

# ***System Manual*** ***ECUcore-5208***

## **User Manual** Version 1.00

**Edition August 2009**

Document No.: L-1202e\_1

SYS TEC electronic GmbH August-Bebel-Straße 29 D-07973 Greiz  
Telefon: +49 (3661) 6279-0 Telefax: +49 (3661) 6279-99  
Web: <http://www.systec-electronic.com> Mail: [info@systec-electronic.com](mailto:info@systec-electronic.com)

## Status/Changes

Status: Released

Date/Version	Section	Changes	Editor
2009/08/12 1.0	All	Creation	A. Stein

This manual includes descriptions for copyrighted products that are not explicitly indicated as such. The absence of the trademark (©) symbol does not infer 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, SYS TEC electronic GmbH assumes no responsibility for any inaccuracies. SYS TEC electronic GmbH neither guarantees nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. SYS TEC electronic GmbH reserves the right to alter the information contained herein without prior notification and does not accept responsibility for any damages which might result.

Additionally, SYS TEC electronic GmbH neither guarantees nor assumes any liability for damages arising from the improper usage or improper installation of the hardware or software. SYS TEC electronic 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 SYS TEC electronic GmbH. rights – including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part – are reserved. No reproduction may occur without the express written consent from SYS TEC electronic GmbH.

Inform yourselves:

Contact	Directly	Your local distributor
Address:	SYS TEC electronic GmbH August-Bebel-Str. 29 D-07973 Greiz GERMANY	Please find a list of our distributors under:  <a href="http://www.systec-electronic.com/distributors">http://www.systec-electronic.com/distributors</a>
Ordering Information:	+49 (0) 36 61 / 62 79-0 <a href="mailto:info@systec-electronic.com">info@systec-electronic.com</a>	
Technical Support:	+49 (0) 36 61 / 62 79-0 <a href="mailto:support@systec-electronic.com">support@systec-electronic.com</a>	
Fax:	+49 (0) 36 61 / 6 79 99	
Web Site:	<a href="http://www.systec-electronic.com">http://www.systec-electronic.com</a>	

1st Edition August 2009

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
<b>2</b>	<b>Overview / Where to find what? .....</b>	<b>6</b>
<b>3</b>	<b>Product Description .....</b>	<b>7</b>
<b>4</b>	<b>Development Kit ECUcore-5208.....</b>	<b>9</b>
4.1	Overview.....	9
4.2	Electric commissioning of the Development Kit ECUcore-5208 .....	10
4.3	Control elements of the Development Kit ECUcore-5208.....	11
4.4	Optional accessory: USB-RS232 Adapter Cable .....	12
<b>5</b>	<b>Application and Administration of the ECUcore-5208.....</b>	<b>13</b>
5.1	System requirements and necessary software tools.....	13
5.2	System start of the ECUcore-5208.....	14
5.2.1	Activation/Deactivation of uClinux Autorun.....	14
5.2.2	Autorun for user software .....	15
5.3	Command prompt of the bootloader "CoLilo" .....	16
5.4	Ethernet configuration of the ECUcore-5208 .....	16
5.5	Login to the ECUcore-5208 .....	19
5.5.1	Login to the command shell.....	19
5.5.2	The Internet Service Daemon.....	19
5.5.3	Login to the Telnet Server .....	19
5.5.4	Login to the FTP server .....	21
5.6	Predefined user accounts.....	22
5.7	Adding and deleting user accounts .....	23
5.8	How to change the password for user accounts .....	23
5.9	Setting the system time .....	24
5.10	Readout and displaying "CoLilo" configuration data .....	25
5.11	Showing the installed uClinux-Version .....	26
5.12	File system of the ECUcore-5208 .....	27
5.13	Preinstalled files in the directory "/home" .....	28
5.14	Using the HTTP server .....	29
5.15	Updating the uClinux-Image .....	30
5.16	Updating the bootloader "CoLilo" .....	33
<b>6</b>	<b>VMware-Image with Linux Development System .....</b>	<b>34</b>
6.1	Overview.....	34
6.2	Installing the Linux VMware-Image .....	34
6.3	Starting the Linux VMware-Image .....	34
6.4	User accounts to log in to the Linux development system .....	36
6.5	Determining the IP address of the Linux development system .....	37
6.6	Access to the Linux development system from a Windows computer .....	37
6.6.1	Access via Windows network environment .....	37
6.6.2	Access via Telnet client .....	38
6.7	Personal configuration and actualization of the Linux VMware-Image .....	39
6.7.1	Adjustment of keyboard layout and time zone.....	39
6.7.2	Adjusting the desktop size .....	41
6.7.3	Setting a static IP address for the Linux VMware-Image .....	42
6.7.4	System update of the Linux VMware-Image.....	43
6.7.5	Changing the computer name in the Windows network environment .....	44
6.7.6	Shrinking the VMware-Image .....	44
<b>7</b>	<b>Software Development for the ECUcore-5208 .....</b>	<b>45</b>
7.1	Necessary steps after 1 <sup>st</sup> start.....	45
7.2	Software structure of the ECUcore-5208 .....	45

7.3	Makefile and environment variables to create projects .....	46
7.4	I/O Driver for the ECUcore-5208 .....	47
7.4.1	Integration of the I/O Driver into own user projects .....	47
7.4.2	I/O Driver demo project.....	48
7.5	CAN Driver for the ECUcore-5208 .....	49
7.5.1	Integration of CAN Driver into own user projects .....	49
7.5.2	CAN driver demo project .....	49
7.6	Transferring programs to the ECUcore-5208 .....	51
7.6.1	Using NFS .....	52
7.6.2	Using FTP .....	53
7.6.2.1	ECUcore-5208 as FTP client .....	53
7.6.2.2	ECUcore-5208 as FTP server .....	54
7.7	Compiling and executing the demo project "demo" .....	55
7.7.1	Usage of "make" .....	55
7.7.2	Using graphical IDE "Eclipse" .....	57
7.7.2.1	How to open and edit the demo project .....	57
7.7.2.2	Compiling the demo project .....	59
7.7.2.3	Debugging the demo project in the IDE .....	59
7.8	Configuration and Compiling of uClinux-Image and CoLilo .....	65
<b>8</b>	<b>Testing the hardware connections .....</b>	<b>67</b>
<b>9</b>	<b>Tips &amp; Tricks for Handling Linux resp. uClinux .....</b>	<b>68</b>
	<b>Appendix A: GNU GENERAL PUBLIC LICENSE.....</b>	<b>71</b>
	<b>Index.....</b>	<b>76</b>

# 1 Introduction

Thank you that you have decided for the SYS TEC ECUcore-5208. This product provides to you an innovative and high-capacity single board computer with uClinux operating system. Due to its numerous interfaces on a small manufactured size and due to its low activity input, it is well-suitable as communication and control processor for embedded applications.

Please take some time to read through this manual carefully. It contains important information about the commissioning, configuration and programming of the ECUcore-5208. It will assist you in getting familiar with the functional range and usage of the ECUcore-5208. This document is complemented by other manuals, e.g. for the hardware of the module. Table 1 in section 2 provides a listing of relevant manuals for the ECUcore-5208. Please also refer to those complementary documents.

For more information, optional products, updates et cetera, we recommend you to visit our website: <http://www.systemec-electronic.com>. The content of this website is updated periodically and provides to you downloads of the latest software releases and manual versions.

## Explanatory notes about EMC law for the ECUcore-5208



The ECUcore-5208 has been designed to be used as vendor part for the integration into devices (further industrial processing) or as Development Board for laboratory development (hard- and software development).

After the integration into a device or when changes/extensions are made to this product, the conformity to EMC-law again must be assessed and certified. Only thereafter products may be launched onto the market.

The CE-conformity is only valid for the application area described in this document and only under compliance with the following commissioning instructions! The ECUcore-5208 is ESD-sensitive and may only be unpacked, used and operated by trained personal at ESD-conform work stations.

The ECUcore-5208 is a module for the application in automation technology. It is programmable under uClinux and uses CAN-bus and standard Ethernet network interfaces for various solutions in the automation industry. Moreover, it uses the standardized CANopen network protocol. Due to all those features, the module implicates shorter development times at reasonable hardware costs.

## 2 Overview / Where to find what?

The present document describes the commissioning of the ECUcore-5208 based on the Development Kit ECUcore-5208 as well as general procedures for software development of this module. There are different hardware manuals for all hardware components such as the ECUcore-5208, development boards and reference circuitry. Software-sided, the ECUcore-5208 is delivered with preinstalled uClinux (a special Linux variant without MMU). Hence, applications that are to be run on this module should be programmed as Linux programs. The Kit contains a completely equipped Linux development system in the form of a VMware-Image and therefore allows trouble-free entry into the software development for the ECUcore-5208. The VMware-Image can be used unmodified within different host systems. Table 1 lists up all relevant manuals for the ECUcore-5208.

Table 1: Overview of relevant manuals for the ECUcore-5208

Information about...	In which manual?
Basic information about the ECUcore-5208 (configuration, administration, connection assignment, software development, reference designs et cetera.)	In this manual
Hardware description for the ECUcore-5208, reference designs et cetera	Hardware Manual ECUcore-5208 (Manual no.: L-1137)
Development Board for ECUcore-Modules (Minimum version for all ECUcore-Modules)	Hardware Manual Development Board ECUcores (Manual no.: L-1179)
CAN Driver	CAN Driver Software Manual (Manual-Nr.: L-1023)
Appropriate reference books about the application programming under Linux	<ul style="list-style-type: none"> <li>Advanced Programming in the UNIX Environment, Stevens Rago, Addison-Wesley</li> <li>GNU Function List: <a href="http://www.silicontao.com/ProgrammingGuide/GNU_function_list">http://www.silicontao.com/ProgrammingGuide/GNU_function_list</a></li> </ul>

**Section 4** of this manual describes the **electrical commissioning** of the ECUcore-5208 on the basis of the Development Kit ECUcore-5208.

**Section 5** exemplifies **details about the usage of the ECUcore-5208**, such as configuration and administration of the module, login to the system, Ethernet configuration, the start process and the file system.

**Section 6** describes the **VMware-Image with the Linux development system**.

**Section 7** outlines the **software development** for the ECUcore-5208 and explains the **integration of the I/O Driver** into own applications as well as the procedure for **translating** user-specific programs and their **transfer** onto the module and **debugging**.

**Section 9** provides **tips & tricks** to simplify the usage of Linux. This section is especially helpful for newcomers of using Linux.

### 3 Product Description

The ECUcore-5208 is another innovative product that extends the SYS TEC electronic GmbH product range within the field of control applications. In the form of an insert-ready core module ("Core"), it provides to the user a complete single board computer that is programmable under uClinux and has available several extension interfaces. Due to CAN and Ethernet interfaces, the ECUcore-5208 is best suitable to perform decentralized control tasks.



Figure 1: Top view of the ECUcore-5208

These are some significant features of the ECUcore-5208:

- High-capacity CPU-Kernel (Freescale 32-Bit MCF5208 ColdFire, 166 MHz CPU Tact, 159 MIPS)
- 32 MByte SDRAM Memory, 4 MByte NOR-FLASH Memory, 64 MByte NAND-Flash Memory
- 1x 10/100 Mbps Ethernet LAN Interface (with on-board PHY)
- 1x CAN 2.0B Interface
- 3x Asynchronous Serial Ports (UART)
- 6 digital inputs, 6 digital outputs
- Externally usable SPI or SPI-over-I<sup>2</sup>C and I<sup>2</sup>C
- On-board Peripherals: RTC, temperature sensor, watchdog, power-fail input
- Operating system: uClinux
- Small dimensions (70 x 40 mm)

Making available a complete single board computer as an insert-ready core module with small dimensions, reduces effort and costs significantly for the development of user-specific controls. The PLCcore-5208 is also very well suitable as intelligent network node for decentralized processing of process signals (CANopen and UDP). Additionally, it can be used for motion control applications or as basic component for special assemblies in the industry.

The ECUcore-5208 provides 6 digital inputs (DI0...DI5, 3.3V level), 6 digital outputs (DO0...DO5, 3.3V level), 1 high-speed counter input and 1 PWM/PTO output. Saving user programs in the on-board Flash-Disk of the module allows an automatic restart in case of power breakdown.

Das ECUcore-5208 is based on uClinux as operating system. This allows for simultaneous execution of several user-specific programs.

The uClinux applied to the ECUcore-5208 is licensed under GNU General Public License, version 2. Appendix A contains the license text. All sources of LinuxBSP are included in the VMware-Image of the Linux development system (SO-1096). If you require the LinuxBSP sources independently from the VMware-Image of the Linux development system, please contact our support: [support@systemec-electronic.com](mailto:support@systemec-electronic.com)

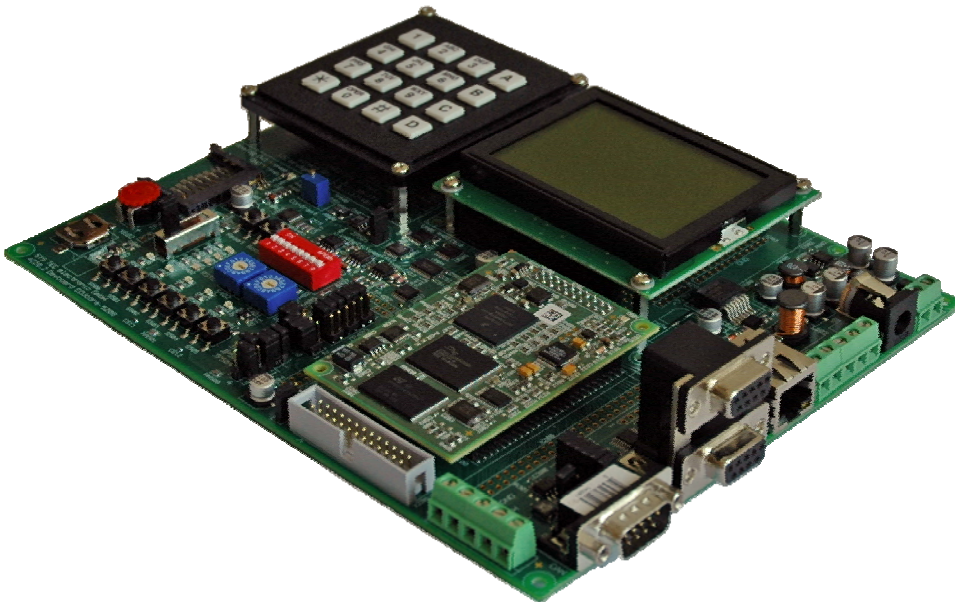




## 4 Development Kit ECUcore-5208

### 4.1 Overview

Due to the Development Board contained in the Kit, the Development Kit ECUcore-5208 allows for a quick commissioning of the ECUcore-5208 and simplifies the design of prototypes for user-specific applications that are based on this module. Among other equipment, the Development Board features several possibilities for power supply, Ethernet interfaces, the connection of 1 CAN buses, 6 push buttons and 5 LED as control elements for the digital in- and outputs. Moreover, it contains a potentiometer for analog input. Signals that are available from plug connectors of the ECUcore-5208 are linked to pin header connectors and enable easy connection of own peripheral circuitry. Hence, the Development Board forms an ideal experimentation and testing platform for the ECUcore-5208.



*Figure 2: Development Kit ECUcore-5208*

The Development Kit ECUcore-5208 ensures quick and problem-free commissioning of the ECUcore-5208. Therefore, it combines all hard- and software components that are necessary to create own applications: the core module ECUcore-5208 itself, the corresponding Development Board containing I/O periphery and numerous interfaces, the Linux development system as well as further accessory. Thus, the Development Kit forms the ideal platform for developing user-specific applications based on the ECUcore-5208.

The Development Kit ECUcore-5208 contains the following components:

- ECUcore-5208
- Development Board for the ECUcore-5208
- 24V Power adapter
- Ethernet cable
- RS232 cable
- DVD with Linux development system, examples, documentation and other tools

The Linux development system included in the Kit serves as software development platform and as debug environment for the ECUcore-5208. In the form of a VMware-Image, the development system can be used unmodified for different host systems. Section 6 exemplifies the usage of the VMware-Image under Windows.

## 4.2 Electric commissioning of the Development Kit ECUcore-5208

The power adapter necessary for running the Development Kit ECUcore-5208 and Ethernet and RS232 cables are already included in the Kit delivery. For commissioning the Kit, it is essential to use at least the power supply connections (X600/X601), COM0 (X304 on top) and ETH0 (X302). Table 2 provides an overview over the connections of the Development Kit ECUcore-5208.

Table 2: Connections of the Development Kit ECUcore-5208

Connection	Labeling on the Development Board	Remark
Power supply	X600 or X601	The 24V DC power adapter included in the delivery is intended for direct connection to X600.
ETH0 (Ethernet)	X302	This interface serves as communication interface with the Linux development system (Programming PC) and can as well be used freely for the user program.
COM0 (RS232)	X304 / on top	This interface is used for the configuration of the unit (e.g. setting the IP-address) and for the diagnosis or debugging. It can be used freely for general operation of the user program.
COM1 (RS232)	X304 / below	Interface can be used freely for the user program.
CAN0 (CAN)	X303	Interface can be used freely for the user program.

Figure 3 shows the positioning of the most important connections of the Development Board for the ECUcore-5208. Instead of using the 24V DC power adapter included in the Kit, the power supply may optionally take place via X600 with an external source of 24V/1A.

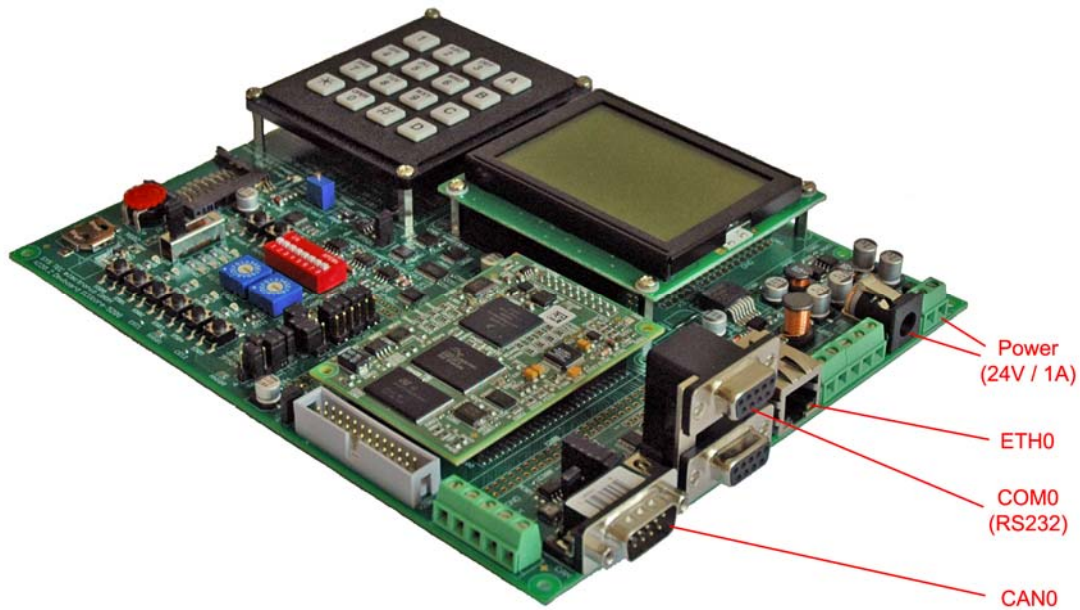


Figure 3: Positioning of most important connections on the Development Board for the ECUcore-5208

**Advice:** Upon commissioning, cables for Ethernet (ETH0, X302) and RS232 (COM0, X304 on top) must be connected prior to activating the power supply (X600 / X601).

### 4.3 Control elements of the Development Kit ECUcore-5208

The Development Kit ECUcore-5208 allows for easy commissioning of the ECUcore-5208. It has available various control elements to configure the module and to simulate in- and outputs for the usage of the ECUcore-5208 as the main component of an industrial control. Table 3 lists the control elements of the Development Board and describes their meaning.

Table 3: Control elements of the Development Board for the ECUcore-5208

Control element	Name	Meaning
Pushbutton 0	S500	Digital Input DI0
Pushbutton 1	S501	Digital Input DI1
Pushbutton 2	S502	Digital Input DI2
Pushbutton 3	S503	Digital Input DI3
Pushbutton 4	S504	Digital Input DI4
Pushbutton 5	S505	Digital Input DI5
LED 0	D500	Digital Output DO0
LED 1	D501	Digital Output DO1
LED 2	D502	Digital Output DO2
LED 3	D503	Digital Output DO3
LED 4	D504	Digital Output DO4

Poti (ADC)	R407	Analog Input AI0
dimnable LED (DAC)	D405	Analog Output AO0
Run/Stop Switch	S405	Control element can be used freely for the user program (e.g. Run / Stop to operate the control program)
Run-LED	D403	Control element can be used freely for the user program (e.g. Display of activity state of the control program)
Error-LED	D402	Control element can be used freely for the user program (e.g. Display of error state of the control program)
Hex-Encoding Switch	S401/402	Control element can be used freely for the user program (e.g. Configuration of node address CAN0)
DIP-Switch	S400	Control element can be used freely for the user program (e.g. Configuration bitrate and master mode CAN0)

#### 4.4 Optional accessory: USB-RS232 Adapter Cable

The SYS TEC USB-RS232 Adapter Cable (order number 3234000) provides a RS232 interface via an USB-Port of the PC. Together with a terminal program, it enables the configuration and diagnosis of the ECUcore-5208 from PCs, e.g. laptop computers which do not have RS232 interfaces themselves (see section 5).



Figure 4: SYS TEC USB-RS232 Adapter Cable

## 5 Application and Administration of the ECUcore-5208

### 5.1 System requirements and necessary software tools

The administration of the ECUcore-5208 requires any Windows or Linux computer that has available an Ethernet interface and a serial interface (RS232). As alternative solution to the on-board serial interface, SYS TEC offers a USB-RS232 Adapter Cable (order number 3234000, see section 4.4) that provides an appropriate RS232 interface via USB port.

All examples referred to in this manual are based on an administration of the ECUcore-5208 using a Windows computer. Procedures using a Linux computer would be analogous.

To administrate the ECUcore-5208 the following software tools are necessary:

- Terminal program**      A Terminal program allows the communication with the **command shell** of the ECUcore-5208 via a **serial RS232 connection to COM0 of the ECUcore-5208**. This is required for the Ethernet configuration of the ECUcore-5208 as described in section 5.4. After completing the Ethernet configuration, all further commands can either be entered in the Terminal program or alternatively in a Telnet client (see below).
- Suitable as Terminal program would be "*TeraTerm*" which is available as Open Source and meets higher demands (downloadable from: <http://tssh2.sourceforge.jp>), PuTTY (downloadable from: <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>) and "*HyperTerminal*" which is included in the Windows delivery or.
- Telnet client**            Telnet-Client allows the communication with **command shell** of the ECUcore-5208 via **Ethernet connection to ETH0 of the ECUcore-5208**. Using Telnet clients requires a completed Ethernet configuration of the ECUcore-5208 according to section 5.4. As alternative solution to Telnet client, all commands can be edited via a Terminal program (to COM0 of the ECUcore-5208).
- Suitable as Telnet client would be "PuTTY" which supports SSH beside Telnet or "*TeraTerm*" which can also be used as Terminal program (see above) "*Telnet*" which is included in the Windows delivery.
- FTP client**              An FTP client allows for file exchange between the ECUcore-5208 (ETH0) and the computer. This allows for example **editing configurations files** by transferring those from the ECUcore-5208 onto the computer where they can be edited and transferred back onto to the ECUcore-5208. Downloading files onto the ECUcore-5208 is also necessary to **update software components**.
- Suitable as FTP client would be "*WinSCP*" which is available as Open Source (download from: <http://winscp.net>). It only consists of one EXE file that needs no installation and can be executed immediately. Furthermore, freeware "*Core FTP LE*" (downloadable from: <http://www.coreftp.com>) or "*Total Commander*" (integrated in the file manager) are suitable as FTP client.
- TFTP server**            The TFTP server is necessary to update the uClinux-Image on the ECUcore-5208. By default, a preconfigured TFTP server is included in the VMware-Image of the Linux development system. To alternatively update the uClinux-Image from a Windows computer, the freeware "*TFTPD32*" (downloadable from: <http://tftpd32.jounin.net>) is suitable. The program only consists of one EXE file that needs no installation and can be executed immediately.

For programs that communicate via Ethernet interface, such as FTP client or TFTP server, it must be paid attention to that rights in the Windows-Firewall are released. Usually Firewalls signal when a program seeks access to the network and asks if this access should be permitted or denied. In this case access is to be permitted.

