

System Manual
ECUcore-E660

User Manual
Version 1.0

Edition May 2014

Document No.: L-1554e_01

SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund
Phone: +49 3765 38600-0 Fax: +49 3765 38600-4100
Web: <http://www.systec-electronic.com> Mail: info@systec-electronic.com

Status/Changes

Status: released

Date/Version	Section	Changes	Editor
2014/05/20 1.0	All	Creation	Dr. Norbert Prang

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:

Contact	Direct	Your local distributor
Address:	SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund GERMANY	Please find a list of our distributors under: http://www.systemec-electronic.com/distributors
Ordering Information:	+49 (0) 3765 38600-0 info@systemec-electronic.com	
Technical Support:	+49 (0) 3765 38600-0 support@systemec-electronic.com	
Fax:	+49 (0) 3765 38600-4100	
Web Site:	http://www.systemec-electronic.com	

1st Edition May 2014

Table of Contents

1	Introduction	8
2	Overview / Where to find what?.....	9
3	Product Description	11
4	Development Kit ECUcore-E660.....	13
4.1	Overview.....	13
4.2	Electric commissioning of the Development Kit ECUcore-E660	14
4.3	Jumper configuration of the Development Kit	16
4.4	Control elements of the Development Kit ECUcore-E660	18
4.5	Optional accessory	19
4.5.1	USB-RS232 Adapter Cable	19
4.5.2	Driver Development Kit (DDK).....	19
5	Application and Administration of the ECUcore-E660.....	20
5.1	System requirements and necessary software tools.....	20
5.2	System start of the ECUcore-E660	22
5.2.1	Boot-Up sequence	22
5.2.2	Reset conditions of the ECUcore-E660 Development Board.....	22
5.2.3	Autostart for user software.....	23
5.3	UEFI Bootloader and EFI Shell	24
5.3.1	UEFI-based Bootloader for ECUcore-E660.....	24
5.3.1.1	Features	24
5.3.1.2	Default boot order.....	24
5.3.1.3	How to create a boot option	25
5.3.1.4	How to read out the system identification	25
5.3.2	Activation of EFI Shell.....	26
5.3.3	Requirements and settings for the communication with the EFI Shell	29
5.4	Ethernet configuration of the ECUcore-E660	29
5.5	Login to the ECUcore-E660.....	31
5.5.1	Login to the command shell.....	31
5.5.2	Login to the FTP server	33
5.6	Predefined user accounts.....	35
5.7	Adding and deleting user accounts	35
5.8	How to change the password for user accounts	36
5.9	Setting the system time	36
5.10	Readout and displaying EFI bootloader configuration data	37
5.11	Showing the installed Linux-Version	38
5.12	File system of the ECUcore-E660	38
5.13	Preinstalled files in the directory "/home"	40
5.14	Using the HTTP server	41
5.15	HMI Components.....	43
5.15.1	Supported HMI Devices.....	43
5.15.2	Connections of Keyboard, Mouse and Display.....	46
5.16	The Use of Qt on the ECUcore-E660.....	47
5.16.1	Overview of the Qt-Components for the ECUcore-E660.....	47
5.16.2	Preinstalled Qt-Libraries on the ECUcore-E660.....	47
5.16.3	Qt-Environment Variables.....	47
5.16.3.1	Automatic Configuration	47
5.16.3.2	Manual Configuration	47
5.16.4	Starting Qt-Programs.....	48
5.17	Updating the Linux-Image	50
6	VMware-Image with Linux Development System	51
6.1	Overview.....	51

