

# QuickStart Instructions Linux-Kit phyCORE-i.MX31

Using Eclipse and the GNU Cross Development Tool Chain

Note: The PHYTEC Linux-i.MX31-Disc includes the electronic version of the English phyCORE-i.MX31 Hardware Manual.

**Edition: August 2009** 

A product of a PHYTEC Technology Holding company

In this manual copyrighted products are not explicitly indicated. The absence of the trademark ( $^{TM}$ ) 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 believed 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 2009 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 be made without the explicit written consent from PHYTEC Messtechnik GmbH.

	Europe	NORTH AMERICA
Address: PHYTEC Technologie Holding AG P		PHYTEC America LLC
	Robert-Koch-Str. 39	203 Parfitt Way SW, Suite G100
	55129 Mainz	Bainbridge Island, WA 98110
	GERMANY	USA
Ordering	+49 (800) 0749832	1 (800) 278-9913
Information:	order@phytec.de	sales@phytec.com
Technical	+49 (6131) 9221-31	1 (800) 278-9913
Support:	support@phytec.de	support@phytec.com
Fax:	+49 (6131) 9221-33	1 (206) 780-9135
Web Site:	http://www.phytec.de	http://www.phytec.com

4th Edition: August 2009

1	Introduction1	
1.1	Rapid Development Kit Documentation 1	- •
1.2	Professional Support Packages Available 1	5 min
1.3	Overview of this QuickStart Instruction	
1.4	Conventions Used in this QuickStart	
1.5	System Requirements	
1.6	Software Development Toolchains	
	Eclipse	
	The GNU Cross Development Toolchain	
2	Getting Started7	2
2.1	Requirements of the Host Platform	
2.2	Configuring the Host Platform	35 min
	Installing Software Packages	
	Set Up Network Card Configuration	
	Disabling the Firewall	
	Set Up TFTP Server	
2.3	Linux-i.MX31-Kit Setup	
	Starting the Setup	
2.4	Advanced Configuration Information	
2.5	Connecting the Host with the Target	
2.6	Copying an Example to the Target	
	Copying a Program to the Target	
	Using Telnet to Execute a Program on the Target	
	Using SSH to Execute a Program on the Target	
2.7	Advanced Information	
	Copying a Program to the Target with the Command Line 39	
	Executing a Program on the Target	
	Executing a Program directly on the Target using SSH	

3	Getting More Involved	41
3.1	Configuring and Compiling the Kernel	41
	Writing the Kernel into the Target's Flash	44
3.2	Opening an Existing Project	49
	Copying the HelloWorld Project	49
	Starting Eclipse and Importing the Example Project	49
3.3	Creating a New Project	54
3.4	Changing the Demo Application	64
	Executing the Program on the Target using Microcom	65
3.5	Starting a Program out of Eclipse on the Target	66
3.6	Automatically Starting the Program when Booting the Ta	rget.69
4	Debugging an Example Project	74
4.1	Starting the GDB Server on the Target	74
4.2	Configuring and Starting the Debugger in Eclipse	76
4.3	Setting a Breakpoint	80
4.4	Stepping and Watching Variable Contents	81
4.5	Changing Variable Values	84
4.6	Using the Memory Monitor	85
5	Further Information	88
6	Summary	89
7	Installing Linux on the phyCORE-i.MX31	90
7.1	Installing the Boot Loader	90
7.2	Configure U-Boot Environment Variables	96
7.3	Restoring the U-Boot Default Configuration	99
7.4	Writing the Kernel / Root File System into Flash	99



70 min



20 min

# **1** Introduction



5 min

In this QuickStart you can find general information on the PHYTEC phyCORE-i.MX31, an overview of the Eclipse software development tool and the GCC C/C++ cross development tool chain. You will also find instructions on how to run example programs on the phyCORE-i.MX31, mounted on the PHYTEC Development Board, in conjunction with the Eclipse development tool.

Please refer to the phyCORE-i.MX31 Hardware Manual for specific information on board-level features, such as jumper configuration, memory mapping, and pin layout.

## **1.1 Rapid Development Kit Documentation**

This "Rapid Development Kit" (RDK) includes the following electronic documentation on the enclosed "PHYTEC Linux-i.MX31-Disc":

- the PHYTEC phyCORE-i.MX31 Hardware Manual
- i.MX31 Controller User's Manuals and Data Sheets
- this QuickStart Instructions with general "Rapid Development Kit" description, software installation advice, and an example program enabling quick out-of-the-box start-up of the phyCORE-i.MX31 in conjunction with the Eclipse and GCC C/C++ software development tool chain

## **1.2 Professional Support Packages Available**

This kit comes with free installation support. If you have any questions concerning installation and setup, you are welcome to contact our support department.

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

Please contact our sales team to discuss the appropriate support option if professional support beyond installation and setup is important to you.

## **1.3** Overview of this QuickStart Instruction

This QuickStart Instruction gives you a general "Rapid Development Kit" description, as well as software installation advice and an example program enabling quick out-of-the-box start-up of the phyCORE-i.MX31 in conjunction with the Eclipse IDE and GCC C/C++ software tools. It is structured as follows:

- 1) The "*Getting Started*" chapter describes the configuration of the host platform and how to setup all the tools used in this manual.
- 2) The "*Getting More Involved*" chapter provides step-by-step instructions on how to configure and build a new kernel, modify an example application, create and build new projects, and copy programs to the phyCORE-i.MX31 using Eclipse.
- 3) The *"Debugging"* chapter provides information on how to debug an application with the Eclipse debugging interface.

In addition to the dedicated data for this Rapid Development Kit, the PHYTEC Linux-i.MX31-Disc contains supplemental information on embedded microcontroller design and development in general.

## 1.4 Conventions Used in this QuickStart

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

- *Italic* Used for file and directory names, program and command names, command-line options, menu items, URLs, and other terms that correspond the terms on your desktop.
- **Bold** Used in examples to show commands or other text that should be typed literally by the user.

Pay attention to notes set apart from the text with the following icons:



At this part you might leave the path of this QuickStart.



This is a warning. It helps you to avoid annoying problems.



You can find useful supplementary information about the topic.



At the beginning of each chapter you can find information of the time to pass that chapter.



You have successfully passed an important part of this QuickStart.



You can find information to solve common problems.

## **1.5 System Requirements**

Use of this "Rapid Development Kit" requires:

- the PHYTEC phyCORE-i.MX31
- the phyCORE Development Board with the included DB-9 serial cable, Ethernet cross-over cable and AC adapter supplying 5 VDC /min. 2 A
- PHYTEC Linux distribution based on OSELAS from Pengutronix
- an IBM-compatible host PC (586 or higher)
- openSUSE 11.0 (x86) and the KDE desktop
- recommended free disk space: at least 2 GB

For more information and updates, please refer to the following sources:



support@phytec.de

# **1.6 Software Development Toolchains**

#### Eclipse

The Eclipse platform provides support for C/C++ development. Because the Eclipse platform is only a framework for developer tools, it doesn't support C/C++ directly; instead it uses external plug-ins. This QuickStart shows 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.

#### 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 doesn't have a native set of compilation tools, or when the host system is faster and has greater resources.

The platform where the actual development takes place is called the *host* platform. The platform where the final application is tested and run is

called the *target platform*. In this QuickStart we are using an x86-based Linux as the host platform. As the target platform we are using the ARM architecture with an i.MX31 CPU.

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 QuickStart we are using the GNU C/C++ Cross Development Toolchain.

# 2 Getting Started



In this chapter you will establish the basis to pass the steps in this QuickStart. First you will learn how to configure the host platform. You will install additional software packages and setup the network configuration for connecting your host to the target. After connecting the host to the target, you will copy an application to the target. At the end of this chapter you will be able to start a first demo application on the target.

### 2.1 Requirements of the Host Platform

To pass the following steps in this QuickStart, you will need a host PC with an installation of openSUSE 11.0 (x86) and the KDE desktop.

When you are installing openSUSE 11.0, you can select *KDE* as *Desktop selection*. The default packages to use openSUSE 11.0 with your host PC will be selected automatically. This default selection will suffice to pass the steps in this QuickStart Instructions. The installation of additional packages and configurations will be described on the following pages.

In the following configuration steps we assume that the host PC is not connected to any other network. The target and host will be connected with a cross-over cable via a peer-to-peer connection. If your host is part of a company's network, we recommend disconnecting your host from such a network.

In these QuickStart Instructions you will have to shutdown the firewall and configure the network card of your host PC. If your host PC is connected to another network, changing the IP address can cause conflicts with existing hosts.

# 2.2 Configuring the Host Platform

In this passage you will learn how to configure the host platform. You will execute the following steps:

- Installing additional software packages. These packages are necessary to accomplish the steps in the QuickStart Instructions.
- Setting up the network configuration to use the host PC with your target.
- Disabling the firewall. If the firewall is enabled, you will have problems connecting to the target.
- Setting up a TFTP server. You can use a TFTP server to download files (e.g. kernel and root file system images) to the target from within the target's boot loader.

#### **Installing Software Packages**

To accomplish the steps in the QuickStart Instructions you will have to install additional packages.



If you don't install all of these packages, the setup may fail or some configuration steps won't work correctly.

Search:				- 0	11
				Applications	-
- 🙆	Administrator Settings YaST				
<u>(</u>	System Information sysinfo:/				
				System Folders	
<b></b>	Home Folder /home/developer				
Ø	My Documents /home/developer/Documents				
je. Ne	Network Folders remote:/				
				Media	
3	6.0G Media (1.9 GB available) /				
Ø	Floppy Drive /dev/fd0				•
$\stackrel{\frown}{\frown}$					
<u>F</u> avorites	History Computer	Applications	<u>L</u> eave		
User devel	oper on linux-36py			openSUSE	
6	/ 🌛 🏠 😓 ¹	2			

- Open the *K menu* from the lower-left corner of the desktop and click on the tab *Computer*.
- Open the Administrator Settings / YaST.

🗙 Run as ro	ot - KDE su 🥘 ? 💶 🗙
R	Please enter the Administrator (root) password to continue.
Command:	/sbin/yast2
<u>P</u> assword:	
	Ignore 🖌 <u>O</u> K 🔀 <u>C</u> ancel

• Enter your root password and click OK.



• Open Software Management in Software.

📮 YaST2@ws-ptx 🎱		_ <b>D</b> ×
<u>F</u> ile <u>P</u> ackage <u>E</u> xtras <u>H</u> elp		
Filter: Search •	Package Summary Files and Libraries mandatory for De	velopment.
libxml2-devel		
Search in Name Summary Description Provides Reguires		••
Search Mode:	Description Iechnical Data Dependencies Versions	]
Contains Case Sensitive Name Disk Usage Used Free Total / 16% 1.59 GB 7.92 GB 9.51 GB	Iibxml2-devel - Include Files and Libraries mandatory for Development.           This package contains all necessary include files and libraries to develop applications that require these.	needed
	Check Dependencies Autocheck Cancel	Accept

- Select Filter Search.
- Type **libxml2-devel** and click the *Search* button.
- Check *libxml2-devel*.

🖡 YaST2@ws-ptx 🎱					- 🗆 🗙
<u>F</u> ile <u>P</u> ackage <u>E</u> xtras <u>H</u> elp					
Filter: Search	Package	Summary evel Include Files a	nd Libraries Mand	atory for Buil	ding Python
Searc <u>h</u> :					
python-devel	1				
Search in  Search in  Summary  Description  Provides  Reguires		1005			()
Search Mode:	Description	<u>T</u> echnical Data	Dependencies	Versions	ĵ .
Contains Case Sensitive Name Disk Usage Used Free Total / 16% 1.60 GB 7.90 GB 9.51 GB	python-de Python Mode The Python p with dynamic programs.	vel - Include Files ules. rogramming langu ally loaded extens	and Libraries Mar age's interpreter c ions and can be e	idatory for Bi an be extend mbedded in	iilding ded other
	Check <u>D</u> epe	ndencies 🗌 🗆 Au	tocheck	<u>C</u> ancel	Accept

- Type **python-devel** and click the *Search* button.
- Check *python-devel*.

