

Quick Start Instructions

Linux-Kit phyCARD-S

Using Eclipse and the GNU Cross Development Tool Chain

Note: The PHYTEC Linux-phyCARD-S-Disc includes the electronic
version
of the English phyCARD-S Hardware Manual.

2nd Edition: October 2009

A product of a PHYTEC Technology Holding company

In this manual copyrighted products are not explicitly indicated. The absence of the trademark (™) 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 Robert-Koch-Str. 39 55129 Mainz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Ordering Information:	+49 (800) 0749832 order@phytec.de	1 (800) 278-9913 sales@phytec.com
Technical Support:	+49 (6131) 9221-31 support@phytec.de	1 (800) 278-9913 support@phytec.com
Fax:	+49 (6131) 9221-33	1 (206) 780-9135
Web Site:	http://www.phytec.de	http://www.phytec.com

2nd Edition: October 2009

Chapter 1	Introduction.....	1	
1.1	Rapid Development Kit Documentation	1	
1.2	Professional Support Packages Available	2	
1.3	Overview of this Quick Start Manual.....	2	
1.4	Conventions Used in this Quick Start Manual	3	
1.5	System Requirements	4	
1.6	Software Development Tool Chains	4	
1.6.1	Eclipse.....	4	
1.6.2	The GNU Cross Development Tool Chain.....	5	
Chapter 2	Getting Started.....	7	
2.1	Requirements of the Host Platform	7	
2.2	Configuring the Host Platform	8	
2.2.1	Installing Software Packages	8	
2.2.2	Set Up Network Card Configuration	14	
2.2.3	Disabling the Firewall.....	15	
2.2.4	Set Up TFTP Server.....	16	
2.3	Linux-phyCARD-S-Kit Setup.....	18	
2.3.1	Starting the Setup.....	19	
2.4	Advanced Configuration Information	28	
2.5	Connecting the Host with the Target.....	29	
2.6	Copying an Example to the Target.....	33	
2.6.1	Copying a Program to the Target.....	34	
2.6.2	Using Telnet to Execute a Program on the Target.....	35	
2.6.3	Using SSH to Execute a Program on the Target.....	37	
2.7	Advanced Information.....	40	
2.7.1	Copying a Program to the Target with the Command..... Line	40	
2.7.2	Executing a Program on the Target	40	
2.7.3	Executing a Program directly on the Target using SSH.....	40	
Chapter 3	Getting More Involved	41	
3.1	Configuring and Compiling the Kernel.....	41	
			70 min

3.1.1	Writing the Kernel into the Target's Flash.....	44
3.2	Opening an Existing Project.....	48
3.2.1	Copying the <i>HelloWorld</i> Project	48
3.2.2	Starting Eclipse and Importing the Example Project ...	48
3.3	Creating a New Project.....	53
3.4	Changing the Demo Application.....	63
3.4.1	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 Target	68
Chapter 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	81
4.4	Stepping and Watching Variable Contents	81
4.5	Changing Variable Values	83
4.6	Using the Memory Monitor.....	85
Chapter 5	Summary.....	88
Chapter 6	Installing Linux on the phyCARD-S.....	89
6.1	Installing the Boot Loader.....	89
6.2	Configure U-Boot Environment Variables	93
6.3	Restoring the U-Boot Default Configuration.....	96
6.4	Writing the Kernel / Root File System into Flash.....	96



TIME

35 min

Chapter 1 Introduction



5 min

In this Quick Start you can find general information on the PHYTEC phyCARD-S, and 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 phyCARD-S, mounted on the PHYTEC Development Board, in conjunction with the Eclipse development tool.

Please refer to the phyCARD-S 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-phyCARD-S-Disc:”

- The PHYTEC phyCARD-S Hardware Manual.
- phyCARD-S Controller [User's Manuals and Data Sheets](#).
- This Quick Start Instructions with general “Rapid Development Kit” description, software installation advice, and an example program enabling quick out-of-the-box start-up of the phyCARD-S 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 Quick Start Manual

This Quick Start manual 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 phyCARD-S in conjunction with the Eclipse IDE and GCC C/C++ software tools. It is structured as follows:

- The “*Getting Started*” chapter describes the configuration of the host platform and how to setup all the tools used in this manual.
- 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 phyCARD-S using Eclipse.
- 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-phyCARD-S-Disc contains supplemental information on embedded microcontroller design and development in general.

1.4 Conventions Used in this Quick Start Manual

The following is a list of the typographical conventions used in this Quick Start manual:

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:



OPTION

At this part you might leave the path of this Quick Start.



CAUTION

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



TIP

Provides useful supplementary information about the topic.



TIME

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



SUCCESS

You have successfully passed an important part of this Quick Start manual.



RESOLVE

Provides information to solve common problems.

1.5 System Requirements

Use of this “Rapid Development Kit” requires:

- The PHYTEC phyCARD-S.(i.MX 27)
- The PHYTEC Development Board with the included DB-9 serial cable, Ethernet cross-over cable and AC adapter supplying 12 VDC (min. 2 A).
- PHYTEC Linux distribution based on OSELAS from Pengutronix.
- An IBM-compatible host PC (586 or higher CPU).
- 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:



<http://www.phytec.de>

support@phytec.de

1.6 Software Development Tool Chains

1.6.1 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 Quick Start shows you how to make use of the *CDT*, a set of plug-ins for C/C++ development in conjunction with the GCC C/C++ tool chain.

The CDT is an open source project (licensed under the Common Public License) implemented purely in Java as a set of plug-ins for the Eclipse SDK platform. These plug-ins add a C/C++ perspective to the Eclipse

Workbench that can now support C/C++ development with a number of views and wizards, along with advanced editing and debugging support.

Due to its complexity, the CDT is broken down into several components, which take the form of separate plug-ins. Each component operates as an autonomous project, with its own set of committers, bug categories, and mailing lists. However, all plug-ins are required for the CDT to work properly. Here is a list of the plug-ins/components:

- **Primary CDT plug-in** is the “framework” for the CDT plug-ins.
- **CDT Feature Eclipse** is the CDT Feature Component.
- **CDT Core** provides Core Model, CDOM, and Core Components.
- **CDT UI** is the Core UI, views, editors, and wizards.
- **CDT Launch** provides the launch mechanism for external tools such as the compiler and debugger.
- **CDT Debug Core** provides debugging functions.
- **CDT Debug UI** provides the user interface for the CDT debugging editors, views, and wizards.
- **CDT Debug MI** is the application connector for MI-compatible debuggers.

1.6.2 The GNU Cross Development Tool Chain

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

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

called the *target platform*. In this Quick Start we are using an x86-based Linux as the host platform. As the target platform we are using the ARM architecture with an phyCARD-S 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 tool chain and cross-compiled libraries. In this Quick Start we are using the GNU C/C++ cross development tool chain.

Chapter 2 Getting Started

**35 min**

TIME

In this chapter you will establish the basis to pass the steps in this Quick Start. 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 Quick Start, 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 these Quick Start 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 Quick Start 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:

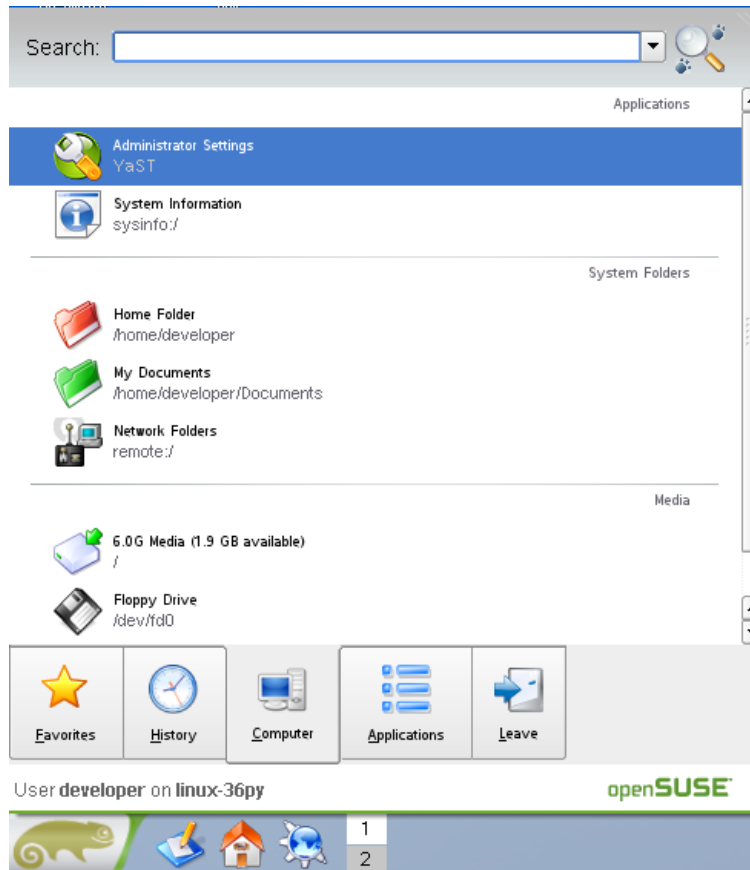
- Install additional software packages. These packages are necessary to accomplish the steps in the Quick Start Instructions.
- Set up the network configuration to use the host PC with your target.
- Disable the firewall. If the firewall is enabled, you will have problems with connecting to the target.
- Set 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.

2.2.1 Installing Software Packages

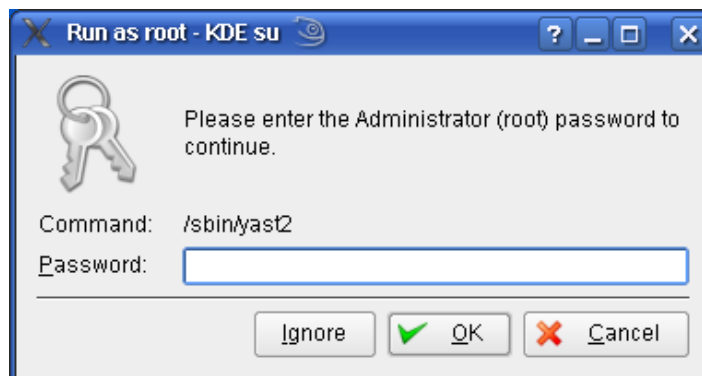
To accomplish the steps in the Quick Start Instructions, you will have to install additional packages.



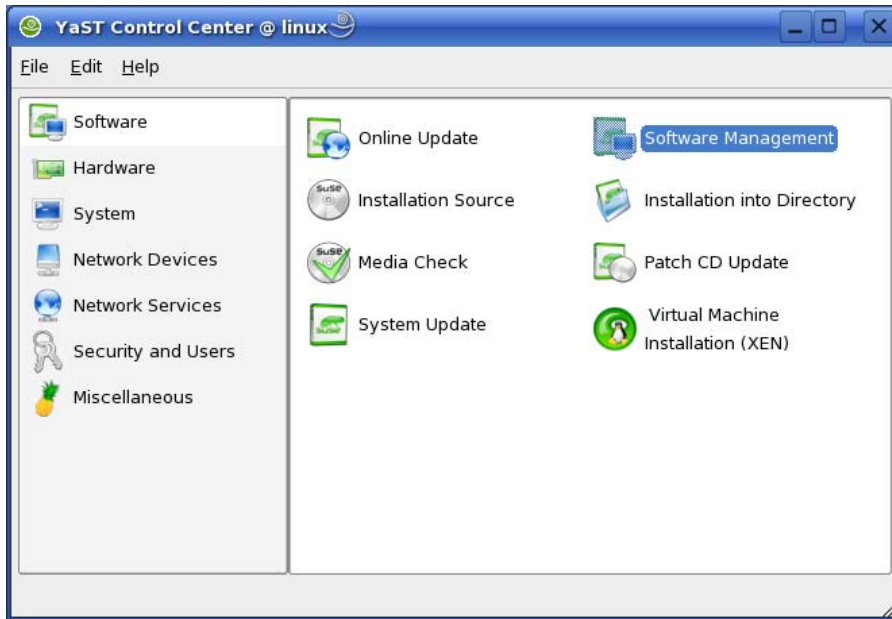
If you don't install all of these packages, the setup may fail or some configuration steps won't work correctly.



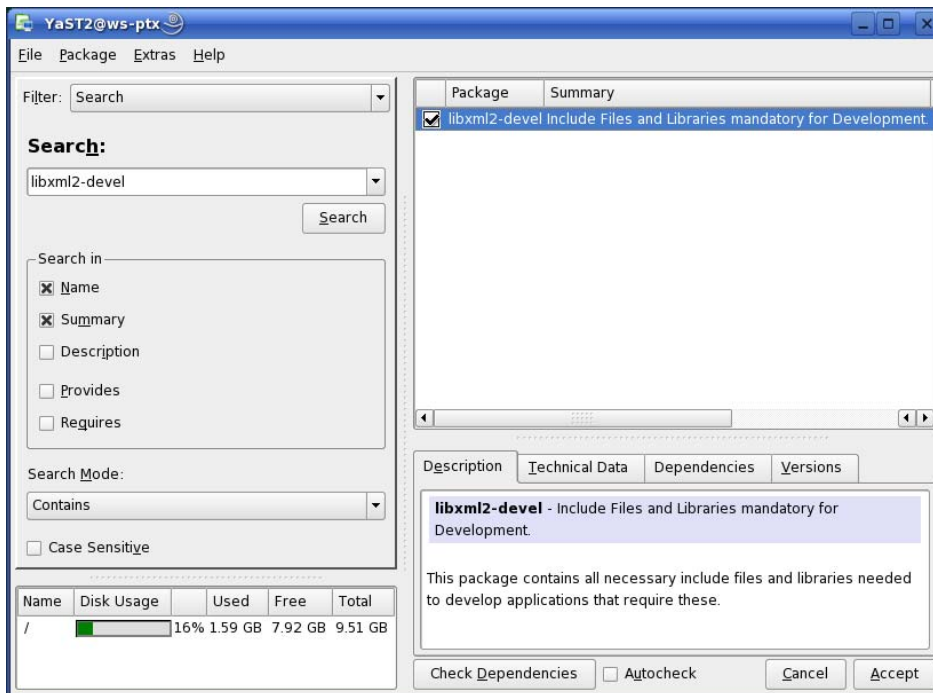
- Open the *K menu* from the lower-left corner of the desktop and click on the tab *Computer*.
- Open the *Administrator Settings / YaST*.



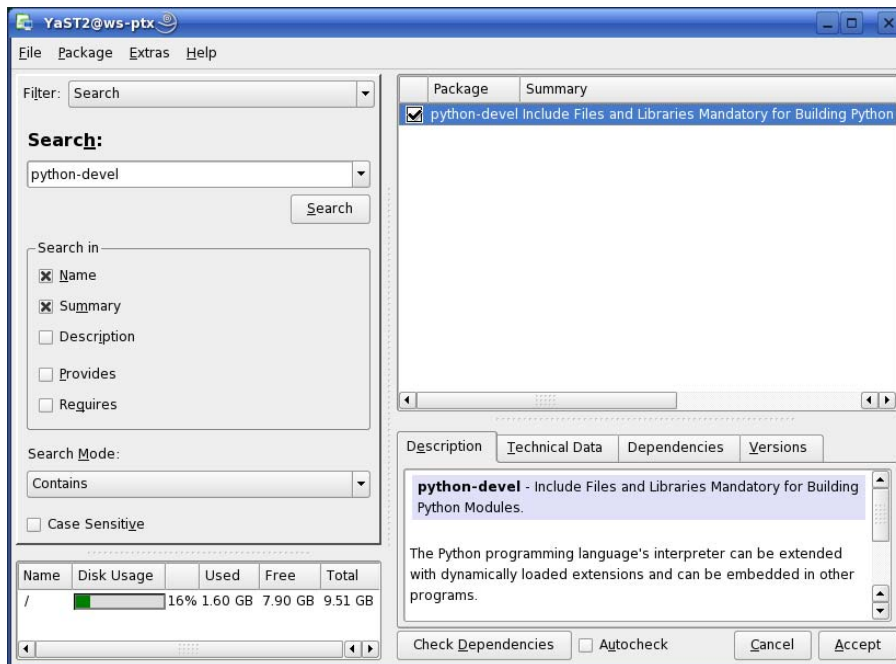
- Enter your root password and click *OK*.



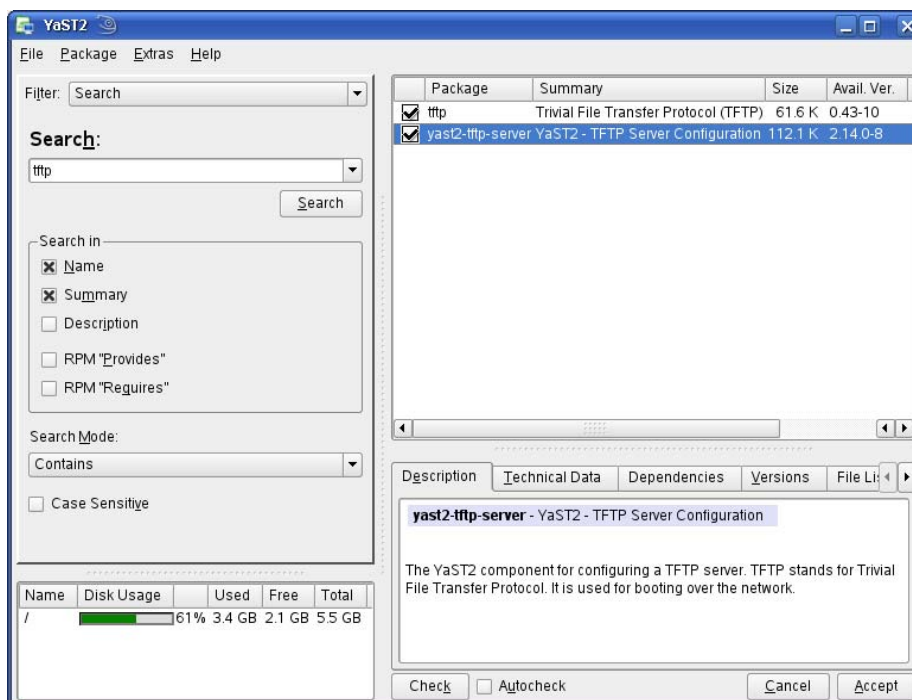
- Open *Software Management* in *Software*.



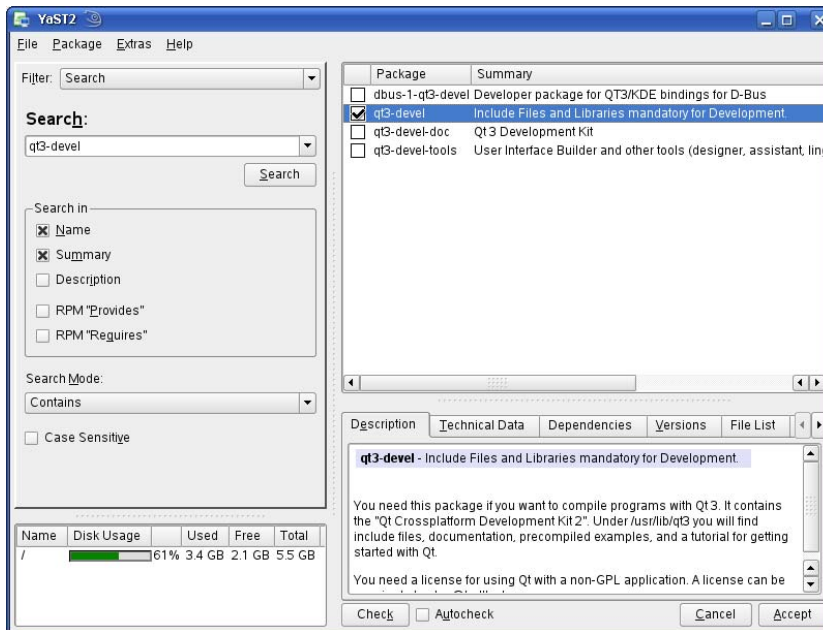
- Select the filter *Search*.
- Type **libxml2-devel** and click the *Search* button.
- Check *libxml2-devel*.



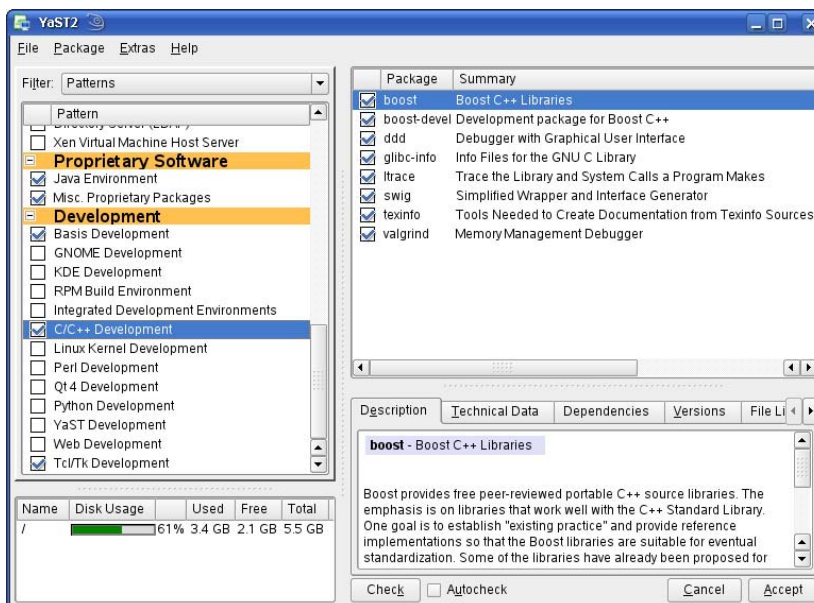
- Type **python-devel** and click the *Search* button.
- Check *python-devel*.



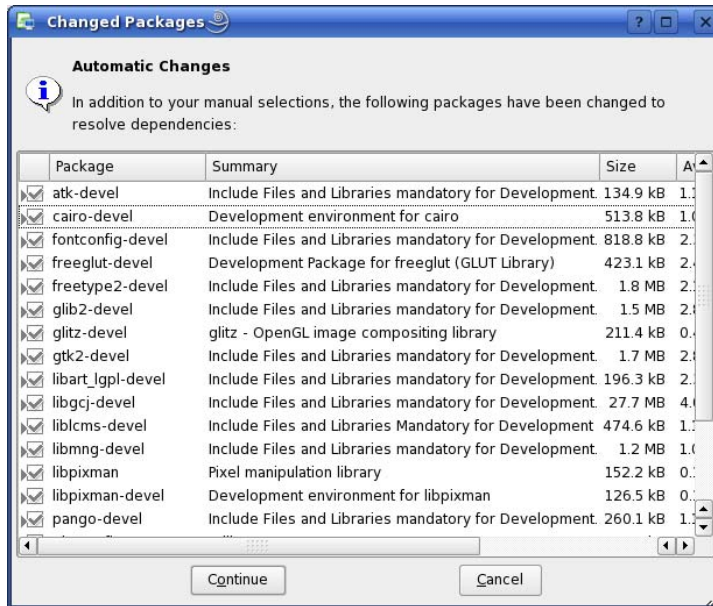
- Type **tftp** and click the *Search* button.
- Check the packages *tftp* and *yast2-tftp-server*.



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



- Select the filter *Patterns*.
- Select *Basis Development*, *C/C++ Development*, and *Tcl/Tk Development*.
- Click *Accept*.

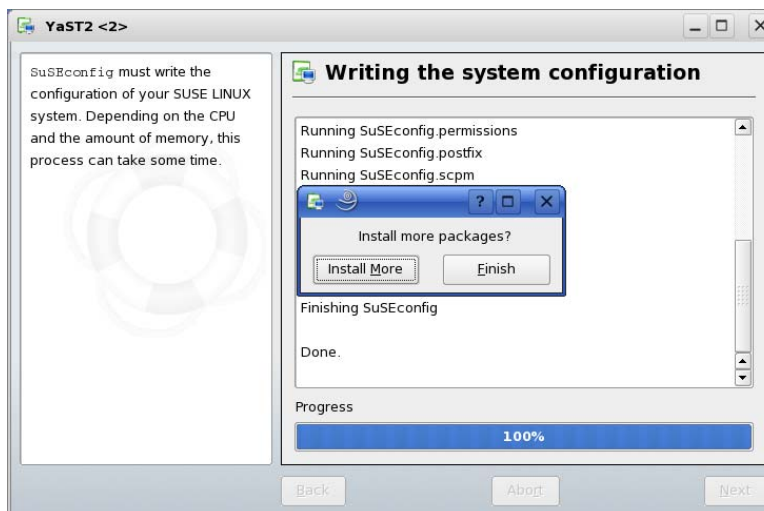


Some additional packages will be selected automatically to resolve any dependencies.



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

- Click *Continue* to install the packages.



- Click *Finish*.