6.2	Installing the Linux VMware-Image	51
6.3	Starting the Linux VMware-Image	51
6.4	User accounts to log in to the Linux development system	53
6.5	Determining the IP address of the Linux development system	54
6.6	Access to the Linux development system from a Windows computer	54
6.6.1	Access via Windows network environment	54
6.6.2	Access via Telnet client	55
6.7	Personal configuration and actualization of the Linux VMware-Image	56
6.7.1	Adjustment of keyboard layout and time zone	56
6.7.2	Adjusting the desktop size	58
6.7.3	Setting a static IP address for the Linux VMware-Image	59
6.7.4	System update of the Linux VMware-Image.....	61
6.7.5	Changing the computer name in the Windows network environment	61
6.7.6	Shrinking the VMware-Image	61
6.8	How to write the SD card image to a SD card	62
6.9	How to write the SD card image to the internal eMMC	62
6.10	How to read out the BSP version from the target system	62
7	Software Development for the ECUcore-E660.....	63
7.1	Software structure of the ECUcore-E660	63
7.2	Makefile and environment variables to create projects	64
7.3	I/O Driver for the ECUcore-E660.....	65
7.3.1	Integration of the I/O Driver into own user projects	65
7.3.2	I/O Driver Demo project.....	67
7.4	CAN Driver for the ECUcore-E660.....	67
7.4.1	Preinstalled CAN Driver in the Linux image	67
7.4.2	CAN Driver Demo project	67
7.5	Transferring programs to the ECUcore-E660.....	69
7.5.1	Using NFS.....	70
7.5.2	Using FTP	71
7.5.2.1	ECUcore-E660 as FTP client	71
7.5.2.2	ECUcore-E660 as FTP server.....	72
7.6	Compilation and execution of demo project "demo"	73
7.6.1	Usage of "make"	73
7.6.2	Using graphical IDE "Eclipse"	75
7.6.2.1	How to open and edit the demo project	76
7.6.2.2	Translating the demo project.....	77
7.6.2.3	Debugging the demo project in the IDE	78
7.7	Configuration and Creation or Update of the ECUcore E660 Linux-Image	85
8	Adaptation and Testing of the hardware connections.....	87
8.1	Driver Development Kit (DDK) for the ECUcore-E660.....	87
8.2	Testing the hardware connections	88
9	Using the USB interface.....	90
9.1	Using the USB interface	90
10	Tips & Tricks for Handling Linux.....	94
	Appendix A: GNU GENERAL PUBLIC LICENSE.....	96
	Index.....	101

List of Tables

Table 1: Overview of relevant manuals for the ECUcore-E660	9
Table 2: Connections of the Development Kit ECUcore-E660.....	14
Table 3: Jumpers of the Development Board for the ECUcore-E660	16
Table 4: Control elements of the Development Board for the ECUcore-E660.....	18
Table 5: Reset management	23
Table 6: Sysfs attributes under /sys/class/dmi/id/	25
Table 7: EFI commands	27
Table 8: EFI Shell - Ethernet configuration commands.....	30
Table 9: Predefined user accounts of the ECUcore-E660	35
Table 10: File system configuration of the ECUcore-E660	39
Table 11: Input Devices of ECUcore-E660	44
Table 12: Possible connections of HMI devices to the Development Board.....	46
Table 13: Qt-Components for the ECUcore-E660.....	47
Table 14: Predefined user accounts of the Linux development system.....	53
Table 15: Overview of most important Debugger commands	82

