

phyBOARD[®] -Mira-i.MX 6

Application Guide

Document No.: **L-806e_0**

SBC Prod. No...: **PB-01501-xxx**

CB PCB No.: **1434.0**

SOM PCB No.: **1429.0**

Edition: February 2015

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

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

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

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

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

	EUROPE	NORTH AMERICA	FRANCE
Address:	PHYTEC Messtechnik GmbH Robert-Koch-Str. 39 D-55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA	PHYTEC France 17, place Saint-Etienne F-72140 Sillé-le-Guillaume FRANCE
Sales:	+49 (6131) 9221-32 sales@phytec.de	+1 (800) 278-9913 sales@phytec.com	+33 (0)2 43 29 22 33 info@phytec.fr
Technical Support:	+49 (6131) 9221-31 support@phytec.de	+1 (206) 780-9047 support@phytec.com	support@phytec.fr
Fax:	+49 (6131) 9221-33	+1 (206) 780-9135	+33 (0)2 43 29 22 34
Web Site:	http://www.phytec.de http://www.phytec.eu	http://www.phytec.com	http://www.phytec.fr

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

List of Figures	iii
List of Tables	iv
Conventions, Abbreviations and Acronyms	v
Preface	viii
1 Introduction.....	1
1.1 Hardware Overview.....	1
1.1.1 Block Diagram.....	2
1.1.2 View of the phyBOARD-Mira-i.MX 6	3
1.2 Software Overview.....	5
1.2.1 Ubuntu	5
1.2.2 Eclipse.....	5
1.2.3 Qt Creator	6
1.2.4 The GNU Cross Development Toolchain	6
2 Application Programming.....	7
2.1 Installing our modified Ubuntu Live System	7
2.2 Working with Eclipse.....	8
2.2.1 Programming in the C/C++ Perspective.....	8
2.2.1.1 Work with the Demo Project	8
2.2.1.2 Creating a New Project	14
2.2.1.3 Modifying the Demo Application	21
2.2.1.4 Starting a Program out of Eclipse on the Target.....	24
2.2.2 Debugging an Example Project	26
2.2.2.1 Starting the GDB Server on the Target	27
2.2.2.2 Configuring and starting the Debugger in Eclipse	27
2.2.2.3 Setting a Breakpoint	32
2.2.2.4 Stepping through and Watching Variable Contents	33
2.2.2.5 Stepping through and Changing Variable Contents	35
2.2.2.6 Using the Memory Monitor.....	36
3 Accessing the phyBOARD-Mira Features.....	39
3.1 Overview of the phyBOARD-Mira Peripherals	39
3.1.1 Connectors and Pin Header	39
3.1.2 LEDs.....	40
3.1.3 Switches	40
3.2 Functional Components on the phyBOARD-Mira SBC.....	41
3.2.1 Power Supply	41
3.2.1.1 Power Connectors (X2).....	41
3.2.1.1.1 PHOENIX 2-pole MINI COMBICON Base Strip (X2)	42
3.2.1.1.2 WAGO 6-pole Male Header (X2)	42
3.2.1.2 Power LED D2	43
3.2.1.3 VBAT, RTC and JP3.....	43
3.2.2 UART Connectivity (X17 and X23)	44
3.2.2.1 Software Implementation	45
3.2.3 Ethernet Connectivity (X4)	46
3.2.3.1 Software Implementation	46
3.2.4 USB Connectivity (X5 and X6)	47
3.2.4.1 Software Implementation	47
3.2.4.1.1 USB Host.....	47

- 3.2.4.1.2 USB OTG 48
- 3.2.5 CAN Connectivity (X3, JP2) 49
 - 3.2.5.1 Software Implementation 50
- 3.2.6 Secure Digital Memory Card/MultiMedia Card (X22)..... 52
 - 3.2.6.1 Software Implementation 52
- 3.2.7 PCIe Connectivity (X7) 53
 - 3.2.7.1 Software Implementation 53
- 3.2.8 Camera Connectivity (X10)..... 54
 - 3.2.8.1 Software Implementation 54
- 3.2.9 HDMI Connectivity (X28) 54
 - 3.2.9.1 Software Implementation 55
- 3.2.10 LVDS Display Connectivity (X9) 56
 - 3.2.10.1 Software Implementation 56
- 3.2.11 Backlight and Control Connector (X8) 56
- 3.2.12 Touch Screen Connectivity (X13, X21) 56
 - 3.2.12.1 Software Implementation 57
- 3.2.13 Multicolor (RGB) LED (D6) 57
 - 3.2.13.1 Software Implementation 57
- 3.2.14 Boot Mode (S2) 58
- 3.2.15 System Reset Button (S1) 59
- 3.2.16 Audio/Video connectors (X13 and X14) 59
- 3.2.17 Expansion connector (X17) 59
- 3.2.18 Addressing the RTC..... 59
- 3.2.19 CPU Core Frequency Scaling 60
- 3.2.20 Using Qt..... 61
- 4 System Level Customizing 62**
 - 4.1 About this Section..... 62
 - 4.2 Software Overview..... 62
 - 4.3 Getting Started with the BSP 62
 - 4.3.1 Working with Yocto..... 62
 - 4.3.2 Writing the Images into the Target’s Flash 65
 - 4.4 Updating the software 69
 - 4.4.1 Creating a bootable SD card 69
 - 4.4.2 Flashing the Bootloader 70
 - 4.4.3 Writing the Kernel / Root File System into Flash 70
 - 4.5 System Level Hardware Information..... 71
 - 4.5.1 USB Connectivity (X5, X6, X7 and X17) 71
 - 4.5.1.1 Rerouting the USB Interfaces to other Connectors (J5, J6, J9 and J10) 71
 - 4.5.2 HDMI Connector (X28) 72
 - 4.5.3 LVDS connector (X9)..... 73
 - 4.5.4 CAN Connectivity 74
 - 4.5.5 Mini PCI express connector (X7) 75

4.5.6	Audio/Video connectors (X13 and X14)	77
4.5.6.1	Software Implementation	80
4.5.6.1.1	Framebuffer	80
4.5.6.1.2	Touch	80
4.5.6.1.3	Audio I ² S.....	80
4.5.6.1.4	I ² C Connectivity	81
4.5.7	Expansion connector (X17)	82
4.5.7.1	Software Implementation	84
4.5.7.1.1	UART Connectivity	84
4.5.7.1.2	SPI Connectivity	84
4.5.7.1.3	I ² C Connectivity	84
4.5.7.1.4	User programmable GPIOs	85
5	Revision History.....	86
	Index.....	87

List of Figures

Figure 1:	Block Diagram of the phyBOARD-Mira-i.MX 6	2
Figure 2:	View of the phyBOARD-Mira-i.MX 6 (top).....	3
Figure 3:	View of the phyBOARD-Mira-i.MX 6 (bottom)	4
Figure 4:	Power Supply Connectors(X2)	41
Figure 5:	RS-232 or RS-485 Interface Connector X23.....	44
Figure 6:	RS-232 / RS-485 Connector Signal Mapping	45
Figure 7:	Ethernet Interface at Connectors X4.....	46
Figure 8:	Components supporting the USB Interfaces.....	47
Figure 9:	Components supporting the CAN Interface	49
Figure 10:	CAN Connector Signal Mapping	50
Figure 11:	SD/MM Card Interface at Connector X22(back side)	52
Figure 12:	PCIe Interface at Connector X7	53
Figure 13:	Camera Interface(phyCAM-S+)at Connector X10	54
Figure 14:	HDMI Interface Connector X28	55
Figure 15:	Touch Screen Connectors X21	57
Figure 16:	Boot Switch (S2).....	58

List of Tables

Table 1:	Abbreviations and Acronyms used in this Manual.....	vi
Table 2:	phyBOARD-Mira Connectors and Pin Headers	39
Table 3:	phyBOARD-Mira LEDs Descriptions	40
Table 4:	phyBOARD-Mira Switches Description.....	40
Table 5:	Pin Assignment of the 2-pole PHOENIX MINI COMBICON Base Strip at X2.....	42
Table 6:	Pin Assignment of the 6-pole WAGO Connector at X2	42
Table 7:	Pinout of the 2-pole pin header JP3	43
Table 8:	Pin Assignment of RS-232 /RS-485 Interface Connector X23	44
Table 9:	Pin Assignment of CAN Connector X3	49
Table 10:	PHYTEC Camera Connector X10	54
Table 11:	Display Power Connector X8 Signal Description	56
Table 12:	Touch Screen Connectivity	57
Table 13:	Multicolor LED Configuration	57
Table 14:	Boot Switch Configuration (S2).....	58
Table 15:	USB1 (USB OTG) Routing Configuration	71
Table 16:	USB2 (USB Host) Routing Configuration	71
Table 17:	Pin Assignment HDMI Connector X28.....	72
Table 18:	Pin Assignment LVDS Display Connector X9.....	73
Table 19:	Configuration for LVDS Display Connector X9	74
Table 20:	Configurations for CAN interface	74
Table 21:	Mini PCI express Connector X7	76
Table 22:	PHYTEC A/V connector X14	78
Table 23:	PHYTEC A/V connector X13	79
Table 24:	A/V Jumper Configuration J1.....	79
Table 25:	I ² C1 Connectivity	81
Table 26:	I ² C Addresses in Use	81
Table 27:	PHYTEC Expansion Connector X17	84

Conventions, Abbreviations and Acronyms

This hardware manual describes the PB-01501-XXX Single Board Computer (SBC) in the following referred to as phyBOARD-Mira-i.MX6. The manual specifies the phyBOARD-Mira-i.MX6's design and function. Precise specifications for the Freescale Semiconductor i.MX6 microcontrollers can be found in the Freescale Semiconductor's i.MX6 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 I²C devices always represent the 7 MSB of the address byte. The correct value of the LSB which depends on the desired command (read (1), or write (0)) must be added to get the complete address byte. E.g. given address in this manual 0x41 => complete address byte = 0x83 to read from the device and 0x82 to write to the device.
- Tables which describe jumper settings show the default position in **bold, blue text**.
- Text in *blue italic* indicates a hyperlink within, or external to the document. Click these links to quickly jump to the applicable URL, part, chapter, table, or figure.
- Text in **bold italic** indicates an interaction by the user, which is defined on the screen.
- Text in `Conso1as` indicates an input by the user, without a premade text or button to click on.
- Text in *italic* indicates proper names of development tools and corresponding controls (windows, tabs, commands etc.) used within the development tool, no interaction takes place.
- **White Text on black background** shows the result of any user interaction (command, program execution, etc.)

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-Mira 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 Freescale Semiconductor i.MX 6 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
NM	Not Mounted
NS	Not Specified
PCB	Printed circuit board.
PDI	PHYTEC Display Interface; defined to connect PHYTEC display adapter boards, or custom adapters
PEB	PHYTEC Expansion 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-C(D)-06 /phyBOARD-Mira-i.MX 6
SMT	Surface mount technology.
SOM	System on Module; used in reference to the PCM-058 /phyCORE-i.MX 6 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

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

Note: The BSP delivered with the phyBOARD-Mira-i.MX 6 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 i.MX 6 Technical Reference Manual, if such information is needed to connect customer designed applications.

The BSP is configured according to the hardware configuration including the expansion board delivered with the kit. Thus some functions of the phyBOARD-Mira-i.MX 6 might not be available if the corresponding pins and drivers are needed to support an expansion board. If the expansion board is removed, or exchanged the BSP must be exchanged, too.

From BSP version Am335x-PD14.1-rc1 on it is possible to configure the BSP in regard to the hardware configuration. This allows to easily adapt the BSP if an expansion board is attached, removed, or exchanged.

Preface

As a member of PHYTEC's new phyBOARD[®] product family the phyBOARD-Mira-i.MX 6 is one of a series of PHYTEC System on Modules (SBCs) that offer off-the-shelf solutions for a huge variety of industrial applications. The new phyBOARD[®] product family consists of a series of extremely compact embedded control engines featuring various processing performance classes. All phyBOARDS are rated for industry, cost optimized and offer long-term availability. The phyBOARD-Mira-i.MX 6 is one of currently six industrial-grade carrier boards which are suitable for series production and that have been realized in accordance with PHYTEC's new SBCplus concept. It is an excellent example of this concept.

SBCplus Concept

The SBCplus concept was developed to meet fine differences in customer requirements with little development effort and thus to greatly reduce the time-to-market.

Core of the SBCplus concept is the SBC design library (a kind of construction set) that consists of a great number of function blocks (so-called "building blocks") which are refined constantly. The recombination of these function blocks allows to develop a customer specific SBC within a short time. Thus, PHYTEC is able to deliver production-ready custom Single Board Computers within a few weeks at very low costs.

The already developed SBCs, such as the phyBOARD-Mira, each represent an intersection of different customer wishes. Because of that all necessary interfaces are already available on the standard versions, thus, allowing to integrate them in a large number of applications without modification. For any necessary detail adjustment extension connectors are available to enable adding of a wide variety of functions.

Cost-optimized with Direct Solder Connect (DSC) Technology

At the heart of the phyBOARD-Mira is the phyCORE-i.MX 6 System on Module (SOM). As with all SBCs of the phyBOARD[®] family the SOM is directly soldered onto the carrier board PCB for routing of signals from the SOM to applicable I/O interfaces. This "Direct Solder Connect" (DSC) of the SOM eliminates costly PCB to PCB connectors, thereby further reducing overall system costs, and making the phyBOARDS ideally suited for deployment into a wide range of cost-optimized and robust industrial applications.

Customized Expandability from PHYTEC

Common interface signals route to standard connector interfaces on the carrier board such as Ethernet, CAN, RS-232, and audio. Due to the easily modifiable phyBOARD design approach (see "*SBCplus concept*"), these plug-and-play interfaces can be readily adapted in customer-specific variants according to end system requirements.

Some signals from the processor populating the SOM also extend to the expansion, and A/V connectors of the phyBOARD-Mira. This provides for customized expandability according to end user requirements. Thus expandability is made easy by available plug-and-play expansion modules from PHYTEC.

- HDMI and LVDS/Parallel Displays
- Power Supply, with broad voltage range
- Industrial I/O (including WLAN)
- Home-Control Board (WiFi, KNX/EIB, I/O)
- M2M Board (GPS, GSM, I/O's)
- Debug Adapter

The default orientation of the expansion bus connectors is parallel and on the top side of the carrier board PCB. However, in custom configurations the connectors can be mounted on the PCB's underside. Connectors in perpendicular orientation can also populate the top or underside of the PCB. This enables maximum flexibility for orientation of expansion modules on the phyBOARD-Mira, as well as integration of the system into a variety of end application physical envelopes and form factors.

Easy Integration of Display und Touch

The phyBOARD and its expansion modules enable easy connection of parallel or LVDS based displays, as well as resistive or capacitive touch screens.

OEM Implementation

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.

Software Support

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.

Ordering Information

Ordering numbers:

phyBOARD-Mira-i.MX 6 Development Kit:

KPB-01501-xxx

phyBOARD-Mira-i.MX 6 SBC:

PB-01501-xxx

Product Specific Information and Technical Support

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-Mira-i.MX6.html> or

<http://www.phytec.eu/europe/support/faq/faq-phyBOARD-Mira-i.MX6.html>

Other Products and Development Support

Aside of the new phyBOARD[®] family, 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 OEM modules, which can be embedded directly into the user's peripheral hardware design.

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

**Declaration of Electro Magnetic Conformity of the PHYTEC
phyBOARD-Mira-i.MX 6**

PHYTEC Single Board Computers (henceforth products) are designed for installation in electrical appliances, or as part of custom applications, 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 obsolescence of a certain part, are being evaluated in order to take the right measures 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-Mira for phyCORE-i.MX 6 is a low-cost, feature-rich software development platform supporting the Freescale Semiconductor i.MX 6 microcontroller. Moreover, due to the numerous standard interfaces the phyBOARD-Mira-i.MX 6 can serve as bedrock for your application. At the core of the phyBOARD-Mira is the PCL-058/phyCORE-i.MX 6 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 i.MX 6 processor. Surrounding the SOM is the PBA-CD-06/phyBOARD-Mira carrier board, adding power input, buttons, connectors, signal breakout, and Ethernet connectivity amongst other things.

- Adding the phyCORE-i.MX 6 SOM into your own design is as simple as orderinDeveloped in accordance with PHYTEC's new SBCplus concept ([Preface](#))
- PHYTEC's phyCORE-i.MX 6 SOM
- Pico ITX standard dimensions (100 mm × 72 mm)
- Boot from MMC or NAND Flash
- Max. 1.2 GHz core clock frequency and up to four cores
- Two different power supply options (5 V via 3.5 mm combicon or 12 V – 24 V through external power module)
- One RJ45 jacks for 10/100/1000 Mbps Ethernet
- One USB Host interface brought out to an upright USB Standard-A connector, or at the mini PCI express connector
- One USB OTG interface available at an USB Micro-AB connector, or at the expansion connector
- One Secure Digital / Multi Media Memory Card interface brought out to a Micro-SD connector at the back side
- CAN interface at 2×5 pin header 2.54 mm
- One HDMI interface brought out to a standard type A connector
- One LVDS interface brought out to a 20 pin FFC connector at the backside, and separate connector for backlight supply and control
- One touch interface at 1×4 pin header 2.54 mm
- One LVDS camera interfaces compatible to PHYTEC phyCAM-S+ camera standard with I²C for camera control
- One PCI interface brought out to a Mini PCI Express connector; SIM-card signals are also available at the expansion connector
- RS-232 or RS-485 transceiver supporting UART3 incl. handshake signals with data rates of up to 1 Mbps (2×5 pin header 2.54 mm)
- Reset-Button
- One multicolor LED
- Audio/Video (A/V) connectors

- Expansion connector with different interfaces
- RTC
- Backup battery supply for RTC with external 2-pole pin-header or with Gold cap (lasts approx. 17 ½ days)
- Industrial temperature range (-40 °C to +85 °C)

1.1.1 Block Diagram

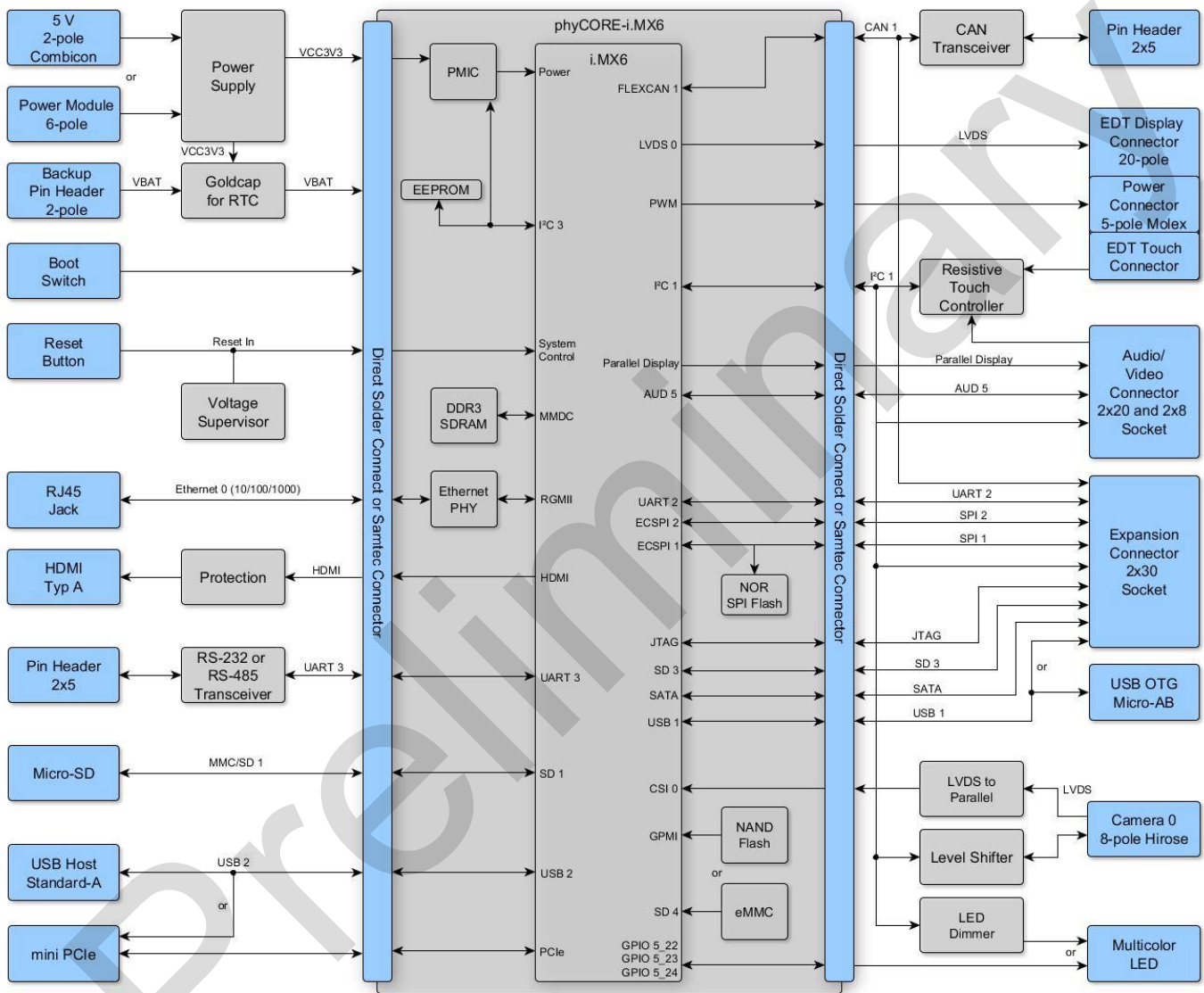


Figure 1: Block Diagram of the phyBOARD-Mira-i.MX 6

1.1.2 View of the phyBOARD-Mira-i.MX 6

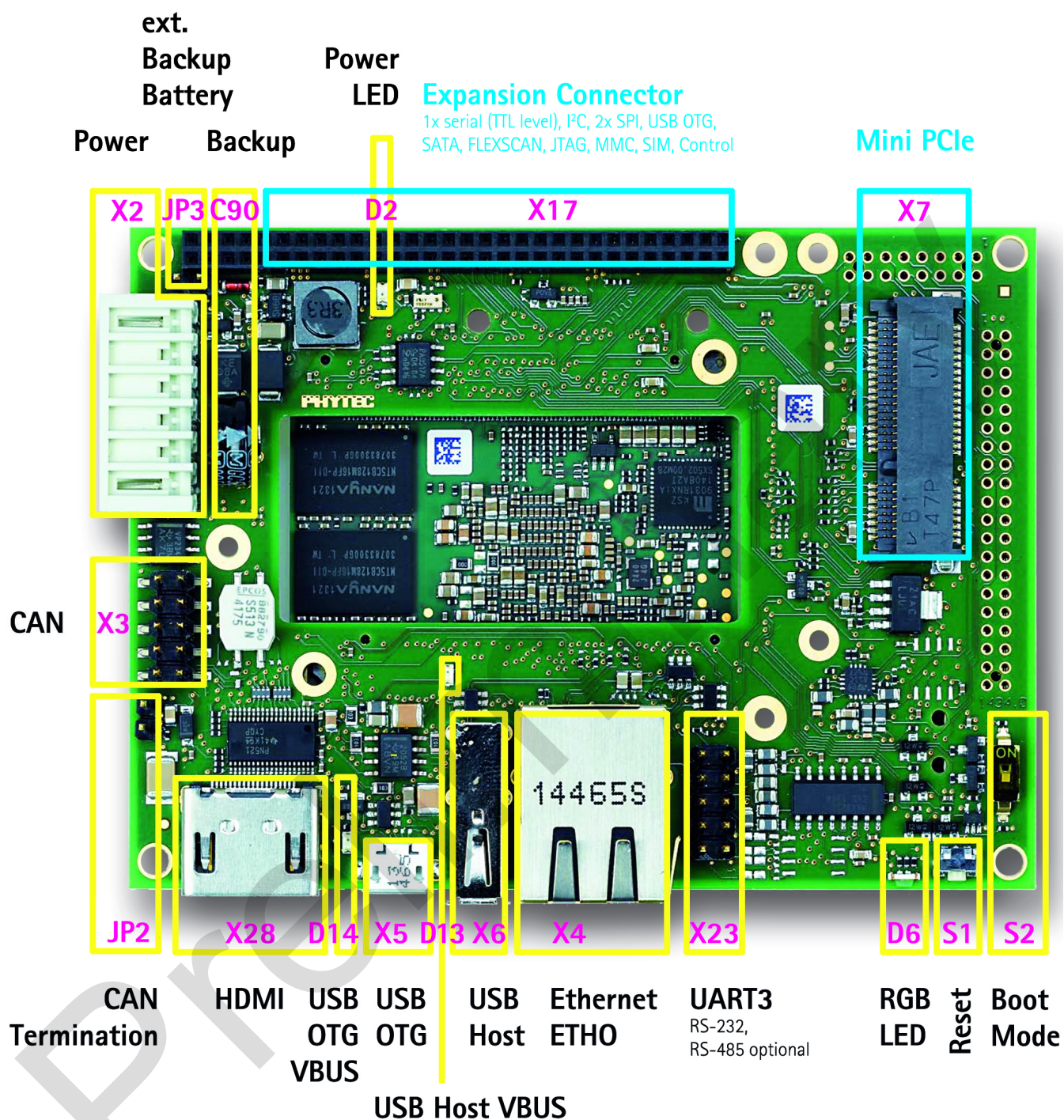


Figure 2: View of the phyBOARD-Mira-i.MX 6 (top)