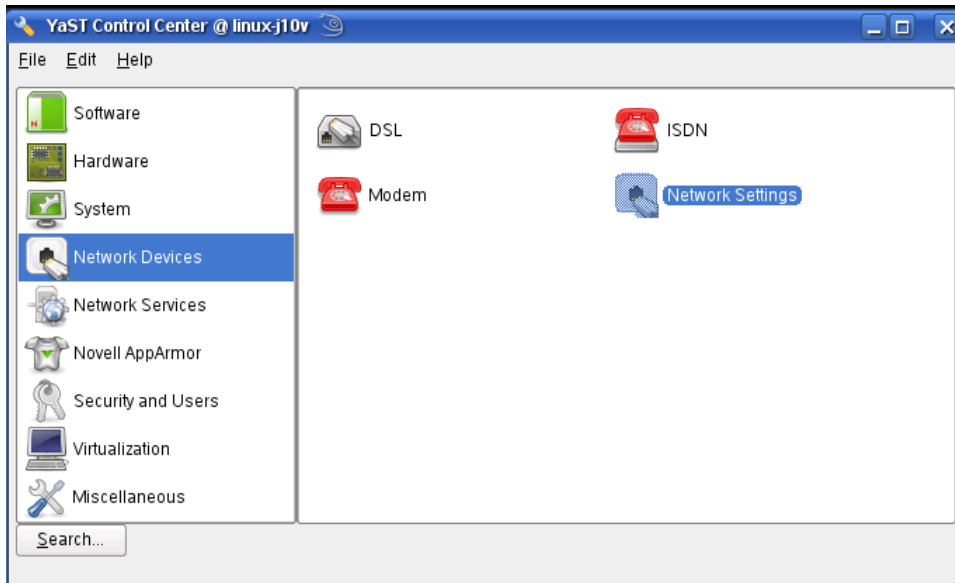
2.2.2 Set Up Network Card Configuration



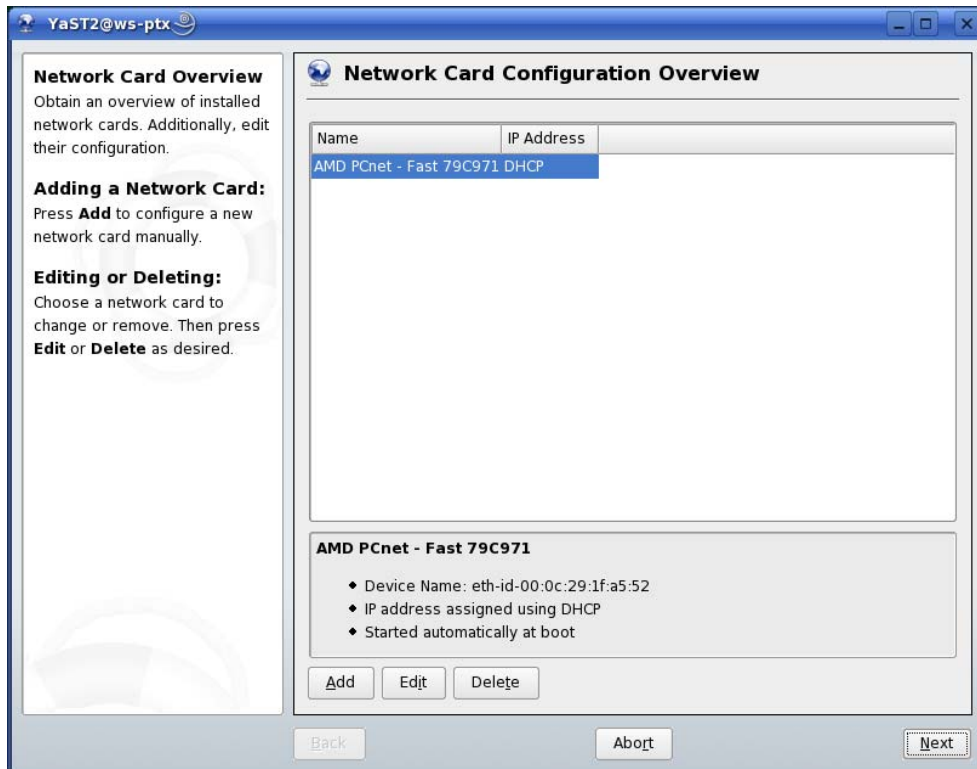
CAUTION

In the following steps you will have to configure the IP address of your host. We recommend disconnecting your host 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*.

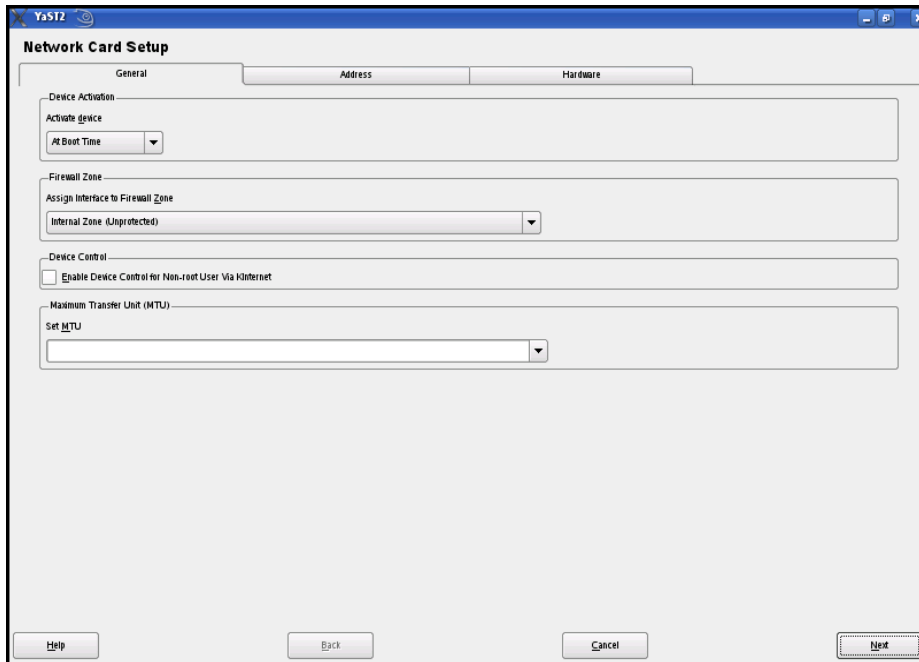


- Select the right network card (if more than one network card is installed on your host).
- Click *Edit* to enter the *Network Card Setup*.
- Choose *Static address setup*.
- Enter IP address **192.168.3.10** and subnet mask **255.255.255.0**

2.2.3 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.



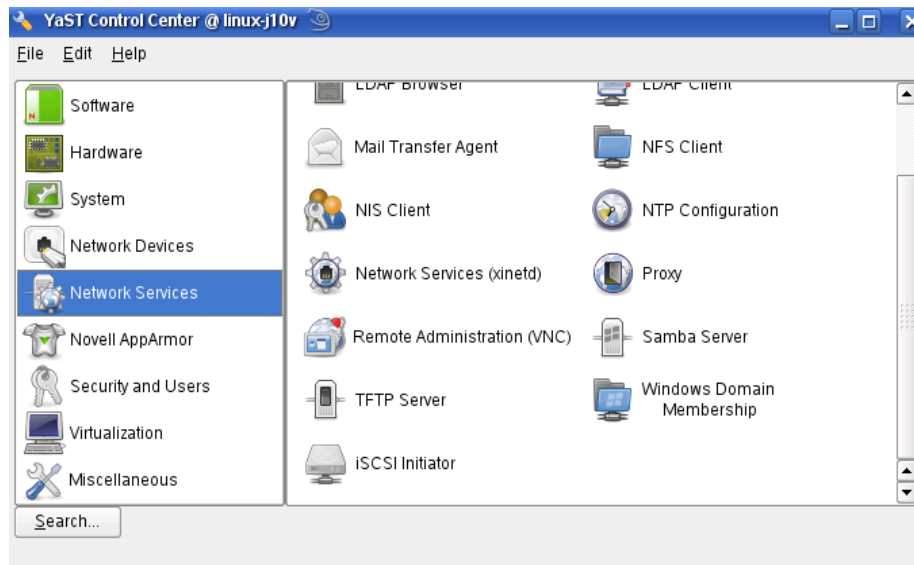
- Use the drop-down box in the *Firewall Zone* settings to set the current interface to *Internal Zone (Unprotected)*.
- Then press *Next*, and in the following window click *Finish* to complete the settings.

The firewall is now disabled for this network card.

2.2.4 Set Up TFTP Server

Later in this Quick Start 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.

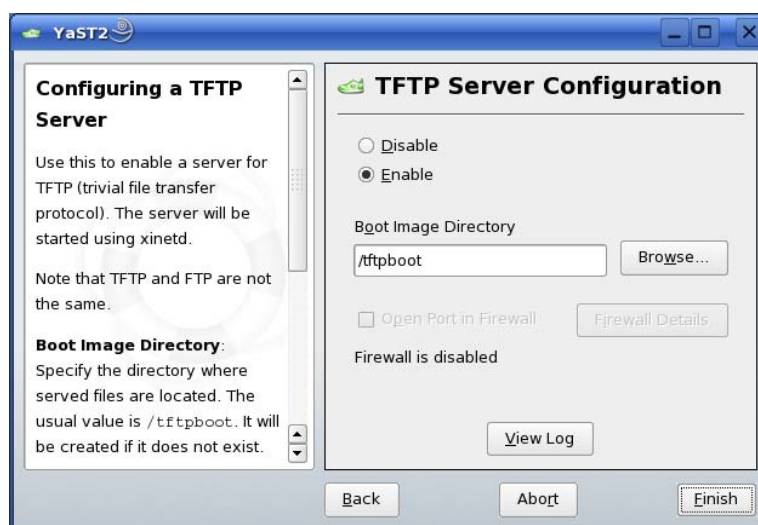
- Open the *YaST Control Center* if it is not already opened.



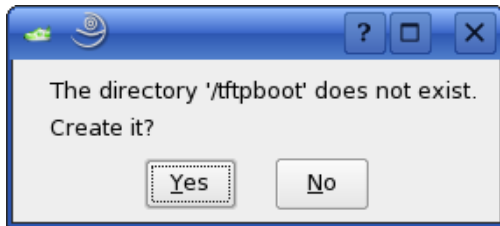
- Choose *TFTP Server* in *Network Services*.



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



- 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-phyCARD-S-Kit Setup

In this section you will find a description of the Linux-phyCARD-S-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 phyCARD-S.

- *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.

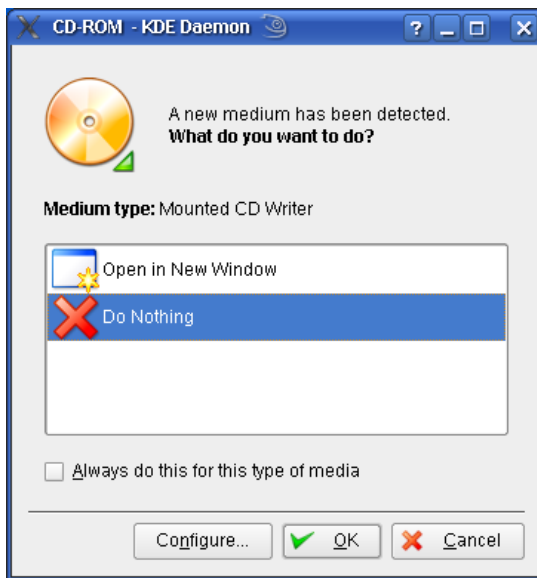
There will be some additional configuration steps performed on your PC:

- 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.

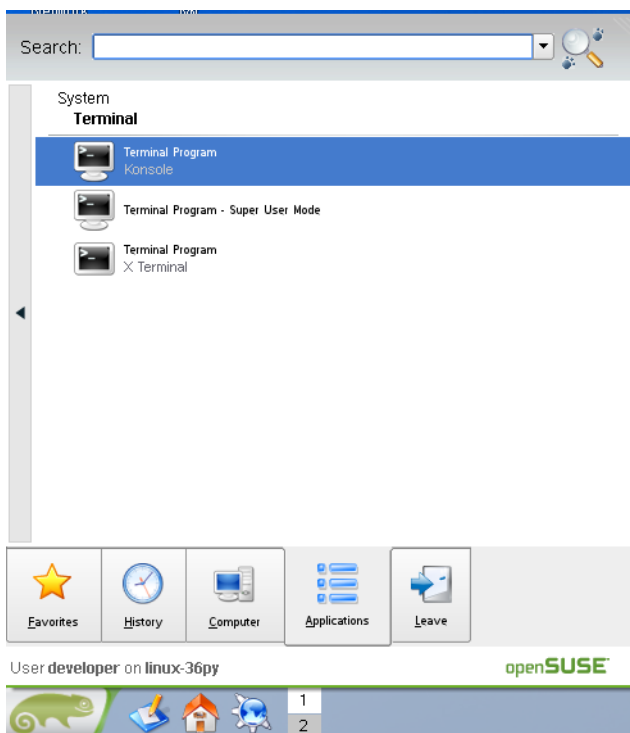
2.3.1 Starting the Setup

- To start with the Linux-phyCARD-S-Kit setup, put your PHYTEC Linux-phyCARD-S-Disc into your CD-ROM drive.

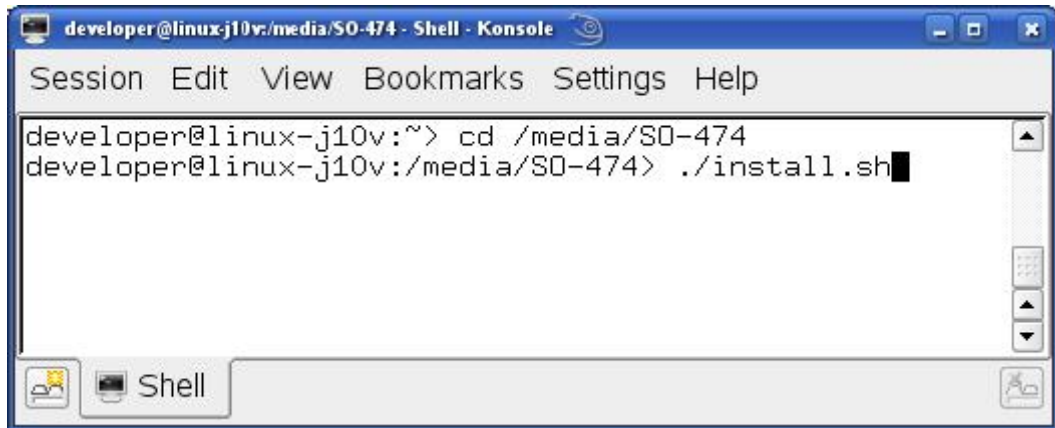
The following dialog may appear:



- Click *Cancel*.



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

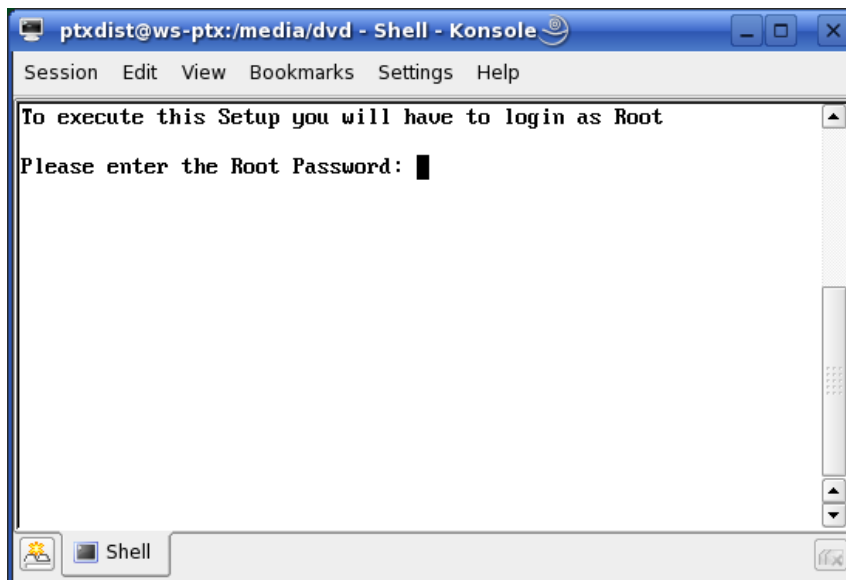


```
developer@linux-j10v:/media/SO-474 - Shell - Konsole
Session Edit View Bookmarks Settings Help
developer@linux-j10v:~> cd /media/SO-474
developer@linux-j10v:/media/SO-474> ./install.sh
```

- Type: **cd /media/SO-474**
- Enter **./install.sh** to launch the setup program.

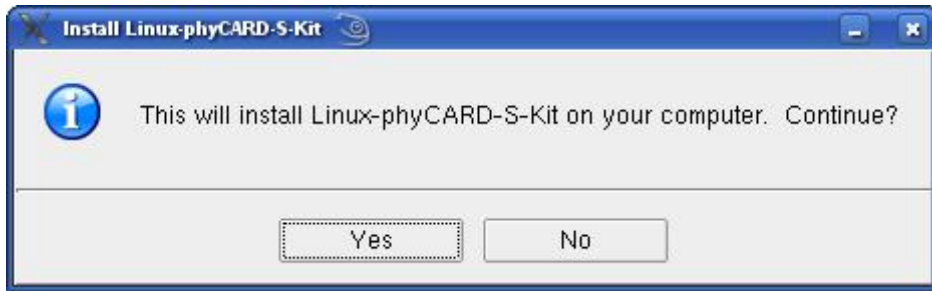


The media may be mounted on a different mount point in the directory */media*. The mount points can be shown with the command **ls /media**. Change to the accordant directory if no directory *SO-4746* should exist.



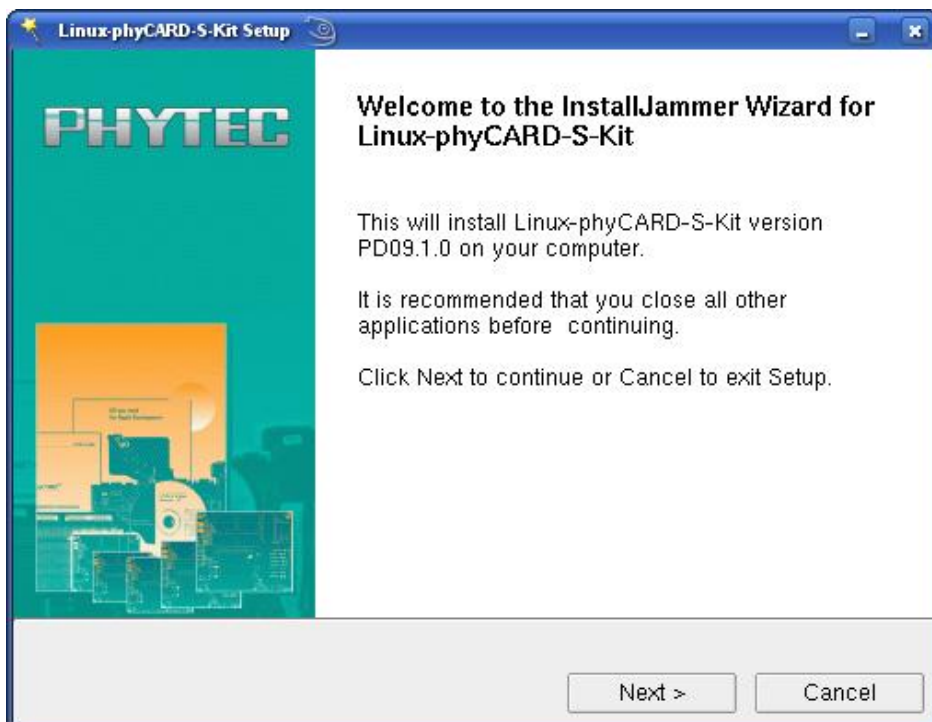
```
ptxdist@ws-ptx:/media/dvd - Shell - Konsole
Session Edit View Bookmarks Settings Help
To execute this Setup you will have to login as Root
Please enter the Root Password: █
```

- Enter the root password.

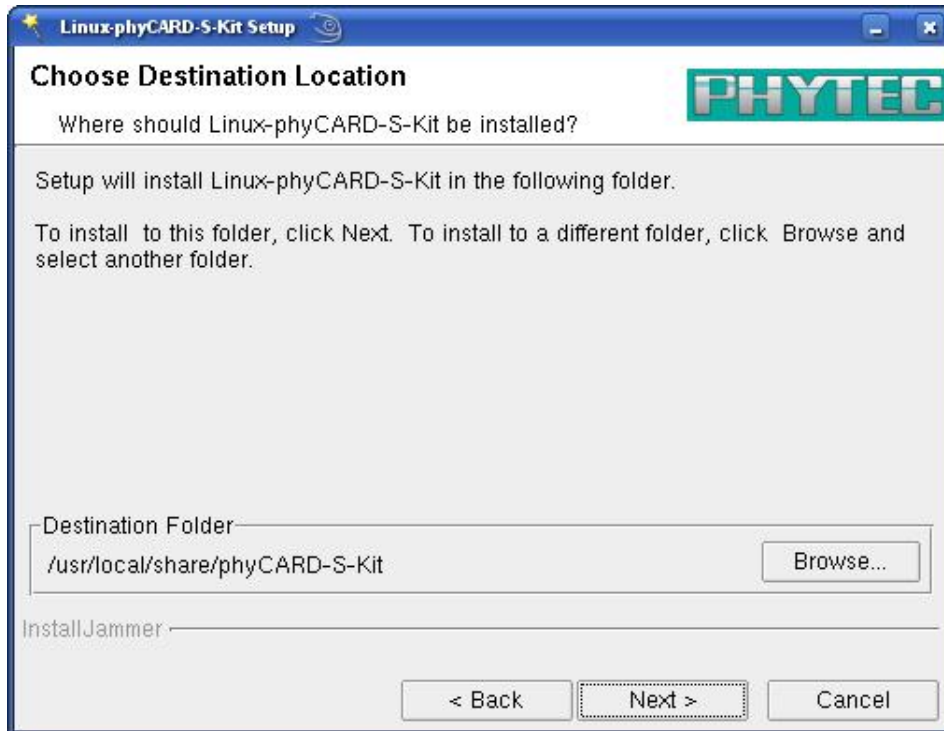


- Click *Yes* to proceed.

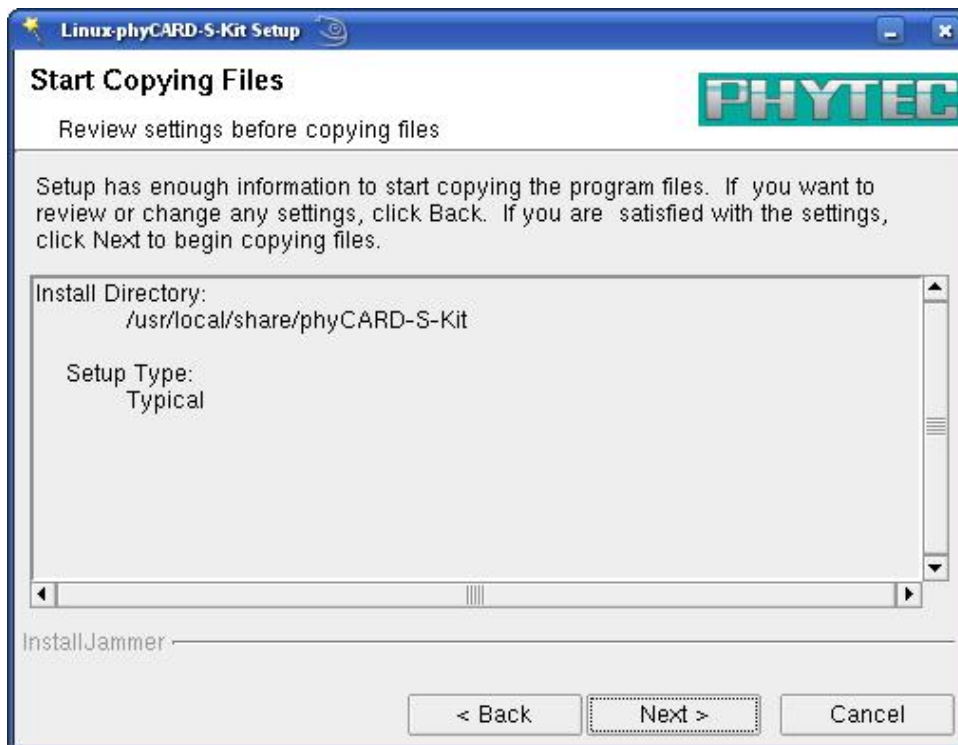
The welcome screen appears.



- Click *Next* to continue.



- Click *Next*.

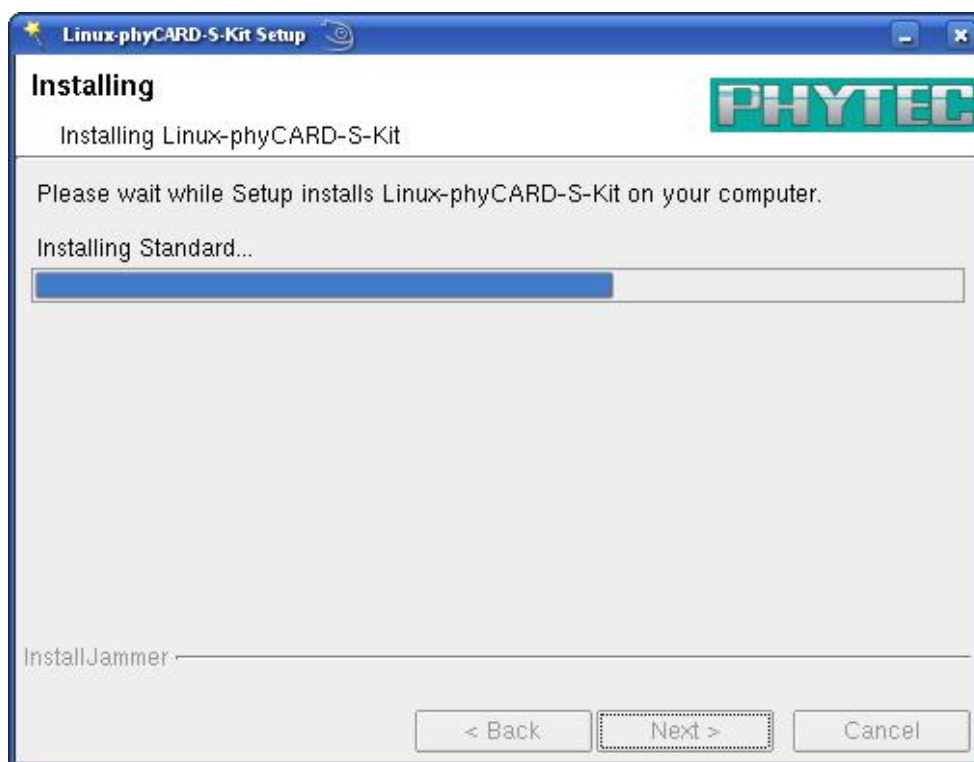


- Click *Next* to copy all files to your hard disk.