## 5.2 System start of the ECUcore-5208

### 5.2.1 Activation/Deactivation of uClinux Autorun

During standard operation mode, the bootloader "CoLilo" automatically starts the uClinux operating system of the module after Reset (or Power-on). Afterwards, the operating system loads all further software components and controls the execution of user programs (see section 5.2.2). For service purposes, such as configuring the Ethernet interface (see section 5.4) or updating the uClinux-Image (see section 5.15), it is necessary to disable this Autorun mode and to switch to "CoLilo" command prompt instead (configuration mode).

The automatic boot of uClinux operating system is connected with the **simultaneous compliance** with various conditions ("AND relation"). Consequently, for disabling uClinux Autorun, it is sufficient to simply **not comply** with one of the conditions.

Table 4 lists up all conditions that are verified by the bootloader „CoLilo“. All of them must be complied with to start an Autorun for the uClinux-Image.

Table 4: Conditions for booting uClinux

No.	Condition	Remark
1	"kfl=1" and "auto=1"	The uClinux-Image stored in the Flash must be declared as valid during CoLilo configuration and the Autorun of the image must be activated after Reset (for corresponding CoLilo commands see section 5.4)
2	No interruption of Autorun via COM0 of the ECUcore-5208	If all conditions are met, CoLilo verifies the serial interface COM0 of the ECUcore-5208 for about 1 second after Reset regarding the reception of SPACE signals (ASCII 20H). If such a signal is received within that time, CoLilo will disable the uClinux Autorun and will activate its own command prompt instead.

According to Table 4, the uClinux boot is disabled after Reset (e.g. pushbutton S403 on the Development Board) and the CoLilo command prompt is activated instead if the following conditions occur:

- (1) Reception of a SPACE signal (ASCII 20H) within 1 second after Reset  
- OR -
- (2) Inside CoLilo configuration: kfl=0  
- OR -
- (3) Inside CoLilo configuration: auto=0

After activating the Reset pushbutton (e.g. S403 on the Development Board), the "CoLilo" command prompt answers.

Communication with the bootloader "CoLilo" only takes place via the serial interface COM0 of the ECUcore-5208. As receiver on the computer, one of the terminal programs must be started (e.g. HyperTerminal or TeraTerm, see section 5.1) and must be configured as follows (see Figure 5):

- 19200 Baud
- 8 Data bit
- 1 Stop bit
- no parity
- no flow control

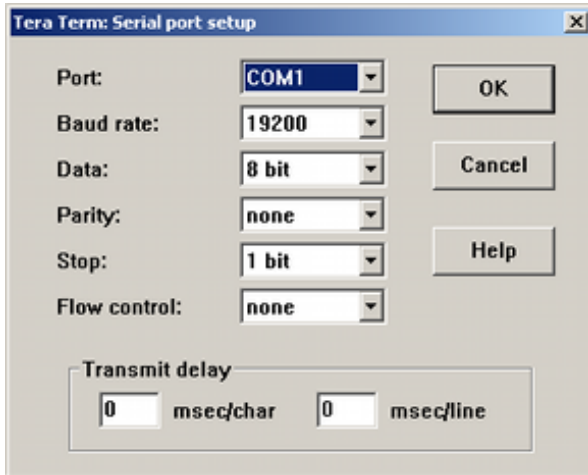
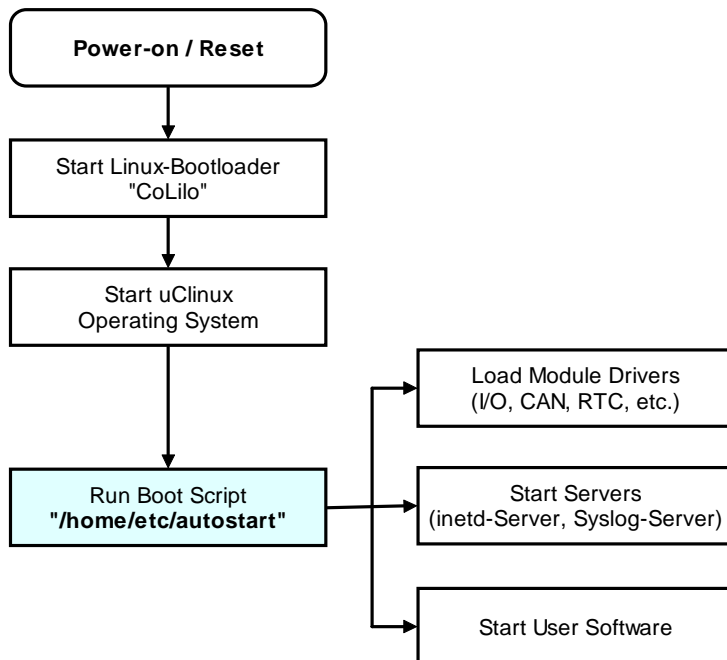


Figure 5: Terminal configuration using the example of "TeraTerm"

### 5.2.2 Autorun for user software

By default, the ECUcore-5208 starts running the uClinux operation system upon Power-on or Reset which loads all necessary software components to execute the user software afterwards. Hence, the ECUcore-5208 is suitable for the usage in autarchic control systems. In case of power breakdown, such systems resume the execution of control programs independently and without user intervention. Figure 6 shows the system start in detail:



For more details on how to deactivate the autarchic uClinux start and to activate the "CoLilo" command prompt compare section 5.2.1.

Details about the programming of the ECUcore-5208 are covered in section 7.

Figure 6: System start of the ECUcore-5208

It is possible to configure the ECUcore-5208 so that the user software starts automatically after Reset. Therefore, all essential commands must be lodged in the start script **"/home/etc/autostart"**. If required, all necessary environment variables can be set and needed drivers can be loaded as well.



The start script `"/home/etc/autostart"` must be adjusted according to desired functionality. The script can be edited directly on the ECUcore-5208 in the FTP client `"WinSCP"` (see section 5.1) using pushbuttons `"F4"` or `"F4 Edit"`.

It has to be regarded that the start of background processes using `&` at the end of the command, but those child process also get terminated when `"/home/etc/autostart"` exits.

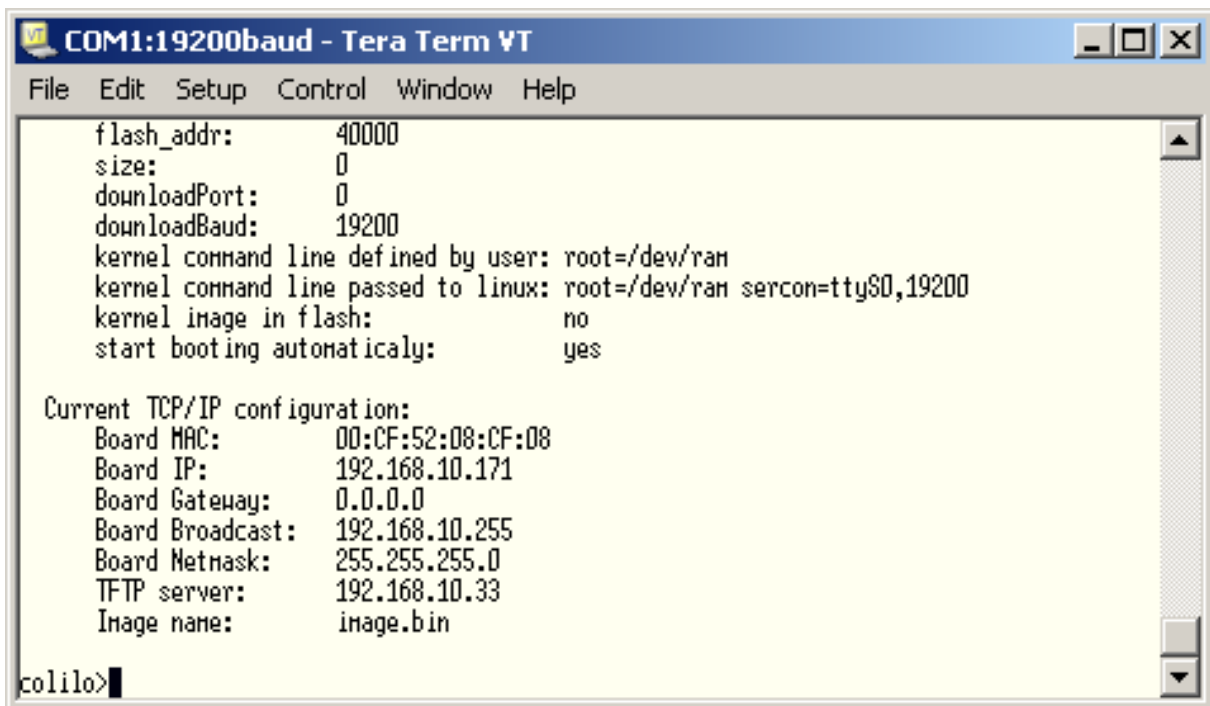
To run a program in background which don't detach themselves from the parent process, it is possible to use the program start-stop-daemon instead.

### 5.3 Command prompt of the bootloader "CoLilo"

After hardware reset of the ECUcore-5208, the bootloader "CoLilo" is the first software component to start immediately. During standard operation mode, the bootloader automatically loads the uClinux operating system that brings about the executions of other user applications. Basic configuration settings for the ECUcore-5208 are determined within the bootloader "CoLilo". The command prompt of the bootloader "CoLilo" is primarily needed for the following tasks:

- Ethernet configuration of the ECUcore-5208 (see section 5.4) and
- Update of the uClinux-Image (see section 5.15)

Section 5.2.1 describes the procedure to activate the "CoLilo" command prompt. Entering `"?"` will show help for all available commands plus the current module configuration (see Figure 7).



```

COM1:19200baud - Tera Term VT
File Edit Setup Control Window Help
flash_addr:      40000
size:            0
downloadPort:    0
downloadBaud:    19200
kernel command line defined by user: root=/dev/ram
kernel command line passed to linux: root=/dev/ram sercon=ttyS0,19200
kernel image in flash:      no
start booting automaticaly: yes

Current TCP/IP configuration:
Board MAC:      00:CF:52:08:CF:08
Board IP:       192.168.10.171
Board Gateway:  0.0.0.0
Board Broadcast: 192.168.10.255
Board Netmask:  255.255.255.0
TFTP server:   192.168.10.33
Image name:    image.bin

colilo>

```

Figure 7: Displaying the current module configuration in the bootloader "CoLilo"

### 5.4 Ethernet configuration of the ECUcore-5208

The main Ethernet configuration of the ECUcore-5208 takes place within the bootloader "CoLilo" and is taken on for all other software components (uClinux, user software, HTTP server etc.). The Ethernet configuration is carried out via the serial interface COM0. **Therefore, the CoLilo command prompt must be activated as explained in section 5.2.1.** Table 5 lists up CoLilo commands necessary for the Ethernet configuration of the ECUcore-5208.

Table 5: "CoLilo" configuration commands of the ECUcore-5208

Configuration	Command	Remark
MAC address	set mac <xx:xx:xx:xx:xx:xx>	The MAC address worldwide is a clear identification of the module and is assigned by the producer. <b>It should not be modified by the user.</b>
IP address of the ECUcore-5208	set ip <xxx.xxx.xxx.xxx>	This command sets the local IP address of the ECUcore-5208. The IP address is to be defined by the network administrator.  To assign a dynamic IP address to the ECUcore-5208 via DHCP, value "0.0.0.0" must be entered as address. <b>Compare the advice about DHCP in the text below!</b>
Network mask	set netmask <xxx.xxx.xxx.xxx>	This command sets the network mask of the ECUcore-5208. The network mask is to be defined by the network administrator.
Gateway address	set gw <xxx.xxx.xxx.xxx>	This command defines the IP address of the gateway which is to be used by the ECUcore-5208. The gateway address is set by the network administrator.  <b>Advice:</b> If ECUcore-5208 and Programming PC are located within the same sub-net, defining the gateway address may be skipped and value "0.0.0.0" may be used instead.
IP address of the Linux development system	set server <xxx.xxx.xxx.xxx>	This command defines the IP address of the Linux development system. It is used for example to update the uClinix-Image (see section 5.15) as well as to include the Linux development system via NFS into the local file system of the ECUcore-5208 (see section 7.6.1).
Saving the configuration	config save	This command saves active configurations in the flash of the ECUcore-5208.

Modified configurations may be verified again by entering "?" in the "CoLilo" command prompt. Active configurations are permanently saved in the Flash of the ECUcore-5208 by command

### **config save**

Modifications are adopted upon next Reset of the ECUcore-5208.

```

COM1:19200baud - Tera Term VT
File Edit Setup Control Window Resize Help

ECUcore-5208 boot...
Press SPACE key to abort autoboot procedure... Autoboot aborted!

colilo>set ip 192.168.10.171
IP address: '192.168.10.171'
colilo>set netmask 255.255.255.0
Netmask: '255.255.255.0'
colilo>config save
Burning 5312 bytes to flash 0x20000 from 0x8000296c
Starting sector: 1
1 128K sector(s) will be erased starting from sector 1, address 0x20000...
Erasing sector 1... ok
Programing...
...
5312 bytes written
colilo>

```

Figure 8: Saving the module configuration of the ECUcore-5208

#### Advice about the application of DHCP:

The uClinux of the ECUcore-5208 is able to request a dynamic IP address via DHCP. Therefore, the local IP address is to be configured as "0.0.0.0":

```
set ip 0.0.0.0
```

The following issues must be taken into consideration for the usage of DHCP:

- DHCP is only supported by uClinux, but not by the bootloader "CoLilo". For example, to update the uClinux-Image (see section 5.15), it is necessary to assign a temporary, static IP address to the module.
- To create a Telnet or FTP connection to the ECUcore-5208 the IP address must be known. The present address that is dynamically assigned by DHCP can be determined using **Terminal programs** (e.g. HyperTerminal or TeraTerm, see section 5.1) via the serial interface **COM0** of the ECUcore-5208. Therefore, logging in to the command shell of the ECUcore-5208 must be accomplished as described in section 5.5.1. Afterwards, the current IP address may be queried using command "*ifconfig*".
- The used DHCP client has the restriction that it cannot run in background during system startup. So it requests just once a dynamic IP address over DHCP which does **not** get renewed when the lease time is expired. It has to be assured otherwise that this IP address doesn't get used for another DHCP client.

**After completing the configuration, all preconditions for a uClinux Autorun must be re-established according to section 5.2.1.**

After Reset (e.g. pushbutton S403 on the Development Board), the module starts with current settings.

**Advice:** After the configuration is finished, the serial connection between the computer and the ECUcore-5208 is no longer necessary.

## 5.5 Login to the ECUcore-5208

### 5.5.1 Login to the command shell

The administration and the software development require the entry of Linux commands in the command shell of the ECUcore-5208. Therefore, the user must be directly logged in at the module. There are two possibilities to login:

- Logging in possible by using a **Terminal program** (e.g. HyperTerminal or TeraTerm, see section 5.1) via the serial interface **COM0** of the ECUcore-5208 – analog to the procedure described for the Ethernet configuration in section 5.4. **For the configuration of the terminal settings pay attention to only use "CR" (carriage return) as end-of-line character.** Login with user name and password is not possible for "CR+LF" (carriage return + line feed)!
- Alternatively, the login is possible using a **Telnet client** (e.g. PuTTY or also TeraTerm, Telnet) via the Ethernet interface **ETH0** of the ECUcore-5208.

### 5.5.2 The Internet Service Daemon

The Internet Service Daemon (inetd for short) can be used for starting several net services if an incoming connection is about to be established. On ECUcore-5208 the inetd services Telnet and FTP. By default all services beside Telnet are disabled for performance and security reasons. To use FTP or Telnet the each service has to be enabled in the inetd configuration file /home/etc/inetd.conf. Therefore several helping scripts for enabling/disabling Telnet and FTP are supplied in the directory "/home/bin".

E.g. the FTP service can be enabled by the following command:

```
enable_ftpd.sh
```

To make the changes effective the inetd service has to be restarted. The following command is necessary:

```
/etc/rc.d/init.d/inetd.sh restart
```

Lines in /home/etc/inetd.conf starting with a '#' sign will be considered as comments and will be ignored. Service lines prefixed with a '#' sign are disabled. If the FTP service is activated the configuration file /home/etc/inetd.conf can also be modified using FTP remote access (with appropriate access permissions like root).

Additionally to Telnet and FTP services a discard service is also provided which is suitable for testing TCP and UDP. All packages will be received and acknowledged if necessary but the data is discarded. This service can cause a high system load and it is recommended to disable it or use it only temporary for testing network connections.

### 5.5.3 Login to the Telnet Server

The Telnet server gets started indirectly by the inetd daemon. To enable Telnet it has to be enabled, if not already done, in the inetd configuration. The proceeding is described in section 5.5.2.

To login on ECUcore-5208 using PuTTY the IP address, configured in section 5.4, has to be entered and the "Connection type" has to be set to "Telnet". By pressing the button "Open" the telnet connection will be established (see figure 9).

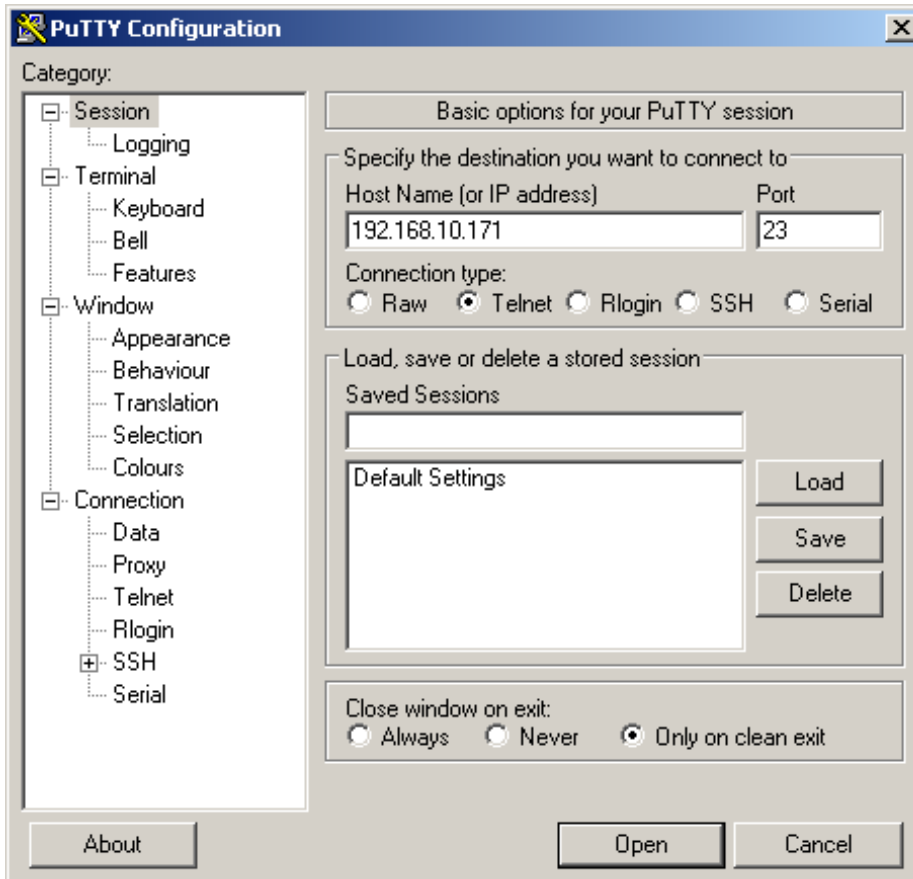


Figure 9: PuTTY Telnet configuration for ECUcore-5208

Logging in to the ECUcore-5208 is possible in the Telnet window (if connected via ETH0). The following user account is preconfigured for the administration of the module upon delivery of the ECUcore-5208 (also compare section 5.6):

User: *PlcAdmin*

Password: *Plc123*

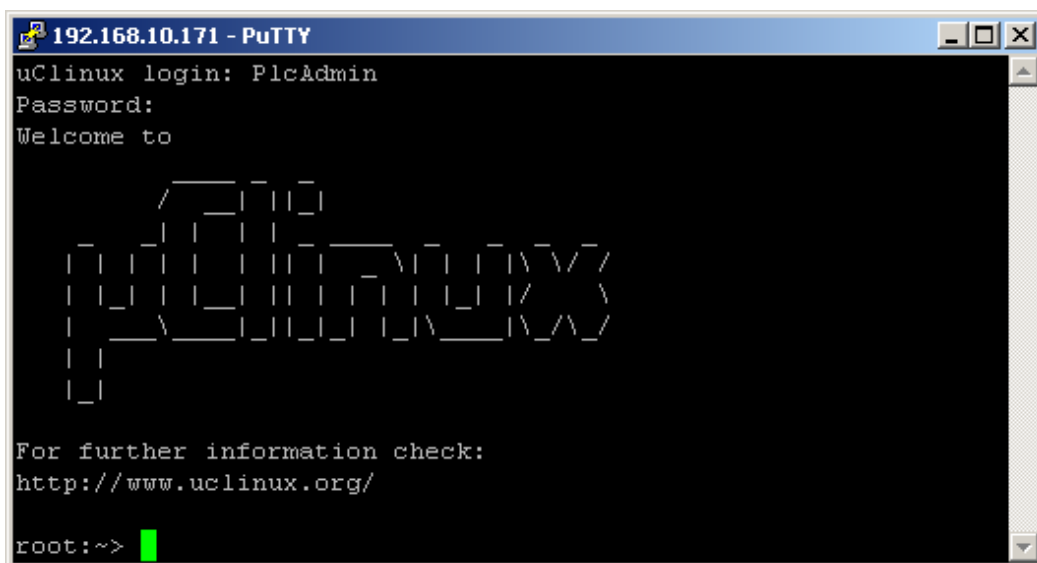


Figure 10: Login to the ECUcore-5208

For logging in to the ECUcore-5208 alternatively via the Windows standard Telnet client, the command "telnet" must be called by using the IP address provided in section 5.4, e.g.

```
telnet 192.168.10.171
```

The login is done the same way as described above using PuTTY.

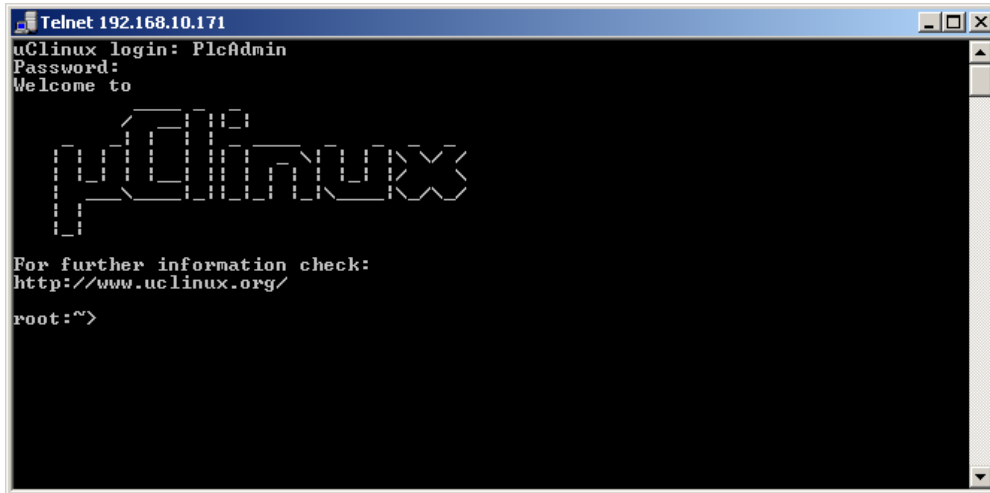


Figure 11: Calling the Telnet client in Windows

**Advice:** The data transmission using Telnet is done in cleartext. That means login name and password are sent unencrypted.

### 5.5.4 Login to the FTP server

The ECUcore-5208 has available a FTP server (FTP Daemon) that allows file exchange with any FTP client (e.g. up- and download of files to or from a computer). Due to security and performance reasons, the FTP server is deactivated by default and must be activated manually if required. As the FTP service is only provided indirectly using inetd, it has to be enabled in the inetd configuration. The proceeding is described in section 5.5.2.