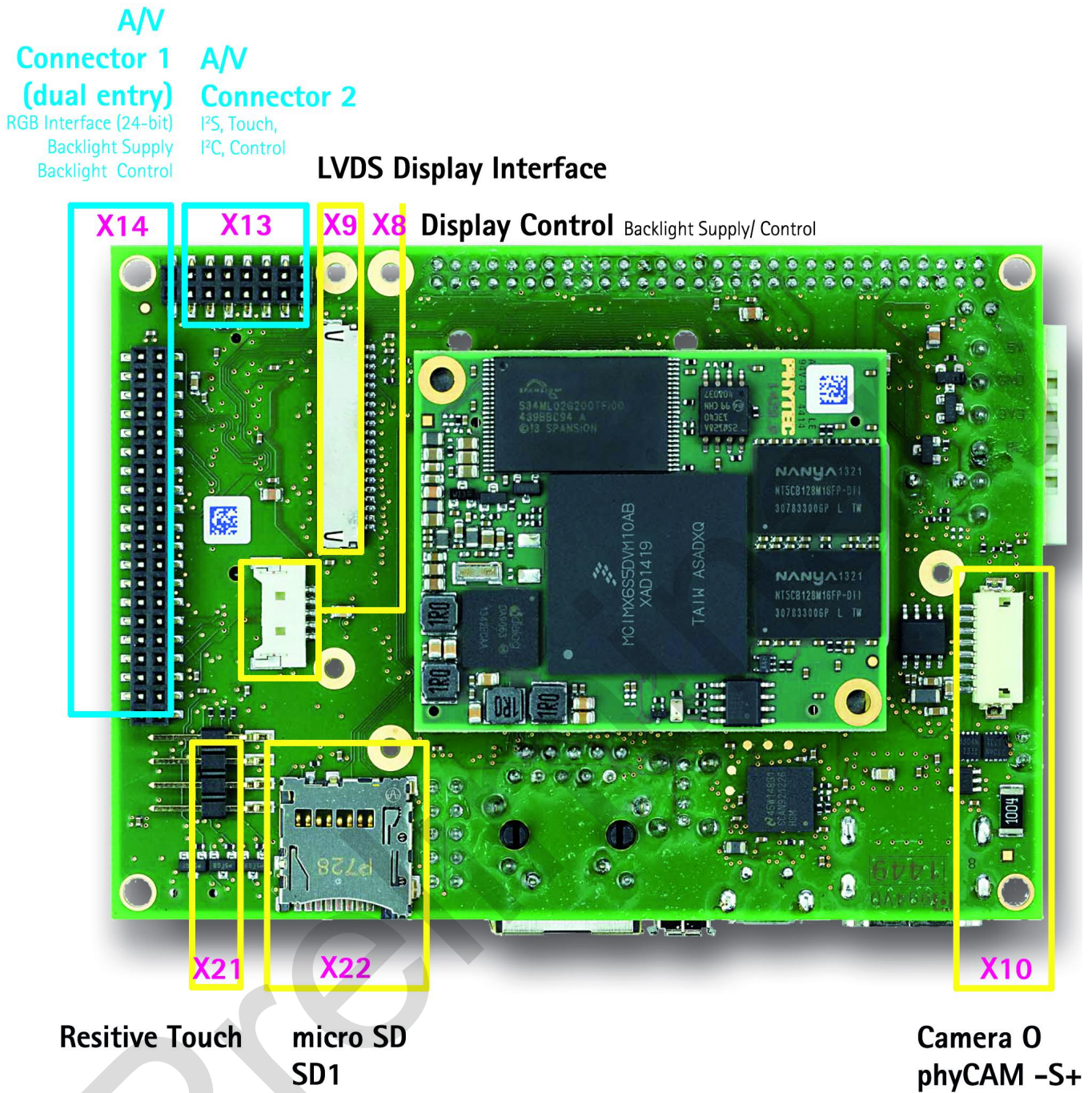


Figure 3: View of the phyBOARD-Mira-i.MX 6 (bottom)

1.2 Software Overview

1.2.1 Ubuntu

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

The *Ubuntu* release which we deliver is *14.04.2* and was released on 20. February 2015. *Ubuntu 14.04* code name "*Trusty Tahr*" is designated as a **Long Term Support (LTS)** release and the first stable release was on 17 April 2014. LTS means that it will be supported and updated for five years.

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

1.2.2 Eclipse

The *Eclipse* platform provides support for *C/C++*. Because the *Eclipse* platform is only a framework for developer tools, it does not support *C/C++* 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++* toolchain.

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 Qt Creator

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

1.2.4 The GNU Cross Development Toolchain

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 does not 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™-A9 architecture on the phyBOARD-Mira-i.MX 6 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 toolchain and cross-compiled libraries. In this Application Guide we are using the *GNU C/C++* cross development toolchain.

2 Application Programming

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

We assume that you have first completed our QuickStart Guide successfully.

	<p>As all changes on the example projects will be lost if you proceed using the live environment we recommend to now install our modified <i>Ubuntu</i> Live System. If you only want to make your own fast experience with our phyBOARD-Mira-i.MX6-Kit you can continue with section 2.2 "Working with Eclipse".</p>
---	---

	<p>To ensure successful introduction to the development with the phyBOARD-Mira-i.MX6 we strongly recommend continuing with the modified <i>Ubuntu</i>, either in a live environment, or completely installed on your PC as described in the next section. Nonetheless; if you want to use your already existing environment we explain how to modify your system to get the same experience as with our Live System in section 0 "Flash the device tree" by typing:</p> <pre>erase /dev/nand0.oftree.bb cp /mnt/tftp/zImage-imx6q-phytec-phyboard-mira.dtb /dev/nand0.oftree.bb</pre> <p>Write the root file system into flash by typing:</p> <pre>ubiformat /dev/nand0.root ubiattach /dev/nand0.root ubimkvol /dev/ubi0 root 0 cp /mnt/tftp/phytec-qt4demo-image-phyboard-mira-imx6-*.ubifs /dev/ubi0.root ".</pre>
---	---

2.1 Installing our modified Ubuntu Live System

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 Live System on your computer or into a virtual machine. We recommend purchasing our bootable USB flash drive or to create your own USB flash drive as explained in our QuickStart Guide please verify the boot priorities of your system. Plug-in the USB flash drive and restart your PC system. If you have our iso-image you can install the Live System in a virtual machine. If you want to install our Live System native you must create a bootable USB flash drive for example with the help of the UNetbootin Tool under Linux.



We recommend to have at least 70 GB of free disk space available to install the customized Live System.

- There are two methods to install our Live System, but for both the first step is to select **Install Ubuntu**.
- The *Preparing to install Ubuntu...* window appears. From *Ubuntu* it is advised that you select **Download updates while installing** and **Install this third-party software now**. Click 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 chosen one click on **Continue**.
- The *Install Ubuntu...* 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 and connect the phyBOARD-Mira-i.XM 6 to your host. 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*.

2.2.1 Programming in the C/C++ Perspective

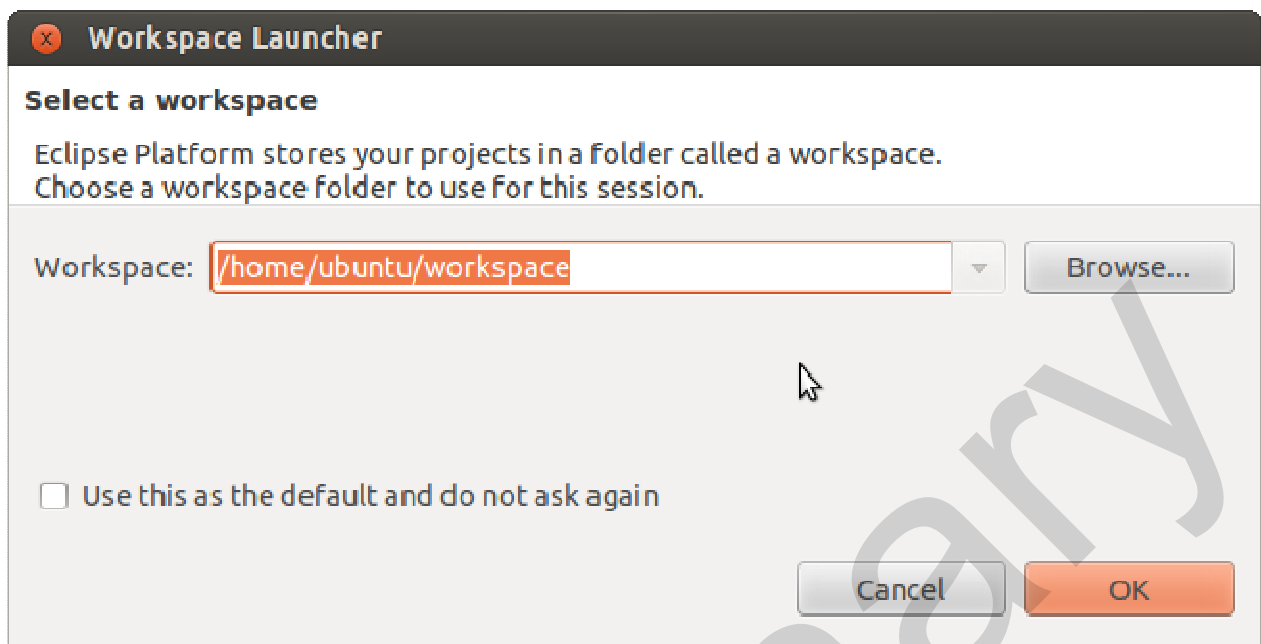
We are starting with the *C/C++* workbench. Therefore you will import an existing Eclipse project into your workspace. The imported example project will be compiled with the cross compiler. After that, you will copy and execute the newly created program on the target.

2.2.1.1 Work with the Demo Project

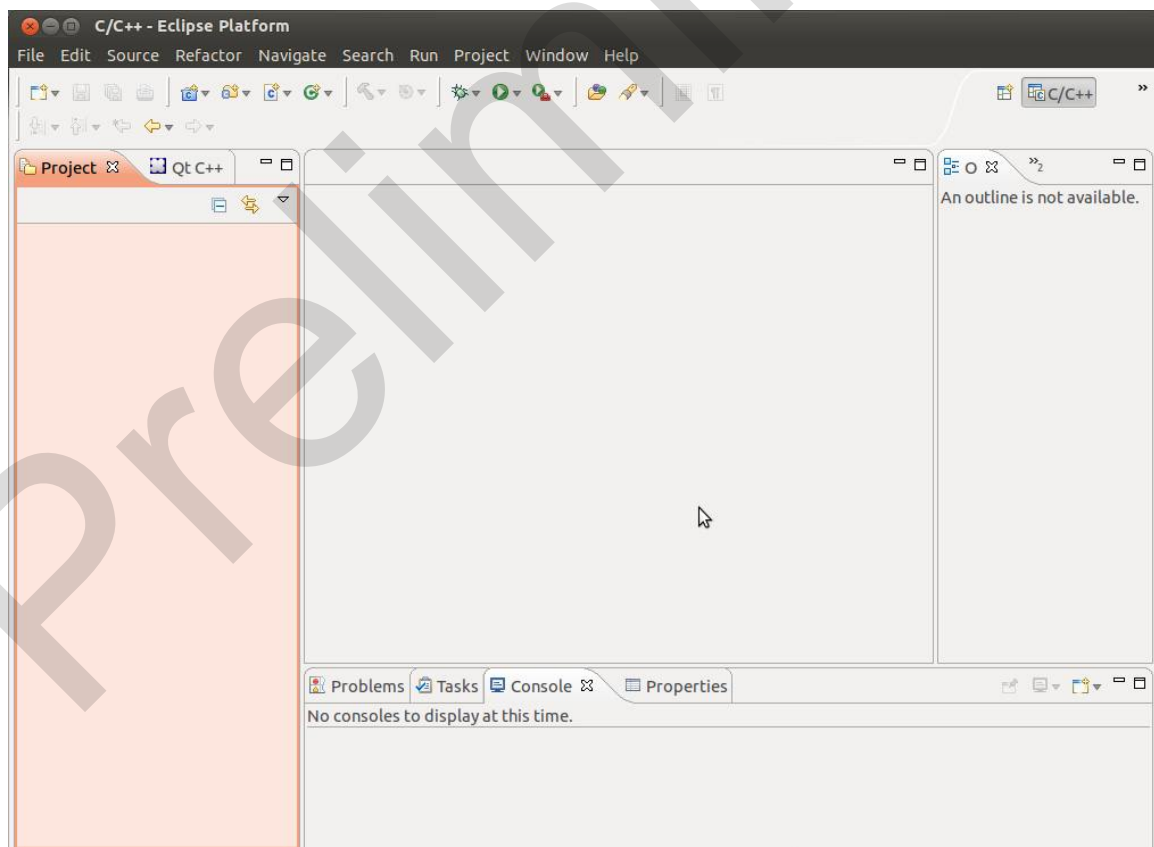
- Click the **Eclipse icon** to start the application. You can find this icon on your desktop.



- Confirm the workspace directory with **OK**

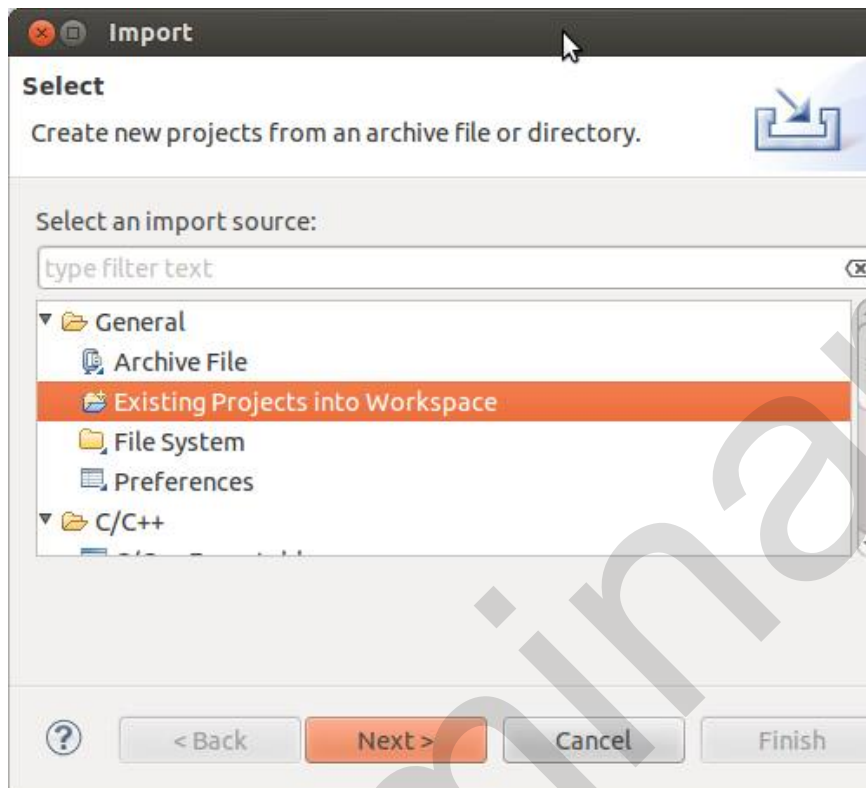


Now you can see the *Eclipse* workbench.

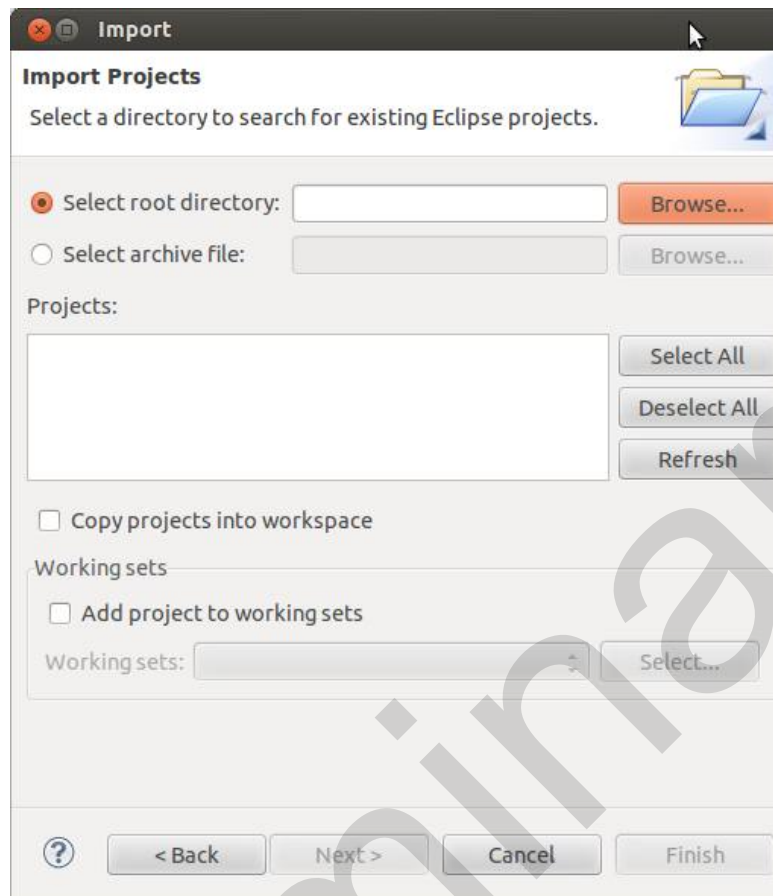


First we will import an existing project.

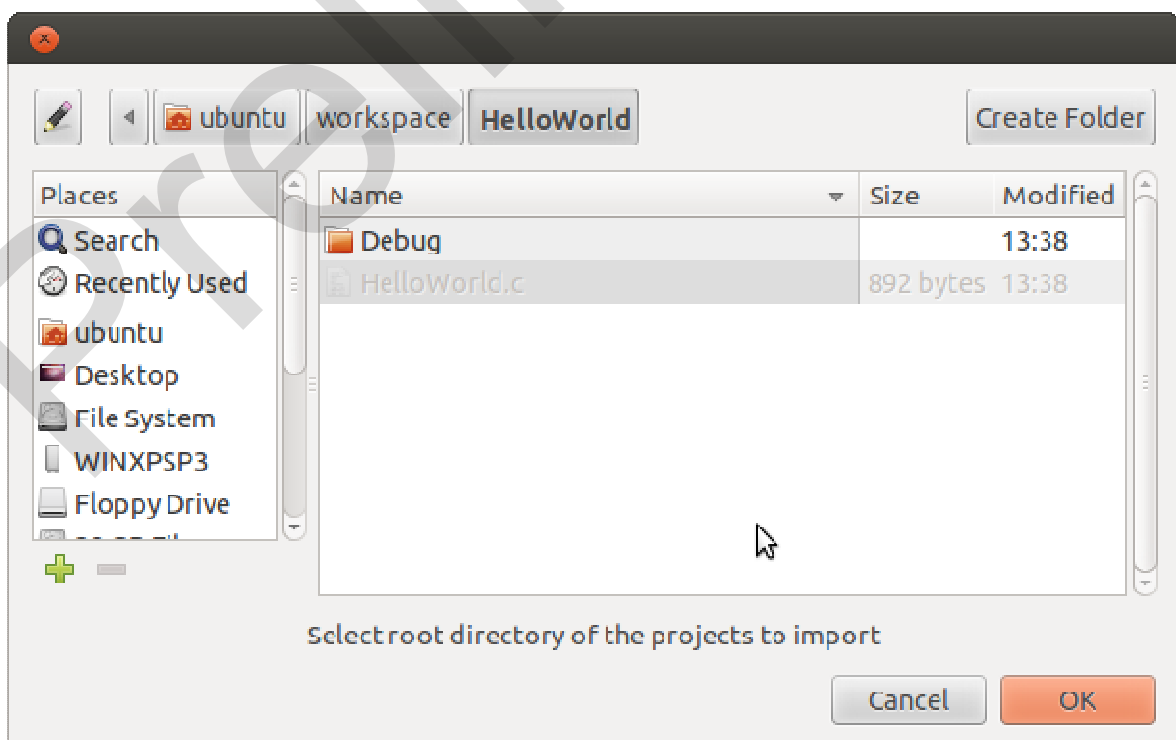
- Select **File ► Import** from the menu bar
- Select **Existing Projects into Workspace** and click **Next**



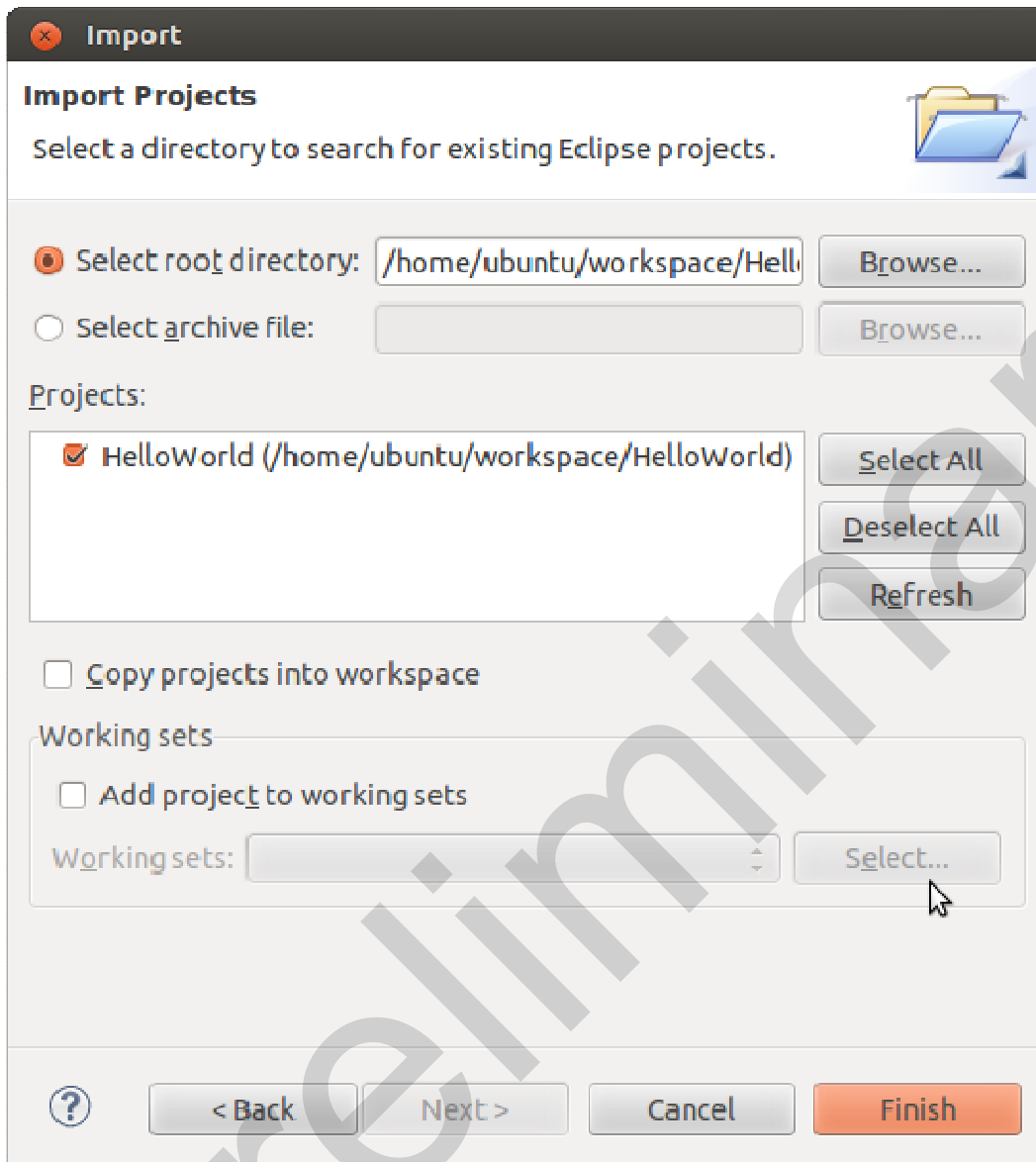
- Select **Browse**



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



- Select **Finish** to import the project



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

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

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

The screenshot shows the Eclipse IDE interface. The main editor window displays the source code for `HelloWorld.c`:

```

MA 02110-1301, USA.
*/
#include <stdio.h>

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

