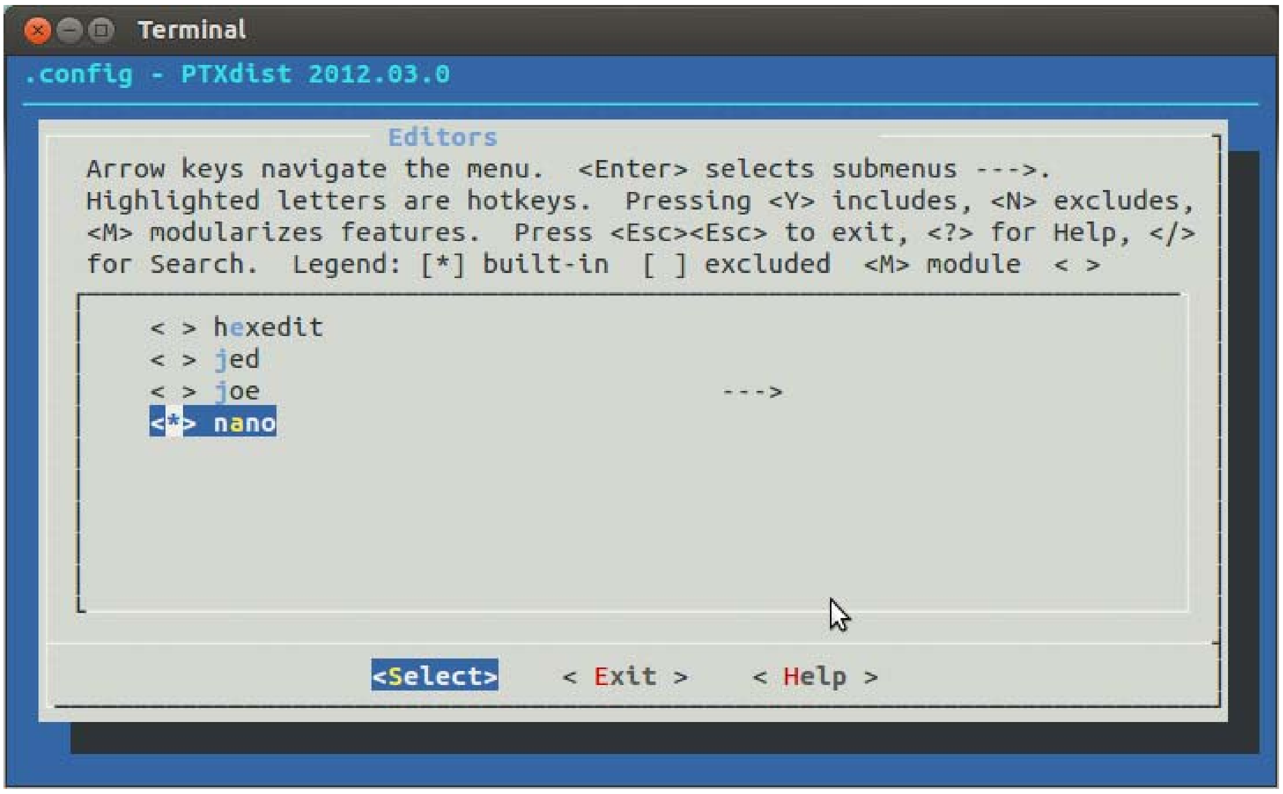
- Now let's take a look into the user space and add a new program in the root file system. Type the following:
  ptxdist menuconfig

You should see the following output:



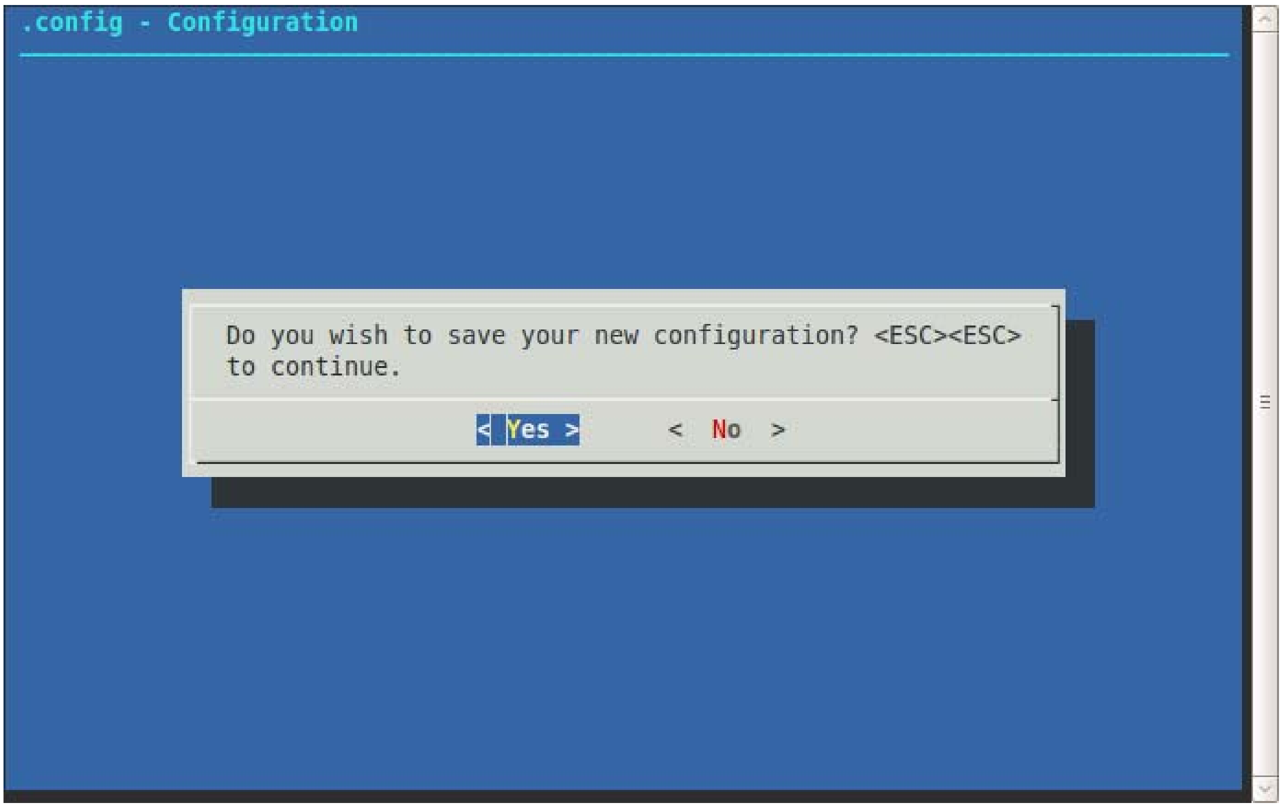
```
.config - PTXdist 2012.03.0
                              PTXdist 2012.03.0
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
 ^(-)
        Project Name & Version            --->
    --- --------- Host Options ------------
        PTXdist Options                   --->
        Host Tools                        --->
        Cross Tools                       --->
        Debug Tools                       --->
    --- -----------------------------------
        Root Filesystem                   --->
        Core (libc, locales)              --->
    [ ] Core (initramfs)                  --->
    --- -----------------------------------
        Shell & Console Tools             --->
        Scripting Languages               --->
        Bytecode Engines / VMs            --->
        Networking Tools                  --->
        Disk and File Utilities           --->
        Communication Utilities           --->
        Applications                      --->
        Editors                           --->
        System Libraries                  --->
        Middleware                        --->
 v(+)

              <Select>    < Exit >    < Help >
```

- For example we add *nano* to the user space which is another editor. With the help of the up and down arrow key we select *Editors* and press **Enter**.
- After that, we navigate to *nano* in the list of editors. By pressing **Y** we select *nano* as "*built-in*".



```
Terminal
.config - PTXdist 2012.03.0
                          Editors
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

      < > hexedit
      < > jed
      < > joe                              --->
      <*> nano

              <Select>    < Exit >    < Help >
```

- Leave the menu by selecting < *Exit* > at the bottom with the arrow right key and press **Enter**. Repeat this until a question appears to save the new configuration.
- Select < *Yes* > and press **Enter** to save the configuration.



```
.config - Configuration

    Do you wish to save your new configuration? <ESC><ESC>
    to continue.

                 < Yes >      <  No  >
```

- Now we can start building the configured BSP. In the terminal type: **ptxdist go**
- After it is finished you can finally create the root file system by calling: **ptxdist images**

You will find the kernel named *linuximage* and the root file system named *root.ubi* in the directory *platform- phyBOARD-WEGA-AM335x /images/*.