"WinSCP" - which is available as open source - would be suitable as FTP client for the computer (see section 5.1). It consists of only one EXE file, needs no installation and may be started immediately. After program start, dialog "WinSCP Login" appears (see Figure 12) and must be adjusted according to the following instructions:

File protocol:	FTP
Host name:	IP address of the ECUcore-5208 as defined in section 5.4
User name:	PlcAdmin (for predefined user account, see section 5.6)
Password:	Plc123 (for predefined user account, see section 5.6)

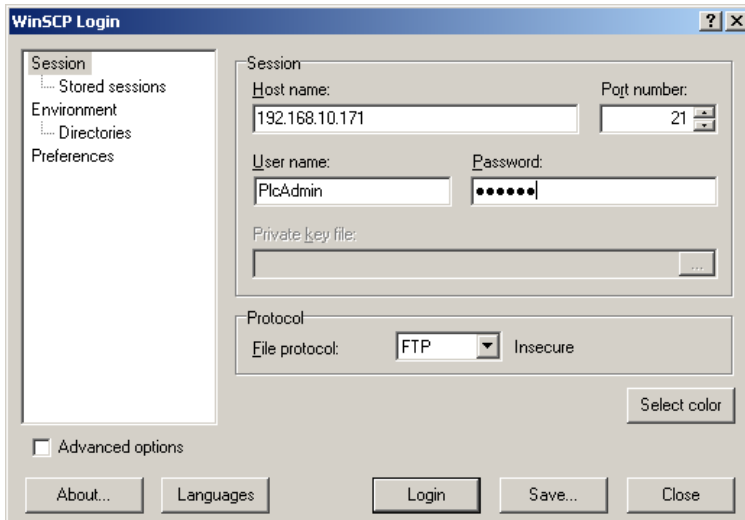


Figure 12: Login settings for WinSCP

After using pushbutton "Login", the FTP client logs in to the ECUcore-5208 and lists up the current content of directory "/home" in the right window. Figure 13 shows the FTP client "WinSCP" after successful login to the ECUcore-5208.

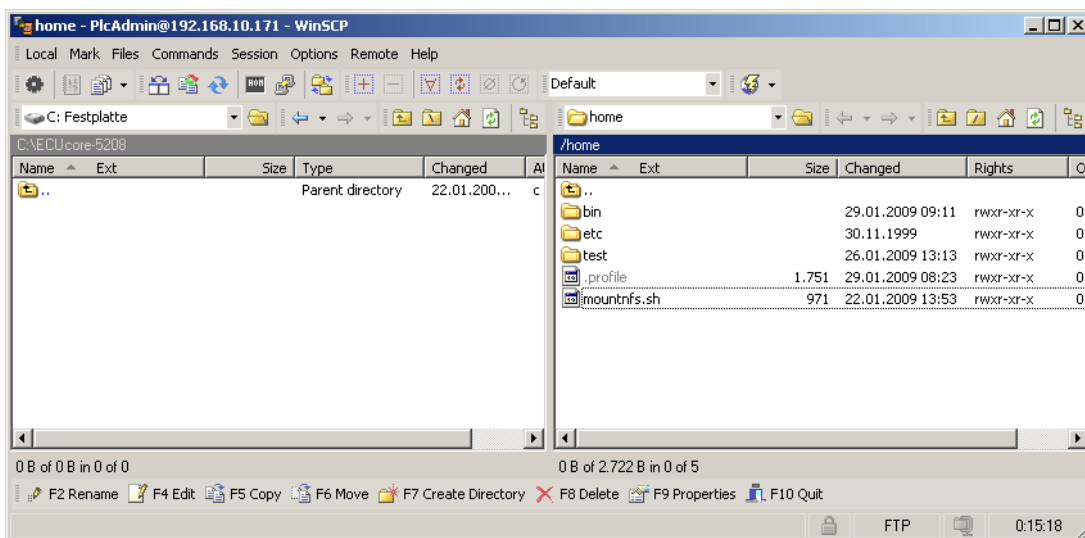


Figure 13: FTP client for Windows "WinSCP"

After successful login, configuration files on the ECUcore-5208 may be edited by using pushbuttons "F4" or "F4 Edit" within the FTP client "WinSCP" (select transfer mode "Text"). With the help of pushbutton "F5" or "F5 Copy", files may be transferred between the computer and the ECUcore-5208, e.g. for data backups of the ECUcore-5208 or to transfer installation files for firmware updates (select transfer mode "Binary").

**Advice:** The data transmission using FTP is done in cleartext. That means login name and password are sent unencrypted.

## 5.6 Predefined user accounts

All user accounts in Table 6 are predefined upon delivery of the ECUcore-5208. Those allow for a login to the command shell (serial RS232 connection or Telnet) and at the FTP server of the ECUcore-5208.

Table 6: Predefined user accounts of the ECUcore-5208

User name	Password	Remark
PlcAdmin	Plc123	Predefined user account for the administration of the ECUcore-5208 (configuration, user administration, software updates etc.)
root	Sys123	Main user account ("root") of the ECUcore-5208

## 5.7 Adding and deleting user accounts

Adding and deleting user accounts requires the login to the ECUcore-5208 as described in section 5.5.1.

**Adding** a new user account takes place via Linux command *"adduser"*. In embedded systems such as the ECUcore-5208, it does not make sense to open a directory for every user. Hence, parameter *"-H"* disables the creation of new directories. By using parameter *"-h /home"* instead, the given directory *"/home"* is rather assigned to the new user. To open a new user account on the ECUcore-5208, Linux command *"adduser"* is to be used as follows:

```
adduser -h /home -H -G <group> <username>
```

Figure 14 exemplifies adding a new user account on the ECUcore-5208 for user *"admin2"*.

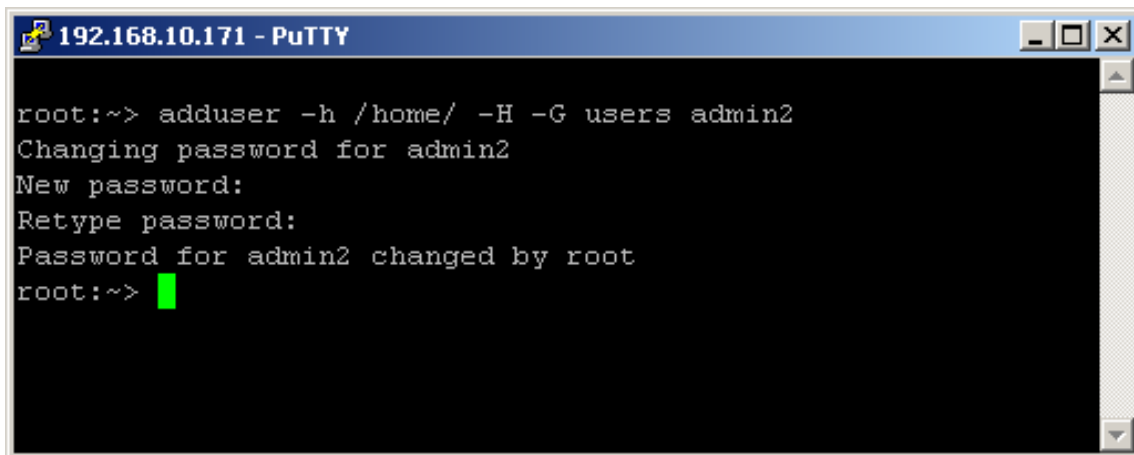


Figure 14: Adding a new user account

To **delete** an existing user account from the ECUcore-5208, Linux command *"deluser"* plus the respective user name must be used:

```
deluser <username>
```

The following message, if it is displayed, during user deletion can be ignored:

```
deluser: can't find <username> in /home/etc/group
```

## 5.8 How to change the password for user accounts

Changing the password for user accounts requires login to the ECUcore-5208 as explained in section 5.5.1.

To change the password for an existing user account on the ECUcore-5208, Linux command *"passwd"* plus the respective user name must be entered:



```
passwd <username>
```

Figure 15 exemplifies the password change for user "PlcAdmin". If the user name is omitted the password of the current user will be changed.

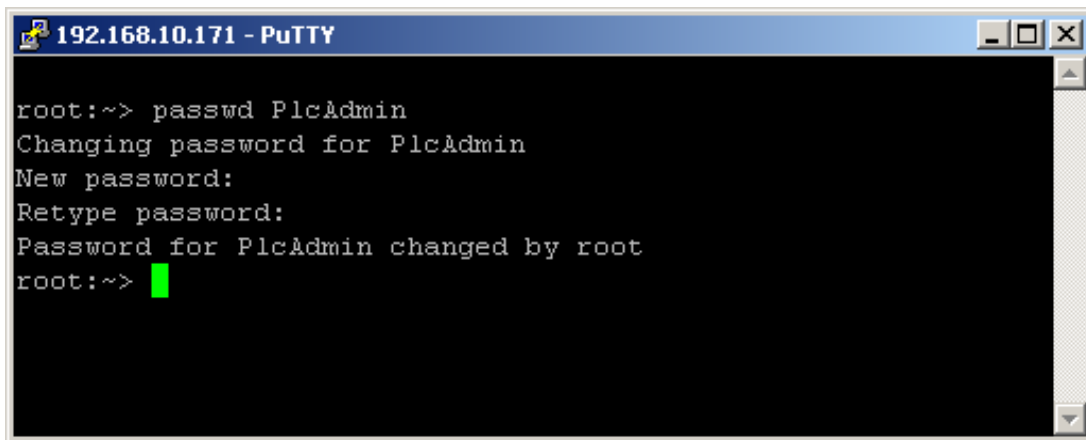


Figure 15: Changing the password for a user account

## 5.9 Setting the system time

Setting the system time requires login to the ECUcore.5208 as described in section 5.5.1.

There are two steps for setting the system time of the ECUcore-5208. At first, the current date and time must be set using Linux command "date". Afterwards, by using Linux command "hwclock -w" the system time is taken over into RTC module of the ECUcore-5208.

Linux command "date" is structured as follows:

```
date [options] [MMDDhhmm[[CC]YY][.ss]]
```

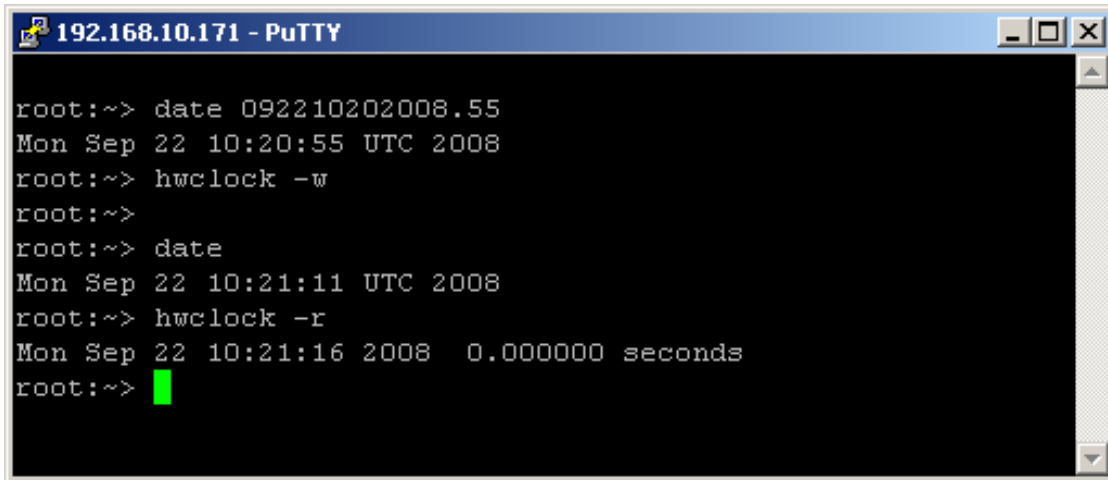
### Example:

```
date    09  22  10  20  2008.55
      |  |  |  |  |
      |  |  |  |  +--- second
      |  |  |  +--- year
      |  |  +----- minute
      |  +----- hour
      +----- day
      +----- month
```

Spaces in the parameter list are only inserted to present the example above clearly. All spaces do not apply when the command is actually used. To set the system time of the ECUcore-5208 to 2008/09/22 and 10:20:55 (as shown in the example above), the following commands are necessary:

```
date 092210202008.55
hwclock -w
```

The current system time is displayed by entering Linux command "date" (without parameter). Linux command "hwclock -r" can be used to recall current values from the RTC. Figure 16 exemplifies setting and displaying the system time.

A screenshot of a PuTTY terminal window titled "192.168.10.171 - PuTTY". The terminal shows a sequence of commands and their outputs. The user sets the system time to "092210202008.55" using the 'date' command. The output shows the current time as "Mon Sep 22 10:20:55 UTC 2008". Then, the user writes the time to hardware using 'hwclock -w'. After a few blank lines, the user reads the time back using 'date', which shows "Mon Sep 22 10:21:11 UTC 2008". Finally, the user reads the hardware time back using 'hwclock -r', which outputs "Mon Sep 22 10:21:16 2008 0.000000 seconds". The prompt "root:~>" is followed by a green cursor.

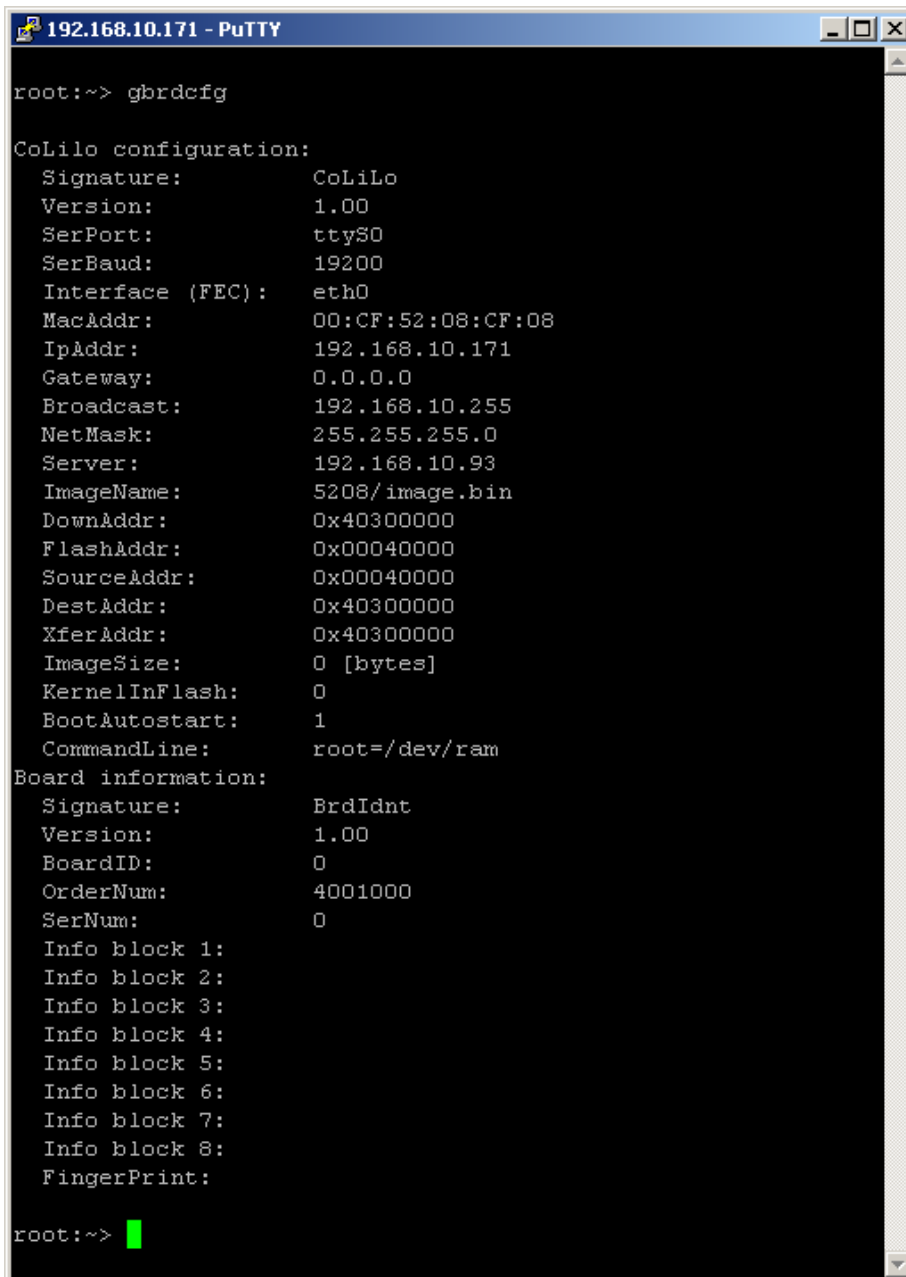
```
root:~> date 092210202008.55
Mon Sep 22 10:20:55 UTC 2008
root:~> hwclock -w
root:~>
root:~>
root:~> date
Mon Sep 22 10:21:11 UTC 2008
root:~> hwclock -r
Mon Sep 22 10:21:16 2008 0.000000 seconds
root:~> █
```

Figure 16: Setting and displaying the system time

Upon start of the ECUcore-5208, date and time are taken over from the RTC and set as current system time of the module.

## 5.10 Readout and displaying "CoLilo" configuration data

Command "*gbrdcfg*" under uClinux enables the access to configuration data of the bootloader "CoLilo". For example, it is used in the start script "*/etc/rc.d/rcS*" to re-use the Ethernet configuration set for the ECUcore-5208 within the "CoLilo" (see section 5.4) to parameterize interface "eth0" under uClinux. The same is essential for taking over the server address (defined in the "CoLilo") into the Shell scripts "*/home/mountnfs.sh*" and "*/home/debug.sh*".



```
192.168.10.171 - PuTTY
root:~> gbrdcfg

CoLilo configuration:
  Signature:      CoLiLo
  Version:        1.00
  SerPort:        ttyS0
  SerBaud:        19200
  Interface (FEC): eth0
  MacAddr:        00:CF:52:08:CF:08
  IpAddr:         192.168.10.171
  Gateway:        0.0.0.0
  Broadcast:      192.168.10.255
  NetMask:        255.255.255.0
  Server:         192.168.10.93
  ImageName:      5208/image.bin
  DownAddr:       0x40300000
  FlashAddr:      0x00040000
  SourceAddr:     0x00040000
  DestAddr:       0x40300000
  XferAddr:       0x40300000
  ImageSize:      0 [bytes]
  KernelInFlash: 0
  BootAutostart: 1
  CommandLine:    root=/dev/ram

Board information:
  Signature:      BrdIdnt
  Version:        1.00
  BoardID:        0
  OrderNum:       4001000
  SerNum:         0
  Info block 1:
  Info block 2:
  Info block 3:
  Info block 4:
  Info block 5:
  Info block 6:
  Info block 7:
  Info block 8:
  FingerPrint:

root:~> █
```

Figure 17: Displaying "CoLilo" configuration data under uClinux using "gbrdcfg"

By calling "gbrdcfg" without indicating parameters, all "CoLilo" configuration data is shown (see Figure 17). By calling "gbrdcfg --help" all parameters for targeted queries of configuration information are listed. The above mentioned scripts illustrate the usage of "gbrdcfg", demonstrate the assignment of requested information to environment variables and show the analysis of information in the Shell script.

## 5.11 Showing the installed uClinux-Version

The uClinux-Version installed on the ECUcore-5208 is shown by calling command "version" (see Figure 18).

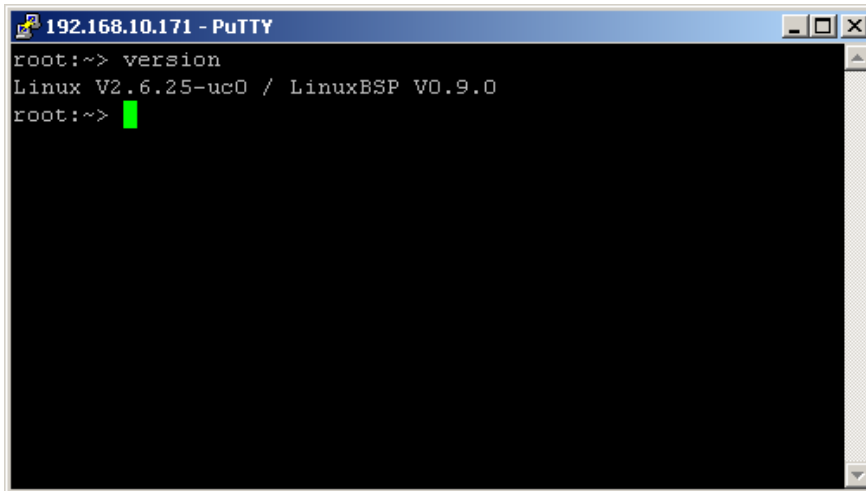


Figure 18: Showing the installed uClinux-Version

## 5.12 File system of the ECUcore-5208

Pre-installed uClinux on the ECUcore-5208 provides part of the system memory in form of a file system. Being usual for embedded systems, most of this file system is “read/only” which means that changes to this part can only be made by creating a new uClinux-Image for the ECUcore-5208. The advantage hereby is the resistance of a read/only file system against damages in case of power breakdowns. Those occur relatively often in embedded systems because embedded systems are usually simply turned off without previous shutdown.

Table 7 lists up writable paths of the file system during runtime. Path **"/home"** comprises a flash disk that provides part of the on-board flash memory of the ECUcore-5208 as file system. This path is used to store all files modifiable and updatable by the user, e.g. configuration files, user programs that have been loaded onto the module. In general the RAM disk directories **"/tmp"** should be used for tests during the development phase – if some parts of the Linux development system are not integrated via NFS anyway (see section 7.6.1).

Table 7: File system configuration of the ECUcore-5208

Path	Size	Description
/tmp	Determined by /var	RAM disk, suitable as intermediate buffer for FTP downloads, but no data preservation in case of power breakdown
/var	2048 kByte	RAM disk which is used by the system to store temporary files, no data preservation in case of power breakdown
/home	2048 kByte	Flash disk to permanently store files modifiable and updatable by the user (e.g. user software and configuration files), data preservation in case of power breakdown
/mnt		Target for integrating remote directories of other systems via NFS, compare section 7.6.1

**Advice:** Size of the file system directories **"/var"** may be modified through uClinux configuration. Therefore a new value for “RAMFS Image” has to be chosen (below “Miscellaneous Configuration -->”). The size of **"/tmp"** doesn’t have to be set explicitly, because it is part of the **/var** tree. It requires a regeneration of the uClinux-Image for the ECUcore-5208.

Sizes of file system paths that are configured or still available can be identified by using the Linux command **"df"** ("DiskFree") – see Figure 19.

```

root:~> df
Filesystem      1k-blocks    Used Available Use% Mounted on
/dev/mtdblock0    1275      1275      0 100% /
/dev/ram1         1979        8    1971   0% /var
/dev/mtdblock4    2048      236    1812  12% /home
root:~>

```

Figure 19: Display of information about the file system

### 5.13 Preinstalled files in the directory "/home"

A Flash disk is mounted to the directory *"/home"*. It provides part of the on-board Flash memory of the ECUcore-5208 as file system. This path is writable during runtime and serves as permanent storage for modifiable and updatable files such as configuration files or user programs (see section 5.12). Upon delivery the ECUcore-5208 includes the following files in the directory *"/home"*:

/home		
+ - etc		
	+ - autostart	Script that automatically starts user software (see section 5.2.2)
	+ - rc.usr	optional user-specific configuration script
	+ - profile	Script is run upon start of a login shell, e.g. it allows for setting environment variables
	+ - resolv.conf	DNS configuration file, it must be adjusted to allow the ECUcore-5208 to resolve DNS names
	+ - passwd	Files for the user administration that are used by programs such as <i>login</i> , <i>adduser</i> , <i>deluser</i> , <i>passwd</i> , <i>tftpd</i> and should only be executed indirectly through those programs
	+ - group	
	+ - inetd.conf	configuration file for inetd
	+ - motd	<b>Message of the Day.</b> Will be displayed after each shell login
+ - bin		
	+ - pc5208drv.ko	Kernelspace-Module of the I/O Driver (see section 7.4.1)
	+ - 5208-spidev.ko	Kernelspace-Module for the I/O Driver
	+ - candrv.ko	Kernelspace-Module of the CAN Driver (see section 7.5.1)
+ - http		Demo files for the HTTP server (compare section 5.14)
+ - mountnfs.sh		Script for a simplified integration of NFS directories into the file system of the ECUcore-5208 (see section 7.6.1)
+ - debug.sh		Script to simplify the start of Demo programs under the control of the Debug server (see section 7.7.2.3)
+ - .profile		user-specific configuration script that is executed with every start of a login shell

If necessary, the delivery status of all files in the directory `"/home"` may be restored by executing the setup script `"setup-ecucore-5208.sh"`. The setup script `"setup-ecucore-5208.sh"` is located in the directory `"SetupFlashdisk_ECUCore-5208"` of the DVD "SO-1096". Section 7.6 describes several possibilities to transfer this file onto the ECUCore-5208.

## 5.14 Using the HTTP server

HTTP server `"boa"` is installed on the ECUCore-5208. Hence, the access to the module is possible via any WEB-Browser (e.g. Mozilla Firefox, Microsoft Internet Explorer, Opera etc.). For the configuration of the server the file `"boa.conf"` is used. Upon starting the server, the directory that contains this file must be given via command line parameter `"-c"`. The delivery status of the ECUCore-5208 includes Demo files in the directory `"/home/http"`. The Demo files illustrate the application of the HTTP server. To activate the Demo configuration, the HTTP server `"boa"` must be started manually by indicating the Demo directory. Therefore, the user must first be logged in to the command shell of the ECUCore-5208 as explained in section 5.5.1. Afterwards, the following command is entered in the Telnet or Terminal window:

```
boa -c /home/http &
```

Figure 20 exemplifies starting the HTTP server `"boa"` for the Demo configuration included in the delivery status of the ECUCore-5208.