```

The console window shows the following output:

```

CDT Build Console [HelloWorld]

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

make all
Building target: HelloWorld
Invoking: GCC C Linker
arm-cortexa8-linux-gnueabi-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 you do not get this result verify that you have the target connected to your host, and that the network has been configured as explained in our QuickStart Guide.



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.

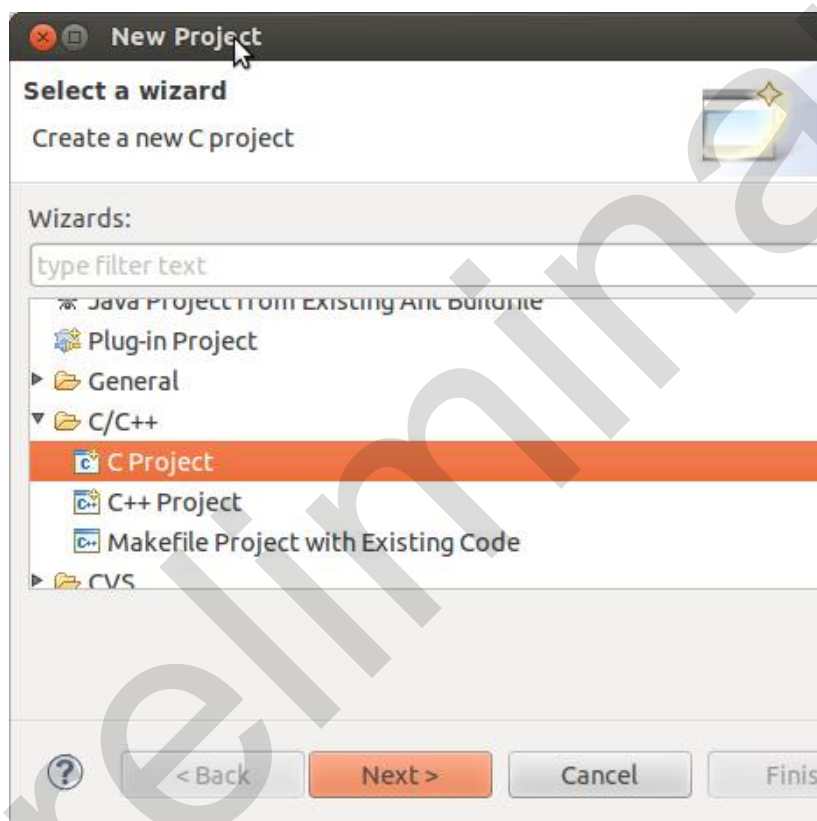
2.2.1.2 Creating a New Project

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.

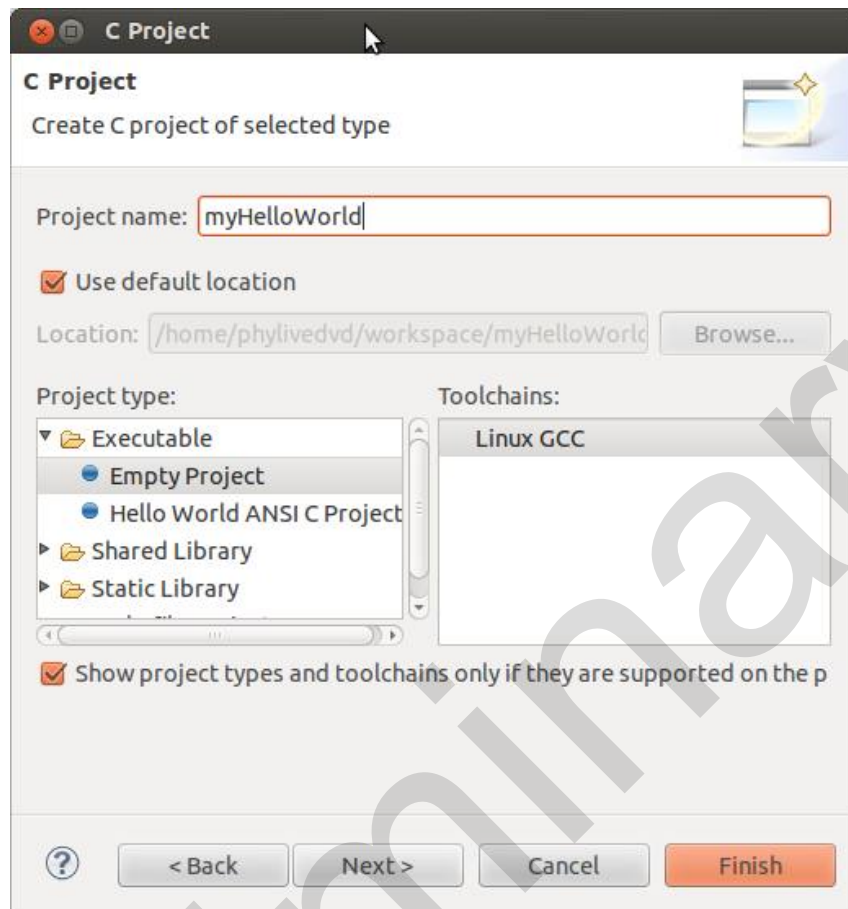
- Open *Eclipse* if it is not already opened
- Select **File ► New ► Project** from the menu bar.

A new dialog opens.

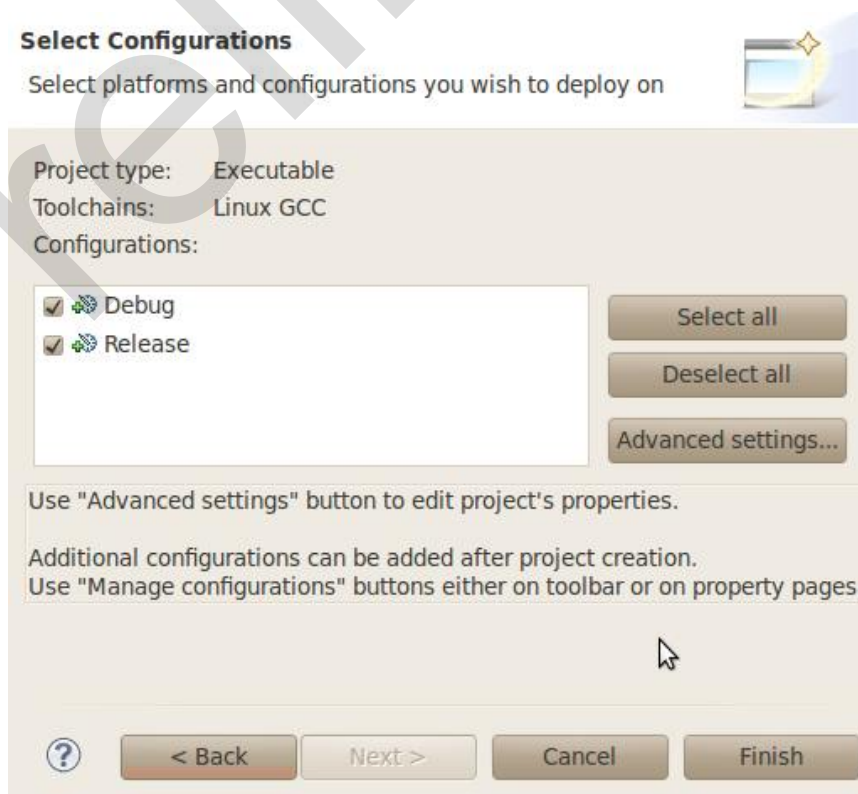
- Select **C Project** and click **Next**



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

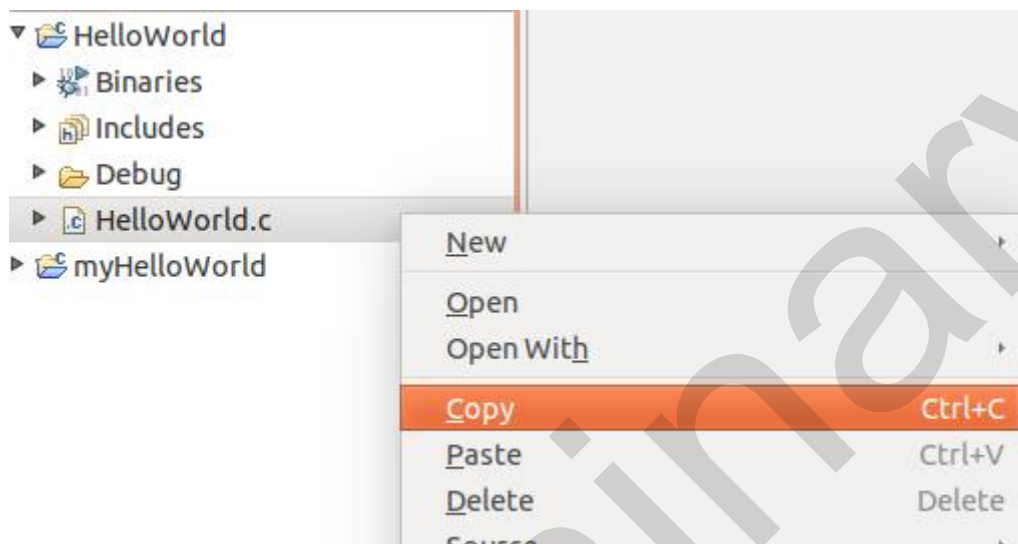


- Click **Finish**



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

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



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

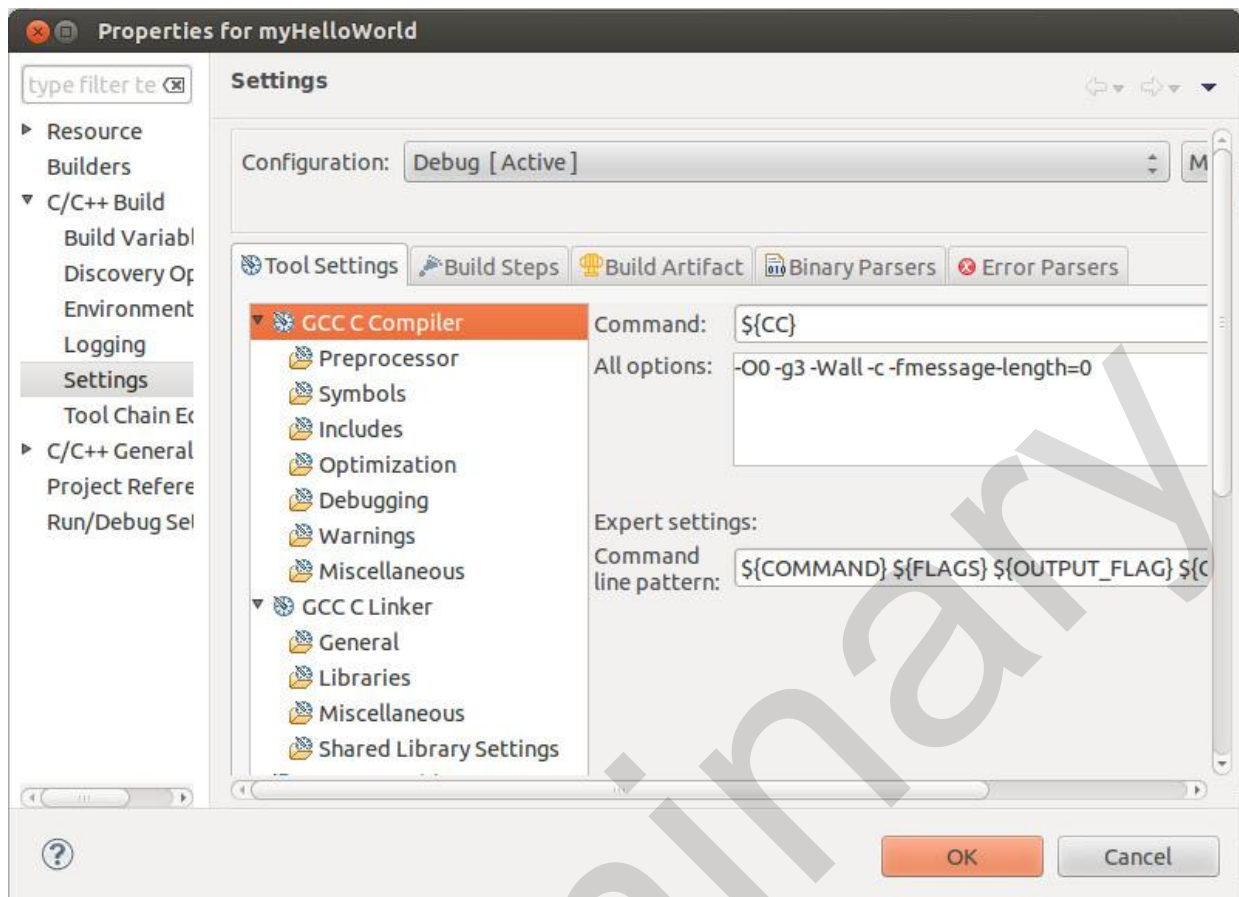
If *Build Automatically* from the *Project* menu is selected, the *HelloWorld* application will now be compiled and created with the standard *GCC C/C++* compiler suitable for your host machine. You will find the executable file, which can only run on your host system, in the *workspace/myHelloWorld/Debug*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

To compile your project for the phyCORE-AM335x instead, you will have to use the *GNU C/C++* cross compiler.

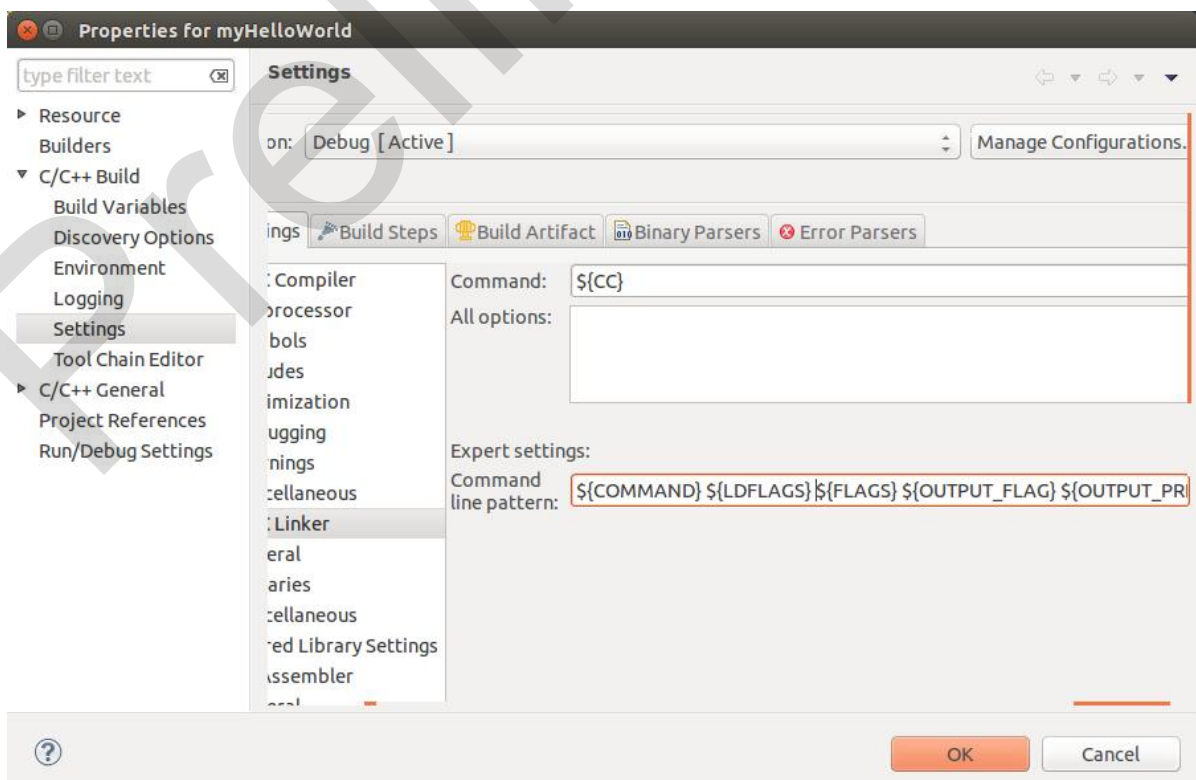
- Right-click the **myHelloWorld** project and choose **Properties**

The *Properties* dialog appears.

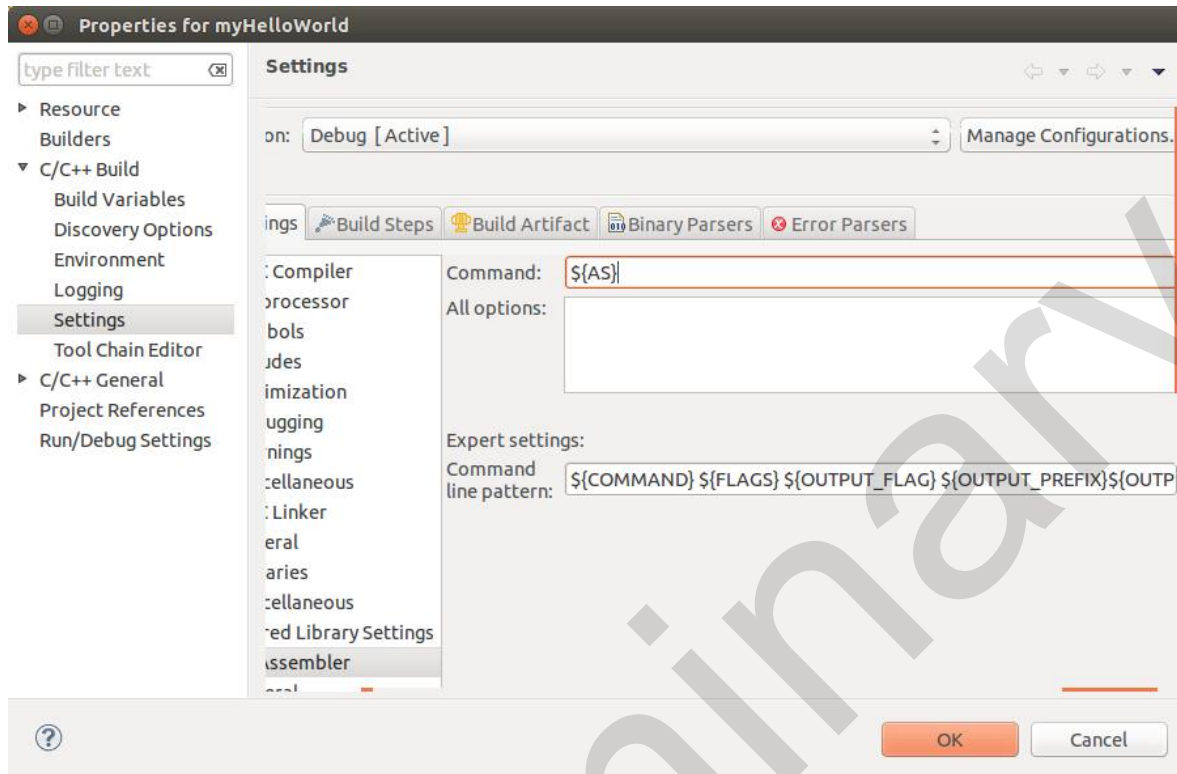
- Select **C/C++ Build ► Settings**
- Enter `${CC}` into the *Command* input field
- Select **GCC C Compiler**



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



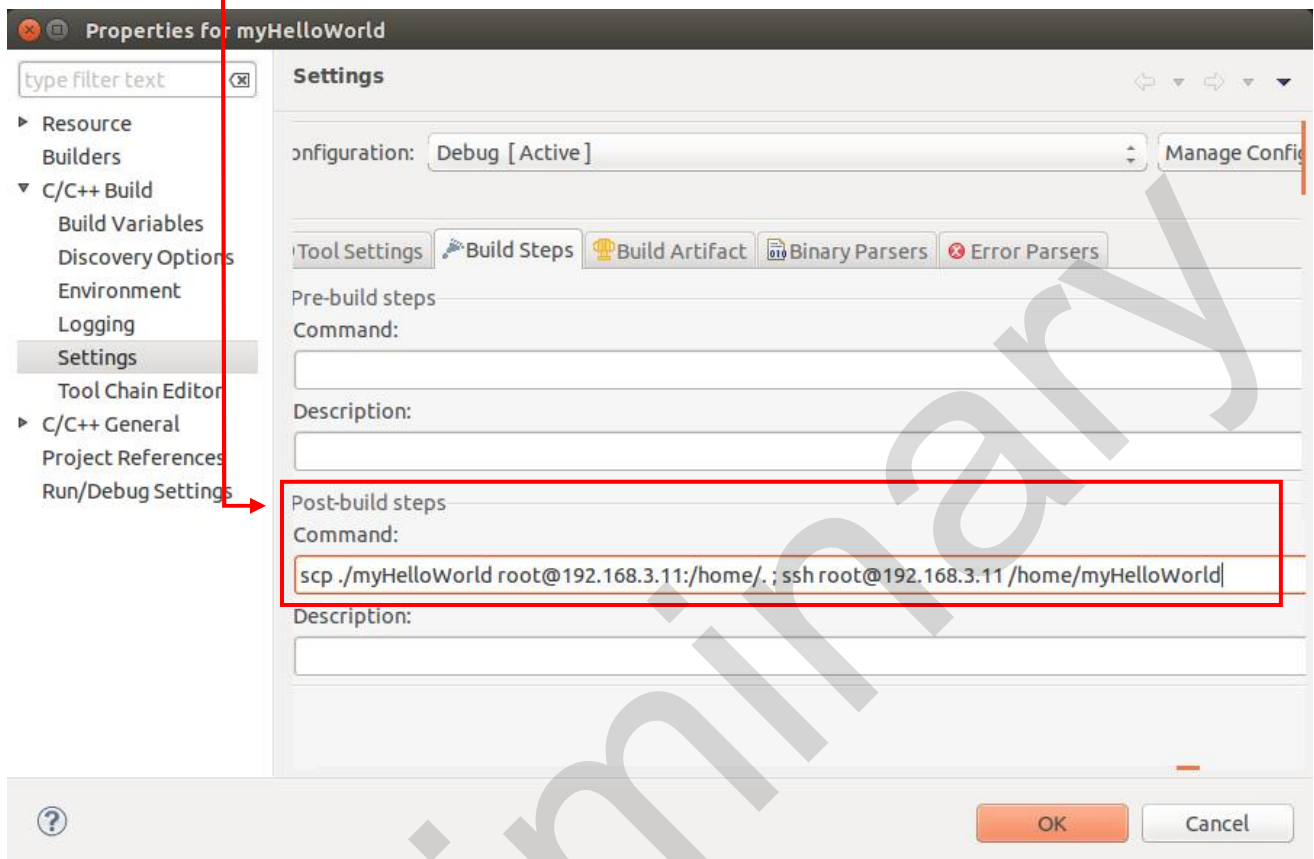
- Select **GCC Assembler**
- In the *Command* input field, change the default as to `${AS}`



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

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

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



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

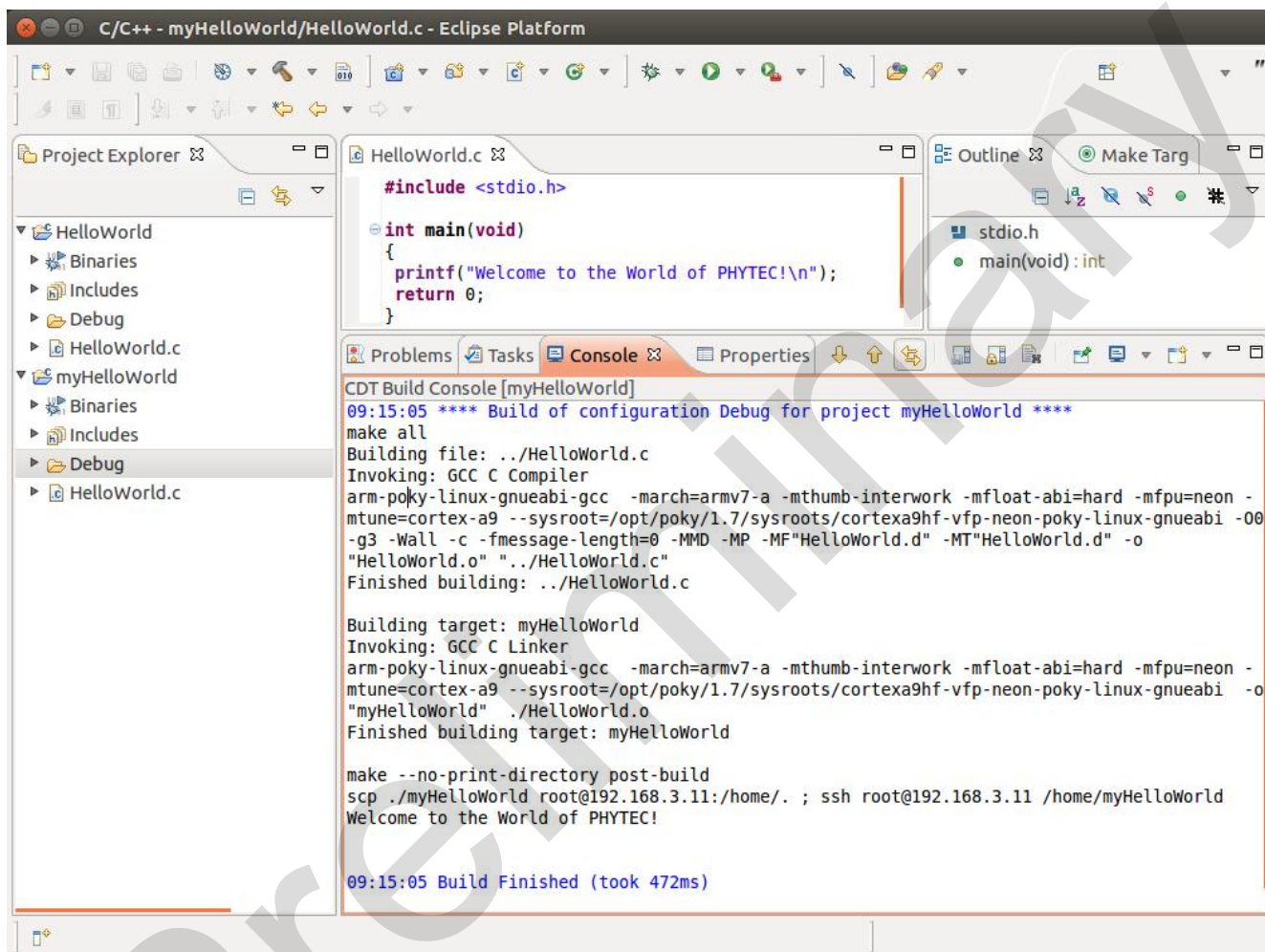
- Click **Apply**
- Click **OK**
- Select **Project** ► **Clean** from the menu bar

- Confirm with **OK**

The project will be rebuilt.

- Select the *Console* tab

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



The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for file operations and running. The Project Explorer on the left shows a project named 'myHelloWorld' with sub-projects 'Binaries', 'Includes', 'Debug', and 'HelloWorld.c'. The main editor displays the source code for 'HelloWorld.c':