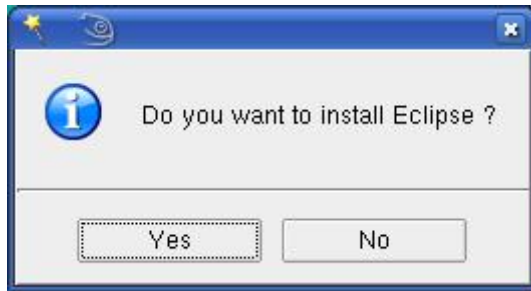
The default destination location is `/usr/local/share/phyCARD-S-Kit`. All path and file statements within this Quick Start manual are based on the assumption that you accept the default installation paths. If you decide to individually choose different paths, you must consider this for all further file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

We strongly recommend accepting the default destination location.



The GCC C/C++ tool chain will be installed to the default directory `/opt/OSELAS.Toolchain-1.99.3/arm-v5te-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.

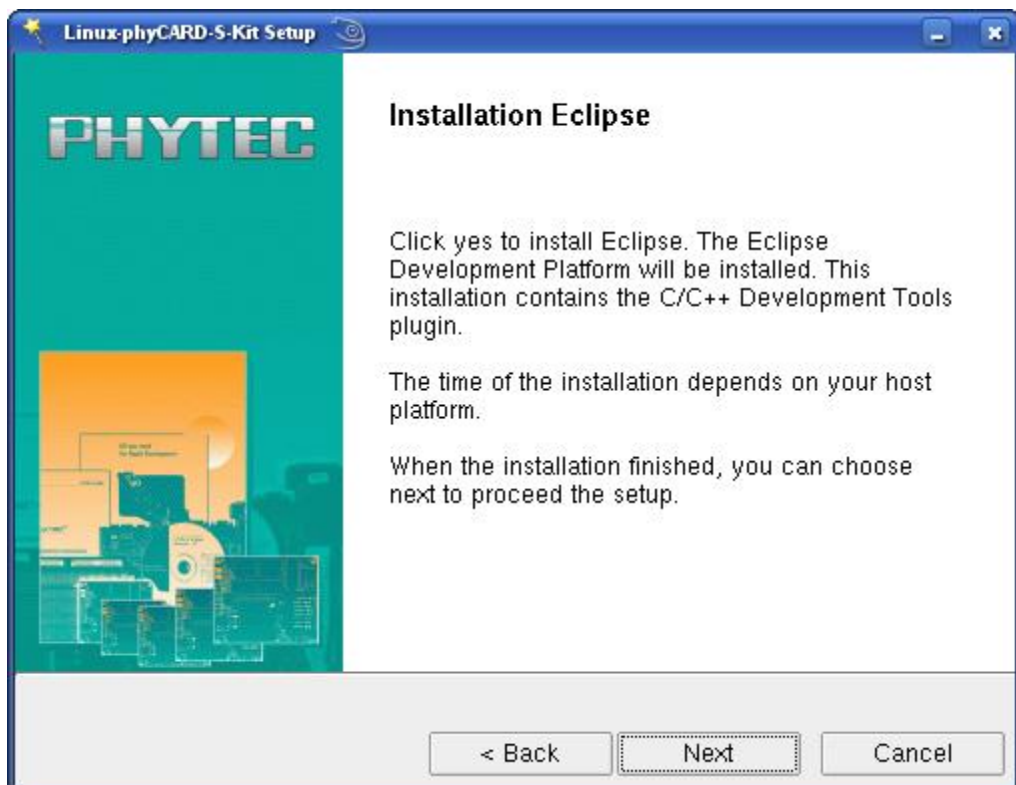
After the files have been copied, a dialog box for the Eclipse installation will appear.



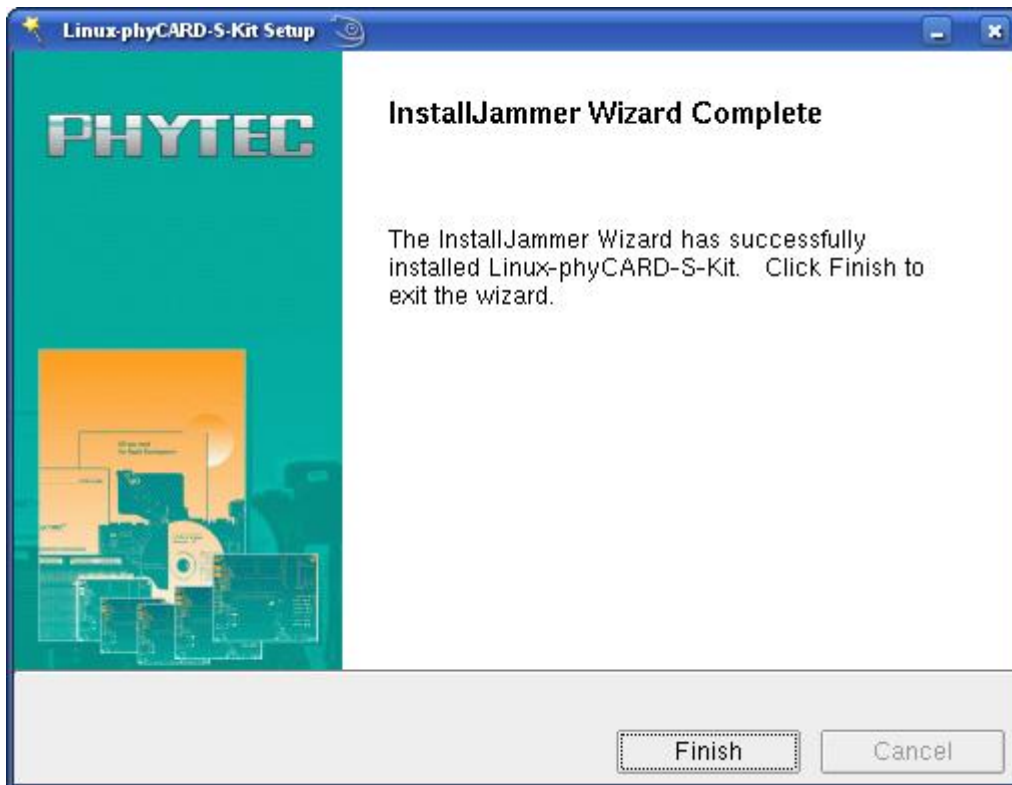
- Click *Yes* to install Eclipse. If you want to skip the installation of Eclipse, choose *No*.



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.

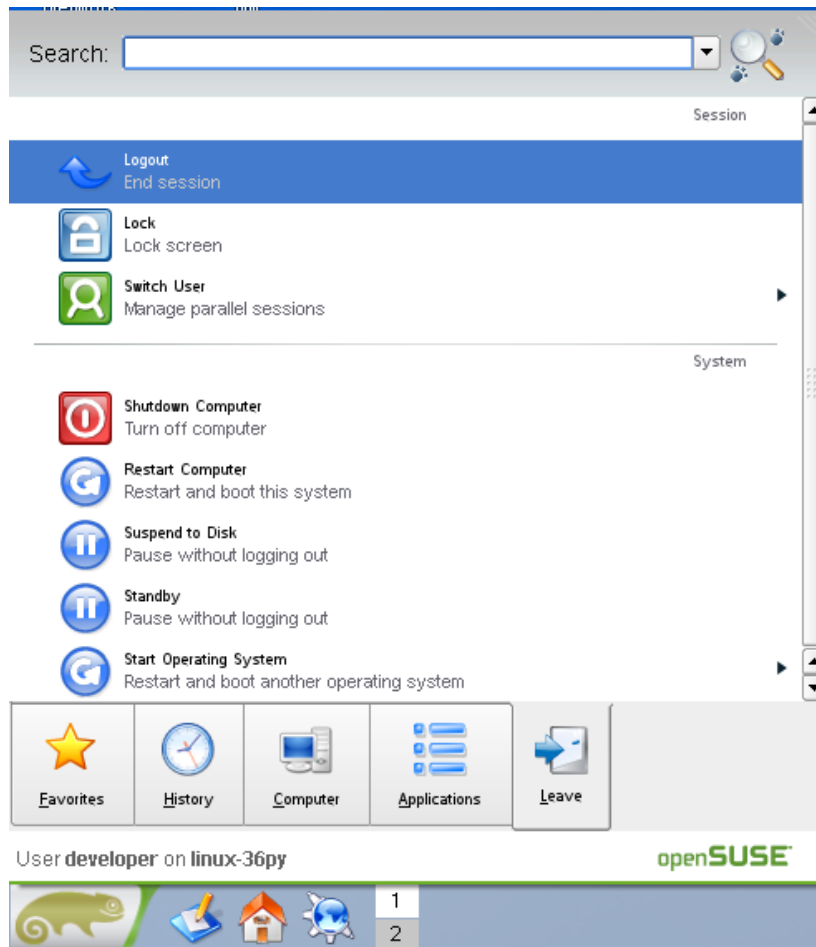


- Click *Next*.



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

Now you will have to restart the KDE desktop.



- 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.



SUCCESS

You have successfully installed the software for the Linux-phyCARD-S-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 phyCARD-S-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-v5te-linux-gnueabi/gcc-4.3.2-glibc-2.8-binutils-2.18-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-v5te-linux-  
gnueabi/gcc-4.3.2-glibc-2.8-binutils-2.18-kernel-2.6.27-  
sanitized/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-v5te-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) 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 Ethernet on the target and the appropriate network card of your host.

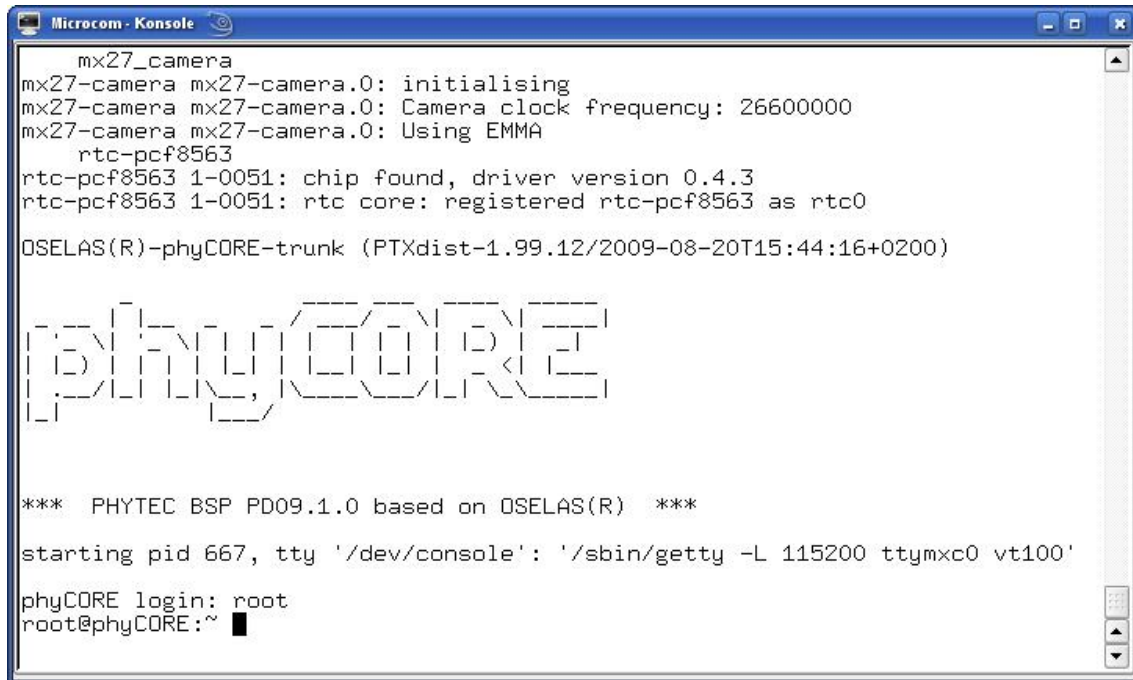


- Click the *Microcom* icon on your desktop.
- Connect the AC adapter with the power supply connector PWR (12V) on your board.



The power connector should have 12 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:



```

Microcom - Konsole
mx27_camera
mx27-camera mx27-camera.0: initialising
mx27-camera mx27-camera.0: Camera clock frequency: 26600000
mx27-camera mx27-camera.0: Using EMMA
  rtc-pcf8563
rtc-pcf8563 1-0051: chip found, driver version 0.4.3
rtc-pcf8563 1-0051: rtc core: registered rtc-pcf8563 as rtc0
OSELAS(R)-phyCORE-trunk (PTXdist-1.99.12/2009-08-20T15:44:16+0200)

phyCORE

*** PHYTEC BSP PD09.1.0 based on OSELAS(R) ***

starting pid 667, tty '/dev/console': '/sbin/getty -L 115200 ttymx0 vt100'

phyCORE login: root
root@phyCORE:~ █

```

- 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 phyCARD-S*” 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 log-structured 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 your 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*.

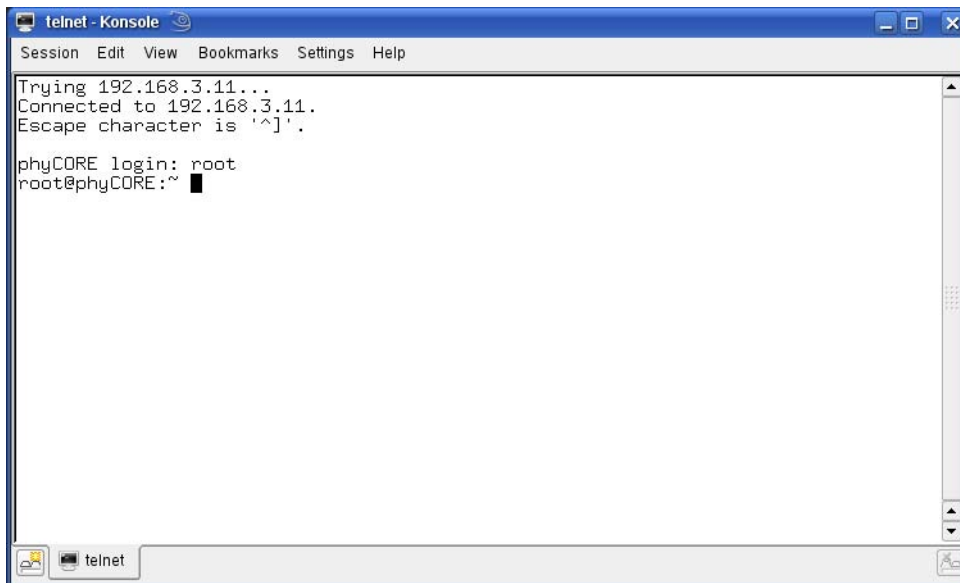
- Click *Accept*.
- Click *Finish* and close YaST.
- You need to log out and log in again for the new group membership to take effect.

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.

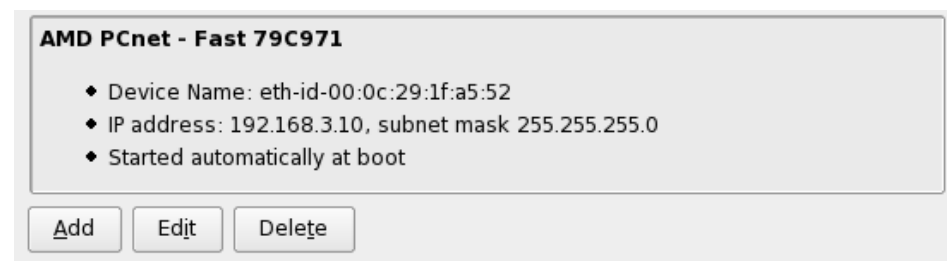
- Close the window.



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:



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 phyCARD-S 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.

2.6.1 Copying a Program to the Target

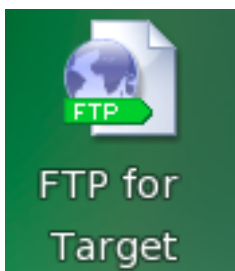


- First click the *phyCARD-S-Kit* icon on your KDE desktop.

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

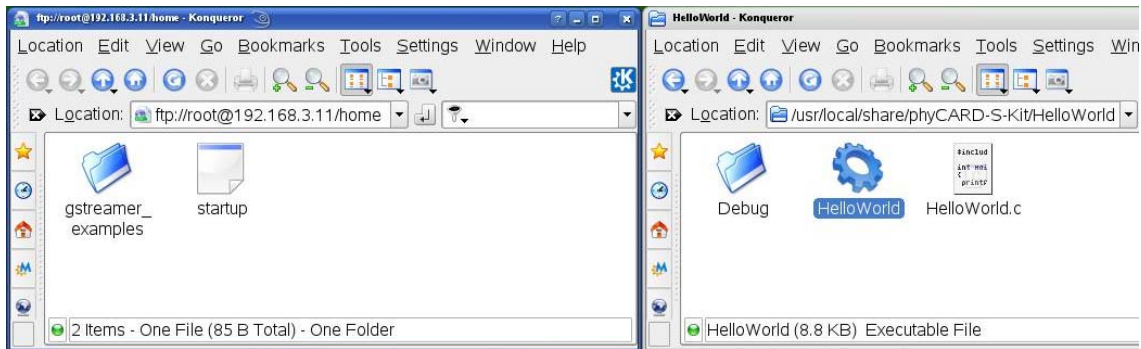


- Enter the directory *HelloWorld*.



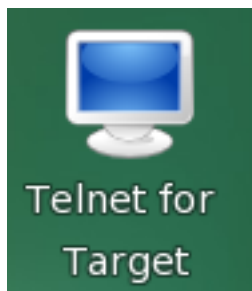
- Click the *FTP for Target* icon on your desktop.

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).

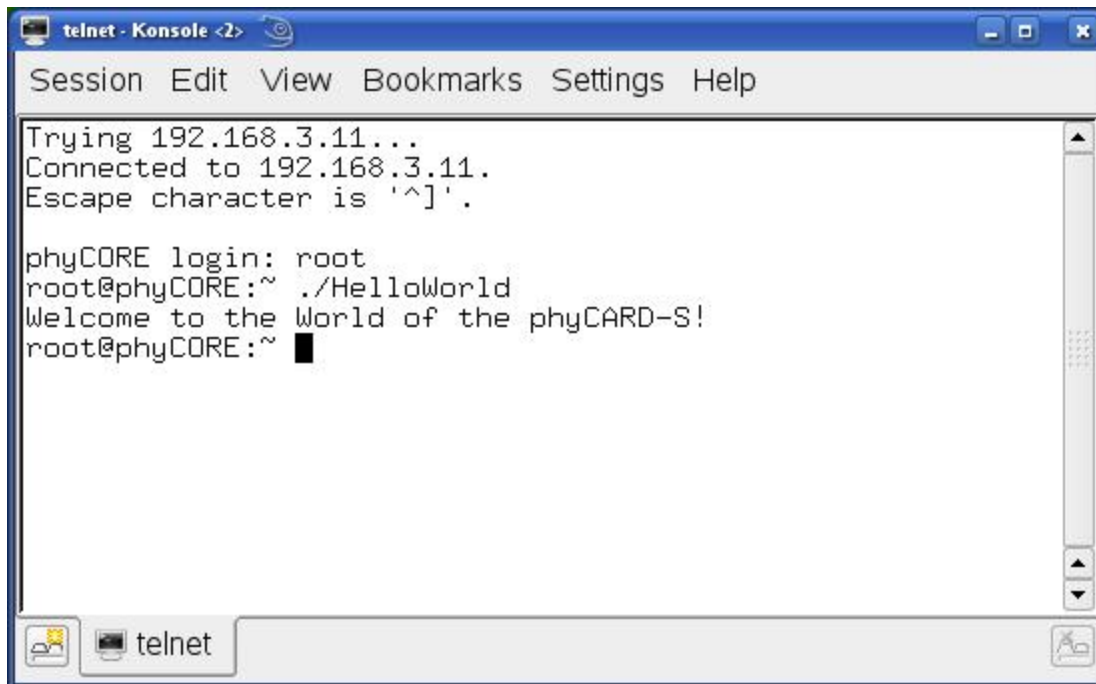


- Select the window that lists *HelloWorld* program on your hard disk.
- Click the *HelloWorld* program and hold the left mouse button pressed.
- Drag the program into the window with the FTP session to the target and release the mouse button.
- Choose *Copy here* in the appearing context menu.
- Close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

2.6.2 Using Telnet to Execute a Program on the Target



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



```
telnet - Konsole <2>
Session Edit View Bookmarks Settings Help
Trying 192.168.3.11...
Connected to 192.168.3.11.
Escape character is '^]'.

phyCORE login: root
root@phyCORE:~ ./HelloWorld
Welcome to the World of the phyCARD-S!
root@phyCORE:~ █
```

- 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 phyCARD-S!

2.6.3 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 phyCARD-S 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 public 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.

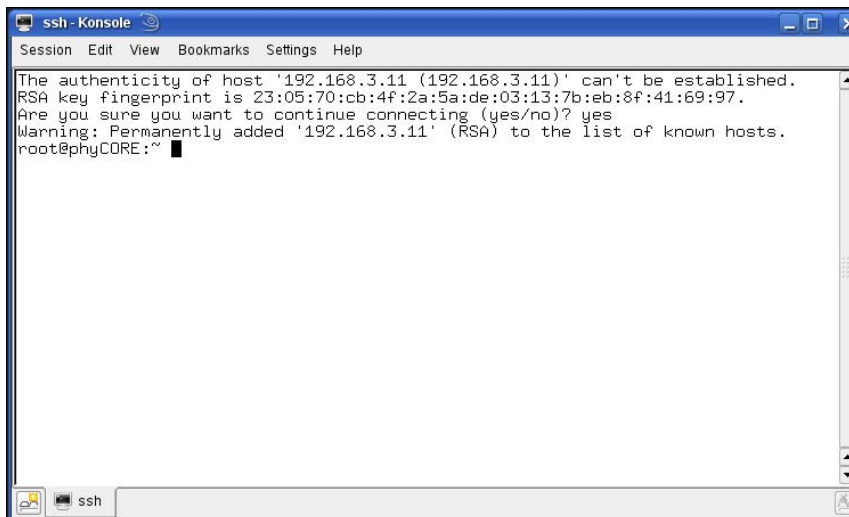


This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing. But note that */etc/hosts.equiv* is, in general, inherently insecure and should be disabled if security is a concern.



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

A new window opens.



In this window you can see that the authenticity of the phyCARD-S 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 phyCARD-S before.

The program starts and you should see the following output:

Welcome to the World of the phyCARD-S!

- Close the SSH window.