4.3.3 Writing the Images into the Target's Flash

In this section you will find a description on how to write the newly created images into the phyBOARD-Wega-AM335x's flash memory. Before the images can be written into the flash, the target will have to download them from a TFTP server. This will be done from the command line of the boot loader. The images will be copied into target's RAM. Then you will have to erase the part of the flash to which you want to copy the images. Finally the images are written from the RAM to the flash.

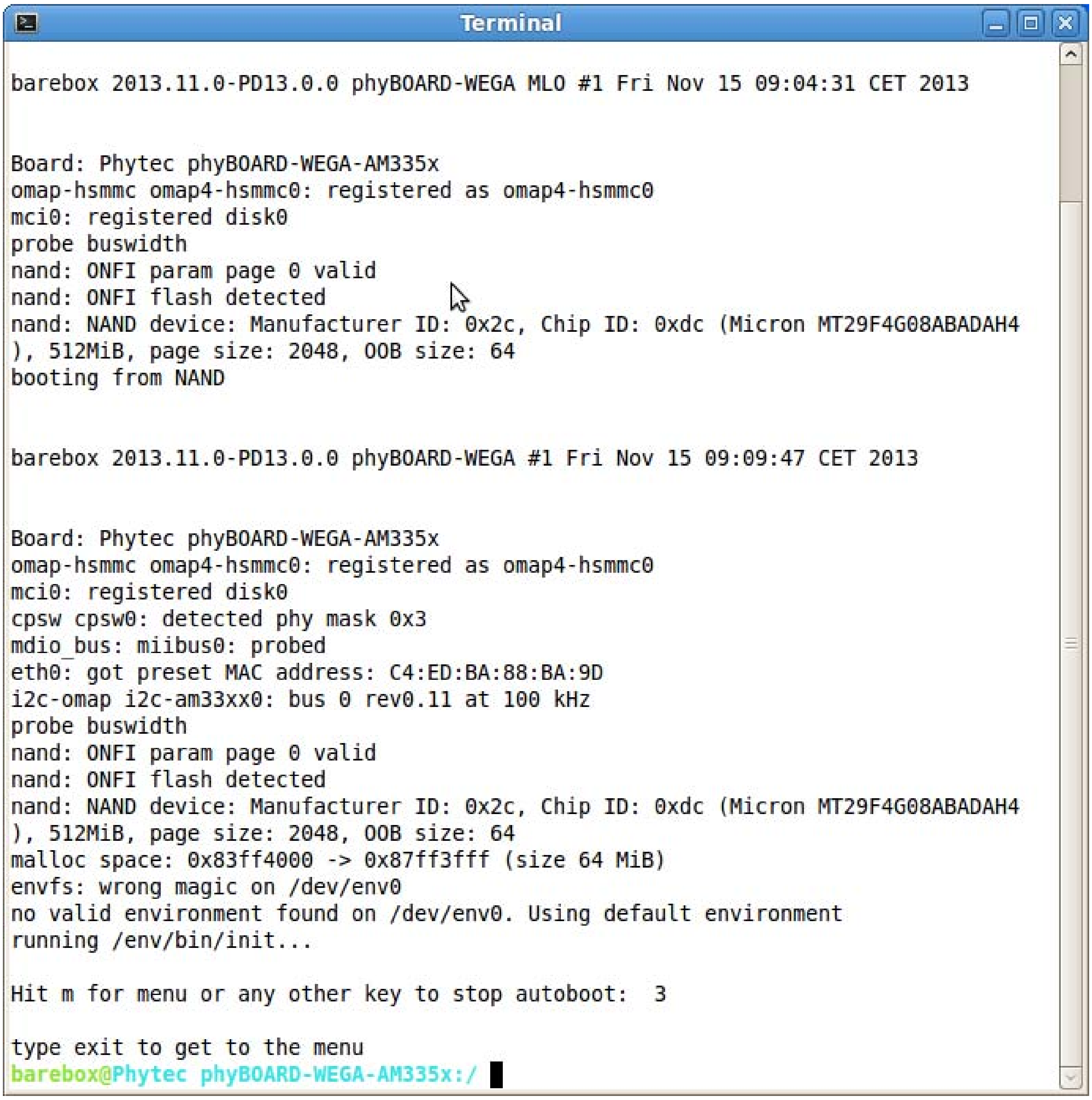
In the default configuration you will find four partitions on the target: The first partition contains the boot loader, the second is used to store the boot loader settings, the third partition stores the Linux kernel, and the fourth contains the root file system.