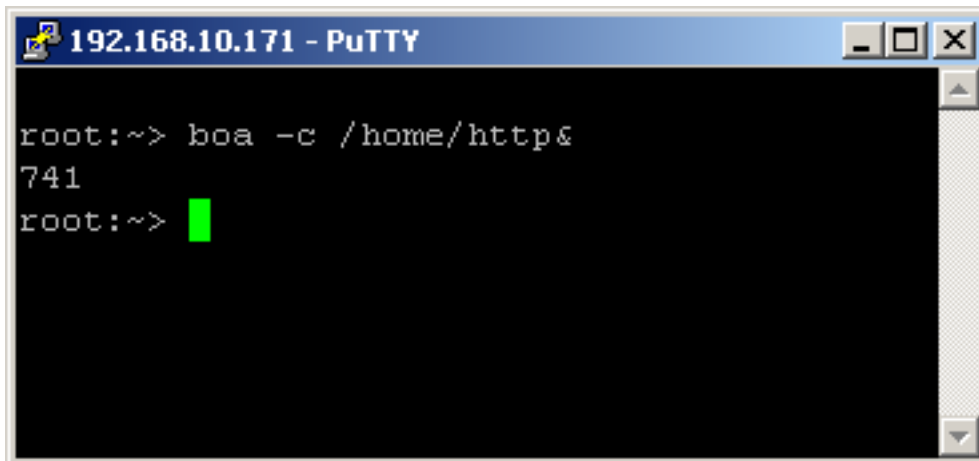


Figure 20: Starting the HTTP server `"boa"`

To call the pages provided by the HTTP server, prefix `"http://"` plus the IP address of the ECUCore-5208 as set in section 5.4 (e.g. `"http://192.168.10.171"`) must be entered in the address bar of the WEB-Browser. Figure 21 shows HTML pages for the ECUCore-5208 in the WEB-Browser.



Figure 21: Display of HTML pages for the ECUcore-5208 in the WEB-Browser

**Advice:** By inserting the start call of the HTTP server (e.g. `"start-stop-daemon -S -b -x boa -- -c /home/http"`) into the start script `"/home/etc/autostart"`, starting the HTTP server upon boot of the ECUcore-5208 may be automated (see section 5.2.2).

## 5.15 Updating the uClinux-Image

Updating the uClinux-Image takes place via TFTP (Trivial FTP) within the Linux bootloader `"CoLilo"`. Therefore, the development computer requires an appropriate TFTP server. By default, such a preconfigured TFTP server is already included in the VMware-Image of the Linux development system.

To alternatively be able to run an update of the uClinux-Image from a Windows computer, the Freeware `"TFTPD32"` could be used for example (see section 5.1). The Windows program only consists of one single EXE file that needs no installation and may be started immediately. After the program start, an appropriate work directory ("Current Directory") should be set up by clicking the pushbutton `"Browse"` (e.g. `"C:\ECUcore-5208"`). This directory must contain the uClinux-Image for the ECUcore-5208 (`image.bin`).

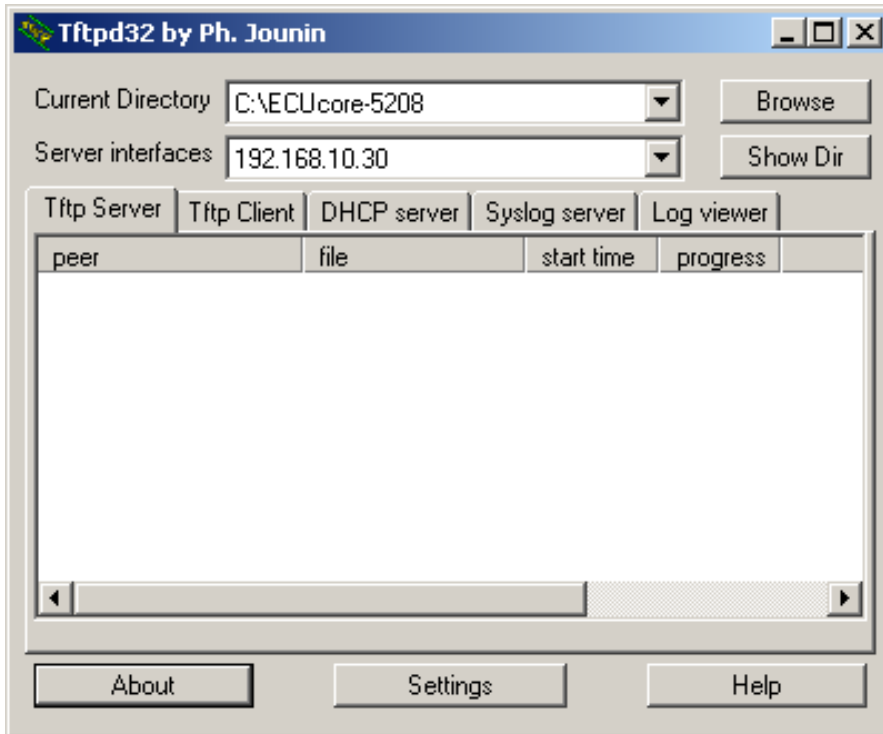


Figure 22: TFTP server for Windows "TFTPD32"

**One requirement** for the TFTP download of the uClinux-Image is a **completed Ethernet configuration** of the ECUcore-5208 **according to section 5.4**. In addition to the Ethernet connection, a serial connection to the ECUcore-5208 is necessary to update the uClinux-Image. Therefore, all setups for the terminal program as described in section 5.2.1 are valid (19200 Baud, 8 Databit, 1 Stopbit, no parity, no flow control).

**An update of the uClinux-Image on the ECUcore-5208 is only possible if uClinux is not yet running. Hence, prior to updating the uClinux-Image the uClinux Autorun must be prevented and it must be switched to the "CoLilo" command prompt instead. All necessary steps are covered in section 5.2.1.**

After Reset (e.g. pushbutton S403 on the Development Board) the "CoLilo" command prompt answers. To update the uClinux-Image all commands explained in the following table must be entered in the same sequence as provided below:



Table 8: Command sequence to update the uClinux-Image on the ECUcore-5208

Command	Meaning
<code>set server &lt;host_ip_addr&gt;</code>	Setting the IP address of the TFTP server. If the TFTP server is used in the VMware-Image of the Linux development system, the necessary IP address must be determined using Linux command " <i>ifconfig</i> " (see section 6.5), if the Windows program " <i>TFTPD32</i> " is used, the IP address is shown on the computer in array "Server Interface"
<code>set image image.bin</code>	Setting the file names for the uClinux-Image that must be loaded
<code>tftp 0x40300000</code>	Downloading the uClinux-Image from the computer onto the ECUcore-5208
<code>flash 0x40000</code>	Saving the uClinux-Image in the Flash of the ECUcore-5208
<code>set kfl 1</code>	Declaring the uClinux-Image saved in the Flash of the ECUcore-5208 as valid
<code>set auto 1</code>	Activating the Autorun for the uClinux-Image after Reset
<code>config save</code>	Storing the current settings in the Flash

```

COM1:19200baud - Tera Term VT
File Edit Setup Control Window Resize Help

ECUcore-5208 boot...
colilo>set server 192.168.10.33
TFTP Server: '192.168.10.33'
colilo>set image image.bin
Image: 'image.bin'
colilo>tftp 0x40300000
TFTP download:
Download address: 40300000
Using FEC: 0
Board MAC: '00:CF:52:08:CF:08'
Board IP: '192.168.10.171'
Board netmask: '255.255.255.0'
Board gateway: '0.0.0.0'
TFTP server: '192.168.10.33'
Image: 'image.bin'
Setup FEC...
Initializing TCP/IP stack... Ready
TFTP transfer completed
Read 3092484 bytes (6041 blocks)

Image size = 3092484/0x2f3004 bytes
colilo>flash 0x40000

```

Figure 23: Downloading the uClinux-Image onto the ECUcore-5208

After finishing the configuration, all preconditions for the uClinux Autorun must be re-established according to section 5.2.1.

After Reset (e.g. pushbutton S403 on the Development Board), the ECUcore-5208 starts using the updated uClinux-Image.

**Advice:** After the configuration is finished, the serial connection between the computer and the ECUcore-5208 is no longer necessary.

## 5.16 Updating the bootloader "CoLilo"

Updating the "CoLilo" bootloader is basically possible, but it involves the risk that the ECUcore-5208 will not boot anymore if an update failed or carried defective software. For this reason, all flash sectors in which the "CoLilo" is integrated are protected from accidental deletion.

That's why the bootloader "CoLilo" should only be updated if absolutely essential. If required please contact our support service:

[support@systemec-electronic.com](mailto:support@systemec-electronic.com)

## 6 VMware-Image with Linux Development System

### 6.1 Overview

The ECUcore-5208 is delivered with a preinstalled uClinux. Hence, all applications that shall run on the module must be developed as Linux programs. The Kit is equipped with a complete Linux development system in the form of a VMware-Image. It allows for an easy introduction into software development for the ECUcore-5208. The VMware-Image may be used unmodified in different host systems. Table 1 in section 2 lists well-suited reference works about Linux programming.

The VMware-Image of the Linux development system includes the following software components:

- GNU-Crosscompiler Toolchain for Freescale MCF52xx-Processors
- Linux-Sourcecode for the ECUcore-5208 (LinuxBSP)
- Eclipse (graphic IDE to simplify the software development)
- Samba server (enables the access "from outside" via Windows network environment)
- FTP server (enables the usage of Linux-Console "from outside", in the form of a Telnet client under Windows as well as data exchange between the ECUcore-5208 and the development computer)
- TFTP server (enables downloading the uClinux-Image for the ECUcore-5208 from the development computer)
- NFS server (enables the integration of the development computer into the local file system of the ECUcore-5208)

### 6.2 Installing the Linux VMware-Image

The Development Kit ECUcore-5208 contains the DVD "SO-1096" which includes the VMware-Image for the Linux development system of the ECUcore-5208 as well as the "VMware Player" for Windows. The "VMware Player" is free-of-charge software for a desktop virtualization so that the VMware-Image of the Linux development system may be executed on a Windows or Linux computer. If required, active versions of the "VMware Player" or Players for other host systems can be downloaded directly from the manufacturer's website <http://www.vmware.com>. Execute the appropriate setup program to install the VMware Player.

**Advice:** During installation of the "VMware Player", standard setup "Bridged Mode" should be retained for the Ethernet interface. Otherwise the communication to the ECUcore-5208 could possibly be defective.

By the means of "**SO-1096.exe**" the VMware-Image included on the DVD unpacks itself. If "**SO-1096.exe**" is started, all files corresponding to the VMware-Image are unpacked onto the local hard disk. That decompressed image requires about 10 GByte.

### 6.3 Starting the Linux VMware-Image

Initially, the "VMware Player" must be started on the host computer. Open file "Xubuntu-ECUcore5208.vmx" in the program window of the Player by using the "Open" symbols (see Figure 24).



Figure 24: Program window of the VMware Player using the "Open" symbol

By executing an image, VMware saves the "Finger Print" of the host computer in form of an UUID in the file \*.vmx. If the Linux VMware-Image is started on another computer, the dialog as shown in Figure 25 appears. If the same Linux-Image is not executed on another computer at the same time in the same network, option "I moved it" should be selected. In doing so, the MAC address of the virtual network card that was used so far, remains valid in the host system. If "I copied it" is set, the VMware generates a new MAC address for the host system. This may involve that a new IP address is assigned to the development system. The Linux-Image is configured in a way so that it dynamically requests an IP address via DHCP client.

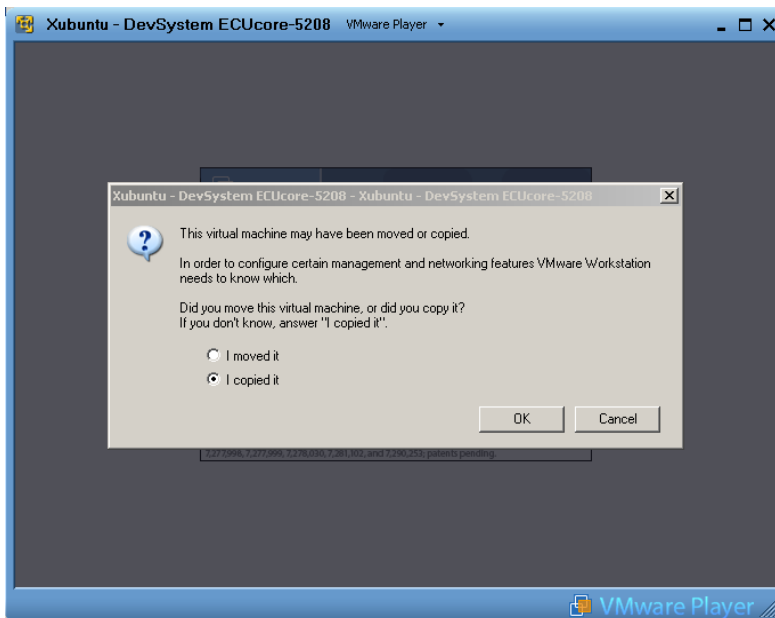


Figure 25: VMware selection dialog to generate or remain the MAC address

Figure 26 shows the desktop of a Linux development system after starting the Linux-Image in the "VMware Player".

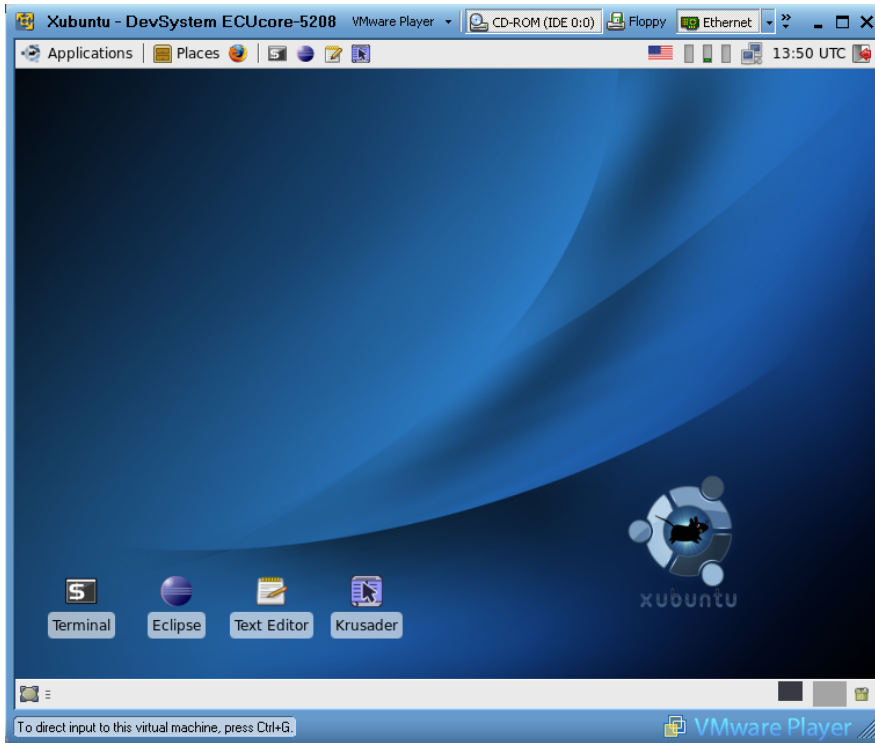


Figure 26: Desktop of the Linux development system

## 6.4 User accounts to log in to the Linux development system

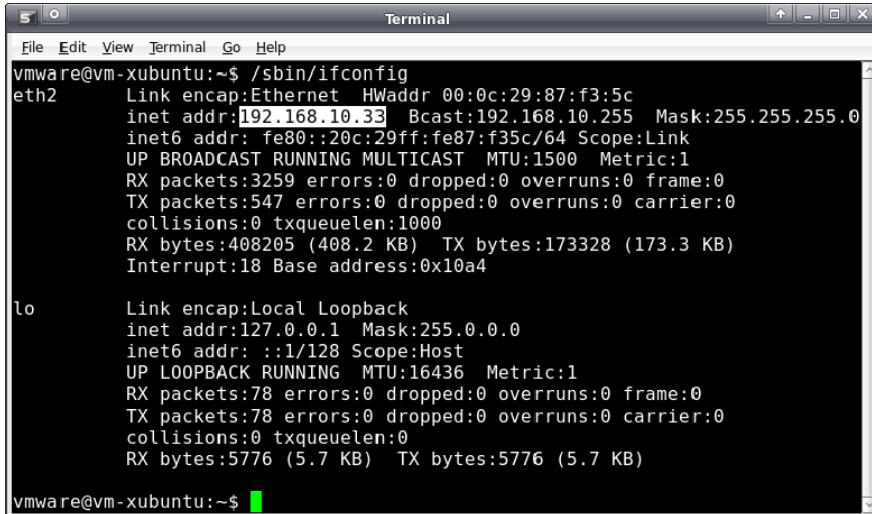
Table 9 lists up all predefined user accounts for logging in to the Linux development system.

Table 9: Predefined user accounts of the Linux development system

Login	User information	Remark
Local console / Terminal (normal user rights)	User: vmware Password: vmware	Predefined user account within the Linux development system
Administrator rights	Command: sudo Password: vmware	The Linux development system that is used does not support explicit login as "root"; to execute a command with administrator rights Linux command "sudo" can be put in front if required, e.g. "sudo cat /etc/shadow"
Windows network environment	Group: Workgroup Computer: Vm-xubuntu User: vmware Password: vmware	Predefined user account to access the Linux development system via Windows network environment (Samba)
Telnet access	User: vmware Password: vmware	Predefined user account to login to the Linux development system via a Telnet client (e.g. Telnet client under Windows)

## 6.5 Determining the IP address of the Linux development system

To determine the IP address that is assigned to the Linux development system via DHCP, a console window must be started in Linux (symbol "Terminal"). After entering command *ifconfig*, among other things the IP address of the Linux-Image is displayed (marked with color in screenshot Figure 27).



```

vmware@vm-xubuntu:~$ /sbin/ifconfig
eth2      Link encap:Ethernet  HWaddr 00:0c:29:87:f3:5c
          inet addr:192.168.10.33  Bcast:192.168.10.255  Mask:255.255.0
          inet6 addr: fe80::20c:29ff:fe87:f35c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3259 errors:0 dropped:0 overruns:0 frame:0
          TX packets:547 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:408205 (408.2 KB)  TX bytes:173328 (173.3 KB)
          Interrupt:18 Base address:0x10a4

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:78 errors:0 dropped:0 overruns:0 frame:0
          TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:5776 (5.7 KB)  TX bytes:5776 (5.7 KB)

vmware@vm-xubuntu:~$

```

Figure 27: Determining the IP address of the Linux development system

## 6.6 Access to the Linux development system from a Windows computer

### 6.6.1 Access via Windows network environment

The access to files in the Linux development system via Windows network environment is possible by using the Samba server that is installed in the VMware-Image. This allows for comfortable creation and editing of source codes by using any Windows editor such as the "Visual Studio". The file system of the Linux development system in the Windows network environment is accessible from path **"Workgroup"** under computer name **"Vm-xubuntu"** (also compare Table 9 in section 6.4).

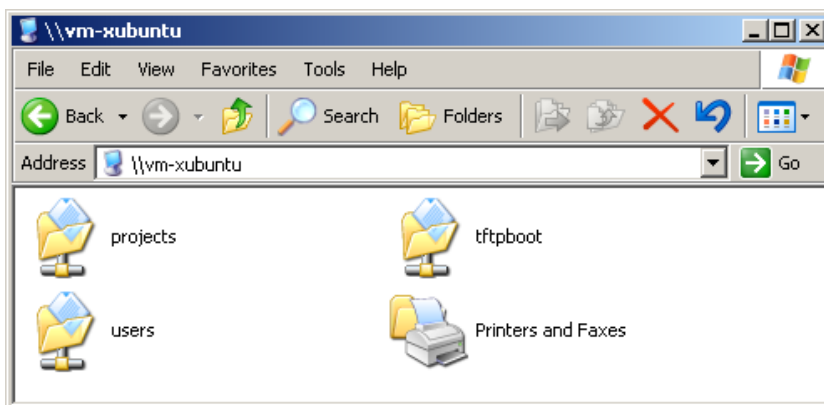


Figure 28: Linux development system in Windows network environment

After double-clicking the symbol *"users"* (see Figure 28), login is possible as user **"vmware"** with the corresponding **password "vmware"** (see Figure 29). In conclusion, path **"\\vm-xubuntu\users\"** is accessible.

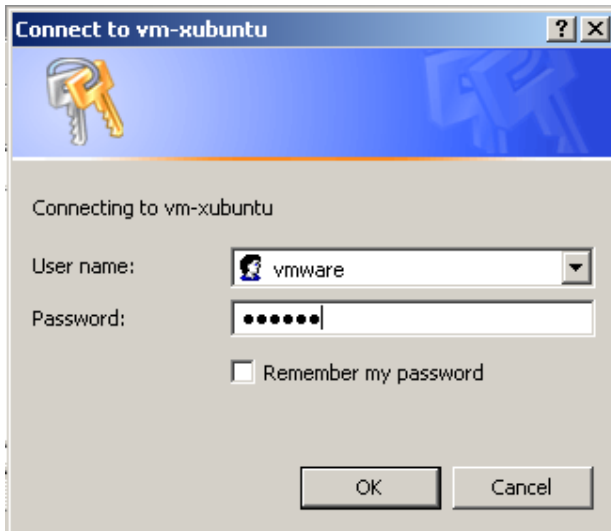


Figure 29: Login to "vm-xubuntu"

Alternatively, the VMware-Image of the Linux development system may be linked directly to a drive letter via Windows command "*net use*". This could be necessary for example if problems occur due to long timeouts during searching the Windows network environment or in general during locating the virtual computer. This may take place either through symbolic names or directly via the IP address of the Linux development system. In the latter case, the IP address of the Linux development system must first be determined as described in section 6.5. Command "*net use*" is as follows:

```
net use <local_device> <\\computername\sharename> /user:<username> [options]
```

To tie the VMware-Image with the Linux development system to the local drive letter "X:" for example, command "*net use*" is to be used as follows:

```
net use x: \\vm-xubuntu\users /user:vmware /persistent:NO
```

Alternatively, instead of the symbolic name, the IP address of the Linux development system may be directly entered, e.g.:

```
net use x: \\192.168.10.33\users /user:vmware /persistent:NO
```

### 6.6.2 Access via Telnet client

Access to a console of the Linux development system is also possible via a Telnet client in Windows because the VMware-Image has a Telnet server installed. This allows for calling command line tools such as "*make*" to translate user projects via Windows user interface.

The access with Telnet client directly takes place via the IP address of the Linux development system. Section 6.5 describes how the IP address of the Linux development system can be determined. To login to the Linux development system via the Telnet client included in Windows by default, call command "*telnet*" and enter the IP address. The procedure is analog to the login to the command shell of the ECUcore-5208 (compare Figure 11 in section 5.5.1), e.g.

```
telnet 192.168.10.33
```

Login as user "**vmware**" with the corresponding password "**vmware**" is possible in the Telnet window (also see Table 9 in section 6.4):

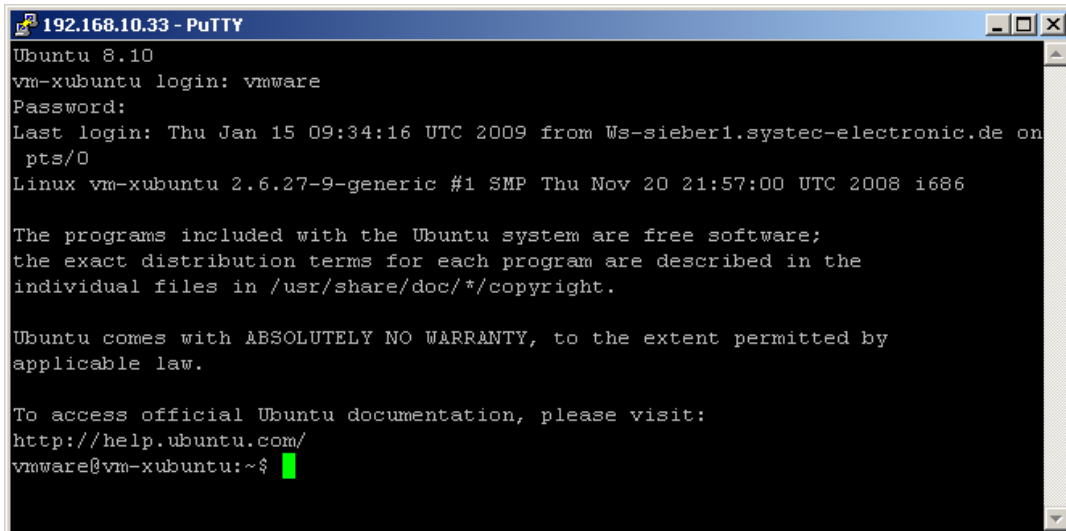


Figure 30: Access to the Linux development system via Telnet client

Figure 30 exemplifies login to the Linux developments by using the Telnet client that is included in Windows as standard.