👼 YaST2 🎱							×
<u>F</u> ile <u>P</u> ackage <u>E</u> xtras <u>H</u> elp							
Fi <u>l</u> ter: Search		Package	Summary		Size	Avail. Ver	
		tftp	Trivial File T	ransfer Protocol (TFTP	) 61.6 K	0.43-10	
Searc <u>h</u> :		yast2-tttp-server	YaSI2-TET	P Server Configuration	1 112.1 K	2.14.0-8	
tttp 💌							
Search							
Search in							
🕱 <u>N</u> ame							
🗙 Su <u>m</u> mary							
Description							
RPM "Provides"							
RPM "Reguires"							
Search <u>M</u> ode:	•					1	Þ
Contains	_				and the second		m
	De	scription Tecl	nnical Data	Dependencies \	ersions	File Li	Ľ
Case Sensiti <u>v</u> e	y	ast2-tftp-server	- YaST2 - TF1	rP Server Configuratio	ı		
	Th	e YaST2 compor	ent for config	juring a TFTP server. T	FTP stand	ls for Trivia	Ľ.
Name Disk Usage Used Free Total	[[E	e transier Frotot	or. It is used	for booking over the ne	WORK.		
/61% 3.4 GB 2.1 GB 5.5 GB							
	Cł	nec <u>k</u> 🗌 A <u>u</u> to	check		<u>C</u> ancel	Accep	t

- Type **tftp** and click the *Search* button.
- Check the packages *tftp* and *yast2-tftp-server*.

🖶 YaST2 🧐					
Eile Package Extras Help Fijter: Search (d3-devel Search Search Search Search Search Reme Summary Description RPM "Provides" RPM "Reguires"	Package dbus-1-q qt3-devel qt3-devel	Summary 13-devel Developer Include Fil doc Q13 Devel tools User Interf	package for QT3/H es and Libraries m opment Kit ace Builder and ot	DE bindings ( landatory for D her tools (desi	or D-Bus evelopment. gner, assistant, lir
Search Mode:	•				41
Contains 💌		+++++++++++++++++++++++++++++++++++++++	** * * * * * * * * * * * * * * * * *	+++++++++++++++++++++++++++++++++++++++	44. 44.
Case Sensitive	Description	Technical Data	Dependencies	Versions	File List
Name Disk Usage Used Free Total /	qt3-devel -       You need thi       the "Qt Cross       include files,       started with 0       You need a I       Check	Include Files and Li s package if you wa splatform Developm documentation, pre 2t. icense for using Qt	braries mandaton nt to compile prog lent Kit 2". Under /r ecompiled exampi with a non-GPL ap	y for Developm rams with Qt 3 usr/lib/qt3 you v es, and a tutori plication. A lice <u>C</u> an	ent. It contains will find al for getting ense can be cel <u>Accept</u>

- Type **qt3-devel** and click the *Search* button.
- Check *qt3-devel*.



- Select Filter Patterns.
- Select *Basis Development*, C/C++ Development, and Tcl/Tk Development.

• Click Accept.

	Automatic Cha In addition to you	es Inges ur manual selections, the following packages have been ch	anged to	
_	Package	Summary	Size	A
	atk-devel	Include Files and Libraries mandatory for Development.	134.9 kB	1.1
	cairo-devel	Development environment for cairo	513.8 kB	1.0
	fontconfig-devel	Include Files and Libraries mandatory for Development.	818.8 kB	2.
	freeglut-devel	Development Package for freeglut (GLUT Library)	423.1 kB	2.
	freetype2-devel	Include Files and Libraries mandatory for Development.	1.8 MB	2.
	glib2-devel	Include Files and Libraries mandatory for Development.	1.5 MB	2.1
	glitz-devel	glitz - OpenGL image compositing library	211.4 kB	0.
	gtk2-devel	Include Files and Libraries mandatory for Development.	1.7 MB	2.1
	libart Igpl-devel	Include Files and Libraries mandatory for Development.	196.3 kB	2.
	libgcj-devel	Include Files and Libraries mandatory for Development.	27.7 MB	4.
	liblcms-devel	Include Files and Libraries Mandatory for Development	474.6 kB	1.1
	libmng-devel	Include Files and Libraries mandatory for Development.	1.2 MB	1.(
	libpixman	Pixel manipulation library	152.2 kB	0.:
	libpixman-devel	Development environment for libpixman	126.5 kB	0.;
	pango-devel	Include Files and Libraries mandatory for Development.	260.1 kB	1.1
1				Þ
		Continue Cancel		

Some additional packages will be selected automatically to resolve dependencies.



If problems occur while resolving dependencies, we recommend going back to a default configuration.

• Click *Continue* to install the packages.

🙀 YaST2 <2>		_ 🗆 🗙
SuSEconfig must write the configuration of your SUSE LINUX system. Depending on the CPU and the amount of memory, this process can take some time.	Writing the system configure Running SuSEconfig.permissions Running SuSEconfig.scpm Install more packages? Install More Einish Finishing SuSEconfig Done. Progress 100%	ration

• Click *Finish*.

#### Set Up Network Card Configuration



In the following steps you will have to configure the IP address of your host. We recommend disconnecting your host CAUTION from any other network. If you change the host's IP, chances are that problems may occur with other hosts in the network.

• Open the *YaST Control Center* if it is not already opened.



• Choose Network Settings in Network Devices.

🚯 YaST2 🍥						
風 Network Se	ttings					
Global Options	Overview	Hostname/DNS	Routin	g		
Name	IP Add	iress				
79c970 [PCnet32 L	ANCE] Not co	onfigured				
79c970 [PCnet32	LANCE]				-	
MAC: 00:0c:29:1c:	09:24				1111	
The device is not c	onfigured. Pre	ss <b>Edit</b> to configure	e.			
<u>A</u> dd Ed		e				-
Help		A	Abo <u>r</u> t	Back	<u>F</u> inish	

- Select the right network card (if more than one network card is installed on your host).
- Click *Edit* to enter the *Network Card Setup*.
- Choose Address

🕵 YaST2 🍥					
Network Card Setup					
General	Address	Hardware			
De <u>v</u> ice Type	De <u>v</u> ice Type C <u>o</u> nfiguration Name				
Ethernet	▼ eth1				
○ No I <u>P</u> Address (for E	Bonding Devices)				
○ D <u>v</u> namic Address [	DHCP 👻				
Statically assigned	P Address				
IP Address	<u>S</u> ubnet Mask	<u>H</u> ostname			
192.168.3.10 255.255.255.0					
Additional Addresses					
Alias Name 🗸 IP Address Netmask					
Ad <u>d</u> Edit Delete					
Help		Cancel Back	<u>N</u> ext		

• Enter IP address **192.168.3.10** and subnet mask **255.255.255.0** 

#### **Disabling the Firewall**

To ensure that there are no problems with connections to the target, the host's firewall should be disabled.

• Select the *General* tab in the upper-left corner.

Device Activation	Address	Hardware	
Activate <u>d</u> evice			
At Boot Time	•		
Assign Interface to Internal Zone (Unr	Firewall <u>Z</u> one	<b></b>	
Device Control —			
<u>Enable Device C</u>	ontrol for Non-root U	ser Via KInternet	
Maximum Transfer	Unit (MTU)		
Set <u>M</u> TU			

- Use the drop-down box in the *Firewall Zone* settings to set the current interface to *Internal Zone (Unprotected)*.
- Then press *Next*, and in the following window click *Finish* to complete the settings.

The firewall is now disabled for this network card.

#### Set Up TFTP Server

Later in this QuickStart you will learn how to write a new kernel image into the flash memory of the target. To download the kernel image from the target, you need have to have a TFTP server running. In this passage we show you how to configure a TFTP server.

• Open the YaST Control Center if it is not already opened.

YaST Control Center @ linux-j10	Ov 🧿		
	LUAF DIOWSEI		
Software	Mail Transfer Agent	NFS Client	
System			
Network Devices	NIS Client		
- 🕵 Network Services	Network Services (xinetd)	Proxy	:::
Novell AppArmor	Remote Administration (VNC)	= 📰 = Samba Server	
Security and Users	= TFTP Server	Windows Domain Membership	
Virtualization		-	
Miscellaneous	3		-
<u>Search</u>			

• Choose TFTP Server in Network Services.



If the *TFTP Server* icon does not exist, restart the YaST Control Center.

Configuring a TFTP 🔶	🗠 TFTP Server Configuration
Server	
Use this to enable a server for TFTP (trivial file transfer	Enable
protocol). The server will be started using xinetd.	Boot Image Directory
Note that TFTP and FTP are not	/tftpboot Browse
the same.	Open Port in Firewall Firewall Details
Boot Image Directory: Specify the directory where served files are located. The	Firewall is disabled
usual value is /tftpboot. It will be created if it does not exist.	<u>V</u> iew Log

- Switch the selection to *Enable*.
- The path of the boot image directory should be */tftpboot*. If there is a different path, change it to */tftpboot*.

• Click Finish.



• Click *Yes* to create the */tftpboot* directory.

The TFTP server will be started.

• Close the YaST Control Center.



You have successfully finished the configuration of the host platform.

# 2.3 Linux-i.MX31-Kit Setup

In this section you will find a description of the Linux-i.MX31-Kit setup. The whole setup will be done by a graphical interface. At the end of the setup you will find all programs to develop applications for the target on your host PC.

The setup will install the following programs:

- *GNU C/C++ cross development tool chain* you can use this tool chain to develop programs for the target on your host PC.
- *Eclipse SDK with CDT* the Eclipse SDK is a platform and application framework for building software which can use the GNU C/C++ cross development tool chain.
- *Microcom* a program for serial communication with the target.
- *Linux Kernel archive* this kernel archive contains the Linux kernel source code as well as all patches needed to compile the kernel for the phyCORE-i.MX31.

- *HelloWorld* this example program can be used to test how to download and execute a program on the target.
- *mkimage* this program will be used to create the kernel image file for the target.

There will be some additional configuration steps performed on your PC:

- The setup program will create desktop links to the installed programs.
- The setup will also create desktop links to access the target via FTP, SSH, and Telnet.
- The path of the cross development tool chain will be added to the \$PATH environment variable.
- Read and write access to the serial interface will be added to your user account so you use the serial communication program Microcom.
- The setup will configure Microcom.

#### Starting the Setup

• To start with the Linux-i.MX31-Kit setup, put your PHYTEC Linuxi.MX31-Disc into your CD-ROM drive. The following dialog may appear:

🗙 CD-ROM - KDE Daemon 🎱 🛛 📪 🗙			
A new medium has been detected. What do you want to do?			
Medium type: Mounted CD Writer			
Open in New Window			
Always do this for this type of media			
Co <u>n</u> figure V <u>O</u> K X <u>C</u> ancel			

• Click Cancel.

s	earch:					<b>-</b>
	Systen Terr	n ninal				
	Ŀ	Terminal Pr Konsole	ogram			
	<b>^</b> _	Terminal Pr	ogram - Super Use	r Mode		
	>_	Terminal Prox	ogram			
•						
	$\bigstar$	$\bigcirc$				
Ē	avorites	History	Computer	Applications	Leave	
Us	er <b>develop</b>	er on linux-	36ру			open <b>SUSE</b> <sup>®</sup>
6		/ 🤣 /	🟫 🕵 i	1 2		

- From the *K menu*, select the *Applications* tab.
- Select System ► Terminal ► Terminal Program / Konsole.



