

# ***System Manual*** ***ECUcore-iMX35***

## **User Manual** **Version 1.0**

**Edition June 2014**

Document No.: L-1569e\_01



详情请通过[sales@hkaco.com](mailto:sales@hkaco.com)联系我们

北京：010-5781 5068  
上海：021-6728 3703  
广州：020-3874 3032  
西安：029-8187 3816

## Status/Changes

Status: released

<b>Date/Version</b>	<b>Section</b>	<b>Changes</b>	<b>Editor</b>
2014/06/17 1.0	All	Creation	T. Volckmann

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 2014 SYS TEC electronic GmbH. All 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:

1st Edition June 2014

## 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>8</b>
<b>4</b>	<b>Development Kit ECUcore-iMX35.....</b>	<b>10</b>
4.1	Overview.....	10
4.2	Electric commissioning of the Development Kit ECUcore-iMX35 .....	11
4.3	Control elements of the Development Kit ECUcore-iMX35.....	12
4.4	Optional accessory .....	13
4.4.1	USB-RS232 Adapter Cable .....	13
4.4.2	Driver Development Kit (DDK).....	13
<b>5</b>	<b>Application and Administration of the ECUcore-iMX35.....</b>	<b>14</b>
5.1	System requirements and necessary software tools.....	14
5.2	System start of the ECUcore-iMX35 .....	15
5.2.1	Activation/Deactivation of Linux Autostart .....	15
5.2.2	Autostart for user software.....	16
5.3	Command prompt of the bootloader "U-Boot" .....	17
5.4	Ethernet configuration of the ECUcore-iMX35 .....	18
5.5	Login to the ECUcore-iMX35.....	21
5.5.1	Login to the command shell.....	21
5.5.2	Login to the FTP server .....	22
5.6	Predefined user accounts.....	23
5.7	Adding and deleting user accounts .....	24
5.8	How to change the password for user accounts .....	24
5.9	Setting the system time .....	25
5.10	Readout and displaying "U-Boot" configuration data .....	26
5.11	Showing the installed Linux-Version .....	27
5.12	File system of the ECUcore-iMX35 .....	27
5.13	Preinstalled files in the directory "/home" .....	28
5.14	Using the HTTP server .....	29
5.15	HMI Components.....	30
5.15.1	Supported HMI Devices.....	30
5.15.2	Connection of Scrollwheel and Matrix Keyboard.....	32
5.15.3	Setting Display Brightness.....	33
5.15.4	Calibration of the Touchscreen.....	34
5.16	The Use of Qt on the ECUcore-iMX35 .....	34
5.16.1	Overview of the Qt-Components for the ECUcore-iMX35.....	34
5.16.2	Qt-Environment Variables.....	35
5.16.3	Starting Qt-Programs.....	35
5.17	Updating the Linux-Image .....	36
5.18	Updating the bootloader "U-Boot" .....	38
<b>6</b>	<b>VMware-Image with Linux Development System .....</b>	<b>39</b>
6.1	Overview.....	39
6.2	Installing the Linux VMware-Image .....	39
6.3	Starting the Linux VMware-Image .....	39
6.4	User accounts to log in to the Linux development system .....	41
6.5	Determining the IP address of the Linux development system .....	42
6.6	Access to the Linux development system from a Windows computer .....	42
6.6.1	Access via Windows network environment .....	42
6.6.2	Access via Telnet client .....	44
6.7	Personal configuration and actualization of the Linux VMware-Image .....	44

6.7.1	Adjustment of keyboard layout and time zone.....	44
6.7.2	Adjusting the desktop size .....	47
6.7.3	Setting a static IP address for the Linux VMware-Image .....	47
6.7.4	System update of the Linux VMware-Image.....	49
6.7.5	Changing the computer name in the Windows network environment .....	50
6.7.6	Shrinking the VMware-Image .....	50
<b>7</b>	<b>Software Development for the ECUcore-iMX35.....</b>	<b>51</b>
7.1	Software structure of the ECUcore-iMX35 .....	51
7.2	Makefile and environment variables to create projects .....	52
7.3	I/O Driver for the ECUcore-iMX35.....	53
7.3.1	Integration of the I/O Driver into own user projects .....	53
7.3.2	I/O Driver Demo project.....	55
7.4	CAN Driver for the ECUcore-iMX35 .....	55
7.4.1	Integration of CAN Driver into own user projects .....	55
7.4.2	CAN Driver Demo project .....	56
7.5	Transferring programs to the ECUcore-iMX35.....	58
7.5.1	Using NFS.....	58
7.5.2	Using FTP .....	59
7.6	Translation and execution of demo project "demo" .....	61
7.6.1	Usage of "make" .....	61
7.6.2	Using graphical IDE "Eclipse".....	63
7.7	Configuration and Translation of Linux-Image and U-Boot.....	72
<b>8</b>	<b>Adaptation and Testing of the hardware connections.....</b>	<b>74</b>
8.1	Driver Development Kit (DDK) for the ECUcore-iMX35 .....	74
8.2	Testing the hardware connections .....	75
<b>9</b>	<b>Using the USB and SD interface.....</b>	<b>77</b>
9.1	Using the USB interface .....	77
9.2	Using the SD interface.....	79
<b>10</b>	<b>Tips &amp; Tricks for Handling Linux.....</b>	<b>80</b>
	<b>Index.....</b>	<b>87</b>

# 1 Introduction

Thank you that you have decided for the SYS TEC ECUcore-iMX35. This product provides to you an innovative and high-capacity single board computer subassembly with Linux operating system. Due to its integrated Target Visualization, high performance as well as extensive on-board periphery, it is particularly suitable for communication and control units for HMI applications in embedded systems.

Please take some time to read through this manual carefully. It contains important information about the commissioning, configuration and programming of the ECUcore-iMX35. It will assist you in getting familiar with the functional range and usage of the ECUcore-iMX35. 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-iMX35. Please also refer to those complementary documents.

## Declaration of Electro Magnetic Conformity for ECUcore-iMX35 (EMC law)



The ECUcore-iMX35 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-iMX35 is ESD-sensitive and may only be unpacked, used and operated by trained personal at ESD-conform work stations.

The ECUcore-iMX35 is a module for the application in automation technology. It is programmable under Linux 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-iMX35 based on the Development Kit ECUcore-iMX35 as well as general procedures for software development of this module. There are different hardware manuals for all hardware components such as the ECUcore-iMX35, development boards and reference circuitry. Software-sided, the ECUcore-iMX35 is delivered with preinstalled Embedded Linux. 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-iMX35. The VMware-Image can be used unmodified within different host systems. Table 1 lists up all relevant manuals for the ECUcore-iMX35.

Table 1: Overview of relevant manuals for the ECUcore-iMX35

Information about...	In which manual?
Basic information about the ECUcore-iMX35 (configuration, administration, connection assignment, software development, reference designs et cetera.)	In this manual
Application of the ECUcore/PLCcore-iMX35 as PLC (Programming of the PLCcore-iMX35 as PLC according to IEC 61131-3, Process Image, Data exchange via Shared Process Image with external applications et cetera)	System Manual PLCcore-iMX35 (Manual no.: L-1567)
Hardware description for the ECUcore-iMX35, reference designs et cetera	Hardware Manual ECUcore-iMX35 (Manual no.: L-1570)
Development Board for the ECUcore-iMX35, reference designs et cetera	Hardware Manual Development Board iMX35 (Manual no.: L-1571)
Driver Development Kit (DDK) for the ECUcore-iMX35	Software Manual Driver Development Kit (DDK) for the ECUcore-iMX35 (Manual no.: L-1572)
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-iMX35 on the basis of the Development Kit ECUcore-iMX35.
- Section 5** exemplifies **details about the usage of the ECUcore-iMX35**, 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-iMX35 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 10** provides **tips & tricks** to simplify the usage of Linux. This section is especially helpful for newcomers of using Linux.



### 3 Product Description

The ECUcore-iMX35 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 subassembly that is programmable under Linux and has available an integrated Target Visualization. Due to CAN and Ethernet interfaces, the ECUcore-iMX35 is best suitable to realize custom specific HMI (**H**uman **M**achine **I**nterface) applications.



Figure 1: Top view of the ECUcore-iMX35

These are some significant features of the ECUcore-iMX35:

- High-performance CPU kernel (ARM 32-Bit ARM1136JF-S, 532 MHz CPU Clock, 740 MIPS)
- 128 MByte SDRAM Memory, 128 MByte FLASH Memory
- LCD Controller supports up to 800x600 pixel resolution with 24-bit color depth
- Support for Scrollwheel und 4x4 Matrix keypad
- 1x 10/100 Mbps Ethernet LAN interface (with on-board PHY)
- 2x CAN 2.0B interface, usable as CANopen Manager (CiA 302-conform)
- 3x asynchronous serial ports (UART)
- 16 digital inputs, 10 digital outputs (standard configuration, modifiable via DDK)
- Externally usable SPI and I<sup>2</sup>C
- On-board peripherals: RTC, watchdog, power-fail input
- Operating system: Linux
- Small dimensions (78 x 54 mm)

Making available a complete single board computer subassembly as an insert-ready core module with small dimensions, reduces effort and costs significantly for the development of user-specific controls. The ECUcore-iMX35 is also very well suitable as basic component for custom specific HMI devices as well as an intelligent network node for decentralized processing of process signals (CANopen and UDP).

Based on the default I/O configuration, the module features 16 digital inputs (DI0...DI15, 3.3V level), 10 digital outputs (DO0...DO9, 3.3V level) as well as Scrollwheel and 4x4 Matrix Keypad support. This default I/O configuration can be adapted for specific application requirements by using the Driver Development Kit (SO-1119). Saving the user application in the on-board Flash-Disk of the module allows an automatic restart in case of power breakdown.

Das ECUcore-iMX35 is based on Embedded Linux as operating system. This allows for simultaneous execution of several user-specific programs.

The Embedded Linux applied to the ECUcore-iMX35 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-1121). If you require the LinuxBSP sources independently from the VMware-Image of the Linux development system, please contact our support:

## 4 Development Kit ECUcore-iMX35

### 4.1 Overview

Due to the Development Board contained in the Kit, the Development Kit ECUcore-iMX35 allows for a quick commissioning of the ECUcore-iMX35 and simplifies the design of prototypes for user-specific applications that are based on this module. Among other equipment, the Development Board features a QVGA Display with Touchscreen, several possibilities for power supply, Ethernet interfaces, connections for CAN bus, connectors for USB and SD card, 4 push buttons and 4 LED as control elements for the digital in- and outputs. Signals that are available from plug connectors of the ECUcore-iMX35 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-iMX35.



Figure 2: Development Kit ECUcore-iMX35

The Development Kit ECUcore-iMX35 ensures quick and problem-free commissioning of the ECUcore-iMX35. Therefore, it combines all hard- and software components that are necessary to create own applications: the core module ECUcore-iMX35 itself, the corresponding Development Board containing the Display, 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-iMX35.

The Development Kit ECUcore-iMX35 contains the following components:

- ECUcore-iMX35
- Development Board for the ECUcore-iMX35
- 12V – 1,5A Power adapter
- Ethernet cable
- RS232 cable
- DVD with Linux development system, examples, documentation and other tools (SO-1120)

The Linux development system included in the Kit serves as software development platform and as debug environment for the ECUcore-iMX35. 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-iMX35

The power adapter necessary for running the Development Kit ECUcore-iMX35 as well as 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 (X100/X101), COM0 (X701A) and ETH0 (X702). Table 2 provides an overview over the connections of the Development Kit ECUcore-iMX35.

Table 2: Connections of the Development Kit ECUcore-iMX35

Connection	Labeling on the Development Board	Remark
Power supply	X100 or X101	The power adapter included in the delivery is intended for direct connection to X1101
ETH0 (Ethernet)	X702	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)	X701A	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)	X701B	Interface can be used freely for the user program.
COM2 (RS485)	X700	Interface can be used freely for the user program.
CAN0 (CAN)	X801A	Interface can be used freely for the user program.
CAN1 (CAN)	X801B	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-iMX35. Instead of using the 24V DC power adapter included in the Kit, the power supply may optionally take place via X100 with an external source of 24V/1,5A.

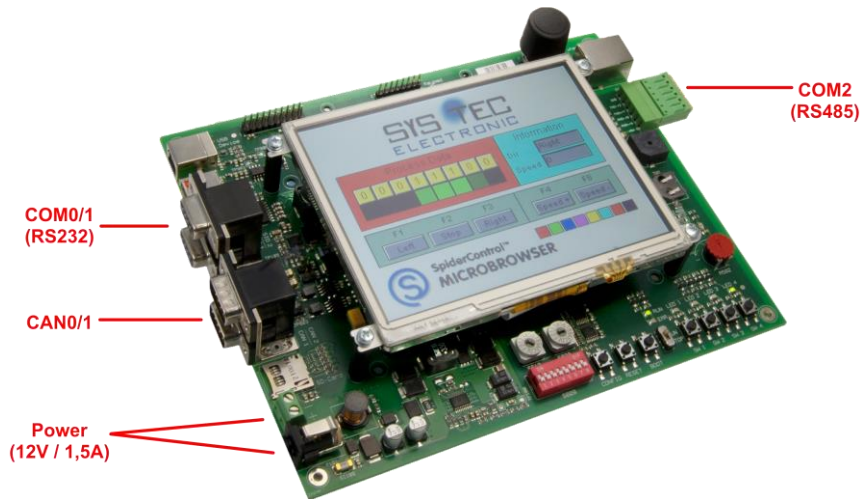


Figure 3: Positioning of the most important connections on the Development Board for the ECUcore-iMX35

**Advice:** Upon commissioning, cables for Ethernet (ETH0, X702) and RS232 (COM0, X701A) must be connected prior to activating the power supply (X100 / X101).

### 4.3 Control elements of the Development Kit ECUcore-iMX35

The Development Kit ECUcore-iMX35 allows for easy commissioning of the ECUcore-iMX35. It has available various control elements to configure the module and to simulate in- and outputs for the usage of the ECUcore-iMX35 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-iMX35

Control element	Name	Meaning
Pushbutton 0	S604	Digital Input DI0
Pushbutton 1	S605	Digital Input DI1
Pushbutton 2	S606	Digital Input DI2
Pushbutton 3	S607	Digital Input DI3
LED 0	D602	Digital Output DO0
LED 1	D603	Digital Output DO1
LED 2	D604	Digital Output DO2
LED 3	D605	Digital Output DO3
Run/Stop Switch	S603	Control element can be used freely for the user program (e.g. Run / Stop to operate the control program)

Run-LED	D600	Control element can be used freely for the user program (e.g. Display of activity state of the control program)
Error-LED	D601	Control element can be used freely for the user program (e.g. Display of error state of the control program)
Hex-Encoding Switch	S608/S610	Control element can be used freely for the user program (e.g. Configuration of node address CAN0)
DIP-Switch	S609	Control element can be used freely for the user program (e.g. Configuration bitrate and master mode CAN0)

## 4.4 Optional accessory

### 4.4.1 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-iMX35 from PCs, e.g. laptop computers which do not have RS232 interfaces any more (see section 5).



Figure 4: SYS TEC USB-RS232 Adapter Cable

### 4.4.2 Driver Development Kit (DDK)

The ECUcore-iMX35 Driver Development Kit (order number SO-SO1119) allows the user to independently adjust the I/O level to his own baseboard. Section 8.1 provides information about the Driver Development Kit.

## 5 Application and Administration of the ECUcore-iMX35

### 5.1 System requirements and necessary software tools

The administration of the ECUcore-iMX35 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 4.4.1) that provides an appropriate RS232 interface via USB port.

All examples referred to in this manual are based on an administration of the ECUcore-iMX35 using a Windows computer. Procedures using a Linux computer would be analogous.

To administrate the ECUcore-iMX35 the following software tools are necessary:

**Terminal program** A Terminal program allows the communication with the **command shell** of the ECUcore-iMX35 via a **serial RS232 connection to COM0 of the ECUcore-iMX35**. This is required for the Ethernet configuration of the ECUcore-iMX35 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 "*HyperTerminal*" which is included in the Windows delivery or "*TeraTerm*" which is available as Open Source and meets higher demands (downloadable from: <http://tssh2.sourceforge.jp>).

**Telnet client** Telnet-Client allows the communication with **command shell** of the ECUcore-iMX35 via **Ethernet connection to ETH0 of the ECUcore-iMX35**. Using Telnet clients requires a completed Ethernet configuration of the ECUcore-iMX35 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-iMX35).

Suitable as Telnet client would be "*Telnet*" which is included in the Windows delivery or "*TeraTerm*" which can also be used as Terminal program (see above).

**FTP client** An FTP client allows for file exchange between the ECUcore-iMX35 (ETH0) and the computer. This allows for example **editing configurations files** by transferring those from the ECUcore-iMX35 onto the computer where they can be edited and transferred back onto to the ECUcore-iMX35. Downloading files onto the ECUcore-iMX35 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 booted 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 Linux-Image on the ECUcore-iMX35. By default, a preconfigured TFTP server is included in the VMware-Image of the Linux development system. To alternatively update the Linux-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 booted 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-iMX35

### 5.2.1 Activation/Deactivation of Linux Autostart

During standard operation mode, the bootloader "U-Boot" automatically starts the Linux 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 Linux-Image (see section 0), it is necessary to disable this Autostart mode and to switch to "U-Boot" command prompt instead (configuration mode).

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

Table 4 lists up all conditions that are verified by the bootloader "U-Boot". All of them must be complied with to start an Autostart for the Linux-Image.

Table 4: Conditions for booting Linux

No.	Condition	Remark
1	Connection "/BOOT" = High (pushbutton S602 on the Development Board <b>not</b> pressed)	The Linux Autostart is released only if the signal "/BOOT" is at H-level ("BOOT" is not active).  The position of connection "/BOOT" on the module pin connector is defined in the Hardware Manual PLCcore-iMX35 (Manual no.: L-1570).
2	<b>No</b> abort of Autostart via COM0 of the ECUcore-iMX35	If the conditions above are met, "U-Boot" checks the serial interface COM0 of the ECUcore-iMX35 for about 1 second after Reset regarding the reception of a SPACE character (ASCII 20H). If such a character is received within that time, "U-Boot" will disable the Linux Autostart and will activate its own command prompt instead.



According to Table 4, the Linux boot is disabled after Reset (e.g. pushbutton S601 on the Development Board) and the "U-Boot" command prompt is activated instead if the following conditions occur:

- (1) `/BOOT = "Low"` Development Board: `/BOOT` = pushbutton S602
- OR -
- (2) Reception of a SPACE character (ASCII 20H) within 1 second after Reset

After activating the Reset pushbutton (e.g. pushbutton S601 on the Development Board), the "U-Boot" command prompt answers.

Communication with the bootloader "U-Boot" only takes place via the serial interface COM0 of the ECUcore-iMX35. 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):

- 115200 Baud
- 8 Data bit
- 1 Stop bit
- no parity
- no flow control

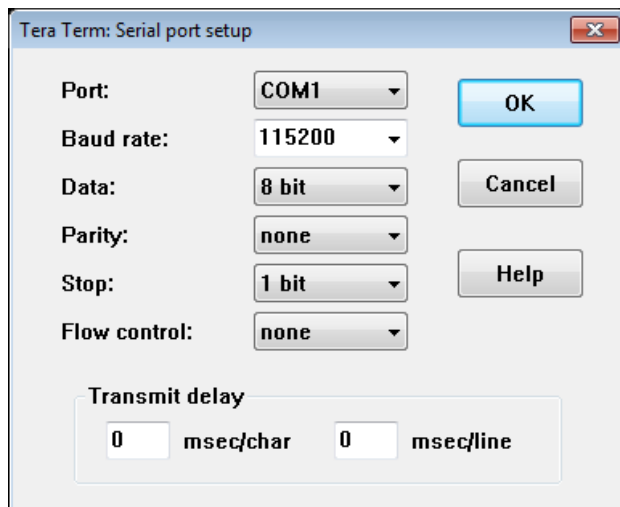


Figure 5: Terminal configuration using the example of "TeraTerm"

## 5.2.2 Autostart for user software

By default, the ECUcore-iMX35 starts running the Linux operation system upon Power-on or Reset which loads all necessary software components to execute the user software afterwards. Hence, the ECUcore-iMX35 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:

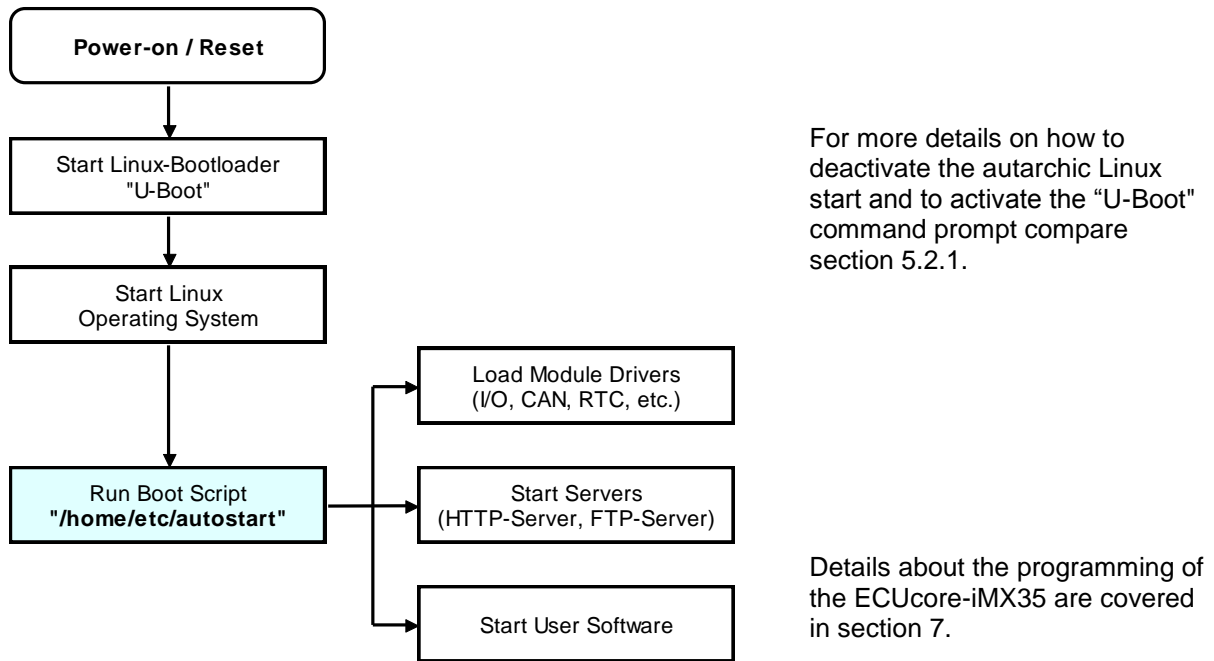


Figure 6: System start of the ECUcore-iMX35

It is possible to configure the ECUcore-iMX35 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. By entering command `"pureftp"` for example, it is possible that the FTP server is called automatically during booting the ECUcore-iMX35. The script can be edited directly on the ECUcore-iMX35 in the FTP client `"WinSCP"` (see section 5.1) using pushbuttons `"F4"` or `"F4 Edit"`.