## 6.7 Personal configuration and actualization of the Linux VMware-Image

### 6.7.1 Adjustment of keyboard layout and time zone

By default, the Linux VMware-Image is set to US keyboard layout and UTC time zone. Via the country symbol in the task menu, it is possible to easily switch to another preinstalled keyboard layout (see Figure 31).

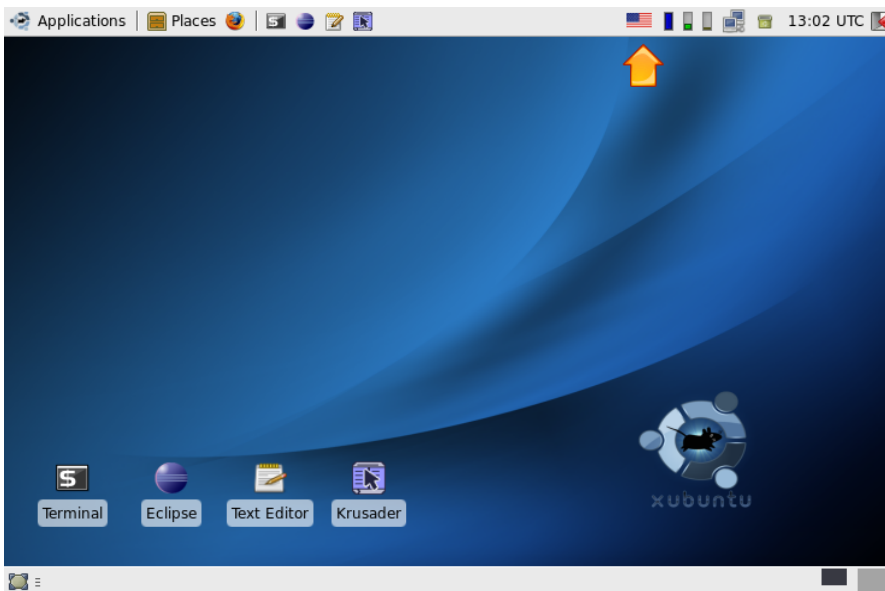


Figure 31: Country symbol for switching keyboard layouts

By clicking with the **right mouse button** on the country symbol in the task menu (see Figure 31) and by calling entry "Properties" from the popup menu, an alternative **keyboard layout** may be **permanently chosen**. Hence, dialog "Configure Keyboard Layout Switcher" opens and the desired layout can be set as "Default Layout" (see Figure 32).



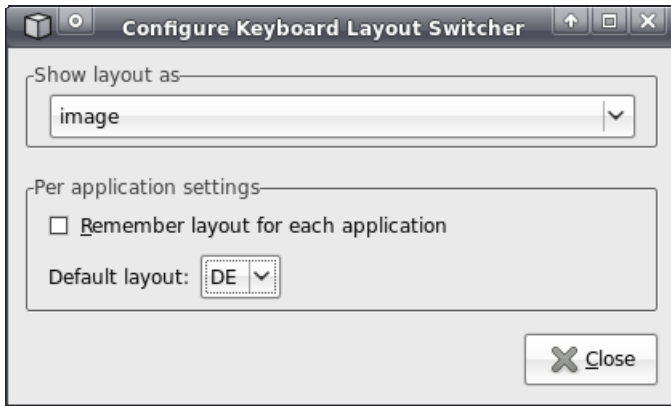


Figure 32: Choosing a permanent keyboard layout

**Adding more keyboard layouts** is possible by using "Xfce Settings Manager" which can be directly called from the start menu: "Applications -> Settings -> Settings Manager" (see Figure 33).

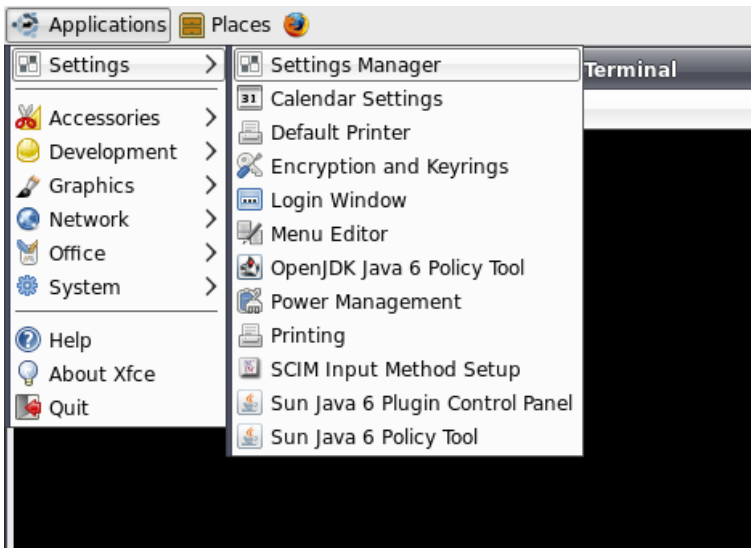


Figure 33: Calling "Xfce Settings Manager" from the start menu

More keyboard layout can be added or deleted within the "Xfce Settings Manager" by using option "Keyboard" and sub-option "Layouts" (see Figure 34).

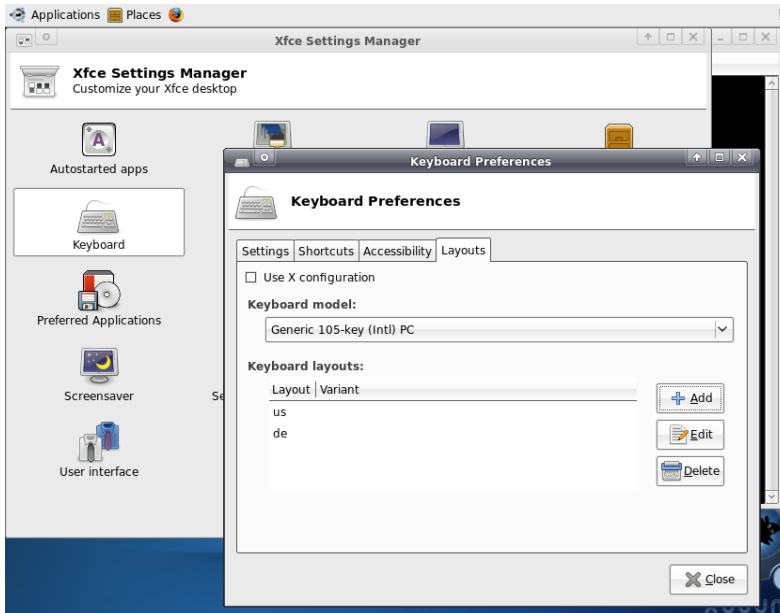


Figure 34: Adding keyboard layouts in the "Xfce Settings Manager"

**Setting the time zone** takes place via a control panel of the system configuration which is as well directly callable from the start menu: "*Applications -> System -> Time and Date*" (see Figure 35). Since changing the time zone is an administrative activity, the dialog must first be released by using pushbutton "*Unlock*" (also see Figure 35). Therefore, the administrator password will be asked analog to the console command "*sudo*". By default, **password "vmware"** must be entered (also compare Table 9 in section 6.4). Afterwards, the capital and the corresponding time zone can be chosen via path "*Time zone*".

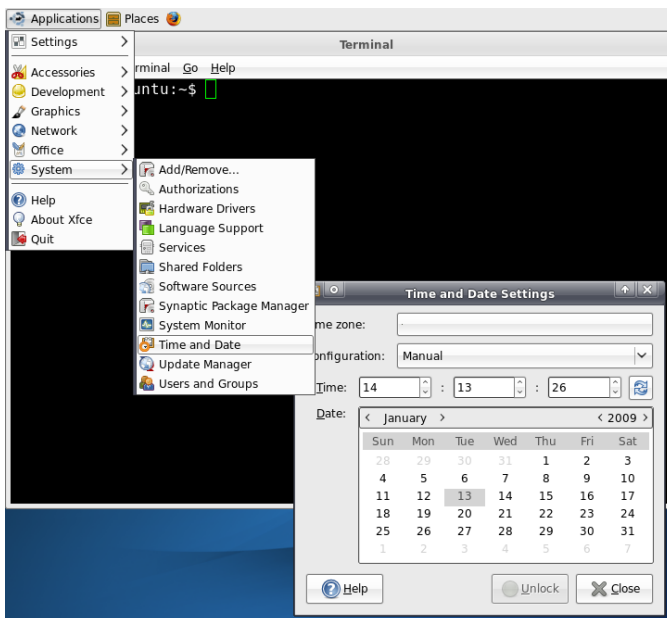


Figure 35: Adjusting the time zone

## 6.7.2 Adjusting the desktop size

The Linux VMware-Image is able to automatically adjust the desktop size to the window size of the VMware-Player. The VMware-Player runs like a normal Windows application. Hence, to change the desktop size for the Linux VMware-Image, use the mouse to adjust the window frame of the VMware-Player to the desired size. The maximum usable window size is defined in the configuration file "**SO-1096\Xubuntu-ECUcore5208.vmx**" of the VMware-Player and may be modified if required:

```
##### display #####
...
svga.maxWidth = "1024"
svga.maxHeight = "768"
...
```

### 6.7.3 Setting a static IP address for the Linux VMware-Image

By default, the dynamic configuration of the IP address via DHCP is activated in the Linux VMware-Image. Hence, in most network environments, the Linux VMware-Image can be used ad hoc without setting parameters manually beforehand. In networks that do not provide a DHCP server, the static IP address for the Linux VMware-Image must be defined by the user. Otherwise, an Ethernet-based communication with the ECUcore-5208 is not possible.

To define a static IP address for the Linux VMware-Image, the symbol of the "Network Manager" in the task menu (see Figure 36) must be clicked on with the **right mouse button** and entry "Edit Connections" must be called from the Popup menu. Afterwards, dialog "Network Connections" opens up (see Figure 37).

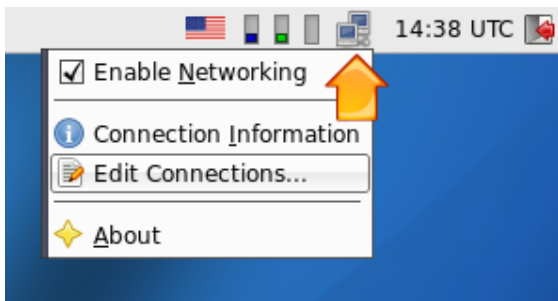


Figure 36: "Network Manager" for the configuration of the Ethernet interface

In Tab sheet "Wired" in the dialog "Network Connections" (see Figure 37), a new network environment can be created by using pushbutton "Add". Afterwards, dialog "Edit Wired Connection" opens up (see Figure 38).

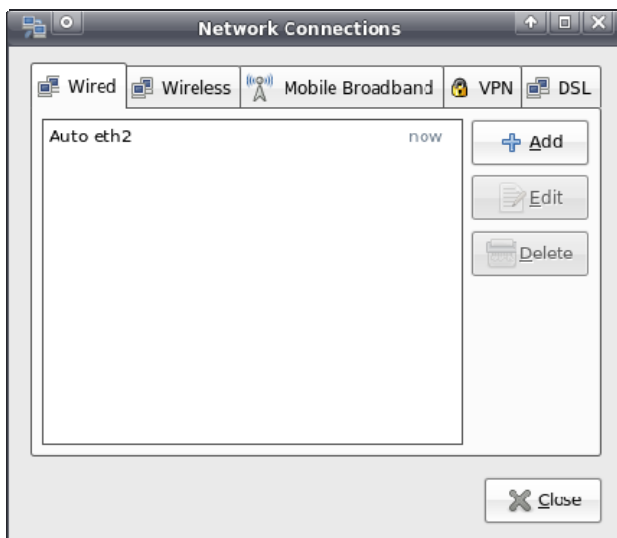


Figure 37: Adding a network connection

In dialog "Edit Wired Connection" (see Figure 38) in the choice box "Method", pre-adjustment "Automatic (DHCP)" is to be changed to the alternative option "Manual". Afterwards, settings for IP address, network mask, gateway DNS server etc. can be undertaken in Tab sheet "IPv4 Settings".

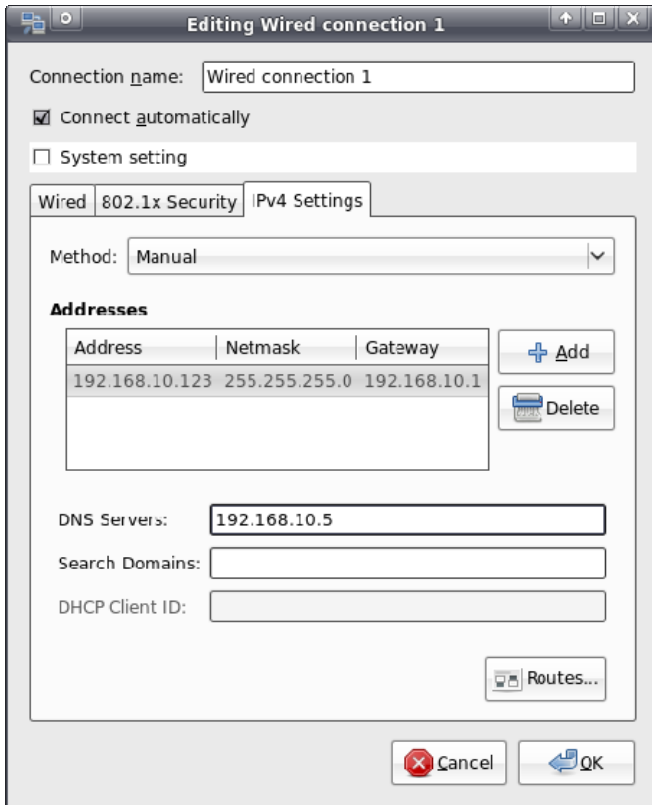


Figure 38: Configuration of the network connection

After the configuration is completed and all dialogs are closed, click again on symbol "Network Manager" in the task menu by using the **left mouse button** this time (see Figure 39). Choose the new network connection with the static IP address as active connection.

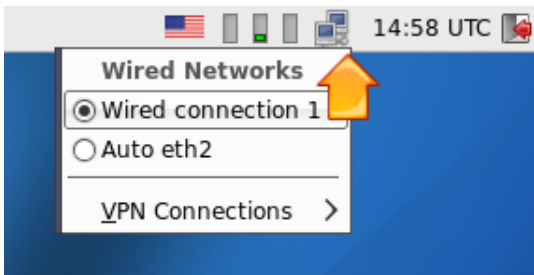


Figure 39: Changing the network configuration in the "Network Manager"

#### 6.7.4 System update of the Linux VMware-Image

The automatic update of packet lists is **deactivated** in the Linux VMware-Image. Hence, it is assured to retain a defined system revision. Although, the user will not receive information if updated packets are available that close security gaps.

Nevertheless, it is possible to re-activate the automatic update of packet lists or to initiate an update manually. This is possible via program "Synaptic Package Manager" (from the menu system) and the console program "aptitude". After updating packet lists, it is also possible to install new packet lists with those programs.

**The user is expressly advised: It can not be guaranteed that the Linux development system of the ECUcore-5208 maintains complete functionality after an update is accomplished. It is strongly recommended to make a backup copy of the Linux development system prior to the update.**

### 6.7.5 Changing the computer name in the Windows network environment

By default, in the Windows network environment the Linux VMware-Image uses the computer name "vm-xubuntu" (also see Table 9 in section 6.4). The access to a computer in the Windows network is controlled via its name. Hence, computer names must be adjusted clearly if the Linux VMware-Image runs in parallel on several computers. This prevents multiple use of the same name which could bring about collisions and access errors.

The computer name is defined via file `/etc/hostname`. Enter command **"sudo gedit /etc/hostname"** to modify this name. The modified name will be taken over after restart. Command `hostname` brings about prompt change of the name. Therefore, the new computer name must be entered as parameter, e.g. **"sudo hostname vm-xubuntu-2"**. Changing the name with this command only lasts temporarily until the next restart – in contrast to modifying file `/etc/hostname`.

### 6.7.6 Shrinking the VMware-Image

Call the VMware-Toolbox by entering command **"sudo vmware-toolbox"** to shrink the VMware-Image to the necessary minimal size. Using the tab sheet "Shrink" the VMware image get be shrunk to minimal size.

## 7 Software Development for the ECUcore-5208

### 7.1 Necessary steps after 1<sup>st</sup> start

After the first start of the VMware-Image the command “make” has to be called once inside the “/projects/ECUcore-5208/uClinux-dist” directory.

### 7.2 Software structure of the ECUcore-5208

All components necessary to develop software for the ECUcore-5208 are filed in path “/projects” in the VMware-Image of the Linux development system (or “\vmm-xubuntu\users\projects” in the Windows network environment). Figure 40 illustrates the directory structure.

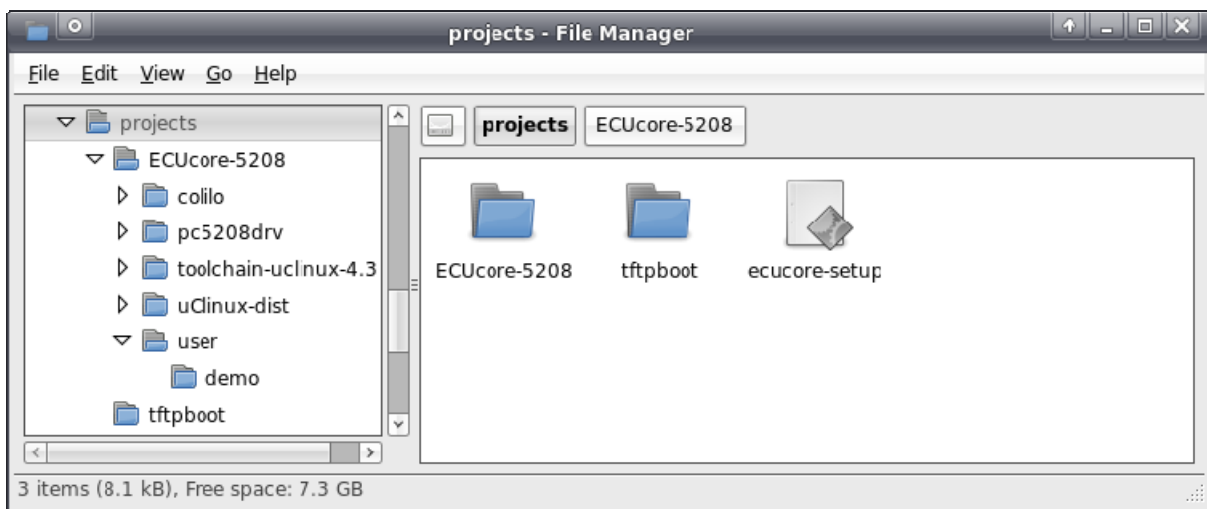


Figure 40: Structure of directory “/projects” in the Linux development system

/projects	
+ - ECUcore-5208	
+- colilo	Source code of the bootloader "CoLilo"
+- driver	<b>Driver Libraries for the ECUcore-5208</b>
+- pc5208drv	I/O Driver for the integration into own user projects (including source code and testing application)
+- candrv	CAN Driver for the integration into own user projects
+- toolchain-uclinux-4.2	uClinux Toolchain for Coldfire 5208
+- uClinux-dist	Project path for creating uClinux-Images for ECUcore-5208 (uClinux-Kernel and user programs)
+- linux-2.6.x	Source code of the uClinux kernel for the ECUcore-5208
+- user	Source code of the user program for the ECUcore-5208
...	other components of the uClinux distribution

	+ - user	Path to file user projects
	+ - demo	<b>Reference and I/O Demo project for the ECUcore-5208</b>
	+ - hellocan	<b>Reference and CAN Demo project for the ECUcore-5208</b>
+ -	tftpboot	<b>NFS directory for the integration by the ECUcore-5208</b>
+ -	ecucore-setup	Shell script to set environment variables

Path ***"/projects/ECUcore-5208/uClinux-dist"*** contains all uClinux sources for the ECUcore-5208.

Directory ***"/projects/ECUcore-5208/driver/pc5208drv"*** contains the source code of the I/O Driver of the ECUcore-5208 (also testing application) as well as **Header files and complete libraries of the I/O Driver** for the integration into own user projects (see section 7.4.1).

Directory ***"/projects/ECUcore-5208/driver/candrv"*** contains **Header files and complete libraries of the CAN Driver** for the integration into own user projects (see section 7.5.1).

Directory ***"/projects/ECUcore-5208/user/demo"*** provides a Demo program that describes access to in- and outputs of the ECUcore-5208 on the one hand (section 7.4.1 includes details) and serves as template for own projects on the other. In the following, all further descriptions about the software development refer to this Demo project (specifically in section 7.7).

Directory ***"/projects/ECUcore-5208/user/hellocan"*** includes a Demo program that describes access to the CAN interface of the ECUcore-5208 on the one hand and serves as template for own projects on the other.

Path ***"/projects/tftpboot"*** is the root directory for the integration of the Linux development system into the local file system of the ECUcore-5208 via NFS (see section 7.6.1).

The shell script ***"/projects/ecucore-setup"*** sets the required environment parameters that are necessary to execute the build system (see section 7.3). This shell script is automatically executed if a console ("Terminal") is opened via file *".bashrc"*. In the same way if the graphical IDE "Eclipse" is started via the appropriate desktop symbol.

### 7.3 Makefile and environment variables to create projects

Creating user programs for the ECUcore-5208 requires the usage of GNU-Crosscompiler Toolchain for Freescale MCF52xx processors. This is completely installed and configured in the VMware-Image of the Linux development system. Environment variables defined in the Shell script ***"/projects/ecucore-setup"*** are relevant in this matter:

```
export M68K_LINUX_BSP_PATH=/projects/ECUcore-5208
export M68K_PATH=$M68K_LINUX_BSP_PATH/toolchain-4.2/bin
```

```
#add toolchains to PATH
export PATH="$M68K_PATH:$PATH"
```

The template in path ***"/projects/ECUcore-5208/user/demo"*** should be the initial point to develop own programs (or *"\\vm-xubuntu\projects\ECUcore-5208\user\demo"* in the Windows network environment). Hereby, variables defined in Makefile (*demo/source/Makefile*) and references to GNU-Crosscompiler Toolchain for Freescale MCF52xx processors (based on definitions in ***"/projects/ecucore-setup"***) are especially relevant.

```

CDEFS          = $(M68K_DBG_MODE)
CFLAGS         += -O0 -g -Wall -mcpu=5208 $(INCLUDES) $(CDEFS)
LDFLAGS        += -mcpu=5208 -Wl,-elf2flt

CROSS          = m68k-uclinux-
CC             = $(CROSS)gcc
AR             = $(CROSS)ar
    
```

Calling Tools via an appropriate Macro such as "\$\$(CC)" takes place within the Makefile:

```
$(CC) $(CFLAGS) -o $(EXEC) hello.c
```

Prepared in the Makefile also is the copying of generated executables into directory *"/tftpboot"* or one of its subdirectories. Through this, the executable program can later on be started directly on the ECUcore-5208 without other intermediate steps (also compare section 7.6.1).

## 7.4 I/O Driver for the ECUcore-5208

### 7.4.1 Integration of the I/O Driver into own user projects

Directory *"/projects/ECUcore-5208/driver/pc5208drv"* of the Linux development system contains the source code of the I/O Driver for the ECUcore-5208 (including testing application). Moreover, it contains header files and already built libraries of the driver for the integration into own user projects.

Figure 41 illustrates the structure of the I/O Driver for the ECUcore-5208. The driver is divided into a kernel space module (*pc5208drv.ko*) which is in charge of accesses to the hardware and a static user space library (*pc5208drv.a*). Both components, the kernel space module and the user space library are necessary to accomplish I/O accesses.

The creation and usage of dynamic libraries are **not** supported yet.