⚠	You should never erase the Barebox partition. If this partition is erased, you won't be able to start your target anymore. In such a case refer to chapter 3 "Updating the software".

⚠	Versions of Barebox and Linux kernel must match. Therefore the following steps should only be done using the hardware that was shipped together with the DVD and thus already contains the same version of the BSP.

Terminal

- First open a new terminal window if it is not opened yet. Then change to the directory */opt/PHYTEC_BSPs/phyBOARD-WEGA-AM335x-PD**
- Copy the new images to the */tftpboot* directory and exit:
  cd platform-phyBOARD-WEGA-AM335x/images
  sudo cp linuximage /tftpboot
  sudo cp root.ubifs /tftpboot
  exit
- 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.



```
barebox 2013.11.0-PD13.0.0 phyBOARD-WEGA MLO #1 Fri Nov 15 09:04:31 CET 2013


Board: Phytec phyBOARD-WEGA-AM335x
omap-hsmmc omap4-hsmmc0: registered as omap4-hsmmc0
mci0: registered disk0
probe buswidth
nand: ONFI param page 0 valid
nand: ONFI flash detected
nand: NAND device: Manufacturer ID: 0x2c, Chip ID: 0xdc (Micron MT29F4G08ABADAH4
), 512MiB, page size: 2048, OOB size: 64
booting from NAND


barebox 2013.11.0-PD13.0.0 phyBOARD-WEGA #1 Fri Nov 15 09:09:47 CET 2013


Board: Phytec phyBOARD-WEGA-AM335x
omap-hsmmc omap4-hsmmc0: registered as omap4-hsmmc0
mci0: registered disk0
cpsw cpsw0: detected phy mask 0x3
mdio_bus: miibus0: probed
eth0: got preset MAC address: C4:ED:BA:88:BA:9D
i2c-omap i2c-am33xx0: bus 0 rev0.11 at 100 kHz
probe buswidth
nand: ONFI param page 0 valid
nand: ONFI flash detected
nand: NAND device: Manufacturer ID: 0x2c, Chip ID: 0xdc (Micron MT29F4G08ABADAH4
), 512MiB, page size: 2048, OOB size: 64
malloc space: 0x83ff4000 -> 0x87ff3fff (size 64 MiB)
envfs: wrong magic on /dev/env0
no valid environment found on /dev/env0. Using default environment
running /env/bin/init...

Hit m for menu or any other key to stop autoboot:  3

type exit to get to the menu
barebox@Phytec phyBOARD-WEGA-AM335x:/
```

- Check the network settings with following command:
  ifup eth0
  devinfo eth0
  The target should present the this 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 download the images from the TFTP server to the target's RAM, then we erase the required flash and write the images from the RAM into the flash.

- Type **erase /dev/nand0.kernel.bb** to erase the kernel partition.
- Type **cp /mnt/tftp/linuximage /dev/nand0.kernel.bb** to download the kernel using TFTP and write it into the target's flash. The copy process can take up to several minutes.
- Now we flash the root filesystem type:
  ubiformat -y /dev/nand0.root
  ubiattach /dev/nand0.root
  ubimkvol /dev/ubi0 root 0
  cp /mnt/tftp/root.ubifs /dev/ubi0.root
- Type **boot** to boot the phyBOARD-Wega-AM335x with the new kernel and root file system.
- After the target has successfully finished booting, type **root** to log in.
- Now we can test our new editor *nano* by trying to open a file with it.
- **nano /etc/profile.environment**
- Close *nano* by pressing **CTRL+X**.
- Close *Microcom* when *nano* is closed.

⚠	Troubleshooting: If any problem occurs after writing the kernel or the root file system into the flash memory, you can restore the original kernel (*linuximage*) and root file system (*root.ubifs*) from the */tftpboot* directory.

	All files are also downloadable from our ftp server or you can find them on our Linux-phyBOARD-Wega-AM335x-Kit-DVD under */PHYTEC/BSP/*. If you want other versions check our ftp server too: ftp://ftp.phytec.de/pub/Products/

👍	In this section you learned how to download images from a tftp server into the RAM of the target. The images have been written from RAM to flash and finally the target was started with the new images.

4.4 Updating the software

If you found a newer BSP on our ftp server ftp://ftp.phytec.de/pub/Products and want to flash it, this chapter shows how to do. Also in case that your phyBOARD-Wega-AM335x doesn’t start anymore because you damaged its software during the previous chapter, you’re right here, too. In the latter case you’ll find all needed original images on the DVD under */PHYTEC/BSP*.