```
#include <stdio.h>

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

The Console tab at the bottom shows the following output:

```
CDT Build Console [myHelloWorld]
09:15:05 **** Build of configuration Debug for project myHelloWorld ****
make all
Building file: ../HelloWorld.c
Invoking: GCC C Compiler
arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -
mtune=cortex-a9 --sysroot=/opt/poky/1.7/sysroots/cortexa9hf-vfp-neon-poky-linux-gnueabi -O0
-g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"HelloWorld.d" -MT"HelloWorld.d" -o
"HelloWorld.o" "../HelloWorld.c"
Finished building: ../HelloWorld.c

Building target: myHelloWorld
Invoking: GCC C Linker
arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb-interwork -mfloat-abi=hard -mfpu=neon -
mtune=cortex-a9 --sysroot=/opt/poky/1.7/sysroots/cortexa9hf-vfp-neon-poky-linux-gnueabi -o
"myHelloWorld" ../HelloWorld.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 /home/myHelloWorld
Welcome to the World of PHYTEC!

09:15:05 Build Finished (took 472ms)
```



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.

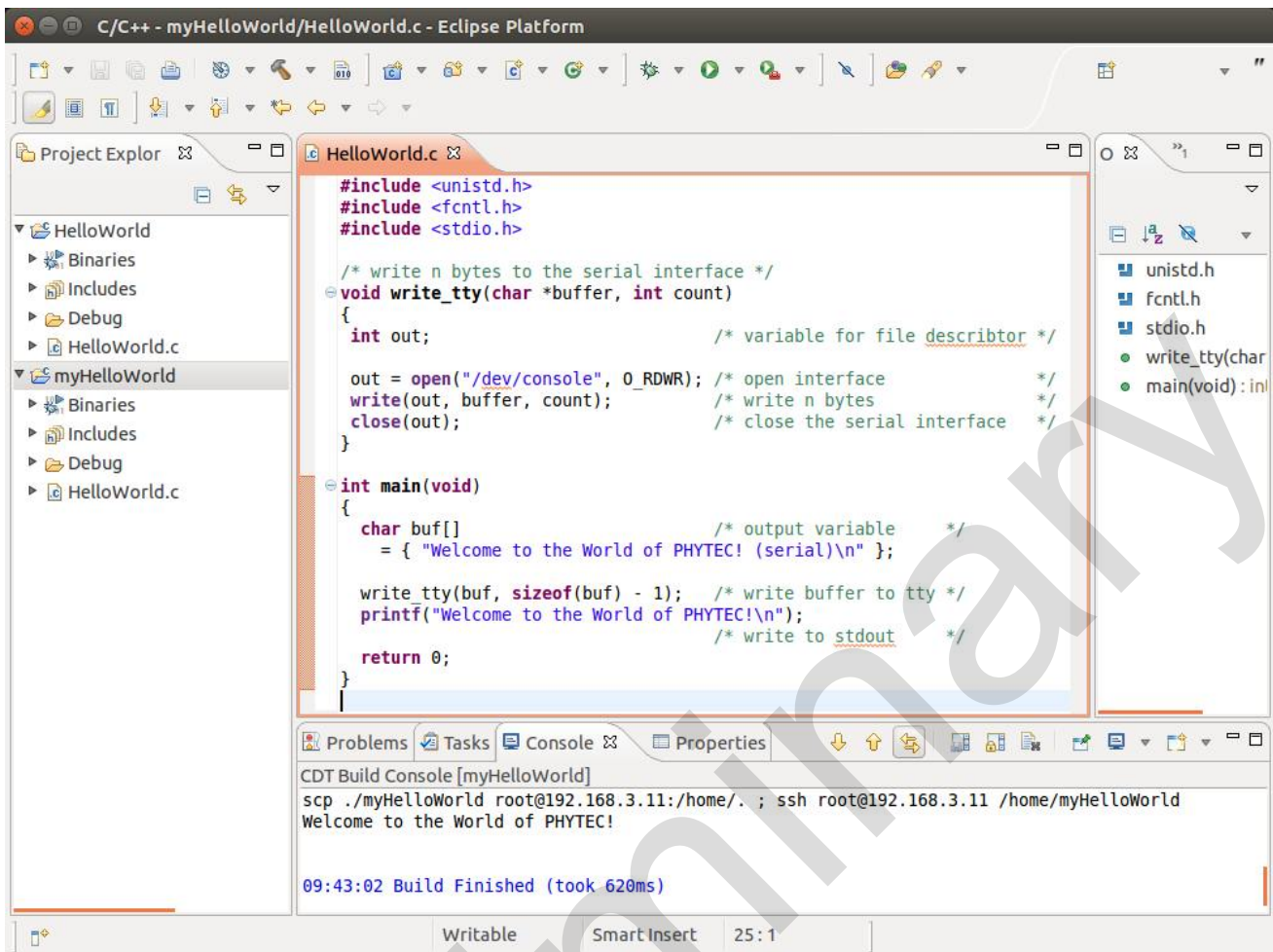
- Open *Eclipse* if it is not opened yet
- Double-click **HelloWorld.c** in the *myHelloWorld* project
- First include the following two additional header files:

```
#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-Mira-i.MX 6, 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



The screenshot shows the Eclipse IDE interface. The main editor window displays the source code for `HelloWorld.c`. The code includes headers for `unistd.h`, `fcntl.h`, and `stdio.h`. It defines a `write_tty` function that opens the serial interface, writes data, and closes it. The `main` function uses `write_tty` to send a message to the serial port and also prints it to the console. The console window at the bottom shows the output of the program: "Welcome to the World of PHYTEC!". The status bar at the bottom indicates "Writable", "SmartInsert", and "25:1".

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

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

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

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

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

CDT Build Console [myHelloWorld]
scp ./myHelloWorld root@192.168.3.11:/home/. ; ssh root@192.168.3.11 /home/myHelloWorld
Welcome to the World of PHYTEC!
09:43:02 Build Finished (took 620ms)

- Save your program after changing the code

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

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



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

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

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



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

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

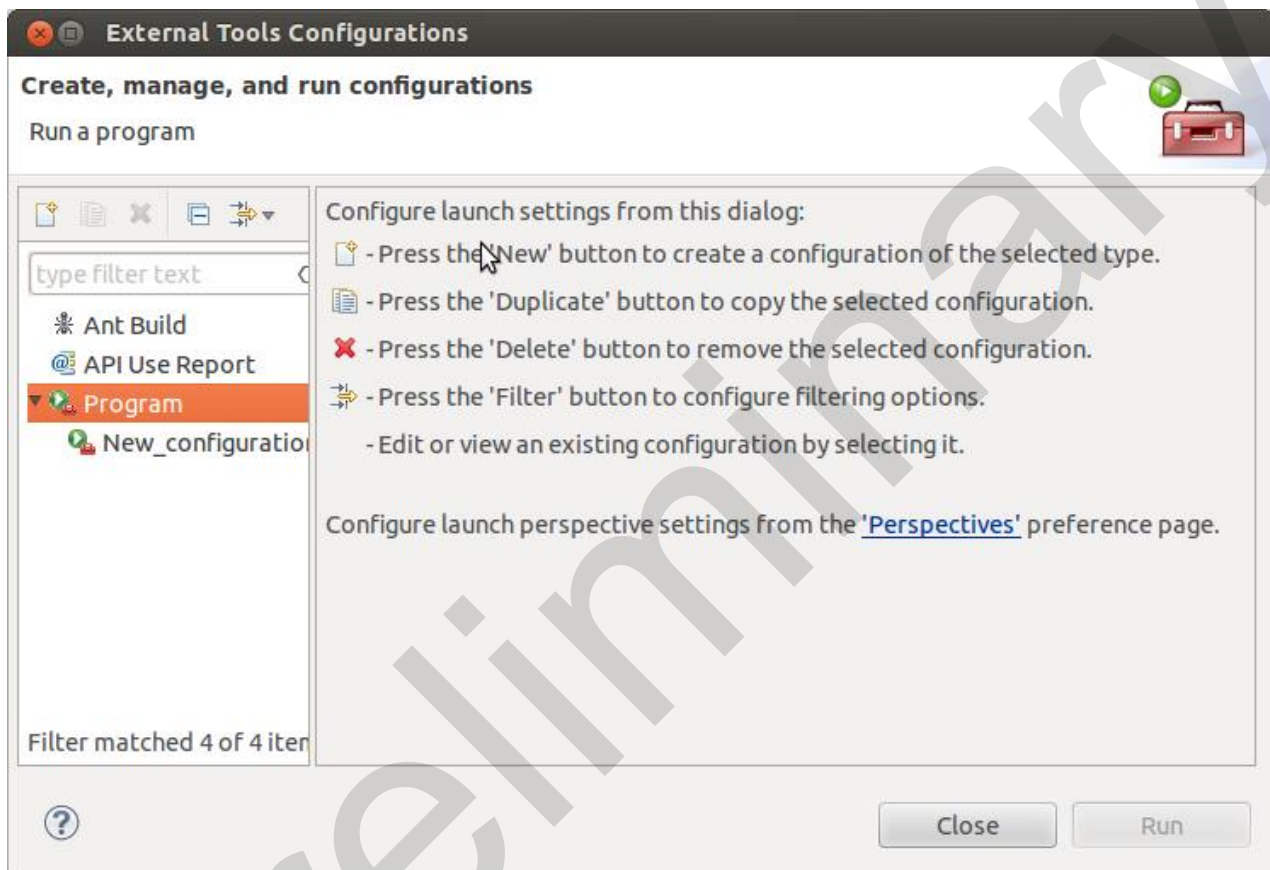
This procedure is quite typical for Linux, because Linux treats everything like a file.

2.2.1.4 Starting a Program out of Eclipse on the Target

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

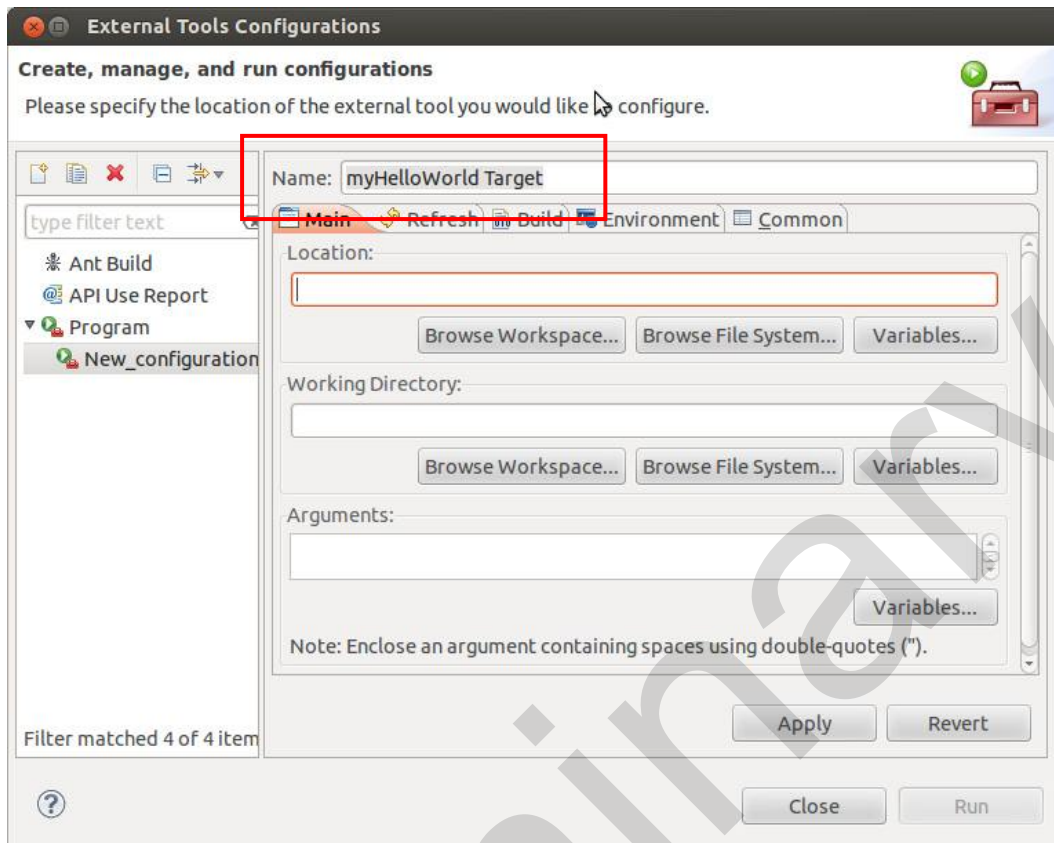
After compiling a project in *Eclipse*, the program is copied to the target and directly executed. A program can also be executed on the target without compiling a project. In the following section you will learn how to start a program on the target as an external tool.

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

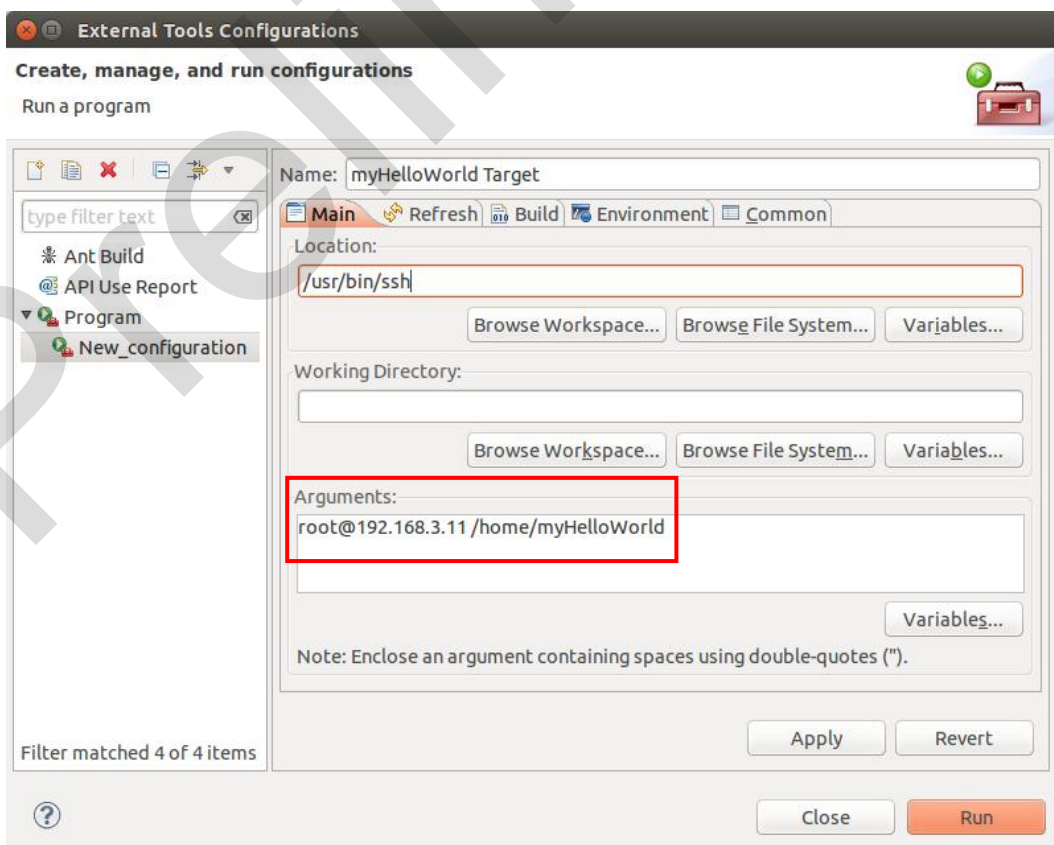


- Double-Click **Program** and select **New_configuration**

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



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



- Select **Apply**
- Select **Run**

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

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



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

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 Application Guide 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. These have already been added while building the program.



- Open *Microcom*
- Type `root` and press **Enter**
- Start the *GDB* server:
`gdbserver 192.168.3.11:10000 /home/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** under *C/C++ Application* (to expand it double click on it)