### 5.3 Command prompt of the bootloader "U-Boot"

After hardware reset of the ECUcore-iMX35, the bootloader "U-Boot" is the first software component to start immediately. During standard operation mode, the bootloader automatically loads the Linux operating system that brings about the executions of other user applications. Basic configuration settings for the ECUcore-iMX35 are determined within the bootloader "U-Boot". The command prompt of the bootloader "U-Boot" is primarily needed for the following tasks:

- Ethernet configuration of the ECUcore-iMX35 (see section 5.4) and
- Update of the Linux-Image (see section 0)

Section 5.2.1 describes the procedure to activate the "U-Boot" command prompt. Entering `"?"` will show help for all available commands, the command `"printenv"` shows the current module configuration (see Figure 7).

```

Tera Term - COM1 VT
File Edit Setup Control Window Help
U-Boot> printenv
baudrate=115200
loadaddr=0x80800000
dnsip=192.168.10.5
ethact=FE00
mtdevname=U-boot
bootargs=console=ttyncd,115200 mohl1 rootfstype=squashfs root=1F05 mtparts=physnap-flash.0:384k(U-boot),128k(U-boot.env),128k(u-boot.env-redundant),256k(mtboots),4096k(Kernel),20480k(
partition=0,0
bootdelay=1
Environment size: 696/131067 bytes
U-Boot>

```

Figure 7: Displaying the current module configuration in the bootloader "U-Boot"

The configuration settings administered by the Bootloader "U-Boot" can also be accessed from Linux; section 5.10 describes the procedure.

## 5.4 Ethernet configuration of the ECUcore-iMX35

The main Ethernet configuration of the ECUcore-iMX35 takes place within the bootloader "U-Boot" and is taken on for all other software components (Linux, user software, HTTP server etc.). The Ethernet configuration is carried out via the serial interface COM0. **Therefore, the "U-Boot" command prompt must be activated as explained in section 5.2.1.** Table 5 lists up "U-Boot" commands necessary for the Ethernet configuration of the ECUcore-iMX35.

Table 5: "U-Boot" configuration commands of the ECUcore-iMX35

Configuration	Command	Remark
MAC address	setenv ethaddr <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-iMX35	setenv ipaddr <xxx.xxx.xxx.xxx>	This command sets the local IP address of the ECUcore-iMX35. The IP address is to be defined by the network administrator.  To assign a dynamic IP address to the ECUcore-iMX35 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	setenv netmask <xxx.xxx.xxx.xxx>	This command sets the network mask of the ECUcore-iMX35. The network mask is to be defined by the network administrator.
Gateway address	setenv gatewayip <xxx.xxx.xxx.xxx>	This command defines the IP address of the gateway which is to be used by the ECUcore-iMX35. The gateway address is set by the network administrator.  <b>Advice:</b> If ECUcore-iMX35 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	setenv serverip <xxx.xxx.xxx.xxx>	This command defines the IP address of the Linux development system. It is used for example to update the Linux-Image (see section 0) as well as to include the Linux development system via NFS into the local file system of the ECUcore-iMX35 (see section 7.5.1).  The procedure to determine the IP address of the Linux development system describes section 6.5.
Saving the configuration	saveenv	This command saves active configurations in the flash of the ECUcore-iMX35.

Modified configurations may be verified again by entering "*printenv*" in the "U-Boot" command prompt. Active configurations are permanently saved in the Flash of the ECUcore-iMX35 by command

**saveenv**

Modifications are adopted upon next Reset of the ECUcore-iMX35.

```

Tera Term - COM1 VT
File Edit Setup Control Window Help

U-Boot 2010.09-v1.0.0-00024-gc8f8cbf (Jun 04 2014 - 15:39:15)
CPU: Freescale i.MX35 at 532 MHz
Board: ECUcore-iMX35 (POR)
RCSR: 00000800
DRAM: 128 MiB
Flash: 128 MiB
In: serial
Out: serial
Err: serial
mx35 cpu clock: 532MHz
ipg clock : 665000000Hz
ipg_per clock : 665000000Hz
uart clock : 100000000Hz
Net: FEC0
Hit any key to stop autoboot: 0
U-Boot> setenv ipaddr 192.168.10.248
U-Boot> setenv netmask 255.255.255.0
U-Boot> setenv gateway 0.0.0.0
U-Boot> setenv serverip 192.168.10.116
U-Boot> saveenv
Saving Environment to Flash...
. done
Un-Protected 1 sectors
. done
Un-Protected 1 sectors
Erasing Flash...
. done
Erased 1 sectors
Writing to Flash... 9...8...7...6...5...4...3...2...1...done
. done
Protected 1 sectors
. done
Protected 1 sectors
U-Boot>

```

Figure 8: Saving the module configuration of the ECUcore-iMX35

### Advice about the application of DHCP:

The Embedded Linux of the ECUcore-iMX35 is able to request a dynamic IP address via DHCP. Therefore, the local IP address is to be configured as "0.0.0.0":

```
setenv ipaddr 0.0.0.0
```

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

- DHCP is only supported by Linux, but not by the bootloader "U-Boot". For example, to update the Linux-Image (see section 0), it is necessary to assign a (temporary), static IP address to the module.
- To create a Telnet or FTP connection to the ECUcore-iMX35 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-iMX35. Therefore, logging in to the command shell of the ECUcore-iMX35 must be accomplished as described in section 5.5.1. Afterwards, the current IP address may be queried using command "*ifconfig*".

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

After Reset (e.g. pushbutton S601 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-iMX35 is no longer necessary.

## 5.5 Login to the ECUcore-iMX35

### 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-iMX35. 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-iMX35 – 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. Telnet or also TeraTerm) via the Ethernet interface **ETH0** of the ECUcore-iMX35.

For logging in to the ECUcore-iMX35 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 192.168.10.248
```

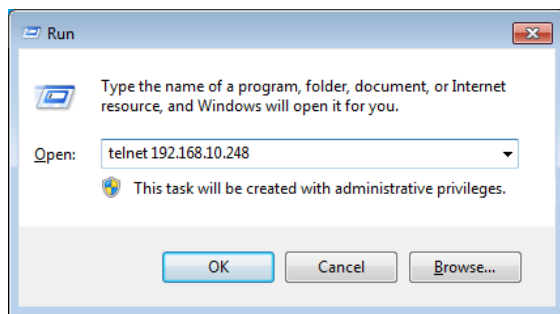


Figure 9: Calling the Telnet client in Windows

Logging in to the ECUcore-iMX35 is possible in the Terminal window (if connected via COM0) or 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-iMX35 (also compare section 5.6):

User: *PlcAdmin*  
Password: *Plc123*

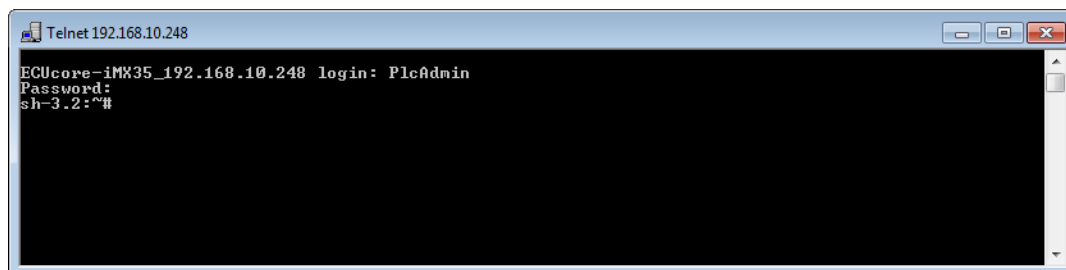


Figure 10: Login to the ECUcore-iMX35

Figure 10 exemplifies the login to the ECUcore-iMX35 using a Windows standard Telnet client.

## 5.5.2 Login to the FTP server

The ECUcore-iMX35 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 started manually if required. Therefore, the user must first be logged in to the command shell of the ECUcore-iMX35 following the procedures described in section 5.5.1. Afterwards, the following command must be entered in the Telnet or Terminal window:

```
pureftp
```

Figure 11 illustrates an example for starting the FTP server ("*pureftp*").

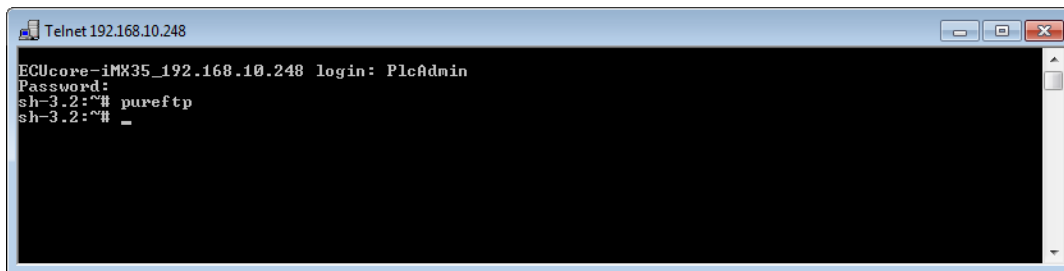


Figure 11: Starting the FTP server

**Advice:** By entering command "*pureftp*" in the start script "*/home/etc/autostart*", the FTP server may be called automatically upon boot of the ECUcore-iMX35 (see section 5.2.2).

"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-iMX35 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)

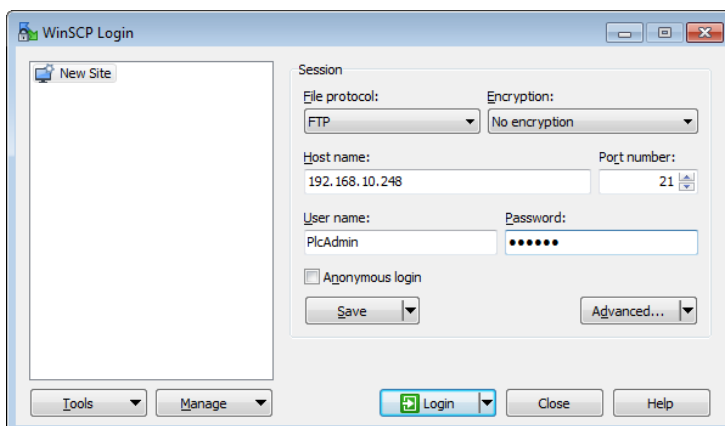


Figure 12: Login settings for WinSCP

After using pushbutton "Login", the FTP client logs in to the ECUcore-iMX35 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-iMX35.

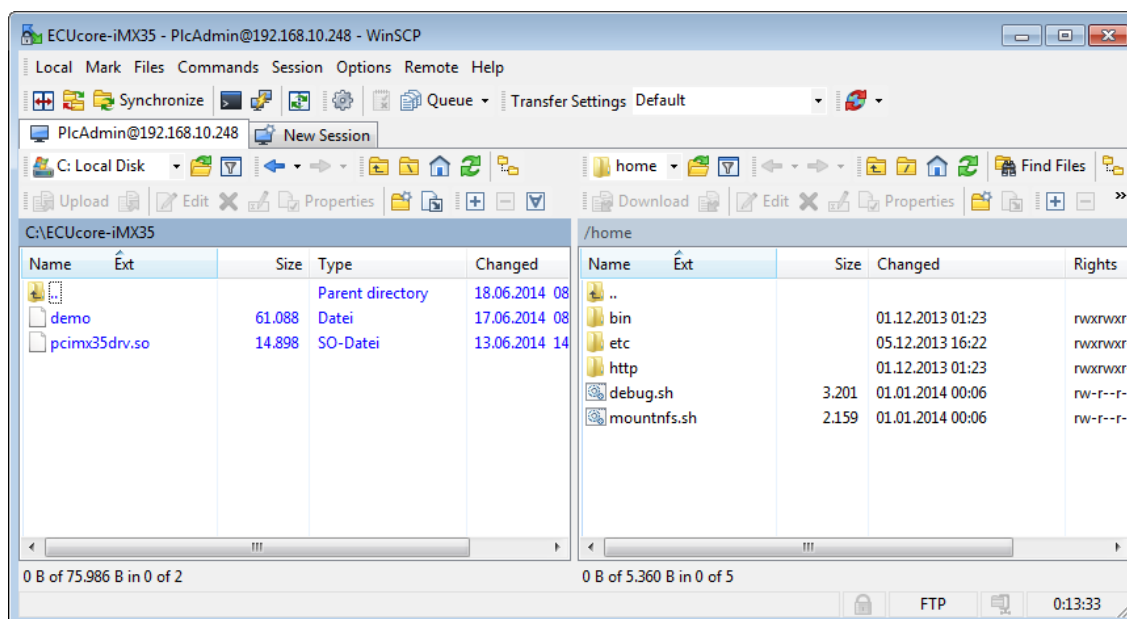


Figure 13: FTP client for Windows "WinSCP"

After successful login, configuration files on the ECUcore-iMX35 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-iMX35, e.g. for data backups of the ECUcore-iMX35 or to transfer installation files for firmware updates (select transfer mode "Binary").

## 5.6 Predefined user accounts

All user accounts in Table 6 are predefined upon delivery of the ECUcore-iMX35. Those allow for a login to the command shell (serial RS232 connection or Telnet) and at the FTP server of the ECUcore-iMX35.

Table 6: Predefined user accounts of the ECUcore-iMX35

User name	Password	Remark
PlcAdmin	Plc123	Predefined user account for the administration of the ECUcore-iMX35 (configuration, user administration, software updates etc.)
root	Sys123	Main user account ("root") of the ECUcore-iMX35



## 5.7 Adding and deleting user accounts

Adding and deleting user accounts requires the login to the ECUcore-iMX35 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-iMX35, it does not make sense to open a directory for every user. Hence, parameter *"-H"* disables the opening 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-iMX35, 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-iMX35 for user *"admin2"*.

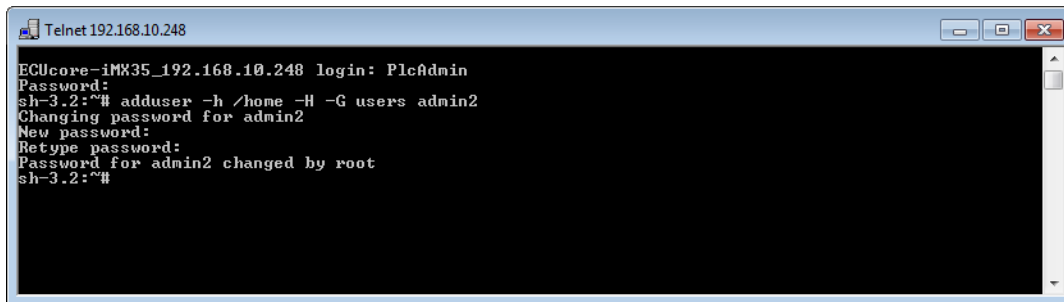


Figure 14: Adding a new user account

To **delete** an existing user account from the ECUcore-iMX35, Linux command *"deluser"* plus the respective user name must be used:

```
deluser <username>
```

## 5.8 How to change the password for user accounts

Changing the password for user accounts requires login to the ECUcore-iMX35 as explained in section 5.5.1.

To change the password for an existing user account on the ECUcore-iMX35, Linux command *"passwd"* plus the respective user name must be entered:

```
passwd <username>
```

Figure 15 exemplifies the password change for user *"PlcAdmin"*.

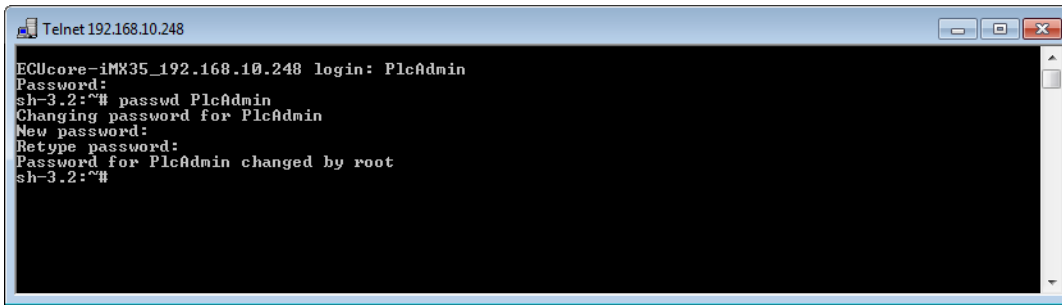


Figure 15: Changing the password for a user account

## 5.9 Setting the system time

Setting the system time requires login to the ECUcore-iMX35 as described in section 5.5.1.

There are two steps for setting the system time of the ECUcore-iMX35. 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-iMX35.

Linux command `"date"` is structured as follows:

```
date [options] [YYYY.]MM.DD-hh:mm[:ss]
```

### Example:

```

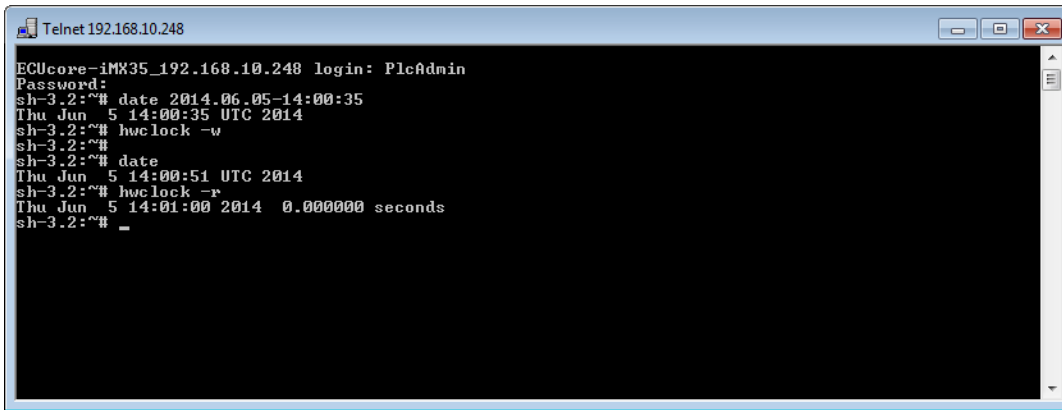
date    2014.06.05-14:00:35
      | | | | | | |
      | | | | | +--- Second
      | | | | +----- Minute
      | | | +----- Hour
      | | +----- Day
      | +----- Month
      +----- Year

```

To set the system time of the ECUcore-iMX35 to 2014/06/05 and 14:00:35 (as shown in the example above), the following commands are necessary:

```
date 2014.06.05-14:00:35
hwclock -w
```

The current system time is displayed by entering Linux command `"date"` (without parameter). The Linux command `"hwclock -r"` can be used to recall current values from the RTC. By using `"hwclock -s"`, the current values of the RTC are taken over as system time for Linux (synchronizing the kernel with the RTC). Figure 16 exemplifies setting and displaying the system time.



```

Telnet 192.168.10.248
ECUcore-iMX35_192.168.10.248 login: PlcAdmin
Password:
sh-3.2:~# date 2014.06.05-14:00:35
Thu Jun  5 14:00:35 UTC 2014
sh-3.2:~# hwclock -w
sh-3.2:~#
sh-3.2:~# date
Thu Jun  5 14:00:51 UTC 2014
sh-3.2:~# hwclock -r
Thu Jun  5 14:01:00 2014  0.000000 seconds
sh-3.2:~# _

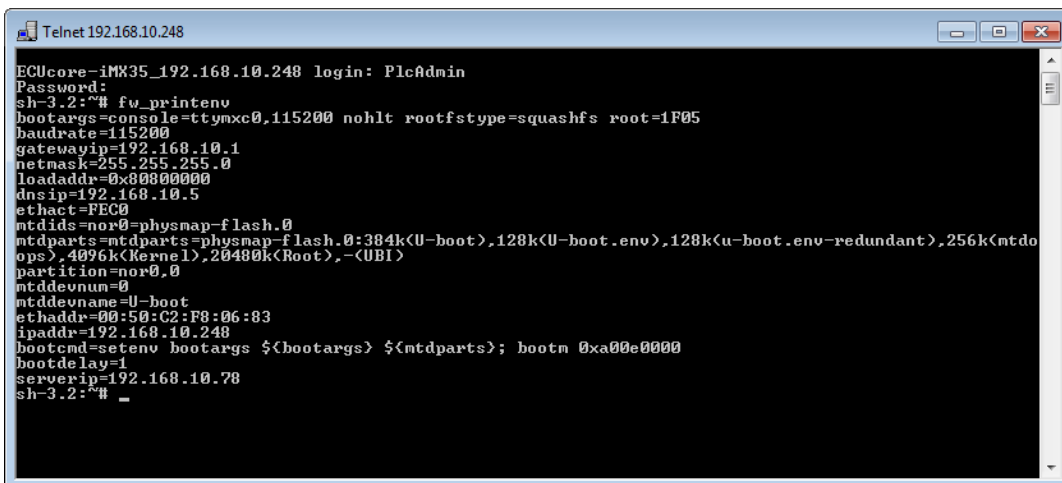
```

Figure 16: Setting and displaying the system time

Upon start of the ECUcore-iMX35, date and time are taken over from the RTC and set as current system time of the module. Therefore, Linux command "hwclock -s" is necessary which is included in start script `"/etc/init.d/hwclock"`.

## 5.10 Readout and displaying "U-Boot" configuration data

Command `"fw_printenv"` under Linux enables the access to configuration data of the bootloader "U-Boot". For example, it is used in the start script `"/etc/rc.d/S05network"` to re-use the Ethernet configuration set for the ECUcore-iMX35 within the "U-Boot" (see section 5.4) to parameterize interface "ETH0" under Linux. The same is essential for taking over the server address (defined in the "U-Boot") into the Shell scripts `"/home/mountnfs.sh"` and `"/home/debug.sh"`.



```

Telnet 192.168.10.248
ECUcore-iMX35_192.168.10.248 login: PlcAdmin
Password:
sh-3.2:~# fw_printenv
bootargs=console=ttymxc0,115200 nohlt rootfstype=squashfs root=1F05
baudrate=115200
gatewayip=192.168.10.1
netmask=255.255.255.0
loadaddr=0x80800000
dnsip=192.168.10.5
ethact=FECC
mtdids=nor0=physmap-flash.0
mtdparts=mtdparts=physmap-flash.0:384k(U-boot),128k(U-boot.env),128k(U-boot.env-redundant),256k(mtdops),4096k(Kernel),20480k(Root),-(UBI)
partition=nor0.0
mtddevnum=0
mtddevname=U-boot
ethaddr=00:50:C2:F8:06:83
ipaddr=192.168.10.248
bootcmd=setenv bootargs ${bootargs} ${mtdparts}; bootm 0xa00e0000
bootdelay=1
serverip=192.168.10.78
sh-3.2:~# _

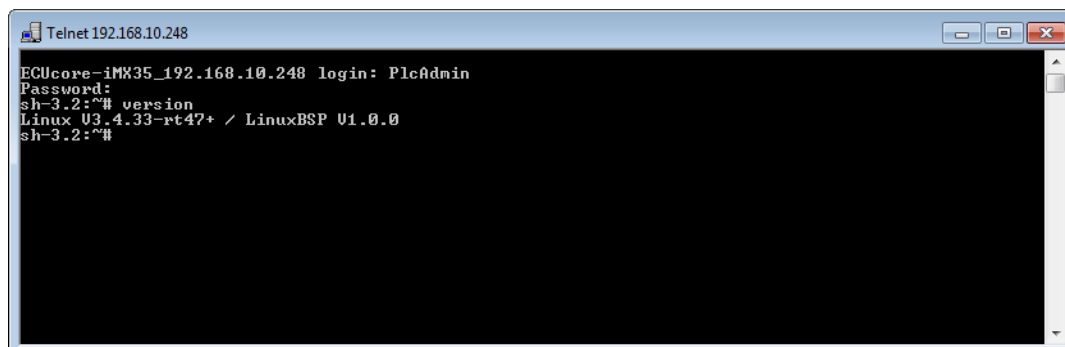
```

Figure 17: Displaying "U-Boot" configuration data under Linux using "fw\_printenv"

By calling `"fw_printenv"` without indicating parameters, all "U-Boot" configuration data is shown (see Figure 17). A direct query of specific configuration information is possible by calling `"fw_printenv <paramname>"` (e.g. `"fw_printenv ipaddr"`). The above mentioned scripts illustrate the usage of `"fw_printenv"`, demonstrate the assignment of requested information to environment variables and show the analysis of information in the Shell script.

## 5.11 Showing the installed Linux-Version

The Linux-Version installed on the ECUcore-iMX35 is shown by calling command "version" (see Figure 18).