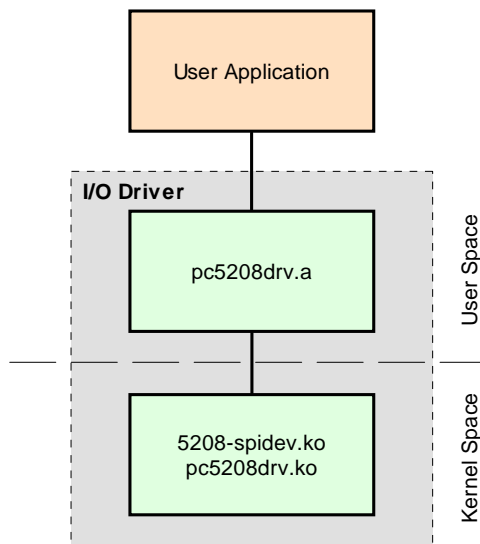


Figure 41: Structure of the I/O Driver for the ECUcore-5208

The following files from the project directory of the I/O Driver are relevant for the integration into own user projects:

```

pc5208drv.h:    /projects/ECUcore-5208/driver/pc5208drv/include/pc5208drv.h
                header file to describe the driver interface
                This header file has to be integrated into the Source code (*.c) of the user project
                using #include.
    
```



*pc5208drv.a*: */projects/ECUcore-5208/driver/pc5208drv/lib/pc5208drv.a*  
User space library of the I/O Driver to be statically linked to the user application (similar to static standard libraries of the C development environment)  
If statically linked, the User space library is intrinsically linked to the user application and cannot be separated.

*pc5208drv.ko*: */projects/ECUcore-5208/driver/pc5208drv/lib/pc5208drv.ko*  
*5208-spidev.ko*: */projects/ECUcore-5208/driver/pc5208drv/lib/5208-spidev.ko*  
Kernelspace modules of the I/O Driver for accesses to hardware (PLD and Port pins).  
This modules must be loaded using Linux command "*insmod*" – prior to starting the user application:

```
insmod 5208-spidev.ko
insmod pc5208drv.ko
```

In the exemplary demo program in section 7.7, the user space library of the I/O Driver is strongly attached to the user application. Therefore, upon calling GCC, the static library *pc5208drv.a* must be added to the list of objects to be linked. Variable "*LIBS=*" is set in the Makefile of the demo project and is transferred to the linker (via GCC call):

```
LIBS = ../lib/pc5208drv.a

@$(CC) $(LDLFLAGS) -o $@ $(OBSJ) $(LIBS) $(LDLIBS)
```

Functions provided by the I/O driver are listed in the header file "*pc5208drv.h*", their usage is kept record of in the demo program "*/projects/ECUcore-5208/user/demo*" (also see section 7.7).

The delivery status of the ECUcore-5208 includes completely generated and ready-to-load (via "*insmod*") Kernel modules of the I/O driver filed in directory "*/home/bin*" (see section 5.13):

```
insmod /home/bin/5208-spidev.ko
insmod /home/bin/pc5208drv.ko
```

Loading the kernel module *5208-spidev.ko* is necessary so the I/O driver *pc5208drv.ko* can access SPI devices using the separate bus. This separation is necessary to use a faster CAN driver (see section 7.5.1)

The Demo project in "*/projects/ECUcore-5208/user/demo*" shows an example for the application of the I/O Driver on the ECUcore-5208 (see sections 7.4.2 and 7.7).

### 7.4.2 I/O Driver demo project

The demo project for the I/O driver is filed in directory "*/projects/ECUcore-5208/user/demo*" of the Linux development system. It illustrates the access to in- and outputs of the module. Therefore, the demo project uses the assistance of the I/O driver filed in "*/projects/ECUcore-5208/driver/pc5208drv*". Section 7.7 in much detail describes the Demo project as reference project for software development for the ECUcore-5208.

Prior to starting the Demo project, command "*insmod*" is used to explicitly load the I/O driver. Afterwards, the demo program can be executed:

```
insmod 5208-spidev.ko
insmod pc5208drv.ko
./demo
```

Figure 47 in section 7.7.1. exemplifies the execution of the Demo project on the ECUcore-5208. The Demo project can be closed either by setting the Run/Stop Switch to position "MRes" or by pressing "Ctrl+C".

## 7.5 CAN Driver for the ECUcore-5208

### 7.5.1 Integration of CAN Driver into own user projects

For the integration into own user projects, directory `"/projects/ECUcore-5208/driver/candrv"` of the Linux development system contains header files and completely generated libraries of the CAN driver for the ECUcore-5208.

The structure of the CAN driver is identical to the I/O Driver structure as specified in section 7.4.1. Hence, the CAN driver also is divided into a kernel space module (`candrv.ko`) that is in charge for accesses to the CAN controller and a user space library (`candrv.a`). Both components, kernel space module and user space library are necessary to access the CAN bus.

From the CAN Driver project directory, the following files are relevant for the integration into own user projects:

`cdrvinc.h`: `/projects/ECUcore-5208/driver/candrv/include/cdrvinc.h`  
header file to describe driver interfaces  
This header file has to be integrated into the source code (\*.c) of the user project using `#include`. It then includes all further header files in directory `"include"` according to the correct sequence.

`candrv.a`: `/projects/ECUcore-5208/driver/candrv/lib/candrv.a`  
User space library of the I/O driver to be statically linked to user applications

`candrv.ko`: `/projects/ECUcore-5208/driver/candrv/lib/candrv.ko`  
kernel space module of the CAN driver to access the CAN controller  
This module must be loaded using Linux command `"insmod"` prior to starting the user application:

```
insmod candrv.ko
```

The CAN Driver Software Manual (Manual no.: L-1023) includes descriptions about the interface and the usage of the CAN driver in own applications. Demo project `"/projects/ECUcore-5208/user/helloCAN"` is an example for the usage of the CAN driver on the ECUcore-5208 (see section 7.5.2).

The delivery status of the ECUcore-5208 provides a completely generated and ready-to-load Kernel module of the CAN Driver filed in directory `"/home/bin"` (see section 5.13):

```
insmod /home/bin/candrv.ko
```

### 7.5.2 CAN driver demo project

The demo project for the CAN driver is filed in directory `"/projects/ECUcore-5208/user/helloCAN"` of the Linux development system. It illustrates the access to the CAN interfaces of the module. Therefore, the Demo program needs the CAN Driver that is filed in `"/projects/ECUcore-5208/driver/candrv"`.

To demonstrate information exchange via the CAN-Bus, an appropriate receiver is needed. Therefore, the CAN analysis tool `"CAN-REport"` in combination with an USB-CANmodul is well suitable. By using `"CAN-REport"`, any CAN messages can be sent and received (see Figure 43).

demo program `"helloCAN"` uses **125kBit/s** as Bitrate. After being started, it initially generates 10 channels to receive CAN messages and 10 channels to send them:

Reception area: CAN messages within the Identifier area 0x100 - 0x109  
Send area: CAN messages within the Identifier area 0x200 - 0x209

After successfully completing initialization, the Demo program "hellocan" only once sends a CAN message with Identifier 0x400 and data "68 61 6C 6C 6f 63 61 6E" (ASCII: "hellocan") to the bus. Afterwards, it passes on to its main loop in which it waits for receiving CAN messages in the specified identifier area 0x100 - 0x109. When a CAN message is received, it is sent back as echo increased by 0x100 identifier (equals identifier area 0x200 - 0x209). Figure 42 shows the execution of demo project "hellocan" on the ECUcore-5208. The demo project can be closed by pressing "Ctrl+C".

```

Telnet 192.168.10.248
ECUcore-5484 login: PlcAdmin
Password:
sh-2.05:~# ./mountnfs.sh 192.168.10.58
Check reachability of nfs server '192.168.10.58'... server is online
mount /mnt/nfs... done.

sh-2.05:~# cd /mnt/nfs/hellocan
sh-2.05:/mnt/nfs/hellocan# ls
candrv.ko  hellocan
sh-2.05:/mnt/nfs/hellocan# insmod candrv.ko
sh-2.05:/mnt/nfs/hellocan# ./hellocan

*****
  CAN demo application for SYSTEC ECUcore-5484
  (c) 2009 SYSTEC electronic GmbH
*****

Runtime configuration:
Supported instances: 2
DevNumber:         0
BtrRate:           125kBaud
Initialize CAN Device Number 0 ... ok
Register Rx CAN-ID pool:
  Register Rx CAN-ID = 0x100 ... ok
  Register Rx CAN-ID = 0x101 ... ok
  Register Rx CAN-ID = 0x102 ... ok
  Register Rx CAN-ID = 0x103 ... ok
  Register Rx CAN-ID = 0x104 ... ok
  Register Rx CAN-ID = 0x105 ... ok
  Register Rx CAN-ID = 0x106 ... ok
  Register Rx CAN-ID = 0x107 ... ok
  Register Rx CAN-ID = 0x108 ... ok
  Register Rx CAN-ID = 0x109 ... ok
Register Tx CAN-ID pool:
  Register Tx CAN-ID = 0x200 ... ok
  Register Tx CAN-ID = 0x201 ... ok
  Register Tx CAN-ID = 0x202 ... ok
  Register Tx CAN-ID = 0x203 ... ok
  Register Tx CAN-ID = 0x204 ... ok
  Register Tx CAN-ID = 0x205 ... ok
  Register Tx CAN-ID = 0x206 ... ok
  Register Tx CAN-ID = 0x207 ... ok
  Register Tx CAN-ID = 0x208 ... ok
  Register Tx CAN-ID = 0x209 ... ok
Register Tx CAN-ID for "hello" message:
  Register Tx CAN-ID = 0x400 ... ok
Send "hello" message ... ok

Enter Main Loop ...

Message #1 received:
CANID=0x100, Size=8 : 00 01 02 03 04 05 06 07
Echo Message:
CANID=0x200, Size=8 : 00 01 02 03 04 05 06 07
Send Echo Message ... ok

Message #2 received:
CANID=0x106, Size=4 : 11 22 33 44 -- -- -- --
Echo Message:
CANID=0x206, Size=4 : 11 22 33 44 -- -- -- --
Send Echo Message ... ok

```

Figure 42: Execution of Demo project "hellocan" on the ECUcore-5208

Figure 43 shows data exchange with the demo program "hellocan" in the CAN bus analysis tool "CAN-REport".

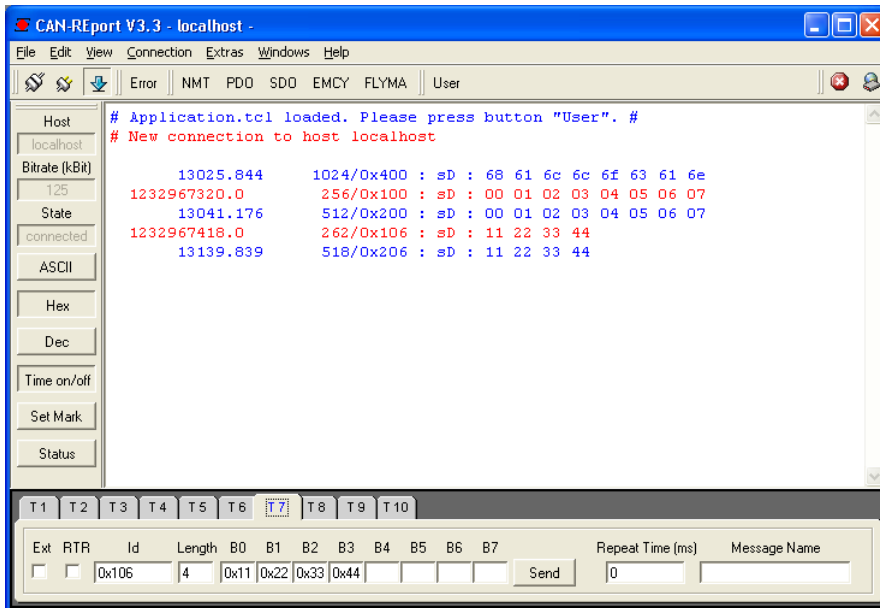


Figure 43: CAN analysis tool "CAN-REport"

## 7.6 Transferring programs to the ECUcore-5208

The configuration of the module as described in section 5.4 is one requirement for transferring – as well as starting – programs on the ECUcore-5208. Afterwards, login to the command shell of the ECUcore-5208 according to section 5.5.1 is necessary.

There are two possibilities for the transmission of programs on the ECUcore-5208 or data exchange between development system (VMware Linux-Image) and the ECUcore-5208 in general. Both imply advantages and disadvantages:

**NFS:** "Network File System" (NFS) represents the easiest way of directly starting a user program (built in the Linux-Image) on the ECUcore-5208. To do so, a directory from the VMware-Image of the Linux development system is mounted into the local file system of the ECUcore-5208. Enter the appropriate command on the ECUcore-5208 to start the program. Required data transfer from the development system to the ECUcore-5208 implicitly takes place via NFS – no further commands from the user are necessary. Consequently, it is assured that the ECUcore-5208 is always running a current program version rather than one that is out-of-date. Hence, NFS is especially suitable during software development. Files can also be copied over the Linux machine using NFS. On the one hand, it is disadvantageous about NFS that it only allows for connections to other Linux machines, but not to a Windows computer for example. On the other hand, NFS only enables rudimentary user administration and access control. Later on this is most likely not desired if devices are used in practice. Section 7.6.1 provides details about the application of NFS.

**FTP:** The "File Transfer Protocol" is a standard and platform-independent protocol that is well-established in practice. Both, FTP server and clients are available for several operating systems such as Linux and Windows. On the contrary to NFS, by using FTP it is possible to transfer files from a Windows computer to the ECUcore-5208 (e.g. program update through service technicians by using Windows laptop). Moreover, FTP allows for detailed access control through authentication via username and password. Disadvantageous about FTP is the fact both username and password are transferred unencrypted. Also the command entry that is required for each data transmission may be considered bothersome or may be even forgotten especially during development phase. It then may occur that an old program version is run on the ECUcore-5208 without noticing it. Section 7.6.2 describes the usage of FTP.

## 7.6.1 Using NFS

The easiest way of starting a user program on the ECUcore-5208 that was first translated within the Linux-Image is a direct integration ("mounting") of a directory from the VMware-Image of the Linux development system into the local file system of the ECUcore-5208. Therefore, directory *"tftpboot"* including all sub-directories are exported by the VMware-Image of the Linux development system. The following steps are necessary to mount this file directory tree of the development system into the local file system of the ECUcore-5208:

### **1. Determining the IP address of the Linux development system**

Section 6.5 describes the procedure to determine the IP address of the Linux-Image.

### **2. Mounting the Linux development system onto the ECUcore-5208**

To mount directory *"tftpboot"* of the Linux-Image into the local file system of the ECUcore-5208, command *"mount"* must be used as follows:

```
mount -t nfs -o nolock <ip_vmware_image>:/tftpboot /mnt/nfs
```

For example, to attach the ECUcore-5208 to the Linux development system via the IP address defined in section 6.5, the following command must be entered on the ECUcore-5208:

```
mount -tnfs -o nolock 192.168.10.33:/tftpboot /mnt/nfs
```

The script *"mountnfs.sh"* that is located in the home directory of the ECUcore-5208 is able to simplify the usage of the mount command. The user is automatically located in that home directory after login to the command prompt. The IP address of the Linux-Image must be given to that script as parameter. Hence, for the example above this would lead to the following call:

```
./mountnfs.sh 192.168.10.33
```

After the mount command is executed, the entire content of directory *"tftpboot"* of the Linux-Image (including potential sub-directories) is available in the local directory *"/mnt/nfs"* of the ECUcore-5208.

Figure 44 summarizes all necessary steps to mount the Linux-Image into the local file system of the ECUcore-5208.



When looking at program "demo" for example that was translated in section 7.3 and copied into directory "/tftpboot/demo", the following command is required to transfer the program via FTP from that directory to the local directory "/tmp" of the ECUcore-5208:

```
ftpget -u vmware -p vmware 192.168.10.33 /tmp/demo /tftpboot/demo/demo
```

### FTP Upload

Uploading files of the ECUcore-5208 into the Linux-Image happens via command "ftpput". Parameters "-u" for username and "-p" for password are needed to authenticate at the host system. Command "ftpput" is called as follows:

```
ftpput -u <username> -p <password> <ip_vmware_image> <remote_dir> <local_file>
```

The following command is required for example to transfer the log file "/var/log/messages" via FTP from the ECUcore-5208 into directory "/tftpboot" of the host system:

```
ftpput -u vmware -p vmware 192.168.10.33 /tftpboot/messages /var/log/messages
```

Figure 45 exemplifies the usage of commands "ftpget" and "ftpput" to download and upload files via FTP.

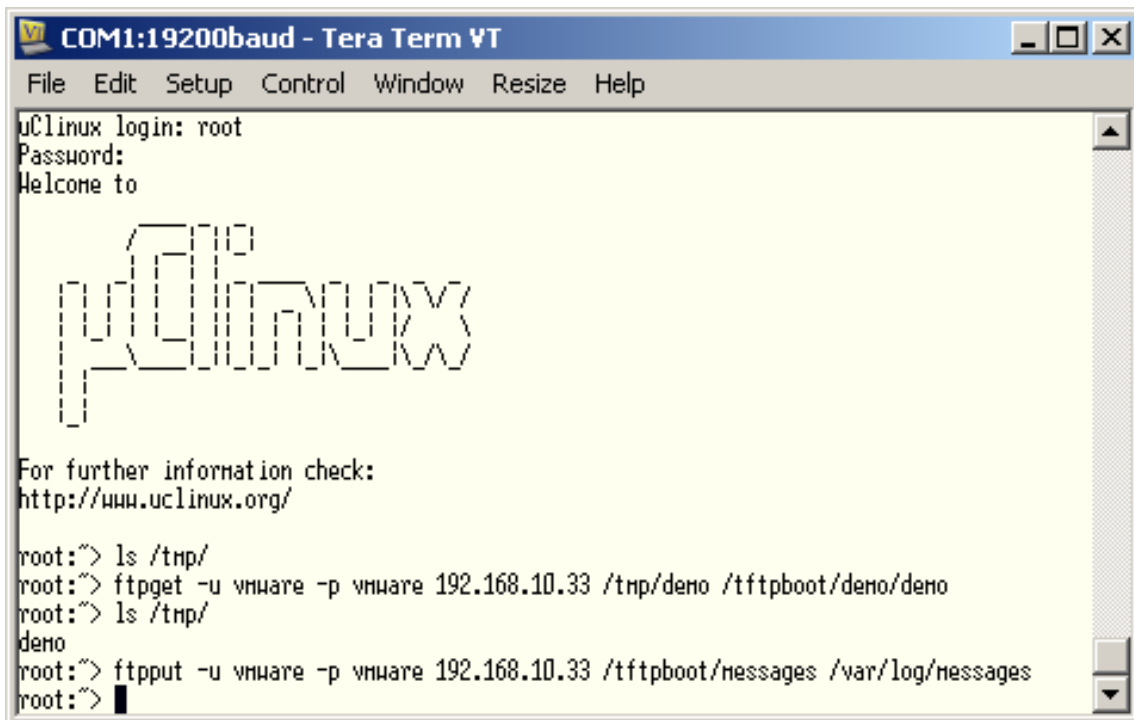


Figure 45: Download and upload via FTP

### 7.6.2.2 ECUcore-5208 as FTP server

The ECUcore-5208 provides a FTP server (using the Internet Service Daemon) that makes possible data exchange with any FTP client (e.g. up- and download of files to or from a computer). For security or performance reasons this FTP server is deactivated by default and must be started manually if needed.

The advantage of using the FTP server on the ECUcore-5208 is that client-sided there are comfortable graphical programs which enable simplified data exchange without knowing Linux commands. For example, if the device is applied in practice later on, this would enable a service technician to transfer a firmware update to the ECUcore-5208 or to select log files from this module via a graphic FTP client on his laptop.

Section 5.5.4 explains the activation of the FTP server on the ECUcore-5208 and the login to the module via a FTP client. Section 5.1 specifies suitable FTP client programs.

## 7.7 Compiling and executing the demo project "demo"

### 7.7.1 Usage of "make"

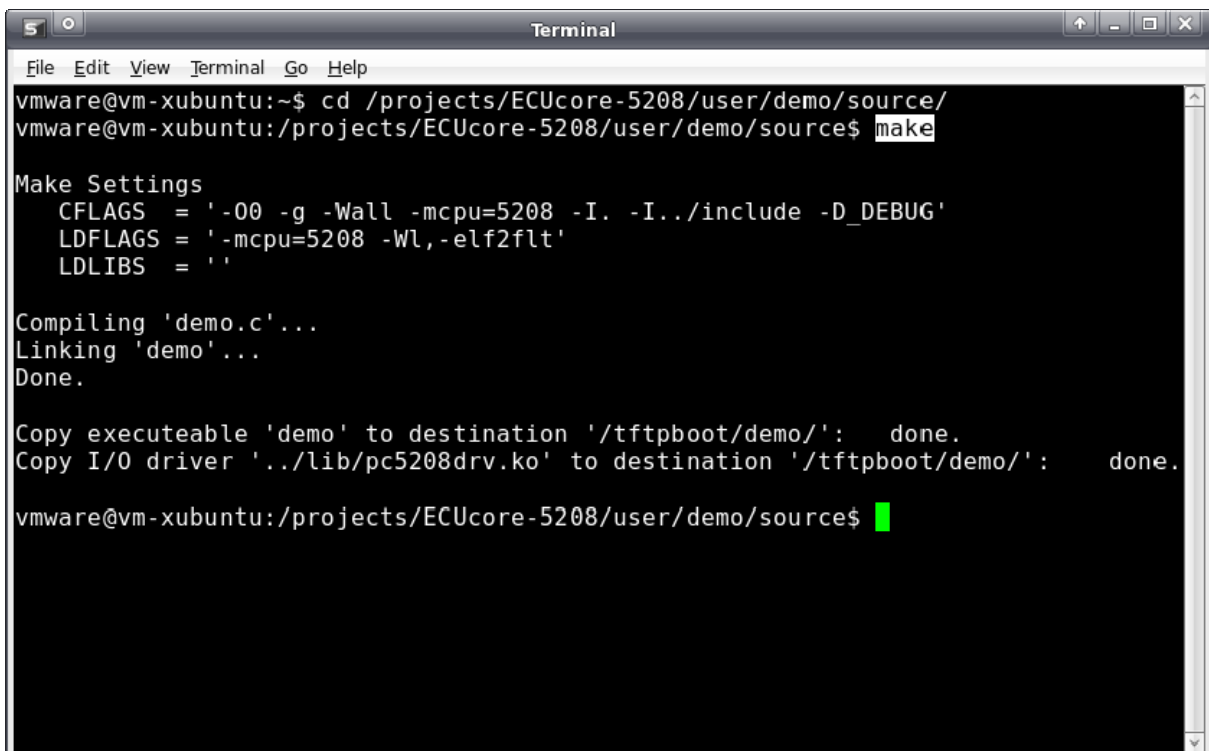
Directory `"/projects/ECUcore-5208/user/demo"` in the VMware-Image of the Linux development system contains a demo program that illustrates accesses to in- and outputs. The I/O driver stored in `"/projects/ECUcore-5208/driver/pc5208drv"` is used by the demo program to access the I/O. It is recommended that the demo project or at least its Makefile should be referred to as template for own projects.

The source code can be created or edited via Windows network environment in any Windows Editor. On the contrary, compiling the project is only possible within a Linux development environment. Therefore, it is possible to either use a console window in the Linux-Image (also called "Terminal") or access must be enabled via a Telnet client "from outside" by following all necessary steps analog. Hence, in the following there will be no difference between the console window in the Linux-Image and the Telnet access "from outside".

Switch to the appropriate project directory that contains the Makefile by using command `"cd"` in the console window. To translate the demo project, use directory `"/projects/ECUcore-5208/user/demo/source"`:

```
cd /projects/ECUcore-5208/user/demo/source
```

Afterwards, command `"make"` must be started. Figure 46 shows the usage of commands `"cd"` and `"make"` for the exemplary demo program contained in the VMware-Image.



```
vmware@vm-xubuntu:~$ cd /projects/ECUcore-5208/user/demo/source/
vmware@vm-xubuntu:/projects/ECUcore-5208/user/demo/source$ make

Make Settings
  CFLAGS = '-O0 -g -Wall -mcpu=5208 -I. -I../include -D_DEBUG'
  LDFLAGS = '-mcpu=5208 -Wl,-elf2flt'
  LDLIBS = ''

Compiling 'demo.c'...
Linking 'demo'...
Done.

Copy executeable 'demo' to destination '/tftpboot/demo/': done.
Copy I/O driver '../lib/pc5208drv.ko' to destination '/tftpboot/demo/': done.

vmware@vm-xubuntu:/projects/ECUcore-5208/user/demo/source$
```

Figure 46: Compiling the demo project in the Linux-Image

During executing the Makefile and after completing the build process successfully, the demo program itself (file `"demo"`) and the I/O driver needed for the execution (files `"5208-spidev.ko"` and



"*pc5208drv.ko*") are copied to directory "*/tftpboot/demo*" (compare screenshot in Figure 46). All required steps are completed after the translation and copying is finished.