Create, manage, and run configurations



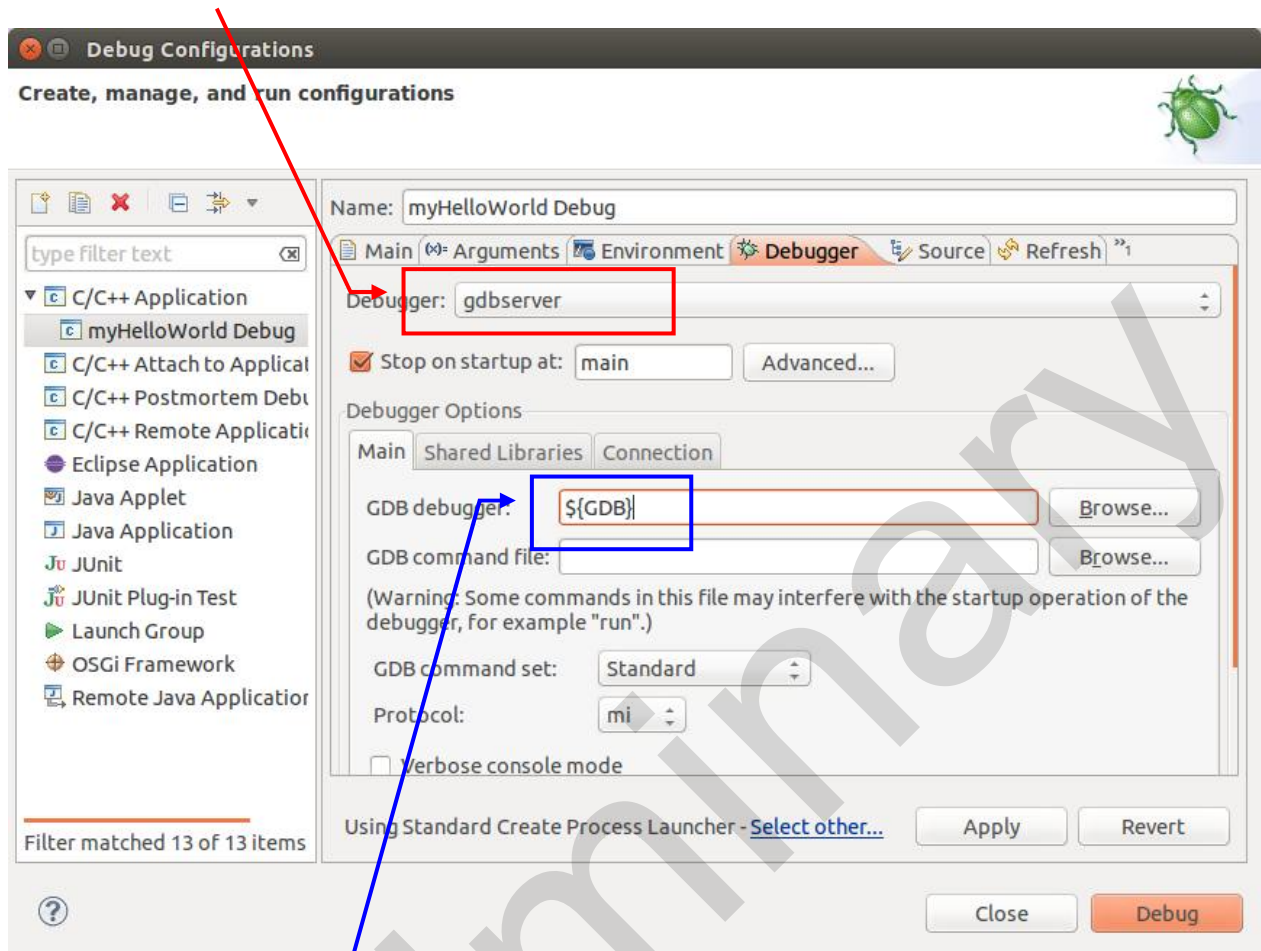
Multiple launchers available - select one to continue

The screenshot shows the Eclipse IDE configuration dialog for a C/C++ application. The left sidebar lists various launchers, with 'myHelloWorld' selected under 'C/C++ Application'. The main area shows the configuration for 'myHelloWorld', including fields for 'Project' (myHelloWorld) and 'C/C++ Application' (Debug/myHelloWorld). The 'Debugger' tab is active, and the 'Apply' button is highlighted. A warning message at the bottom states 'Multiple launchers available - Select one...'. Buttons for 'Close' and 'Debug' are visible at the bottom right.

- Select the **Debugger** tab

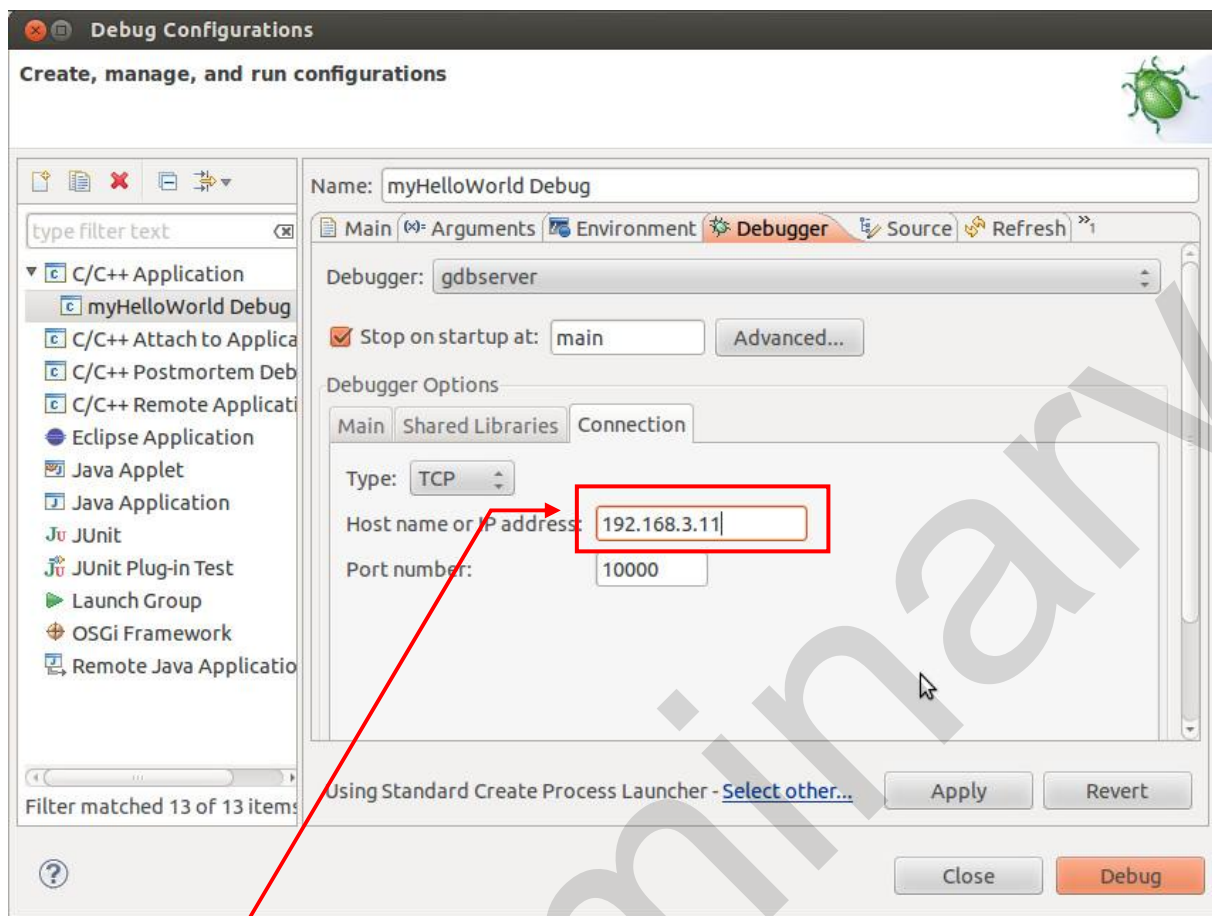
The screenshot shows the Eclipse IDE configuration dialog for a C/C++ application, now in the 'Debugger' tab. The left sidebar lists various launchers, with 'myHelloWorld Debug' selected under 'C/C++ Application'. The main area shows the configuration for 'myHelloWorld Debug', including fields for 'Debugger' (gdbserver), 'Stop on startup at' (main), and 'Debugger Options' (GDB debugger: gdb, GDB command file: .gdbinit, GDB command set: Standard, Protocol: mi). The 'Apply' button is highlighted, and the 'Debug' button is now orange. A warning message at the bottom states 'Using Standard Create Process Launcher - Select other...'. Buttons for 'Close' and 'Debug' are visible at the bottom right.

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



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

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



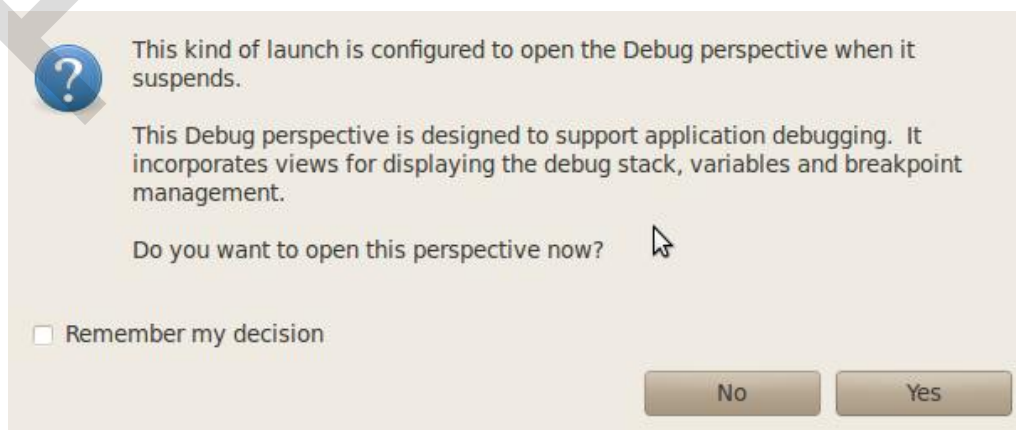
- Enter 192.168.3.11 (the target's IP address) in the *Host name* input field

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

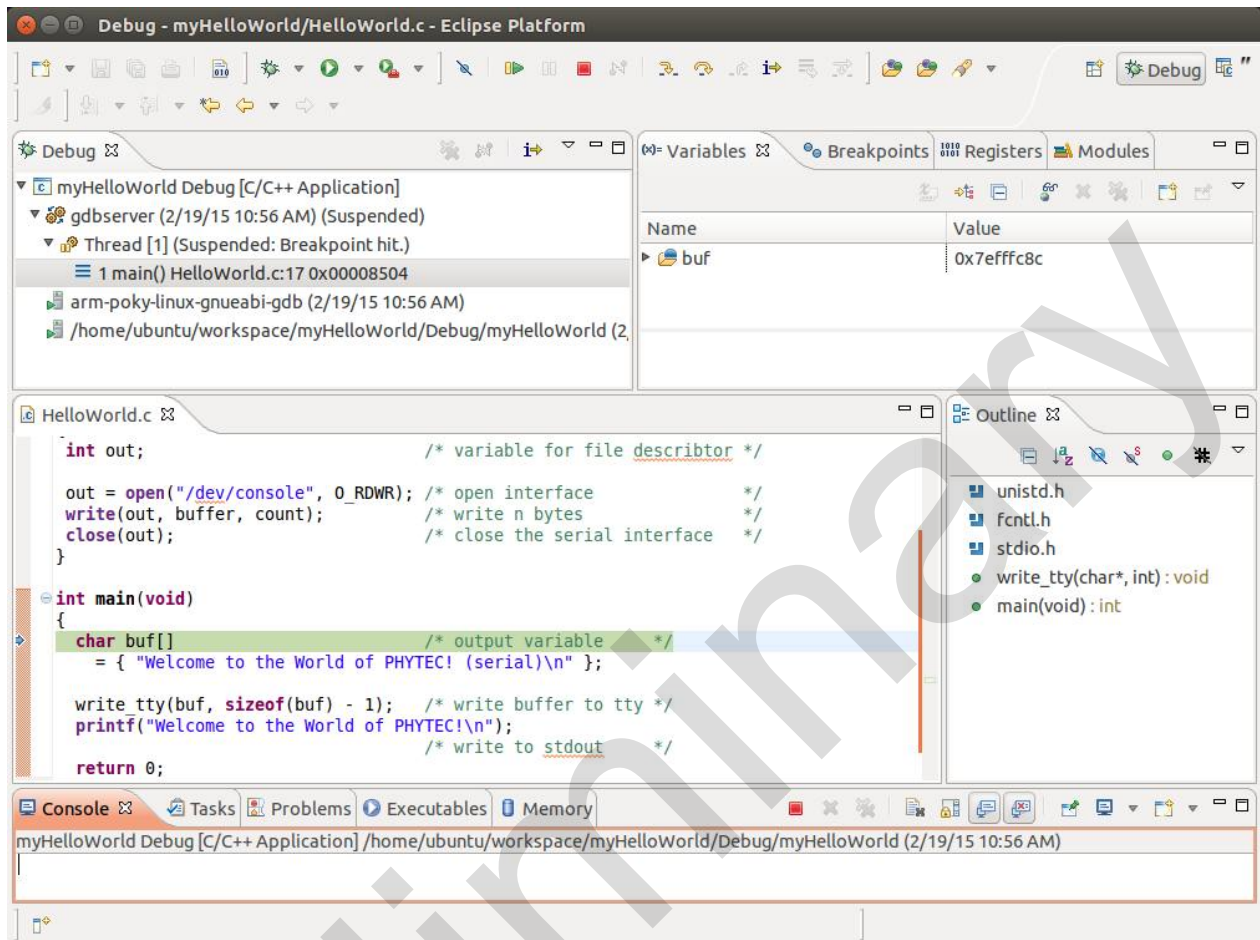
- Click **Apply**
- Click **Debug**

A new dialog appears.

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



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

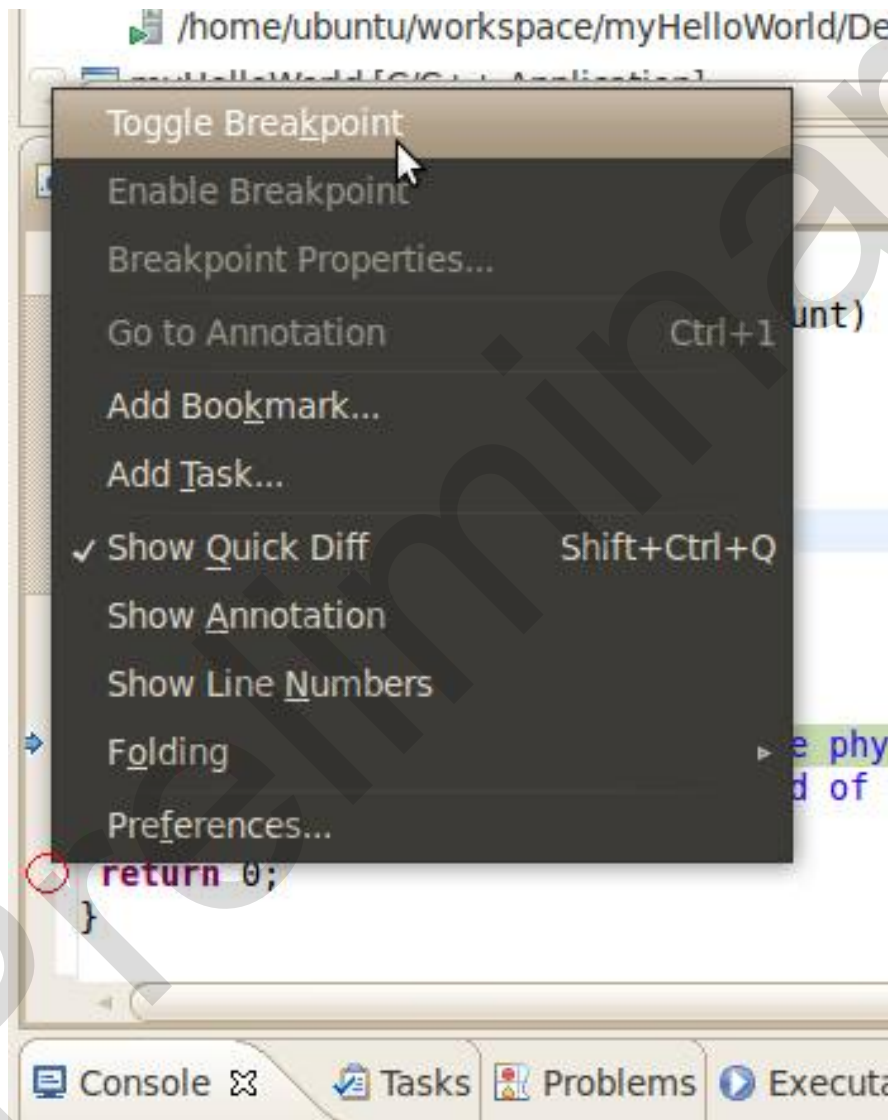


You have configured your project for remote debugging. You have started the *GNU* debugger in *Eclipse* and connected the host's *GDB* with the target's *GDB* server. You can now start to debug the project.

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.

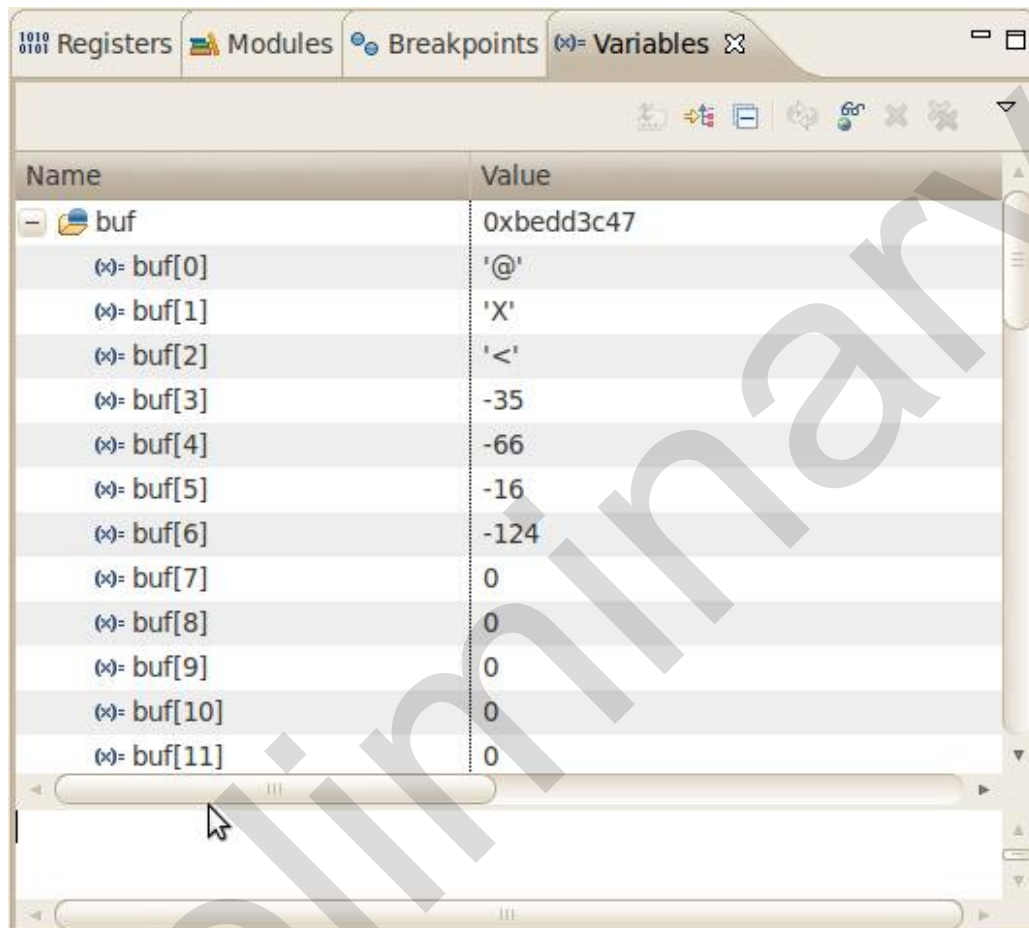
- Select the last line in `main()`
- Right-click into the small grey border on the left-hand side and select **Toggle Breakpoint** to set a new breakpoint



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.

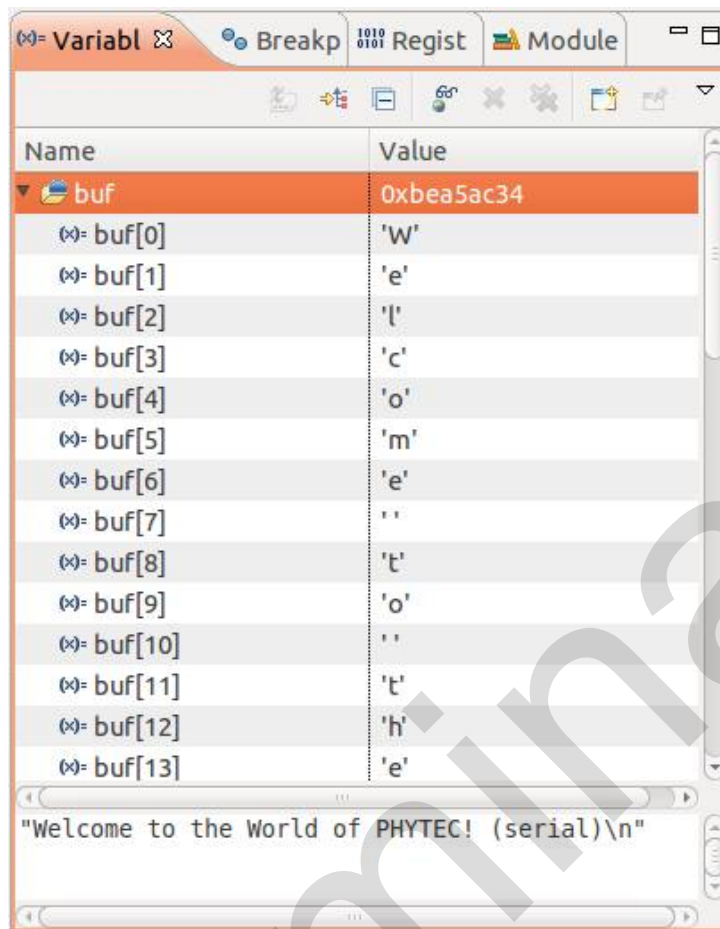
- Expand **buf** in the *Variables* window



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



- Click on the variable *buf*

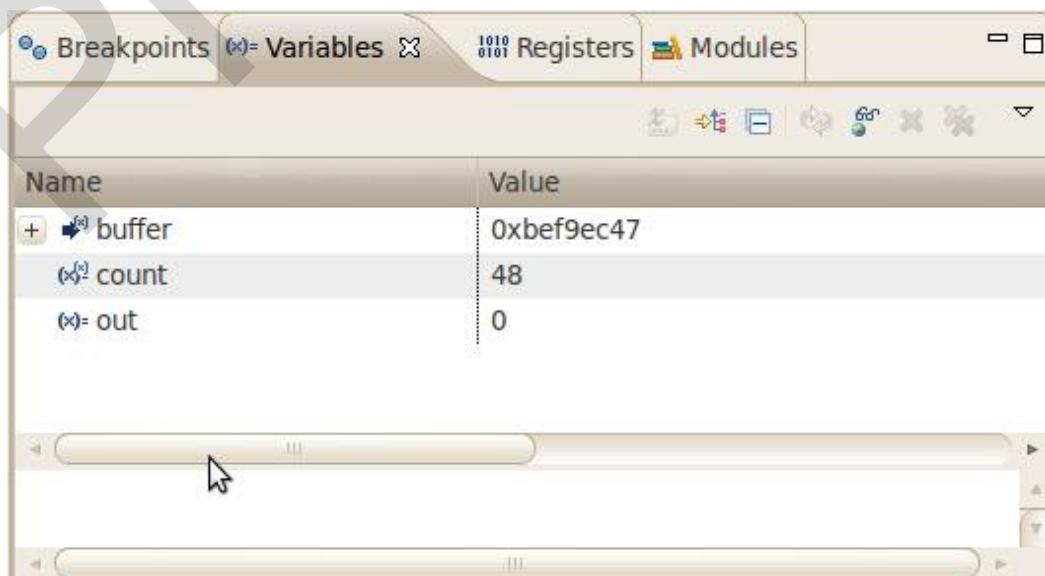


- Then click the button **Step into** to enter the function *write_tty()*



- The debugger stops in *write_tty()*

You will see the following variable window:



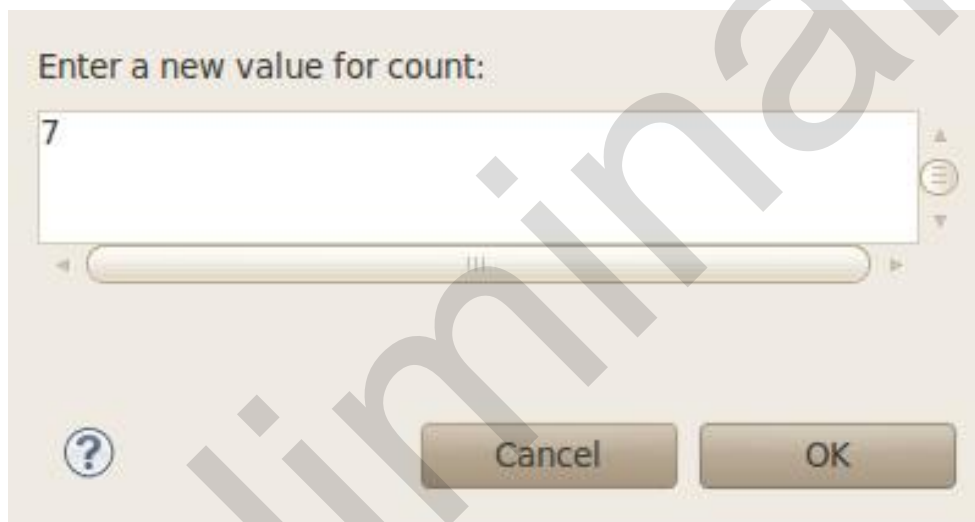
- Click on the variable **buffer**

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

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**



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



- Switch to *Microcom*

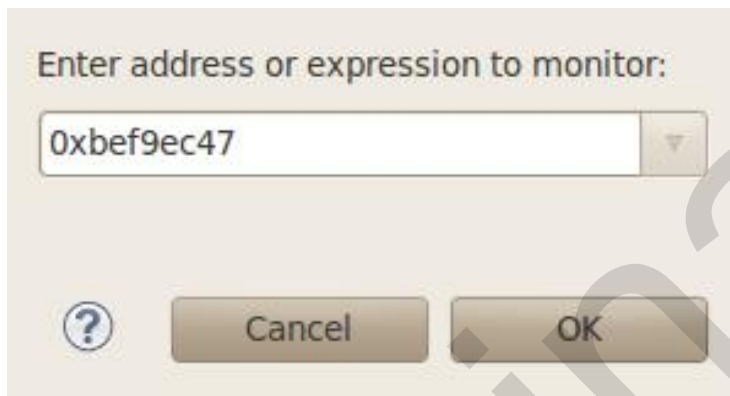
```
root@phyBOARD-WEGA-AM335x:~# gdbserver 192.168.3.11:10000 myHelloWorld
Process myHelloWorld created; pid = 827
Listening on port 10000
Welcome to the World of PHYTEC! (serial)
Remote debugging from host 192.168.3.10
Welcome
```

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

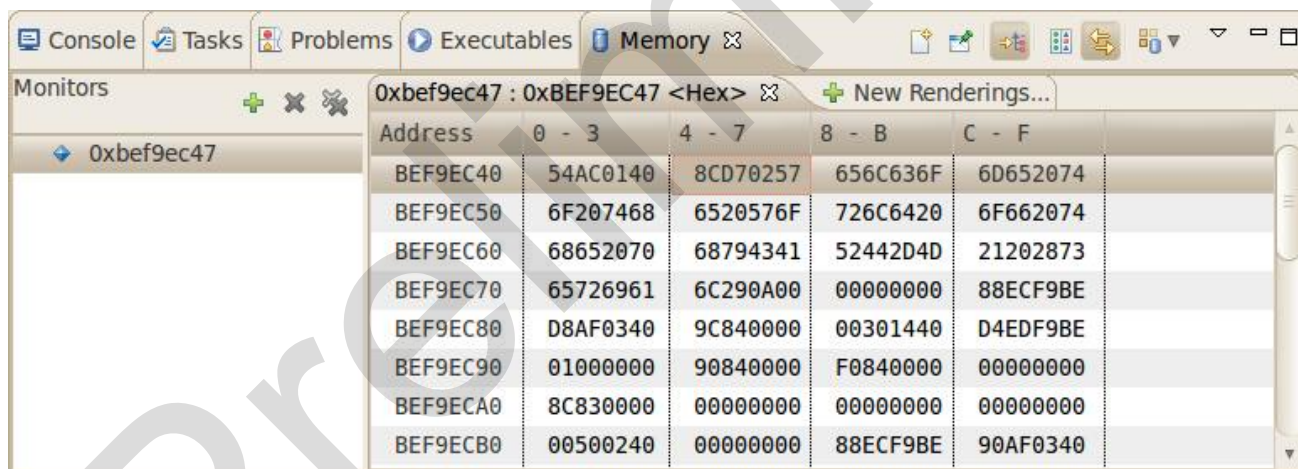
2.2.2.6 Using the Memory Monitor

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

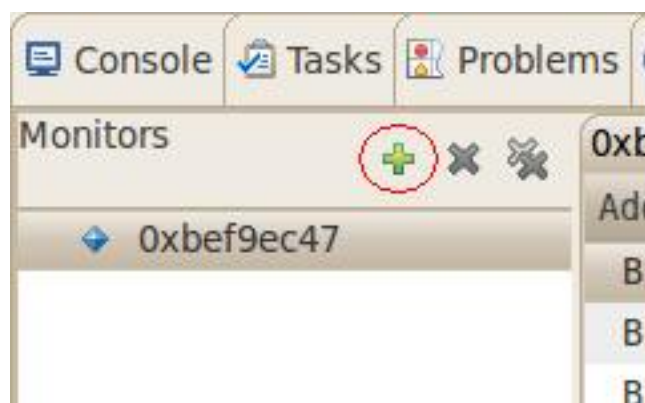
- Select the **Memory** tab in the frame where you can find the *Console*
- Click **+ Add Memory Monitor**
- Enter the address of the buffer and click **OK**. Remember that the variable's address might be different on your system



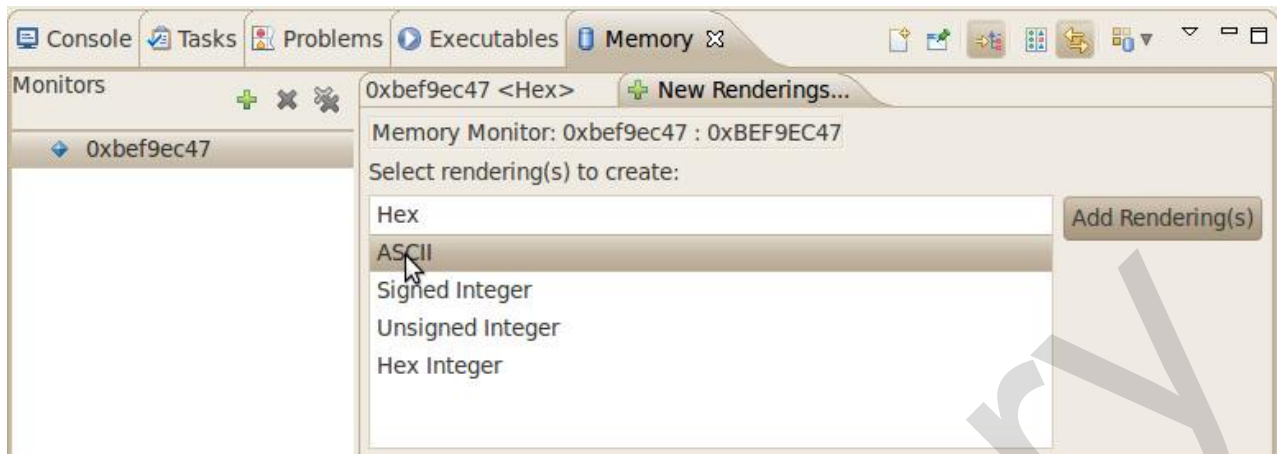
- Change the size of the window



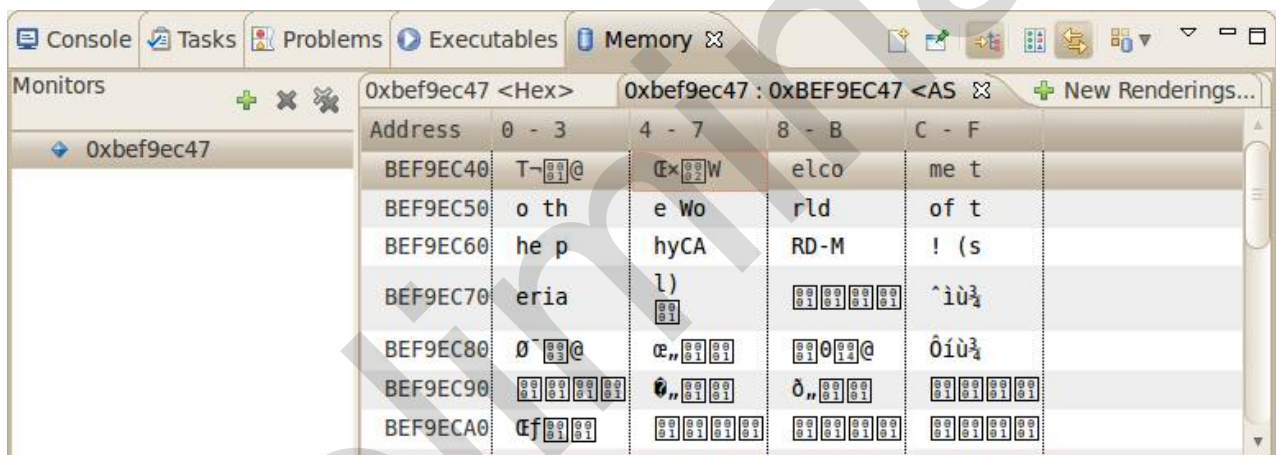
- Click **Add Rendering**



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



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



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



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