4.4.1 Creating a bootable SD card

In case that your phyBOARD-Wega-AM335x doesn’t start anymore due to a damaged bootloader, you need to boot from an SD card. This SD card must be formatted in a special way, because the AM335x doesn’t know anything about file systems. Instead it’s hard coded at which sectors of the SD card the AM335x expects the bootloader to be.

Use the following script on order to get an SD card properly formatted:

```
#!/bin/bash
if [ ! "$1" = "/dev/sda" ] ; then
    unset LANG
    DRIVE=$1
        umount $DRIVE"1"
        umount $DRIVE"2"
    if [ -b "$DRIVE" ] ; then
        dd if=/dev/zero of=$DRIVE bs=1024 count=1024
        SIZE=`fdisk -l $DRIVE | grep Disk | awk '{print $5}'`
        echo DISK SIZE - $SIZE bytes
        CYLINDERS=`echo $SIZE/255/63/512 | bc`
        echo CYLINDERS - $CYLINDERS
        {
        echo ,9,0x0C,*
        echo ,,,-
        } | sfdisk -D -H 255 -S 63 -C $CYLINDERS $DRIVE
        mkfs.vfat -F 32 -n "boot" ${DRIVE}1
        mke2fs -j -L "rootfs" ${DRIVE}2
    fi
fi
```

It’s important that after that you’ll copy image *MLO* to partition named *boot* at first, so that it will be written to the right sectors of the SD card. You may freely choose the order in which you copy all further images, but *MLO* must always be the very first!

Copy *barebox-image*, *linuximage* and *root.ubi* to partition *boot*, too.

In case that you want to boot the whole Linux from SD card, furthermore untar *root.tgz* to partition *rootfs*:

```
sudo tar zxf root.tgz -C /media/rootfs
```

4.4.2 Flashing the Bootloader

Use bootable SD card in order to boot into Barebox.

Without any valid Bootloader in NAND, board will automatically try SD card for booting. Otherwise you need to force booting from SD card by connecting X_LCD_D2 (pin 8 at X70) to a high-level (e.g. VCC3V3 – pin 13 at X71) during the power-up sequence.

Partition *boot* of the SD card should automatically be mounted by Barebox as directory */mnt/disk*. Otherwise you can mount manually using the following Barebox commands:

```
mkdir /mnt/disk
mci0.probe=1
mount /dev/disk0.0 /mnt/disk
```

Flash x-loader MLO by typing:

```
erase /dev/nand0.xload.bb
cp /mnt/disk/MLO /dev/nand0.xload.bb
```

Write Barebox into flash by typing:

```
erase /dev/nand0.barebox.bb
cp /mnt/disk/barebox.bin /dev/nand0.barebox.bb
```

4.4.3 Writing the Kernel / Root File System into Flash

Flash Linux kernel by typing:

```
erase /dev/nand0.kernel.bb
cp /mnt/disk/linuximage /dev/nand0.kernel.bb
```

Write root file system into flash by typing:

Note that normally you should not flash Linux’s root file system into NAND the way described below. Ubifs keeps erase counters within the NAND in order to be able to balance write cycles equally over all NAND sectors. So if there’s already an ubifs on your module and you want to replace it by a new one, using erase and cp will also erase these erase counters, and this “brute way” should be avoided and only be used for troubleshooting, i.e. if the procedure described in *section 4.3.3* causes errors.

```
erase /dev/nand0.root.bb
cp /mnt/disk/root.ubi /dev/nand0.root.bb
```

4.5 Setup your own Linux-Host-PC

This chapter is for developers who want to use there own existing environment and not our modified Ubuntu version. It is not needed if you have already installed our modified Ubuntu version like explained in the Application Guide.
We give an overview in this chapter of the modifications which we made to the Ubuntu version on the phyBOARD-WEGA-AM335x Tool DVD in comparison to the original Ubuntu. 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 don't 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't 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.

4.5.1 Essential settings

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

4.5.1.1 Installation of software packages

At first various software packages 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 libncurses-dev gawk flex bison texinfo gettext
2. Packages for developoment
   sudo apt-get -y install vim eclipse
3. Packeges to set up the TFTP server:
   sudo apt-get -y install tftpd-hpa

During installation it can happen that some programs ask for a license agreement. Just go threw the steps.

The first preparations are completed. In the next sections you will proceed with the installation of Eclipse and the set up of the TFTP server.