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

2.7 Advanced Information

2.7.1 Copying a Program to the Target with the Command Line

- Open a new terminal window.
- Change to */usr/local/share/phyCARD-S-Kit/HelloWorld*:

```
cd /usr/local/share/phyCARD-S-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.

2.7.2 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.

2.7.3 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.

Chapter 3 Getting More Involved

**70 min**

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 phyCARD-S 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 phyCARD-S.

As the first step, open a new terminal.



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

```
cp /usr/local/share/phyCARD-S-Kit/linux-*-PCA100.tar.bz2 ~
```

```
cd ~
```

- Unpack the kernel source archive:

```
tar xvjf linux-*-PCA100.tar.bz2
```

- Change to the newly created directory:

```
cd linux-*
```

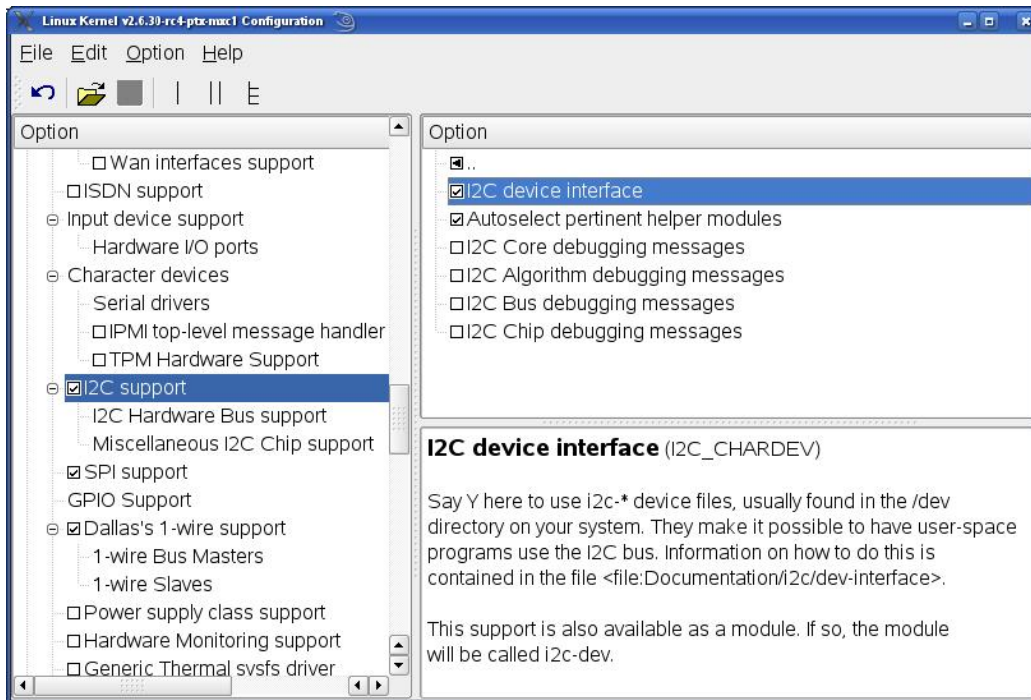
The kernel would normally be built for the native machine architecture of your host. To use the ARM architecture and ARM cross compiler suitable for the phyCARD-S instead, you will have to specify the architecture and the cross compiler on the command line.

- Type:

```
make xconfig ARCH=arm
```

The kernel configuration tool *qconf* starts.

In our example we show how to add I2C support with the help of *qconf*.



- Select *I2C support* in *Device drivers*.
- Check *I2C support*.
- Check *I2C device interface*.
- Save your configuration and exit the configuration tool.
- Type:

make ARCH=arm CROSS_COMPILE=arm-v5te-linux-gnueabi-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**.

- 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.

3.1.1 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 phyCARD-S'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:

```
0xc0000000 - 0xc003ffff (U-Boot)
0xc0040000 - 0xc005ffff (U-Boot Environment)
0xc0060000 - 0xc025ffff (Linux Kernel)
0xc0260000 - 0xc1ffffff (Linux Root File System)
```

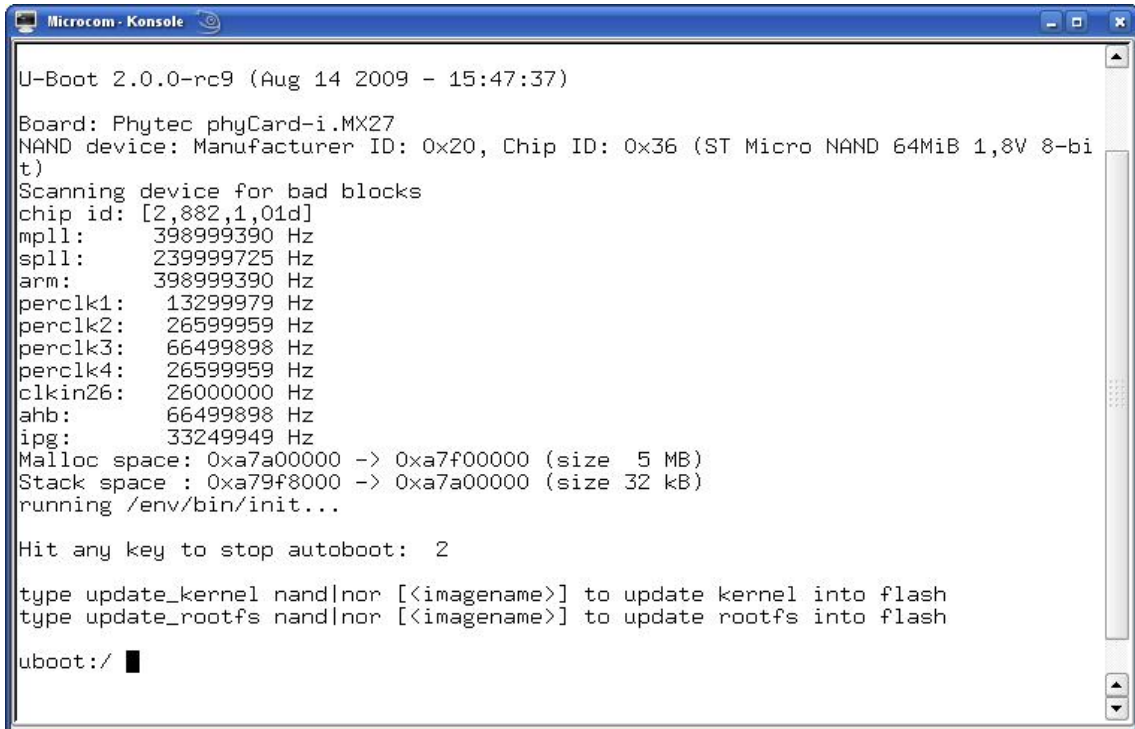


You should never erase the U-Boot partition. If this partition is erased, you won't be able to start your target anymore. Refer to the chapter "*Installing Linux on the phyCARD-S*" for detailed information on how to restore your U-Boot partition in such a case.



- 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 (Aug 14 2009 - 15:47:37)

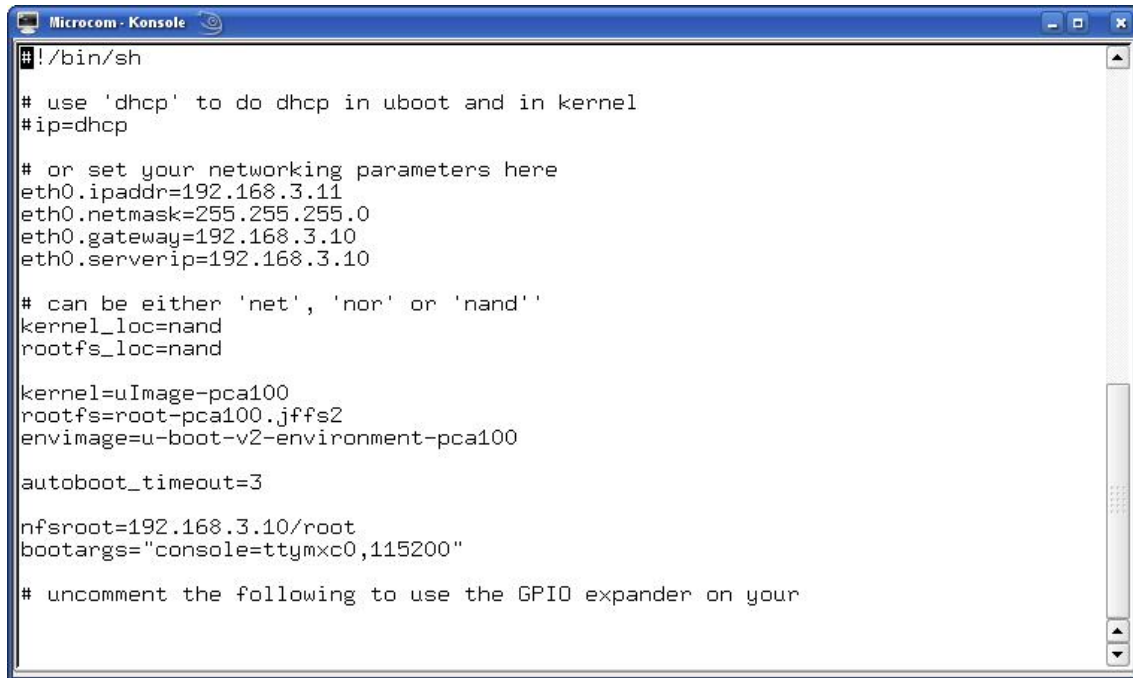
Board: Phytex phyCard-i.MX27
NAND device: Manufacturer ID: 0x20, Chip ID: 0x36 (ST Micro NAND 64MiB 1,8V 8-bit)
Scanning device for bad blocks
chip id: [2,882,1,01d]
mp11: 398999390 Hz
sp11: 239999725 Hz
arm: 398999390 Hz
perclk1: 13299979 Hz
perclk2: 26599959 Hz
perclk3: 66499898 Hz
perclk4: 26599959 Hz
clk26: 26000000 Hz
ahb: 66499898 Hz
ipg: 33249949 Hz
Malloc space: 0xa7a00000 -> 0xa7f00000 (size 5 MB)
Stack space : 0xa79f8000 -> 0xa7a00000 (size 32 kB)
running /env/bin/init...

Hit any key to stop autoboot: 2

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.

A screenshot of a terminal window titled "Microcom - Konsole". The terminal shows the following configuration parameters for U-Boot:

```
#!/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=nand
rootfs_loc=nand

kernel=uImage-pca100
rootfs=root-pca100.jffs2
envimage=u-boot-v2-environment-pca100

autoboot_timeout=3

nfsroot=192.168.3.10/root
bootargs="console=ttyMxc0,115200"

# uncomment the following to use the GPIO expander on your
```

- 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 phyCARD-S reboots with the new settings applied. Again, 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*.

- Type **update_kernel nand 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 phyCARD-S 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/PCA100 phyCARD-S/Linux-Kit/BSP/Images*.

- To restore the kernel, copy the file *uImage-pca100* to your host's */tftpboot* directory.
- Type **update_kernel nand uImage-pca100** 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/PCA100 phyCARD-S/Linux-Kit/BSP/Images* directory on your setup CD-ROM.

- To restore the root file system, copy the file *root-pca100.jffs2* to your host's */tftpboot* directory.
- Type **update_rootfs nand root-pca100.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.

3.2.1 Copying the *HelloWorld* Project

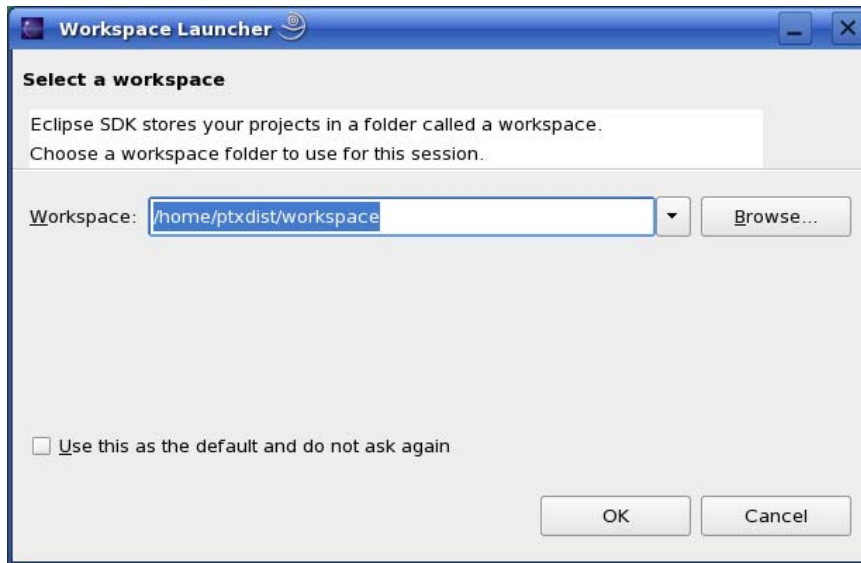


- Click the *phyCARD-S-Kit* icon on your KDE desktop.
- Right-click the *HelloWorld* directory and select *Copy*.
- Browse to your home directory.
- If the *workspace* directory doesn't exist, create a directory *workspace* in your home directory.
- Enter the *workspace* directory.
- Right-click in the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

3.2.2 Starting Eclipse and Importing the Example Project

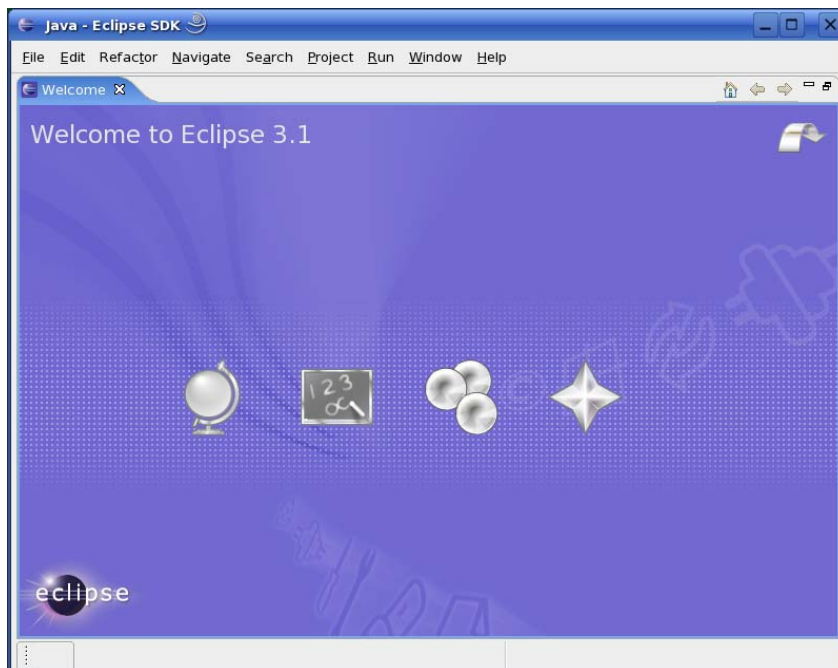


- Click the *Eclipse* icon to start the application. You can find this icon on your desktop.

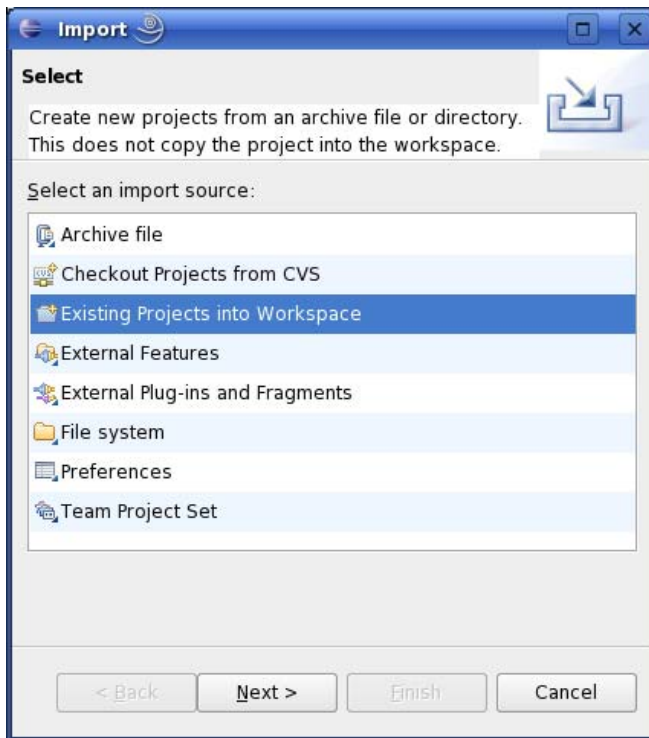


- Confirm the Workspace directory with *OK*.

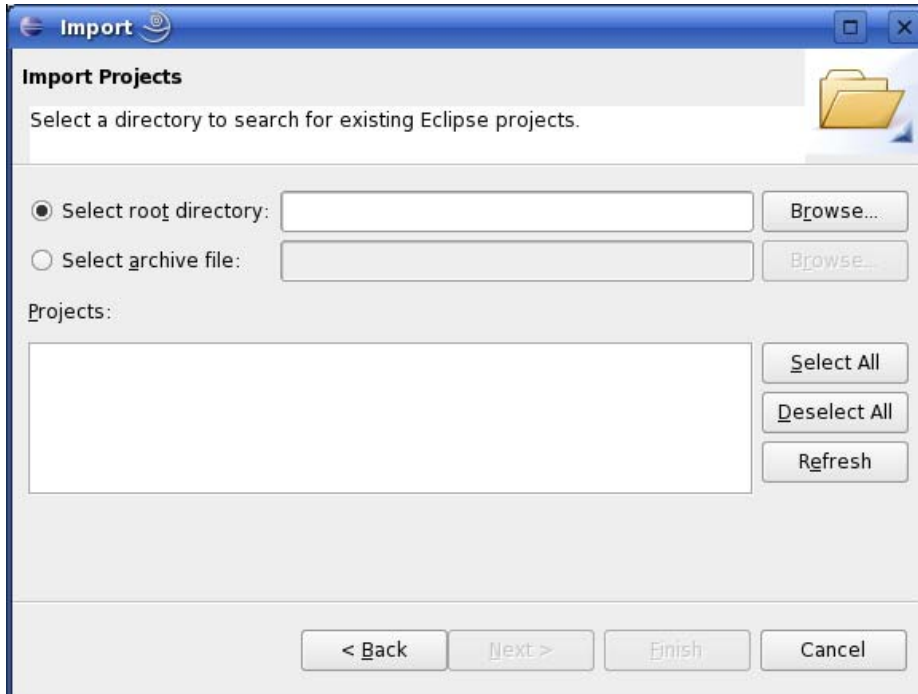
The *Welcome* screen will appear.



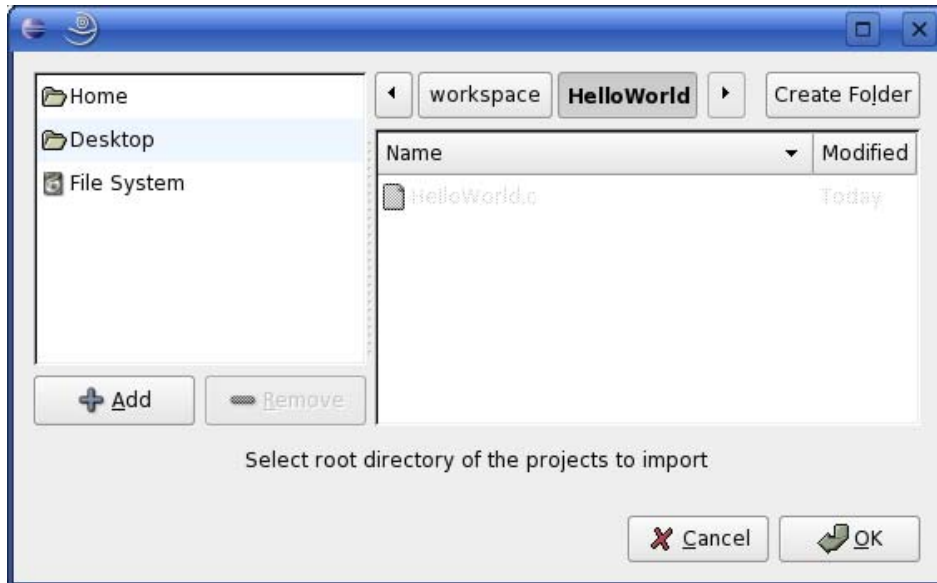
- Select *File* ► *Import* from the menu bar.



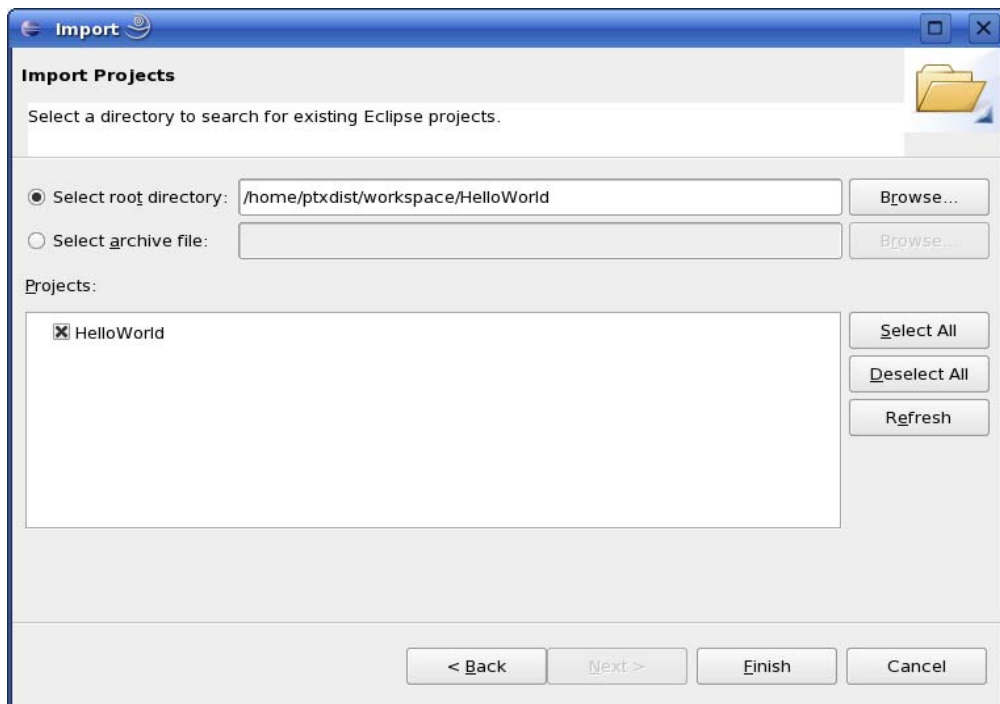
- Select *Existing Projects into Workspace*.
- Click *Next*.



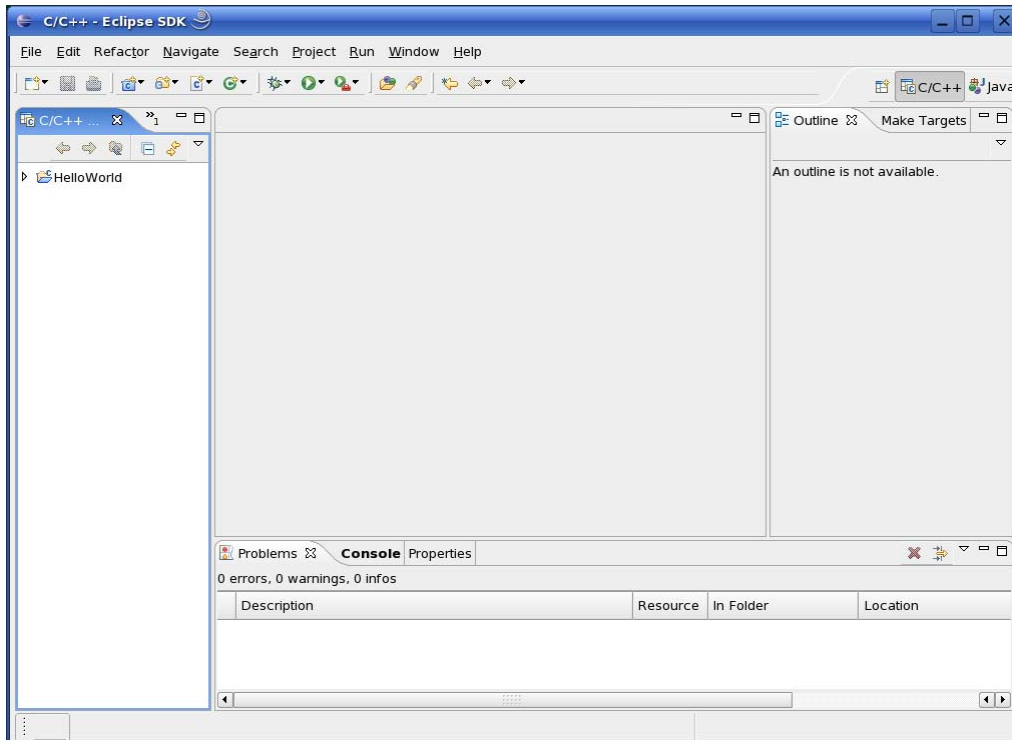
- Select *Browse*.



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



- Select *Finish* to import the project.



- 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 phyCARD-S!*” message in the *Console* window.

- Select the *Console* tab.

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



```

C-Build [HelloWorld]
HelloWorld.c
Finished building: ../HelloWorld.c

Building target: HelloWorld
Invoking: GCC C Linker
arm-v5te-linux-gnueabi-gcc -oHelloWorld ../HelloWorld.o
Finished building target: HelloWorld

make --no-print-directory post-build
ftp -u ftp://root:root@192.168.3.11/ ../HelloWorld;ssh root@192.168.3.11 ../HelloWorld
Welcome to the World of the phyCARD-S!