- Type: cd /media/SO-458
- Enter ./install.sh to launch the setup program.



The media may be mounted on a different mount point in the directory */media*. The mount points can be shown with the command **ls /media**. Change to the accordant directory if no directory *SO-458* should exist.



• Enter the root password.



• Click Yes to proceed.

The welcome screen appears.



• Click *Next* to continue.

🔨 Linux-i.MX31-Kit Setup 🍥 📃 🗙			
Choose Destination Location			
Where should Linux-i.MX31-Kit be installed?			
Setup will install Linux-i.MX31-Kit in the following folder.			
Destination Folder			
/usr/local/share/iMX31-Kit Browse			
To install to a different folder, click browse and select another folder. NOTE: We suggest you to install Linux-i.MX31-Kit using the default location. Changes of the default folder must be considered when following the QuickStart Instructions.			
< Back Next > Cancel			

• Click Next.

🔧 Linux-i.MX31-Kit Setup 🍥 📃 🗙
Start Copying Files
Review settings before copying files
Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.
Install Directory:
/usr/local/share/IMX31-Kit
<pre>&lt; Back Next &gt; Cancel</pre>

• Click *Next* to copy all files to your hard disk.



The default destination location is /usr/local/share/i.MX31-Kit. All path and file statements within this QuickStart manual are **OPTION** based on the assumption that you accept the default installation paths. If you decide to individually choose different paths, you must consider this for all further file and path statements when working with this QuickStart.

> We strongly recommend accepting the default destination location.

🔨 Linux-i.MX31-Kit Setup 🕥		×
Installing		
Installing Linux-i.MX31-Kit		
Please wait while Setup installs Linux-i.MX31-Kit on your computer.		
Copying /usr/local/share/iMX31-Kit/linux-2.6.16-pcm037.tar.bz2		
< Back Next >	Cancel	

The GCC C/C++ tool chain will be installed to the default directory /opt/OSELAS.Toolchain-1.99.3/arm-1136jfs-linux-gnueabi. The program mkimage will be installed to /usr/local/bin. All other programs and examples will be installed to the selected destination directory.

After the files have been copied, a dialog box for the Eclipse installation will appear.



• Click *Yes* to install Eclipse. If you want to skip the installation of Eclipse, choose *No*.



We recommend installing Eclipse even if you already have installed Eclipse on your system. The version of Eclipse provided on the setup CD-ROM includes additional plug-ins.

• Click *Next*.



- Click *Finish* to exit the setup.
- Close the terminal window.

Now you will have to restart the KDE desktop.

Search:		<b>-</b>
		Session
<u>ک</u>	Logout End session	
	Lock Lock screen	
R	Switch User Manage parallel sessions	•
		System
0	Shutdown Computer Turn off computer	555 555
$\bigcirc$	Restart Computer Restart and boot this system	
	Suspend to Disk Pause without logging out	
	Standby Pause without logging out	
C	Start Operating System Restart and boot another operating system	► <b>▲</b>
$\stackrel{\frown}{\succ}$		
<u>-</u> avorites	History Computer Applications	
User <b>devel</b> e	oper on linux-36py	open <b>SUSE</b> <sup>®</sup>
6		

- Open the *K Menu* from the lower-left corner of the desktop.
- Select the *Leave* tab and choose *Logout*.
- When the display manager appears, enter your login name and password to restart the KDE desktop.



You have successfully installed the software for the Linuxi.MX31-Kit. You can now use the programs you need to develop your own applications for the target on your host system. The setup program did all necessary configurations. In the following passage you can find some advanced configuration information.

## 2.4 Advanced Configuration Information

In this part you can find some information on how to change the configuration steps of the setup program by your own. The setup program performed all the following configuration steps. The information in this part is for users who want to use the i.MX31-Kit with a Linux distribution other than openSUSE. This is also interesting for users who want to see what configurations the setup program did.

During the setup program, the GCC C/C++ cross compiler was installed in the directory /opt/OSELAS.Toolchain-1.99.3/arm-1136jfs-linuxgnueabi/gcc-4.3.2-glibc-2.8-binutils-2.19-kernel-2.6.27-sanitized/bin. To start the cross compiler directly from every location of the system, the directory of the cross compiler was added to the \$PATH environment variable. You can manually add the directory of the cross compiler to the \$PATH by adding the following line in the file /etc/profile:

# export PATH=/opt/OSELAS.Toolchain-1.99.3/arm-1136jfs-linuxgnueabi/gcc-4.3.2-glibc-2.8-binutils-2.19-kernel-2.6.27sanitized/bin:"\$PATH"

You can open a terminal program and use the cross compiler directly from the command line. For example, you can compile a C program with the following command:

#### arm-1136jfs-linux-gnueabi-gcc -o HelloWorld HelloWorld.c

In the standard configuration only the root user has write access to the serial interface. To use a serial communication tool like Microcom with normal user rights, you have to be a member of the group *uucp*. A user can be added to this group with the following command:

#### groupmod -A username uucp

The serial communication program was configured during the setup with the following configuration:

115200 baud, 1 start bit, 8 data bits, 1 stop bit, no parity, no flow control.

# **2.5** Connecting the Host with the Target

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

• Connect the serial cable with the UART1 (connector P1, TOP) port on the target and the first serial interface on your host.



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

• Connect the cross-over Ethernet cable with the connector X27 on the target and the appropriate network card of your host.



- Click the *Microcom* icon on your desktop.
- Connect the AC adapter with the power supply connector X30 (5V) on your board.



The power connector should have 5 VDC inside, and outside should be ground.

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



- Type **root** to login.
- After you have successfully logged in, you can close Microcom.

If you don't see the U-Boot and Linux starting and don't get a login prompt, you probably have a kit with Windows CE preinstalled. Please refer to the chapter "*Installing Linux on the phyCORE-i.MX31*" for instructions on how to install Linux in such a case.



When the target is connected with the power supply, first the boot loader U-Boot is loaded from the flash memory. Then the boot loader is uncompressing and booting the Linux kernel from the flash. The kernel will then mount the root file system, which is also located in the target's flash. The root file system uses the *Journaling Flash File System, Version 2*.

JFFS2 is the successor, and a complete rewrite, of the original JFFS by Red Hat. As its name implies, the JFFS2 implements a journaling file system on the memory technology device (MTD) it manages. JFFS2 does not attempt to provide a translation layer that enables the use of a traditional file system with the device. Instead, it implements a logstructured file system directly on the MTD. The file system structure itself is recreated in RAM at mount time by JFFS2 through a scan of the MTD's log content.

In addition to its log-structured file system, JFFS2 implements wear levelling and data compression on the MTD it manages, while providing power-down reliability. JFFS2 can gracefully restart, and is capable of restoring a file system's content, without requiring outside intervention regardless of power failures.



#### **Troubleshooting:**

If you don't see any output in the Microcom window, check the serial connection between the target and your host.

At the end of the setup, you had to restart the KDE desktop. If you haven't done yet, restart the KDE desktop now and try again.

It is also possible that your user account is missing read and write access to the serial interface:

- Open the *YaST Control Center*.
- Choose Security and Users.
- Choose User Management.

In the line of you user name should be the group *uucp*.

- If the group is missing, select your user name and click the *Edit* button.
- Select the tab *Details*.
- In *Groups*, check the group *uucp*.
- Click Accept.

- Click *Finish* and close YaST.
- You need to log out and log in again for the new group membership to take effect.

Now you can test the network connection to the target.



• Click the *Telnet for Target* icon on your desktop.

A new window with a connection to the target opens.



If you can see the user login in the opened window, the network configurations were configured correctly.

• Close the window.


### **Troubleshooting:**

If you don't see the user login, check the connection between the target and the host. If you have installed more than one network card on your host, be sure to connect the cable with the network card you have configured with the IP address 192.168.3.10.

If you do not see the login, you may not have set up the right IP address of your host. You can check the settings of your network card by opening YaST. In the YaST Control Center you can select *Network Settings* in *Network Devices*. There should be the following configuration:

AMD PCnet - Fast 79C971
<ul> <li>Device Name: eth-id-00:0c:29:1f:a5:52</li> <li>IP address: 192.168.3.10, subnet mask 255.255.255.0</li> <li>Started automatically at boot</li> </ul>
<u>A</u> dd Ed <u>i</u> t Dele <u>t</u> e

Information on how to configure your network device can be found in the section *Configuring the Host Platform*.

You have successfully set up all configurations to access your phyCOREi.MX31 from your host.

## 2.6 Copying an Example to the Target

In this section you will learn how to copy an example program to the target using the FTP protocol with the *Konqueror* browser. After that you will execute the example on the target. At the end of this passage you can find some information on how to copy and execute a file on the target using the command line.

#### Copying a Program to the Target



• First click the *i.MX31-Kit* icon on your KDE desktop.

A new window with the contents of the installation directory opens.

🖆 HelloWorld - Konqueror 🍥	?_□	×
Location Edit View Go Bookmarks Tools Settings Window Help		
i 🔾 🔍 🕢 🕜 😣 📥 🔍 🔍 🛄 🗉 🛤		Ϋ́
💈 🗈 Location: 📄 /usr/local/share/iM031-Kit/HelloWorld 💌 🗐 🚏 Filter This	s Folder	-
HelloWorld		
● One Item - One File (2.7 KB Total) - No Folders		

• Enter the directory *HelloWorld*.



• Click the *FTP for Target* icon on your desktop.

A window with an FTP session to the target opens. Now you have two windows opened, one for the target and one for the host. You can use these two windows to copy files per "drag and drop" from the host to the target (and vice versa).



- Select the window that lists *HelloWorld* program on your hard disk.
- Click the *HelloWorld* program and hold the left mouse button pressed.
- Drag the program into the window with the FTP session to the target and release the mouse button.
- Choose *Copy here* in the appearing context menu.
- Close the two windows.

#### Using Telnet to Execute a Program on the Target



• Click the *Telnet for Target* icon on your KDE desktop.



- Enter **root** as login and press **Enter**.
- Enter ./HelloWorld and press Enter.

The program starts and you should see the following output:

Welcome to the World of the phyCORE-i.MX31!

### Using SSH to Execute a Program on the Target

SSH can be used if you want to execute a program directly from the host on the target. Later, this will be used to execute programs out of Eclipse on the target. Before you can start programs out of Eclipse, you have to log in to the target via SSH from the command line for one time. This is necessary to add the RSA public key of the target to the list of known hosts.



There are several authentication methods when using SSH. The method used on the phyCORE-i.MX31 is the *hosts.equiv* method combined with RSA-based host authentication.

If the machine the user logs in from is listed in */etc/hosts.equiv* on the remote machine, and the user name is the same on both sides, the user is allowed to log in.

On the target, the file /etc/hosts.equiv has the following entry:

```
# file: /etc/hosts.equiv
#
# Allow access from everywhere.
#
+ +
```

The "+ +" means that every user can log in from every host.

When the host connects to the target, the file ~/.ssh/known hosts (on the host) is consulted when using hosts.equiv with RSA host authentication to check the public key of the target. The key must be listed in this file to be accepted. When the host connects to the target for the first time, you will be asked to store the target's RSA pubic key to your ~/.ssh/known hosts. If you agree to do this, then the host will be able to connect to the target without entering a password.



This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing. But note that */etc/hosts.equiv* is, in general, inherently insecure and should be disabled if security is a concern.



• Click the *SSH for Target* icon on the desktop.

A new window opens.



In this window you can see that the authenticity of the phyCOREi.MX31 can't be established. This is normal if you want to create an SSH connection for the first time.

• Enter yes and press Enter to continue. The RSA public key of the target will be permanently added to the list of the known hosts.