4.5.1.2 Setting up Eclipse and integrate plug ins

The instructions in this chapter show how to setup *Eclipse* and integrate the plug ins for *QT* and *C/C++*. Thereby you can assign own programs, written in *Eclipse,* to the target.

- First you must create a *workspace* folder, in which you can save your *Eclipse* projects. In the example this is done in the */home/* folder:
  mkdir /home/$USER/workspace
- Thereafter you can open *Eclipse* and insert the path to the created workspace in the pop-up window:
  eclipse
- Eclipse is started and now we click at *Help* in the menu bar *Install new Software*.
- Thereupon a window was opened and in the text field "*Work with*" we enter the following address: **http://download.eclipse.org/tools/cdt/releases/indigo** and click on **"Add"**.
- After a while the software which we can integrate appears in the area with a scrollbar. We check **"CDT Main Features"** and click at **"Next"**.
- Now the system shows us an overview of the installation details, which we skip with a click on **Next**.
- At last the system shows us the licensing agreement, which we accept and after we have clicked on **Finish** it begins to install the required software.

The *CDT* plug in is installed. We close *Eclipse* and go on with the *QT* integration.

- For the *QT* integration you must use an external package from the *Nokia* website. This package is already available on our DVD under */PHYTEC/Applications*. Unpack the compressed archive in the */usr/lib* folder:
  cd /usr/lib
  tar xzf ...PHYTEC/Applications/qt-eclipse-integration-linux.x86-1.6.1.tar.gz
- After that start *Eclipse* with the option *clean*, which cleans any cached data:
  eclipse --clean
- *Eclipse* is started and now you must include two path variables:
  In the menu bar click on **"Window"** ► **"Preferences"** ► "**QT**" and enter the following QT specific information **Name:** *QT4* ► **Bin Path:** */opt/PHYTEC_BSPs/phyCORE-AM335x-PD13.1.0/platform-phyCORE-AM335x/sysroot-cross/bin* ► **Include Path:** */opt/PHYTEC_BSPs/phyCORE-AM335x-PD13.1.0/platform-phyCORE-AM335x/sysroot-host/include*)
- Finally close *Eclipse* and start it again with the *clean* option.

Congratulations! You have successfully integrated two new plugins in Eclipse and now you can start programming in C/C++ and QT in an Eclipse environment. In the next sub-section you will find a short introduction on how to setup a TFTP server.

4.5.1.3 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 must 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
- At last we must set the right permissions:
  chmod 777 /tftpboot

You have successfully set up the TFTP server. In the future the phyBOARD-Wega-AM335x can access to the /tftpboot/ folder to load the images.

4.5.2 Optional settings

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

- Installation of *vim*, the improved *vi* editor.
- Creation of desktop-icons for faster and easier start of required programs.
- The following modifications were made to the look of Ubuntu:
  – Other wallpaper and associated options were adjusted with the help of **gsettings**
  – Changing the color of the *Gnome* terminal.
- **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.

Congratulations! You have successfully configured your Ubuntu to work with.

4.6 System Level Hardware Information

4.6.1 Audio/Video connectors

The Audio/Video (A/V) connectors X70 and 71 provide an easy way to add typical A/V functions and features to the phyBOARD-Wega. Standard interfaces such as parallel display, I²S and I^{2}C as well as different supply voltages are available at the two A/V female dual entry connectors. Special feature of these connectors are their connectivity from the bottom or the top. The pinout of the A/V connectors are shown in *Table 8* and *Table 9.*

Pin #	Signal Name	Description
1	GND	Ground
2	X_LCD_D21	LCD D21
3	X_LCD_D18	LCD D18
4	X_LCD_D16	LCD D16
5	X_LCD_D0	LCD D0
6	GND	Ground
7	X_LCD_D1	LCD D1
8	X_LCD_D2	LCD D2
9	X_LCD_D3	LCD D3
10	X_LCD_D4	LCD D4
11	GND	Ground
12	X_LCD_D22	LCD D22
13	X_LCD_D19	LCD D19
14	X_LCD_D5	LCD D5
15	X_LCD_D6	LCD D6
16	GND	Ground
17	X_LCD_D7	LCD D7
18	X_LCD_D8	LCD D8
19	X_LCD_D9	LCD D9
20	X_LCD_D10	LCD D10
21	GND	Ground
22	X_LCD_D23	LCD D23
23	X_LCD_D20	LCD D20
24	X_LCD_D17	LCD D17
25	X_LCD_D11	LCD D11
26	GND	Ground
27	X_LCD_D12	LCD D12