Build complete for project HelloWorld
  
```



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



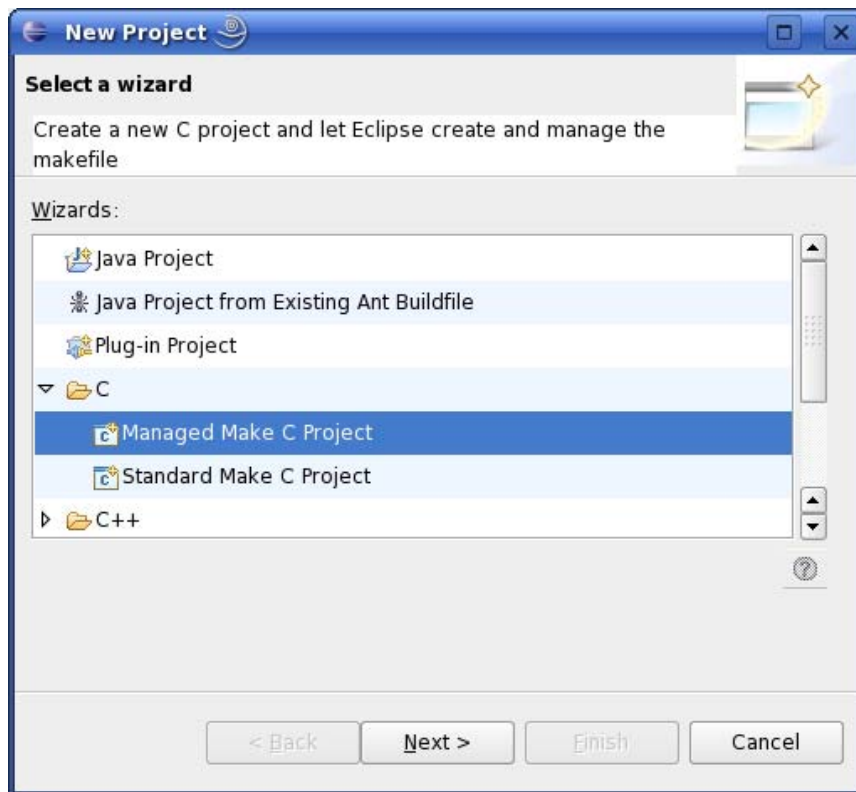
You have successfully passed the first steps with the Eclipse IDE. You are now able to import existing projects into the Eclipse Workspace. You can compile an existing project and execute the program on the target.

3.3 Creating a New Project

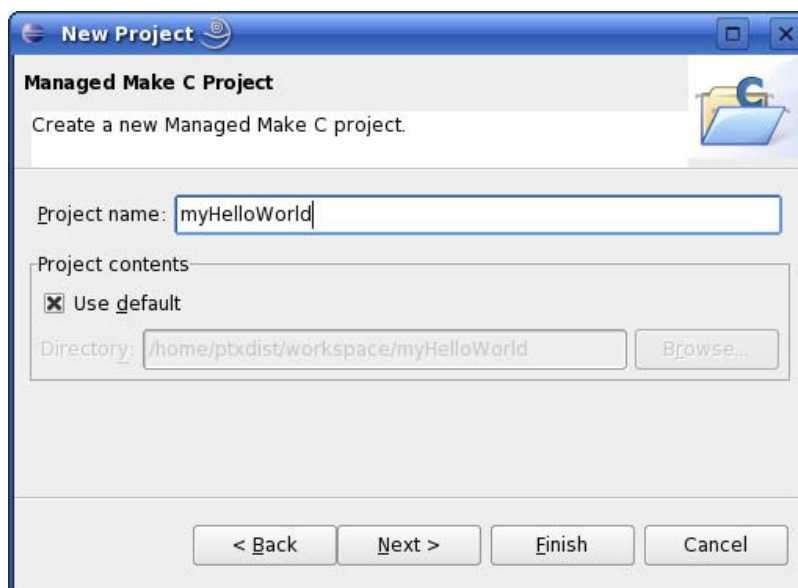
In this section you will learn how to create a new project with Eclipse and how to configure the project for use with the GNU C/C++ cross development tool chain.

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

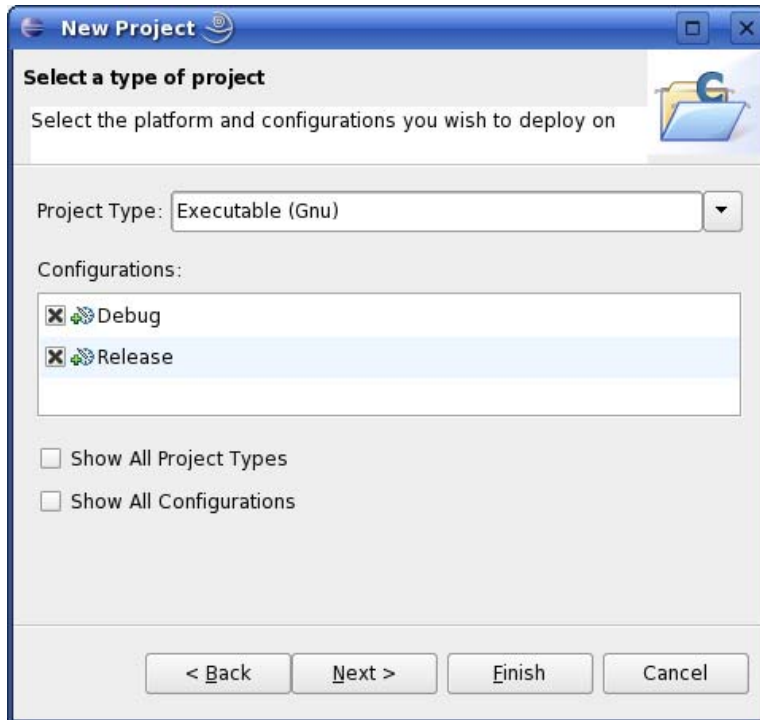
A new dialog opens.



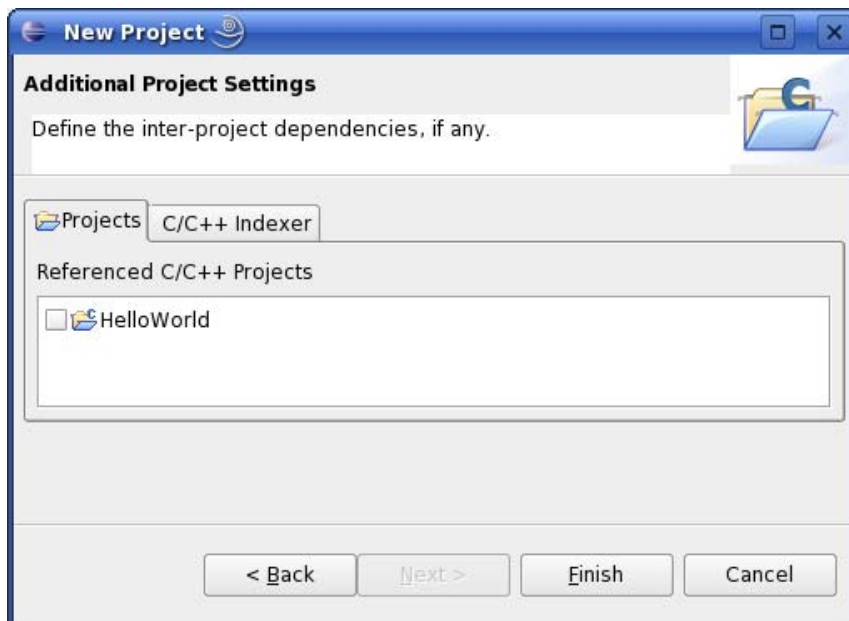
- Select *Managed Make C Project* and click *Next*.



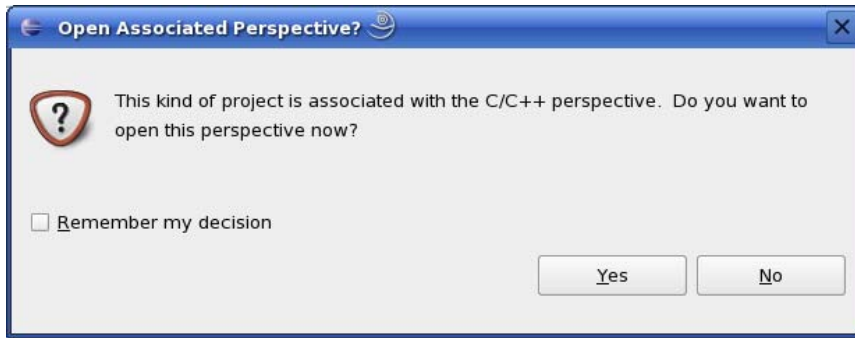
- Enter the project name *myHelloWorld* and click *Next*.



- Click *Next*.

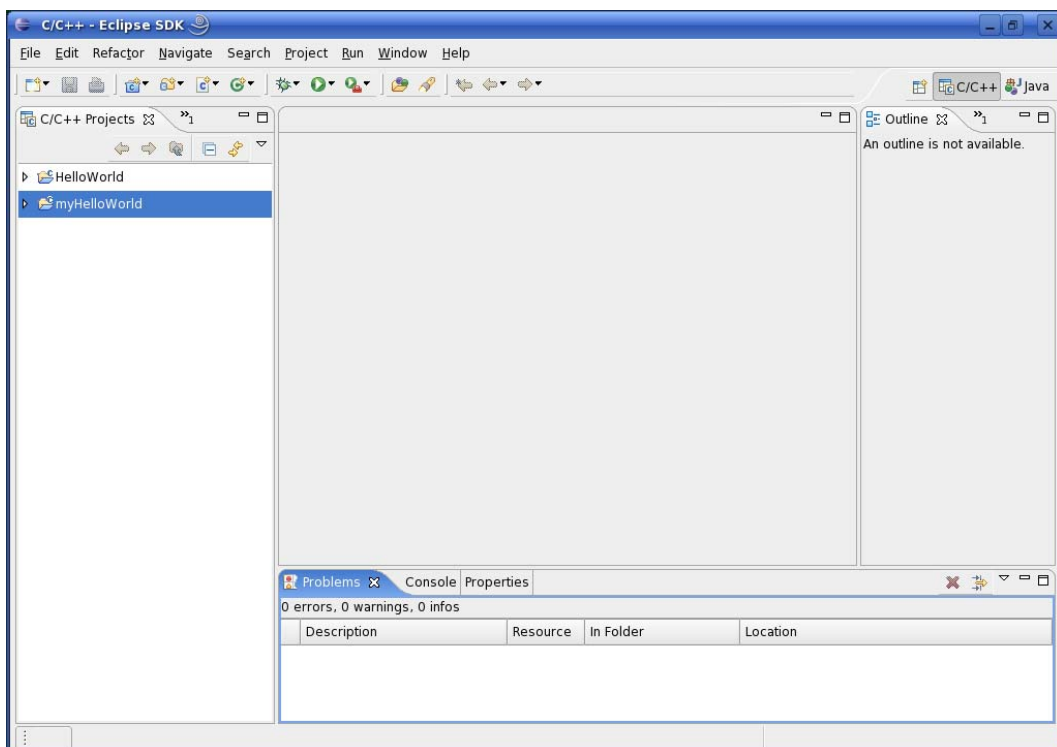


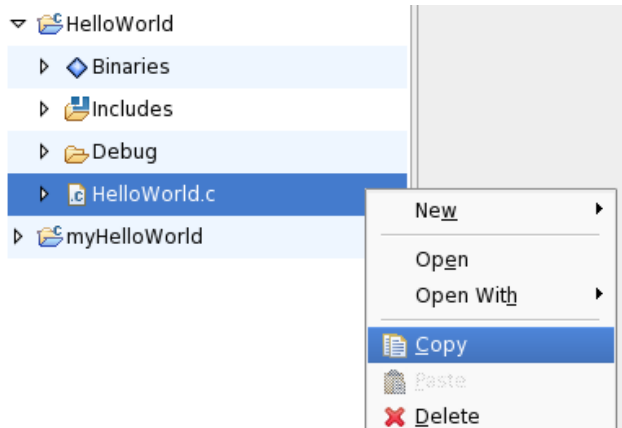
- Click *Finish*.



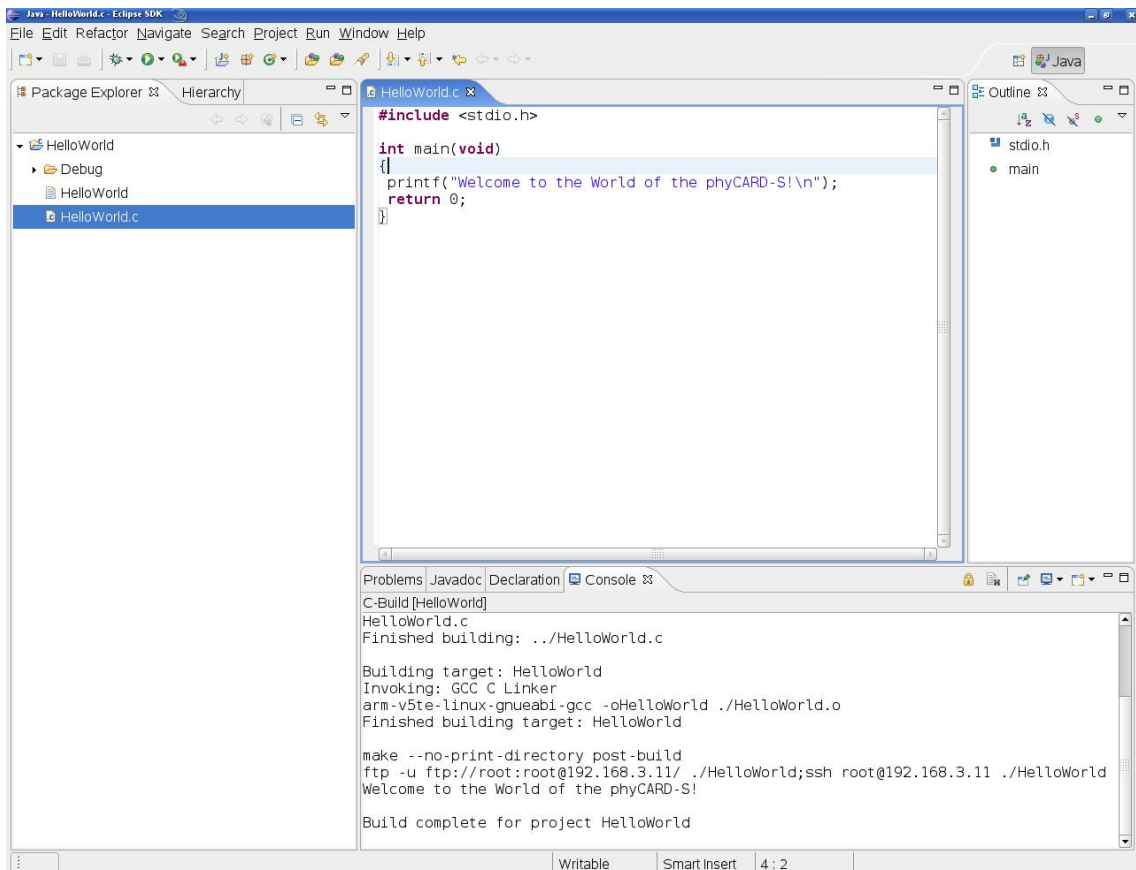
- Select *Yes* to open the C/C++ perspective.

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





- 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*.

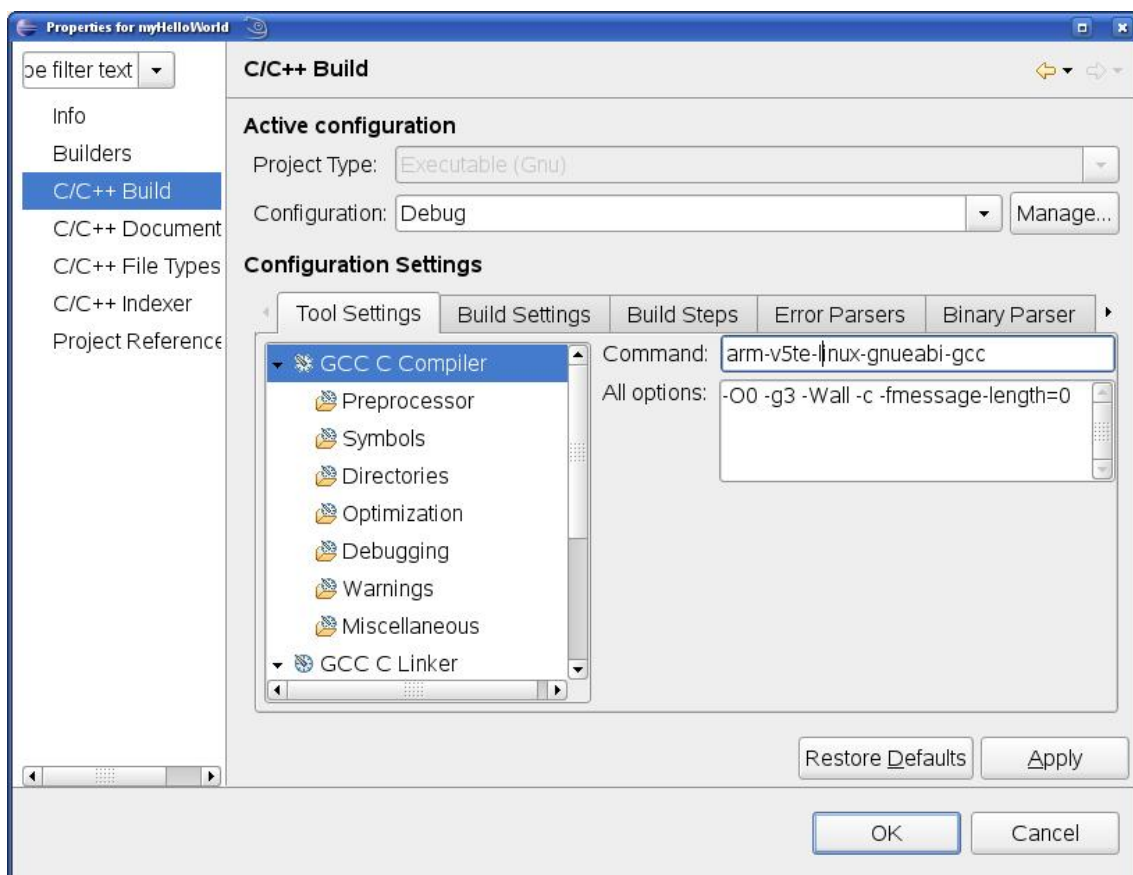
- Double-click on *HelloWorld.c* in the *myHelloWorld* project.

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.

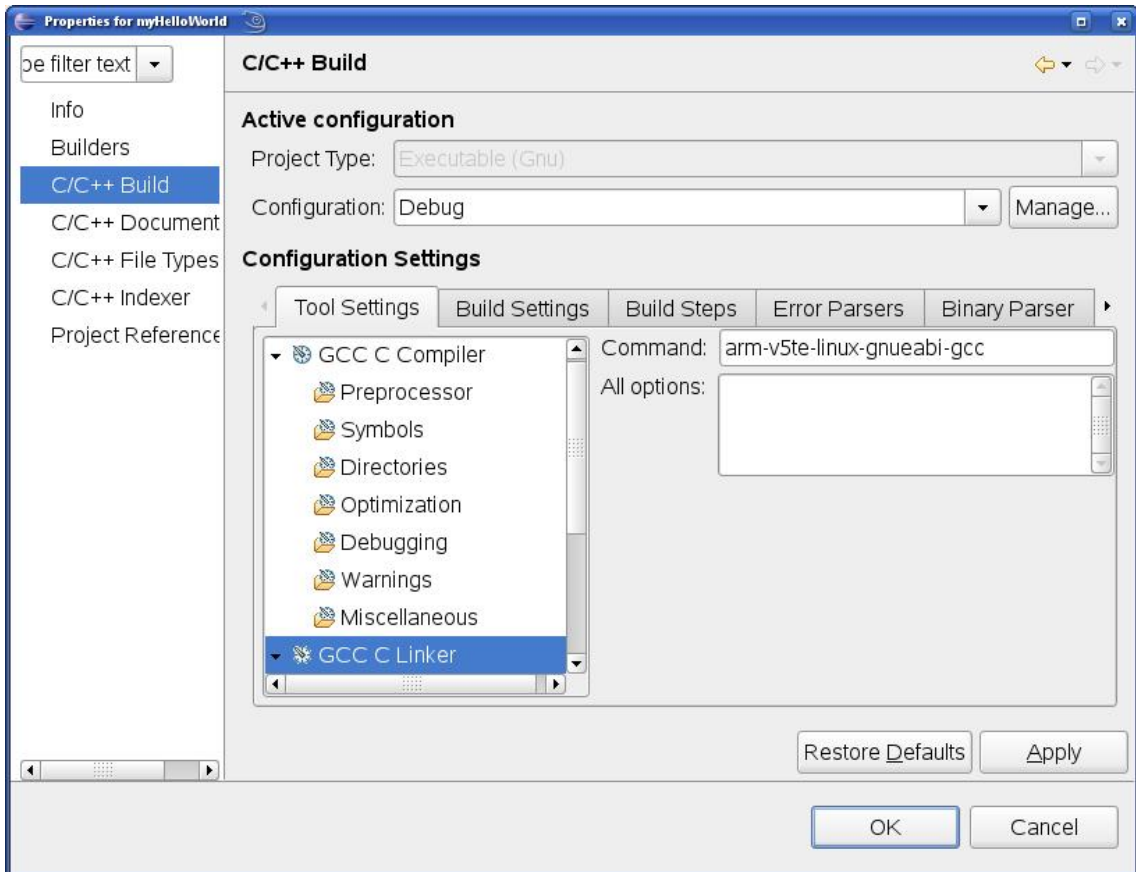
To compile your project for the phyCARD-S instead, you will have to use the GNU C/C++ cross compiler.

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

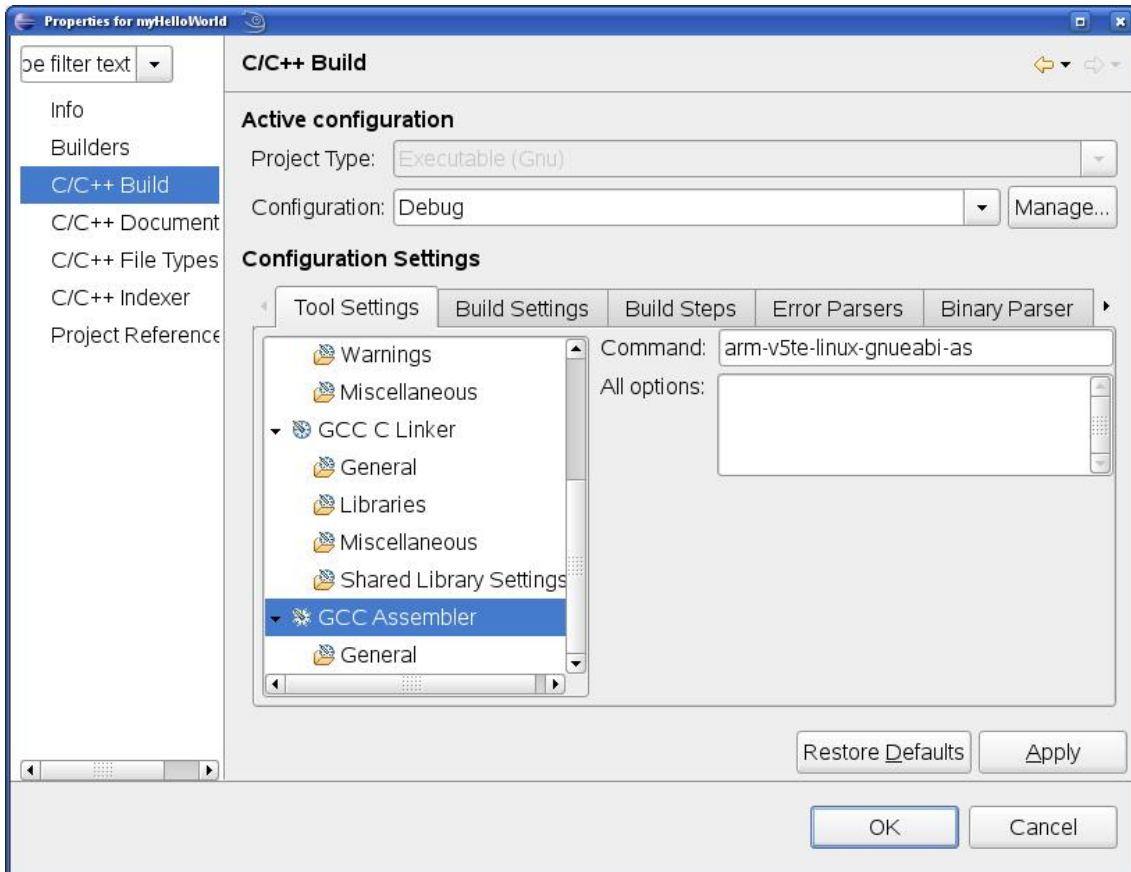
The *Properties* dialog appears.



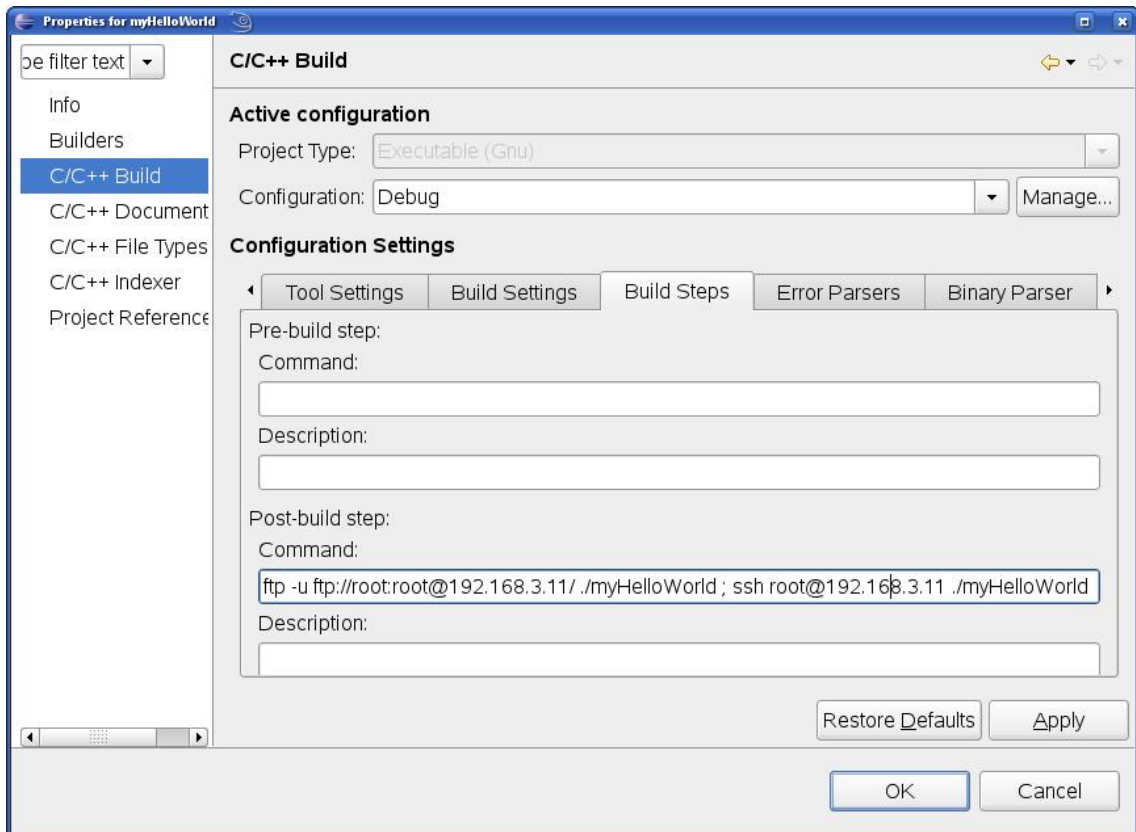
- Select *C/C++ Build*.
- Enter **arm-v5te-linux-gnueabi-gcc** into the *Command* input field.



- Select *GCC C Linker*.
- Enter **arm-v5te-linux-gnueabi-gcc** into the *Command* input field.



- Select *GCC Assembler*.
- In the *Command* input field, change the default **as** to **arm-v5te-linux-gnueabi-as**.
- Click *Apply*.



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

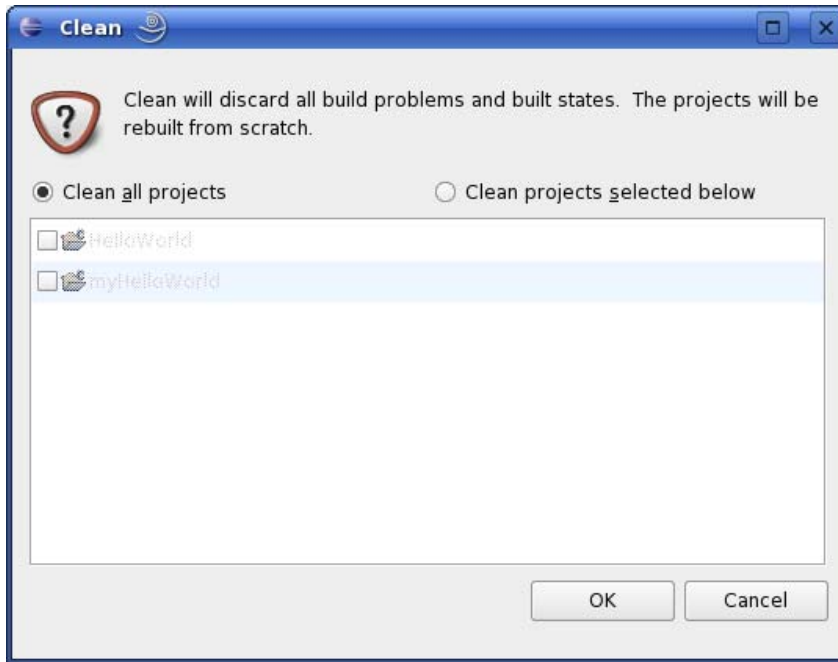
```
ftp -u ftp://root:root@192.168.3.11/ ./myHelloWorld ; ssh  
root@192.168.3.11 ./myHelloWorld
```



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

CAUTION Be sure the file *myHelloWorld* on the target will have execution rights, because otherwise *ssh* will fail.

- Click *Apply*.
- Click *OK*.



- Select *Project* ► *Clean* from the menu bar.
- Confirm with *OK*.

The project will be rebuilt.

- Select the *Console* tab.

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

```
C-Build [myHelloWorld]
myhelloworld.c
Finished building: ../myHelloWorld.c

Building target: myHelloWorld
Invoking: GCC C Linker
arm-v5te-linux-gnueabi-gcc -omyHelloWorld ../myHelloWorld.o
Finished building target: myHelloWorld

make --no-print-directory post-build
ftp -u ftp://root:root@192.168.3.11/ ../myHelloWorld ; ssh root@192.168.3.11 ../myHelloWorld
Welcome to the World of the phyCARD-S!