```

Telnet 192.168.10.248
ECUcore-iMX35_192.168.10.248 login: PlcAdmin
Password:
sh-3.2:~# version
Linux U3.4.33-rt47+ / LinuxBSP U1.0.0
sh-3.2:~#

```

Figure 18: Showing the installed Linux-Version

## 5.12 File system of the ECUcore-iMX35

The Embedded Linux which is pre-installed on the ECUcore-iMX35 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 Linux-Image for the ECUcore-iMX35. 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.

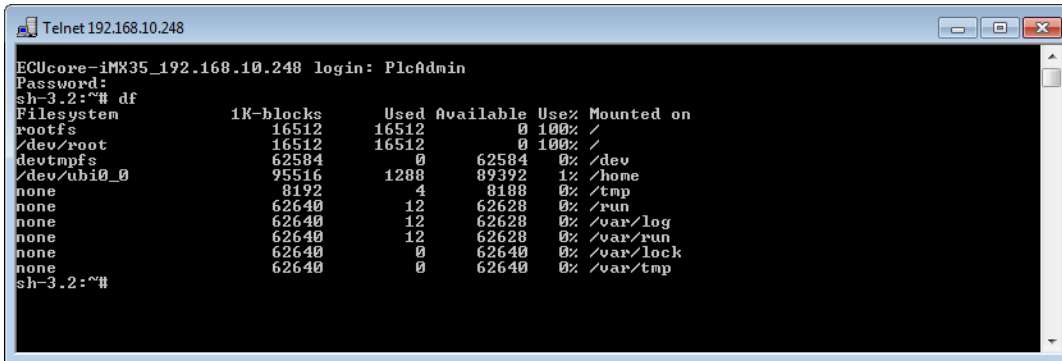
Table 7 lists up writable paths of the file system during runtime. Path **"/home"** comprises a flash disk that provides part of the on-board flash memory of the ECUcore-iMX35 as file system. This path is used to store all files modifiable and updatable by the user, e.g. configuration files, user programs that have been loaded onto the module. In general, one of the two RAM disk directories **"/var/log"** or **"/tmp"** should be used for tests during the development phase – if not anyway some parts of the Linux development system are integrated via NFS (see section 7.5.1).

Table 7: File system configuration of the ECUcore-iMX35

Path	Size	Description
/home	95516 kByte	Flash disk to permanently store files modifiable and updatable by the user (e.g. user software and configuration files); it is not overwritten during the update of the Linux kernel and Root file system; data preservation in case of power breakdown
/tmp	8192 kByte	RAM disk, suitable as intermediate buffer for FTP downloads, but no data preservation in case of power breakdown
/var/log	62640 kByte	RAM disk which is used by the system to store temporary files, no data preservation in case of power breakdown
/mnt		Target for integrating remote directories of other systems via NFS, compare section 7.5.1

**Advice:** Sizes of the file system directories **"/tmp"** and **"/var/log"** may be modified through adjustment in the configuration file **"/etc/fstab"**.

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



```

Telnet 192.168.10.248
ECUcore-iMX35_192.168.10.248 login: PlcAdmin
Password:
sh-3.2:~# df
Filesystem            1K-blocks      Used Available Use% Mounted on
rootfs                 16512          0    16512    0% /
/dev/root              16512          0    16512    0% /
devtmpfs               62584          0    62584    0% /dev
/dev/ubi0_0            95516        1288    89392    1% /home
none                   8192           4     8188    0% /tmp
none                   62640         12    62628    0% /run
none                   62640         12    62628    0% /var/log
none                   62640         12    62628    0% /var/run
none                   62640          0    62640    0% /var/lock
none                   62640          0    62640    0% /var/tmp
sh-3.2:~#

```

Figure 19: Display of information about the file system

Particular information about the system login and handling the Linux command shell of the ECUcore-iMX35 is given attention in section 5.5.1.

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

A Flash disk is bound to the directory "/home" ("mounted"). It provides part of the on-board Flash memory of the ECUcore-iMX35 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-iMX35 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
+- hotplug.sh	Scripts for response of connecting/disconnecting USB Sticks
+- diskadded.sh	(see section 9.1)
+- diskremoved.sh	
+- bin	
+- pcimx35drv.ko	Kernelspace-Module of the I/O Driver (see section 7.3.1)
+- iodrvdemo	Userspace program to test the hardware connection (see section 8.2)
+- 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-iMX35 (see section 7.5.1)
+- debug.sh	Script to simplify the start of Demo programs under the control of the Debug server (see section 7.6.2.3)

If necessary, the delivery status of all files in the directory "/home" may be restored by executing the setup script "setup-ecucore-imx35.sh". The setup script "setup-ecucore-imx35.sh" is located in the directory "SetupFlashdisk\_ECUcore-iMX35" of the DVD "SO-1121". Section 7.5 describes several possibilities to transfer this file onto the ECUcore-iMX35.

## 5.14 Using the HTTP server

HTTP server *"lighttpd"* is installed on the ECUcore-iMX35. Hence, the access to the module is possible via any WEB-Browser (e.g. Microsoft Internet Explorer, Mozilla Firefox etc.). For the configuration of the server the file *"/home/http/lighttpd.conf"* is used. Upon starting the server, this file must be identified via command line parameter *"-f"*. The delivery status of the ECUcore-iMX35 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 *"lighttpd"* must be started manually by indicating the Demo directory. Therefore, the user must first be logged in to the command shell of the ECUcore-iMX35 as explained in section 5.5.1. Afterwards, the following command is entered in the Telnet or Terminal window:

```
lighttpd -f /home/http/lighttpd.conf
```

Figure 20 exemplifies starting the HTTP server *"lighttpd"* for the Demo configuration included in the delivery status of the ECUcore-iMX35.

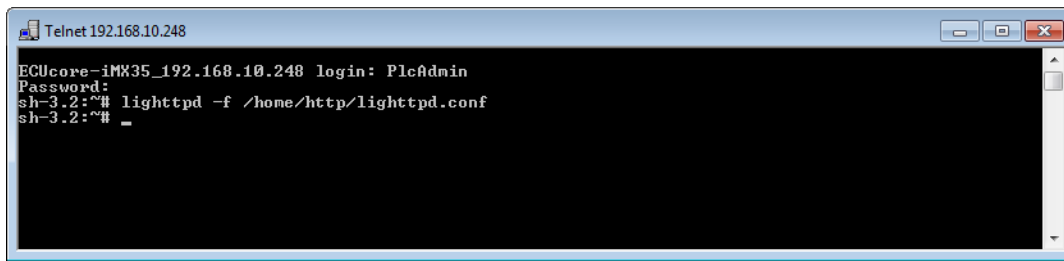


Figure 20: Starting the HTTP server *"lighttpd"*

To call the pages provided by the HTTP server, prefix *"http://"* plus the IP address of the ECUcore-iMX35 as set in section 5.4 (e.g. *"http://192.168.10.248"*) must be entered in the address bar of the WEB-Browser. Figure 21 shows HTML pages for the ECUcore-iMX35 in the WEB-Browser.



Figure 21: Display of HTML pages for the ECUcore-iMX35 in the WEB-Browser

**Advice:** By entering the start call of the HTTP server (e.g. "`lighttpd -f /home/http/lighttpd.conf`") into the start script "`/home/etc/autostart`", calling the HTTP server upon boot of the ECUcore-iMX35 may be automated (see section 5.2.2).

## 5.15 HMI Components

### 5.15.1 Supported HMI Devices

The ECUcore-iMX35 has been developed especially for realizing user-specific HMI (**H**uman **M**achine **I**nterface) applications. Thereby, the module supports HMI typical input and output devices, as described in the following.

## **Display**

The ECUcore-iMX35 contains an integrated LCD Controller with support for displays with maximum 800x600 pixel resolution at 24 bit color depth. The Linux-Image installed on the ECUcore-iMX35 for delivering is supporting the display with integrated touch screen contained in the Development Kit ECUcore-iMX35:

Manufacturer:	EDT
Manufacturer Article No.:	ETQ570G2DH6
SYSTEC Article-No.:	191010
Size:	5,7"
Resolution:	QVGA (320x240 Pixel)
Color Depth:	24 Bit
Color-Coding:	555 LE (RGB, 5Bit / Color, Little Endian)

Alternatively, connection of another display with maximum SVGA-resolution (800x600 Pixel) to the ECUcore-iMX35 is possible. Depending on the used Display-Type, adaption of the Linux-Image might be necessary. Please contact our Support Department in case of any further questions:

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

## **Input Devices**

The ECUcore-iMX35 supports the input devices list in Table 8. The corresponding device drivers are aligned to the periphery contained in the Development Kit ECUcore-iMX35 but can be adjusted if needed. The necessary sources are included in the LinuxBSP of the ECUcore-iMX35 (Software package **SO-1121**, "VMware-Image of the Linux-Development System for the ECUcore-iMX35").

*Table 8: Input Devices des ECUcore-iMX35*

<b>Device File</b>	<b>Input Device</b>	<b>Input Events</b>
/dev/input/event0	Matrixtastatur	Event Type: 1 (Key) Event Codes: F1 ... F16
/dev/input/event1	Touchscreen	Event Type: 3 (Absolute) Event Codes: X, Y, Pressure
/dev/input/event2	Scrollwheel X-Achse	Event Type: 2 (Relative) Event Values: 1 (Clockwise) -1 (CounterClockwise)
/dev/input/event3	Scrollwheel Push-Button	Event Type: 1 (Key) Event Codes: ENTER

Figure 22 shows the assignment of keyboard events for the Matrix-Keyboard as well as the Scrollwheel and its Push-Button.



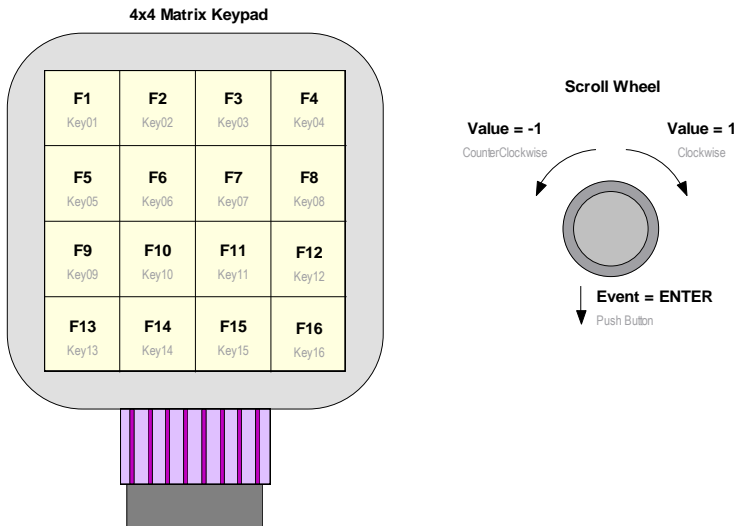


Figure 22: Assignment of Membrane Keypad and Scrollwheel

For the diagnosis of input devices, the Linux command "evtest" is suited. It shows detailed information to the single input devices as well as the events generated by them:

```
evtest /dev/input/event<X>
```

Figure 23 demonstrates the command "evtest" on the example of the Matrix-Keyboard (/dev/input/event0) and shows the outputs generated when pressing the keys.

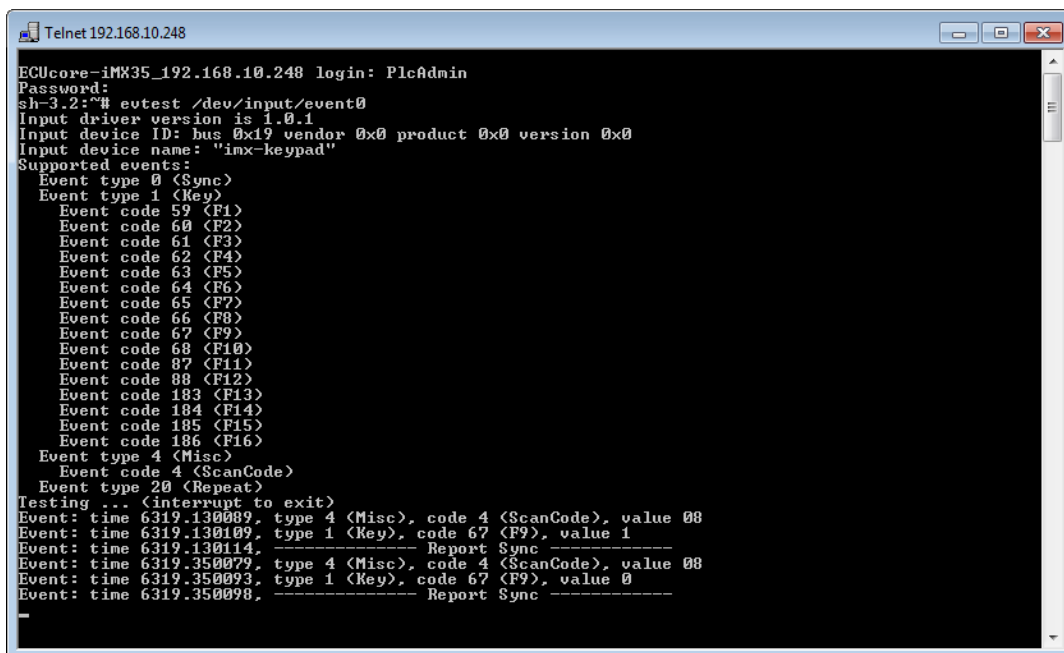


Figure 23: Diagnosis of Input Devices by means of "evtest"

### 5.15.2 Connection of Scrollwheel and Matrix Keyboard

The module connectors "MATRIX\_KEYPAD\_IO0 ... MATRIX\_KEYPAD\_IO7" are intended for a connection of a 4x4 matrix keyboard. Furthermore, the module connectors "IO\_PE19", "IO\_PB10" and

"IO\_PD11" allow for the connection of a Scrollwheel with Push-Button. The schematic of the Development Board serves as reference design. Figure 24 shows the connection of the foil keyboard, contained in the Development Kit ECUcore-iMX35, to the development board.

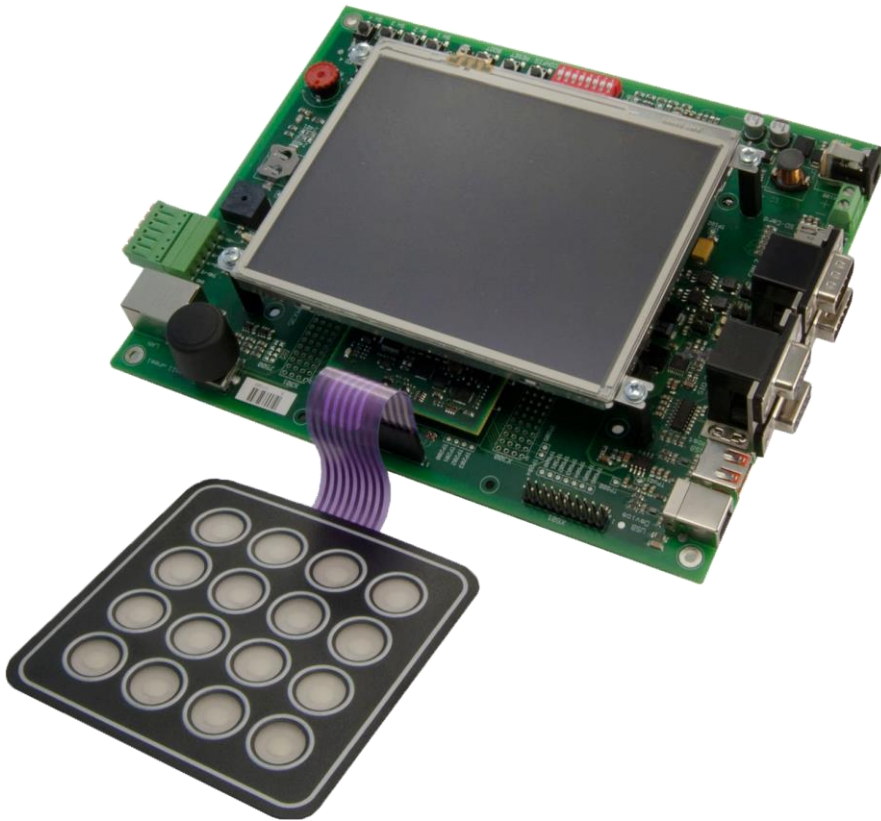


Figure 24: Connection of the Foil Keyboard to the Development Board

### 5.15.3 Setting Display Brightness

Control of display brightness on the ECUcore-iMX35 occurs via read- and write accesses on items of the display driver in the file system. All items of the display driver are contained in folder:

```
/sys/devices/platform/mx3_sdc_fb/backlight/mx3fb-bl/
```

Here, the following driver items are relevant:

<code>bl_power:</code>	1 = Display is pushed with maximum brightness, independent from the value " <i>brightness</i> " 0 = Display brightness is defined through the value entered in " <i>brightness</i> "
<code>brightness:</code>	Brightness value (1=min ... 255=max), only operative for " <i>bl_power</i> = 0"
<code>actual_brightness:</code>	currently effective brightness value <i>bl_power</i> := 0 -> copy of the value " <i>brightness</i> " <i>bl_power</i> := 1 -> constant at 0
<code>max_brightness:</code>	Constant for maximum brightness value (= 255)

### 5.15.4 Calibration of the Touchscreen

The PLCcore-iMX35 has no on-board touch controller. Hence, an external touch controller is necessary to use resistive Touchscreens. Touchscreen and touch controller have to be adjusted – that means calibrated – to another before its first use. Without a calibration, the touchscreen works extremely imprecise which normally makes a correct operation impossible.

The calibration of the touchscreen occurs interactively, by the operators click on the markings (“Reticles”) given on the display. The calibration program needed for it is started from the command line, which requires login to the ECUcore-iMX35 as described in section 5.5.1. After that, the following command has to be entered in the Telnet- or Terminal-window:

```
ts_calibrate
```

In the course of the calibration sequence, 5 markings (“Reticles”, in each corner and in the middle) are shown one after another on the display, which are to click by the user. The more exact the shown markings are clicked, the higher the achievable accuracy during the later operation of the Touchscreen. It is therefore recommended to use a touchpen or stylus during calibration as it is used for Handhelds, PDAs or drawing tablets.

After finishing calibration, the calibration data are stored in file *"/home/etc/pointercal"*. In case this file gets lost, e.g. through reformatting of the flash-disk, the calibration has to be carried out again.

## 5.16 The Use of Qt on the ECUcore-iMX35

Qt is a platform-independent C++-class library for programming graphical user interfaces. On the ECUcore-iMX35, Qt can be used in the form of Qt/Embedded. As Qt requests an X-Window-System, Qt/Embedded directly operates on the Linux-Frame buffer and is therefore the ideal solution for embedded systems like the ECUcore-iMX35.

### 5.16.1 Overview of the Qt-Components for the ECUcore-iMX35

Together with the Linux-Image, the Qt-libraries and demo programs are generated for the ECUcore-iMX35 (see section 7.7). After completion of the build process, the needed software components are located in directory

```
/projects/ECUcore-iMX35/LinuxBSP/ECUcore_iMX35/platform/packages
```

Inside of the VMware-Images for the Linux Development System. The directory *"Qt-Addons"* on the DVD *"SO-1121"* contains libraries that have already been created. Table 9 lists the available Qt-Components for the ECUcore-iMX35.

*Table 9: Qt-Components for the ECUcore-iMX35*

File	Category	Meaning
qt4_4.8.4_armel.ipk	Mandatory	Qt Runtime libraries for execution on the ECUcore-iMX35 (for installation instruction see section 0)
qtdemo_1.0.0_armel.ipk	Optional	Qt Demo program for execution on the ECUcore-iMX35 (see hint below, for installation instruction refer to section 0, program call in section 5.16.3)
demogui_1.0_armel.ipk	Optional	Qt Demo program for process visualization on the PLCcore-iMX35 (uses Shared Process Image for data

		exchange with the PLC)
profile.local	Mandatory	Script for setting the environment variable needed for runtime (see section 5.16.2)

**Note:** Through execution of the sample program *"qtdemo"* it can be checked easily whether the requested environment variables are set correctly. The execution of the program is described in section 5.16.3.

## 5.16.2 Qt-Environment Variables

Qt requests different environment variables for runtime to determine for example the path to the runtime libraries, font files and the available input devices. On the ECUcore-iMX35, those environment variables are to be set as follows:

```
QWS_KEYBOARD="LinuxInput:/dev/input/event0 LinuxInput:/dev/input/event1"
QWS_MOUSE_PROTO=Tslib
POINTERCAL_FILE=${TSLIB_CALIBFILE}
QT_QWS_FONTDIR=/home/packages/usr/lib/fonts/
LD_LIBRARY_PATH=/home/packages/usr/lib/:${LD_LIBRARY_PATH}
```

The setting of the environment variables typically occurs in the system configuration file *"/home/etc/profile.local"*. A version of *"profile.local"* adapted for the ECUcore-iMX35 is located on the DVD "SO-1121" in directory *"Qt-Addons"*. This file can be copied directly via FTP to directory *"/home/etc"* of the ECUcore-iMX35. As the execution right of a file is generally deleted during a FTP-Transfer (*"eXecutable"*-Flag in the file attributes), it has to be set again on the ECUcore-iMX35 through command *"chmod"*. Details on the FTP-Transfer as well as command *"chmod"* are described in section 5.5.2.

## 5.16.3 Starting Qt-Programs

The execution of Qt-programs on the ECUcore-iMX35 requires the correct setting of needed environment variables (see sections 0 and 5.16.2).

As Qt/Embedded does not use an X-Window System but is directly accessing the Linux-Frame buffer, the Qt/Embedded-application has to function as window-server itself. The Qt/Embedded-Framework however supports the execution of an application in the server as well as in the client-mode. To set a Qt/Embedded-application explicitly in the server-mode, parameter *"-qws"* has to be entered when the application is started.

In order to start the sample program *"qtdemo"* (see section 5.16) on the ECUcore-iMX35, the following command line is needed:

```
/usr/bin/qtdemo -qws
```



Figure 25: Output of "qtdemo" on the ECUcore-iMX35

Figure 25 shows the display output of "qtdemo" generated on the ECUcore-iMX35. By clicking the button "Exit" on the Touchscreen, the program is closed.

## 5.17 Updating the Linux-Image

Updating the Linux-Image takes place via TFTP (Trivial FTP) within the Linux bootloader "U-Boot". 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 Linux-Image from a Windows computer, Freeware "TFTPD32" could be used for example (see section 5.1). The Windows program only exists 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-iMX35"). This directory must contain the Linux-Image for the ECUcore-iMX35 ("root.sum.jffs2").

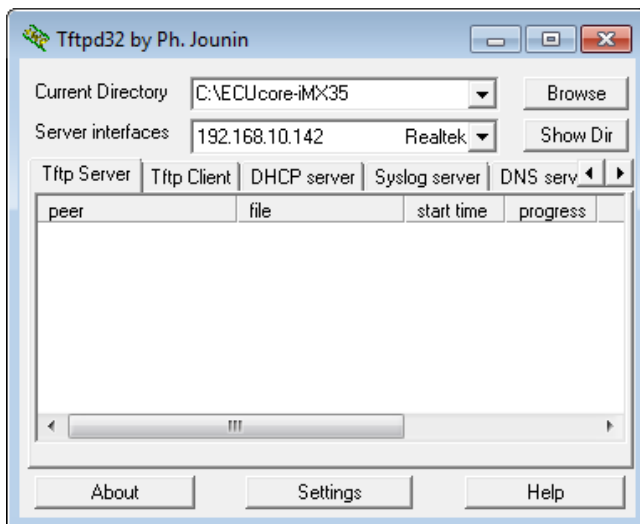


Figure 26: TFTP server for Windows "TFTPD32"

**One requirement** for the TFTP download of the Linux-Image is a **completed Ethernet configuration** of the ECUcore-iMX35 **according to section 5.4**. In addition to the Ethernet connection, a serial connection to the ECUcore-iMX35 is necessary to update of the Linux-Image. Therefore, all setups for the terminal program as described in section 5.2.1 are valid (115200 Baud, 8 Databit, 1 Stopbit, no parity, no flow control).

An update of the Linux-Image on the ECUcore-iMX35 is only possible if Linux is not yet running. Hence, prior to updating the Linux-Image the Linux Autostart must be prevented and it must be switched to the "U-Boot" command prompt instead. All necessary steps are covered in section 5.2.1.

After Reset (e.g. pushbutton S601 on the Development Board) the "U-Boot" command prompt answers. To update the Linux-Image all commands explained in the following table must be entered in the same sequence as provided below:

Table 10: Command sequence to update the Linux-Image on the ECUcore-iMX35

Command	Meaning
<code>setenv serverip &lt;host_ip_addr&gt;</code>	Setting the IP address of the TFTP server. If "TFTPD32" is used, the address is shown in field "Server Interface" on the PC.
<code>mtdparts default</code>	Use the standard partition table defined on the ECUcore-iMX35
<code>tftp linuximage</code>	Downloading the Linux-Image from the Development PC onto the ECUcore-iMX35
<code>erase nor0,4</code>	Erase the Flash area, needed by Linux-Image
<code>cp.b \${fileaddr} 0xa00e0000 \${filesize}</code>	Saving the Linux-Image in the Flash of the ECUcore-iMX35
<code>tftp root.squashfs</code>	Downloading the Root File System from the Development PC onto the ECUcore-iMX35
<code>erase nor0,5</code>	Erase the Flash area, needed by Root File System
<code>cp.b \${fileaddr} 0xa04e0000 \${filesize}</code>	Saving the Root File System in the Flash of the ECUcore-iMX35

```

Tera Term - COM1 VT
File Edit Setup Control Window Help

U-Boot 2010.09->v1.0.0-00024-gc8f8cbf (Jun 04 2014 - 15:39:15)

CPU: Freescale i.MX35 at 532 MHz
Board: ECUcore-iMX35 (P08)
PCSR: 00000800
DRAM: 128 MiB
Flash: 128 MiB
Int: serial
Out: serial
Err: serial
i2c: i2c0 clock: 532KHz
ipg clock : 665000000Hz
ipg_per clock : 665000000Hz
uart clock : 100000000Hz
Net: FEC0
Hit any key to stop autoboot: 0
U-Boot> setenv serverip 192.168.10.142
U-Boot> tftp linuximg2
Using FEC0 device
TFTP from server 192.168.10.142; our IP address is 192.168.10.248
Filename 'linuximg2'
Load address: 0xa0400000
Loading: =====
done
Bytes transferred = 26338 (26338 hex)
U-Boot> erase nor0.4
Erase Flash Partition nor0.4, bank 0, 0xa0400000 - 0xa044fff
..... done
Erased 32 sectors
U-Boot> cp.b $(fileaddr) 0xa040000 $(filesize)
Copy to Flash... 9.....8.....7.....6.....5.....4.....3.....2.....1.....done
U-Boot> tftp root.squashfs
Using FEC0 device
TFTP from server 192.168.10.142; our IP address is 192.168.10.248
Filename 'root.squashfs'
Load address: 0xa0400000
Loading: =====
done
Bytes transferred = 16879616 (1019000 hex)
U-Boot> erase nor0.5
Erase Flash Partition nor0.5, bank 0, 0xa0400000 - 0xa044fff
..... done
Erased 160 sectors
U-Boot> cp.b $(fileaddr) 0xa040000 $(filesize)
Copy to Flash... 9.....8.....7.....6.....5.....4.....3.....2.....1.....done
U-Boot>

```

Figure 27: Downloading the Linux-Image onto the ECUcore-iMX35

**After finishing the configuration, all preconditions for the Linux Autostart must be re-established according to section 5.2.1.**

After Reset (e.g. pushbutton S601 on the Development Board), the ECUcore-iMX35 starts using the actual Linux-Image.

**Advice:** After the configuration is finished, the serial connection between the computer and the ECUcore-iMX35 is no longer necessary.

## 5.18 Updating the bootloader "U-Boot"

Updating the "U-Boot" bootloader is basically possible, but it involves the risk that the ECUcore-iMX35 will not boot anymore if an update failed or carried defective software. For this reason, all flash sectors in which the "U-Boot" is integrated are protected from accidental deletion.

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

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

## 6 VMware-Image with Linux Development System

### 6.1 Overview

The ECUcore-iMX35 is delivered with a preinstalled Embedded Linux. 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-iMX35. 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 ARM11-Processors
- Linux-Sourcecode for the ECUcore-iMX35 (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-iMX35 and the development computer)
- TFTP server (enables downloading the Linux-Image for the ECUcore-iMX35 from the development computer)
- NFS server (enables the integration of the development computer into the local file system of the ECUcore-iMX35)

### 6.2 Installing the Linux VMware-Image

The Development Kit ECUcore-iMX35 contains the DVD "SO-1121" which includes the VMware-Image for the Linux development system of the ECUcore-iMX35 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-iMX35 could possibly be defective.

The VMware-Image is included on the DVD as self-extracting archive "**SO-1121.exe**". If "SO-1121.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-ECUcore-iMX35.vmx*" in the program window of the Player by using the "Open" symbols (see Figure 28).





Figure 28: 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 29 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.

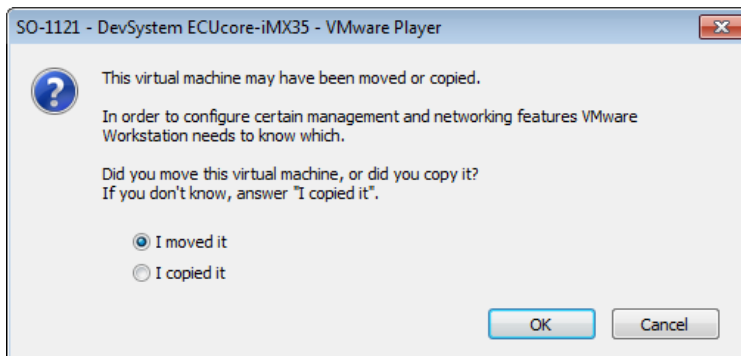


Figure 29: VMware selection dialog to generate or remain the MAC address

Figure 30 shows the desktop of a Linux development system after starting the Linux-Image in the "VMware Player".

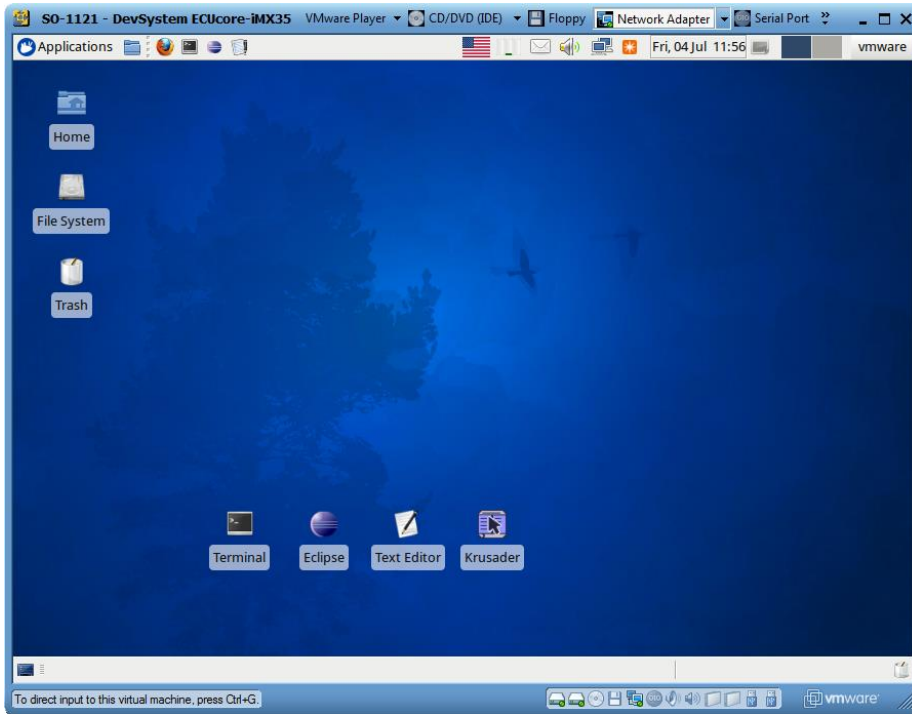


Figure 30: Desktop of the Linux development system

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

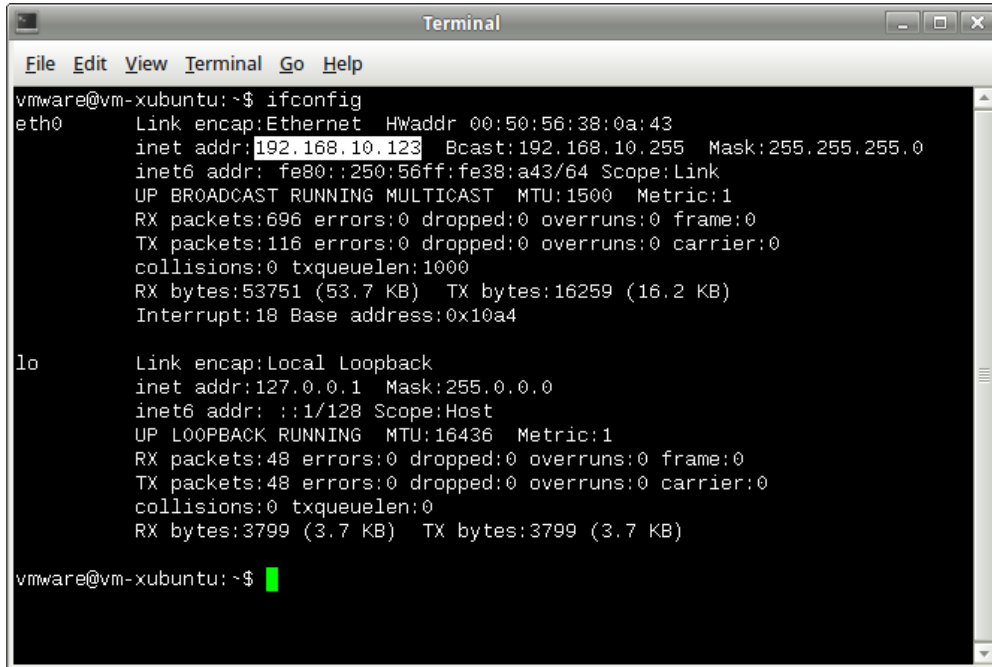
Table 11 lists up all predefined user accounts for logging in to the Linux development system.

Table 11: 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 31*).



```
vmware@vm-xubuntu: ~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:38:0a:43
          inet addr:192.168.10.123  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe38:a43/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:696 errors:0 dropped:0 overruns:0 frame:0
          TX packets:116 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:53751 (53.7 KB)  TX bytes:16259 (16.2 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:48 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3799 (3.7 KB)  TX bytes:3799 (3.7 KB)

vmware@vm-xubuntu: ~$
```

Figure 31: 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 11 in section 6.4).

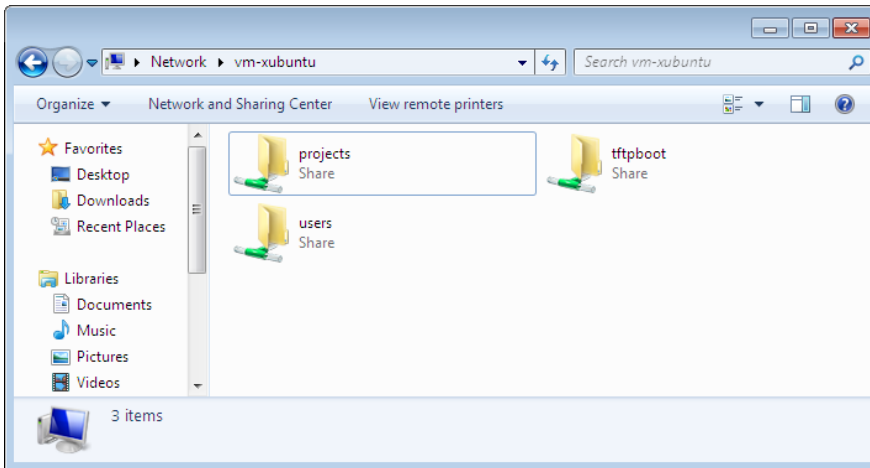


Figure 32: Linux development system in Windows network environment

After double-clicking the symbol "users" (see Figure 32), login is possible as user "vmware" with the corresponding password "vmware" (see Figure 33). In conclusion, path "\\Vm-xubuntu\users\" is accessible.

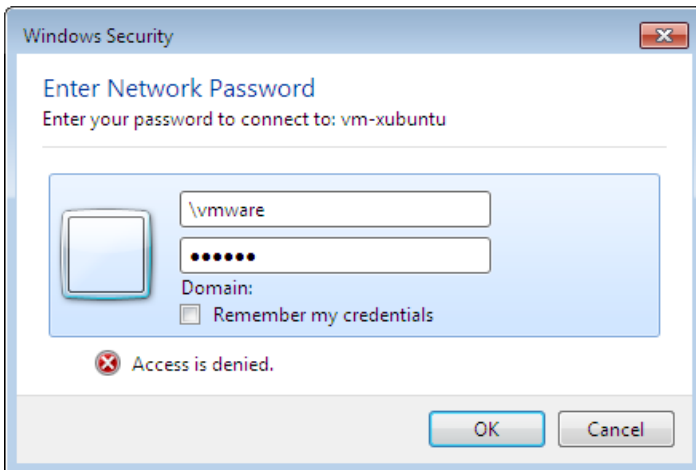


Figure 33: Login to "Vm-xubuntu"

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: \\192.168.10.123\users /user:vmware /persistent:NO
```

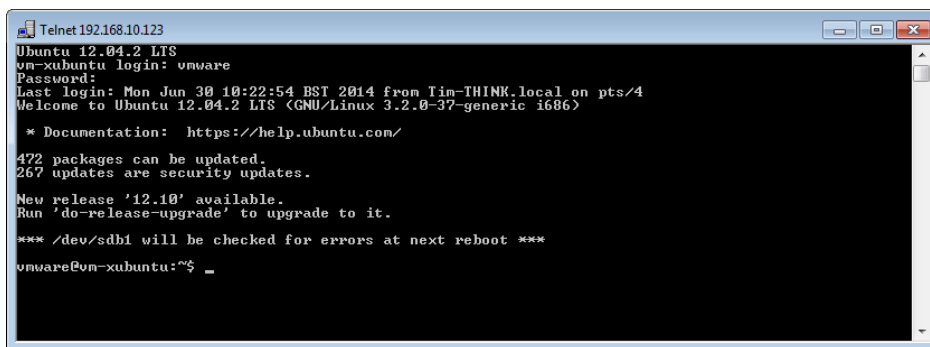
## 6.6.2 Access via Telnet client

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-iMX35 (compare Figure 9 in section 5.5.1), e.g.

```
telnet 192.168.10.123
```

Login as user **"vmware"** with the corresponding password **"vmware"** is possible in the Telnet window (also see Table 11 in section 6.4):



```
Telnet 192.168.10.123
Ubuntu 12.04.2 LTS
vm-xubuntu login: vmware
Password:
Last login: Mon Jun 30 10:22:54 BST 2014 from Tim-THINK.local on pts/4
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

472 packages can be updated.
267 updates are security updates.

New release '12.10' available.
Run 'do-release-upgrade' to upgrade to it.

*** /dev/sdb1 will be checked for errors at next reboot ***
vmware@vm-xubuntu:~$ _
```

Figure 34: Access to the Linux development system via Telnet client

Figure 34 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 35).

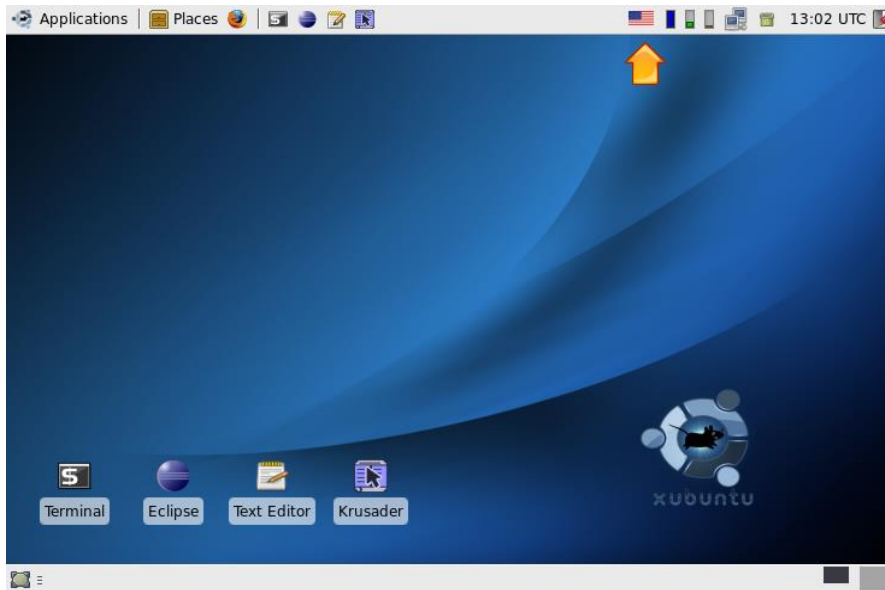


Figure 35: Country symbol for switching keyboard layouts

By clicking with the **right mouse button** on the country symbol in the task menu (see Figure 35) 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 36).

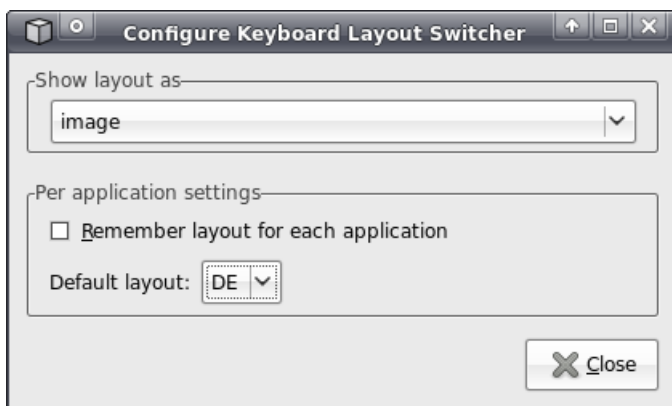


Figure 36: Choosing a permanent keyboard layout

**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 37).

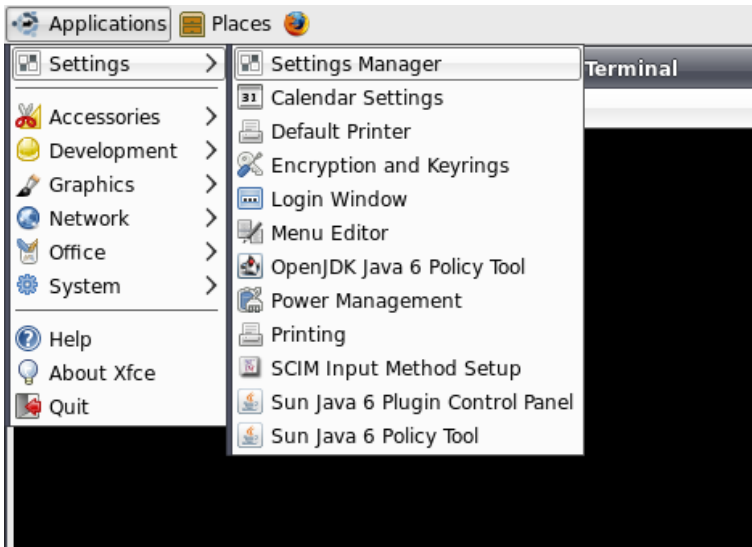


Figure 37: Calling "Xfce Settings Manager" from the start menu

More keyboard layout can be added or deleted within the "Xfce Settings Manager" by using option "Keyboard" and sub-option "Layouts" (see Figure 38).

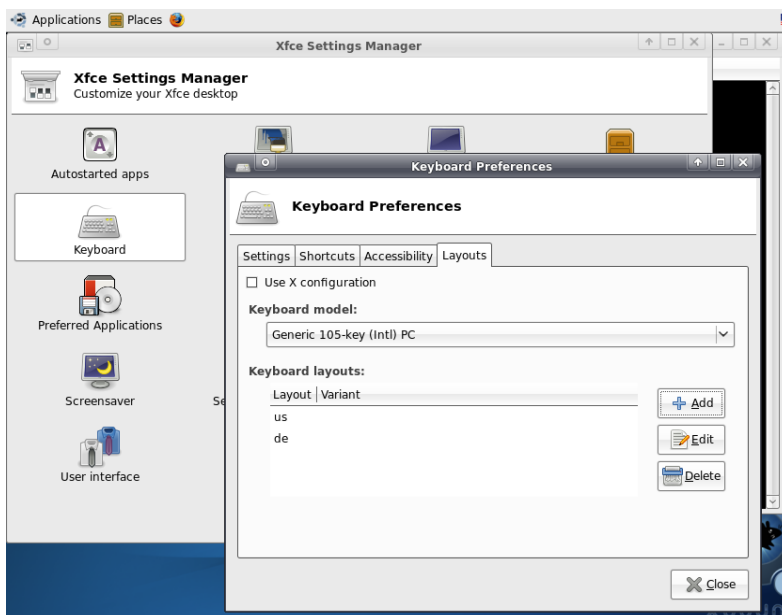


Figure 38: Adding keyboard layouts in the "Xfce Settings Manager"

**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 39). Since changing the time zone is an administrative activity, the dialog must first be released by using pushbutton "*Unlock*" (also see Figure 39). Therefore, the administrator password will be asked analog to the console command "*sudo*". By default, **password "vmware"** must be entered (also compare Table 11 in section 6.4). Afterwards, the capital and the corresponding time zone can be chosen via path "*Time zone*".

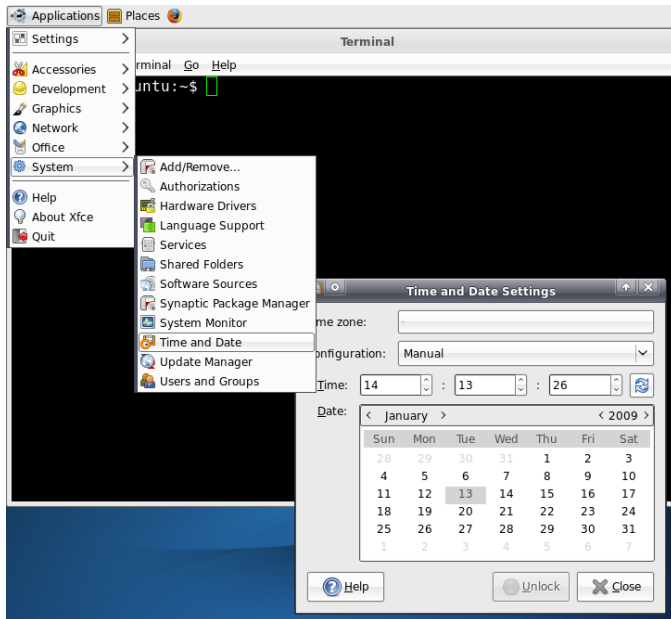


Figure 39: Adjusting the time zone

## 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-1121Xubuntu-ECUcore-iMX35.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-iMX35 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 40) 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 41).