28	X_LCD_D13	LCD D13
29	X_LCD_D14	LCD D14
30	X_LCD_D15	LCD D15
31	GND	Ground
32	X_LCD_PCLK	LCD Pixel Clock
33	X_LCD_BIAS_EN	LCD BIAS
34	X_LCD_HSYNC	LCD Horizontal Synchronisation
35	X_LCD_VSYNC	LCD Vertical Synchronisation
36	GND	Ground
37	GND	Ground
38	X_PWM1_OUT	Pulse Wide Modulation
39	VCC_BL	Backlight power supply
40	VCC5V	5 V power supply

Table 8: PHYTEC A/V connector X70

Pin #	Signal Name	Description
1	X_I2S_CLK	I²S Clock
2	X_I2S_FRM	I²S Frame
3	X_I2S_ADC	I²S Analog-Digital converter (microfone)
4	X_I2S_DAC	I²S Digital-Analog converter (speaker)
5	X_AV_INT_GPIO1_30	A/V interrupt; GPIO1_30
6	X_MCASP0_AHCLKX	McASP0 high frequency clock
7	GND	Ground
8	nRESET_OUT	Reset
9	TS_X+	Touch X+
10	TS_X-	Touch X-
11	TS_Y+	Touch Y+
12	TS_Y-	Touch Y-
13	VCC3V3	3.3 V power supply
14	GND	Ground
15	X_I2C0_SCL	I²C Clock
16	X_I2C0_SDA	I²C Data

Table 9: PHYTEC A/V connector X71

4.6.2 Software Implementation

4.6.3 Audio I^{2}S

Audio support on the module is done via the I2S interface and controlled via I2C.

On the phyBOARD-Wega the audio codec's registers can be accessed via the I2C0 interface at address 0x18 (7-bit MSB addressing).

As of the printing of this manual the BSP delivered with the phyCARD-Wega-Am335x does not support the audio interfaces. Please visit the PHYTEC website for a road map of the BSP.

4.6.4 I^{2}C Connectivity

The I^{2}C interface of the AM335x is available at different connectors on the phyBOARD-Wega. The following table provides a list of the connectors and pins with I^{2}C connectivity.

Connector	Location
Expansion connector X69	pin 11 (I^{2}C_SDA); pin 13 (I^{2}C_SCL)
A/V connector X71	pin 16 (I^{2}C_SDA); pin 15 (I^{2}C_SCL)

Table 10: I^{2}C Connectivity

To avoid any conflicts when connecting external I^{2}C devices to the phyBOARD-Wega the addresses of the on-board I^{2}C devices must be considered. *Table 11* lists the addresses already in use. The table shows only the default address.

Board	Prod. No.	Device	Address used (7 MSB)
phyCORE-AM335x	PCL-051	EEPROM	0x52
		RTC	0x68
		PMIC	0x2D, 0x12
phyBOARD-Wega	PBA-CD-02	Audio	0x18
AV-Adapter HDMI	PEB-AV-01	HDMI Core	0x70
		CEC Core	0x34
AV-Adapter Display	PEB-AV-02	GPIO Expander	0x41
Evaluation Board	PEB-EVAL-01	EEPROM	0x56
M2M Board	PEB-C-01	GPIO Expander	0x20
		GPIO Expander	0x21
		GPIO Expander	0x22

Table 11: I^2C Addresses in Use

4.6.5 Expansion connector

The expansion connector X69 provides an easy way to add other functions and features to the phyBOARD-Wega. Standard interfaces such as UART, SPI and I^2C as well as different supply voltages and some GPIOs are available at the expansion female connector. The pinout of the expansion connector is shown in Secure Digital Memory Card/ MultiMedia Card (X11)
.