Build complete for project myHelloWorld
```



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 phyCARD-S, is connected to the system console */dev/console*):

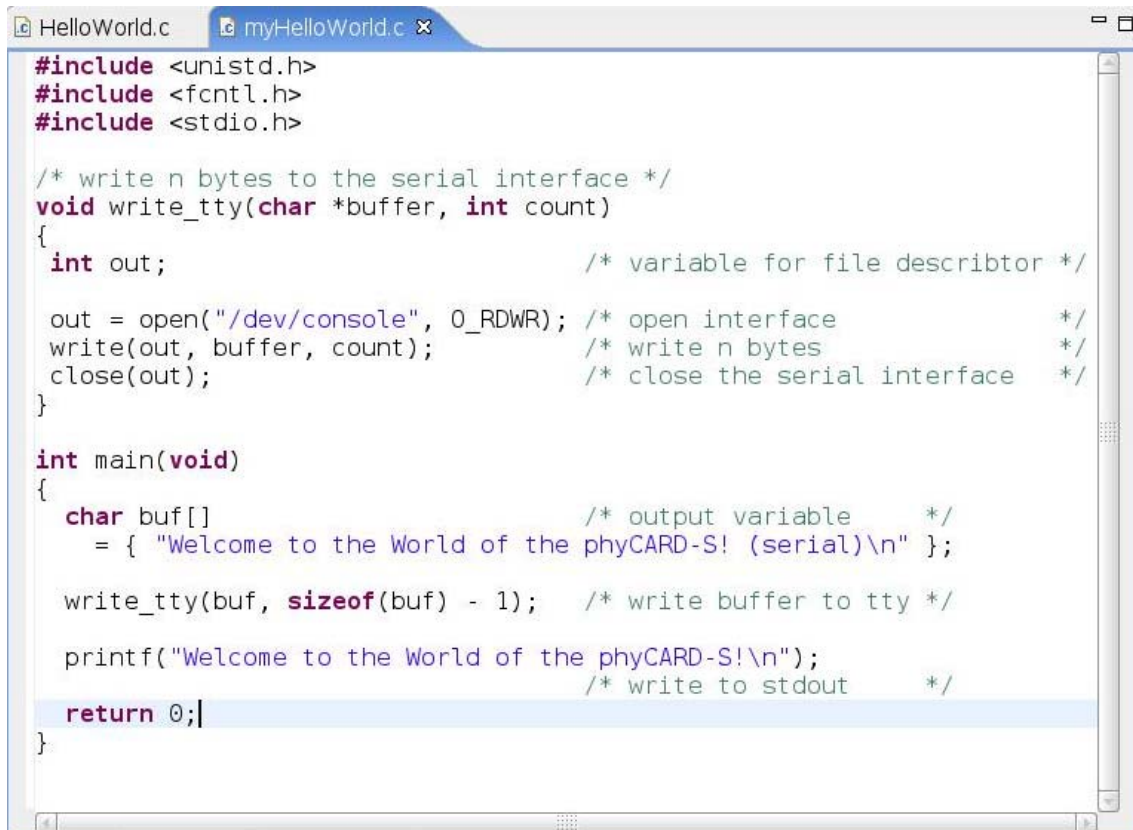
```
void write_tty(char *buffer, int count)
{
    int out;

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

- Enter the following two lines in the *main()* function to declare the buffer and call the *write_tty()* function.

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

In the next screenshot you can see the complete program.



```

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

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

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

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


    write_tty(buf, sizeof(buf) - 1); /* write buffer to tty */

    printf("Welcome to the World of the phyCARD-S!\n");
                                        /* write to stdout */
    return 0;
}

```

- Save your program after changing the code.

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



```

Problems Console Properties
C-Build [myHelloWorld]
Finished building: ../myHelloWorld.c

Building target: myHelloWorld
Invoking: GCC C Linker
arm-v5te-linux-gnueabi-gcc -omyHelloWorld ../myHelloWorld.o
Finished building target: myHelloWorld

make --no-print-directory post-build
ftp -u ftp://root:root@192.168.3.11/ ../myHelloWorld ; ssh root@192.168.3.11 ../myHelloWorld
Welcome to the World of the phyCARD-S!

Build complete for project myHelloWorld

```

3.4.1 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 phyCARD-S! (serial)
Welcome to the World of the phyCARD-S!
- 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.



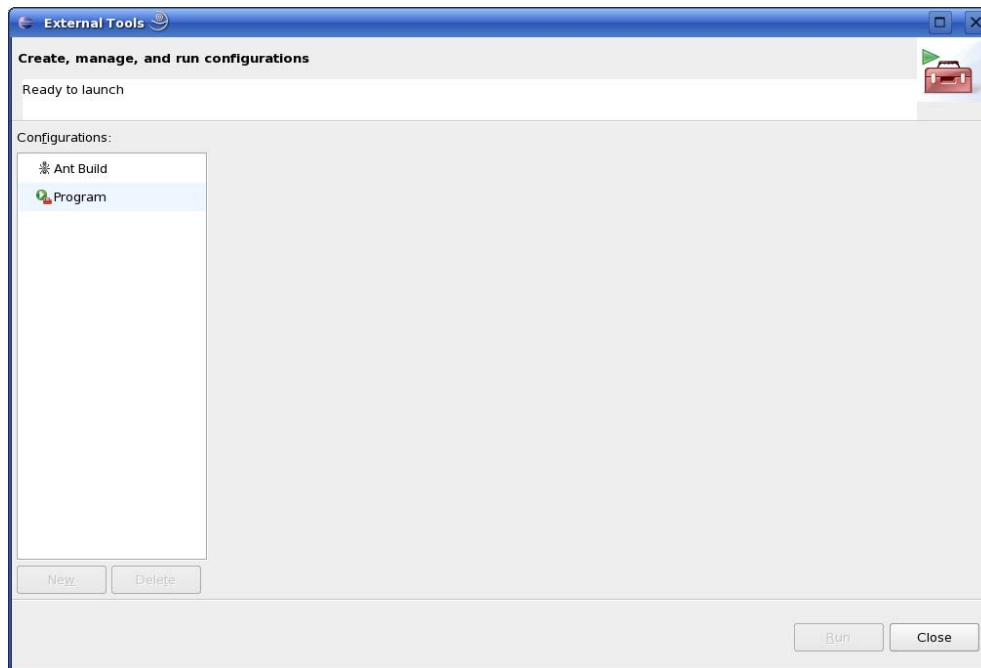
The first line is a direct output on the serial interface. You can't see this line in an SSH session, because you are connected over a TCP/IP connection to the target. With Microcom, however, you have direct access to serial interface, so you can also see the line that it written to the serial console.

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()*.

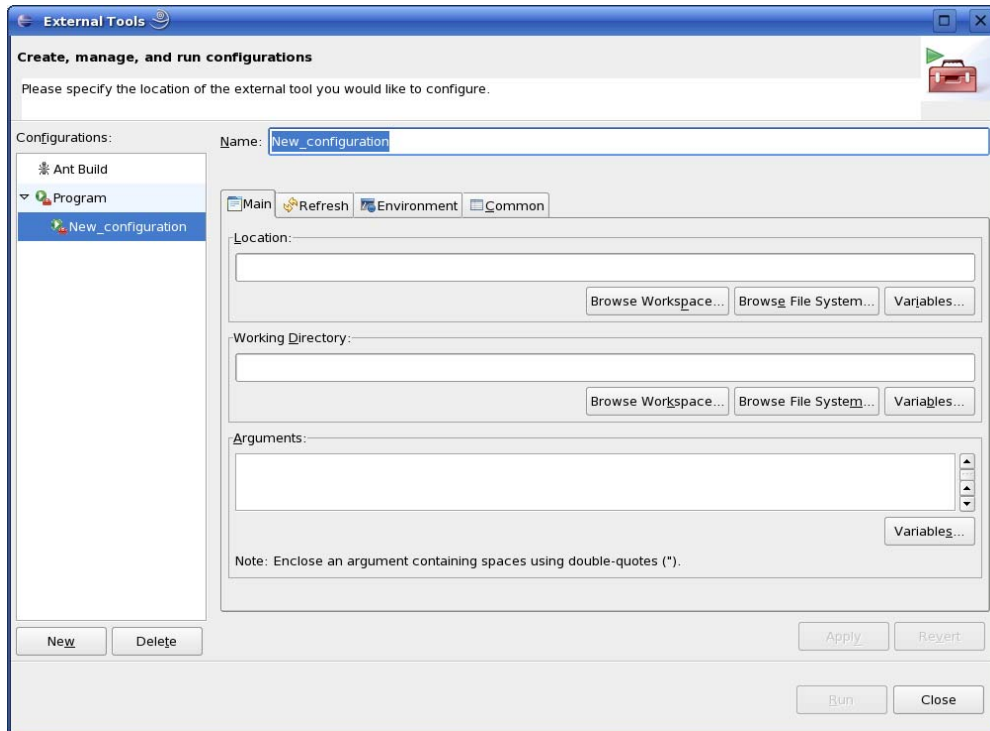
This procedure is in principle quite typical for Linux because Linux treats everything like a file.

3.5 Starting a Program out of Eclipse on the Target

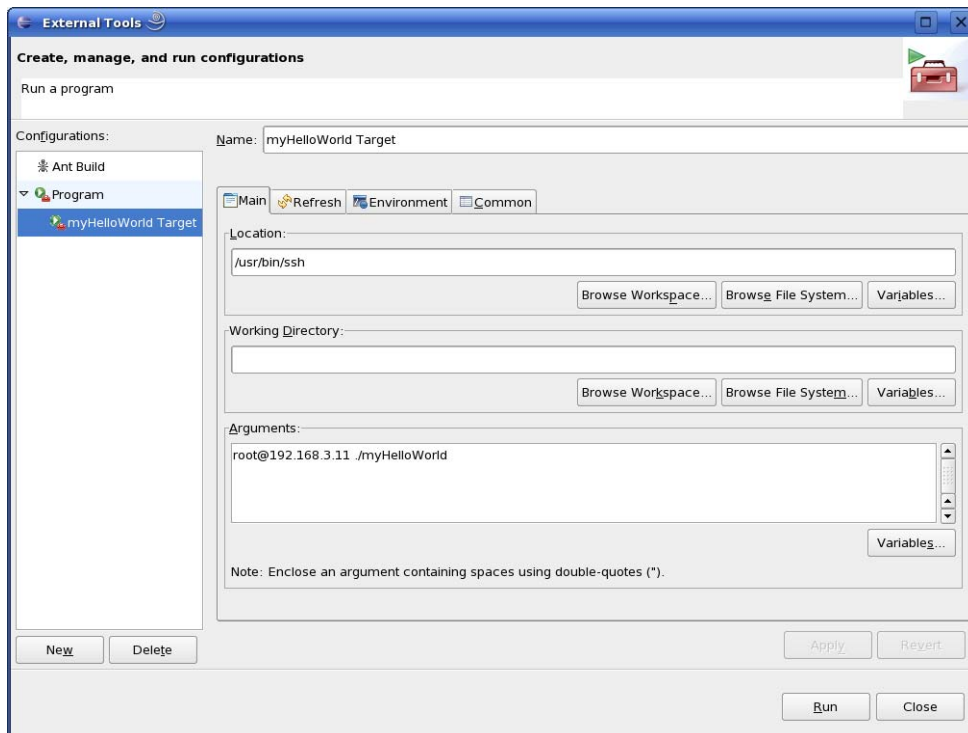
After compiling a project in Eclipse, the program is copied to the target and directly executed. A program can also be executed on the target without compiling a project. In the following section you will learn how to start a program on the target as an external tool.



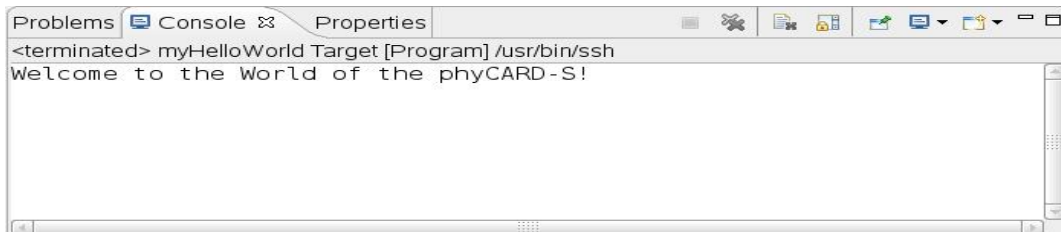
- Select *Run* ► *External Tools* ► *External Tools* from the menu bar.



- Select *Program*.
- Select *New*.



- 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.

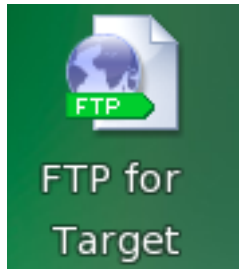


TIP

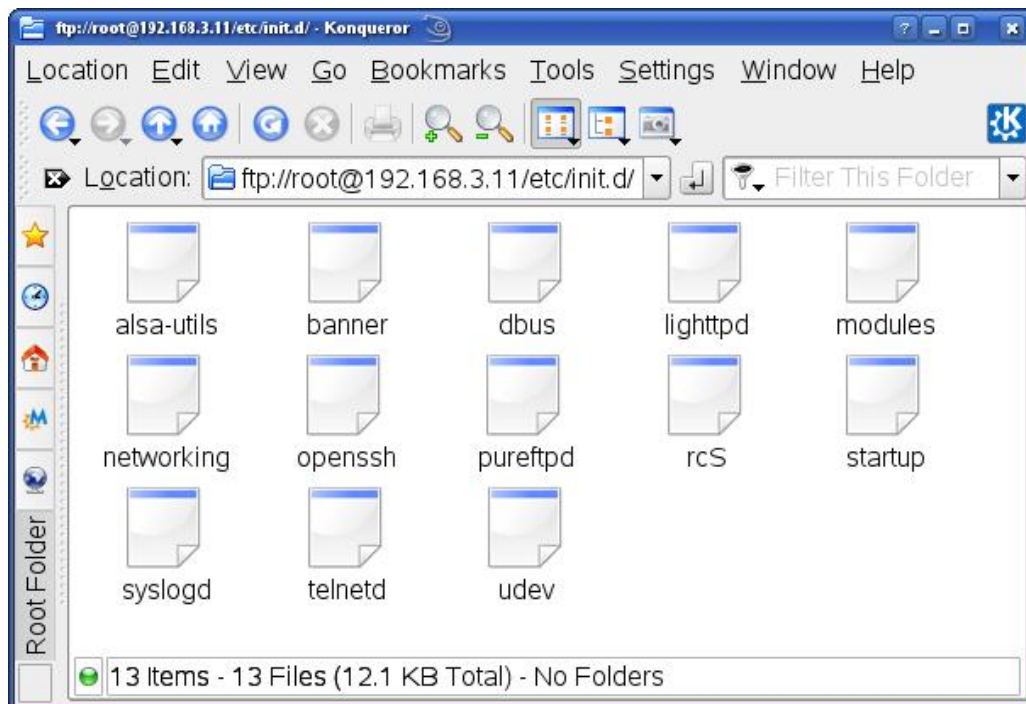
The scripts for controlling the system startup live in */etc/init.d*. These are executed directly or indirectly by */sbin/init*, the father of all processes. The configuration of */sbin/init* is placed in */etc/inittab*.

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*.



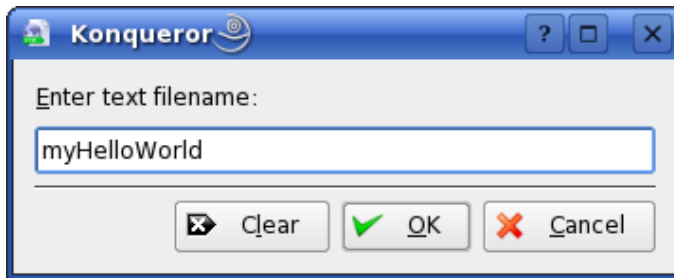
- Click the *FTP for Target* icon on your KDE desktop.



- 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).

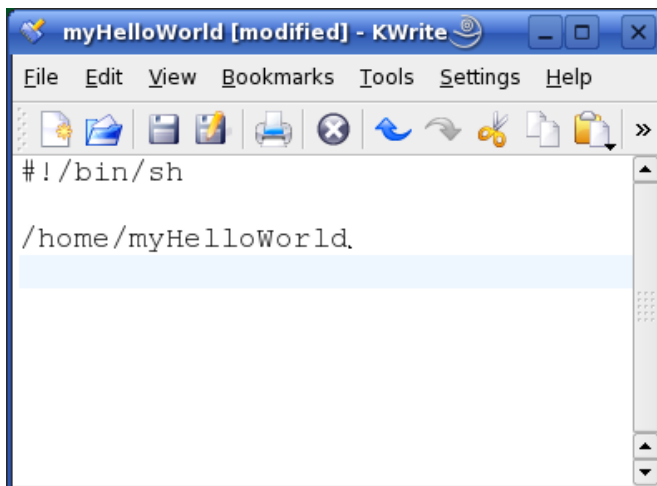
In the directory */etc/init.d* you can see the existing scripts.

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



- 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.

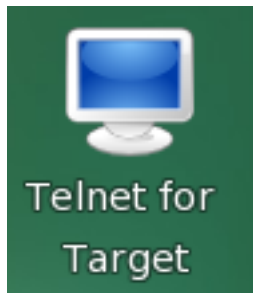


- Enter the following two lines:

```
#!/bin/sh
```

```
/home/myHelloWorld
```

- Select *File* ► *Save*.
- Close the *KWrite* window.
- Close the *FTP* window.



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

 A screenshot of a telnet terminal window titled "telnet - Konsole". The window shows the command prompt "root@phyCORE:/etc/rc.d" followed by "ls -l". The output is a long list of files with permissions, owner, group, date, time, and symbolic link targets. The files are listed in alphabetical order of their link names, starting with S00udev and ending with S99zzz_PHYTEC_BSP_version_startup_script.


```

root@phyCORE:/etc/rc.d ls -l
lrwxrwxrwx 1 root root 14 Aug 20 2009 S00udev -> ../init.d/udev
lrwxrwxrwx 1 root root 17 Aug 20 2009 S08syslog -> ../init.d/syslogd
lrwxrwxrwx 1 root root 14 Aug 20 2009 S12dbus -> ../init.d/dbus
lrwxrwxrwx 1 root root 17 Aug 20 2009 S16openssh -> ../init.d/openssh
lrwxrwxrwx 1 root root 17 Aug 20 2009 S16telnetd -> ../init.d/telnetd
lrwxrwxrwx 1 root root 20 Aug 20 2009 S21alsa-utils -> ../init.d/alsa-utils
lrwxrwxrwx 1 root root 20 Aug 20 2009 S26networking -> ../init.d/networking
lrwxrwxrwx 1 root root 18 Aug 20 2009 S91lighttpd -> ../init.d/lighttpd
lrwxrwxrwx 1 root root 18 Aug 20 2009 S91pureftpd -> ../init.d/pureftpd
lrwxrwxrwx 1 root root 17 Aug 20 2009 S98modules -> ../init.d/modules
lrwxrwxrwx 1 root root 16 Aug 20 2009 S99banner -> ../init.d/banner
lrwxrwxrwx 1 root root 17 Aug 20 2009 S99startup -> ../init.d/startup
-rwxr-xr-x 1 root root 77 Aug 20 2009 S99zzz_PHYTEC_BSP_version_startup_script
root@phyCORE:/etc/rc.d █
  
```

- 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 **ls -l** to list the directory contents.

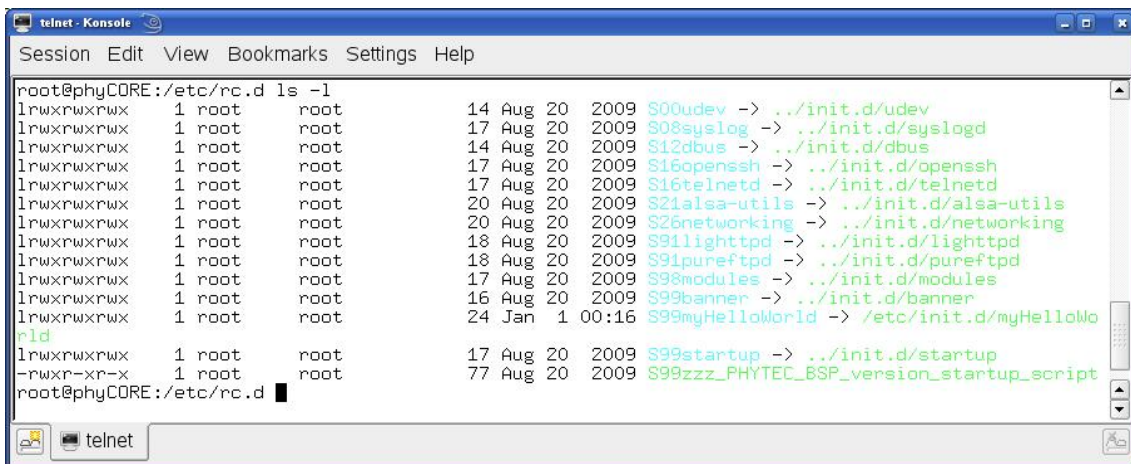
You can see the different links to the scripts in the directory */etc/init.d*. The scripts are started in alphabetic order: The script *udev* is the first script started because the link starts with *S00...*, whereas *S99zzz_PHYTEC_BSP_version_startup_script* will be started last.

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:

In -s /etc/init.d/myHelloWorld /etc/rc.d/S99myHelloWorld

- Type **ls -l** again to check the newly created link.