### **Troubleshooting:**

If an error occurs and you can't see the *root@phyCORE:~>* prompt, open a terminal window and enter the following command:

• rm ~/.ssh/known\_hosts

Try to log in again by entering:

- ssh root@192.168.3.11
- Enter **yes** to add the target to the list of known hosts.

Now you should see the target's prompt.



We expect that you haven't changed the SSH configuration file on your host. If you change this file, the authentication may not work. Now you are logged in, you can execute programs on the target.

• Type ./HelloWorld to start the program you had copied to the phyCORE-i.MX31 before.

The program starts and you should see the following output:

Welcome to the World of the phyCORE-i.MX31!

• Close the SSH window.



You have successfully copied and executed an example application on the target.

### 2.7 Advanced Information

#### Copying a Program to the Target with the Command Line

- Open a new terminal window.
- Change to /usr/local/share/i.MX31-Kit/HelloWorld:

#### cd /usr/local/share/i.MX31-Kit/HelloWorld

• Copy the application to the target by typing:

#### ftp -u ftp://root:root@192.168.3.11/ HelloWorld

Be sure to enter a slash followed by a space after the IP address.

#### **Executing a Program on the Target**

• Open a Telnet session to the target:

#### telnet 192.168.3.11

- Type **root** and press **Enter**.
- Type ./HelloWorld to start the application.

### Executing a Program directly on the Target using SSH

• To start the program, type:

## ssh root@192.168.3.11 ./HelloWorld

After the program has finished, SSH will logout automatically.

# **3** Getting More Involved



In this chapter you will pass some continuative topics. First you will configure and compile your own kernel. With the kernel configuration tool you can add additional features, or disable them if they are not needed. After compiling the kernel, you will learn how to write the newly created kernel into target's flash memory and how to start the new kernel.

Then you will start working with the Eclipse platform using the C/C++Development Tools (CDT) in conjunction with the GCC C/C++ tool chain. You will learn how to configure the Eclipse platform and how to open an existing project. After that you will create your first own project and modify the example's source code.

At the end of this chapter you will execute the program as an external application out of Eclipse. Additionally, you will add your application to the startup configuration of the target so it is automatically started when the phyCORE-i.MX31 boots.

## 3.1 Configuring and Compiling the Kernel

In this part you will learn how to configure and build a new Linux kernel. First you will copy the kernel archive to your home directory and extract the kernel source. Then you will configure the kernel with the graphical user interface *qconf*. After the configuration you will compile the new kernel using the GNU cross development tool chain.

The kernel used by PHYTEC is based a standard kernel available from *www.kernel.org*. Additionally, the kernel archive in your setup installation directory already includes all necessary patches for the phyCORE-i.MX31.

As the first step, open a new terminal.

<sup>© 2009</sup> PHYTEC Messtechnik GmbH L-703e\_4



- Click the terminal icon on your desktop.
- Type the following commands to copy the kernel archive to your home directory:

```
cp /usr/local/share/i.MX31-Kit/linux-*-PCM037.tar.bz2 ~
```

**cd** ~

• Unpack the kernel source archive:

#### tar xvjf linux-\*-PCM037.tar.bz2

• Change to the newly created directory:

#### cd linux-\*

The kernel would normally be built for the native machine architecture of your host. To use the ARM architecture and ARM cross compiler suitable for the phyCORE-i.MX31 instead, you will have to specify the architecture and the cross compiler on the command line.

• Type:

#### make xconfig ARCH=arm

The kernel configuration tool qconf starts.



- Select I2C support in Device drivers.
- Check *I2C support*.
- Check *I2C device interface*.



- Select I2C Hardware Bus support.
- Check *IMC I2C interface*.
- Save your configuration and exit the configuration tool.
- Type:

# make ARCH=arm CROSS\_COMPILE=arm-1136jfs-linuxgnueabi- uImage

The kernel sources will be compiled and the new kernel will be built. This will take a few minutes. The new kernel will be written to *arch/arm/boot/uImage*.



If the process of building the kernel stops with an error, check the values of both **ARCH** and **CROSS\_COMPILE**.

CAUTION Another possible problem could be an entry in the file /etc/profile . Check if the directory of the cross compiler was added to the \$PATH environment variable. In etc/profile you must find following entry:

# export PATH=/opt/OSELAS.Toolchain-1.99.3/arm-1136jfs-linux-gnueabi/gcc-4.3.2-glibc-2.8-binutils-2.19kernel-2.6.27-sanitized/bin:"\$PATH"

If not you can add this line manually.

• Close the terminal window.

In this section you learned how to configure and compile a new kernel. Now you can add new features to your kernel, or remove features you do not need.

### Writing the Kernel into the Target's Flash

In this passage you will find a description on how to write the newly created kernel into the phyCORE-i.MX31's flash memory. Before the kernel can be written into the flash, the target will have to download

the kernel from a TFTP server. This will be done from the command line of the boot loader. The kernel will be copied into target's RAM. Then you will have to erase the part of the flash where you want to copy the kernel image to. Finally the kernel is 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.

The four partitions have the following address ranges:

```
0xa000000 - 0xa003ffff (U-Boot)
0xa0040000 - 0xa005ffff (U-Boot Environment)
0xa0060000 - 0xa025ffff (Linux Kernel)
0xa0260000 - 0xa1ffffff (Linux Root File System)
```



You should never erase the U-Boot partition. If this partition is erased, you won't be able to start your target anymore. Refer to the chapter *"Installing Linux on the phyCORE-i.MX31"* for detailed information on how to restore your U-Boot partition in such a case.



- First open a new terminal window if it is not opened yet.
- Copy the new kernel image to the */tftpboot* directory and exit:

cp ~/linux-\*/arch/arm/boot/uImage /tftpboot; exit

#### 🜉 Microcom - Konsole . . • U-Boot 2.0.0-rc9 (Jul 28 2009 - 10:50:08) Board: Phytec phyCORE-i.MX31 NAND device: Manufacturer ID: 0x20, Chip ID: 0x76 (ST Micro NAND 64MiB 3,3V 8t) Scanning device for bad blocks RCSR\_BTP: 20 Using environment in NOR Flash got MAC address from EEPROM: 00:50:C2:95:4E:08 m×31 cpu clock: 513MHz ipg clock : 64133247Hz Malloc space: 0x87b00000 -> 0x87f00000 (size 4 MB) Stack space: 0x87af8000 -> 0x87b00000 (size 32 kB) envfs: wrong magic on /dev/env0 no valid environment found on /dev/env0. Using default environment running /env/bin/init... Hit any key to stop autoboot: 3 type update\_kernel nand|nor [<imagename>] to update kernel into flash type update\_rootfs nand|nor [<imagename>] to update rootfs into flash uboot:/ 📕

• 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.

You can download the kernel from the TFTP server to the target's RAM, erase the required flash area, and write the kernel from the RAM into the flash with just one simple command: *update\_kernel*.

#### 🜉 Microcom - Konsole \_ = × #!/bin/sh \* use 'dhcp' to do dhcp in uboot and in kernel # #ip=dhcp # or set your networking parameters here eth0.ipaddr=192.168.3.11 eth0.netmask=255.255.255.0 eth0.gateway=192.168.3.10 eth0.serverip=192.168.3.10 # can be either 'net', 'nor' or 'nand'' kernel\_loc=nor rootfs\_loc=nor kernel=uImage-pcm037 rootfs=root-pcm037.jffs2 envimage=u-boot-v2-environment-pcm037 autoboot\_timeout=3 nfsroot=192.168.3.10/root bootargs="console=ttym×c0,115200" #bootargs="\$bootargs video=mx3fb:TX090"

• Type the following command to check your U-Boot settings:

### edit /env/config

You will see the configuration file which holds U-Boot's environment variables.

• Make sure that the following values are set in the configuration file:

```
eth0.ipaddr=192.168.3.11
eth0.netmask=255.255.0.0
eth0.serverip=192.168.3.10
```

- Type *CTRL-D* to save the settings to the file.
- If you made any changes to the U-Boot environment, type **save** to write these changes to the U-Boot environment partition, and then press the target's RESET button. The phyCORE-i.MX31 reboots with the new settings applied. Again, press any key to stop autoboot.

- Type **update\_kernel nor uImage** to download the kernel using TFTP and write it into the target's flash. The copy process can take up to a minute.
- Press the RESET button on the target to restart the phyCORE-i.MX31 with the new kernel. The target will boot the newly created kernel.
- Close Microcom when the target has successfully finished with booting the kernel and mounting the root file system.



### **Troubleshooting:**

If any problem occurs after writing the kernel into flash, you can restore the original kernel from your setup CD-ROM.

You can find this kernel in the directory *PHYTEC/PCM037* phyCORE-i.MX31/Linux-Kit/BSP/Images.

- To restore the kernel, copy the file *uImage-pcm037* to your host's */tftpboot* directory.
- Type **update\_kernel nor uImage-pcm037** to download and write the kernel into the target's flash.

If you ever happen to damage your target's root file system, you can also find the original root file system in the *PHYTEC/PCM037 phyCORE-i.MX31/Linux-Kit/BSP/Images* directory on your setup CD-ROM.

- To restore the root file system, copy the file *rootpcm037.jffs2* to your host's */tftpboot* directory.
- Type **update\_rootfs nor root-pcm037.jffs2** to download and write the file system into the target's flash.



In this section you learned how to download a kernel image from a TFTP server into the RAM of the target. The kernel was written from RAM to flash, and finally the target was started with the new kernel.

# 3.2 Opening an Existing Project

In this section you will import an existing Eclipse project into your workspace. The imported example project will be compiled with the cross compiler. After compiling the project, you will copy and execute the newly created program on the target.

### Copying the HelloWorld Project



- Click the *i.MX31-Kit* icon on your KDE desktop.
- Right-click the *HelloWorld* directory and select *Copy*.
- Browse to your home directory.
- If the *workspace* directory doesn't exist, create a directory *workspace* in your home directory.
- Enter the *workspace* directory.
- Right-click in the *workspace* directory and select *Paste*.

#### **Starting Eclipse and Importing the Example Project**



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

🕘 Workspa	ace Launcher 🎐		_ ×
Select a wo	rkspace		
Eclipse SDK	stores your projects in a folder called a	workspace.	
Choose a wo	orkspace folder to use for this session.		
<u>W</u> orkspace:	/home/ptxdist/workspace	<b>•</b>	<u>B</u> rowse
🗌 <u>U</u> se this a	as the default and do not ask again		
		ок	Cancel

• Confirm the Workspace directory with *OK*.

The Welcome screen will appear.



• Select *File*  $\blacktriangleright$  *Import* from the menu bar.

🗧 Import 🎐	
<b>Select</b> Create new projects from an archive file or directory. This does not copy the project into the workspace.	è
Select an import source:	
🕼 Archive file	
📽 Checkout Projects from CVS	
💕 Existing Projects into Workspace	
Resternal Features	
sternal Plug-ins and Fragments	
📮 File system	
Preferences	
🝓 Team Project Set	
< Back Next > Finish	Cancel

- Select Existing Projects into Workspace.
- Click Next.

🗧 Import 🧕	
Import Projects	
Select a directory to search for existing Eclipse projects.	
Select roo <u>t</u> directory:	B <u>r</u> owse
○ Select <u>a</u> rchive file:	Browse
<u>P</u> rojects:	
	Select All
	Deselect All
	R <u>e</u> fresh
< <u>B</u> ack Next > Enish	Cancel

• Select Browse.

🗁 Home	workspace HelloWorld	Cre	ate Folder
🎦 Desktop	Name	+	Modified
🗟 File System	HelloWorld.c		Today

- Double-click the *HelloWorld* directory in your home directory.
- Click OK.