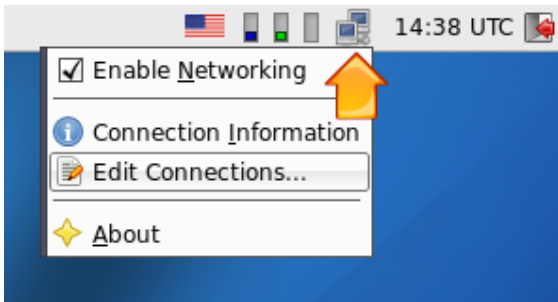


Figure 40: "Network Manager" for the configuration of the Ethernet interface

In Tab sheet "Wired" in the dialog "Network Connections" (see Figure 41) a new network environment can be created by using pushbutton "Add". Afterwards, dialog "Edit Wired Connection" opens up (see Figure 42).

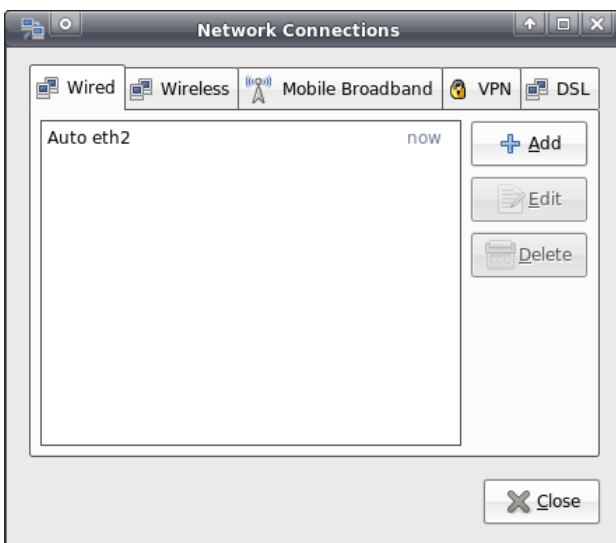


Figure 41: Adding a network connection

In dialog "Edit Wired Connection" (see Figure 42) 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".

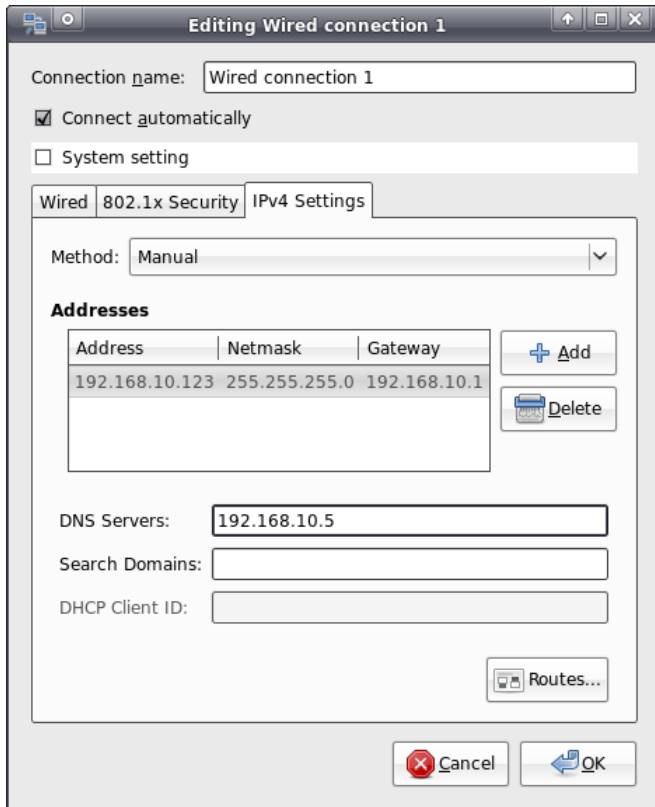


Figure 42: Configuration of the network connection

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 43). Choose the new network connection with the static IP address as active connection.

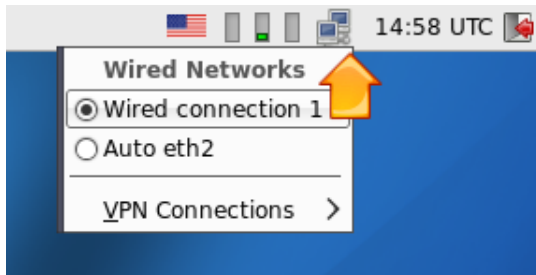


Figure 43: Changing the network configuration in the "Network Manager"

#### 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-iMX35 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 11 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. The VMware Image can be minimized ("shrunk") to its necessary minimum size via tab sheet "*Shrink*".

## 7 Software Development for the ECUcore-iMX35

### 7.1 Software structure of the ECUcore-iMX35

All components necessary to develop software for the ECUcore-iMX35 are filed in path `"/projects"` in the VMware-Image of the Linux development system (or `"\\Vm-xubuntu\users\projects"` in the Windows network environment). Figure 44 illustrates the directory structure.

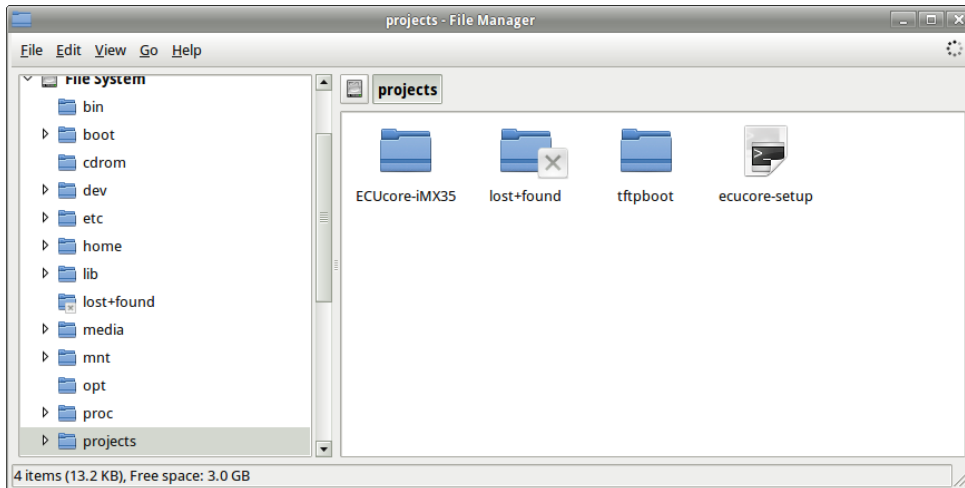


Figure 44: Structure of directory `"/projects"` in the Linux development system

<code>/projects</code>	
-- ECUcore-iMX35	
-- driver	<b>Driver Libraries for the ECUcore-iMX35</b>
-- pcimx35drv	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
-- LinuxBSP	<b>Source code of the Linux-Images for the ECUcore-iMX35</b> (Linux kernel and user programs)
-- linux	Source code of the Linux kernel for the ECUcore-iMX35
-- ptxdist	User programs and Build System for the ECUcore-iMX35
-- toolchain	GCC-Toolchain for the ECUcore-iMX35
-- user	Path to file user projects
-- demo	<b>Reference and I/O Demo project for the ECUcore-iMX35</b>
-- hellocan	<b>Reference and CAN Demo project for the ECUcore-iMX35</b>
-- tftpboot	<b>NFS directory for the integration by the ECUcore-iMX35</b>
-- ecucore-setup	Shell script to set environment variables

Path `"/projects/ECUcore-iMX35/LinuxBSP"` contains all Linux sources for the ECUcore-iMX35.

Directory ***"/projects/ECUcore-iMX35/driver/pcimx35drv"*** includes the source code of the I/O Driver of the ECUcore-iMX35 (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 7.3.1).

Directory ***"/projects/ECUcore-iMX35/driver/candrv"*** contains **Header files and complete libraries of the CAN Driver** for the integration into own user projects (see section 7.4.1).

Directory ***"/projects/ECUcore-iMX35/user/demo"*** provides a Demo program that describes access to in- and outputs of the ECUcore-iMX35 on the one hand (section 7.3.1 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 7.6).

Directory ***"/projects/ECUcore-iMX35/user/helloCAN"*** includes a Demo program that describes access to the CAN interface of the ECUcore-iMX35 on the one hand and serves as template for own projects on the other.

Path ***"/projects/fttpboot"*** is the root directory for the integration of the Linux development system into the local file system of the ECUcore-iMX35 via NFS (see section 7.5.1).

Shell script ***"/projects/ecucore-setup"*** set the required environment parameters that are necessary to execute the build system (see section 7.2). 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.

## 7.2 Makefile and environment variables to create projects

Creating user programs for the ECUcore-iMX35 requires the usage of GNU-Crosscompiler Toolchain for ARM9 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:

```
ARM_IMX35_BASE_PATH=/projects/ECUcore-iMX35
ARM_IMX35_LINUX_BSP_PATH=$ARM_IMX35_BASE_PATH/LinuxBSP
ARM_IMX35_PTXDIST_PROJECT_PATH=$ARM_IMX35_LINUX_BSP_PATH/ECUcore_iMX35
ARM_IMX35_PTXDIST_PLATFORM_PATH=$ARM_IMX35_PTXDIST_PROJECT_PATH/platform
ARM_IMX35_LINUX_KDIR_PATH=$ARM_IMX35_PTXDIST_PLATFORM_PATH/build-target/linux-3.4.33/
ARM_IMX35_CC_PATH=$ARM_IMX35_BASE_PATH/toolchain/arm-2012.03/bin
ARM_IMX35_CC_PREFIX=$ARM_IMX35_CC_PATH/arm-none-linux-gnueabi-
ARM_IMX35_CFLAGS=
ARM_IMX35_GDB_PATH=$ARM_IMX35_CC_PATH

export ARM_IMX35_BASE_PATH
export ARM_IMX35_LINUX_BSP_PATH
export ARM_IMX35_LINUX_KDIR_PATH
export ARM_IMX35_CC_PATH
export ARM_IMX35_CC_PREFIX
export ARM_IMX35_CFLAGS
export ARM_IMX35_GDB_PATH

# Path to target sysroot for compiling/linking applications outside of ptxdist
export TARGET_SYSROOT=$ARM_IMX35_PTXDIST_PLATFORM_PATH/sysroot-target

# add toolchain to PATH
export PATH=$ARM_IMX35_CC_PATH:$PATH

# add ptxdist to PATH
export PATH=$ARM_IMX35_LINUX_BSP_PATH/ptxdist/bin:$PATH
```

The template in path `"/projects/ECUcore-iMX35/user/demo"` should be the initial point to develop own programs (or `"\\vm-xubuntu\users\projects\ECUcore-iMX35\user\demo "` in the Windows network environment). Hereby, variables defined in Makefile (`demo/source/Makefile`) and references to GNU-Crosscompiler Toolchain for ARM9 processors (based on definitions in `"/projects/ecucore-setup"`) are especially relevant.

```
CDEFS          = $(ARM_IMX35_CFLAGS) -D$(DBG_MODE)
CFLAGS         += -O0 -g -Wall -Wno-pointer-sign -Wno-enum-compare $(INCLUDES)
               $(CDEFS)
LDLFLAGS       +=

CROSS          = $(ARM_IMX35_CC_PREFIX)
LD_LIB_PATH   =
CC             = $(CROSS)gcc
STRIP         = $(CROSS)strip
AR            = $(CROSS)ar
```

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

```
@$(CC) $(CFLAGS) -c $(notdir $*.c) -o $*.o
```

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-iMX35 without other intermediate steps (also compare section 7.5.1).

## 7.3 I/O Driver for the ECUcore-iMX35

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

Directory `"/projects/ECUcore-iMX35/driver/pcimx35drv"` of the Linux development system contains the source code of the I/O Driver for the ECUcore-iMX35 (including testing application). Moreover, it contains Header files and complete libraries of the Driver for the integration into own user projects.

Figure 45 illustrates the structure of the I/O Driver for the ECUcore-iMX35. The driver is divided into a Kernel space module (`pcimx35drv.ko`) which is in charge of accesses to the hardware (Port pins) and a User space library (`pcimx35drv.a` as static library or `pcimx35drv.so` as dynamically loadable library). Both components, the Kernel space module and the User space library (static or dynamic) are necessary to accomplish I/O accesses.

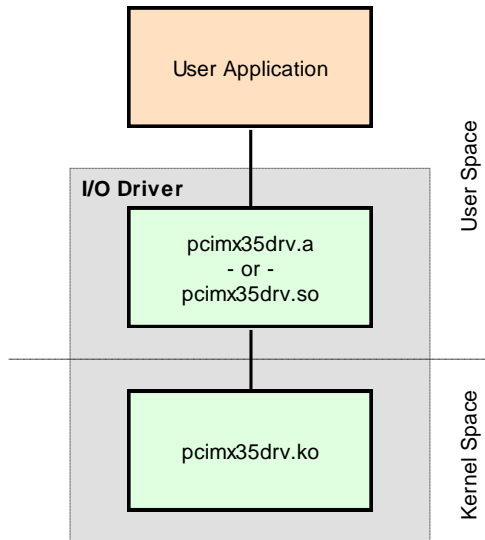


Figure 45: Structure of the I/O Driver for the ECUcore-iMX35

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

*pcimx35drv.h*: `/projects/ECUcore-iMX35/driver/pcimx35drv/include/pcimx35drv.h`  
Header file to describe the Driver interface  
This Header file is to be integrated into the Source code (\*.c) of the user project via `#include`.

*pcimx35drv.a*: `/projects/ECUcore-iMX35/driver/pcimx35drv/lib/pcimx35drv.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. This implies the advantage that explicit library paths must no longer be indicated when the application is started on the ECUcore-iMX35.

*pcimx35drv.so*: `/projects/ECUcore-iMX35/driver/pcimx35drv/lib/pcimx35drv.so`  
User space library of the I/O Driver to be dynamically loaded by the user application during runtime (similar to the usage of DLL in Windows)  
This version of the User space library is loaded into the address space of the user application during runtime. This implies the advantage that Driver libraries and user application are independent from each other and therefore interchangeable.  
Different user applications together can use the same version of the Driver library.  
Therefore, the environment variable `"LD_LIBRARY_PATH="` must be set on the ECUcore-iMX35, e.g.:

```
export LD_LIBRARY_PATH=.
```

*pcimx35drv.ko*: `/projects/ECUcore-iMX35/driver/pcimx35drv/lib/pcimx35drv.ko`  
Kernelspace module of the I/O Driver for accesses to hardware (PLD and Port pins).  
This module must be loaded using Linux command `"insmod"` – prior to starting the user application:

```
insmod pcimx35drv.ko
```

In the Demo program in section 7.6, the User space library of the I/O Driver is statically linked to the user application. Therefore, upon calling GCC, the static library *pcimx35drv.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/pcimx35drv.a
@$(CC) $(LDFLAGS) -o $@ $(OBJS) $(LIBS) $(LDLIBS)
```

Functions provided by the I/O Driver are listed in Header file "*pcimx35drv.h*", their usage is kept record of in the Demo program "*/projects/ECUcore-iMX35/user/demo*" (also see section 7.6).

The delivery status of the ECUcore-iMX35 includes a completely generated and ready-to-load (via "*insmod*") Kernel module of the I/O Driver filed in directory "*/home/bin*" (see section 5.13):

```
insmod /home/bin/pcimx35drv.ko
```

The Demo project in "*/projects/ECUcore-iMX35/user/demo*" shows an example for the application of the I/O Driver on the ECUcore-iMX35 (see sections 7.3.2 and 7.6).

### 7.3.2 I/O Driver Demo project

The Demo project for the I/O Driver is filed in directory "*/projects/ECUcore-iMX35/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-iMX35/driver/pcimx35drv*". Section 7.6 in much detail describes the Demo project as reference project for software development for the ECUcore-iMX35.

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

```
insmod pcimx35drv.ko
./demo
```

Figure 51 in section 7.6.1. exemplifies the execution of the Demo project on the ECUcore-iMX35. The Demo project can be closed by pressing "Ctrl+C".

## 7.4 CAN Driver for the ECUcore-iMX35

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

For the integration into own user projects, directory "*/projects/ECUcore-iMX35/driver/candrv*" of the Linux development system contains header files and completely generated libraries of the CAN Driver for the ECUcore-iMX35.

To access the CAN-Bus only the Userspace driver (*socketcan.a*) is required and must be linked to own projects. From the CAN Driver project directory, the following files are relevant for the integration into own user projects:

*cdrvinc.h*:           */projects/ECUcore-iMX35/driver/candrv/include/cdrvinc.h*  
Header file to describe Driver interfaces  
This Header file is to be integrated into the Source code (\*.c) of the user project via *#include*. It then includes all further Header files in directory "*include*" according to the correct sequence.



*candrv.a*:            /*projects/ECUcore-iMX35/driver/candrv/lib/candrv.a*  
                      Userspace library of the CAN Driver to be statically linked to user applications

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-iMX35/user/hellocan*" is an example for the usage of the CAN Driver on the ECUcore-iMX35 (see section 7.4.2).

Accessing a CAN interface requires a basic initialization as follows:

```
BR=125000
IF=can0
ifconfig $IF down
ip link set $IF type can bitrate $BR
ifconfig $IF up
```

The above commands are initializing can0 (IF) with a baud rate of 125 kBit/s (BR). Corresponding adjustments are needed if another CAN instance or baud rate is required.

## 7.4.2 CAN Driver Demo project

The Demo project for the CAN Driver is filed in directory "*projects/ECUcore-iMX35/user/hellocan*" of the Linux development system. It illustrates the access to the CAN interfaces of the module.

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 47).

**Advice:**        The USB-CANmodul and CAN-REport are not included on the scope of delivery of Development Kit ECUcore-iMX35. Both products are available as Bundle under order number SO-1054-U.

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 46 shows the execution of Demo project "*hellocan*" on the ECUcore-iMX35. The Demo project can be closed by pressing "Ctrl+C".

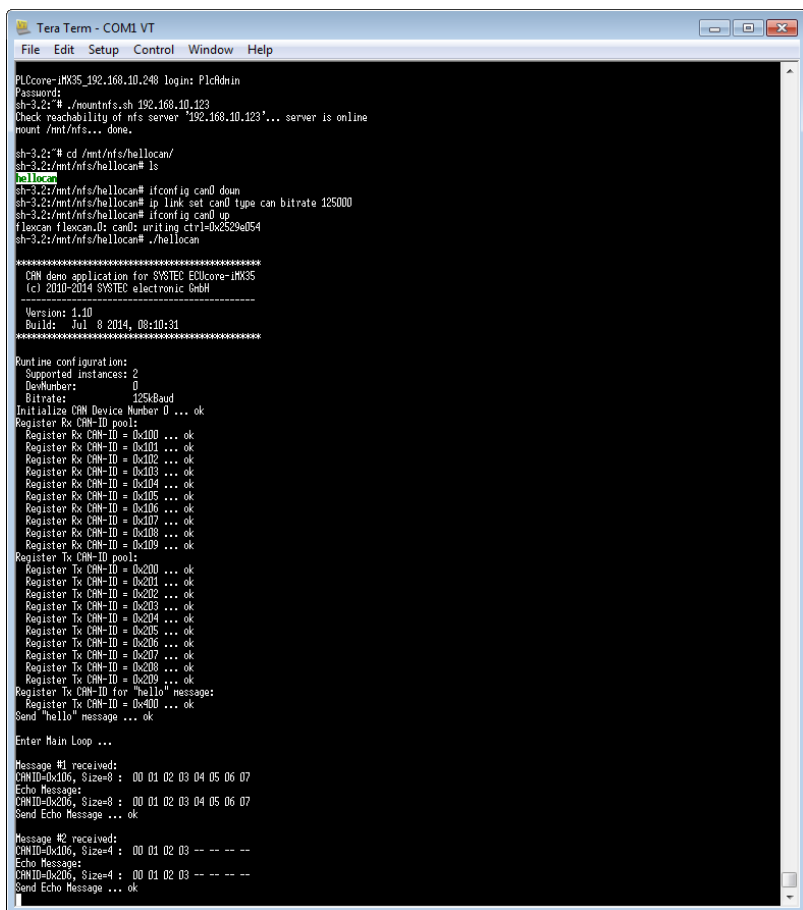


Figure 46: Execution of Demo project "hellocan" on the ECUcore-iMX35

Figure 47 shows data exchange with the Demo program "hellocan" in the CAN-Bus analysis tool "CAN-REport".

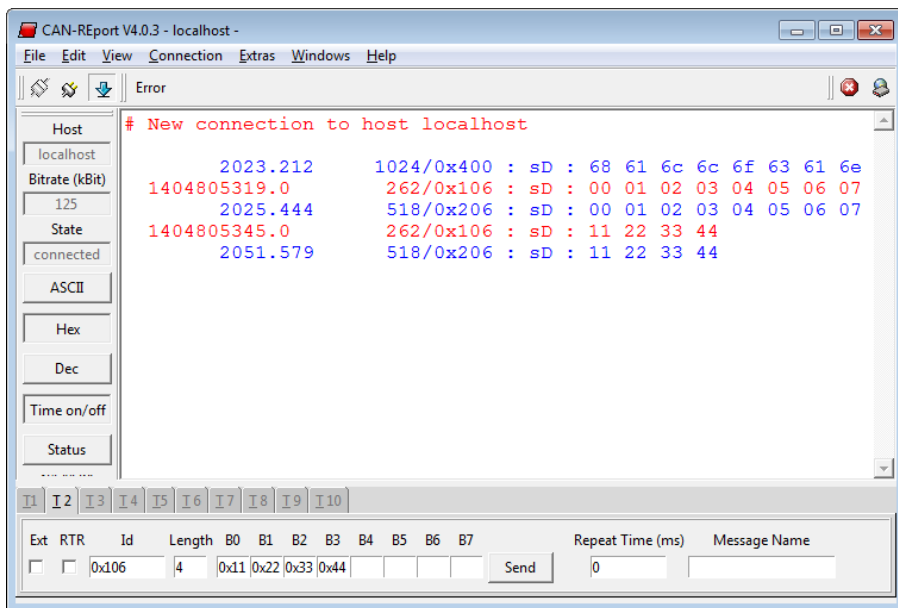


Figure 47: CAN analysis tool "CAN-REport"

## 7.5 Transferring programs to the ECUcore-iMX35

The configuration of the module as described in section 5.4 is one requirement for transferring – as well as starting – programs on the ECUcore-iMX35. Afterwards, login to the command shell of the ECUcore-iMX35 according to section 5.5.1 is necessary.

There are two possibilities for the transmission of programs on the ECUcore-iMX35 or data exchange between development system (VMware Linux-Image) and the ECUcore-iMX35 in general. Both imply advantages and disadvantages:

**NFS:** "Network File System" (NFS) represents the easiest way of directly starting a user program (translated in the Linux-Image) on the ECUcore-iMX35. To do so, from the VMware-Image of the Linux development system a directory is mounted into the local file system of the ECUcore-iMX35 ("mounted"). Enter the appropriate command on the ECUcore-iMX35 to start the program. Required data transfer from the development system to the ECUcore-iMX35 implicitly takes place via NFS – no further commands from the user are necessary. Consequently, it is assured that the ECUcore-iMX35 is always running a current program version rather than one that is out-of-date. Hence, NFS is especially suitable during software development. 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 7.5.1 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-iMX35 (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 command entry that is required for each data transmission. Usually, this is 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-iMX35 without noticing it. Section 7.5.2 describes the usage of FTP.

### 7.5.1 Using NFS

The easiest way of starting a user program on the ECUcore-iMX35 that was first translated within the Linux-Image is a direct integration ("to mount") of a directory from the VMware-Image of the Linux development system into the local file system of the ECUcore-iMX35. 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-iMX35:

#### 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-iMX35

To mount directory "*tftpboot*" of the Linux-Image into the local file system of the ECUcore-iMX35, command "*mount*" must be used as follows:

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

For example, to attach the ECUcore-iMX35 to the Linux development system via the IP address defined in section 6.5, the following command must be entered on the ECUcore-iMX35:

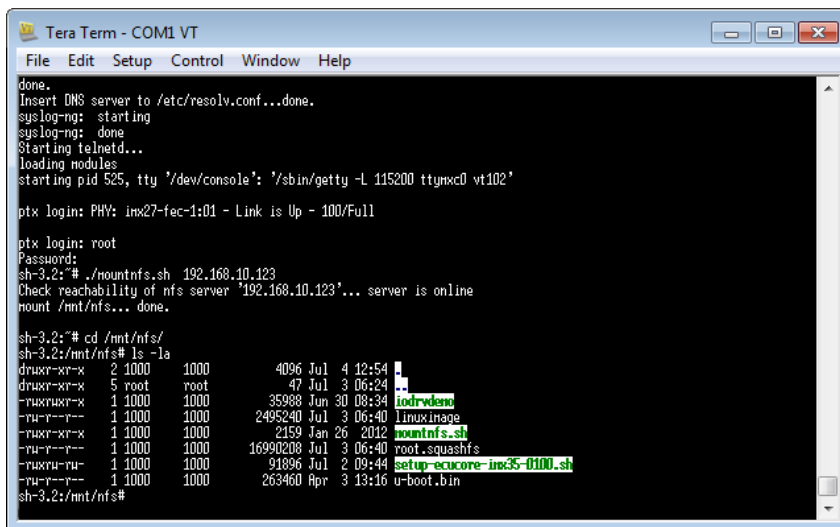
```
mount -t nfs -o nolock 192.168.10.123:/tftpboot /mnt/nfs
```

Script "*mountnfs.sh*" that is located in the home directory of the ECUcore-iMX35 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 192.168.10.123
```

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-iMX35.