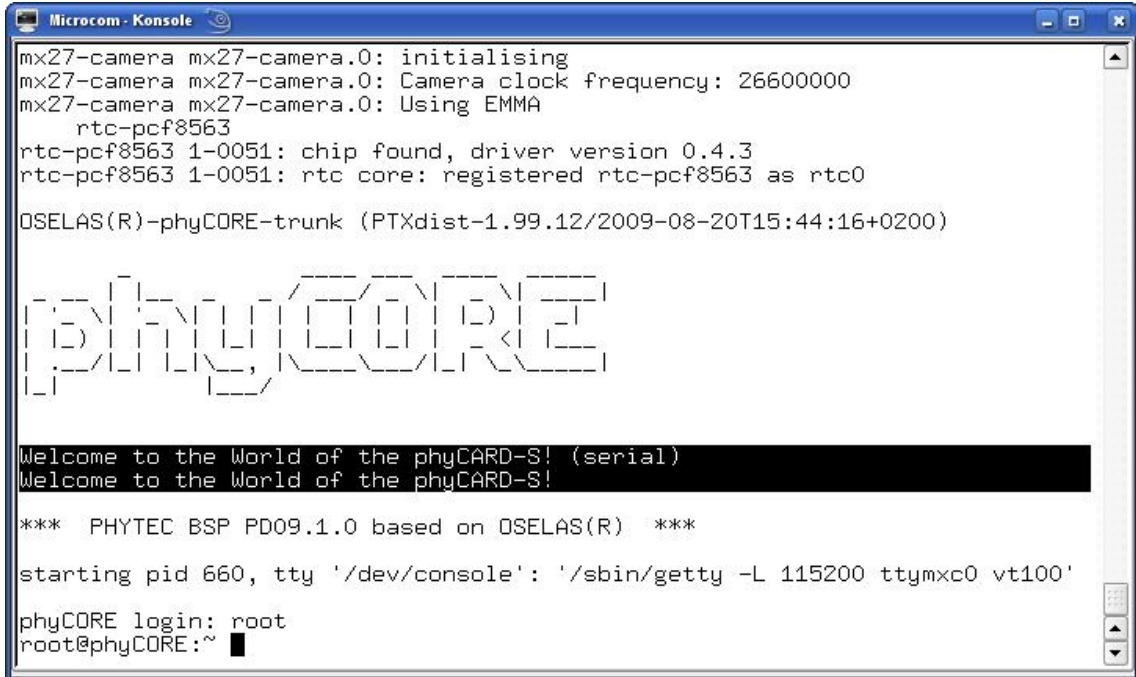


```

telnet - Konsole
Session Edit View Bookmarks Settings Help
root@phyCORE:/etc/rc.d ls -l
lrwxrwxrwx 1 root root 14 Aug 20 2009 S00udev -> ../init.d/udev
lrwxrwxrwx 1 root root 17 Aug 20 2009 S08syslog -> ../init.d/syslogd
lrwxrwxrwx 1 root root 14 Aug 20 2009 S12dbus -> ../init.d/dbus
lrwxrwxrwx 1 root root 17 Aug 20 2009 S16openssh -> ../init.d/openssh
lrwxrwxrwx 1 root root 17 Aug 20 2009 S16telnetd -> ../init.d/telnetd
lrwxrwxrwx 1 root root 20 Aug 20 2009 S21alsa-utils -> ../init.d/alsa-utils
lrwxrwxrwx 1 root root 20 Aug 20 2009 S26networking -> ../init.d/networking
lrwxrwxrwx 1 root root 18 Aug 20 2009 S91lighttpd -> ../init.d/lighttpd
lrwxrwxrwx 1 root root 18 Aug 20 2009 S91pureftpd -> ../init.d/pureftpd
lrwxrwxrwx 1 root root 17 Aug 20 2009 S98modules -> ../init.d/modules
lrwxrwxrwx 1 root root 16 Aug 20 2009 S99banner -> ../init.d/banner
lrwxrwxrwx 1 root root 24 Jan 1 00:16 S99myHelloWorld -> /etc/init.d/myHelloWo
rld
lrwxrwxrwx 1 root root 17 Aug 20 2009 S99startup -> ../init.d/startup
-rwxr-xr-x 1 root root 77 Aug 20 2009 S99zzz_PHYTEC_BSP_version_startup_script
root@phyCORE:/etc/rc.d █

```

- Close the window.
- Open Microcom.
- Push the RESET button on the target to restart your system.



```

Microcom - Konsole
mx27-camera mx27-camera.0: initialising
mx27-camera mx27-camera.0: Camera clock frequency: 26600000
mx27-camera mx27-camera.0: Using EMMA
  rtc-pcf8563
rtc-pcf8563 1-0051: chip found, driver version 0.4.3
rtc-pcf8563 1-0051: rtc core: registered rtc-pcf8563 as rtc0

OSELAS(R)-phyCORE-trunk (PTXdist-1.99.12/2009-08-20T15:44:16+0200)

Welcome to the World of the phyCARD-S! (serial)
Welcome to the World of the phyCARD-S!

*** PHYTEC BSP PD09.1.0 based on OSELAS(R) ***

starting pid 660, tty '/dev/console': '/sbin/getty -L 115200 ttymxc0 vt100'

phyCORE login: root
root@phyCORE:~ █

```

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 phyCARD-S boots.



You have successfully passed the “Getting More Involved” part of this Quick Start.

Chapter 4 Debugging an Example Project

**20 min**

TIME

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 Quick Start 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.

A screenshot of a terminal window titled "Microcom - Konsole". The terminal output shows the following text:

```
mx27-camera mx27-camera.0: Camera clock frequency: 26600000
mx27-camera mx27-camera.0: Using EMMA
rtc-pcf8563
rtc-pcf8563 1-0051: chip found, driver version 0.4.3
rtc-pcf8563 1-0051: rtc core: registered rtc-pcf8563 as rtc0
OSELAS(R)-phyCORE-trunk (PTXdist-1.99.12/2009-08-20T15:44:16+0200)

*** PHYTEC BSP PD09.1.0 based on OSELAS(R) ***

starting pid 658, tty '/dev/console': '/sbin/getty -L 115200 ttyMxc0 vt100'

phyCORE login: root
root@phyCORE:~# gdbserver 192.168.3.11:10000 myHelloWorld
Process myHelloWorld created; pid = 665
Listening on port 10000
```

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

`gdbserver 192.168.3.11:10000 myHelloWorld`

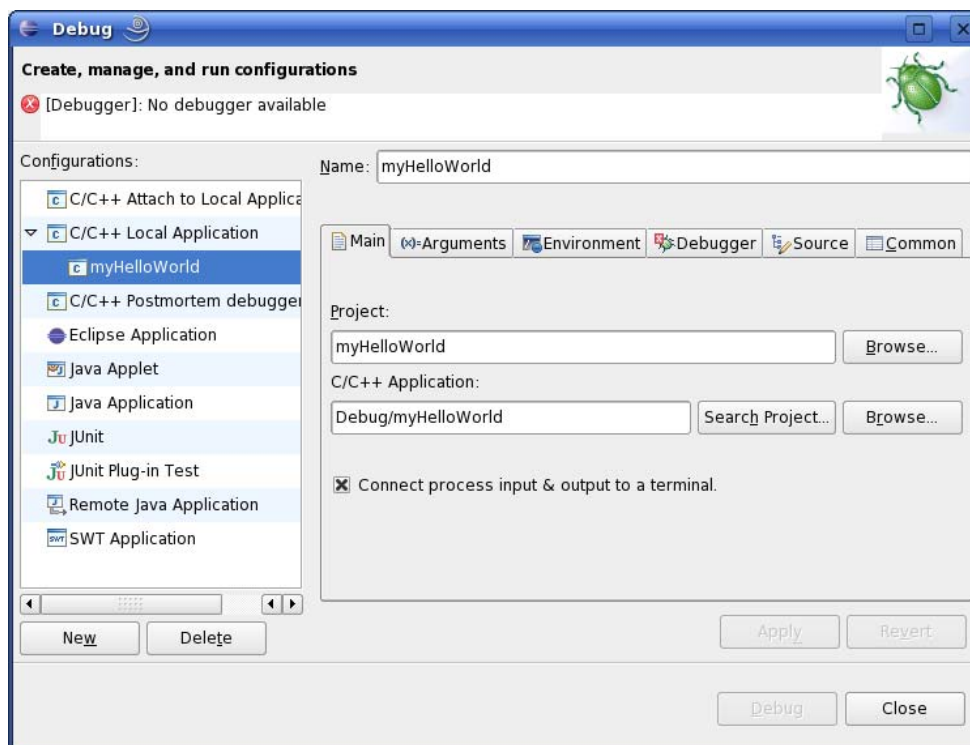
You have started the GDB server on the target. The GDB server is now waiting for connections on TCP port 10000.

4.2 Configuring and Starting the Debugger in Eclipse

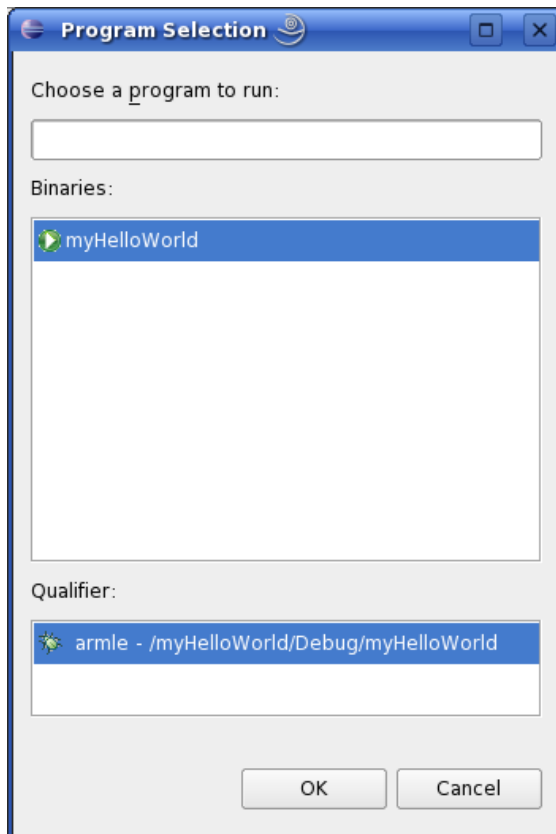
In this passage you will learn how to configure your project settings to use Eclipse with the GNU debugger. After the configuration of your project settings, the GNU debugger will start and connect to the GDB server on the target.

- Start Eclipse if the application is not started yet.
- Select *myHelloWorld* in the *Navigator* window.
- Select *Run* ► *Debug* from the menu bar.

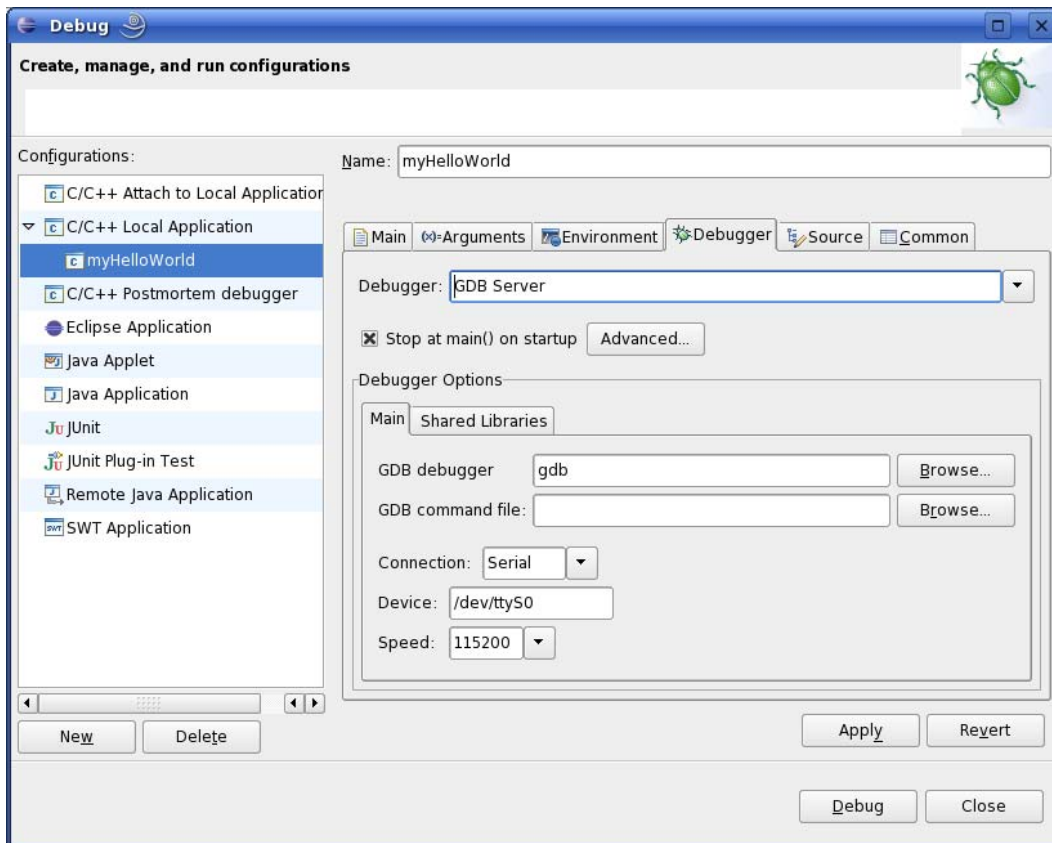
A dialog to create, manage, and run applications will appear.



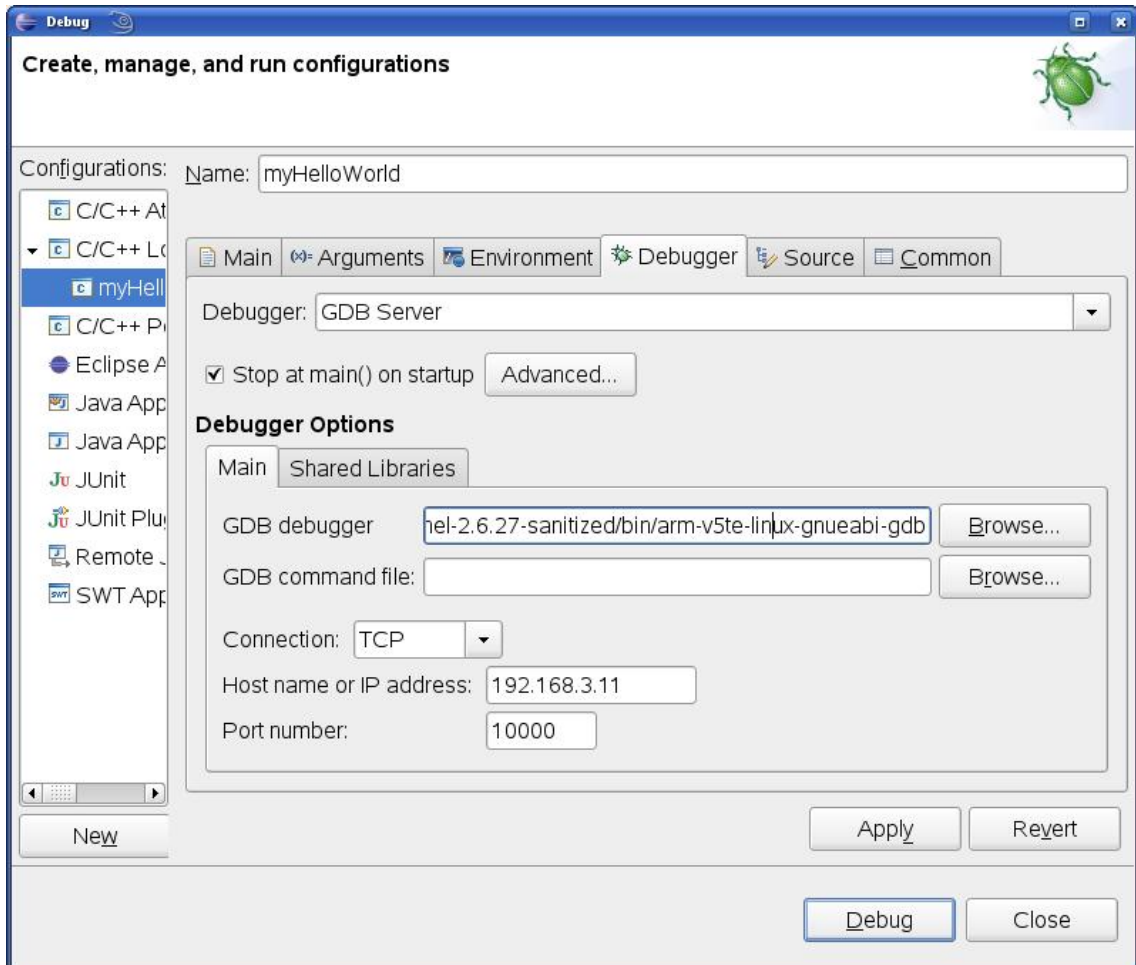
- Select *C/C++ Local Application*.
- Click *New*.



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



- 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-v5te-linux-gnueabi/gcc-4.3.2-glibc-2.8-binutils-2.18-kernel-2.6.27-sanitized/bin`.
- Select the file `arm-v5te-linux-gnueabi-gdb`.
- Click *OK*.



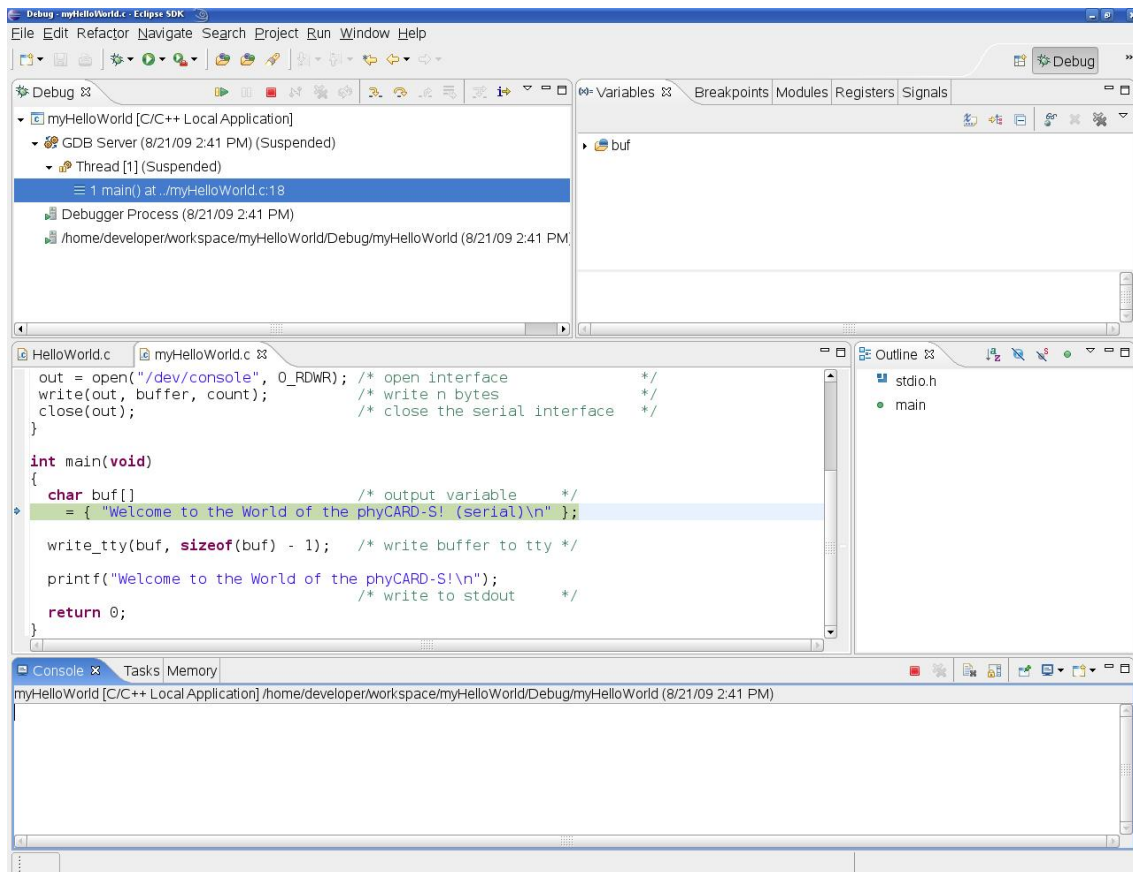
- From the *Connection* drop-down box, select *TCP*.
- Enter **192.168.3.11** (the target's IP address) in the *Host name* input field. The host's GDB will connect to this IP address to communicate with the target's GDB server.
- Click *Apply*.
- Click *Debug*.

A new dialog appears.



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

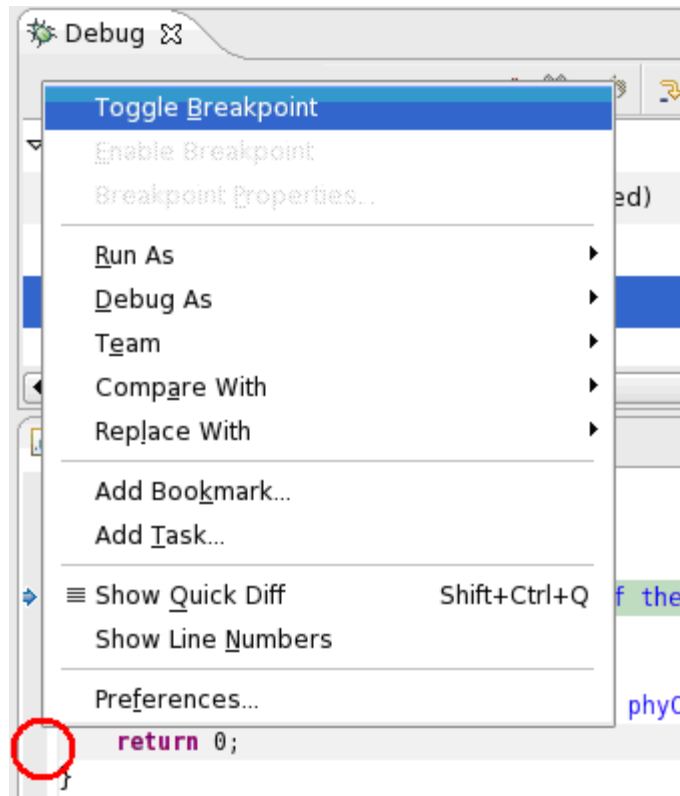
The *Debug* perspective opens and the debugger stops at the first line automatically. The host's GDB is now connected to the GDB server on the target.



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.

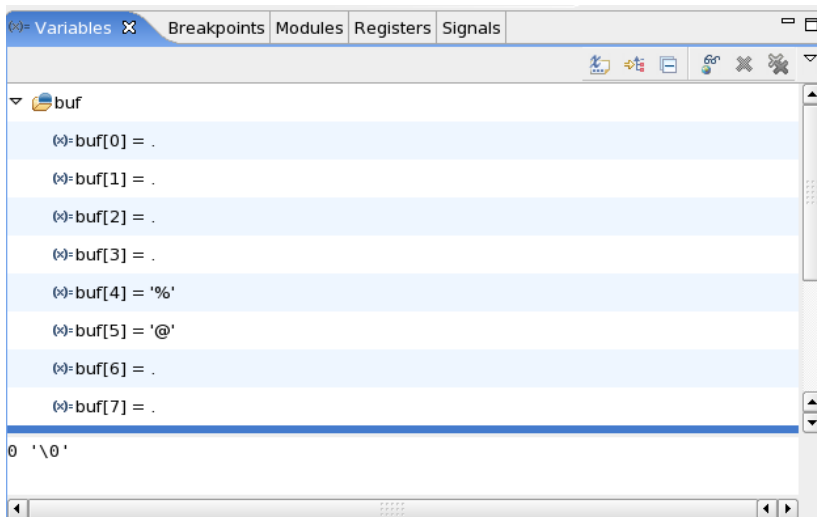



- Select the last line in *main()*.
- Right-click into the small grey border on the left-hand side and select *Toggle Breakpoint* to set a new breakpoint.

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.

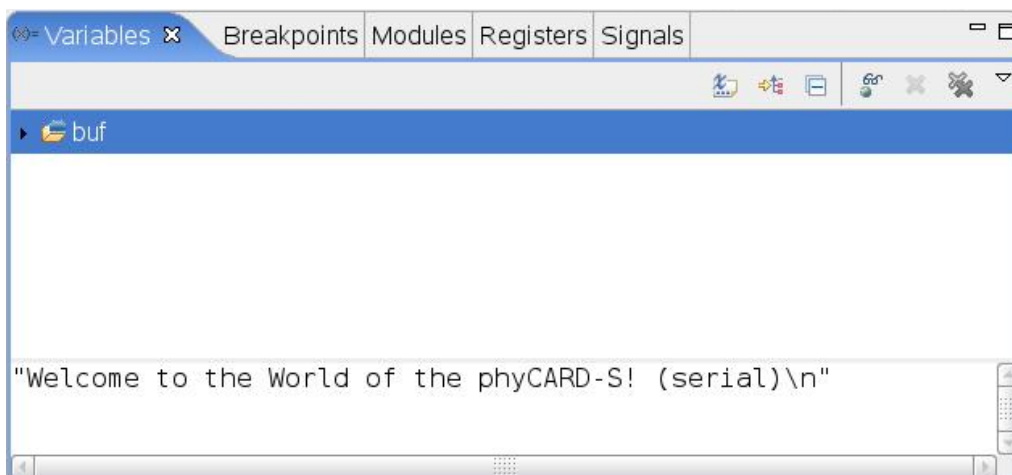
- Expand *buf* in the *Variables* window.




- 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.

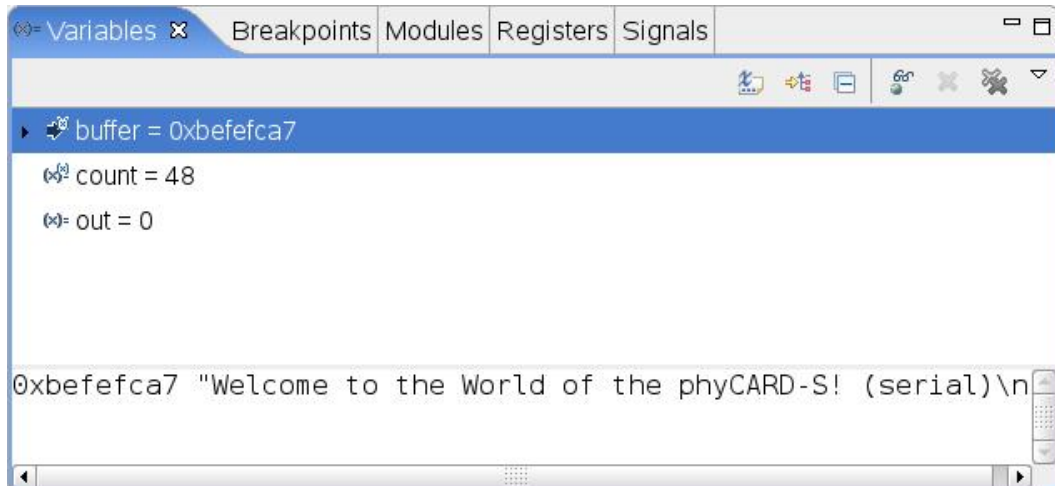
- Click on the variable *buf*.