All further steps exclusively take place on the ECUcore-5208. Therefore, login to the command shell of the module is necessary (see section 5.5.1). Afterwards, the following steps must be executed in the Terminal program or Telnet client:

1. Integration of directory "*/projects/tftpboot*" via NFS from the Linux development system to the local file system of the ECUcore-5208 (see section 7.6.1):

```
mount -t nfs -o noexec 192.168.10.33:/tftpboot /mnt/nfs
```

2. Switching to NFS directory in the local file system by using command "*cd*":

```
cd /mnt/nfs/demo
```

Directory "*/mnt/nfs*" on the ECUcore-5208 is identical with directory "*/tftpboot*" of the Linux development system in the VMware-Image. Accordingly, all files copied by the Makefile to directory "*/tftpboot/demo*" in the Linux development system are accessible by the ECUcore-5208 in directory "*/mnt/nfs/demo*" of its file system. It is not necessary to explicitly download executable binary files to the ECUcore-5208.

3. Starting the demo program on the ECUcore-5208.

Since the demo program accesses in- and outputs of the ECUcore-5208, it requires I/O driver "*pc5208drv*" to do so. Consequently, the I/O driver must be loaded using command "*insmod*". Afterwards, the demo program can be started:

```
insmod 5208-spidev.ko
insmod pc5208drv.ko
./demo
```

```

192.168.10.171 - PuTTY
root:~> mount -t nfs -o nolock 192.168.10.33:/tftpboot /mnt/nfs/
root:~> cd /mnt/nfs/demo/
root:/mnt/nfs/demo> ls
demo          pc5208drv.ko
root:/mnt/nfs/demo> insmod pc5208drv.ko
root:/mnt/nfs/demo> ./demo

*****
  I/O demo application for SYSTEC ECUcore-5208
  (c) 2009 SYSTEC electronic GmbH
-----
  Version: 1.00
  Build:   Jan 29 2009, 13:41:21
*****

I/O Driver version:  KernelModule=3.00, UserLib=3.00

Hardware:  CPU Board: 4218.00 (#00H)
           CPU PLD:  0.00  (#00H)
           IO Board: 4220.00 (#00H)
Driver:    Config:    0000H

Start basic I/O main loop...

DI=00-00-00   DO=00-00-01   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-00-02   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-00-04   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-00-08   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-00-10   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-00-20   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-00-40   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-00-80   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-01-00   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-02-00   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
DI=00-00-00   DO=00-04-00   bHexSw=0x00   bDipSw=0x00   R/S/M-Sw=RUN
^C
root:/mnt/nfs/demo>

```

Figure 47: Executing the demo project "demo" on the ECUcore-5208

Figure 47 demonstrates the execution of the demo project on the ECUcore-5208. To finish the demo project either set Run/Stop Switch to position "MRes" or simply press "Ctrl+C".

## 7.7.2 Using graphical IDE "Eclipse"

"Eclipse" as graphical IDE is installed in the VMware-Image of the Linux development system. This allows for accomplishing work steps in the software development process such as editing, compiling and debugging user programs within a comfortable development environment. The following sections describe the application of IDE "Eclipse" using the exemplary demo project that is included in the VMware-Image and filed in directory *"/projects/ECUcore-5208/user/demo"* (also compare descriptions in section 7.7.1)

### 7.7.2.1 How to open and edit the demo project

The graphical IDE "Eclipse" is started by clicking on the appropriate desktop symbol. Dialog "Workspace Launcher" appears as shown in Figure 48. Enter the path for the workspace directory in

the project tree in area "Workspace". For the demo project of the Linux development system this would be *"/projects/ECUcore-5208/user/demo/workspace"*.

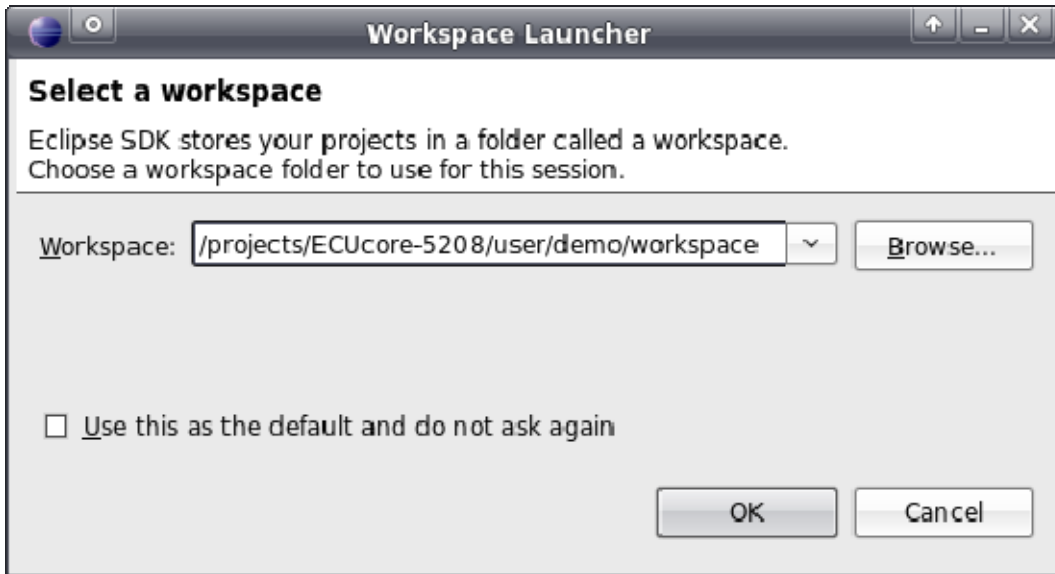


Figure 48: Eclipse dialog "Workspace Launcher"

After activating pushbutton "OK" the graphical surface of the IDE starts automatically and loads the workspace entered. Figure 49 provides an overview of "Eclipse" after it started.

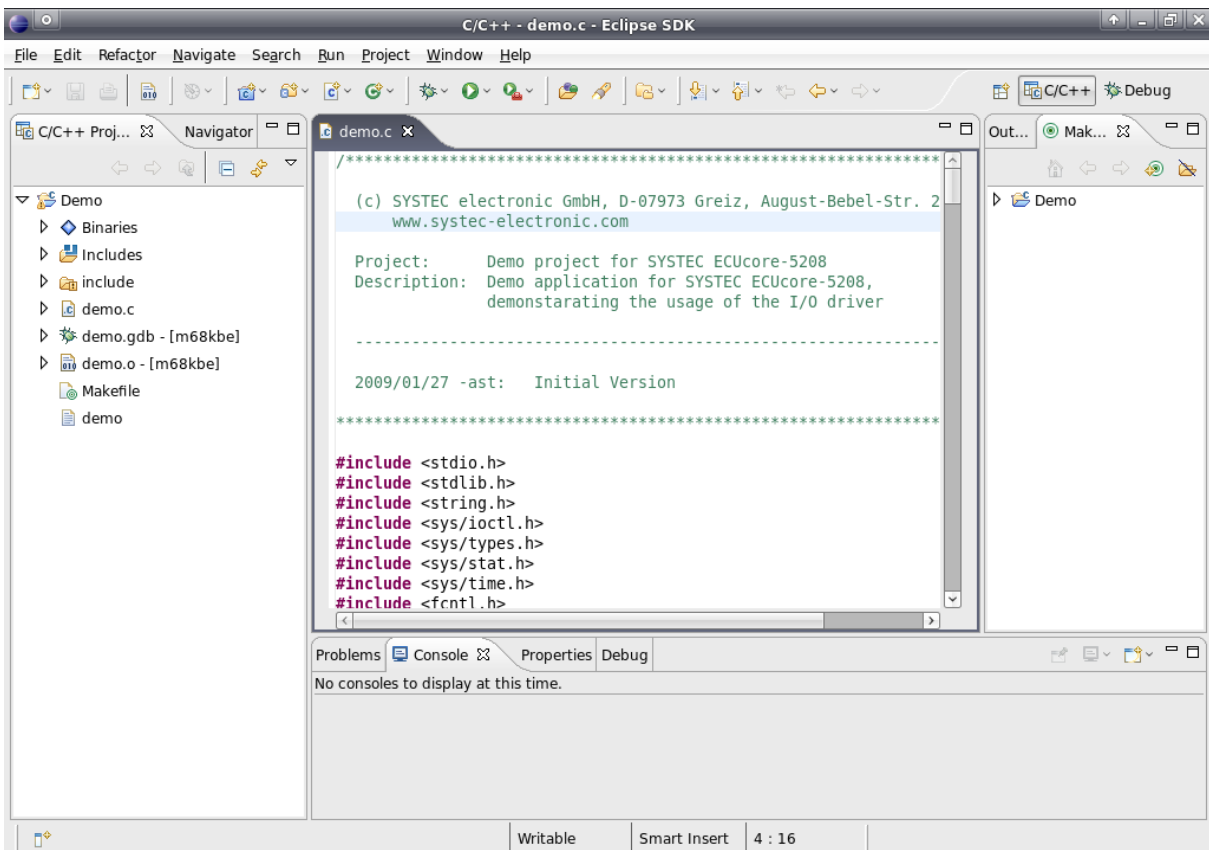


Figure 49: The graphical IDE "Eclipse"

The source code of the demo program can be edited in the editor window. By double-clicking an entry in the project tree (left), all files of the project can be opened and edited.

### 7.7.2.2 Compiling the demo project

To compile the demo project open entry "Demo" in window "Make Targets" by clicking on the triangle that is put in front. Afterwards, call the translating process by double-clicking on entry "Build Project" (see Figure 50). Hence, Eclipse calls the Makefile of the demo project in directory "source" ("*/projects/ECUcore-5208/user/demo/source/Makefile*"). This is the same Makefile that was called manually in a Terminal window as described section 7.7.1. Accordingly, all messages shown in IDE window "Console" are identical to the ones shown in Figure 46. In the same way, the demo program (file "demo") and the I/O drivers needed to execute it (files "5208-spidev.ko" and "pc5208drv.ko") are copied into directory "*/tftpboot/demo*". Consequently, the demo program can be started on the ECUcore-5208 after successful completion of the Build process as explained in section 7.7.1.

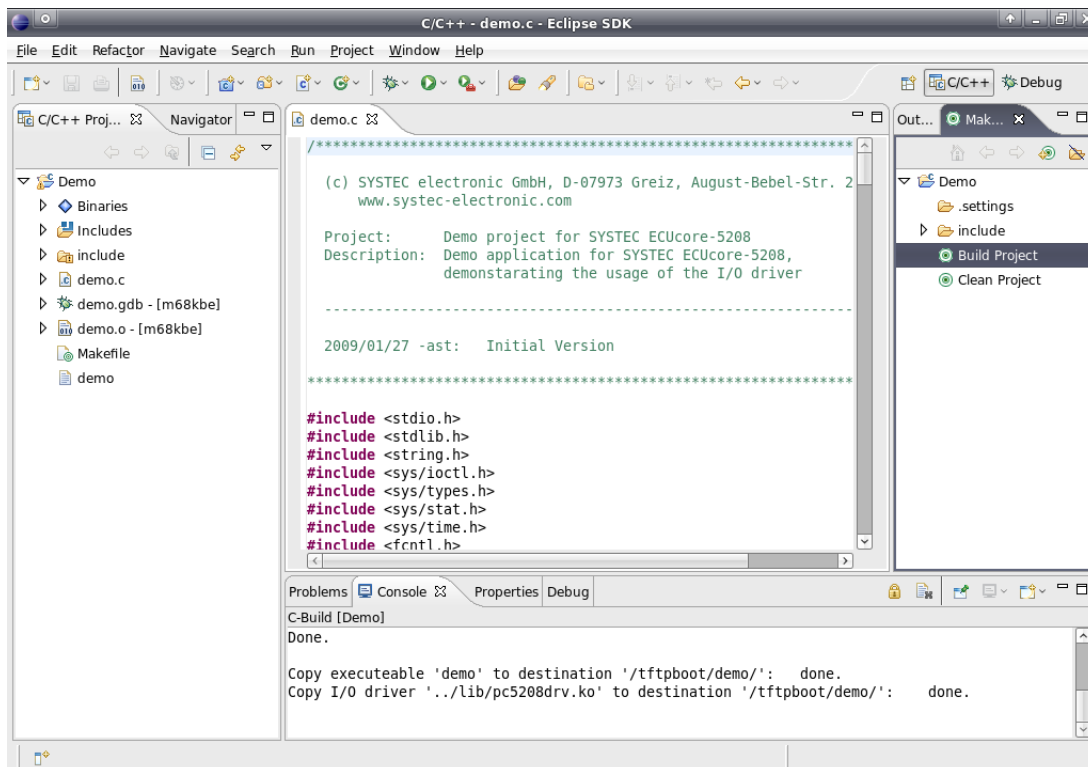


Figure 50: Translating the demo project in Eclipse

If problems or warnings occur during translation (e.g. as a result of modifying the demo project by the user), they are clearly illustrated in IDE window "Problems". By double-clicking an entry the IDE opens the corresponding source file and marks the row in the editor.

By double-clicking entry "Clean Project" (see Figure 50) all generated files are deleted. Targets can be edited if entries "Demo Project" or "Clean Project" are clicked on with the right mouse button (e.g. modifying the corresponding name).

### 7.7.2.3 Debugging the demo project in the IDE

One of the most considerable advantages for using Eclipse is the possibility to debug the compiled program directly on high-level language within the IDE. This includes for example single step execution of programs on C-level, setting breakpoints directly in the source code and observing variables within the IDE. While on the computer Eclipse-IDE runs with the Linux development environment, the program to be debugged is directly executed on the ECUcore-5208 – controlled by a debug server (gdbserver). This procedure is called "remote debugging". Therefore, access to the Linux development environment (Host computer) and to the ECUcore-5208 (Target) is required.

In the following, all steps necessary to debug a user application are exemplarily described for the demo project included in the VMware-Image of the Linux development system:

### **1. Starting the Debug server with the user program on the ECUcore-5208**

To debug simply start the demo program on the ECUcore-5208 directly from NFS directory of the Linux development system. Therefore, login to the command shell of the module is required (see section 5.5.1). Afterwards, complete the following steps in the Terminal program or Telnet client:

- 1.1 Integration of directory `"/projects/tftpboot"` from the Linux development system into the local file system of the ECUcore-5208 via NFS (see section 7.6.1):

```
mount -t nfs -o nolock 192.168.10.58:/tftpboot /mnt/nfs
```

- 1.2 Use command `"cd"` to switch to NFS directory in the local file system:

```
cd /mnt/nfs/demo
```

Directory `"/mnt/nfs"` on the ECUcore-5208 is identical with directory `"/tftpboot"` of the Linux development system in the VMware-Image. Accordingly, all files copied by the Makefile to directory `"/tftpboot/demo"` in the Linux development system are accessible by the ECUcore-5208 in directory `"/mnt/nfs/demo"` of its file system. It is not necessary to explicitly download executables to the ECUcore-5208.

- 1.3 The demo program needs the I/O driver `"pc5208drv"` to access the in- and outputs on the ECUcore-5208. Consequently, command `"insmod"` must be used to explicitly load the driver:

```
insmod 5208-spidev.ko  
insmod pc5208drv.ko
```

- 1.4 Starting the demo program on the ECUcore-5208 is controlled by the debug server. Therefore, command `"gdbserver"` is required and is called as follows:

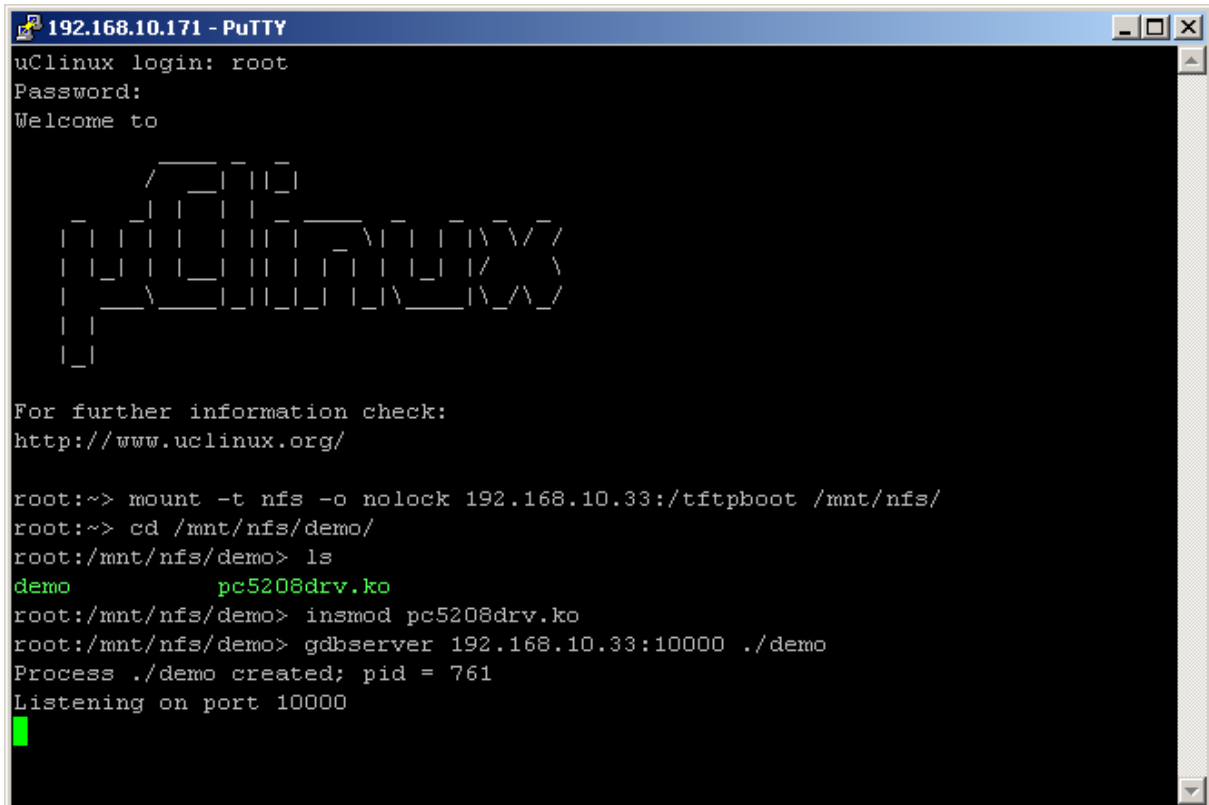
```
gdbserver <ip_vmware_image>:<port> <program> [args ...]
```

For the example above this would be the following call:

```
gdbserver 192.168.10.33:10000 ./demo
```

Use `"192.168.10.33"` as IP address of the Linux development system (compare section 6.5 for the determination of the IP address). The port number following the colon may be chosen freely (here: `"10000"`), but it must correspond with the port number of the Target connection within Eclipse as described in the following section 2.3 (also compare Figure 54).

Figure 51 illustrates the steps to be accomplished on the ECUcore-5208.



```
192.168.10.171 - PuTTY
uClinux login: root
Password:
Welcome to

          /-----\
         /          \
        /            \
       /              \
      /                \
     /                  \
    /                    \
   /                      \
  /                        \
 /                          \
/                            \
\                            /
 \                          /
  \                        /
   \                      /
    \                    /
     \                  /
      \                /
       \              /
        \            /
         \          /
          \-----/

For further information check:
http://www.uclinux.org/

root:~> mount -t nfs -o nolock 192.168.10.33:/tftpboot /mnt/nfs/
root:~> cd /mnt/nfs/demo/
root:/mnt/nfs/demo> ls
demo          pc5208drv.ko
root:/mnt/nfs/demo> insmod pc5208drv.ko
root:/mnt/nfs/demo> gdbserver 192.168.10.33:10000 ./demo
Process ./demo created; pid = 761
Listening on port 10000
```

Figure 51: Starting the debug server on the ECUcore-5208

To **simplify** this process, the delivery status of the ECUcore-5208 includes a shell script called **"debug.sh"** in the directory **"/home"**. This script summarizes all in this point 1 mentioned commands. Hence, the shell script must be started in the Terminal program or Telnet client so that entering commands manually is no longer necessary:

```
cd
./debug.sh
```

Command **"cd"** without any parameter switches to the home directory (**"/home"**) where the shell script **"debug.sh"** is located.

## **2. Configuring the Debugger in the IDE (only necessary once upon first call)**

The configuration of the Debugger in the IDE is only necessary once upon first call. The configuration takes place within "Eclipse" via menu item **"Run -> Debug..."**. Thus, a configuration dialog opens as shown in Figure 52. Hence, follow the steps listed below:

## 2.1 Determining the program that is to be executed in the Debugger (see Figure 52)

Therefore, the name of the program that is to be debugged must be entered in Tabsheet "Main" in area "C/C++ Application" (this would be "demo.gdb" for the above example).

**Important** on this step is to select the ELF file with the file extension **.gdb**. On the ECUcore-5208 the so called FLAT file format ist used while the debugger needs an ELF file. Both files are created by the C compiler gcc.

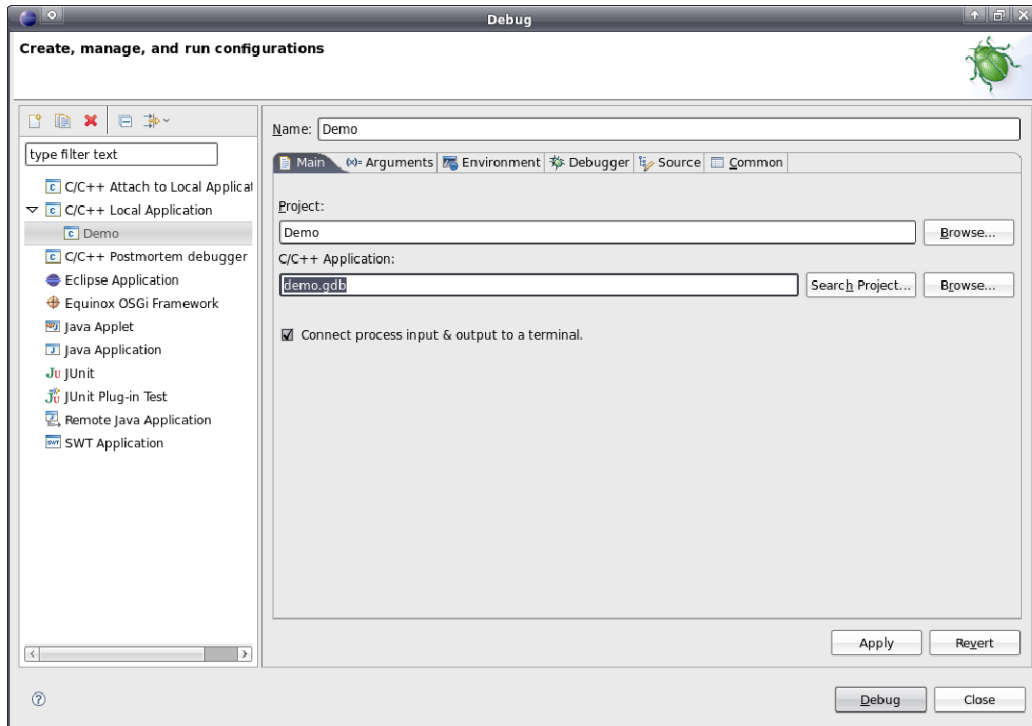


Figure 52: Determining the application that is to be debugged

## 2.2 Selecting the GDB Debugger to be used (see Figure 53)

On Host side use the GDB debugger that is included in the GNU crosscompiler toolchain for Freescale MCF52xx processors. To select it, activate Tabsheet "Debugger". Enter the complete path for the GDB Debugger of the Crosscompiler Toolchain in section "Debugger Options" in Sub-Tabsheet "Main" area "GDB debugger" (see Figure 53; for the above example this would be: "/projects/ECUcore-5208/toolchain-uclinux-4.2/bin/m68k-uclinux-gdb").