🖨 Import 🎐					
Import Projects					
Select a directory to sear	ch for existing Eclip	se projects.			
Select root directory:	/home/ptxdist/work	space/HelloWorld	ŧ		B <u>r</u> owse
O Select <u>a</u> rchive file:					Browse
<u>P</u> rojects:					·
X HelloWorld					Select All
					Deselect All
					R <u>e</u> fresh
		- Pack	Norta	Finish	Cancol
		< Back	[ <u>iv</u> ext>]	Emish	Cancel

• Select *Finish* to import the project.

👙 C/C++ - Eclipse SDK 🎐					_ <b>—</b> ×
<u>F</u> ile <u>E</u> dit Refac <u>t</u> or <u>N</u> avigat	e Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp				
] 📬 🖩 👜 ] 🔂 🔂 🔂	G•   \$• 0• 4•   \$ 4	• •			🖹 🗟 C/C++ 🐉 Java
E C/C++ X <sup>≫</sup> 1 □				Br Outline ⊠	Make Targets
	Problems & Console Properties o errors, 0 warnings, 0 infos Description	Resourc	e In Folde	r	Location
					••

• Close the *Welcome* screen.

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 FTP. After the file has been copied to the target, the program is executed on the target using SSH. You should now see the *"Welcome to the World of the phyCORE-i.MX31!"* message in the *Console* window.

• Select the *Console* tab.

You will see the following content in the Console window:

Problems Javadoc Declaration 📮 Console 🛿			đ	<b>.</b>	[] ▼	- 8
C-Build [HelloWorld]						
Building target: HelloWorld Invoking: GCC C Linker arm-1136jfs-linux-gnueabi-gcc -oHelloWorld ./HelloWorld.o Finished building target: HelloWorld						
makeno-print-directory post-build ftp -u ftp://root:" "@192.168.3.11/ ./HelloWorld; ssh root@1 Welcome to the World of the phyCORE-i.MX31!	.92.1	168.3	. 11	./Hell	loWorl	.d
Build complete for project HelloWorld						-



If the project is not built automatically, you will have to check *Project*  $\blacktriangleright$  *Build automatically* from the menu bar.



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

# 3.3 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 tool chain.

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

A new dialog opens.

🗧 New Project 🧶 🛛	
Select a wizard	$\diamond$
Create a new C project and let Eclipse create and manage the makefile	1º
<u>W</u> izards:	
/샬 Java Project	
影 Java Project from Existing Ant Buildfile	255
🞥 Plug-in Project	
▼ ( <del>)</del> C	
🖻 Managed Make C Project	
🔁 Standard Make C Project	
▶ 🧀 C++	J
	0
< Back Next > Einish Canc	el

• Select Managed Make C Project and click Next.

🖨 New Projec	<b>t</b>			
Managed Make	C Project			-G
Create a new M	anaged Make C	project.		
Project name: Project content I Use default Directory: //ho	myHelloWorld 5 me/ptxdist/work	space/myHelloW	orid	Browse
	< <u>B</u> ack	Next >	<u>F</u> inish	Cancel

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

🖨 New Project 🧶	
Select a type of project Select the platform and configurations you wish to deploy on	G
Project Type: Executable (Gnu) Configurations:	•
🗶 🔊 Debug	
🗙 🔊 Release	
<ul> <li>Show All Project Types</li> <li>Show All Configurations</li> </ul>	
< Back Next > Einish C	Cancel

• Click Next.

🖨 New Project 🎐	
Additional Project Settings	G
Define the inter-project dependencies, if any.	
Projects C/C++ Indexer Referenced C/C++ Projects	
HelloWorld	
< Back Next > Finish	Cancel

• Click *Finish*.



• Select *Yes* to open the C/C++ perspective.

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



≂ 😤 HelloWorld	
👂 🔷 Binaries	
👂 🛃 Includes	
👂 🔁 Debug	
🕨 🖻 HelloWorld.c	New
👂 📂 myHelloWorld	
	Op <u>e</u> n
	Open Wit <u>h</u> ▶
	<u>∎</u> <u>С</u> ору
	💼 Paste
	💢 <u>D</u> elete

- Right-click on *HelloWorld.c* in the *HelloWorld* project which we have worked with previously.
- Select *Copy*.



- Select the *myHelloWorld* project.
- Right-click and 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 be run on your host system, in the *workspace/myHelloWorld/Debug* directory.

To compile your project for the phyCORE-i.MX31 instead, you will have to use the GNU C/C++ cross compiler.

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

👙 Properties for myHello	World 🎱								
type filter text 💌	C/C++ Build								↓ ↓ +
Info	Active configura	ation							
Builders	Project Type:	Executable (Gnu)							-
C/C++ Build C/C++ Documentation	Configuration:	Debug						•	Manage
C/C++ File Types	Configuration S	ettings							
C/C++ Indexer	Tool Settings	Build Settings	Build S	teps	Error F	Parsers	Binary Parser	Environment	Macros
	<ul> <li>         ● 第 GCC 4     </li> <li>         ● Pre         ● Syn         ● Syn         ● Dir         ● Dir         ● Dir         ● Dir         ● Wa         ● Mas         ●</li></ul>	C Compiler processor nbols ectories timization bugging urnings scellaneous C Linker enral raries		All o	mand: otions:	arm-11	36jfs-linux-gnue.	abi-as	
	ي Mis ک Sh ★ St GCC / ن Ge	scellaneous ared Library Setting Assembler neral	gs						
							Restore	Defaults	Apply
							0	к	Cancel

The Properties dialog appears.

- Select C/C++ Build.
- Enter arm-1136jfs-linux-gnueabi-gcc into the Command input field.

Se SDK Properties for HelloWo	rtd 🎱 🗖 🖬
type filter text 👻	C/C++ Build 🔅 🔻 🖒 🕫
Info Builders C/C++ Build C/C++ Documentation C/C++ File Types C/C++ Indexer Project References	Active configuration Project Type: Executable (Gnu) Configuration: Debug Tool Settings Tool Settings Build Settings Build Steps Error Parsers Binary Parser Environment Macros           Configuration Settings         Tool Settings Build Settings Build Steps Error Parsers Binary Parser Environment Macros <ul> <li>Second Settings</li> <li>Second Settings</li> <li>Command: arm-1136jfs-linux-gnueabl-gcc</li> <li>All options:</li> <li>Directories             <ul> <li>Optimization</li> <li>Debugging</li> <li>Warnings</li> <li>Miscellaneous</li> <li>Shared Library Settings</li> <li>Shared Library Settings</li> <li>Score al</li> <li>S</li></ul></li></ul>
	Restore Defaults Apply
	OK Cancel

- Select GCC C Linker.
- Enter **arm-1136jfs-linux-gnueabi-gcc** into the *Command* input field.

🍯 Properties for HelloWo	e @	X
type filter text 💌	C/C++ Build 🗢 🔻	\$.
Info Builders C/C++ Build C/C++ Documentation C/C++ File Types C/C++ Indexer Project References	Active configuration Project Type: Executable (Gnu) Configuration: Debug  Configuration Settings Tool Settings Build Steps Error Parsers Binary Parser Environment Mac	ge
	All options:	
(	Restore Defaults Apply	/
	OK Cancel	

- Select GCC Assembler.
- In the *Command* input field, change the default **as** to **arm-1136jfslinux-gnueabi-as.**
- Click *Apply*.

🗧 Properties for myH	ielloWorld 🎐 🔹 🗆
ype filter text 💌	C/C++ Build ⇔ ▼ ⇔ ▼
Info Builders C/C++ Build C/C++ Documentatic C/C++ File Types	Active configuration Project Type: Executable (Gnu) Configuration: Debug Configuration Settings
C/C++ Indexer	Tool Settings Build Settings Build Steps Error Parsers Binary Parser Environment
Project References	Pre-build step: Command: Description: Post-build step: Command: ftp -u ftp://root:" "@192.168.3.11 //myHelloWorld;ssh root@192.168.3.11 ./myHelloWorld Description:
(	Restore Defaults Apply
	OK Cancel

- Select the *Build Steps* tab.
- Enter following command in the *Command* input field:

ftp -u ftp://root:root@192.168.3.11/ ./myHelloWorld ; ssh root@192.168.3.11 ./myHelloWorld



Be sure to enter the semicolon between ./myHelloWorld and ssh.

- Click *Apply*.
- Click OK.



- Select *Project*  $\blacktriangleright$  *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:



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

# **3.4** Changing 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 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 phyCORE-i.MX31, 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.

In the next screenshot you can see the complete program.

_						
.c	] mył	HelloWorld.c 🗙			-	
	#in #in #in	c <b>lude</b> <unistd.h> c<b>lude</b> <fcntl.h> c<b>lude</b> <stdio.h></stdio.h></fcntl.h></unistd.h>				
	/* <b>voi</b>	write n bytes to the serial inter <mark>d write_tty(char *buffer, int cou</mark>	face */ nt)			
	L	int out;	/* variable for file describtor	*/		
	}	<pre>out=open("/dev/ttymxc0",0_RDWR); write(out,buffer,count); close(out);</pre>	/* open interface /* write n bytes /* close the serial interface	*/ */ */		
	int {	main()				
		<pre>char buf[]={"Welcome to the World")</pre>	d of the phyCORE-iMX31! (serial)\n"	'}; /* output variable 🛛 ᡟ/		
		<pre>write_tty(buf,sizeof(buf)-1); printf("Welcome to the World of return 0;</pre>	<pre>the phyCORE-i.MX31!\n");</pre>	/* write buffer to tty */ /* write to stdout    */		
	}				-	]

• Save your program after changing the code.

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

Problems	Javadoc	Declaration	📃 Console 🕱	۵		2	•	[] ▼	- 0
C-Build (my	/HelloWor	id]							
Building Invoking: arm-1136 Finished	target: GCC C L jfs-linu: building	myHelloWor Linker x-gnueabi-( g target: n	rld gcc -oHelloWorld .∕HelloWorl nyHelloWorld	ld.o					
makeno ftp -u fi myHelloWo Welcome i	)-print-0 :p://roof )rld :o the Wo	directory p t:" "@192.] orld of the	oost-build 168.3.1 / ./myHelloWorld;ssh e phyCORE-i.MX31	n root@192	.168.	3.11	.7		
Build com	iplete fo	or project	myHelloWorld						•

#### **Executing the Program on the Target using Microcom**

- Click the *Microcom* icon on the desktop.
- If you are not logged in, enter root and press Enter.
- Type ./myHelloWorld to start the application.
- You will see the following output: *Welcome to the World of the phyCORE-i.MX31! (serial) Welcome to the World of the phyCORE-i.MX31!*
- Close Microcom.

When you start the application over 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't 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 serial interface, so you can also see the line that it written to the serial console.

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

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

## **3.5** Starting a Program out of Eclipse on the Target

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

😂 External Tools 🎱	
Create, manage, and run configurations	
Ready to launch	
Configurations:	
影 Ant Build	
Q Program	
New Delete	
	Run Close

• Select *Run* ► *External Tools* ► *External Tools* from the menu bar.