List of Figures

Figure 1: Top view of the ECUcore-E660.....	11
Figure 2: Development Board ECUcore-E660	13
Figure 3: Positioning of most important connections on the Development Board for the ECUcore-E660	16
Figure 4: SYS TEC USB-RS232 Adapter Cable	19
Figure 5: Configuration of the serial port for Tera Term	21
Figure 6: Memory test during Boot-Up	22
Figure 7: System start of the ECUcore-E660	23
Figure 8: UEFI boot menu	26
Figure 9: EFI Shell started.....	27
Figure 10: Ethernet configuration of the ECUcore-E660 by EFI Shell commands	31
Figure 11: Calling the Telnet client in Windows	32
Figure 12: Login to the ECUcore-E660	32
Figure 13: Starting the FTP server	33
Figure 14: Login settings for WinSCP	34
Figure 15: FTP client for Windows "WinSCP"	34
Figure 16: Adding a new user account.....	35
Figure 17: Changing the password for a user account	36
Figure 18: Setting and displaying the system time.....	37
Figure 19: Displaying EFI bootloader configuration data under Linux using "fw_printenv"	38
Figure 20: Showing the installed Linux-Version	38
Figure 21: Display of information about the file system	40
Figure 22: Starting the HTTP server "lighttpd"	42
Figure 23: Display of HTML pages for the ECUcore-E660 in the WEB-Browser.....	42
Figure 24: How to determine the device filename of input devices.....	45
Figure 25: Diagnosis of Input Devices by means of "evtest".....	45
Figure 26: Connections for Keyboard, Mouse and Display to the Development Board.....	46
Figure 27: Output of "qtdemo" on the ECUcore-E660.....	48
Figure 28: Program window of the VMware Player using the "Open" symbol	52
Figure 29: VMware selection dialog to generate or remain the MAC address.....	52
Figure 30: Desktop of the Linux development system	53
Figure 31: Determining the IP address of the Linux development system.....	54
Figure 32: Linux development system in Windows network environment	54
Figure 33: Login to "Vm-xubuntu".....	55
Figure 34: Access to the Linux development system via Telnet client.....	56
Figure 35: Country symbol for switching keyboard layouts.....	56
Figure 36: Choosing a permanent keyboard layout	57
Figure 37: Calling "Xfce Settings Manager" from the start menu	57
Figure 38: Adding keyboard layouts in the "Xfce Settings Manager"	58
Figure 39: Adjusting the time zone	58
Figure 40: "Network Manager" for the configuration of the Ethernet interface.....	59
Figure 41: Adding a network connection	60
Figure 42: Configuration of the network connection.....	60
Figure 43: Changing the network configuration in the "Network Manager".....	61
Figure 44: Version output over Tera Term terminal console with option "-a".....	62
Figure 45: Structure of directory "/projects" in the Linux development system	63
Figure 46: Structure of the I/O Driver for the ECUcore-E660.....	65
Figure 47: Execution of Demo project "hellocan" on the ECUcore-E660.....	68
Figure 48: CAN analysis tool "CAN-REport"	69
Figure 49: Mounting the directory "/tftpboot/" of the VM into the local file system of the ECUcore-E660	71
Figure 50: Download and upload via FTP	72
Figure 51: Translating the demo project in the VMware Linux development system.....	73
Figure 52: Executing the demo project "demo" on the ECUcore-E660.....	75
Figure 53: Eclipse dialog "Workspace Launcher"	76
Figure 54: The graphical IDE "Eclipse"	76
Figure 55: Translating the demo project in Eclipse	77
Figure 56: Starting the Debug server on the ECUcore-E660.....	79

Figure 57: Determining the application that is to be debugged 80
Figure 58: Selecting the GDB Debugger 81
Figure 59: Configuring the connection to the Target 82
Figure 60: Debugging the Demo project in Eclipse 83
Figure 61: Terminal outputs of the ECUcore-E660 during debugging 84
Figure 62: User surface for the configuration of the Linux kernel 85
Figure 63: User interface to configure user applications including BusyBox 86
Figure 64: Overview of the Driver Development Kit for the ECUcore-E660 87
Figure 65: Testing the hardware connections using "iodrvdemo" 89
Figure 66: Internal procedures for plugging or removing USB memory sticks 90
Figure 67: Output of messages redirected into file "/var/log/hotplug.log" 92

1 Introduction

Thank you that you have decided for the SYS TEC ECUcore-E660. 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-E660. It will assist you in getting familiar with the functional range and usage of the ECUcore-E660. 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-E660. Please also refer to those complementary documents.

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

Declaration of Electro Magnetic Conformity for ECUcore-E660 (EMC law)



The ECUcore-E660 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-E660 is ESD-sensitive and may only be unpacked, used and operated by trained personal at ESD-conform work stations.

The ECUcore-E660 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-E660 based on the Development Kit ECUcore-E660 as well as general procedures for software development of this module. There are different hardware manuals for all hardware components such as the ECUcore-E660, development boards and reference circuitry. Software-sided, the ECUcore-E660 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-E660. The VMware-Image can be used unmodified within different host systems. Table 1 lists up all relevant manuals for the ECUcore-E660.