- 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:

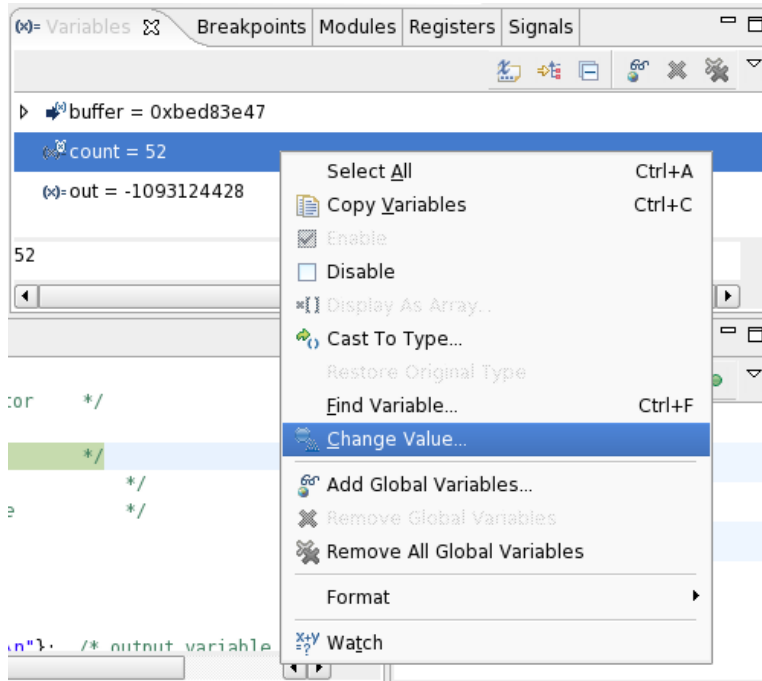


- Click on the variable *buffer*.

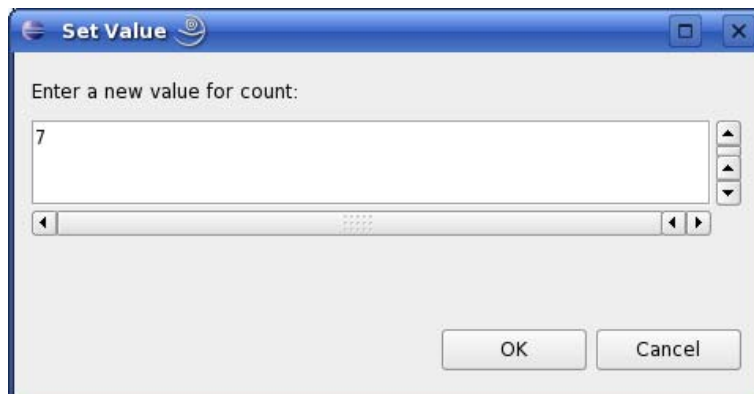
You will probably see a different address at the *buffer* pointer. Remember what address is shown in your case; you will need this address later.


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.



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



- Change the value of *count* to **7** and click *OK*.
- Open Microcom if the application is not already opened.
- Go back to Eclipse.
- Click the *Step Over*  button **two times**.
- Change to Microcom.

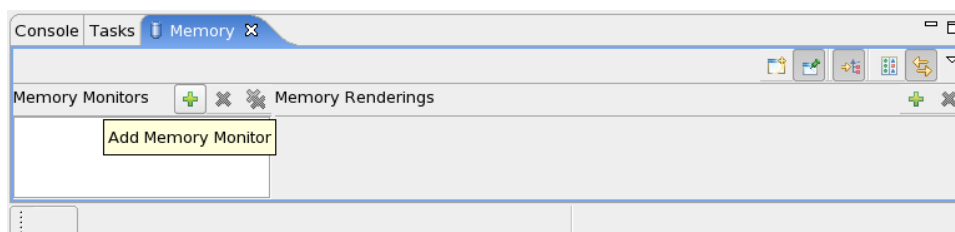

```
root@phyCORE:~# gdbserver 192.168.3.10:10000 myHelloWorld
Process myHelloWorld created; pid = 434
Listening on port 10000

ls
Quit
Remote debugging from host 192.168.3.10
Welcome█
```

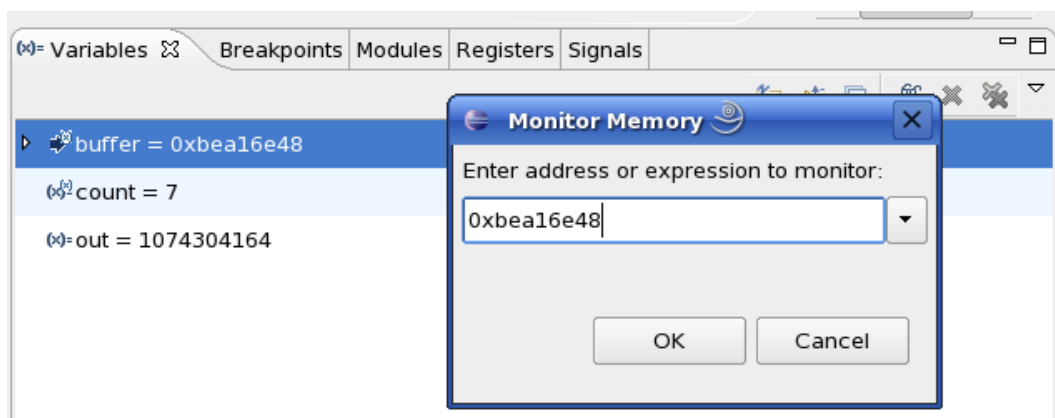
You will see the output *Welcome* in the Microcom window. This means that due to changing the *counter* variable's value, instead of printing the full "Welcome to the World of the phyCARD-S" string, only the first seven characters of the buffer were written to the screen.

4.6 Using the Memory Monitor

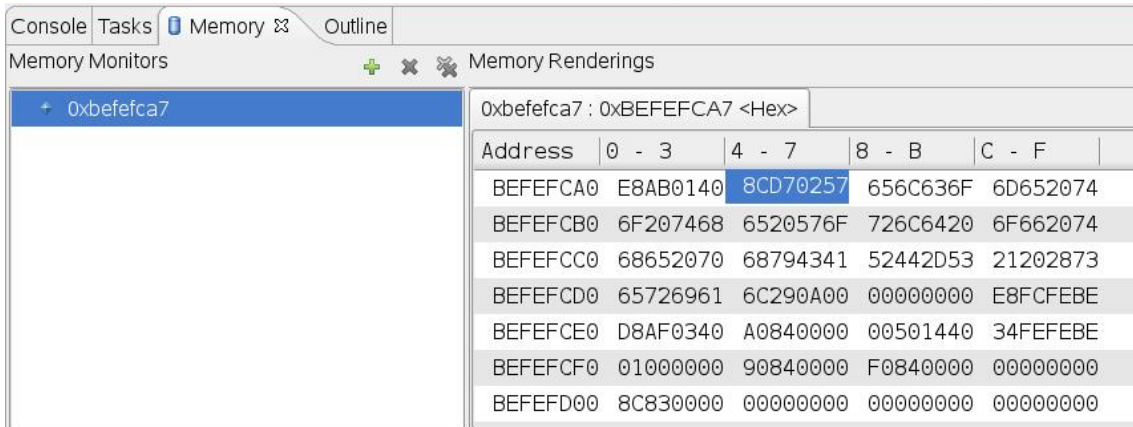
In the last section of this chapter you will use the memory monitor to watch the content at a memory address.



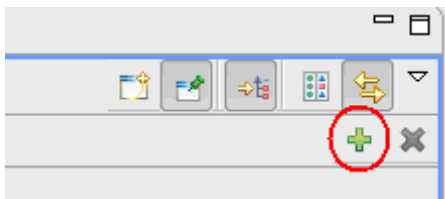
- Select the *Memory* tab.
- Click *Add Memory Monitor*.



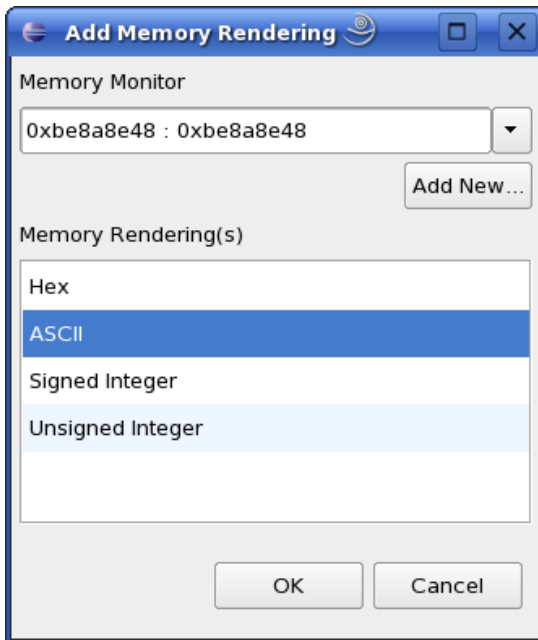
- Enter the address of *buffer* and click on *OK*. Remember that the variable's address might differ on your system.



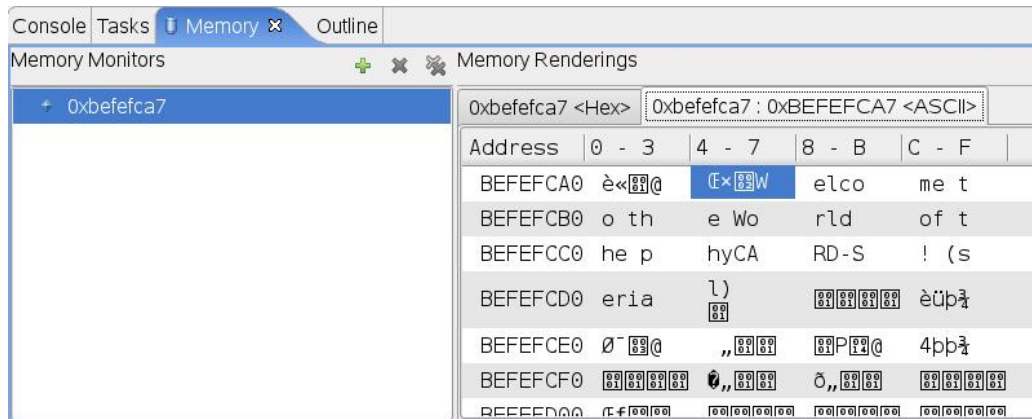
- Change the window size.



- Click *Add Rendering*.



- Select *ASCII* and click *OK*.



You can see the contents of the variable *buffer* at the address *0xbedab6e48* (or whatever address is used on your system).

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

```

HelloWorld.c  *my>HelloWorld.c x
{
  int out; /* variable for file descriptor */
  out = open("/dev/console", 0_RDWR); /* open interface */
  write(out, buffer, count); /* write n bytes */
  close(out); /* close the serial interface */
}

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

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

Chapter 5 **Summary**

This Quick Start 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 phyCARD-S 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 phyCARD-S using Eclipse.

The *Debugging* part of this Quick Start 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.

Chapter 6 Installing Linux on the phyCARD-S

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

6.1 Installing the Boot Loader

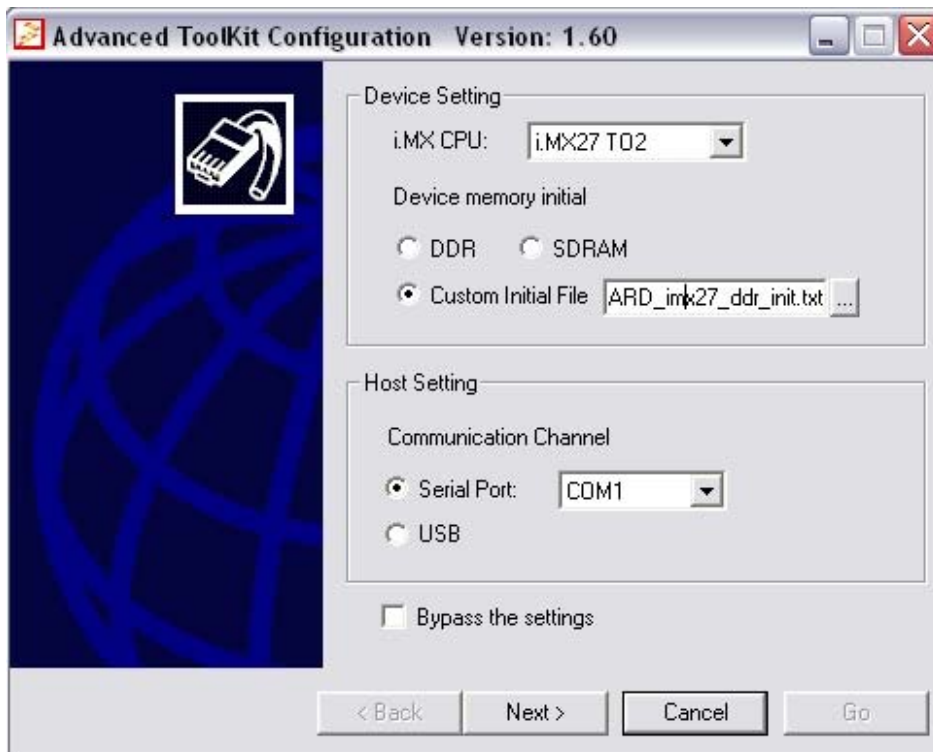
The boot loader used on the phyCARD-S is *U-Boot*, the Universal Boot Loader. The installation of the boot loader will be performed by the *AdvancedToolKit* provided by Freescale. The *AdvancedToolKit* must be executed on a PC running *Microsoft Windows*.

- Insert your PHYTEC Linux-phyCARD-S-Disc, navigate to the *PHYTEC\PCA100 phyCARD-S\Linux-Kit\Software\Tools\ADS_Toolkit* directory, and execute *Setup.exe* to install the *AdvancedToolKit*.
- There are three additional files in the *PHYTEC\PCA100 phyCARD-S\Linux-Kit\Software\Tools\ADS_Toolkit* directory: One ending with *.cfg*, the second ending with *.bin* and the other ending with *.txt*.
- 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.
- Copy the file ending with *.txt* to the *C:\Program Files\Freescale\AdvancedToolKit-STD* directory on your hard disk.
- Configure JP1 by closing PIN 1+2 to boot with UART.
- Connect the UART1 (connector P1) to your computer.

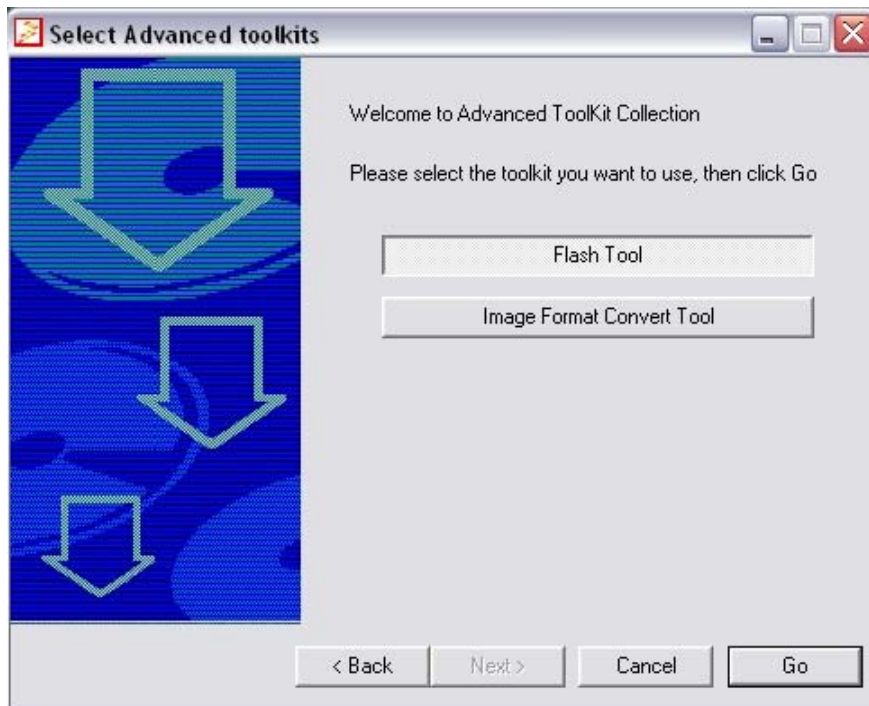
- Power up the baseboard.
- Start the program *AdvancedToolKit*.

The *AdvancedToolKit* dialog opens.

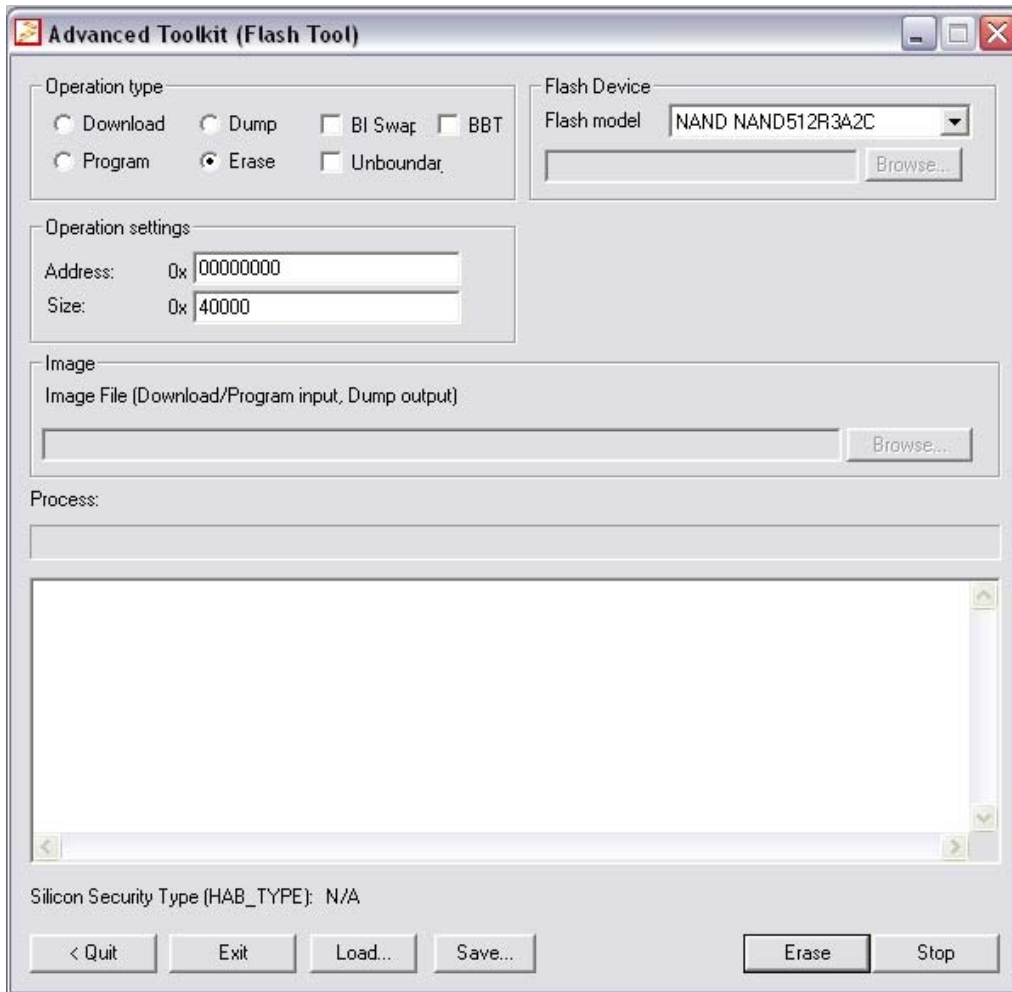
- Choose *i.MX27 TO2* from the *i.MX CPU* drop-down box.
- Select Custom Initial File, press the browse button on the right side of the box and navigate to *C:\Program Files\Freescale\AdvancedToolKit-STD* and choose the file *phyCARD_imx27_ddr_init.txt* which we copied before.
- Select the *Serial Port* which the phyCARD-S is connected to.
- Click *Next*.



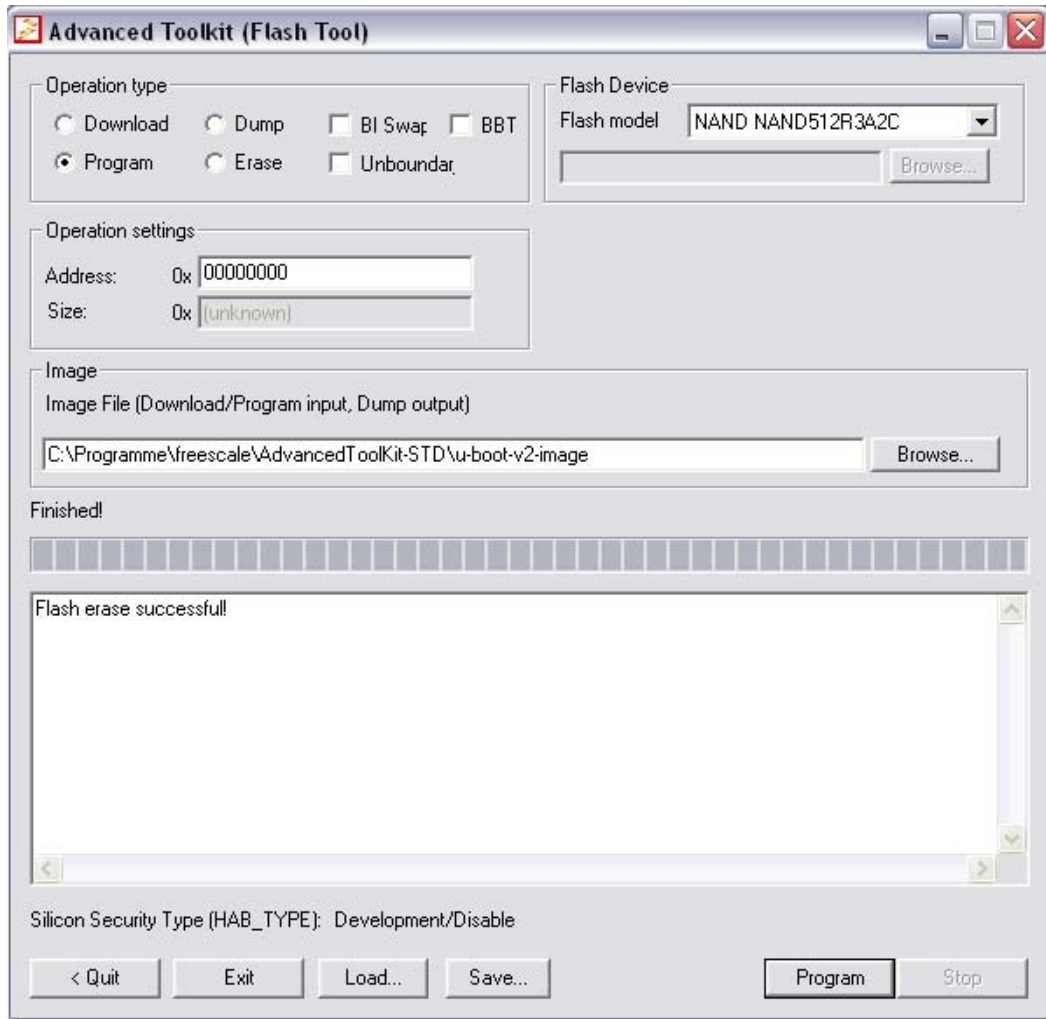
- In the next window, select *Flash Tool*.
- Click *Go* to run the selected tool.



- Select *Erase* in the *Operation type* section.
- Choose NAND NAND512R3A2C from the *Flash model* drop-down box.
- In the *Operating settings* section, set *Address* to **c0000000** 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*.



- 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\PCA100 phyCARD-S\Linux-Kit\BSP\U-Boot\bin` on your setup CD-ROM and choose the *.bin* file you find there.
- Now click *Program*.



- When finished, power down the baseboard.
- Configure JP1 by opening PIN 1+2 to boot with NAND Flash

6.2 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-phyCARD-S-Disc. Information on how to configure the host platform can be found in the *Getting Started* part of this Quick Start.

- Connect the serial cable with the UART1 (connector P1) 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 PWR (12V) on your board.

```

Microcom - Konsole

U-Boot 2.0.0-rc9 (Jul 13 2009 - 16:32:10)

Board: Phytex phyCORE-i.MX27
NAND device: Manufacturer ID: 0x20, Chip ID: 0x36 (ST Micro NAND 64MiB 1,8V 8-bit)
Scanning device for bad blocks

Using environment in NOR Flash
initialising PLLs
chip id: [1,882,1,01d]
mp11:      398999390 Hz
sp11:      239999725 Hz
arm:       398999390 Hz
perclk1:   14777755 Hz
perclk2:   26599959 Hz
perclk3:   66499898 Hz
perclk4:   26599959 Hz
clkkin26:  26000000 Hz
ahb:       132999796 Hz
ipg:       66499898 Hz
Malloc space: 0xa7700000 -> 0xa7f00000 (size 8 MB)
Stack space : 0xa76f8000 -> 0xa7700000 (size 32 kB)
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.
- Type **edit /env/config** to check and 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
eth0.ethaddr=XX:XX:XX:XX:XX:XX █

# can be either 'net', 'nor' or 'nand'
kernel_loc=nor
rootfs_loc=nor

kernel=uImage-pcm038
rootfs=root-pcm038.jffs2
envimage=u-boot-v2-environment-pcm038

autoboot_timeout=3

nfsroot=192.168.3.10/root
bootargs="console=ttyMxc0,115200"

```

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

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 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.

6.3 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:

- **erase /dev/nand0.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.

6.4 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/PCA100 phyCARD-S/Linux-Kit/BSP/Images* on your setup CD-ROM you can find a file called *uImage-pca100* – this file is the Linux kernel image. There is another file, *root-pca100.jffs2*; this file contains the Linux root file system.

- Copy the files *uImage-pca100* and *root-pca100.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 nand uImage-pca100** to download and write the kernel into the target's flash.
- Type **update_rootfs nand root-pca100.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: phyCARD-S with Linux
Quick Start Instructions
Document Number: L-728e_2 October 2009

How would you improve this manual?

Did you find any mistakes in this manual? page

Submitted by:

Customer number: _____

Name: _____

Company: _____

Address: _____

Return to:

PHYTEC Messtechnik GmbH

Robert-Koch-Str. 39

D-55129 Mainz

Fax: +49 (6131) 9221-26