🖨 External Tools 🎐	
Create, manage, and run o	configurations
Please specify the location of	the external tool you would like to configure.
Con <u>f</u> igurations:	Name: New_configuration
∦ Ant Build	
✓ Q Program New configuration	Aan AREFresh Refresh Refresh Refresh
va New_configuration	Location:
	Browse Workspace Browse File System Varjables
	Working Directory:
	Browse Wor <u>k</u> space Browse File Syste <u>m</u> Varia <u>b</u> les
	Arguments:
	Variables
	Note: Enclose an argument containing spaces using double-quotes (").
Ne <u>w</u> Dele <u>t</u> e	ApplyRevert
	Bun Close

- Select Program.
- Select New.

uur a program     Infigurations:     * Ant Build     Program     Man @Refresh @Environment @Common     Location:     Location:     Location:     Morking Directory:     Browse Workspace Browse File System Variables     Working Directory:     Image: Totol@192.168.3.11 /myHelloWorld     New     Delege     Apply     Apply	reate, manage, and run	configurations		2
Implyingurations:       Name: myHelioWorld Target         Implyingurations:       Implyingurations:         Implyingurations:       Implyingurations: </th <th>un a program</th> <th></th> <th></th> <th></th>	un a program			
* Ant Build         Program         Main @Refresh TEnvironment Common         Location::         //usr/bin/ssh         Browse Workspace         Browse File System         Variables         Working Directory:         Browse Workspace         Browse File System         Variables         Arguments:         root@192.168 3.11 /myHelloWorld         Variables         Variables         Note: Enclose an argument containing spaces using double-quotes (").         Apply         Revent	nfigurations:	Name: myHelloWorld Target		
Program         Imain @Refresh @Environment ©Common         Location::         /usr/bin/ssh         Browse Workspace         Browse File System         Variables         Working Directory::         Browse Workspace         Browse File System         Variables         Arguments:         root@192.168.3.11 /myHelloWorld         Variables         New         Delete	条 Ant Build			
Location: Location: /Variables /Variables	Program	Main Refresh REEnvironment Common		
Ausr/bin/ssh         Browse Workspace       Browse File System       Variables         Working Directory:	wannynen <del>on ond range</del> t	Location		
Browse Workspace       Browse File System       Variables         Working Directory:		/usr/bin/ssh		
Working Directory:         Browse Workspace         Browse File System         Variables         Arguments:         root@192.168.3.11 ./myHelloWorld         Variableg         Note: Enclose an argument containing spaces using double-quotes (*).         New         Delete			Browse Workspace Brow	vs <u>e</u> File System Var <u>i</u> ables
Browse Workspace       Browse File System       Variables         Arguments:       root@192.168.3.11 ./myHelloWorld       Variableg         Note: Enclose an argument containing spaces using double-quotes (*).       Variableg         New       Delete       Apply		-Working Directory		
Browse Workspace       Browse File System       Variables         Arguments:       root@192.168.3.11 ./myHelloWorld       Variableg         Note: Enclose an argument containing spaces using double-quotes (*).       Variableg         New       Delete       Apply		Working <u>Directory</u> .		
Browse Workspace       Browse File System       Variables.         Arguments:       root@192.168.3.11 ./myHelloWorld       Variableg.         Note:       Enclose an argument containing spaces using double-quotes (*).       Variableg.         New       Delete       Apply       Reyzert				
Arguments:         root@192.168.3.11 ./myHelloWorld         Variableg         Note: Enclose an argument containing spaces using double-quotes (*).         New       Delete         Apply       Reyzert			Browse Wor <u>k</u> space Brow	vse File Syste <u>m</u> Varia <u>b</u> les
root@192.168.3.11 ./myHelloWorld     Variableg       Note: Enclose an argument containing spaces using double-quotes (*).     Apply       New     Delete		Arguments:		
Note: Enclose an argument containing spaces using double-quotes (*).         Variableg           New         Delete         Apply         Reyert		root@192.168.3.11 ./myHelloWorld		
Ne <u>w</u> Delete Apply Revert		Note: Enclose an argument containing spaces usir	ng double-quotes (").	Variable <u>s</u>
Ne <u>w</u> Delete Apply Reyert				
	Ne <u>w</u> Dele <u>t</u> e	]		ApplyRevert

- In the Name input field, enter: myHelloWorld Target
- Enter /usr/bin/ssh in the Location input field.
- Enter root@192.168.3.11 ./myHelloWorld into the Arguments field.
- Select *Apply*.



• Select *Run*.

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


## **3.6** Automatically Starting the Program when Booting the Target

In this passage you will integrate the *myHelloWord* program into the startup process of the target. When you have finished this part, the *myHelloWorld* application will be started automatically each time you are starting the target.



The scripts for controlling the system startup live in */etc/init.d.* These are executed directly or indirectly by */sbin/init*, the father of all processes. The configuration of */sbin/init* is placed in */etc/inittab*.

After system startup, */sbin/init* will switch to the default run level, as configured in */etc/inittab*. It calls the run level master script */etc/init.d/rcS* to start or stop services provided by the other scripts in */etc/init.d*. This is done by the help of symbolic links in the directory */etc/rc.d*. These links point to the actual startup scripts in */etc/init.d*.

First you will have to create a startup script in /etc/init.d.



• Click the *FTP for Target* icon on your KDE desktop.

-	ftp://root@192.1	.68.3.11/etc/in	it.d - Konquero	r)	? _ 🗆 🗙						
Loc	ation <u>E</u> dit <u>V</u> iew	<u>G</u> o <u>B</u> ookmar	ks <u>T</u> ools <u>S</u> ettin	gs <u>W</u> indow	<u>H</u> elp						
G											
▶ Location: (a) ftp://root@192.168.3.11/etc/init.d											
🕼 📧 😍 Home Folder 🕟 🗱 🕲	banner Danner networking thttpd	gpio pure-ftpd udev	gpioevent pwm xorg	inetd rcS	modules telnetd						
	😝 13 Items - 13 F	iles (9.1 KB Tota	l) - No Folders								

• Browse to the target's */etc/init.d.* If an authorization dialog should appear, just click on *OK* (no password is required for FTP access).

In the directory */etc/init.d* you can see the existing scripts.

• Right-click in the opened window and select *Create New* ► *Text File*.

🗿 Konqueror 🎐	? 🗆 🗙
<u>E</u> nter text filename:	
myHelloWorld	
😰 C <u>l</u> ear 🔽 <u>O</u> K	🔀 <u>C</u> ancel

- Enter *myHelloWorld*.
- Click OK.
- Right-click on *myHelloWorld* and select *Open with*.
- Enter **kwrite** and click *OK*.

The text editor *KWrite* starts with an empty document.



• Enter the following two lines:

#!/bin/sh

/home/myHelloWorld

- Select *File*  $\blacktriangleright$  *Save*.
- Close the *KWrite* window.
- Close the *FTP* window.



• Click the *Telnet for Target* icon on your desktop.

Session Edit V	lew Bookmarks	Settings Help			
root@phyCORE	E:/etc/rc.d	ls -1		A THE REPORT OF A DESCRIPTION OF A DESCRIPTION OF	
lrwxrwxrwx	1 root	root	14 Apr 20	2009 SOOudev ->/init.d/udev	
lrwxrwxrwx	1 root	root	17 Apr 20	2009 S08syslog ->/init.d/syslogd	
lrwxrwxrwx	1 root	root	17 Apr 20	2009 S16openssh ->/init.d/openssh	
lrwxrwxrwx	1 root	root	17 Apr 20	2009 S16telnetd ->/init.d/telnetd	
lrwxrwxrwx	1 root	root	20 Apr 20	2009 S21alsa-utils ->/init.d/alsa-utils	
lrwxrwxrwx	1 root	root	20 Apr 20	2009 S26networking ->/init.d/networking	
lrwxrwxrwx	1 root	root	18 Apr 20	2009 S91lighttpd ->/init.d/lighttpd	
lrwxrwxrwx	1 root	root	18 Apr 20	2009 S91pureftpd ->/init.d/pureftpd	
lrwxrwxrwx	1 root	root	17 Apr 20	2009 S98modules ->/init.d/modules	
lrwxrwxrwx	1 root	root	16 Apr 20	2009 S99banner ->/init.d/banner	
-rwxr-xr-x	1 root	root	77 Apr 20	2009 S99zzz_PHYTEC_BSP_version_startup_script	
root@phyCORE	E:/etc/rc.d				
· · ·		2			ſ

- Enter **root** and press **Enter** to login.
- The startup script we have just created with KWrite must be made executable in order to run it later:

## chmod a+x /etc/init.d/myHelloWorld

• Change to the directory */etc/rc.d.* Type the following command:

#### cd /etc/rc.d

• Enter **Is** -**I** to list the directory contents.

You can see the different links to the scripts in the directory */etc/init.d.* The scripts are started in alphabetic order: The script *udev* is the first script started because the link starts with *S00...*, whereas *S99zzz\_PHYTEC\_BSP\_version\_startup\_script* will be started last.

To start your *myHelloWorld* application automatically when the system boots, you have to create a new link to the start script /*etc/init.d/myHelloWorld*.

• In */etc/rc.d*, create a symbolic link which points to */etc/init.d/myHelloWorld*. Enter the following command:

## ln -s /etc/init.d/myHelloWorld /etc/rc.d/S99myHelloWorld

- Type **ls** -**l** again to check the newly created link.
- Close the window.
- Open Microcom.
- Push the RESET button on the target to restart your system.



The program *myHelloWorld* now starts automatically on startup. Because its link in */etc/rc.d* starts with *S99...*, you should see *myHelloWorld*'s output near the output of the two other scripts that start with *S99...* (which print all sorts of version information).

• Close Microcom.

Now you can add your own programs to the root file system and start these programs automatically when your phyCORE-i.MX31 boots.



You have successfully passed the "Getting More Involved" part of this QuickStart.

# 4 Debugging an Example Project



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

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

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

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

# 4.1 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. With the default Eclipse settings, these debugging symbols have already been added when building the program.



• Open Microcom.



- Type root and press Enter.
- Start the GDB server:

## gdbserver 192.168.3.11:10000 myHelloWorld

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

## 4.2 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.
- Select *myHelloWorld* in the *Navigator* window.
- Select  $Run \triangleright Debug$  from the menu bar.

A dialog to create, manage, and run applications will appear.

Con <u>f</u> igurations:	<u>N</u> ame: m	yHelloWorld			
C/C++ Attach to Local Applic	2				
▼ C/C++ Local Application	Main	(x)=Arguments		Bebugger Sourc	e Common
c myHelloWorld		er Arguments		pocougger poure	
C/C++ Postmortem debugge	Project:				
Eclipse Application	myHello	World			Browse
🕎 Java Applet	C/C++ A	pplication		j	
🗊 Java Application	Debug/	nvHelloWorld		Search Project	Browse
J <mark>u</mark> JUnit	[Debug,	ingrice in o fronta		[ Scarell Hoject]	D_ONDC
🔐 JUnit Plug-in Test	Conr	ect process in	nut & output to a	terminal	
🖳 Remote Java Application	in com	eee process ii	iput di output to u		
SWT Application					
<b>(</b> ) (۱)					
				Appaly	

- Select *C*/*C*++ *Local Application*.
- Click New.

🖨 Program Selection 🎱 📃 🗙
Choose a <u>p</u> rogram to run:
Binaries:
💓 myHelloWorld
Qualifier:
🇱 armle - /myHelloWorld/Debug/myHelloWorld
OK Cancel

- Select the *Search Project* button.
- Click OK.

Con <u>f</u> igurations:	Name: myHelloWorld	
C/C++ Attach to Local Application	or	
C/C++ Local Application	Main @-Arguments REnvironment Spebugger	Source Common
c myHelloWorld		
C/C++ Postmortem debugger	Debugger: GDB Server	
Eclipse Application	Stop at main() on startup Advanced	
题 Java Applet	-Debugger Ontions	
🗊 Java Application		
Ju JUnit	Main Shared Libraries	
📅 JUnit Plug-in Test	GDB debugger gdb	Browse
Remote Java Application	GDB command file	Browse
SWT Application		
	Connection: Serial 🔻	
	Device: /dev/ttyS0	
	Speed: 115200 -	

- Select the *Debugger* tab.
- Select *GDB Server* from the *Debugger* drop-down box.
- Click the *Browse* button right beside the *GDB debugger* input field.