Figure 48 summarizes all necessary steps to mount the Linux-Image into the local file system of the ECUcore-iMX35.



```
Tera Term - COM1 VT
File Edit Setup Control Window Help
done.
Insert DNS server to /etc/resolv.conf...done.
syslog-ng: starting
syslog-ng: done
Starting telnetd...
loading modules
starting pid 525, tty "/dev/console": "/sbin/getty -L 115200 ttyxcl vt102"
ptx login: PHY: imx27-fec-1:01 - Link is Up - 100/Full
ptx login: root
Password:
sh-3.2:~# ./mountnfs.sh 192.168.10.123
Check reachability of nfs server "192.168.10.123"... server is online
mount /mnt/nfs... done.
sh-3.2:~# cd /mnt/nfs/
sh-3.2:/mnt/nfs# ls -la
drwxr-xr-x  2 1000    1000      4096 Jul  4 12:54 .
drwxr-xr-x  5 root    root      47 Jul  3 06:24 ..
-rwxr-xr-x  1 1000    1000      35988 Jun 30 08:34 .iodrvdno
-rw-r--r--  1 1000    1000      2495240 Jul  3 06:40 linux.image
-rwxr-xr-x  1 1000    1000      2159 Jan 26 2012 .mountnfs.sh
-rw-r--r--  1 1000    1000      16990208 Jul  3 06:40 root.squashfs
-rwxr-xr-x  1 1000    1000      91896 Jul  2 09:44 setup-ecucore-imx35-0100.sh
-rw-r--r--  1 1000    1000      263460 Apr  3 13:16 u-boot.bin
sh-3.2:/mnt/nfs#
```

Figure 48: Mounting the Linux-Image into the local file system of the ECUcore-iMX35

## 7.5.2 Using FTP

As an alternative to the integration of the Linux-Image into the local file system of the ECUcore-iMX35 via NFS, data may be transferred into both directions between the development computer and the ECUcore-iMX35 via FTP. There, the ECUcore-iMX35 can function as both, server and client.

**When executable programs are transferred via FTP connection, it must be kept in mind that the "x"-Flag in data attributes ("eXecuteable") is always deleted. Hence, after each FTP transfer the "x"-Flag must be set again by using command "*chmod*" (also compare chapter "Calling programs" in section 10, "Tips & Tricks for Handling Linux"):**

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

### 7.5.2.1 ECUcore-iMX35 as FTP client

If the ECUcore-iMX35 is used as FTP client, the Linux development system acts as server. This option implies the advantage that there will be no safety risk if the device is applied in practice later on. The reason for this is that the server service on the module does not have to be activated and explicit user administration is not necessary. For using the ECUcore-iMX35 as FTP client, the IP address of the

Linux development system assigned via DHCP must first be determined. The procedure is given explanation in section 6.5.

### FTP Download

Downloading files from the Linux-Image onto the ECUcore-iMX35 takes place via command "*ftpget*". Parameters "*-u*" for username and "*-p*" for password are necessary for an authentication at the host system. Command "*ftpget*" is written as follows:

```
ftpget -u <username> -p <password> <ip_vmware_image> <local_file> <remote_file>
```

When looking at program "*demo*" for example that was translated in section 7.2 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-iMX35:

```
ftpget -u vmware -p vmware 192.168.10.123 /tmp/demo /tftpboot/demo/demo
```

### FTP Upload

Uploading files of the ECUcore-iMX35 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 written as follows:

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

The following command is required for example to transfer startup script "*autostart*" via FTP from the ECUcore-iMX35 into directory "*/tftpboot*" of the host system:

```
ftpput -u vmware -p vmware 192.168.10.123 /tftpboot/autostart  
/home/etc/autostart
```

Figure 49 exemplifies the usage of commands "*ftpget*" and "*ftpput*" to download and upload files via FTP.

```

Tera Term - COM1 VT
File Edit Setup Control Window Help
UBI: background thread "ubi_bgt0d" started, PID 490
UBIFS: recovery needed
UBIFS: recovery completed
UBIFS: mounted UBI device 0, volume 0, name "userpart"
UBIFS: file system size: 106326528 bytes (103834 KiB, 101 MiB, 812 LEBs)
UBIFS: journal size: 5368704 bytes (5242 KiB, 5 MiB, 41 LEBs)
UBIFS: media format: u4/r0 (latest is u4/r0)
UBIFS: default compressor: lzo
UBIFS: reserved for root: 4952683 bytes (4836 KiB)
done.
mounting filesystems...done
mtodoops: Attached to MTU device 3
running rc.d services...
Configure network interface lo...
Configure network interface eth0
MACADDR=00:50:c2:f8:06:87
IPADDR=192.168.10.190
GATEWAY=0.0.0.0
NETMASK=255.255.255.0
DNSSERVER=0.0.0.0
Enable netueth0: Freescale FEC PHY driver [Generic PHY] (nii_bus:phy_addr=imx27-fec-1:01, irq=-1)
rk interface...done.
Add default route...route: SIOCADDRT: Invalid argument
done.
Insert DNS server to /etc/resolv.conf...done.
syslog-ng: starting
syslog-ng: done
Starting telnetd...
loading modules
starting pid 525, tty '/dev/console': '/sbin/getty -L 115200 ttyux0 vt102'

ECUcore-iMX35 login: PHY: imx27-fec-1:01 - Link is Up - 100/Full

ECUcore-iMX35 login: PlcHdnIn
Password:
sh-3.2:~# ftpput -u cmmiare -p cmmiare 192.168.10.123 /tftpboot/autostart /home/etc/autostart
sh-3.2:~# ls /tmp
sh-3.2:~# ftpget -u cmmiare -p cmmiare 192.168.10.123 /tmp/demo /tftpboot/demo/demo
sh-3.2:~# ls /tmp
demo
sh-3.2:~#

```

Figure 49: Download and upload via FTP

### 7.5.2.2 ECUcore-iMX35 as FTP server

The ECUcore-iMX35 provides a FTP server (FTP 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-iMX35 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-iMX35 or to select log files from this module via a graphic FTP client on his Windows laptop.

Section 5.5.2 explains the activation of the FTP server on the ECUcore-iMX35 and the login to the module via a FTP client. Section 5.1 specifies suitable FTP client programs.

## 7.6 Translation and execution of demo project "demo"

### 7.6.1 Usage of "make"

Directory ***"/projects/ECUcore-iMX35/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 filed in ***"/projects/ECUcore-iMX35/driver/pcimx35drv"*** 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.

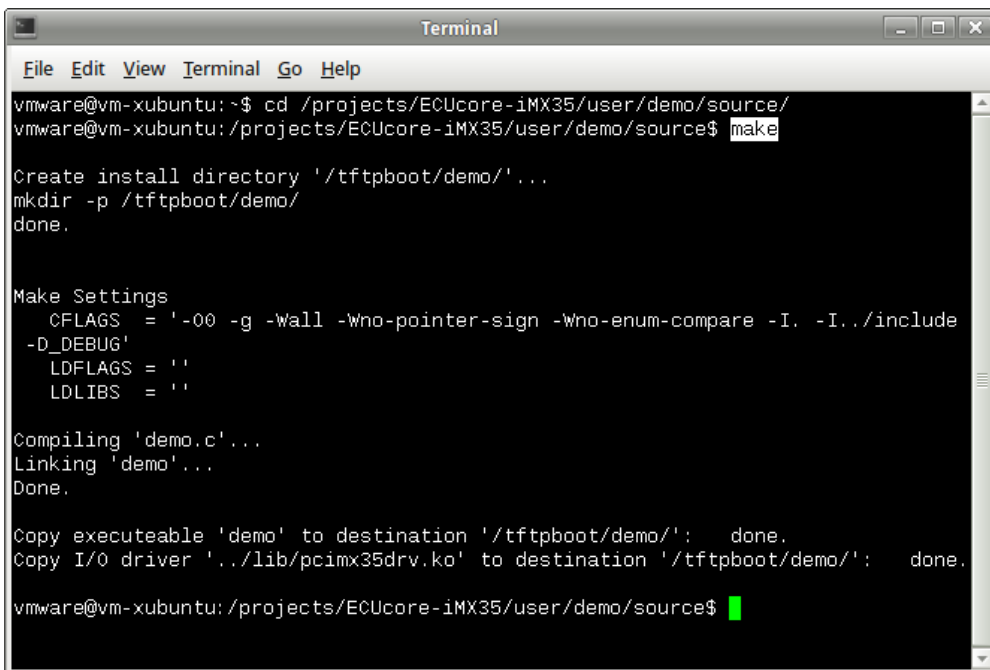
The source code can be created or edited via Windows network environment in any Windows Editor. On the contrary, translating the project is only possible within a Linux development environment.

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-iMX35/user/demo/source"`:

```
cd /projects/ECUcore-iMX35/user/demo/source
```

Afterwards, command "make" must be started. Figure 50 shows the usage of commands "cd" and "make" for the exemplary demo program contained in the VMware-Image.



```

Terminal
File Edit View Terminal Go Help
vmware@vm-xubuntu:~$ cd /projects/ECUcore-iMX35/user/demo/source/
vmware@vm-xubuntu:/projects/ECUcore-iMX35/user/demo/source$ make

Create install directory '/tftpboot/demo/'...
mkdir -p /tftpboot/demo/
done.

Make Settings
  CFLAGS = '-O0 -g -Wall -Wno-pointer-sign -Wno-enum-compare -I. -I../include
  -D_DEBUG'
  LDFLAGS = ''
  LDLIBS = ''

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

Copy executable 'demo' to destination '/tftpboot/demo/': done.
Copy I/O driver '../lib/pcimx35drv.ko' to destination '/tftpboot/demo/': done.

vmware@vm-xubuntu:/projects/ECUcore-iMX35/user/demo/source$ █

```

Figure 50: Translating 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 (file "pcimx35drv.ko") are copied to directory `"/tftpboot/demo"` (compare screenshot in Figure 50). All required steps are completed after the translation and copying is finished.

All further steps exclusively take place on the ECUcore-iMX35. 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-iMX35 (see section 7.5.1):

```
mount -t nfs -o nolock 192.168.10.123:/tftpboot /mnt/nfs
```

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

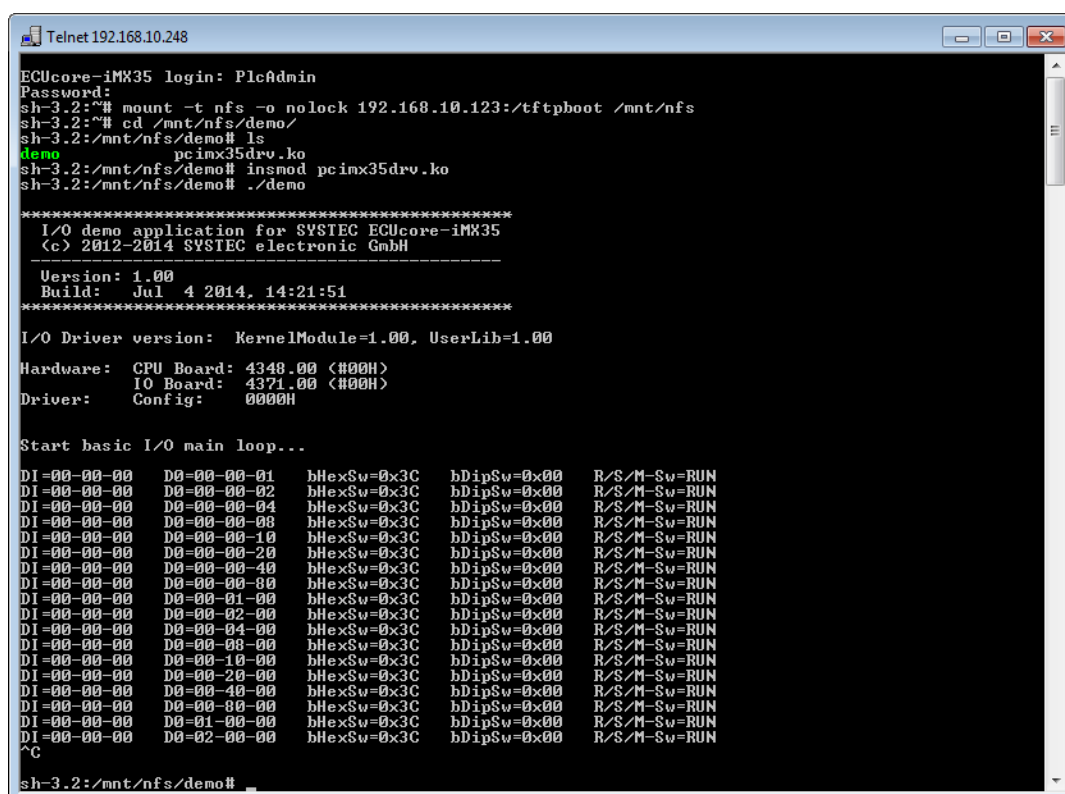
```
cd /mnt/nfs/demo
```

Directory `"/mnt/nfs"` on the ECUcore-iMX35 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-iMX35 in directory `"/mnt/nfs/demo"` of its file system. It is not necessary to explicitly download executable binary files to the ECUcore-iMX35.

### 3. Starting the demo program on the ECUcore-iMX35.

Since the demo program accesses in- and outputs of the ECUcore-iMX35, it requires I/O driver `"pcimx35drv"` to do so. Consequently, the I/O driver must be loaded using command `"insmod"`. Afterwards, the demo program can be started:

```
insmod pcimx35drv.ko
./demo
```



```
Telnet 192.168.10.248
ECUcore-iMX35 login: PlcAdmin
Passwort:
sh-3.2:~# mount -t nfs -o nolock 192.168.10.123:/tftpboot /mnt/nfs
sh-3.2:~# cd /mnt/nfs/demo/
sh-3.2:/mnt/nfs/demo# ls
demo
sh-3.2:/mnt/nfs/demo# ./demo
I/O demo application for SVSTEC ECUcore-iMX35
(c) 2012-2014 SVSTEC electronic GmbH
-----
Version: 1.00
Build: Jul 4 2014, 14:21:51
-----
I/O Driver version: KernelModule=1.00, UserLib=1.00
Hardware: CPU Board: 4348.00 <#00H>
          IO Board: 4371.00 <#00H>
Driver:   Config: 0000H
Start basic I/O main loop...
DI=00-00-00 D0=00-00-01 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-00-02 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-00-04 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-00-08 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-00-10 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-00-20 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-00-40 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-00-80 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-01-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-02-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-04-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-08-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-10-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-20-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-40-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=00-80-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=01-00-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
DI=00-00-00 D0=02-00-00 bHexSw=0x3C bDipSw=0x00 R/S/M-Sw=RUN
^C
sh-3.2:/mnt/nfs/demo#
```

Figure 51: Executing the demo project "demo" on the ECUcore-iMX35

Figure 51 demonstrates the execution of the demo project on the ECUcore-iMX35. To finish the demo project simply press "Ctrl+C".

## 7.6.2 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, translating and debugging user programs within a comfortable development environment – similar to "Visual Studio" for example. 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-iMX35/user/demo"` (also compare descriptions in section 7.6.1).



### 7.6.2.1 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 52. 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-iMX35/user/demo/workspace"*.

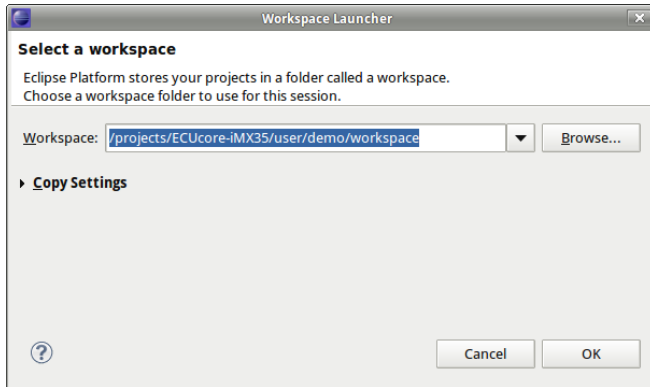


Figure 52: Eclipse dialog "Workspace Launcher"

After activating pushbutton "OK" the graphical surface of the IDE starts automatically and loads the workspace entered. Figure 53 provides an overview of "Eclipse" after it started.

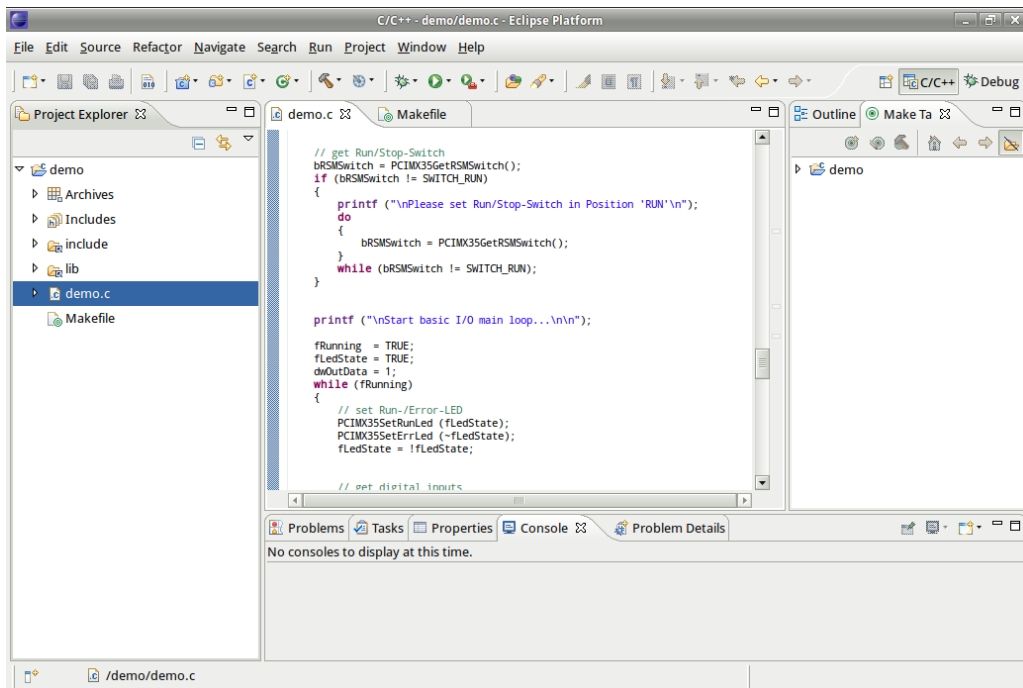


Figure 53: The graphical IDE "Eclipse"

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.

### 7.6.2.2 Translating the demo project

To translate 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 54). Hence, Eclipse executes the Makefile of the demo project in directory "source" ("projects/ECUcore-iMX35/user/demo/source/Makefile"). This is the same Makefile that was called manually in a Terminal window as described section 7.6.1. Accordingly, all messages shown in IDE window "Console" are identical to the ones shown in Figure 50. In the same way, the demo program (file "demo") and the I/O drivers needed to execute it (file "pcimx35drv.ko") are copied into directory "/tftpboot/demo". Consequently, the demo program can be started on the ECUcore-iMX35 after successful completion of the Build process as explained in section 7.6.1.

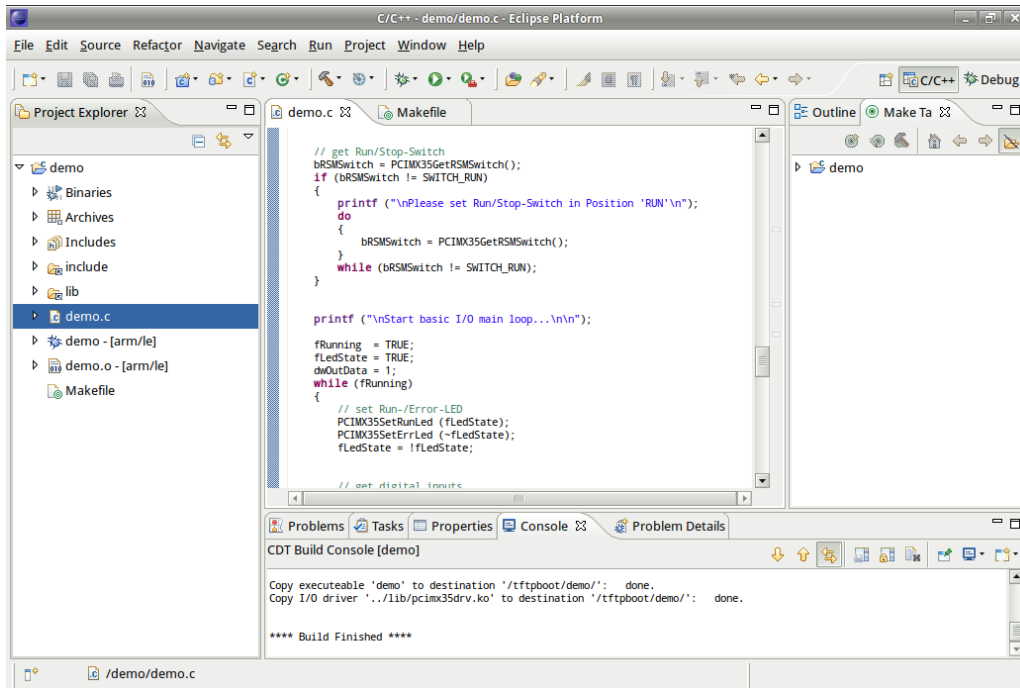


Figure 54: Translating the demo project in Eclipse

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 54) 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).