```
HelloWorld.c


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

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

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

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



	<p>You have successfully passed the debugging chapter. You are now able to configure and use <i>Eclipse</i> 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.</p>
---	--

3 Accessing the phyBOARD-Mira Features

PHYTEC phyBOARD-Mira is fully equipped with all mechanical and electrical components necessary for the speedy and secure start-up.

3.1 Overview of the phyBOARD-Mira Peripherals


The phyBOARD-Mira is depicted in [Figure 2](#). 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 2](#) highlights the location of each peripheral for easy identification.

3.1.1 Connectors and Pin Header

[Table 2](#) lists all available connectors on the phyBOARD-Mira. [Figure 2](#) highlights the location of each connector for easy identification.

Reference Designator	Description	See Section
X2	Power supply 5 V only (via 6-pole WAGO male header, or 2-pole PHOENIX MINI COMBICON base strip)	3.2.1.1
X3	CAN connector (2×5 pin header 2.54 mm pitch)	3.2.5
X4	Ethernet 0 connector (RJ45 with speed and link LED)	0
X5	USB On-The-Go connector (USB Micro-AB)	3.2.4
X6	USB Host connector (USB 2.0 Standard-A)	3.2.4
X7	PCI Express connector (Mini PCI Express)	3.2.7
X8	Display backlight and control connector (5-pole Molex)	3.2.11
X9	Display LVDS connector (20 pin FCC connector 1 mm pitch)	0
X10	Camera phyCAM-S connector (8-pole Hirose Board-to-Wire Connector 1.25 mm pitch)	0
X13	A/V connector #1 (2×8 dual entry connector 2 mm pitch)	3.2.16
X14	A/V connector #2 (2×20 dual entry connector 2 mm pitch)	3.2.16
X17	Expansion connector (2×30 socket connector 2 mm pitch)	3.2.17
X21	Touch connector (1×4 pin header 2.54 mm pitch)	3.2.12
X22	Secure Digital / Multi Media Card (Micro-slot)	3.2.6
X23	RS-232 with RTS and CTS, or RS-485 (UART3 2×5 pin header 2.54 mm pitch)	3.2.2
X28	HDMI connector (Typ-A)	3.2.9
JP3	Backup voltage connector (1×2 pin header 2.54 mm pitch)	3.2.1.3

Table 2: phyBOARD-Mira Connectors and Pin Headers

	<p>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.</p>
---	---

3.1.2 LEDs

The phyBOARD-Mira is populated with three LEDs to indicate the status of the USB VBUS voltages, as well as of the power supply voltage. The fourth LED is a user programmable RGB-LED.

Figure 2 shows the location of the LEDs. Their function is listed in the table below:

LED	Color	Description	See Section
D2	red	3.3 V voltage generation of the phyBOARD-Mira	3.2.1.2
D6	RGB	User programmable RGB-LED	0
D13	green	Indicates presence of VBUS at the USB Host interface	3.2.4
D14	green	Indicates presence of VBUS at the USB OTG interface	3.2.4

Table 3: phyBOARD-Mira LEDs Descriptions


3.1.3 Switches

The phyBOARD-Mira is populated with two switches, one to reset the phyBOARD-Mira and another to configure the boot sequence.

Figure 2 shows the location of the switches. Their function is listed in the table below:

Switch	Description	See Section
S1	Reset Button	3.2.15
S2	Boot Switch	0

Table 4: phyBOARD-Mira Switches Description

	<p>Detailed descriptions of the assembled connectors, jumpers and switches can be found in the following chapters.</p>
---	--

3.2 Functional Components on the phyBOARD-Mira SBC

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

3.2.1 Power Supply



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

3.2.1.1 Power Connectors (X2)

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

1. a 2-pole PHOENIX MINI COMBICON base strip 3.5 mm connector (X2) suitable for a single 5 V supply voltage, or
2. a 6-pole WAGO male header (X2) to attach the Power Module for phyBOARDs (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-Mira, 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.



Figure 4: Power Supply Connectors(X2)

3.2.1.1.1 PHOENIX 2-pole MINI COMBICON Base Strip (X2)

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

Figure 4 and the following table show the pin assignment.

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

Table 5: Pin Assignment of the 2-pole PHOENIX MINI COMBICON Base Strip at X2

3.2.1.1.2 WAGO 6-pole Male Header (X2)

If a WAGO 6-pole male header is mounted on your board (Figure 2 and Figure 4) your board is prepared to connect to a phyBOARD Power Module (PEB-POW-01), or a custom power supply circuitry. The ordering number of the mating connector from WAGO is: 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, by use of customized Power Modules which match the requirements
- 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 (input voltage of power module)
5	PMOD_PWRGOOD	Power good signal (connected to reset nRESET_IN)
6	nPMOD_PWRFAIL	Power fail signal

Table 6: Pin Assignment of the 6-pole WAGO Connector at X2

A detailed description of the Power Module for phyBOARDS can be found in the Application Guide for phyBOARD Expansion Boards (L-793e).

3.2.1.2 Power LED D2

The red LED D2 next to expansion connector X17 (*Figure 2*) indicates the presence of the 3.3 V supply voltage generated from the 5 V input voltage.

3.2.1.3 VBAT, RTC and JP3

The phyBOARD-Mira features an external RTC at U12 (*Figure 2*) in addition to the RTC of the Power Management IC mounted on the phyCORE-i.MX 6 module. It is used for real-time or time-driven applications. To backup the RTC a Gold cap (C90) (*Figure 2*) is placed on the phyBOARD-Mira. Alternatively the 2-pole pin header JP3 can be used to connect an external battery (3.3 V) to VBAT to feed in the backup voltage.

Figure 4 and the following table show the pin assignment of JP3.


Pin	Signal	Description
1	VBAT	Backup Battery voltage
2	GND	Ground

Table 7: Pinout of the 2-pole pin header JP3

The backup voltage source (either Gold cap at C90, or external battery via JP3) supplies the external RTC at U12 and connects to the backup voltage pin VBAT (A5) of the phyCORE-i.MX 6 supplying some critical registers and the RTC of the Power Management IC when the primary system power, VCC5V, is removed. The backup supply lasts approximately 17 ½ days.

3.2.2 UART Connectivity (X17 and X23)

The i.MX 6 SOM supports up to 5 so called UART units. On the phyBOARD-Mira TTL level signals of UART1 and UART2 (the standard console) are routed to expansion connector X17. UART3 is available at pin header connector X23 at RS-232 level, or optionally at RS-485 level.

	<p>The Evaluation Board (PEB-EVAL-01) delivered with the kit plugs into the expansion connector and allows easy use of the standard console (UART2) which is required for debugging. Please find additional information on the Evaluation Board in Application Guide for phyBOARD Expansion Boards (L-793e).</p>
---	--

Further information on the expansion connector can be found in [section 4.5.7](#).

Pin header connector X23 is located next to the Ethernet connector ([Figure 5](#)) and provides the UART3 signals of the i.MX 6 at RS-232, or RS-485 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 8](#) shows the signal mapping of the RS-232 and RS-485 level signals at connector X23.

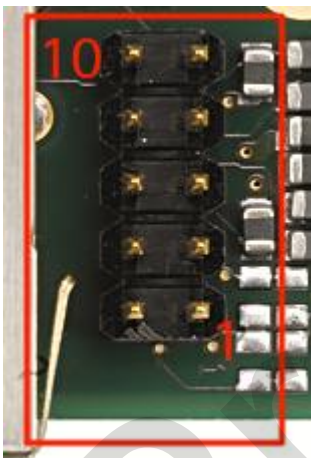


Figure 5: RS-232 or RS-485 Interface Connector X23

Pin	Signal	Pin	Signal
10	NC	9	GND
8	X_UART3_RS485_B	7	X_UART3_RS485_A
6	X_UART3_CTS_RS232	5	X_UART3_TXD_RS232
4	X_UART3_RTS_RS232	3	X_UART3_RXD_RS232
2	NC	1	NC

Table 8: Pin Assignment of RS-232 /RS-485 Interface Connector X23

An adapter cable is included in the phyBOARD-Mira-i.MX 6 Kit to facilitate the use of the UART3 interface. The following figure shows the signal mapping of the adapter.

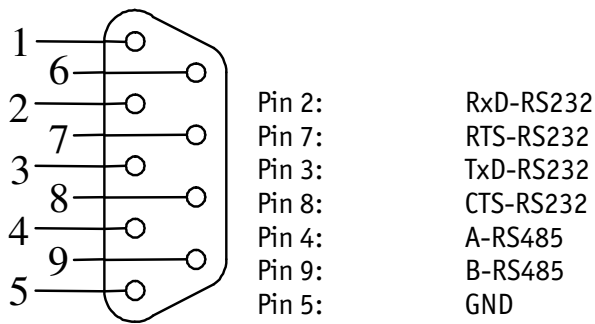


Figure 6: RS-232 / RS-485 Connector Signal Mapping

3.2.2.1 Software Implementation

Only `/dev/ttymx1` for UART2 and `/dev/ttymx2` for UART3 have been implemented within the BSP.

The standard console UART2 is accessible as `/dev/ttymx1` and is mainly used for debugging and control of software updates.

UART1 can be accessed as `/dev/ttymx2`. It is intended for user applications.

3.2.3 Ethernet Connectivity (X4)

The Ethernet 0 interface of the phyBOARD-Mira is accessible at the RJ45 connector X4.



ETH0

Figure 7: Ethernet Interface at Connectors X4

The standard phyBOARD-Mira is equipped with an RJ45 connector supporting a 10/100Base-T network connection. Optionally, it can be ordered with the phyCORE-i.MX 6 module configured for also 1000 Mbps Ethernet and a different RJ45 connector allowing to use the phyBOARD-Mira in a 10/100/1000Base-T network¹. The LEDs for LINK (green) and SPEED (yellow) indication are integrated in the connector. The Ethernet transceiver supports Auto MDI-X, 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.

3.2.3.1 Software Implementation

The i.MX6 SOM features Ethernet, which is being used to provide the network interface eth0 with static IP 192.168.3.11 by default. The interface offers a standard Linux network port at RJ45 connector X4 which can be programmed using the BSD socket interface.

¹: Please contact our sales team for more information.

3.2.4 USB Connectivity (X5 and X6)

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

USB1 is accessible at connector X5 (USB Micro-AB) and 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.

USB2 is accessible at connector X6 (USB Standard-A) and is configured as USB Host.

Both connectors are on the top side of the phyBOARD-Mira and located next to each other ([Figure 8](#)).



USB OTG **USB Host**

Figure 8: Components supporting the USB Interfaces

LED D14 displays the status of X_USB1_VBUS and LED D13 the status of X_USB2_VBUS.

Numerous jumpers allow to configure the USB interfaces according to your needs. Please refer to [section 4.5.1](#) for more information.

3.2.4.1 Software Implementation

3.2.4.1.1 USB Host

The i.MX 6 CPU embeds an 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.

3.2.4.1.2 USB OTG

In order to activate USB OTG support you need to load an appropriate module with *modprobe*, for example the mass storage gadget *g_mass_storage*, which lets the phyBOARD-Mira behave like a regular USB flash drive including a MS-DOS partition table:

```
dd if=/dev/zero of=/tmp/file.img bs=1M count=64
modprobe g_mass_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 partition table (replace *sdC* by the device the computer really shows):

```
sudo fdisk /dev/sdC
```

Use the following sequence of *fdisk* commands to create a new partition table and a single FAT32 partition on the USB flash drive: Press *o* to create an empty partition table, press *n* to create a new partition, press *p* to select the partition type as primary, press *1* to use the first partition on the device, press the enter key to select the first available sector and press the enter key again to use the last available sector as the end of the new partition. Now press *t* to toggle the partition type and press *c* to select the partition type *W95 FAT32*. After that you can write the new partition table to the USB stick by pressing *w*.

To create a FAT32 filesystem execute

```
sudo mkfs.vfat /dev/sdC1
```

You should be able to mount the newly formatted USB flash drive with

```
sudo mount /dev/sdC1 /media
```

The BSP does not provide support for using USB OTG as host.

3.2.5 CAN Connectivity (X3, JP2)

The Flexible Controller Area Network (FLEXCAN) module of the iMX 6 implements the CAN protocol according to the CAN 2.0B protocol specification. The first interface (FLEXCAN1) of the Flexible Controller Area Network is accessible at connector X3 (2×5 pin header, 2.54 mm pitch).

Jumper JP2 can be installed to add a 120 Ohm termination resistor across the CAN data lines if needed.

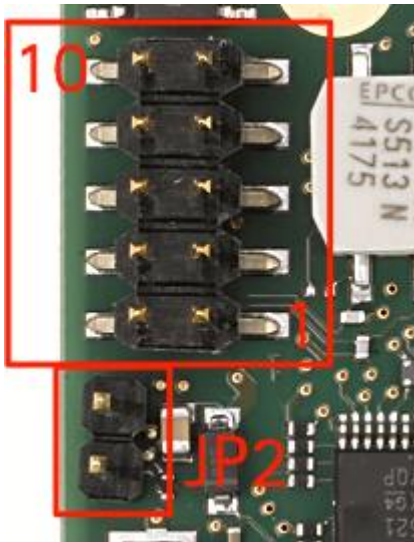


Figure 9: Components supporting the CAN Interface

Table 9 below shows the signal mapping of the CAN1 signals at connector X3.

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

Table 9: Pin Assignment of CAN Connector X3

An adapter cable is included in the phyBOARD-Mira-i.MX 6 Kit to facilitate the use of the CAN interface. The following figure shows the signal mapping of the adapter.

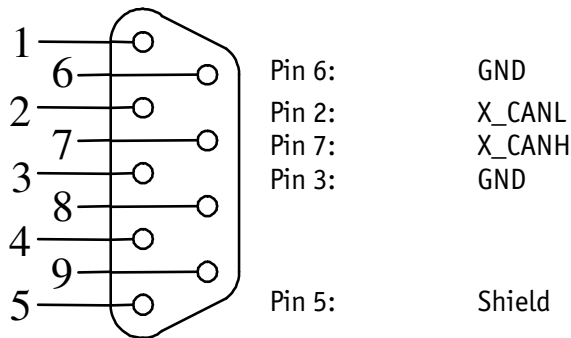


Figure 10: CAN Connector Signal Mapping

Depending on the muxing options and jumper configuration the CAN 1 interface a second CAN interface (FLEXCAN2) is available on expansion port (X17). Please refer to section 4.5.4 for more information.

3.2.5.1 Software Implementation

CAN is supported by drivers using the proposed Linux standard CAN framework "Socket-CAN", which 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
```

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

The 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 file `projectroot/sources/meta-phytec/meta-phyimx6/recipes-core/init-ifupdown/mx6/can-pre-up` instead and rebuild the BSP.