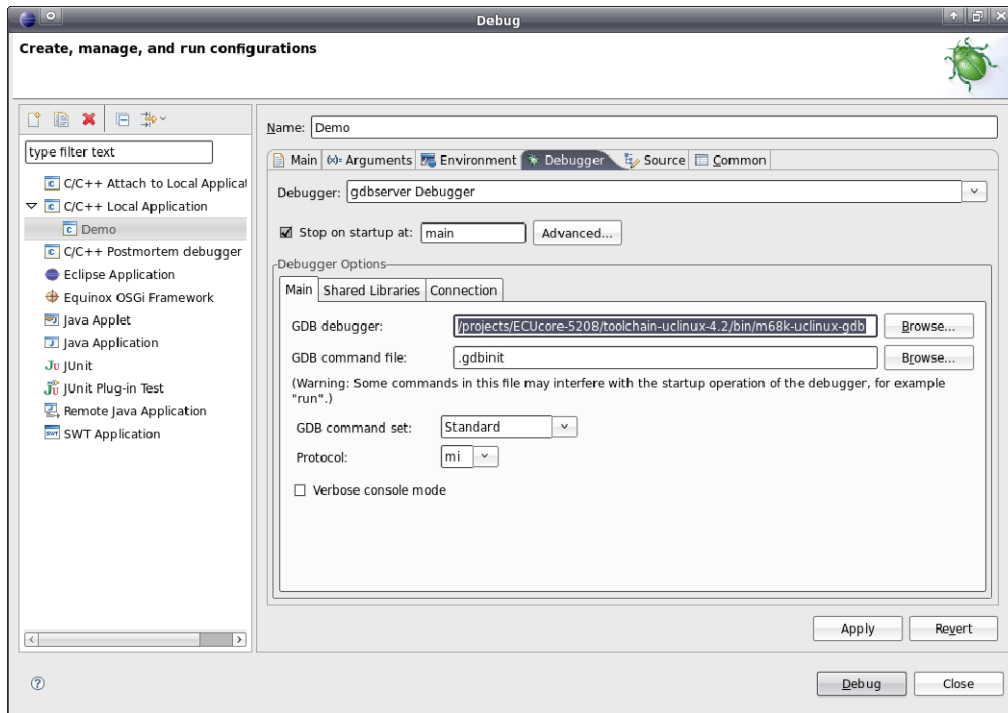


Figure 53: Selecting the GDB Debugger

### 2.3 Configuring the connection to the Target (see Figure 54)

The configuration of the connection to the target also takes place in Tabsheet "Debugger". Hence, enter the appropriate information in section "Debugger Options" in Sub-Tabsheet "Connection" (see Figure 54). In area "Host name or IP address" the IP address of the ECUcore-5208 as defined in section 5.4 must be entered ("192.168.10.171" for the example). Enter the Port number defined in point 1.4 in area "Port number" when command "gdbserver" is called (for this example: "10000").

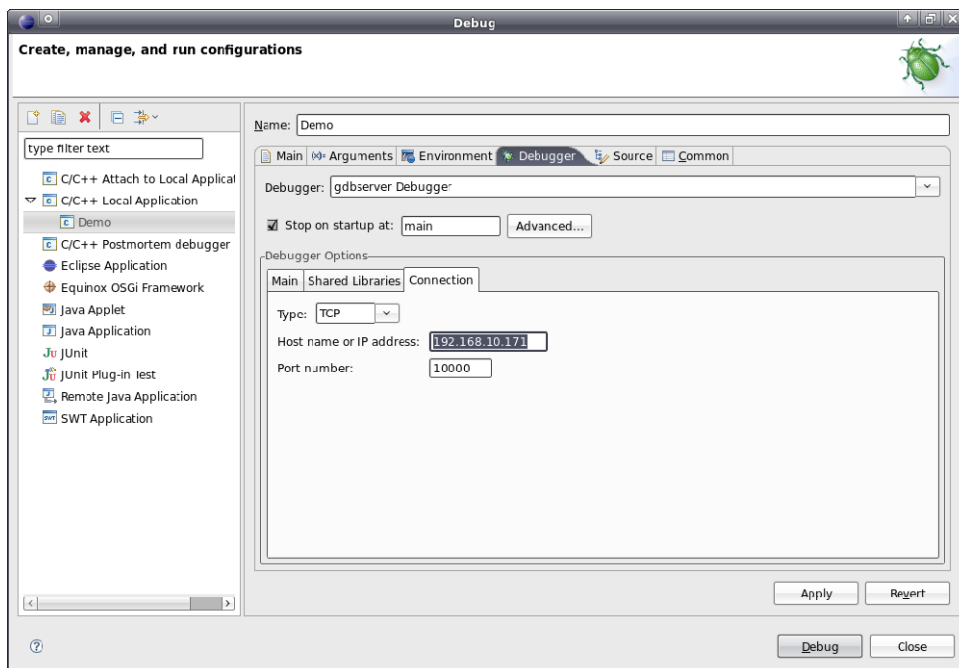


Figure 54: Configuring the connection to the Target



Thus, the configuration of the Debuggers is completed. By activating pushbutton "Debug", the Debugger starts with the current settings.

### 3. Executing the Debugger in the IDE

After the configuration described in point 2 is completed, the Debugger can be called in "Eclipse" via menu item "Run -> **Debug Last Launched**". Thereby, the IDE switched from "C/C++ Perspective" to "Debug Perspective" (see Figure 55). Table 10 lists up the most important Debugger commands.

Table 10: Overview of most important Debugger commands






Command	Function
F5 	Step Into
F6 	Step Over
F7 	Step Return
F8 	Run (if necessary until the next Breakpoint)
Ctrl+Shift+B (Double click)	Toggle Line Breakpoint
	Terminate

Figure 55 shows the Debugging process of the Demo program within the IDE "Eclipse". If the mouse pointer is positioned to a variable, its current value is shown.

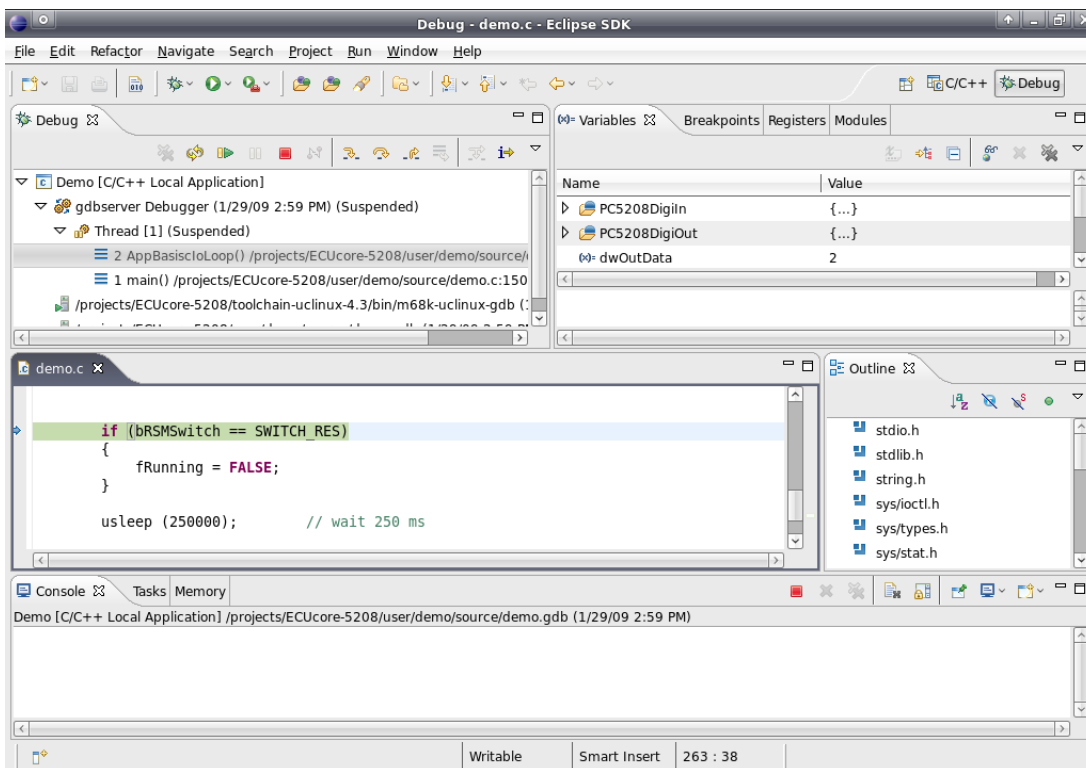


Figure 55: Debugging the Demo project in Eclipse

Figure 56 illustrates Terminal outputs that are generated on the ECUcore-5208 during the debugging process.

```

192.168.10.171 - PuTTY
root:~> mount -t nfs -o nolock 192.168.10.33:/tftpboot /mnt/nfs/
root:~> cd /mnt/nfs/demo/
root:/mnt/nfs/demo> ls
demo                pc5208drv.ko
root:/mnt/nfs/demo> insmod pc5208drv.ko
root:/mnt/nfs/demo> gdbserver 192.168.10.33:10000 ./demo
Process ./demo created: pid = 761
Listening on port 10000
Remote debugging from host 192.168.10.33

*****
  I/O demo application for SYSTEC ECUcore-5208
  (c) 2009 SYSTEC electronic GmbH
  -----
  Version: 1.00
  Build:   Jan 29 2009, 13:41:21
*****

I/O Driver version:  KernelModule=3.00, UserLib=3.00

Hardware:  CPU Board: 4218.00 (#00H)
           CPU PLD:  0.00  (#00H)
           IO Board: 4220.00 (#00H)
Driver:    Config:   0000H

Start basic I/O main loop...

DI=00-00-00  DO=00-00-01  bHexSw=0x00  bDipSw=0x00  R/S/M-Sw=RUN

```

Figure 56: Terminal outputs of the ECUcore-5208 during debugging

## 7.8 Configuration and Compiling of uClinux-Image and CoLilo

All uClinux sources for the ECUcore-5208 are stored in the VMware-Image of the Linux development system in directory `"/projects/ECUcore-5208/uClinux-dist"`. To modify the configuration of the uClinux kernel or of user space applications included in the Image, use command `"cd"` in a console window to switch to the directory for LinuxBSP. Afterwards, call command `"make menuconfig"`:

```
cd /projects/ECUcore-5208/uClinux-dist
make menuconfig
```

Figure 57 shows the typical surface for the configuration of uClinux-Kernel and user space applications.

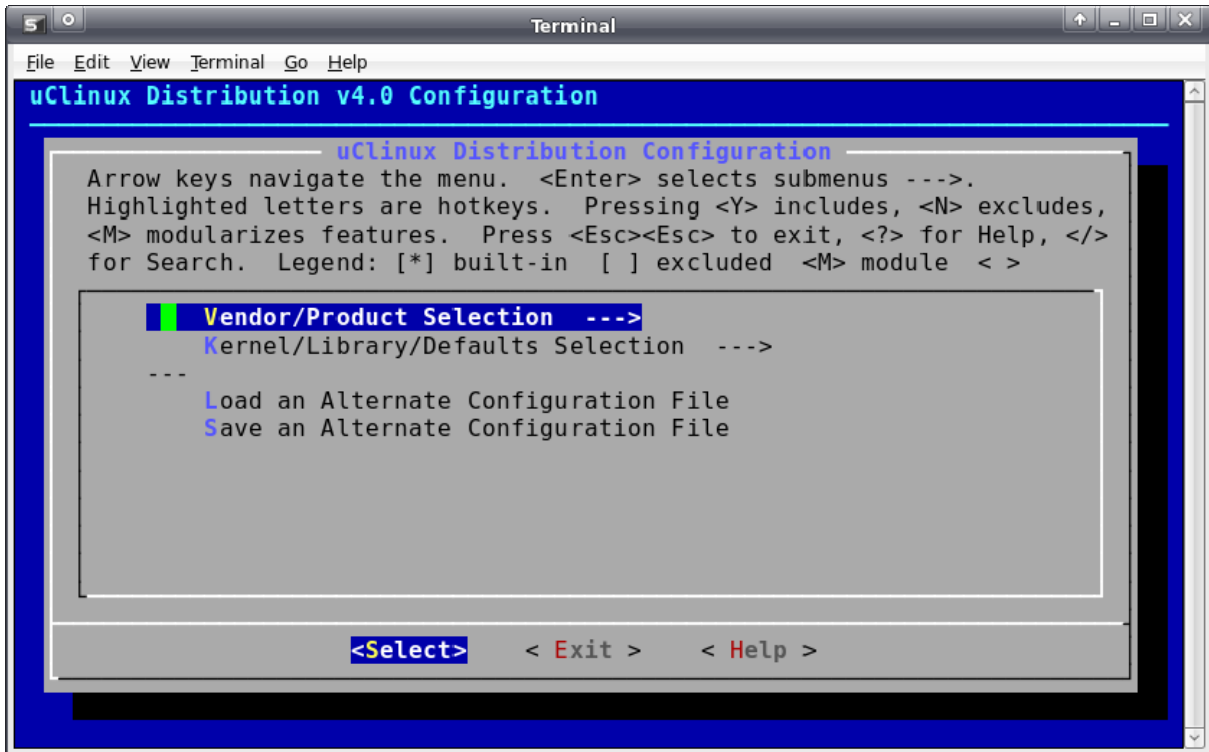


Figure 57: User surface for the configuration of the uClinux distribution

After finishing the configuration, uClinux kernel, user space applications are compiled by calling "make". Afterwards, they are merged to a new Image:

```
cd /projects/ECUcore-5208/uClinux-dist
make
```

By calling the following command the uClinux-Image will be copied into directory "/tftpboot". From there it can be loaded onto the ECUcore-5208 (see section 5.15):

```
cp images/image.bin /tftpboot/
```

To rebuild the bootloader "CoLilo", enter the command "make":

```
cd /projects/ECUcore-5208/colilo
make
```



## 9 Tips & Tricks for Handling Linux resp. uClinux

This chapter provides a brief summary of specific features that should be paid attention to when using Linux and therewith uClinux. It is only possible to address most important issues. If necessary, more detailed information can be gathered from appropriate Linux reference books.

- **Case-sensitive**

Contrary to DOS/Windows Linux (as other Unix derivatives) are case-sensitive on names for directories, files, programs, ...

Therefore it has to be make sure that commands have the proper case. Especially after using external access from windows it has to be ensured in the Linux-Image that the correct case has been used.

- **Line break**

Windows is using different control characters for a line break than other Unix derivatives (therefore also Linux). Under Windows a CR LF (carriage return & line feed = `\r\n`) sequence is used, while Linux uses just LF (line feed = `\n`). Most Linux programs can handle both variants, but there are chances that Windows programs cause displaying problems. Especially important is the correct (Linux) line break on the creation of shell scripts. Shell scripts using Windows line breaks will **not** be executed and simply raise an error.

- **Hidden files and directories**

Hidden file system entries are marked by a period "." at the beginning of a file or directory name. Those entries are not displayed by a simple list (e.g. using the command "ls"). Nevertheless to display hidden entries an additional parameter "-a" (all) has to be appended to "ls". This may relevant during FTP access.

- **Calling programs (search path)**

Similar to DOS/Windows, if a command is called Linux only searches through paths defined in the environment variable "PATH". It does not search through the current directory, except the current directory (".") is added to PATH which isn't done normally due to security reasons. For example, to call program "demo" which is located in the current directory, it must explicitly be pointed to the current directory by adding "./" to the call. Thus, program "demo" would be called as "./demo".

Standard commands such as "ls" can be called without specifying the path, because they are located within a path that is defined by the environment variable "PATH".

- **Calling programs (execution rights)**

To run a program in Linux, the corresponding file must be explicitly marked as executable. Responsible for this is the so called "x"-Flag in the file attributes ("eXecuteable") which are shown if "ls -l" is called, e.g.:

```
ls -l ./mountnfs.sh
-rwxr-xr-x  1 1000      users          2236 Jan 21  2009 ./mountnfs.sh
```

If this Flag is not set, the file is not marked as being executable and the respective command cannot be called. For example, this is generally the case after downloading a file via FTP. The "x"-Flag can again be set by using command "chmod":

```
chmod +x ./mountnfs.sh
```

Command "chmod" generally enables influencing all file attributes (setting and deleting). Details can be obtained by calling "chmod --help".

- **Automatic completion of entries via TAB button**

The automatic completion of file or command names via TAB button is a really convenient feature in Linux. As many as being unique characters will be inserted. If an input cannot be completed an additional TAB button press all possibilities will be displayed.

Example: After entering "l" and the doubled TAB pressing all commands starting with "l" will be displayed. After entering an "o" afterwards and pressing TAB the input will expand to "log" because there are no other possibilities (see list after 1st input on Figure 59). Adding an additionally "g" (input is now at "logg") TAB will expand the input line to "logger", because no other candidates are left. Figure 59 shows the complete example.

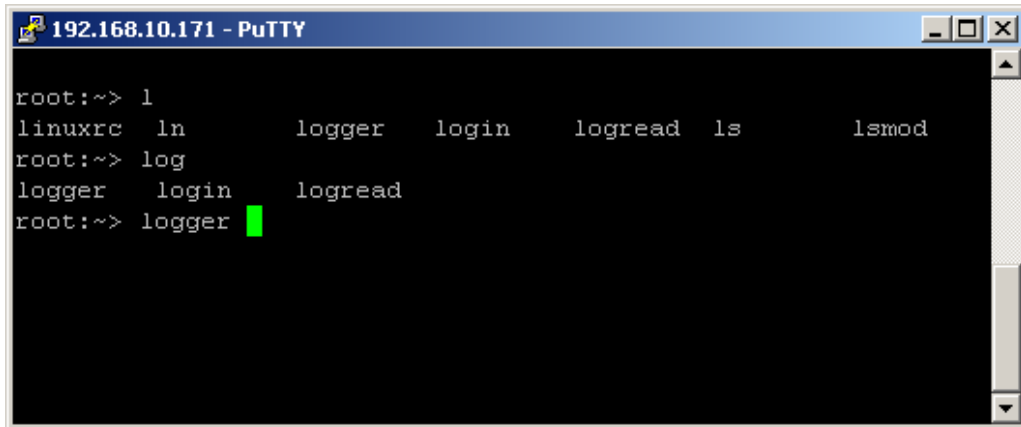


Figure 59: Tab completion in the shell

- **Displaying environment variables**

Displaying environment variables is possible by the command "echo". In shells environment variables are addressed by a prepended dollar sign "\$". To display the current search path the command "echo \$PATH" is suitable.

- **Setting environment variables**

When setting an environment variable, consider that it is only visible within the current shell instance. If for example, an environment variable is defined in a shell script, it will no longer be (re-)defined after the shell script is executed. The reason is for executing a script a new shell instance is started that processes the script. The respective environment variable is accessible within this shell instance. After finishing the script, the shell instance that was started to execute the script is closed. Consequently, also the environment variable definition or redefinition for this shell instance gets discarded.

One possibility to set environment variables (e.g. "TZ=") permanently is to define it in script "/home/etc/profile". This script is executed once when starting a login shell. This implies that all variables defined in that shell instance remain valid during the entire runtime. To relay environment variables to sub processes it has to be exported. To relay the "TZ" variable to sub processes the command "export TZ" has to be executed once which can also be done concurrently with the variable assignment.

- **Assistance in a program**

By entering the program name followed by "*--help*" (e.g. "*mount --help*"), it is possible in Linux to call brief help and assistance in a program including description of parameters used.

Alternatively the so called man pages (manual pages) can be used. To use them the command "*man*" followed by the program which man page should be opened needs to be appended, e.g. "*man ls*". The man pages can be scrolled up- and downwards using the arrow keys. To exit the man program the "*q*" (quit) simply has to be pressed. Lots of man pages are also available HTML page on the internet.

- **Error diagnostics in shell scripts**

To simplify error diagnostics when executing shell scripts, the script that is to be analyzed can be called via command "*sh -x <script\_file>*". Option "*-x*" instructs the shell to output each executed line on the console. Thus, it can be easily reproduced which branches of a conditional execution ("*if*", "*case*", "*while*" etc.) really have been taken.

- **Setting the time zone for the ECUcore-5208**

Setting the time zone for the ECUcore-5208 takes place by defining the environment variable "*TZ*" in start script "*/home/etc/profile*". The appropriate definition for Germany (UTC +1:00) including begin and end of summer time is already provided as a comment. The definition can be adjusted for other time zones.

- **Stack sizes of programs**

Programs being compiled for uClinux have a standard stack size of 4KiB. Because there is no MMU there is no way to handle case where the stack size is insufficient. In case of program aborts or worse kernel aborts or panics, a stack set too small could be the reason. To increase the application size a program called **flthdr** of the toolchain has to be used. An example is:

```
m68k-uclinux-flthdr -s 20480 romfs/bin/busybox
```

This sets the stack size of the busybox application from the directory romfs/bin (relative to the current working directory) to 20 KiB.

## Appendix A: GNU GENERAL PUBLIC LICENSE

The uClinux used on the ECUcore-5208 is licensed under GNU General Public License, version 2. The entire license text is specified below. A German translation is available from <http://www.gnu.de/documents/gpl-2.0.de.html>. Be advised that this translation is not official or legally approved.

Additional SYS TEC system software and programs developed by the user are **not** subject to the GNU General Public License!

### **GNU GENERAL PUBLIC LICENSE Version 2, June 1991**

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

#### **Preamble**

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software -- to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.



## **GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### END OF TERMS AND CONDITIONS

## How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>  
Copyright (C) <year> <name of author>
```

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author  
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.  
This is free software, and you are welcome to redistribute it under certain conditions;  
type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items -- whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

```
<signature of Ty Coon>, 1 April 1989  
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

# Index

/	
/etc/rc.d/fsconfig .....	27
/home .....	27, 28
/tmp .....	27
/var .....	27
<b>I</b>	
\\vm-xubuntu .....	37
<b>A</b>	
Accessory .....	12
adduser .....	23
Administration	
System requirements .....	13
Autorun .....	15
<b>B</b>	
boa .....	29
<b>C</b>	
CAN driver .....	49
CAN0 .....	10
candrv .....	49
chmod .....	53, 68
CoLilo command prompt	
Activation .....	14
Terminal settings .....	14
CoLilo commands	
Ethernet configuration .....	16
Update uClinux-Image .....	31
COM0 .....	10
COM1 .....	10
Configuration	
Commands .....	16
Readout and display .....	25
Configuration mode .....	14
Connection network drive .....	38
<b>D</b>	
date .....	24
Debugger commands .....	64
deluser .....	23
Demo project	
CAN driver .....	49
Development Board	
Connections .....	10
Control elements .....	11
Development Kit .....	9
df .....	27
DHCP .....	35
ECUcore-5208 .....	17
Dimensions .....	7
<b>E</b>	
Eclipse .....	57
Configuring Debugger .....	61
Debugger commands .....	64
Debugging .....	59
Open project .....	58
Translating project .....	59
EMV law .....	5
Environment variables	
Display .....	69
Setting .....	69
ETH0 .....	10
Execution rights .....	68
<b>F</b>	
File system .....	27
Files	
preinstalled .....	28
fsconfig .....	27
FTP .....	53
FTP client .....	13
FTP server .....	54
ftpd .....	54
ftpget .....	53
ftpput .....	54
<b>G</b>	
gbrdcfg .....	25
gdbserver .....	60
GNU .....	7
<b>H</b>	
hellocan .....	49
HTTP server .....	29
hwclock .....	24
<b>I</b>	
ifconfig .....	37
insmod .....	48, 49
iodrvdemo .....	67
<b>L</b>	
Linux .....	7
Linux VMware-Image .....	35
LinuxBSP .....	65
<b>M</b>	
Manuals	
Overview .....	6
mount .....	52
<b>N</b>	
net use .....	38
Network environment .....	37
NFS	
mount .....	52

**P**

PATH .....68  
 Predefined user accounts.....22  
 (development system).....36

**S**

Search path .....68  
 Setting RTC .....24  
 Setting system time .....24  
 setup-ecucore-5208.sh.....29  
 SO-1096.exe .....34  
 Software structure .....45  
 Software update  
     CoLilo .....33  
     uClinux-Image.....30  
 System start.....15

**T**

TAB completion .....69  
 Telnet  
     Login to ECUcore-5208 .....21  
     Login to Linux development system .....38  
 Telnet client .....13  
 Terminal program .....13  
 Terminal settings .....14  
 Testing Hardware Connections .....67  
 TFTP32 .....30  
 Time zone.....70

TZ..... 70

**U**

uClinux ..... 7  
 USB-RS232 Adapter Cable ..... 12  
 User accounts  
     Addition and deletion..... 23  
     Changing password ..... 23  
     Predefined ..... 22  
     predefined (development system)..... 36

**V**

VMware-Image  
     Determining IP address..... 37  
     Determining static IP address ..... 42  
     Installation ..... 34  
     Keyboard layout ..... 39  
     Overview ..... 34  
     Shrinking ..... 44  
     Start ..... 35  
     System update ..... 43  
     Time zone..... 41  
 VMware-Player  
     Installation ..... 34

**W**

Windows network environment..... 37  
 WinSCP ..... 21

**Document:** System Manual ECUcore-5208  
**Document number:** L-1202e\_1, 1st Edition August 2009

---

**How would you improve this manual?**

---

---

---

---

**Did you detect any mistakes in this manual?**

Page

---

---

---

---

**Submitted by:**

Customer number: \_\_\_\_\_

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

---

**Please return your suggestions to:** SYS TEC electronic GmbH  
August-Bebel-Str. 29  
D - 07973 Greiz  
GERMANY  
Fax : +49 (0) 36 61 / 6279-99  
Email: [info@systec-electronic.com](mailto:info@systec-electronic.com)