Table 1: Overview of relevant manuals for the ECUcore-E660

Information about...	In which manual?
Basic information about the ECUcore-E660 (configuration, administration, connection assignment, software development, reference designs et cetera.)	In this manual
Application of the ECUcore/PLCcore-E660 as PLC (Programming of the PLCcore-E660 as PLC according to IEC 61131-3, Process Image, Data exchange via Shared Process Image with external applications et cetera)	System Manual PLCcore-E660 (Manual no.: L-1555)
Hardware description for the ECUcore-E660, reference designs et cetera	Hardware Manual ECUcore-E660 (Manual no.: L-1562)
Development Board for the ECUcore-E660, reference designs et cetera	Hardware Manual Development Board E660 (Manual no.: L-1563)
Driver Development Kit (DDK) for the ECUcore-E660	Software Manual Driver Development Kit (DDK) for the ECUcore-E660 (Manual no.: L-1561)
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"> • Advanced Programming in the UNIX Environment, Stevens Rago, Addison-Wesley • GNU Function List: http://www.silicontao.com/ProgrammingGuide/GNU_function_list

- Section 4** of this manual describes the **electrical commissioning** of the ECUcore-E660 on the basis of the Development Kit ECUcore-E660.
- Section 5** exemplifies **details about the usage of the ECUcore-E660**, 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-E660 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-E660 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, USB and Ethernet interfaces, the ECUcore-E660 is best suitable to realize custom specific HMI (Human Machine Interface) applications.

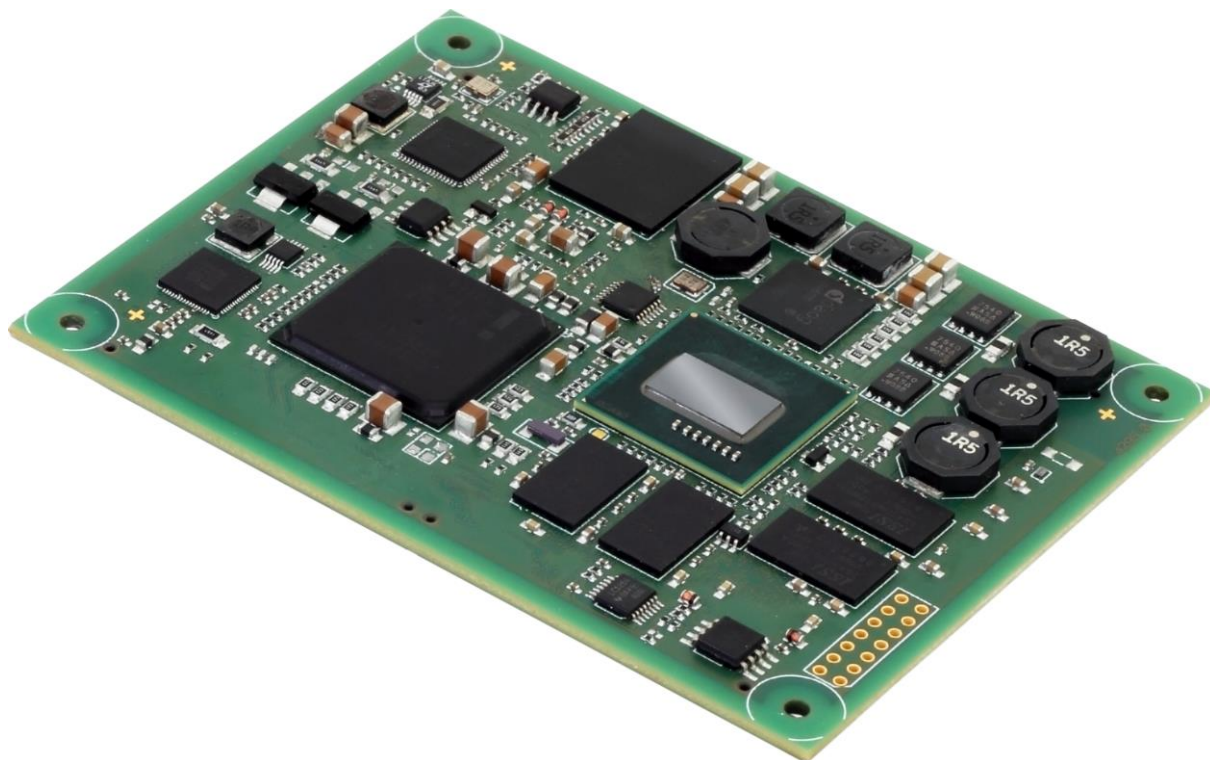


Figure 1: Top view of the ECUcore-E660

These are some significant features of the ECUcore-E660:

- High-performance CPU kernel (Intel Atom E660 1.3 GHz CPU, CPU clock, 1.0 GHz)
- 1 GByte SDRAM Memory, 4GByte FLASH Memory
- 1 MiB dedicated Video RAM
- LVDS LCD Controller supports up to 800x600 pixel resolution with 16-bit color depth
- 4 USB Host interfaces
- 2 USB Device interfaces
- Possibility for connecting HMI devices such as USB Keyboard, USB Mouse and DVI Graphic Cards
- 2x 10/100/1000 Mbps Ethernet LAN interface
- 2x CAN 2.0B interface, usable as CANopen Manager
- 4x asynchronous serial ports (UART)
- On the Development Board:
8 digital inputs, 9 digital outputs
- On the core module:
9 digital in/ouputs, 9 digital outputs (I/Os modifiable as requested by the user)
- Externally usable SPI and I²C
- On-board peripherals: RTC, temperature sensor, watchdog, SDC
- Operating system: Linux

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-E660 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.

The default I/O configuration can be adapted for specific application requirements by using the Driver Development Kit (SO-1117). Saving the user application in the on-board Flash-Disk of the module allows an automatic restart in case of power breakdown.

Das ECUcore-E660 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-E660 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-1116). If you require the LinuxBSP sources independently from the VMware-Image of the Linux development system, please contact our support:

support@systemec-electronic.com

4 Development Kit ECUcore-E660

4.1 Overview

Due to the Development Board contained in the Kit, the Development Kit ECUcore-E660 allows for a quick commissioning of the ECUcore-E660 and simplifies the design of prototypes for user-specific applications that are based on this module. Among other equipment, the Development Board features are the possibility of connecting HMI devices, 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-E660 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-E660.

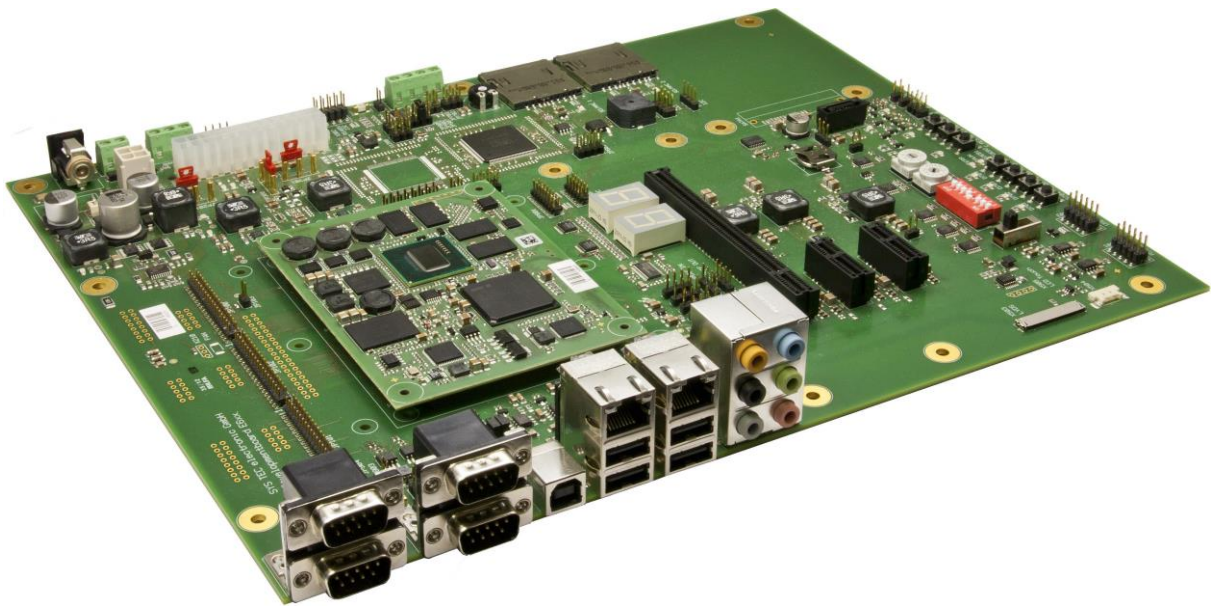


Figure 2: Development Board ECUcore-E660

The Development Kit ECUcore-E660 ensures quick and problem-free commissioning of the ECUcore-E660. Therefore, it combines all hard- and software components that are necessary to create own applications: the core module ECUcore-E660 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-E660.