A new dialog opens to choose the GDB executable.

- Click on *File System*.
- Navigate to the directory /opt/OSELAS.Toolchain-1.99.3/arm-1136jfslinux-gnueabi/gcc-4.3.2-glibc-2.8-binutils-2.19-kernel-2.6.27sanitized/bin.
- Select the file *arm-1136jfs-linux-gnueabi-gdb*.
- Click OK.

😝 Debug 🍥		
Create, manage, and run configurations image: Main): Program does not exist		Ť.
Configurations: C C/C++ Attach to Local Application C C/C++ Local Application C C/C++ Dostmortem debugger Eclipse Application J Java Applet J Java Application JU JUnit JU JUnit Remote Java Application S WT Application	Name: New_configuration  Main to-Arguments   Environment   Debugger Es Source   Common Debugger: GDB Server  Stop at main() on startup Advanced  Debugger Options Main Shared Libraries GDB debugger 28-binutils-219-kernel-26 27-sanitized/bin/rm-1136/fs-linuxcagnueabl-gdt GDB command file: Connection: Serial   Device: //devthySO Speed: 115200	Browse Browse
Ne <u>w</u> Dele <u>t</u> e	Apply	Revert
	Debug	Close

- From the Connection drop-down box, select TCP.
- 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.

🖨 Conf	irm Perspective Switch 🎐		×				
?	This kind of launch is configured to open the Debug perspective when it suspends. Do you want to open this perspective now?						
□ <u>R</u> eme	ember my decision	Yes	No				

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



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

🖕 Debug - myHelloWorld.c - Eclipse SDK 🕘				_	I X
<u>Eile E</u> dit Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp					
] 🖸 🕶 📓 🚵 📄 🕸 🕶 🔕 🕶 🥔 🥔 🏈 🖢 🐄 🖓 🖉 🐐 🌾 🗇 т 🔶 т			E	🐝 Debug	»
🏇 Debug 🕱 🗖 🗖 🚧 Variables 🕺 Breakpoints Mod	dules Reg	isters Signals			
▶ III ■ ▶ ¾ ∅ 3. ۞ ⊥Ê ➡ ℜ i→ ▼		<b>*</b> =>ta		۵ 🛪	× ×
👻 💽 myHelloWorld [C/C++ Local Application]					
👻 🤣 GDB Server (12/15/06 2:11 PM) (Suspended)					
👻 🔐 Thread [1] (Suspended)					
≡ 1 main() at/myHelloWorld.c:17					
📕 Debugger Process (12/15/06 2:11 PM)					<b>A</b>
A /home/developer/workspace/mvHelloWorld/Debug/mvHelloWorld (12/15/06 2:1					
		_			
Re myHelloWorld.c 🕱		Outline 🛛			- 0
<pre>out=open(*/dev/ttymxc0*,0_FDWR); /* open interface */ vrite(out,buffer,count); /* write n bytes */ close(out); /* close the serial interface */ } int main() { char buf[]={"Welcome to the World of the phyCORE-iMX31! (serial)\n"}; /* output variable */ </pre>		<ul> <li>unistd.h</li> <li>fcntl.h</li> <li>stdio.h</li> <li>write_tty</li> <li>main</li> </ul>	Jªz	<i>₹ ¥</i> <sup>5</sup>	• ~
<pre>write tty/uur,sizeu/tour)-1/; /* write burrer to tty */ printf("welcome to the World of the phyCORE-i.MX31!\n"); /* write to stdout */ return 0;</pre>					
E Console X Tasks Memory		🍇 🔓 🚮	2	📮 🕇 📑	
myHelloWorld [C/C++ Local Application] /home/developer/workspace/myHelloWorld/Debug/myHelloWorld (12/15/06 2:11 PM)					
			_		

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.

# 4.3 Setting a Breakpoint

Now you will set a breakpoint in your program. The breakpoint will be set on the last line. 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.

# 4.4 Stepping and Watching Variable Contents

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

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

🔅 Variables 🕱 Breakpoints Mode	ules Registers	Signals							
			<u>*</u>	⇒ti	E	66°	ж	*	~
▽ 🥭buf									
(×)=buf[0] = .									
(×)=buf[1] = .									
(×)=buf[2] = .									
(×)=buf[3] = .									
(×)=buf[4] = '%'									
(×)=buf[5] = '@'									
(×)=buf[6] = .									
(≫=buf[7] = .									
0 '\0'									
								_	
•								• •	

• Click on 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.

၊ Variabl	es ;	x	Break	points	M	lodules	Regis	ters	Sigr	nals					' 🗆 🛛
									<u>x</u>	⇒ŧ	E	60	×	*	▽
🕨 🚝 buf												_			
															_
"Welcome	to	the	World	of t	he	phyCOF	RE-iMX	31!	(se	rial	}\n"				
•															9

• Click on the variable *buf*.

k	myH	felloWorld.c 🕱			- 0
	yoi	d write_tty( <mark>char</mark> *buffer, <b>int</b> cou	it)		
	ı	int out;	/* variable for file describtor	*/	
۰.		<pre>out=open("/dev/ttymxc0",0_RDWR);</pre>	/* open interface	*/	
		write(out,buffer,count);	/* write n bytes	*/	
	}	close(out);	/* close the serial interface	*/	
	int	main()			
		<pre>char buf[]={"Welcome to the World")</pre>	f of the phyCORE-iMX31! (serial)\n"}	; /* output variable     */	/ <mark>  _</mark> _
		<pre>write_tty(buf,sizeof(buf)-1);</pre>		/* write buffer to tty */	
					Þ

• Then click on the button *Step into* to enter the function *write\_tty()*.

The debugger stops in *write\_tty()*.

You will see the following variable window:

(x)= Variables 🗙	Breakpoints	Modules	Regist	ters Sigr	nals					' 🗆
				2	¢ti		60 0	×	*	~
▶ 🧳 buffer = Oxbed	b9d1b									
(x) <sup>≥</sup> count = 52										
(×)= out = 0										
Oxbedb9d1b "Welco	ome to the	World o	f the p	phyCORE	-iMX	31!	(seri	al)\	"n"	
4			•••						Þ	- C - J

• Click on the variable *buffer*.

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

## 4.5 Changing Variable Values

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.

😡= Variables 🔉 🛛 Breakpoi	ints Modules Registers Signals	- 8
	2. +ti 📄	<i>§</i> ° 💥 🦋 ▽
▶ I where a state of the st		
<pre>%<sup>X</sup> count = 52 ↔=out = -1093124428 52 </pre>	Select <u>A</u> ll Copy <u>V</u> ariables Enable Disable	Ctrl+A Ctrl+C
*/	🚔 <u>C</u> hange Value	
*/	<ul> <li>Add Global Variables</li> <li>Remove Global Variables</li> <li>Remove All Global Variables</li> </ul>	
n"}. /* outout variable	Format ≚ở Wa <u>t</u> ch ♥ ♥	•

- Select the *count* variable in the *Variables* window.
- Right-click on *count* and select *Change Value*.

Enter a new value	ofor count:		
•			••
		Or	Cancel

- 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 **two times.**
- Change to Microcom.

```
root@phyCORE:~ gdbserver 192.168.3.10:10000 myHelloWorld
Process myHelloWorld created; pid = 434
Listening on port 10000
Is
Quit
Remote debugging from host 192.168.3.10
Welcome∎
```

You will see the output *Welcome* in the Microcom window. This means that due to changing the *counter* variable's value, instead of printing the full "Welcome to the World of the phyCORE-i.MX31" string, only the first seven characters of the buffer were written to the screen.

#### 4.6 Using the Memory Monitor

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

Console Tasks 🗓 Memory 🛛	- 8
	📑 🛃 🦗 🏭 🔄 🎽
Memory Monitors 🛛 🙀 🎇 Memory Renderings	÷ 🗶
Add Memory Monitor	

- Select the *Memory* tab.
- Click Add Memory Monitor.

🗱 Variables 🕱 Breakpoints M	odules	Registers	Signals				
		A Mon	itor Mer	norv 🍭	N- +	*	~
▶ \$ <sup>#</sup> buffer = 0xbea16e48		Enter add	ress or a		a to monitor:		
¢્રેcount = 7		Oxbeal6		expression			
⊗=out = 1074304164		OVDERIO	240				
				ок	Cancel		
Oxbeal6e48 "Welcome to the W	World	of the p	yhC0RE-F	PXA270 (	ttyS0)∖n"		
•		**** **** ****				• •	

• Enter the address of *buffer* and click on *OK*. Remember that the variable's address might differ on your system.

Console Tasks 🔋 Memory 🗙		- 0
		📑 🛃 📲 🔄 🎽
Memory Monitors 🚽 🛣	Memory Renderings	+ 🗶
🕆 Oxbeal6e48	0xbeal6e48 : 0xBEAl6E48 <hex></hex>	
	Address 0 - 3 4 - 7 8 - B C - F	▲ 
	BEA16E40 C4850000 2C850000 57656C63 6F6D6520	
	BEA16E50 746F2074 68652057 6F726C64 206F6620	
	BEA16E60 74686520 70796843 4F52452D 50584132	
	BEA16E70 37302028 74747953 30290A00 00000000	
	BEA16E80 8C6EA1BE 30480340 84850000 00000000	-

• Change the window size.

	- 0
E2 🛃 🕫	1 <b>4</b>
	(+)×

• Click Add Rendering.

🖨 Add Memory Rendering 🎐	
Memory Monitor	
0xbe8a8e48 : 0xbe8a8e48	•
	Add New
Memory Rendering(s)	
Hex	
ASCII	
Signed Integer	
Unsigned Integer	
ОК	Cancel

• Select *ASCII* and click *OK*.

Console Tasks 🚺 Memory	3	- 0
		📑 🛃 🦛 🏭 🔄 🎽
Memory 🛛 🕂 🙀 🦌	Memory Renderings	+ ×
🔶 0xbeda6e48	0xbeda6e48 <hex> 0xbeda6e48 : 0xBED</hex>	A6E48 <ascii></ascii>
	Address 0 - 3 4 - 7 8 - B C - F	
	BEDA6E40 Ä , Welc ome	
	BEDA6E50 to t he W orld of	
	BEDA6E60 the pyhC ORE- PXA2	
	BEDA6E70 70 ( ttyS <sup>0)</sup>	•

You can see the contents of the variable *buffer* at the address *0xbeda6e48* (or whatever address is used on your system).

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

<u>F</u>	] *my	HelloWorld.c 🛿			- 8
	}	<pre>out=open("/dev/ttymxc0",0_RDWR); write(out,buffer,count); close(out);</pre>	/* open interface /* write n bytes /* close the serial interface	*/ */ */	
	int	main()			
	i	<pre>char buf[]={"Welcome to the World")</pre>	d of the phyCORE-iMX31! (serial)\n"	}; /* output variable	*/
		<pre>write_tty(buf,sizeof(buf)-1); printf("Welcome to the World of "</pre>	the phyCORE-i.MX31!\n");	/* write buffer to tty /* write to stdout	*/
0		return 0;			
	}				-

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

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



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

# **5** Further Information

In the PTXdist User Manual you can find further information. The manual is located on your setup CD-ROM in the directory:

PHYTEC/PCM037 phyCORE-i.MX31/Linux-Kit/BSP/OSELAS

The PTXdist User Manual includes information on the following topics:

- Installation and Configuration of PTXdist.
- Building and using a tool chain.
- Create and activate a PTXdist project.
- Running the phyCORE-i.MX31 from network only.
- Running the phyCORE-i.MX31 stand-alone.
- U-Boot on the phyCORE-i.MX31.
- The phyCORE-i.MX31's Board Support Package (BSP).
- Using CAN on the phyCORE-i.MX31.