### 7.6.2.3 Debugging the demo project in the IDE

One of the most considerable advantages for using Eclipse is the possibility to debug the translated program directly on high-level language within the IDE. This includes for example line-by-line-eradication 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-iMX35 – 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-iMX35 (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-iMX35**

For debug simply start the demo program on the ECUcore-iMX35 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-iMX35 via NFS (see section 7.5.1):

```
mount -t nfs -o nolock 192.168.10.123:/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-iMX35 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-iMX35 in directory `"/mnt/nfs/demo"` of its file system. It is not necessary to explicitly download executable binary files to the ECUcore-iMX35.

- 1.3 The demo program needs the I/O driver `"pcimx35drv"` to access the in- and outputs on the ECUcore-iMX35. Consequently, command `"insmod"` must be used to explicitly load the driver:

```
insmod pcimx35drv.ko
```

- 1.4 Starting the demo program on the ECUcore-iMX35 is controlled by the debug server. Therefore, command `"gdbserver"` is required and is written as follows:

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

For the example above this would be the following call:

```
gdbserver 192.168.10.123:10000 ./demo
```

Use `"192.168.10.123"` 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 58).

Figure 55 illustrates the steps to be accomplished on the ECUcore-iMX35.

```

Tera Term - COM1 VT
File Edit Setup Control Window Help
rk interface...done.
Add default route...route: $!OCDADRRT: Invalid argument
done.
Insert DNS server to /etc/resolv.conf...done.
syslog-ng: starting
syslog-ng: done
Starting telnetd...
loading modules
starting pid 525, tty '/dev/console': '/sbin/getty -L 115200 ttymxc0 vt102'
ptx login: PHY: imx27-fec-1:01 - Link is Up - 100/Full
ptx login: PlcAdmin
Password:
sh-3.2: # mount -t nfs -o nolock 192.168.10.123:/ftpboot /mnt/nfs
sh-3.2: # cd /mnt/nfs/deno/
sh-3.2: /mnt/nfs/deno# ls
deno          pcimx35drv.ko
sh-3.2: /mnt/nfs/deno# insmod pcimx35drv.ko
sh-3.2: /mnt/nfs/deno# gdbserver 192.168.10.122:10000 ./deno
Process ./deno created; pid = 561
Listening on port 10000

```

Figure 55: Starting the Debug server on the ECUcore-iMX35

To **simplify** this process, the delivery status of the ECUcore-iMX35 includes Shell script "**debug.sh**" in the directory *"/home"*. This script summarizes all in 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 switched 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 56. Hence, follow the steps listed below:

## 2.1 Determining the program that is to be executed in the Debugger (see Figure 56)

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" for the above example).

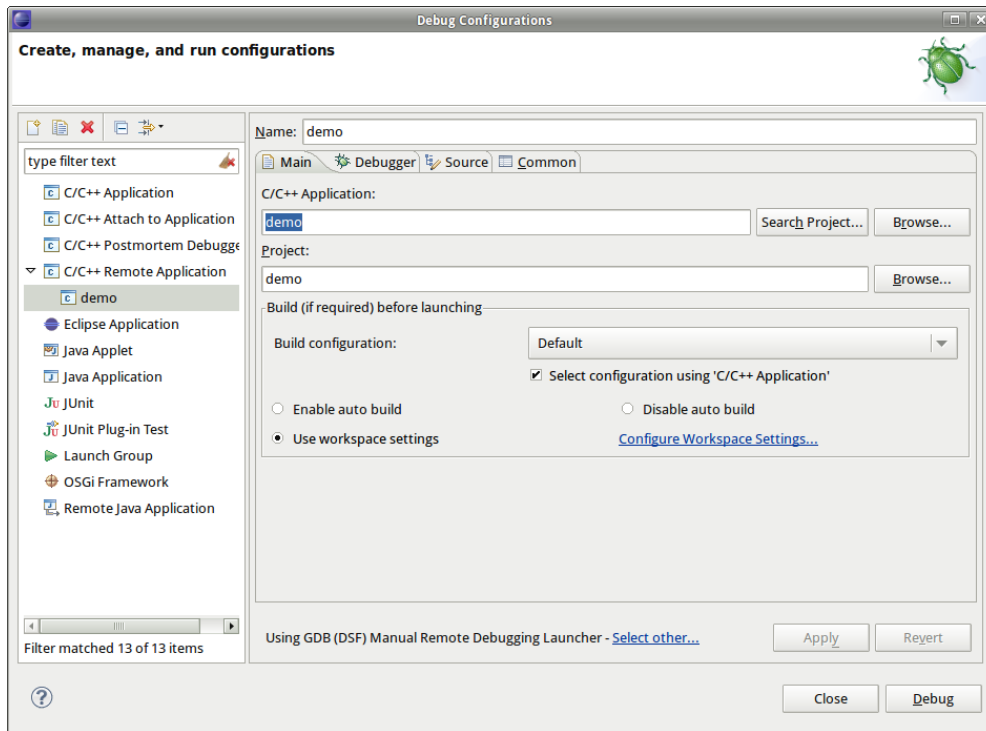


Figure 56: Determining the application that is to be debugged

## 2.2 Selecting the GDB Debugger to be used (see Figure 57)

Host-sided use the GDB Debugger that is included in the GNU-Crosscompiler Toolchain for ARM11 processors. To select it, activate Tabsheet "Debugger". Enter GDB Debugger of the Crosscompiler Toolchain in section "Debugger Options" in Sub-Tabsheet "Main" area "GDB debugger" (see Figure 57; for the above example this would be: "arm-none-linux-gnueabi-gdb").

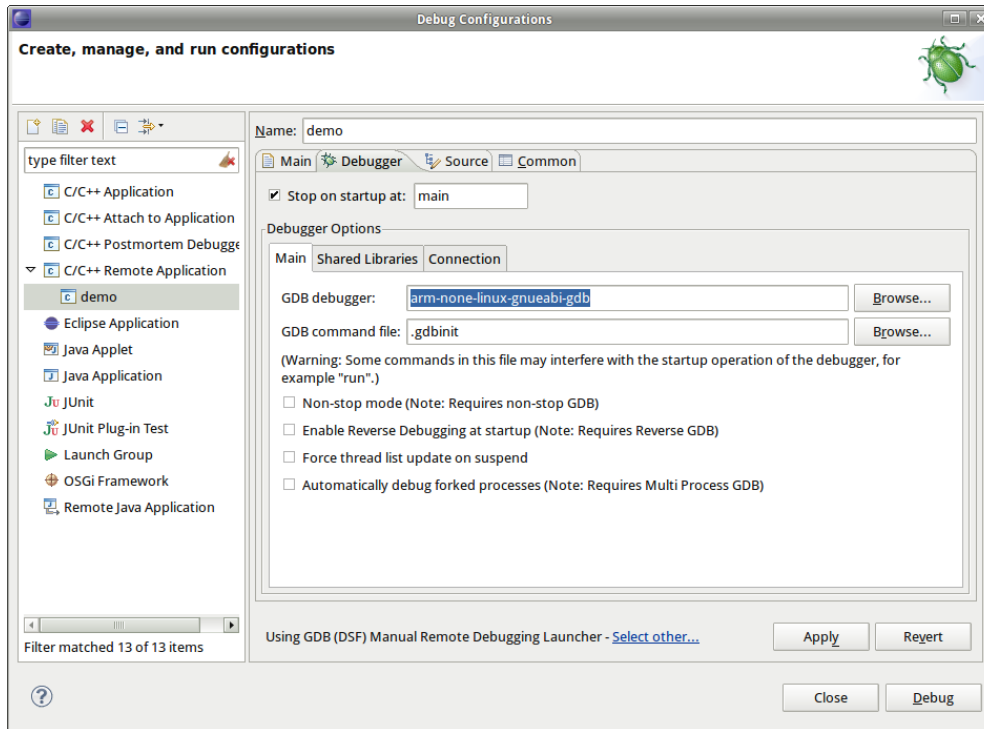


Figure 57: Selecting the GDB Debugger

### 2.3 Configuring the connection to the Target (see Figure 58)

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 58). In area *"Host name or IP address"* the IP address of the ECUcore-iMX35 as defined in section 5.4 must be entered ("*192.168.10.248*" for the example above). Enter the Port number defined in point 1.4 in area *"Port number"* when command *"gdbserver"* is called (for this example: "*10000*").

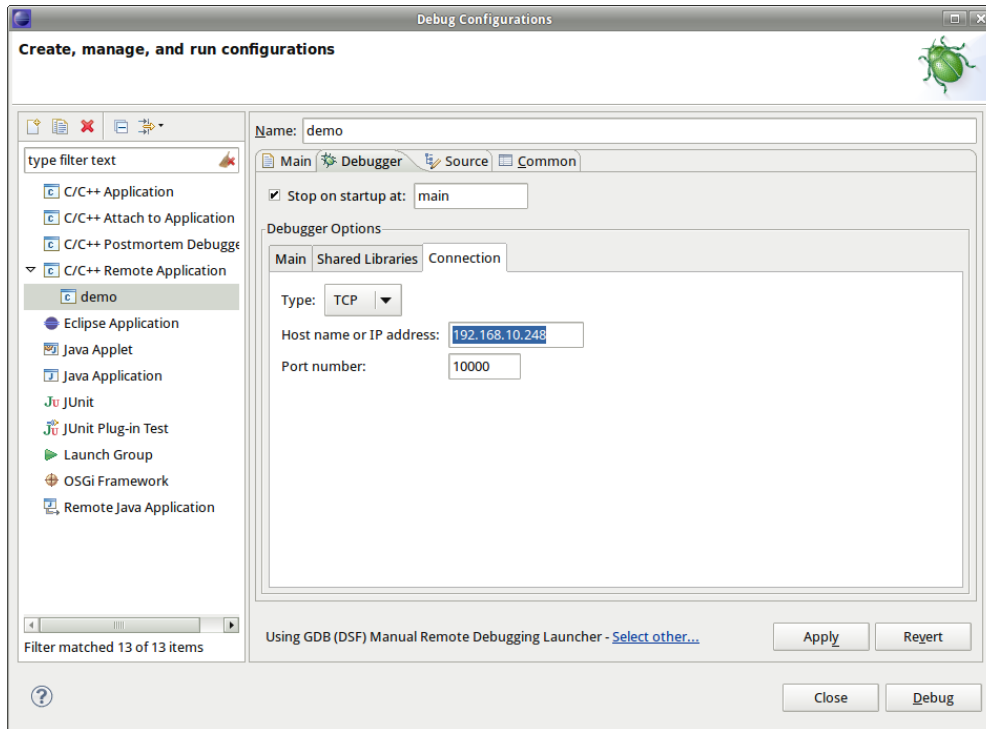


Figure 58: Configuring the connection to the Target

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 59). Table 12 lists up the most important Debugger commands.

Table 12: 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 59 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.

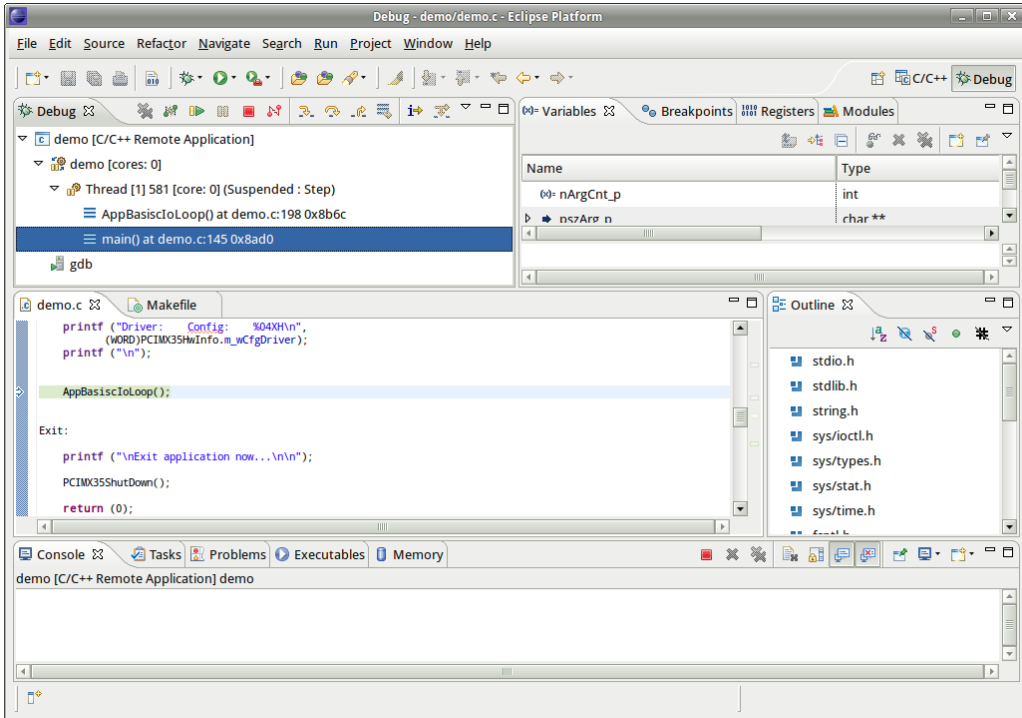


Figure 59: Debugging the Demo project in Eclipse

Figure 60 illustrates Terminal outputs that are generated on the ECUcore-iMX35 during the debugging process.

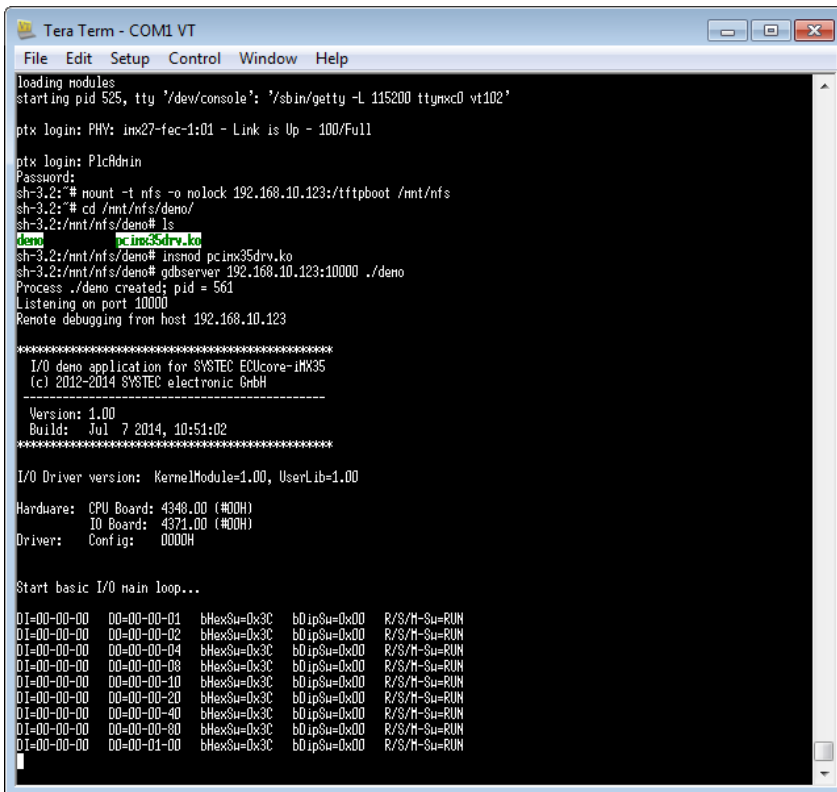


Figure 60: Terminal outputs of the ECUcore-iMX35 during debugging



## 7.7 Configuration and Translation of Linux-Image and U-Boot

All Linux sources for the ECUcore-iMX35 are filed in the VMware-Image of the Linux development system in directory `"/projects/ECUcore-iMX35/LinuxBSP"`. To modify the configuration of the Linux kernel, use command `"cd"` in a console window to switch to the directory `"/projects/ECUcore-iMX35/LinuxBSP/ECUcore-iMX35"`. Afterwards, call command `"ptxdist kernelconfig"`:

```
cd /projects/ECUcore-iMX35/LinuxBSP/ECUcore-iMX35
ptxdist kernelconfig
```

Figure 61 shows the typical surface for the configuration of Linux kernel.

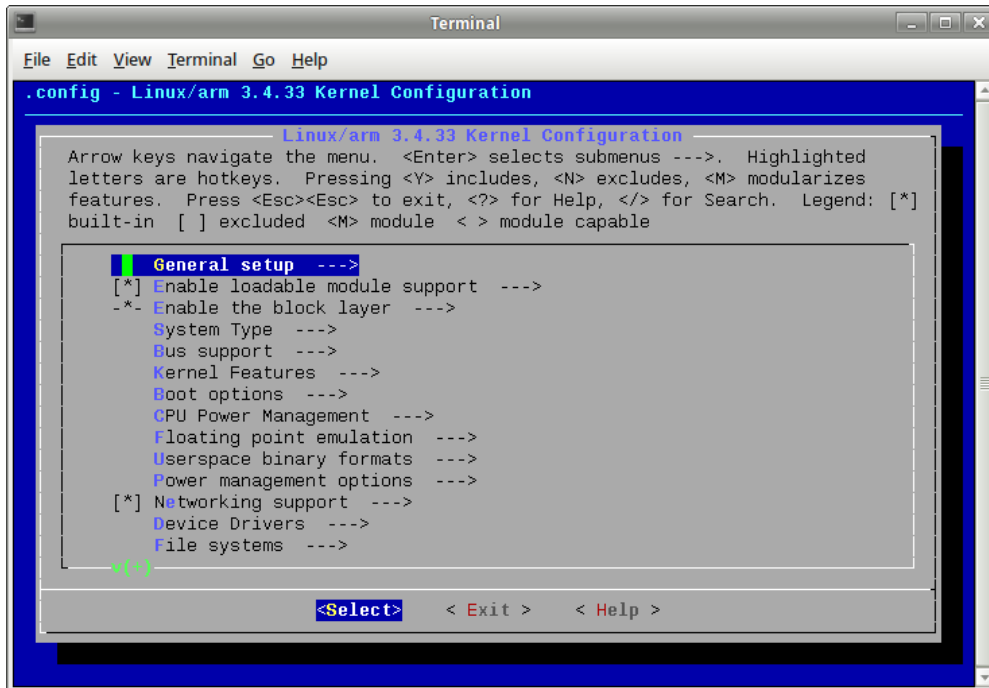


Figure 61: User surface for the configuration of the Linux kernel

The configuration of user applications including BusyBox takes place in the same directory where the configuration of the Linux kernel takes place. Therefore, command `"ptxdist menuconfig"` must be called:

```
cd /projects/ECUcore-iMX35/LinuxBSP/ECUcore-iMX35
ptxdist menuconfig
```

Figure 62 shows the typical user interface to configure user applications including BusyBox.

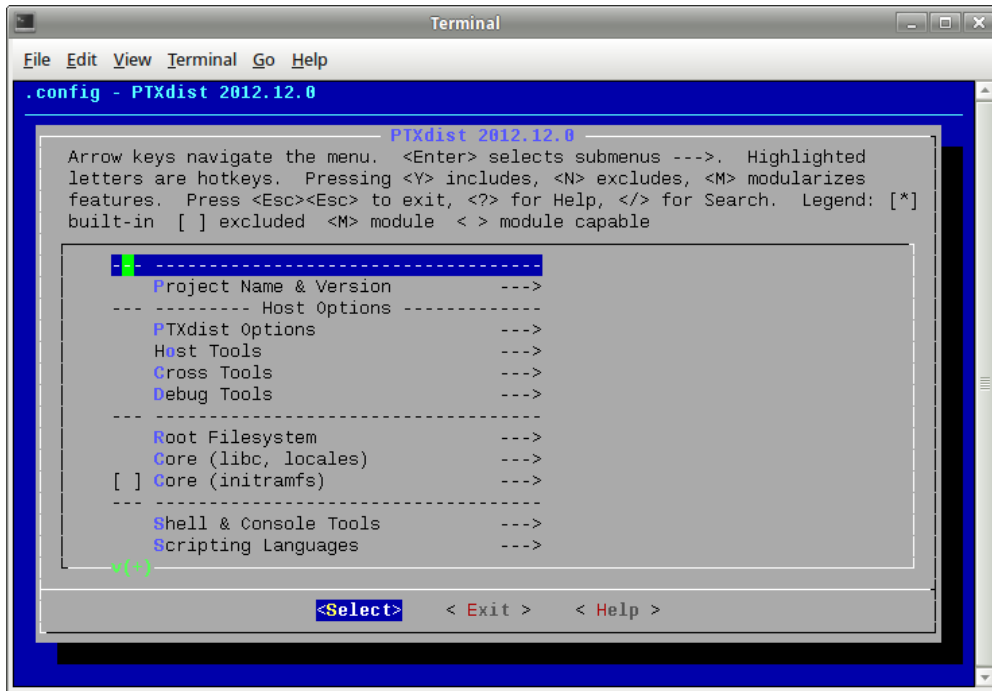


Figure 62: User interface to configure user applications including BusyBox

After configuring the Linux kernel, user applications and BusyBox, all software components are translated and summarized to a new image:

```
cd /projects/ECUcore-iMX35/LinuxBSP/ECUcore-iMX35
ptxdist clean
ptxdist images
```

To regenerate the bootloader "U-Boot", switch to directory **"/projects/ECUcore-iMX35/LinuxBSP/ECUcore-iMX35 "** and call the following commands:

```
cd /projects/ECUcore-iMX35/LinuxBSP/ECUcore-iMX35
ptxdist clean u-boot
ptxdist images
```

By calling Shell script **"copy\_image"**, the Linux-Image and "U-Boot" are copied into directory **"/ftpboot"**. From there, both software components can be loaded onto the ECUcore-iMX35 (see section 0):

```
cd /projects/ECUcore-iMX35/LinuxBSP
./copy_image
```

## 8 Adaptation and Testing of the hardware connections

### 8.1 Driver Development Kit (DDK) for the ECUcore-iMX35

The Driver Development Kit (DDK) for the ECUcore-iMX35 is distributed as additional software package with the order number SO-1119. It is not included in the delivery of the Development Kit ECUcore-iMX35. The "Software Manual Driver Development Kit for the ECUcore-iMX35" (Manual no.: L-1572) provides details about the DDK.

The Driver Development Kit for the ECUcore-iMX35 enables the user to adapt an I/O level to self-developed baseboards. Consequently, the user is able to completely adapt the I/O driver to own requirements.

By using the DDK, the following resources may be integrated into the I/O level:

- Periphery (usually GPIO) of the ARM11JF-S
- Address-/Data Bus (memory-mapped periphery)
- SPI-Bus and I<sup>2</sup>C-Bus
- All other resources provided by the operating system, e.g. file system and TCP/IP

Figure 63 provides an overview of the DDK structure and its components. The DDK contains amongst others the source code of the Linux kernel driver (*pcimx35drv.ko*) and the Linux user library (*pcimx35drv.so*).

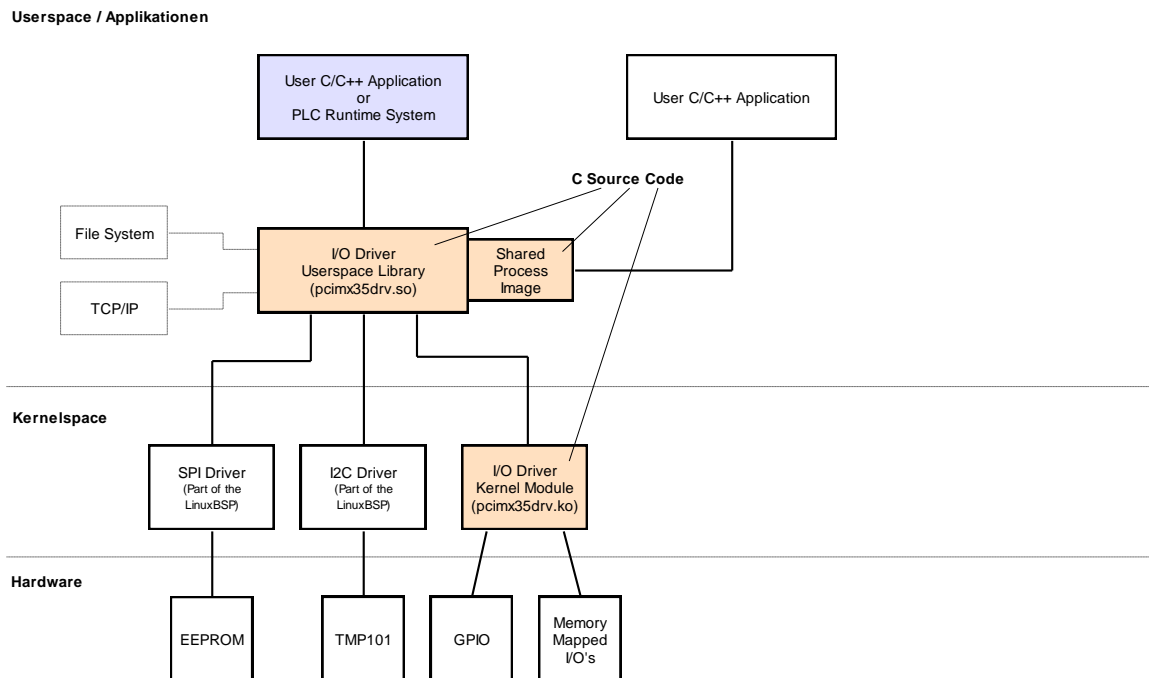


Figure 63: Overview of the Driver Development Kit for the ECUcore-iMX35

### **Scope of delivery / components of the DDK:**

The DDK contains the following components:

1. Source code for the Linux kernel driver (*pcimx35drv.ko*, see Figure 63); includes all files necessary to regenerate kernel drivers (C and H files, Make file etc.)
2. Source code for the Linux user library (*pcimx35drv.so*, see Figure 63); contains all files (incl. implementation of Shared Process Image) necessary to regenerate a user library (C and H files, Make file etc.)
3. I/O driver demo application (*iodrvdemo*) in the source code; allows for a quick and trouble-free test of the I/O drivers
4. Documentation

The Driver Development Kit is based on the software package **SO-1121** ("VMware-Image of the Linux development system"). It contains sources of the LinuxBSP used and it includes the necessary GNU-Crosscompiler Toolchain for ARM11 processors.

## **8.2 Testing the hardware connections**

The ECUcore-iMX35 primarily is designed as vendor part for the application in industrial controls. Hence, the ECUcore-iMX35 typically is integrated in a user-specific baseboard. To enable trouble-free inspection of correct I/O activation, the Testing program "*iodrvdemo*" can be used. This test program is directly tied in with the I/O driver and allows quick and direct access to the periphery. The Testing program is preinstalled as ready-to-use Binary "*/home/bin/iodrvdemo*" on the ECUcore-iMX35. Moreover, the program sources are included as reference application in the I/O driver project (see section 7.3).

Start the Testing program "*iodrvdemo*" as follows:

```
cd /home/bin
insmod pcimx35drv.ko
./iodrvdemo
```

Figure 64 illustrates testing the hardware connection by using "*iodrvdemo*".