Pin #	Signal Name	Description
1	VCC3V3	3.3 V power supply
2	VCC5V	5 V power supply
3	VDIG1_1P8V	1.8 V power supply
4	GND	Ground
5	X_SPI0_CS0	SPI 0 chip select 0
6	X_SPI0_MOSI	SPI 0 master output/slave input
7	X_SPI0_MISO	SPI 0 master input/slave output
8	X_SPI0_CLK	SPI 0 clock output
9	GND	Ground
10	X_UART0_RXD	UART 0 receive data (standard debug interface)
11	X_I2C0_SDA	I²C 0 Data
12	X_UART0_TXD	UART 0 transmit data (standard debug interface)

13	X_I2C0_SCL	I²C 0 Clock
14	GND	Ground
15	X_JTAG_TMS	JTAG Chain Test Mode Select signal
16	X_nJTAG_TRST	JTAG Chain Test Reset
17	X_JTAG_TDI	JTAG Chain Test Data Input
18	X_JTAG_TDO	JTAG Chain Test Data Output
19	GND	Ground
20	X_JTAG_TCK	JTAG Chain Test Clock signal
21	X_USB1_DP_EXP	USB1 data plus
22	X_USB1_DM_EXP	USB1 data minus
23	nRESET_OUT	Reset
24	GND	Ground
25	X_MMC2_CMD	MMC command
26	X_MMC2_DAT0	MMC data 0
27	X_MMC2_CLK	MMC clock
28	X_MMC2_DAT1	MMC data 1
29	GND	Ground
30	X_MMC2_DAT2	MMC data 2
31	X_UART2_RX_GPIO3_9	UART 2 receive data; GPIO3_19
32	X_MMC2_DAT3	MMC data 3
33	X_UART2_TX_GPIO3_10	UART 2 transmit data; GPIO3_10
34	GND	Ground
35	X_UART3_RX_GPIO2_18	UART 3 receive data; GPIO2_18
36	X_UART3_TX_GPIO2_19	UART 3 transmit data; GPIO2_19
37	X_INTR1_GPIO0_20	Interrupt 1; GPIO0_20
38	X_GPIO0_7	GPIO0_7
39	X_AM335_EXT_WAKEUP	External wakeup
40	X_INT_RTCn	Interrupt from the RTC
41	GND	Ground
42	X_GPIO3_7	GPIO3_7; Caution: Also connected to power fail signal through R415!
43	nRESET_IN	Push-button reset
44	X_GPIO3_8	GPIO 3_8; Caution: Also connected to power down circuit through R412!
45	X_AM335_NMIn	AM335x non-maskable interrupt
46	GND	Ground
47	X_AIN4	Analog input 4

48	X_AIN5	Analog input 5
49	X_AIN6	Analog input 6
50	X_AIN7	Analog input 7
51	GND	Ground
52	X_USB1_DRVVBUS	USB 1 bus control output
53	X_USB1_ID	USB 1 port identification
54	USB1_VBUS	USB 1 bus voltage
55	X_USB1_CE	USB 1 charger enable
56	GND	Ground
57	X_PMIC_POWER_EN	Enable Power Management IC for AM335x
58	X_PB_POWER	Power On for Power Management IC for AM335x
59	GND	Ground
60	VCC5V_IN	5 V input supply voltage

Table 12: PHYTEC Expansion Connector X69

4.6.5.1 Software Implementation

4.6.6 SPI Connectivity

The driver for SPI can be accessed by its API within the kernel. If you connect a chip to SPI, you should implement its driver into the kernel, as well. The SPI driver offers no /dev/spi entry in userspace, because using it would mean to place the driver for your chip in userspace, too, what most probably would n**o**t be **use**ful. Please note that the BSP **already includes** a couple of deactivated drivers for **various** chips**. These drivers** might be helpful for you **even though they are usually not tested.**

4.6.7 User programmable GPIOs

There are different User programmable GPIOs available. The signals are available on the expansion connector or the corresponding expansion-boards like PEB-EVAL-01. For more information look at Application Guide for the Expansion Boards or section “Expansion connector” in the System Guide.

4.6.7.1 Software Implementation

For setting and resetting the green user LED on the expansion board just enter:

```
cd /sys/devices/platform/leds-gpio/leds/peb_eval_01:green:led3
echo 1 > brightness
echo 0 > brightness
```

For getting values from the three user buttons on the expansion board just enter

```
cd /sys/class/gpio
echo 20 > export
echo 7 > export
echo 106 > export
```

and get the values with

```
cat gpio20/value
cat gpio7/value
cat gpio106/value
```

5 Revision History

Date	Version #	Changes in this manual
25.11.2013	Manual L-792e_0	First edition. Describes the phyBOARD-Wega-AM335x SOM (PCB 1397.0) with phyBOARD-Wega- Carrier Board (PCB 1405.0).

Index

Document: phyBOARD-Wega-AM335x
Document number: L-792e_0, November 2013

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

Published by

© PHYTEC Messtechnik GmbH 2013

Ordering No. L-792e_0

Printed in Germany