# 6 Summary

This QuickStart manual provided a general "Rapid Development Kit" description, as well as software installation advice and an example program enabling quick out-of-the-box start-up of the phyCORE-i.MX31 in conjunction with the Eclipse IDE and GNU C/C++ software tools.

In the *Getting Started* section you learned how to configure your host to provide a basis for working with your target platform. You installed the Rapid Development Kit software and learned how to copy and run a program on the target.

In the *Getting More Involved* section you got step-by-step instructions on how to configure and build a new kernel, modify the example application, create and build new projects, and copy programs to your phyCOREi.MX31 using Eclipse.

The *Debugging* part of this QuickStart gave you information on setting up and using the GNU debugger with the Eclipse IDE. You learned how to set breakpoints, watching and changing variable contents, and using the memory monitor.

# 7 Installing Linux on the phyCORE-i.MX31

This part provides instructions on how to install the boot loader (*U-Boot*) on the phyCORE-i.MX31 and how to write a kernel and/or a root file system image into the target's flash memory.

# 7.1 Installing the Boot Loader

The boot loader used on the phyCORE-i.MX31 is *U-Boot*, the Universal Boot Loader. The installation of the boot loader will be performed by the *AdvancedToolKit* provided by Freescale. The AdvancedToolKit must be executed on a PC running *Microsoft Windows*.

- Insert your PHYTEC Linux-i.MX31-Disc, navigate to the *PHYTEC\PCM037 phyCORE-i.MX31\Linux-Kit\Software\Tools\ ADS\_Toolkit\FSL\_ATK\_TOOL\_WINS\_STD\_INSTALL\_1\_3* directory, and execute *Setup.exe* to install the AdvancedToolKit.
- There are two additional files in the *PHYTEC\PCM037 phyCOREi.MX31\Linux-Kit\Software\Tools\ADS\_Toolkit* directory: One ending with .*cfg*, and the other one ending with .*bin*.
  - Copy the file ending with *.cfg* to the *C:\Program Files\Freescale\AdvancedToolKit-STD/Config* directory on your hard disk.
  - Copy the file ending with *.bin* to the *C:\Program Files\Freescale\AdvancedToolKit-STD/Image* directory on your hard disk.
- Configure the switch S5 to boot with UART (5,3 = on, all other = off):



Boot Mode/CLK-Sel.

• Connect the UART1 (connector P1, TOP) to your computer.



- Power up the baseboard.
- Start the program *AdvancedToolKit*.

The AdvancedToolKit dialog opens.

- Choose *i*.*MX31 TO2* from the *i*.*MX CPU* drop-down box.
- Select the Serial Port which the phyCORE-i.MX31 is connected to.
- Click Next.

Advanced ToolKit Config	uration Version: 1.60	
	Device Setting i.MX CPU: i.MX31 T02 Device memory initial ODR2 OMDDR Custom Initial File Host Setting Communication Channel Serial Port: COM1 USB Bypass the settings	
	< Back Next > Cancel	Go

- In the next window, select *Flash Tool*.
- Click *Go* to run the selected tool.

Select Advanced toolkits	
	Welcome to Advanced ToolKit Collection Please select the toolkit you want to use, then click Go
	Flash I 001
	Image Format Convert Tool
J.	
	< Back Next> Cancel Go

- Select *Erase* in the *Operation type* section.
- Choose *NOR\_INTEL\_CFI* from the *Flash model* drop-down box.
- In the *Operating settings* section, set *Address* to **a0000000** and *Size* to **40000**. A size of 40000 will delete the U-Boot flash partition only. If you also want to delete the U-Boot environment partition to start over with the default U-Boot environment, set *Size* to **60000**.
- Click *Erase*.

🎽 Advanced Toolkit (Flash Tool)	
Operation type       Flash Device         C Download       C Dump       BI Swar       BBT         C Program       Erase       Unboundar	_CFI  _ Browse
Operation settings           Address:         0x           Size:         0x	
Image Image File (Download/Program input, Dump output)	
J Process:	Browse
	>
Silicon Security Type (HAB_TYPE): N/A	
< Quit Exit Load Save	Erase Stop

- When finished, select *Program* in the *Operation type* section.
- Click the *Browse* button in the *Image* section to select the U-Boot image you want to write to the target. Navigate to the directory \*PHYTEC\PCM037 phyCORE-i.MX31\Linux-Kit\BSP\U-Boot\bin* on your setup CD-ROM and choose the *u-boot-v2-image* file you find there. In the *File filter* drop down box, you may have to choose *all files* (\*.\*) to list the file.
- Now click *Program*.

Advanced Toolkit (Flash Tool)	
Operation type       Flash Device         Operation type       BI Swap         Program       Erase         Unboundar,	▼ Browse
Operation settings Address: 0x a0000000 Size: 0x 40000	
Image Image File (Download/Program input, Dump output) \u-boot-v2-image	Browse
Finished!	
Flash erase successful! Flash program successful!	
	<b>~</b>
	2
Silicon Security Type (HAB_TYPE): Development/Disable	_
< Quit Exit Load Save Program	Stop

- When finished, power down the baseboard.
- Configure the switch S5 to boot with NOR Flash (all switches = off):



Boot Mode/CLK-Sel.

# 7.2 Configure U-Boot Environment Variables

The following steps will be done on a Linux platform. We assume that you have configured your host platform and have executed the setup program provided on the Linux-i.MX31-Disc. Information on how to configure the host platform can be found in the *Getting Started* part of this QuickStart.

- Connect the serial cable with the UART1 (connector P1, TOP) on the target and the first serial interface on your host.
- Connect the cross-over Ethernet cable with the connector X27 on the target and the right network card of your host.



• Click the *Microcom* icon on your desktop

Microcom was configured during the setup with the following configuration:

115200 baud, 1 start bit, 8 data bits, 1 stop bit, no parity, no flow control.

If you want to use a program other than Microcom for serial communication, you will have to setup that program with these settings.

• Connect the AC adapter with the power supply connector X26 (5V) on your board.

. .

#### 🜉 Microcom - Konsole 🧐

```
U-Boot 2.0.0-rc9 (Jul 28 2009 - 10:50:08)

Board: Phytec phyCORE-i.MX31

NAND device: Manufacturer ID: 0×20, Chip ID: 0×76 (ST Micro NAND 64MiB 3,3V 8-

t)

Scanning device for bad blocks

RCSR_BTP: 20

Using environment in NOR Flash

got MAC address from EEPRDM: 00:50:C2:95:4E:08

mx31 cpu clock: 513MHz

ing clock : 64133247Hz

Malloc space: 0×87b00000 -> 0×87f00000 (size 4 MB)

Stack space: 0×87af8000 -> 0×87f00000 (size 32 kB)

envfs: wrong magic on /dev/env0

no valid environment found on /dev/env0. Using default environment

running /env/bin/init...

Hit any key to stop autoboot: 3

type update_kernel nand|nor [<imagename>] to update kernel into flash

type update_rootfs nand|nor [<imagename>] to update rootfs into flash

uboot:/ ■
```

• Press any key to stop autoboot.

If no configuration can be found in the flash, the built-in default configuration will be used and you will see the following messages:

```
envfs: wrong magic number on /dev/env0
no valid environment found on /dev/env0
using default environment
running /env/bin/init...
```

You will be asked to enter the MAC address of your board. Please enter the MAC address found on your module, for example:

#### 00:50:c2:52:9e:3d

The MAC address can also be changed by editing the configuration file.

• Type edit /env/config to edit the configuration file.

```
📕 Microcom - Konsole
                                                                       _ =
#!/bin/sh
                                                                            *
 use 'dhcp' to do dhcp in uboot and in kernel
#
#ip=dhcp
# or set your networking parameters here
eth0.ipaddr=192.168.3.11
eth0.netmask=255.255.255.0
eth0.gateway=192.168.3.10
eth0.serverip=192.168.3.10
# can be either 'net', 'nor' or 'nand''
kernel_loc=nor
rootfs_loc=nor
kernel=uImage-pcm037
rootfs=root-pcm037.jffs2
envimage=u-boot-v2-environment-pcm037
autoboot_timeout=3
nfsroot=192.168.3.10/root
bootargs="console=ttymxc0,115200"
#bootargs="$bootargs video=mx3fb:TX090"
```

You will see the configuration file that holds U-Boot's environment variables. The MAC address can be set by adding or editing a line similar to the following one:

eth0.ethaddr=00:50:c2:99:83:ee

(This is an example; be sure to enter the MAC address of your board.)

The default IP address of the target is 192.168.3.11, and the default server IP address is 192.168.3.10. If you want to set up a different network configuration, you can also edit the following lines of the configuration file:

```
eth0.ipaddr=target IP address
eth0.netmask=target netmask
eth0.serverip=server IP address
```

- Type *CTRL-D* to save the settings to the file.
- Type **save** to write the settings to the target's flash.

• Press the RESET button on your board. The target will restart with the new settings applied.

## 7.3 Restoring the U-Boot Default Configuration

If you want to restore the default U-Boot configuration, you can use the following commands to delete the U-Boot environment partition:

- unprotect /dev/nor0.ubootenv
- erase /dev/nor0.ubootenv

After pressing the RESET button on your board, the default U-Boot configuration will be used. This also means that you will be asked to enter the MAC address of your board again.

# 7.4 Writing the Kernel / Root File System into Flash

Before the kernel and/or root file system can be written into the target's flash, the target will have to download the image from a TFTP server. This will be done from the command line of the boot loader. First the image will be downloaded via TFTP and copied into target's RAM. Then the part of the flash where the kernel or root file system should be written to must be erased. Finally the kernel or root file system image can be transferred from RAM into flash.

In the directory *PHYTEC/PCM037* phyCORE-i.MX31/Linux-*Kit/BSP/Images* on your setup CD-ROM you can find a file called *uImagepcm037* – this file is the Linux kernel image. There is another file, root*pcm037.jffs2;* this file contains the Linux root file system.

• Copy the files *uImage-pcm037* and *root-pcm037.jffs2* to your host's */tftpboot* directory.

You can download the kernel or root file system from the TFTP server into the target's RAM, erase the corresponding flash partition, and write the kernel from RAM into flash with one simple command, *update\_kernel* or *update rootfs*, respectively.

Before executing these commands, you should check that your U-Boot environment is properly configured.

• Open Microcom and press the RESET button on the target.

You will see the message "Hit any key to stop autoboot."

- Press any key to stop autoboot.
- Type the following command to check your U-Boot settings:

#### edit /env/config

You will see the configuration file which holds U-Boot's environment variables.

• Make sure the following values are set within the configuration file:

eth0.ipaddr=target IP address eth0.netmask=target netmask eth0.serverip=server IP address

- Type *CTRL-D* to save the settings to the file.
- If you have made any changes to the U-Boot environment, type **save** to write these changes to the target's flash. Then press the RESET button on the target. The board reboots with the new settings applied. Then, again, press any key to stop autoboot.
- Type **update\_kernel nor uImage-pcm037** to download and write the kernel into the target's flash.
- Type **update\_rootfs nor root-pcm037.jffs2** to download and write the root file system into the target's flash.

The copy process can take quite a while, depending on the speed of your system.

• Press the RESET button on the board to restart your target.

The target should now start with the kernel and root file system you have written into the flash.

Document:	phyCORE-i.MX31 with Linux QuickStart Instructions	
Document Number:	L-703e_4 August 2009	
How would you im	prove this manual?	
Did you find any n	nistakes in this manual?	page
Submitted by:		
Customer number:		
Name:		
Company:		
Address:		
Return to:		
	PHYTEC Messtechnik GmbH	
	Robert-Koch-Str. 39	
	D-55129 Mainz	
	Fax:+49(6131)9221-26	