```
Telnet 192.168.10.248
ECUcore-iMX35 login: PlcAdmin
Password:
sh-3.2:~# cd /home/bin/
sh-3.2:~/bin# insmod pcimx35drv.ko
sh-3.2:~/bin# ./iodrvdemo

*****
Test application for SYSTEC PLCcore-iMX35 board driver
Version: 1.00
(c) 2011-2014 SYS TEC electronic GmbH, www.systec-electronic.com
*****

I/O Driver version: KernelModule=1.00, UserLib=1.00

Hardware: CPU Board: 4348.00 (<#00H>)
          IO Board: 4371.00 (<#00H>)
IO config: Digital In: 16
          Digital Out: 10
          Analog In: 4
          Analog Out: 0
          Counter: 0
          PWM/PTO: 1
          TempSensor: 1
Driver: Config: 0000H

Please Select:
0 - Exit this application
1 - Run Basic I/O test <digital I/O and user switches>
2 - Run Counter test
3 - Run PWM test <pre-configured demo>
4 - Run PWM test <manual parameter input>
5 - Run PTO test <pre-configured demo>
6 - Run PTO test <manual parameter input>
7 - Run ADC test
8 - Run EEPROM test
T - Run Temperature Sensor test
W - Watchdog test
Select: 1

=== Basic I/O Test ===
Start basic I/O main loop... <press ESC to abort>
DI=0x00-0x00-0x00 D0=0x00-0x00-0x01 bHexSwitch=0x3C bDipSwitch=0x00 R/S/M-Switch = RUN
DI=0x00-0x00-0x00 D0=0x00-0x00-0x02 bHexSwitch=0x3C bDipSwitch=0x00 R/S/M-Switch = RUN
DI=0x00-0x00-0x00 D0=0x00-0x00-0x04 bHexSwitch=0x3C bDipSwitch=0x00 R/S/M-Switch = RUN
```

Figure 64: Testing the hardware connections using "iodrvdemo"

## 9 Using the USB and SD interface

### 9.1 Using the USB interface

The Embedded Linux used on ECUcore-iMX35 supports the usage of USB devices via "Hot Plug&Play". The ECUcore-iMX35 functions as USB host and is able to address USB devices such as USB memory sticks. All necessary drivers are already included in the Linux-Image of ECUcore-iMX35.

Responses to plugging and removing USB memory sticks can be flexibly adjusted by the user. "mdev" which is included in the Linux-Image undertakes the signaling and processing of events on system-level such as plugging or removing USB memory sticks. All relevant events are further reported to the user-specific Shell script `/home/etc/hotplug.sh` through an entry in configuration file `/etc/mdev.conf`. The default implementation of `hotplug.sh` which is included in the scope of delivery of ECUcore-iMX35 implements the following:

When a stick is plugged, the Shell script `/home/etc/hotplug.sh` automatically mounts the new device into subdirectory `/mnt/usb` (Linux command `mount`). Afterwards, the user-specific script `/home/etc/diskadded.sh` is executed. When the stick is removed, the connection to the subdirectory is canceled (Linux command `umount`) and the user-specific script `/home/etc/diskremoved.sh` is processed. Upon calling both scripts, parameter `"$1"` is assigned to both scripts as appropriate directory for the corresponding USB device. Hence, user-defined actions can be automated as being a reaction to these events. Figure 65 illustrates the above descriptions.

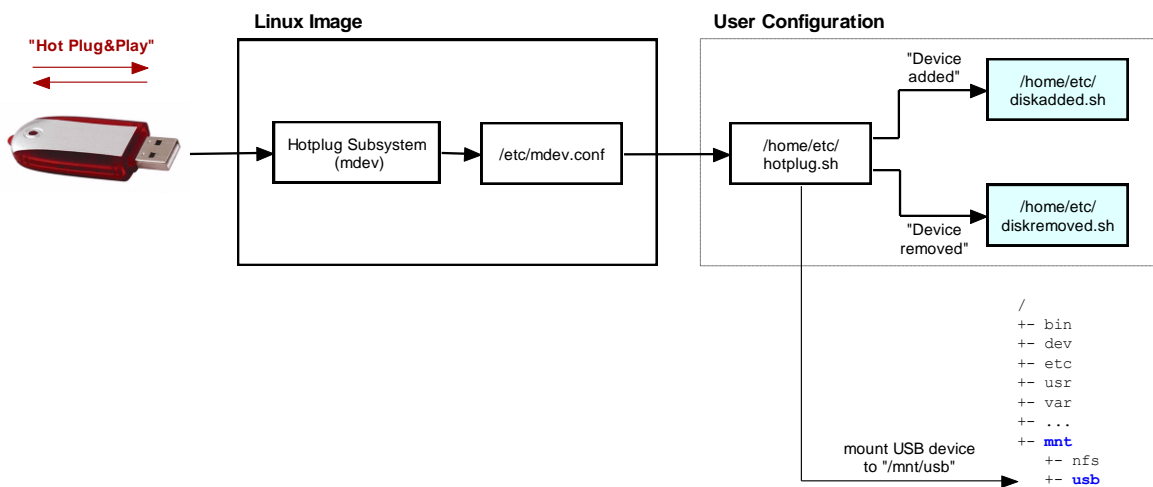


Figure 65: Internal procedures for plugging or removing USB memory sticks

The following example shows a simplified version for the Shell script `diskadded.sh` which automatically lists the files of an USB memory stick when it is plugged:

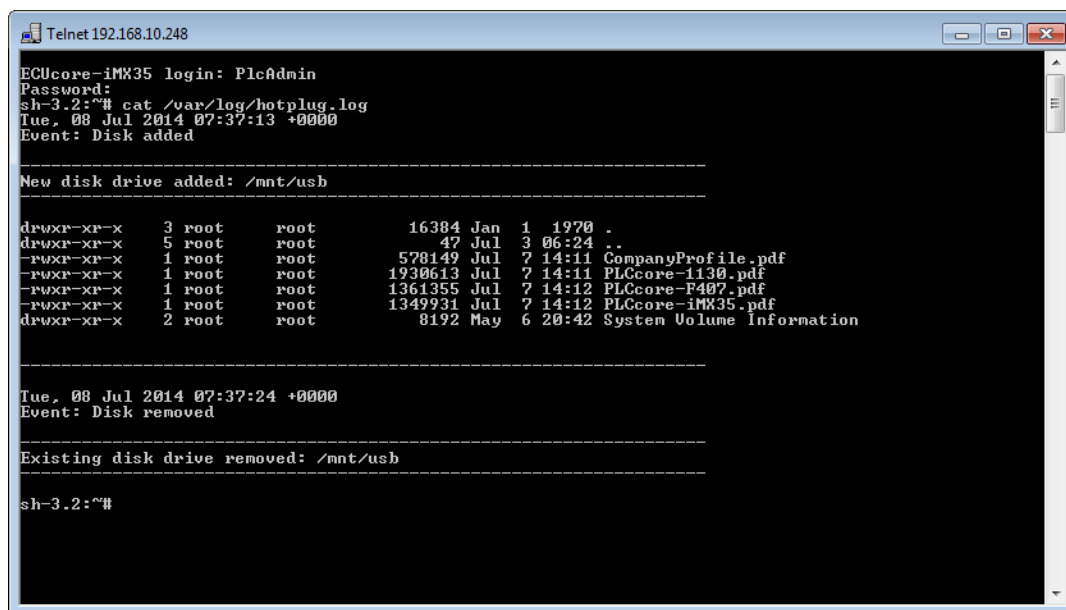
```
#!/bin/sh

echo
echo "-----"
echo -n "New disk drive added: $1"
echo "-----"
echo
ls -la $1
echo
echo "-----"
echo
```

The user-specific Shell script `hotplug.sh` as well as the scripts `diskadded.sh` and `diskremoved.sh` called by it, are all executed within a system process to which a console is assigned. Consequently, the outputs generated by the scripts do not appear in the terminal window. To still display these outputs anyway, script `hotplug.sh` redirects all messages into file `/var/log/hotplug.log`. This file could for example be output in the terminal window by using Linux command `cat`:

```
cat /var/log/hotplug.log
```

Figure 66 exemplifies the output of messages generated by the scripts when plugging and removing USB memory sticks.



```
Telnet 192.168.10.248
ECUcore-iMX35 login: PlcAdmin
Password:
sh-3.2:~# cat /var/log/hotplug.log
Tue, 08 Jul 2014 07:37:13 +0000
Event: Disk added

-----
New disk drive added: /mnt/usb
-----
drwxr-xr-x  3 root  root    16384 Jan  1  1970 .
drwxr-xr-x  5 root  root     47 Jul  3  06:24 ..
-rwxr-xr-x  1 root  root   578149 Jul  7  14:11 CompanyProfile.pdf
-rwxr-xr-x  1 root  root   1930613 Jul  7  14:11 PLCcore-1130.pdf
-rwxr-xr-x  1 root  root   1361355 Jul  7  14:12 PLCcore-P407.pdf
-rwxr-xr-x  1 root  root   1349931 Jul  7  14:12 PLCcore-iMX35.pdf
drwxr-xr-x  2 root  root     8192 May  6  20:42 System Volume Information

-----
Tue, 08 Jul 2014 07:37:24 +0000
Event: Disk removed

-----
Existing disk drive removed: /mnt/usb

sh-3.2:~#
```

Figure 66: Output of messages redirected into file `/var/log/hotplug.log`

The preinstalled version of Shell script `diskadded.sh` which is included in the delivery of ECUcore-iMX35 verifies if a file `autostart` is located in the root directory of the plugged USB memory stick. If such a file is provided, it will be called and executed. This allows for individual actions such as automated upload of software updates onto the ECUcore-iMX35.

**Advice for using USB memory sticks:**

Writing accesses to USB sticks take place asynchronously. The return of the "write" function does not imply that all data has been completely written onto the stick. In fact, the data is temporarily buffered in the RAM of the ECUcore-iMX35 and afterwards transferred onto the stick (in the background). In C/C++ programs, the writing process is finished after calling "close". Command "sync" is used for Shell scripts.

**9.2 Using the SD interface**

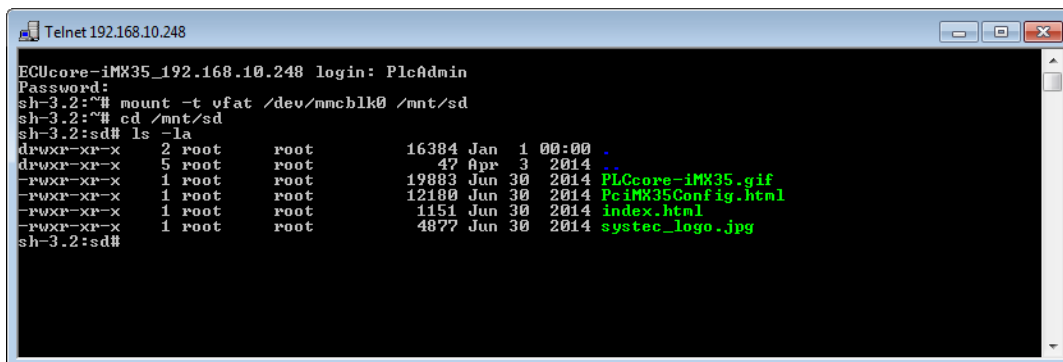
The Embedded Linux used for the ECUcore-iMX35 supports the usage of SD cards. All necessary drivers are already included in the Linux-Image of the ECUcore-iMX35. The SD card can be mounted into the local file system of the ECUcore-iMX35 by using Linux command "mount":

```
mount -t vfat /dev/mmcblk0 /mnt/sd
```

After executing the "mount" command, the SD card is mounted in path "/mnt/sd". Use the following command to list all files includes on the SD card:

```
cd /mnt/sd
ls -la
```

Figure 67 illustrates the loading process of drivers, the mounting of the SD card into the local file system and the access to the SD card at ECUcore-iMX35.



```
Telnet 192.168.10.248
ECUcore-iMX35_192.168.10.248 login: PlcAdmin
Password:
sh-3.2:~# mount -t vfat /dev/mmcblk0 /mnt/sd
sh-3.2:~# cd /mnt/sd
sh-3.2:sd# ls -la
drwxr-xr-x  2 root  root   16384 Jan  1  00:00 .
drwxr-xr-x  5 root  root    47 Apr  3  2014 ..
-rwxr-xr-x  1 root  root  19883 Jun 30  2014 PLCore-iMX35.gif
-rwxr-xr-x  1 root  root  12180 Jun 30  2014 Pc iMX35Config.html
-rwxr-xr-x  1 root  root   1151 Jun 30  2014 index.html
-rwxr-xr-x  1 root  root   4877 Jun 30  2014 systec_logo.jpg
sh-3.2:sd#
```

Figure 67: Access to the SD card at ECUcore-iMX35

**Advices for using the SD card:**

Writing accesses to the SD card take place asynchronously. The return of the "write" function does not mean that all data has been completely written onto the SD card. In fact, the data from the ECUcore-iMX35 is cached in the RAM and afterwards, it is transferred to the SD card in the background. For C/C++ programs, the writing procedure is only completed after calling "close". For Shell scripts use command "sync".



## 10 Tips & Tricks for Handling Linux

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

- **Calling programs (search path)**

On the contrary 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. 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 (*"eXecutable"*) which are shown if *"ls -la"* is called, e.g.:

```
ls -la ./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. The user must only enter as much signs of the file or command name as necessary until only one name is left that those signs apply to. By pressing the TAB button, the remaining signs of the entry are completed automatically.

Example: It is sufficient to enter *"/m"* to call the Shell script *"./mountnfs.sh"* in directory *"/home"* of the ECUcore-iMX35. By pressing the TAB button, the shell automatically completes the signs to the full command *"./mountnfs.sh"*.

- **Displaying environment variables**

Command *"echo"* is necessary to display environment variables. For example, command *"echo \$PATH"* would display the current configured search path.

- **Setting environment variables**

When setting an environment variable, consider that it is only visible within the current shell instance or within the sub-shells called by the current shell. If for example, an environment variable is defined in a Shell script, it will no longer be valid after the Shell script is executed. The reason for this is that for executing a script a new shell instance is called that finishes the script. The respective environment variable is accessible within this shell instance (and for this reason within the script being executed in it). After finishing the script, the shell instance that was started to execute the script is closed. Consequently, also the environment variable that was defined for this shell instance is no longer valid.

One possibility to set environment variables (e.g. "TZ=") permanently is to define it in script *"/etc/profile.environment"*. This script is started once when starting the first shell instance during the boot process. This implies that all variables defined in that shell instance remain valid during the entire runtime and they will be "passed on" to all other sub-shells started afterwards.

- **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.

- **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 sectors of a conditional execution (*"if"*, *"case"*, *"while"* etc.) really have been executed.

- **Setting the time zone for the ECUcore-iMX35**

Setting the time zone for the ECUcore-iMX35 takes place by defining the environment variable *"TZ"* in start script *"/etc/profile.environment"*. 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.

## Appendix A: GNU GENERAL PUBLIC LICENSE

The Embedded Linux used on the ECUcore-iMX35 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/fstab .....	27
/home .....	27, 28
/home/etc/autostart .....	17
/home/etc/diskadded.sh .....	78
/home/etc/diskremoved.sh .....	78
/home/etc/hotplug.sh .....	78
/home/etc/pointercal .....	34
/projects/ecucore-setup .....	53
/tmp .....	27
/var/ .....	27
/var/log/hotplug.log .....	79
<b>I</b>	
\Vm-xubuntu .....	44
<b>A</b>	
Accessory .....	13
adduser .....	24
Administration	
System requirements .....	14
autostart .....	17
Autostart .....	16
<b>B</b>	
Boot configuration .....	17
brightness control for display .....	33
<b>C</b>	
CAN driver .....	56
CAN0 .....	11
CAN1 .....	11
candrv .....	56
chmod .....	60, 81
COM0 .....	11
COM1 .....	11
COM2 .....	11
Configuration	
Commands .....	18
Readout and display .....	26
Configuration mode .....	15
Connection network drive .....	44
<b>D</b>	
date .....	25
Debugger commands .....	71
deluser .....	24
Demo project .....	62
CAN driver .....	57
I/O Driver .....	56
Development Board	
Connections .....	11
Control elements .....	12
Development Kit .....	10
df .....	28
DHCP .....	41
ECUcore-iMX35 .....	19
Dimensions .....	8
Display .....	31
brightness control .....	33
Driver Development Kit .....	13, 75
<b>E</b>	
Eclipse .....	64
Configuring Debugger .....	68
Debugger commands .....	71
Debugging .....	66
Open project .....	65
Translating project .....	66
ecucore-setup .....	53
Embedded Linux .....	9
EMC law .....	5
Environment variables .....	53
Display .....	81
Setting .....	82
ETH0 .....	11
evtest .....	32
Execution rights .....	81
<b>F</b>	
File system .....	27
Files	
preinstalled .....	28
fstab .....	27
FTP .....	60
Login to ECUcore-iMX35 .....	22
FTP client .....	14
FTP server .....	62
ftpget .....	61
ftpput .....	61
fw_printenv .....	26
<b>G</b>	
gdbserver .....	67
GNU .....	9
<b>H</b>	
hellocan .....	57
HTTP server .....	29
hwclock .....	25
<b>I</b>	
ifconfig .....	43
Input Devices .....	31
insmod .....	55, 57
iodrvdemo .....	76
<b>L</b>	
LCD	
Brightness control .....	33
lighttpd .....	29



Linux .....9  
 Linux VMware-Image.....41  
 LinuxBSP .....73

**M**

make .....62  
 Makefile .....53  
 Manuals  
     Overview .....6  
 Matrix Keyboard .....31, 33  
 mount.....59

**N**

net use .....44  
 Network environment.....43  
 NFS  
     mount .....59

**P**

PATH .....81  
 Predefined user accounts  
     (development system).....42  
     ECUcore-iMX35 .....23  
 ptxdist kernelconfig .....73  
 ptxdist menuconfig.....73  
 pureftpd.....22, 62

**Q**

Qt .....34  
     Environment variables .....35  
     Installation .....35  
     Overview of Components.....34  
     Program call .....35  
 qtdemo .....34, 35  
 -qws .....35

**R**

root.sum.jffs2 .....36

**S**

Scrollwheel .....31, 33  
 Search path .....81  
 Setting RTC .....25  
 Setting system time .....25  
 setup-ecucore-imx35.sh .....28  
 SO-1119 .....75  
 SO-1121.exe.....40  
 Software structure .....52  
 Software update  
     Linux-Image .....36  
     U-Boot .....39  
 System start.....16

**T**

TAB completion..... 81  
 Telnet  
     Login to ECUcore-iMX35 ..... 21  
     Login to Linux development system ..... 45  
 Telnet client..... 14  
 Terminal program..... 14  
 Terminal settings..... 16  
 Testing Hardware Connections ..... 76  
 TFTP32 ..... 36  
 Time zone ..... 82  
 Toolchain ..... 53  
 Touchscreen ..... 31  
     Calibration ..... 33  
 TZ..... 82

**U**

U-Boot ..... 17  
 U-Boot command prompt  
     Activation ..... 15  
 U-Boot command prompt  
     Terminal settings ..... 16  
 U-Boot commands  
     Ethernet configuration ..... 18  
     Update Linux-Image ..... 37  
 USB-RS232 Adapter Cable ..... 13  
 User accounts  
     Addition and deletion..... 24  
     Changing password..... 24  
     predefined (development system)..... 42  
     Predefined (ECUcore-iMX35) ..... 23

**V**

VMware-Image  
     Determining IP address..... 43  
     Determining static IP address ..... 48  
     Installation ..... 40  
     Keyboard layout ..... 45  
     Overview ..... 40  
     Shrinking ..... 51  
     Start ..... 41  
     System update ..... 50  
     Time zone..... 47  
 VMware-Player  
     Installation ..... 40

**W**

Windows network environment..... 43  
 WinSCP ..... 22



**Document:** System Manual ECUcore-iMX35  
**Document number:** L-1569e\_1, 1<sup>st</sup> Edition June 2014

---

**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:**



详情请通过[sales@hkaco.com](mailto:sales@hkaco.com)联系我们

北京 : 010-5781 5068

上海 : 021-6728 3703

广州 : 020-3874 3032

西安 : 029-8187 3816