To get the information on *can0* (which represents i.MX 6's CAN1 routed to X65) type `ifconfig can0`

The information will look like the following

```
can0    Link encap:UNSPEC  HWaddr 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:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:10
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
Interrupt:55
```

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 5A1#11.2233.44556677.88
candump can0
```

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

3.2.6 Secure Digital Memory Card/MultiMedia Card (X22)




Figure 11: SD/MM Card Interface at Connector X22 (back side)

The phyBOARD-Mira provides a standard microSDHC card slot at X22 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, which features card detection, a lock mechanism and a smooth extraction function by Push-in/-out of card.

DIP switch S2 allows to toggle between NAND boot and boot from SD card. In order to boot from SD card S2 must be switched ON (refer to [section 3.2.7](#) for further information).

3.2.6.1 Software Implementation


The driver for Micro Secure Digital Cards and Micro Multi Media Cards is accessible as for general purpose block devices. These devices can be used in the same way as any other block devices.

	This kind of devices are hot pluggable, so you must pay attention not to unplug the device while it is 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 file system ("floppy" like handling). In this case <code>/dev/mmcblk0</code> must be used for formatting and mounting.
---	--

The partitions can be formatted with any kind of file system 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.

3.2.7 PCIe Connectivity (X7)

The 1-lane PCI express interface of the phyBOARD-Mira-i.MX 6 provides PCIe Gen. 2.0 functionality which supports 5 Gbit/s operations. Furthermore the interface is fully backwards compatible to the 2.5 Gbit/s Gen. 1.1 specification. The present and the wake signals are realized by GPIOs. The PCIe interface is brought out to the mini PCIe connector X7, pictured in the following figure.



Figure 12: PCIe Interface at Connector X7

The USB Host interface can be jumpered onto PCIe connector X7. Please refer to section 4.5.1 for more information.

The SIM card signals of a connected PCIe module can be available at expansion connector X17. Please refer to Table 27 for more information about the jumper settings.

3.2.7.1 Software Implementation

To use a specific PCIe device you mostly have to enable the corresponding kernel module in the kernel configuration. You can use the shell command `lspci` to show all recognized PCIe devices.

3.2.8 Camera Connectivity (X10)

The phyBOARD-Mira-i.MX 6 has one camera interface at connector X10. The camera interface is compatible with the PHYTEC phyCAM-S+ camera interface standard.



Figure 13: Camera Interface(phyCAM-S+)at Connector X10

The table below shows the pinout of connector X10:

Pin #	Signal Name	Description
1	Camera0_LO+	LVDS Input+
2	Camera0_LO-	LVDS Input-
3	Camera0_RXCLK-	LVDS Clock-
4	I2C_SDA_CAMERA	I ² C Data
5	I2C_SCL_CAMERA	I ² C Clock
6	Camera0_RXCLK +	LVDS Clock+
7	VCC_CAMERA0	Power supply camera (3.3 V)
8	GND	Ground

Table 10: PHYTEC Camera Connector X10

3.2.8.1 Software Implementation



Support of the camera interface is not implemented at the time of completion of this Application Guide. Please find a road map for the BSP features at www.phytec.eu, or contact our support team.

3.2.9 HDMI Connectivity (X28)

The phyBOARD-Mira-i.MX 6 provides a High-Definition Multimedia Interface (HDMI) which is compliant to HDMI 1.4a, DVI 1.0, HDCP 1.4. It supports a maximum pixel clock of up to 340 MHz for up to 720p at 100 Hz and 720i at 200 Hz, or 1080p at 60 Hz and 1080i/720i at

120 Hz HDTV display resolutions, and a graphic display resolution of up to 2048x1536 (QXGA). Audio streams reach a sampling rate of up to 192 kHz. Please refer to the *i.MX 6 Applications Processor Reference Manual* for more information.

The HDMI interface is brought out at a standard HDMI type A connector (X28) on the phyBOARD-Mira-i.MX 6 and comprises the following signal groups: three pairs of data signals, one pair of clock signals, an I²C bus which is exclusively for the HDMI interface, the Consumer Electronics Control (CEC) signal and the hot plug detect (HPD) signal. Level shifters shift the I²C interface signals and the hot plug detect signal from IO voltage (VCC3V3) to 5 V, while the data and clock signals extend directly from the phyCORE-Connector to the HDMI receptacle. The hot plug detect signal is pulled down to ground.


	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.
---	--



Figure 14: HDMI Interface Connector X28

3.2.9.1 Software Implementation

This driver gains access to the display via device node `/dev/fb0` for HDMI. For this BSP the PHYTEC display connector is default.

A simple test of the framebuffer feature can then be run with:
`fb-test`

This will show various pictures on the display.

3.2.10 LVDS Display Connectivity (X9)


Meanwhile there are a few LVDS displays with kind of standardized interfaces on the market. The LVDS display connector X9 is intend to connect these displays with screen diagonals from 7" up to 12.1" at different resolutions.

The display connector X9 is a 20-pole male pin connector with 1 mm pitch.

Bild

To set up the right configuration for a custom display and more detail information about the pin assignment please refer to section 4.5.3.

3.2.10.1 Software Implementation

	Support of the LVDS interface is not implemented at the time of completion of this Application Guide. Please find a road map for the BSP features at www.phytec.eu , or contact our support team.
---	--

3.2.11 Backlight and Control Connector (X8)

In order to support a backlight for the LVDS display X8 provides the supply voltages and control signals necessary.

The following table shows the pinout of the 1.25 mm pitch Molex PanelMate™ header.

Pin #	Signal name	ST	SL	Description
1	NC	-	-	Not connected
2	X_PWM1_OUT	0	3.3 V	PWM brightness output
3	X_LVDS_DISP_EN	0	5.0 V	3.3 V power supply display
4	GND	-	-	Ground
5	VCC_BL	0	n.s.	Backlight power supply

Table 11: Display Power Connector X8 Signal Description

3.2.12 Touch Screen Connectivity (X13, X21)

As many smaller applications need a touch screen as user interface, different provisions are made to connect as well 4- wire resistive touch screens as various capacitive touch screens.

The following table summarizes the different connectors available to connect a touch screen.

Connector	Touch Screen
X21	4-wire resistive touch
X13 (pins 9 – 12)	4-wire resistive touch
X13 (pins 15 – 16)	Capacitive touch screen with I ² C interface

Table 12: Touch Screen Connectivity

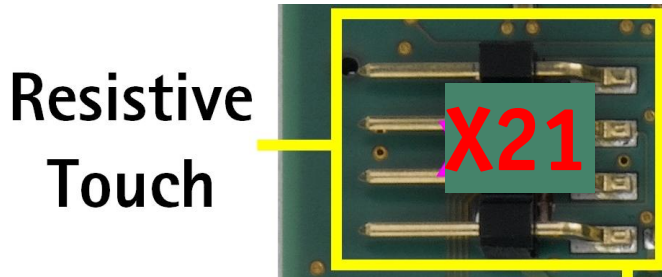



Figure 15: Touch Screen Connectors X21

3.2.12.1 Software Implementation

	<p>Support of the Touch Screen interface is not implemented at the time of completion of this Application Guide. Please find a road map for the BSP features at www.phytec.eu, or contact our support team.</p>
--	--

3.2.13 Multicolor (RGB) LED (D6)

The phyBOARD-Mira provides one multicolor (RGB) LED (D6) (Figure 2) for user applications. The colors can be controlled with the following GPIOs

Color	Signal	Description
Red	X_CSIO_DAT4	GPIO5_IO22 of the i.MX 6
Green	X_CSIO_DAT5	GPIO5_IO23 of the i.MX 6
Blue	X_CSIO_DAT6	GPIO5_IO24 of the i.MX 6

Table 13: Multicolor LED Configuration

As an option the LED can be controlled with the LED dimmer IC at U6². The controller can be accessed via I2C1 at address 0X62 and dynamically controls the LED with PWM signals.

3.2.13.1 Software Implementation

The Multicolor LED can be controlled through the `sysfs` interface. There is a folder for each color in `/sys/class/leds`: `phyboard-mira:blue` for blue, `phyboard-mira:red` for red and `phyboard-mira:green` for green.

²: The phyBOARD-Mira in the standard kit is not equipped with the LED dimmer. Please contact our sales team for more information.

Each color can be configured independently. To switch on the green LED write 255 in the file *brightness* like:

```
echo 255 > /sys/class/leds/phyboard-mira\:blue/brightness
```

To switch it off again, write 0 to the file *brightness*:

```
echo 0 > /sys/class/leds/phyboard-mira\:blue/brightness
```

It is also possible to connect each color to a predefined trigger. To see all available triggers read from the file *trigger* with `cat /sys/class/leds/phyboard-mira\:blue/trigger`. The output maybe

```
[none] rc-feedback nand-disk mmc0 timer oneshot heartbeat backlight gpio
```

For the color blue the trigger *mmc0* is used by default.

3.2.14 Boot Mode (S2)

The phyBOARD-Mira has two defined boot sequence which can be selected with DIP switch S2.

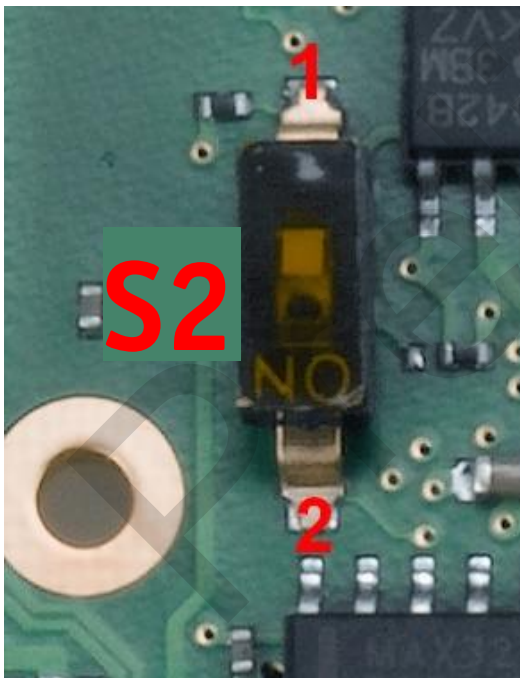


Figure 16: Boot Switch (S2)

Boot Mode	Description
Boot mode 1 (S2 = OFF)	Boot from NAND
Boot mode 2 (S2 = ON)	Boot from SD/MMC 1

Table 14: Boot Switch Configuration (S2)

3.2.15 System Reset Button (S1)

The phyBOARD-Mira is equipped with a system reset button at S1. Pressing this button will toggle the X_nRESET pin (X1D32) of the phyCORE SOM low, causing the module to reset.

3.2.16 Audio/Video connectors (X13 and X14)

The Audio/Video (A/V) connectors X13 and X14 provide an easy way to add typical A/V functions and features to the phyBOARD-Mira. Standard interfaces such as parallel display, I²S and I²C 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 4.5.6](#). Information on the expansion boards available for the A/V Connectors can be found in the Application Guide for phyBOARD-Mira Expansion Boards (L-793e).

3.2.17 Expansion connector (X17)

The expansion connector X17 provides an easy way to add other functions and features to the phyBOARD-Mira. Standard interfaces such as JTAG, UART, MMC, SPI and I²C as well as different supply voltages and some GPIOs are available at the expansion female connector.

For further information of the expansion connector and the pinout see [chapter 4.5.7](#). Information on the expansion boards available for the expansion connector can be found in the Application Guide for phyBOARD-Mira Expansion Boards (L-793e).

3.2.18 Addressing the RTC

The RTC RV-4162-C7 on the phyCORE-i.MX 6 can be accessed as `/dev/rtc0`. It also has the entry `/sys/class/rtc/rtc0` in the sysfs file system, where you can, for example, read 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.

3.2.19 CPU Core Frequency Scaling

The phyBOARD-Mira-i.MX 6 supports dvfs (dynamic voltage and frequency scaling). Several different frequencies are supported. Type

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

to get a complete list. The result will be:

```
996000 792000 396000
```

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

You can decrease the maximum frequency (e.g. to 792000)

```
echo 792000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
```

or increase the minimum frequency (e.g. to 792000)

```
echo 792000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
```

Asking for the current frequency is done with:

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

So called governors are selecting one of these frequencies in accordance to their goals, automatically. The governors available can be listed with the following command:

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

The result will be:

```
userspace powersave ondemand performance
```

userspace allows the user or userspace program running as root to set a specific frequency (e.g. to 792000). Type:

```
echo 792000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

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.

performance always selects the highest possible CPU core frequency.

In order to ask for the current governor, type

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

and you will normally get:

ondemand

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

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

For more detailed information about the governors refer to the *Linux* kernel documentation in:

Documentation/cpu-freq/governors.txt.

3.2.20 Using Qt

Nokia's Qtopia is very commonly used for embedded systems and it is 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.

Some demos that show the capabilities of Qt version 4.8.4 are included in the BSP. Use of an HDMI monitor requires an USB mouse connected to the board in order to test the demos. They are located in */usr/bin/qt4-demos*. 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.

4 System Level Customizing

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
- 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-Mira at a system level

4.2 Software Overview

In [chapter 2](#) you have learned how to work with *Eclipse*. The following section shows you how to work with the *Yocto Project*.

The *Yocto Project* is an open source collaboration project. The build system which the Yocto Project uses is based on the OpenEmbedded Project. With this build system you can construct complete linux images for different platforms and architectures. The Yocto project have components to design, develop, build, debug, simulate and test your software.

4.3 Getting Started with the BSP

In this chapter you will go through some software topics. First of all we take a look into our BSP and learn how to add a package. After compiling the new images, you will learn how to write the newly created kernel and root file system into the target's flash memory and how to start from it.

4.3.1 Working with Yocto

In this part you will learn how to use the Yocto Project and our BSP. You can find our pre-build BSP in our Live System under `/opt/PHYTEC_BSPs/phyBOARD-MIRA/`. All necessary tools and programs are already preinstalled to directly start with Yocto.

Open a new terminal if you have not already done and change to the directory of the pre-installed BSP:



Click the **terminal icon** on your desktop

- Type the following command to change to the BSP-directory:
`cd /opt/PHYTEC_BSPs/phyBOARD-MIRA/`

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

```

Terminal
ubuntu@ubuntu:/opt/PHYTEC_BSPs/phyBOARD-MIRA$ source sources/poky/oe-init-build-env
### Shell environment set up for builds. ###
You can now run 'bitbake <target>'

Tested build targets for the yocto bsp are:
bitbake phytec-hwbringup-image
bitbake phytec-qt4demo-image

PHYTEC

ubuntu@ubuntu:/opt/PHYTEC_BSPs/phyBOARD-MIRA/build$

```

The environment is set and we are now in the build folder. Before we start to modify the BSP we must check if we have selected the right *MACHINE* in the configuration. This BSP supports both Kit versions the PB-01501-001 and the PB-01501-002 by default the configuration is set to the PB-01501-001 variant.

- Open the configuration file with any editor, e.g. VI as with the following command:
`vi conf/local.conf`



In the second row of this file you will see the *MACHINE* variable which is set to *phyboard-mira-imx6-2*. This is the correct *MACHINE* for the lowcost module. If you have the PB-01501-001 module set the *MACHINE* to *phyboard-mira-imx6-1*. In VI you can change to the insert mode by type **I**.

```

Terminal
BSP_VERSION = "PD14.2-rc1"
MACHINE ?= "phyboard-alcor-imx6-1"
DISTRO ?= "poky"
IMAGE_FSTYPES += "sdcard"
IMAGE_FSTYPES += "tar.gz"
IMAGE_FSTYPES += "ubifs"
DEPLOY_DIR = "${TOPDIR}/deploy"
PACKAGE_CLASSES ?= "package_ipk"
#SDKMACHINE ?= "x86_64"
EXTRA_IMAGE_FEATURES = "debug-tweaks tools-debug eclipse-debug"
USER_CLASSES ?= "buildstats image-mklibs image-prelink"
#TEST_IMAGE = "1"
OE_TERMINAL = "auto"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS = "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
"conf/local.conf" 28L, 859C
1,1 Top

```

- Save the changes and close the file with **:x** and **Enter**.
- Now we integrate a new package in our image. Again open the local configuration:
`vi conf/local.conf`
- At the end of the file add following line:
`IMAGE_INSTALL_append = " nano"`


```

Terminal
#TEST_IMAGE = "1"
OE_TERMINAL = "auto"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS = "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K"
INHERIT += "phytec-mirrors"
INHERIT += "rm_work"
#Qemu config
#PACKAGECONFIG_append_pn-qemu-native = " sdl"
#PACKAGECONFIG_append_pn-nativesdk-qemu = " sdl"
#ASSUME_PROVIDED += "libsdl-native"
CONF_VERSION = "1"
IMAGE_INSTALL_append = " nano"
~
29,30 Bot

```

- Save the changes and close the file with **:x** and **Enter**.

In our example we want to add the nano editor to our image. When we start building our images normally the package is first downloaded from the according server, but in our Live System we have also pre-downloaded this package.

	<p>You can search for more available packages at: http://layers.openembedded.org/layerindex/branch/master/recipes/</p> <p>Be sure that the layer on which the package depends is in our BSP. You can check our integrated layers in the file <code>conf/bblayers.conf</code>.</p> <p>When your needed layer is not in our BSP you must first checkout the layer to the folder <code>/opt/PHYTEC_BSPs/phyBOARD-MIRA/source/</code>.</p> <p>But for all new added packages there isn't a guarantee that the build process successfully proceed.</p>
---	--

Now we can start building the configured BSP. There are two methods how to create the images: `bitbake phytec-hwbringup-image` and `bitbake phytec-qt4demo-image`

If you want to use Qt in your own created image you must select the second method. The first option is the faster and smaller method but without all Qt relevant things.

- In our example we build the images by type following command:
`bitbake phytec-qt4demo-image`

You will find the kernel named `zImage`, the device tree named `zImage-imx6q-phytec-phyboard-mira.dtb` and the root file system named `phytec-qt4demo-image-phyboard-mira-imx6-2.ubifs` in the directory `deploy/images/phyboard-mira-imx6-1/`.

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

4.3.2 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-Mira-i.MX 6'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 the 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 will not be able to start your target anymore. In such a case, refer to [section 4.4 "Updating the software"](#).



The versions of the *Barebox* and the Linux kernel must match. Therefore the following steps should only be done using the hardware that was shipped together with the Live System 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-MIRA/build/deploy/images/phyboard-mira-imx6-2/`
- Copy the new images to the `/tftpboot` directory and exit:

```
sudo cp zImage /tftpboot  
sudo cp phytec-qt4demo-image-phyboard-mira-imx6-2.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*.


```

Microcom_ttyUSB0
barebox 2014.11.0-PD14.2-rc2-00008-g57b87ae-dirty #1 Fri Feb 13 17:46:56 CET 201
5
Board: Phytex phyCORE-i.MX6 Duallite/SOLO
detected i.MX6 Solo revision 1.1
mdio_bus: miibus0: probed
nand: ONFI param page 0 valid
nand: ONFI flash detected
nand: NAND device: Manufacturer ID: 0x2c, Chip ID: 0xdc (Micron MT29F4G08ABADAWP
), 512MiB, page size: 2048, OOB size: 64
Bad block table found at page 262080, version 0x01
Bad block table found at page 262016, version 0x01
imx-esdhc 2190000.usdhc: registered as 2190000.usdhc
imx-ipuv3 2400000.ipu: IPUv3H probed
netconsole: registered as netconsole-1
malloc space: 0x17f00000 -> 0x1fdfffff (size 127 MiB)
mmc0: detected SD card version 2.0
mmc0: registered mmc0
barebox-environment environment-sd.11: setting default environment path to /dev/
mmc0.barebox-environment
envfs: wrong magic
no valid environment found on /dev/mmc0.barebox-environment. Using default envir
onment
running /env/bin/init...

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

type exit to get to the menu
barebox@Phytex phyCORE-i.MX6 Duallite/SOLO: /

```

- Check the network settings with following command:

```
ifup eth0
```

```
devinfo eth0
```

The target should present these lines:

```

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

```

If you need to change something, type:

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

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

Now we download the images from the TFTP server to the target's RAM, after that we erase the required flash and write the images from the RAM into the flash.

- Enter `erase /dev/nand0.kernel.bb` to erase the kernel partition.
- Type `cp /mnt/tftp/zImage /dev/nand0.kernel.bb` to download the kernel using TFTP and to write it into the target's flash. The copy process can take up to several minutes.

- Enter `erase /dev/nand0.ofmtree.bb` to erase the device tree partition.
- Type `cp /mnt/tftp/zImage-imx6q-phytec-phyboard-mira.dtb /dev/nand0.ofmtree.bb`
- Now we flash the root filesystem type:
`ubiformat /dev/nand0.root`
`ubiattach /dev/nand0.root`
`ubimkvol /dev/ubi0 root 0`
`cp /mnt/tftp/phytec-qt4demo-image-phyboard-mira-imx6-2.ubifs /dev/ubi0.root`
- Enter `boot` to boot the phyBOARD-Mira-i.MX 6 with the new kernel and root file system.
- After the target has successfully finished booting, type `root` to log in.
- Now we can test the newly installed editor *nano* by trying to open a file with it.
- Enter `nano /etc/profile` .
- Close *nano* by pressing **CTRL+X**.
- Close *Microcom* 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 (*zimage*) and root file system (*phytec-qt4demo-image-phyboard-mira-imx6-*.ubifs*) from the */tftpboot* directory.



All files are also downloadable from our ftp server or you can find them on our Linux-phyBOARD-Mira-i.MX 6-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 it. In case that your phyBOARD-Mira-i.MX 6 does not start anymore because you damaged its software during the previous chapter, you are right here, too.



Make sure that the BSP version matches the hardware version of the phyBOARD-Mira-i.MX6. Please visit <http://www.phytec.eu> and navigate to "Support" / "FAQ/Download" / "phyBOARDS – Single Board Computer" / "phyBOARD-Mira" and see FAQ "Choosing the right Linux-BSP version" for more information.

4.4.1 Creating a bootable SD card

In case that your phyBOARD-Mira-i.MX 6 does not 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 i.MX 6 does not use file systems. Instead it is hard coded at which sectors of the SD card the i.MX 6 expects the bootloader to be.

The image `phytec-*-image-phyboard-mira-imx6*.sdcard` contains the complete correct created partition with all the needed files. So after we detect the right device for our inserted SD card we only need one command to create a bootable SD card.

You can detect the correct device with `sudo fdisk -l` or the last outputs of `dmesg` after inserting the SD card.

Now we can start creating the bootable SD card with following command:

```
sudo dd if=phytec-*-image-phyboard-mira-imx6*.sdcard of=/dev/sdX  
(where X is the correct device letter for the inserted SD card.)
```

4.4.2 Flashing the Bootloader

Set the DIP switch S2 on the base board to ON.

- Check the network settings with following command:

```
ifup eth0  
devinfo eth0
```

The target should present these lines:

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

If you need to change something, type:

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

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

Write the Barebox into flash by typing:

```
barebox_update /mnt/tftp/barebox-phyboard-mira-imx6-*.bin
```



Without any valid Bootloader in NAND, the board will automatically try to use the 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.

4.4.3 Writing the Kernel / Root File System into Flash

Flash the Linux kernel by typing:

```
erase /dev/nand0.kernel.bb  
cp /mnt/tftp/zImage /dev/nand0.kernel.bb
```

Flash the device tree by typing:

```
erase /dev/nand0.oftree.bb  
cp /mnt/tftp/zImage-imx6q-phytec-phyboard-mira.dtb /dev/nand0.oftree.bb
```


Write the root file system into flash by typing:

```
ubiformat /dev/nand0.root  
ubiattach /dev/nand0.root  
ubimkvol /dev/ubi0 root 0  
cp /mnt/tftp/phytec-qt4demo-image-phyboard-mira-imx6-*.ubifs  
/dev/ubi0.root
```

4.5 System Level Hardware Information

4.5.1 USB Connectivity (X5, X6, X7 and X17)

Numerous jumpers allow to configure the USB interfaces according to your needs.

	<p>Due to the small footprint of the jumpers we do not recommend manual jumper modifications. This might also render the warranty invalid. Please contact our sales team if you need one of the USB configurations described below.</p>
---	---

4.5.1.1 Rerouting the USB Interfaces to other Connectors (J5, J6, J9 and J10)

For later expansion boards the USB1 OTG interface can be routed to the expansion connector (X17). Moreover the second USB2 interface can be routed to the mini PCI express connector (X7) The following table shows all possible configurations.

Mode	J5	J6
USB1 at USB-OTG connector X5	1+2	1+2
USB1 at expansion connector X17	2+3	2+3

Table 15: USB1 (USB OTG) Routing Configuration

Mode	J9	J10
USB2 at USB-A connector X6	1+2	1+2
USB2 at mini PCIe connector X7	2+3	2+3

Table 16: USB2 (USB Host) Routing Configuration

4.5.2 HDMI Connector (X28)

Pin #	Signal name	Type	SL	Description
2, 5, 8, 11, 17	GND	-	-	Ground
1	HDMI_TX2+	OUT	HDMI	HDMI data channel 2 negative output
3	HDMI_TX2-	OUT	HDMI	HDMI data channel 2 positive output
4	HDMI_TX1+	OUT	HDMI	HDMI data channel 1 positive output
6	HDMI_TX1-	OUT	HDMI	HDMI data channel 1 negative output
7	HDMI_TX0+	OUT	HDMI	HDMI data channel 0 negative output
9	HDMI_TX0-	OUT	HDMI	HDMI data channel 0 positive output
10	HDMI_TXC+	OUT	HDMI	HDMI clock positive output
12	HDMI_TXC-	OUT	HDMI	HDMI clock positive output
13	HDMI_CONN_CEC	I/O	3.3 V	HDMI consumer electronic control
14	NC	-	-	Not connected
15	HDMI_CONN_DSCL	I/O	5 V	HDMI I ² C clock signal
16	HDMI_CONN_DSDA	I/O	5 V	HDMI I ² C data signal
18	HDMI_5V	OUT	5 V	5 V power supply
19	HDMI_CONN_HPDP	IN	5 V	HDMI hot plug detection
20, 21, 22, 23	Shield	-	-	Shield

Table 17: Pin Assignment HDMI Connector X28

4.5.3 LVDS connector (X9)

In the following table is a complete overview of the LVDS display connector pin assignment.

Pin #	Signal name	Type	SL	Description
1	VCC3V3	OUT	3.3 V	Power supply display ³
2	VCC3V3	OUT	3.3 V	Power supply display
3	X_LVDS_CONFIG1	OUT	3.3 V	Display configuration pin 1
4	X_LVDS_CONFIG2	OUT	3.3 V	Display configuration pin 2
5	X_LVDS0_TX0-	OUT	3.3 V	LVDS 0 data channel 0 negative output
6	X_LVDS0_TX0+	OUT	3.3 V	LVDS 0 data channel 0 positive output
7	GND	-	-	Ground
8	X_LVDS0_TX1-	OUT	3.3 V	LVDS 0 data channel 1 negative output
9	X_LVDS0_TX1+	OUT	3.3 V	LVDS 0 data channel 1 positive output
10	GND	-	-	Ground
11	X_LVDS0_TX2-	OUT	3.3 V	LVDS 0 data channel 2 negative output
12	X_LVDS0_TX2+	OUT	3.3 V	LVDS 0 data channel 2 positive output
13	GND	-	-	Ground
14	X_LVDS0_CLK-	OUT	3.3 V	LVDS 0 clock channel negative output
15	X_LVDS0_CLK+	OUT	3.3 V	LVDS 0 clock channel positive output
16	GND	-	-	Ground
17	X_LVDS_CONFIG3	OUT	3.3 V	Display configuration pin 3 or LVDS 0 data channel 3 negative output
18	X_LVDS_CONFIG4	OUT	3.3 V	Display configuration pin 4 or LVDS 0 data channel 3 positive output
19	X_LVDS_CONFIG5	OUT	3.3 V	Display configuration pin 5 or LVDS 0 data channel 3 negative output
20	X_LVDS_CONFIG6	OUT	3.3 V	Display configuration pin 6 or LVDS 0 data channel 3 positive output

Table 18: Pin Assignment LVDS Display Connector X9

Several jumpers and resistors allow connecting and configuring several displays. The following table gives an overview of possible configurations. Please consider the used display data sheet for the right settings.

³: Provided to supply any logic on the display adapter. Max. draw 100 mA

Pin #	Signal name	J or R	Setting	Description
3	X_LVDS_CONFIG1	R101	If mounted	High level
		R103	If mounted	Low level
4	X_LVDS_CONFIG2	R100	If mounted	High level
		R102	If mounted	Low level
17	X_LVDS_CONFIG3	J17	1+2	High level
			2+3	X_LVDS0_TX3- connected
18	X_LVDS_CONFIG4	J18	1+2	X_LVDS0_TX3+ connected
			n.m.	Not connected
19	X_LVDS_CONFIG5	J19	1+2	High level if R106 is mounted Low level if R108 is mounted
			2+3	X_LVDS0_TX3- connected
20	X_LVDS_CONFIG6	J20	1+2	High level if R105 is mounted Low level if R107 is mounted
			2+3	X_LVDS0_TX3- connected

Table 19: Configuration for LVDS Display Connector X9

4.5.4 CAN Connectivity

For later expansion boards the second CAN interface (FLEXCAN2) is available at the expansion connector (X17). Moreover the first CAN interface (FLEXCAN1) can be routed to the expansion connector (X17). The following table shows the possible configurations for CAN1.

Mode	R110	R111	U2
CAN1 at pin header X3	n.m.	n.m.	m.
CAN1 at expansion connector X17	1+2	1+2	n.m.

Table 20: Configurations for CAN interface

4.5.5 Mini PCI express connector (X7)

In the following table is a complete overview of the mini PCI express connector pin assignment.

Pin #	Signal name	Type	SL	Description
1	X_EIM_EB1/PCIe_nW_DISABLE	IN	3.3 V	PCIe 0_WAKE
2	VCC3V3	OUT	3.3 V	3.3 V power supply
3	X_ECSPi2_SS0/PCIe_COEX1	I/O	3.3 V	Coexistence Pins for wireless solutions
4	GND	-	-	Ground
5	X_CSI1_DATA06/PCIe_COEX2	I/O	3.3 V	Coexistence Pins for wireless solutions
6	VCC1V5	OUT	1.5 V	1.5 V power supply
7	X_ECSPi2_SCLK/PCIe_nCLKREQ	IN	3.3 V	Clock Request Support Reporting and Enabling
8	X_SIM_VCC	IN	-	UIM ⁵ _PWR
9	GND	-	-	Ground
10	X_SIM_IO	I/O	-	UIM_DATA
11	X_PCIe0_CLK-	OUT	DIFF	PCIe 0 clock negative
12	X_SIM_CLK	IN	-	UIM_CLK
13	X_PCIe0_CLK+	OUT	DIFF	PCIe 0 clock positive
14	X_SIM_RST	IN	-	UIM_RESET
15	GND	-	-	Ground
16	X_SIM_VPP	IN	-	UIM_VPP
17	RSVD3	-	-	Not connected
18	GND	-	-	Ground
19	RSVD4	-	-	Not connected
20	X_EIM_EB1/PCIe_nW_DISABLE	OUT	3.3 V	Disable signal
21	GND	-	-	Ground
22	X_ECSPi2_MISO/PCIe_nPERST or X_nRESET if J12 1+2	OUT	3.3 V	Functional card reset by GPIO or X_nRESET
23	X_PCIe_RXN	IN	DIFF	PCIe receive negative
24	VCC3V3	OUT	3.3 V	3.3 V power supply
25	X_PCIe_RXP	IN	DIFF	PCIe receive positive
26	GND	-	-	Ground
27	GND	-	-	Ground

⁵: User Identity Module (UIM) signals

28	VCC1V5	OUT	1.5 V	1.5 V power supply
29	GND	-	-	Ground
30	X_I2C1_SCL	I/O	3.3 V	I ² C 1 Clock
31	X_PCIE_TXN	OUT	DIFF	PCIe transmit negative
32	X_I2C1_SDA	I/O	3.3 V	I ² C 1 Data
33	X_PCIE_TXP	OUT	DIFF	PCIe transmit positive
34	GND	-	-	Ground
35	GND	-	-	Ground
36	X_USB2_DM_PCIE	I/O	DIFF	USB_data minus ⁶
37	GND	-	-	Ground
38	X_USB2_DP_PCIE	I/O	DIFF	USB_data plus ⁶
39	VCC3V3	OUT	3.3 V	3.3 V power supply
40	GND	-	-	Ground
41	VCC3V3	OUT	3.3 V	3.3 V power supply
42	TP6	IN	n.s.	Test point for LED_WWAN
43	GND	-	-	Ground
44	TP7	IN	n.s.	Test point for LED_WLAN
45	RSVD9	-	-	Not connected
46	TP8	IN	n.s.	Test point for LED_WPAN
47	RSVD10	-	-	Not connected
48	VCC1V5	OUT	1.5 V	1.5 V power supply
49	RSVD11	-	-	Not connected
50	GND	-	-	Ground
51	RSVD12	-	-	Not connected
52	VCC3V3	OUT	3.3 V	3.3 V power supply
S1	GNDM1	-	-	Ground
S2	GNDM2	-	-	Ground

Table 21: Mini PCI express Connector X7

⁶: J9 and J10 needs to be set to 2+3. Please refer to section 4.5.1.1 for more information.

4.5.6 Audio/Video connectors (X13 and X14)

The Audio/Video (A/V) connectors X13 and X14 provide an easy way to add typical A/V functions and features to the phyBOARD-Mira. Standard interfaces such as parallel display, I²S and I²C 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 is shown in [Table 22](#) and [Table 23](#).

The A/V connector is intended for use with phyBOARD-Mira-i.MX 6 Expansion Boards⁷, and to add specific audio/video connectivity with custom expansion boards.

Pin #	Signal Name	Type	SL	Description
1	GND	-	-	Ground
2	X_LCD_DATA16	OUT	3.3V	LCD Data 16 – red 0
3	X_LCD_DATA17	OUT	3.3 V	LCD Data 17 – red 1
4	X_LCD_DATA18	OUT	3.3 V	LCD Data 18 – red 2
5	X_LCD_DATA19	OUT	3.3 V	LCD Data 19 – red 3
6	GND	-	-	Ground
7	X_LCD_DATA20	OUT	3.3 V	LCD Data 20 – red 4
8	X_LCD_DATA21	OUT	3.3 V	LCD Data 21 – red 5
9	X_LCD_DATA22	OUT	3.3 V	LCD Data 22 – red 6
10	X_LCD_DATA23	OUT	3.3 V	LCD Data 23 – red 7
11	GND	-	-	Ground
12	X_LCD_DATA8	OUT	3.3 V	LCD Data 8 – green 0
13	X_LCD_DATA9	OUT	3.3 V	LCD Data 9 – green 1
14	X_LCD_DATA10	OUT	3.3 V	LCD Data 10 – green 2
15	X_LCD_DATA11	OUT	3.3 V	LCD Data 11 – green 3
16	GND	-	-	Ground
17	X_LCD_DATA12	OUT	3.3 V	LCD Data 12 – green 4
18	X_LCD_DATA13	OUT	3.3 V	LCD Data 13 – green 5
19	X_LCD_DATA14	OUT	3.3 V	LCD Data 14 – green 6
20	X_LCD_DATA15	OUT	3.3 V	LCD Data 15 – green 7
21	GND	-	-	Ground
22	X_LCD_DATA00	OUT	3.3 V	LCD Data 00 – blue 0
23	X_LCD_DATA01	OUT	3.3 V	LCD Data 01 – blue 1
24	X_LCD_DATA02	OUT	3.3 V	LCD Data 02 – blue 2
25	X_LCD_DATA03	OUT	3.3 V	LCD Data 03 – blue 3

⁷: Please find additional information on phyBOARD-Mira-i.MX 6 Expansion Boards in the corresponding application guide (L-793e).

26	GND	-	-	Ground
27	X_LCD_DATA04	OUT	3.3 V	LCD Data 04 – blue 4
28	X_LCD_DATA05	OUT	3.3 V	LCD Data05 – blue 5
29	X_LCD_DATA06	OUT	3.3 V	LCD Data 06 – blue 6
30	X_LCD_DATA07	OUT	3.3 V	LCD Data 07 – blue 7
31	GND	-	-	Ground
32	X_LCD_CLK	OUT	3.3 V	LCD Pixel Clock
33	X_LCD_ENABLE	OUT	3.3 V	LCD BIAS ENABLE
34	X_LCD_HSYNC	OUT	3.3 V	LCD Horizontal Synchronization
35	X_LCD_VSYNC	OUT	3.3 V	LCD Vertical Synchronization
36	GND	-	-	Ground
37	GND	-	-	Ground
38	X_PWM1_OUT	OUT	3.3 V	Pulse Width Modulation
39	VCC_BL	OUT	n.s.	Backlight power supply ⁸
40	VCC5V	OUT	5.0 V	5.0 V power supply

Table 22: PHYTEC A/V connector X14

⁸: Voltage level is not specified and depends on the connected power module and the attached voltage.

Pin #	Signal Name	Type	SL	Description	
1	X_AUD5_TXC	I/O	3.3 V	I ² S Clock	
2	X_AUD5_TXFS	I/O	3.3 V	I ² S Frame	
3	X_AUD5_RXD	I/O	3.3 V	I ² S Analog-Digital converter (microphone)	
4	X_AUD5_TXD	I/O	3.3 V	I ² S Digital-Analog converter (speaker)	
5	X_CSI1_DATA07/AV_INT	I/O	3.3 V	A/V interrupt; GPIO3_I002	
6	X_CSI1_DATA10/LCD_PWCTRL	I/O	3.3 V	LCD control; GPIO3_I022	
7	GND	-	-	Ground	
8	X_nRESET	OUT	3.3 V	Reset ⁹	(J1 1+2)
	X_LCD_RESET	OUT	3.3 V	Reset LCD	(J1 2+3)
9	TS_X+	IN	1.8 V	Touch X+	
10	TS_X-	IN	1.8 V	Touch X-	
11	TS_Y+	IN	1.8 V	Touch Y+	
12	TS_Y-	IN	1.8 V	Touch Y-	
13	VCC3V3	OUT	3.3 V	3.3 V power supply	
14	GND	-	-	Ground	
15	X_I2C1_SCL	I/O	3.3 V	I ² C 1 Clock	
16	X_I2C1_SDA	I/O	3.3 V	I ² C 1 Data	

Table 23: PHYTEC A/V connector X13

Jumper J1 connects either signal X_LCD_RESET or signal X_nRESET to pin 8 of X13. The following table shows the available configurations:

J1	Description
1+2	X_nRESET
2+3	X_LCD_RESET

Table 24: A/V Jumper Configuration J1

⁹: Reset signal depends on J1, please refer to Table 24 for more information.

4.5.6.1 Software Implementation

4.5.6.1.1 Framebuffer

This driver gains access to the display via device node `/dev/fb0` for PHYTEC display connector. A simple test of the framebuffer feature can then be run with:

```
fbtest
```

This will show various pictures on the display.

You can check your framebuffer resolution with the command:

```
fbset
```



Make sure that the BSP version matches the hardware version of the phyBOARD-Mira **and** the display adapter connected. Please visit <http://www.phytec.eu> and navigate to “Support” / “FAQ/Download” / “phyBOARDS – Single Board Computer” / “phyBOARD-Mira” and see FAQ “Choosing the right Linux-BSP version” for more information.

4.5.6.1.2 Touch

A simple test of this feature can be run with

```
ts_calibrate
```

to calibrate the touch and

```
ts_test
```

to do a simple application using this feature.



Support of the Touch interface is not implemented at the time of completion of this Application Guide. Please find a road map for the BSP features at www.phytec.eu, or contact our support team.

4.5.6.1.3 Audio I²S

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

On the phyBOARD-Mira 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 phyBOARD-Mira-i.MX 6 does not support the audio interfaces. Please visit the PHYTEC website for a road map of the BSP.

4.5.6.1.4 I²C Connectivity

The I²C1 interface of the i.MX 6 is available at different connectors on the phyBOARD-Mira. The following table provides a list of the connectors and pins with I²C connectivity.

Connector	Location
Mini PCIe connector X7	pin 32 (I2C_SDA); pin 30 (I2C_SCL)
phyCAM-S connector X10	pin 4 (I2C_SDA); pin 5 (I2C_SCL)
Expansion connector X17	pin 11 (I2C_SDA); pin 13 (I2C_SCL)
A/V connector X13	pin 16 (I2C_SDA); pin 15 (I2C_SCL)

Table 25: I²C1 Connectivity

To avoid any conflicts when connecting external I²C devices to the phyBOARD-Mira the addresses of the on-board I²C devices must be considered. Table 26 lists the addresses already in use. The table shows only the default address.

Board	Prod. No.	Device	Address used (7 MSB)
I ² C3			
phyCORE-i.MX 6	PCM-058	EEPROM	0x50
		PMIC	0xB0, 0xB1
I ² C1			
phyBOARD-Mira	PBA-C-06	Touch controller	0x44
		LED dimmer	0x62
		RTC	0x68
		mini PCIe	- ¹⁰
		phyCAM-S+	- ¹⁰
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 26: I²C Addresses in Use

¹⁰: Depends on the connected device

4.5.7 Expansion connector (X17)

The expansion connector X17 provides an easy way to add other functions and features to the phyBOARD-Mira. Standard interfaces such as UART, SPI and I²C as well as different supply voltages and some GPIOs are available at the expansion female connector.

The expansion connector is intended for use with phyBOARD-Mira-i.MX 6 Expansion Boards¹¹, and to add specific functions with custom expansion boards

The pinout of the expansion connector is shown in the following table.

Pin #	Signal Name	Type	SL	Description
1	VCC3V3	OUT	3.3 V	3.3 V power supply
2	VCC5V	OUT	5.0 V	5.0 V power supply
3	VCC1V5	OUT	1.5 V	1.5 V power supply
4	GND	-	-	Ground
5	X_ECSPi1_SS0	OUT	3.3 V	SPI 1 chip select 0
6	X_ECSPi1_MOSI	OUT	3.3 V	SPI 1 master output/slave input
7	X_ECSPi1_MISO	IN	3.3 V	SPI 1 master input/slave output
8	X_ECSPi1_CLK	OUT	3.3 V	SPI 1 clock output
9	GND	-	-	Ground
10	X_UART2_RX_DATA	IN	3.3 V	UART 2 receive data (standard debug interface)
11	X_I2C1_SDA	I/O	3.3 V	I ² C 1 Data
12	X_UART2_TX_DATA	OUT	3.3 V	UART 2 transmit data (standard debug interface)
13	X_I2C1_SCL	I/O	3.3 V	I ² C 1 Clock
14	GND	-	-	Ground
15	X_JTAG_TMS	IN	3.3 V	JTAG Chain Test Mode Select signal
16	X_JTAG_TRSTB	IN	3.3 V	JTAG Chain Test Reset
17	X_JTAG_TDI	IN	3.3 V	JTAG Chain Test Data Input
18	X_JTAG_TDO	OUT	3.3 V	JTAG Chain Test Data Output
19	GND	-	-	Ground
20	X_JTAG_TCK	IN	3.3 V	JTAG Chain Test Clock signal
21	X_USB1_DP_EXP	I/O	DIFF	USB 1 data plus ¹²
22	X_USB1_DM_EXP	I/O	DIFF	USB 1 data minus ¹²
23	X_nRESET	OUT	3.3 V	Reset

¹¹: Please find additional information on phyBOARD-Mira-i.MX 6 Expansion Boards in the corresponding application guide (L-793e).

¹²: Jumpers J5-J6 allow to route the USB interface to the expansion connector ([Table 15](#)).

24	GND	-	-	Ground	
25	X_SD3_CMD	I/O	3.3 V	SD/MMC command	
26	X_SD3_DATA0	I/O	3.3 V	SD/MMC data 0	
27	X_SD3_CLK	I/O	3.3 V	SD/MMC clock	
28	X_SD3_DATA1	I/O	3.3 V	SD/MMC data 1	
29	GND	-	-	Ground	
30	X_SD3_DATA2	I/O	3.3 V	SD/MMC data 2	
31	X_CSIO_DAT11/ECSPI2_SS0	I/O	3.3 V	SPI 2 chip select 0; UART1RX; GPIO5_IO29	
32	X_SD3_DATA3	I/O	3.3 V	SD/MMC data 3	
33	X_CSIO_DAT10/ECSPI2_MISO	I/O	3.3 V	SPI 2 MISO; UART1TX; GPIO5_IO28	
34	GND	-	-	Ground	
35	X_SD3_DATA4	I/O	3.3 V	SD/MMC data 4; UART2RX	
36	X_SD3_DATA5	I/O	3.3 V	SD/MMC data 5; UART2TX	
37	X_SATA_TXP	OUT	DIFF	SATA transmit positive	
38	X_SD3_DATA6	I/O	3.3 V	SD/MMC data 6	
39	X_SATA_TXN	OUT	DIFF	SATA transmit negative	
40	X_SD3_DATA7	I/O	3.3 V	SD/MMC data 7	
41	GND	-	-	Ground	
42	X_ECSPi2_RDY/nPMON_PWRFAIL	OUT	3.3 V	Power fail signal	(J26 1+2)
	X_SIM_VPP	-	-	UIM_VPP	(J26 2+3)
43	X_SATA_RXP	IN	DIFF	SATA receive positive	
44	X_CSIO_DAT8/ECSPi2_SCLK	OUT	3.3 V	SPI 2 clock output	(J21 1+2)
	X_SIM_VCC	OUT	-	UIM VCC	(J21 2+3)
45	X_SATA_RXN	IN	DIFF	SATA receive negative	
46	GND	-	-	Ground	
47	X_FLEXCAN1_TX_EXP	OUT	3.3 V	CAN 1 transmit data ¹³	
48	X_FLEXCAN1_RX_EXP	IN	3.3 V	CAN 1 receive data ¹³	
49	X_USB_OTG_OC/FLEXCAN2_TX	OUT	3.3 V	CAN 2 transmit data	
50	X_USB_OTG_PWR/FLEXCAN2_RX	IN	3.3 V	CAN 2 receive data	
51	GND	-	-	Ground	
52	X_CSIO_DAT9/ECSPi2_MOSI	OUT	3.3 V	SPI 2 MOSI	(J25 1+2)
	X_SIM_RST	OUT	-	UIM Reset	(J25 2+3)
53	X_USB1_ID	IN	3.3 V	USB 1 identification	
54	X_USB1_VBUS	OUT	5.0 V	USB 1 bus voltage	
55	X_USB_OTG_CHD_B	OUT	3.3 V	USB 1 charger enable	(J23 1+2)

¹³: Please refer to section 4.5.4 to make CAN 1 available on expansion connector

	X_SIM_IO	OUT	-	UIM Data	(J23 2+3)
56	GND	-	-	Ground	
57	VCC_BL	OUT	n.s.	Backlight power supply ¹⁴	
58	X_ECSPi2_SS1	OUT	3.3 V	SPI 2 chip select 1	(J22 1+2)
	X_SIM_CLK	OUT	-	UIM Clock	(J22 2+3)
59	GND	-	-	Ground	
60	VCC5V_IN	IN	5.0 V	5 V input supply voltage	

Table 27: PHYTEC Expansion Connector X17

4.5.7.1 Software Implementation

4.5.7.1.1 UART Connectivity

Only `/dev/tty00` for UART0 and `/dev/tty01` for UART1 (at X66) have been implemented within the BSP.

The standard console UART0 is accessible as `/dev/tty00` and is mainly used for debugging and control of software updates.

Usage of `/dev/tty02` for UART2 requires additional software development.

4.5.7.1.2 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 not be useful. 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.

If you really want to access SPI from userland, you can use `SPIDEV` in the linux kernel. See the kernel documentation in *Documentation/spi/spidev*.

4.5.7.1.3 I²C Connectivity

The driver for I²C can be accessed by its API within the kernel. If you connect a chip to I²C, you should implement its driver into the kernel, as well. The I2C driver offers no `/dev/i2c` entry in userspace, because using it would mean to place the driver for your chip in userspace, too, what most probably would not be useful. 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.

¹⁴: Voltage level is not specified and depends on the connected power module and the attached voltage.

It is also possible to access the I²C Bus from userland in a rudimentary way. You may use the tools `i2cdetect`, `i2cget`, `i2cset` and `i2cdump` for some quick experiments. To program the character devices `/dev/i2c-*` directly see the kernel documentation in *Documentation/i2c/dev-interface*.

4.5.7.1.4 User programmable GPIOs

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

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
22.02.2015	Manual L-806e_0	Preliminary edition. Describes the phyBOARD-Mira-i.MX 6 SOM (PCB 1429.0) with phyBOARD-Mira-Carrier Board (PCB 1434.0).

Preliminary

Index

A

A/V Connector 59

B

Backlight 56

Block Diagram..... 2

Boot Modes 58

C

CAN 49

Connectors..... 39

D

D657

E

EMC xi

Ethernet 46

Expansion Connector..... 59

H

HDMI 54

J

J10 71

J5 71

J6 71

J9 71

JP2 49

JP3 43

L

LEDs 40

D13 47

D14 47

D243

RGB..... 57

LINK LED 46

LVDS

Display 56

P

Peripherals..... 39

Pin Header 39

Power Connector

PHOENIX 2-pole X2.....42

WAGO 6-pole X2.....42

Power Connectors41

R

Reset59

RGB LED.....57

RS-23244

RS-48544

RTC43

S

S159

S258

SD card.....52

SPEED LED46

Switches40

T

Touch.....56

U

USB 47, 71

USB 2.047

V

VBAT43

X

X10.....54

X13.....59

X14.....59

X17..... 44, 59, 71

X241

X22.....52

X23.....44

X28.....54

X349

X446

X5 47, 71

X6 47, 71

X7 53, 71

X856

X956

Preliminary

Document: phyBOARD-Mira-i.MX 6
Document number: L-806e_0, February 2015

How would you improve this manual?

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

Submitted by:

Customer number: _____

Name: _____

Company: _____

Address: _____

Return to:

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

Preliminary

Published by

PHYTEC

© PHYTEC Messtechnik GmbH 2015

Ordering No. L-806e_0

Printed in Germany