The Development Kit ECUcore-E660 contains the following components:

- ECUcore-E660
- Development Board for the ECUcore-E660
- 24V Power adapter
- Ethernet cable
- RS232 cable
- DVD with Linux development system, examples, documentation and other tools (SO-1116)

The Linux development system included in the Kit serves as software development platform and as debug environment for the ECUcore-E660. 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-E660

The power adapter necessary for running the Development Kit ECUcore-E660 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 (X200/X201), UART1 (P901 Top) and ETH0 (X1001). Table 2 provides an overview over the connections of the Development Kit ECUcore-E660.

Table 2: Connections of the Development Kit ECUcore-E660

Connection	Label on the Development Board	Remark
Main power supply	X200 or X201 + X217	Connection for a power supply with the following parameters: 24 VDC, 2 – 4 A (depending on used peripherals). X217 provides additional front panel LED connection.
Alternative power supply	X208 and X209	Connection for a power supply with ATX2.2 standard interface. Alternative use need to be selected with jumper JP201, JP202, JP300 and JP301
PCIe0 (PCIexpress)	X400	Connection for a PCIexpress card Named "PCIe Slot 1" in Figure 3
PCIe1 (PCIexpress)	X401	Connection for a PCIexpress card Named "PCIe Slot 2" in Figure 3
SDVO Interface	X501	Connection for SDVO graphic card. Graphic card is included.
LCD interface	X503 and X504	Connection for a LCD display. Include LVDS (X503) and backlight (X504) connectors. Display AOU G070VW01V0 is supported.
LCD touch interface	X503	Not supported
SD card slot 1	X601	The SD card contains the Embedded Linux operating system (refer to sections 6.8 and 7.7 about how to create the SD card image).
SD card slot 2	X604	Optional SD card interface. Remark: JP600 need to be configured The SD-Card Interface is also connected to an eMMC on the ECUcore, so the external SD-Card Interface only can be used with a special variant of the ECUcore where the eMMC is not assembled!
SATA0	X600	This interface can be used in combination with a SATADOM device. Remark: JP604 need to be configured.

Connection	Label on the Development Board	Remark
SATA1	X602 + X605	This interface (X602) is used for SATA device (2,5" hard disc). X605 provides additional front panel LED connection.
LPC	X607	This interface serves LPC interface. Remark: JP601 and JP603 need to be configured.
I2C	X701	This interface can be used for user specific I2C interface. Remark: JP701 need to be configured.
SPI	X603	This interface can be used for user specific SPI interface. Remark: JP704 need to be configured.
AUDIO	X800	This interface can be used for user specific AUDIO interface. Remark: JP800 need to be configured.
CAN1 (CAN)	P900A Bottom or X900	Interface can be used freely for the user program. Remark: JP900 + JP903 + JP904 + JP909 need to be configured
UART0 (RS232)	P900B Top	Interface can be used freely for the user program. Remark: JP904 + JP909 need to be configured
UART1 (RS232)	P901B Top	This interface is used for the configuration of the unit (e.g. setting the IP-address) and for the diagnosis or debugging. It can be used freely for general operation of the user program. Remark: JP909 need to be configured
UART2 (RS232)	P901A Bottom	Interface can be used freely for the user program. Remark: JP909 need to be configured
UART3 (RS232 or RS485)	X902	Interface can be used freely for the user program. Remark: JP902 + JP905 + JP906 + JP907 need to be configured
SDC UART1 (RS232)	X901	not supported
USB0 (HOST)	X1000 Bottom	This interface serves as USB HOST interface.
USB1 (HOST)	X1000 Middle	This interface serves as USB HOST interface.
USB2 (HOST)	X1001 Bottom	This interface serves as USB HOST interface.
USB3 (HOST)	X1001 Middle	This interface serves as USB HOST interface.
USB4 (HOST)	X1003	This interface serves as USB HOST interface.
USB5 (HOST)	X1003	This interface serves as USB HOST interface.
USB (Device)	X1002	not supported
ETH0 (Ethernet)	X1001 Top	This interface serves as communication interface with the Linux development system (Programming PC) and can as well be used freely for the user program.
ETH1 (Ethernet)	X1000 Top	This 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-E660. Instead of using the 24V DC power adapter included in the Kit, the power supply may optionally take place via X200 with an external source of 24V/1A.

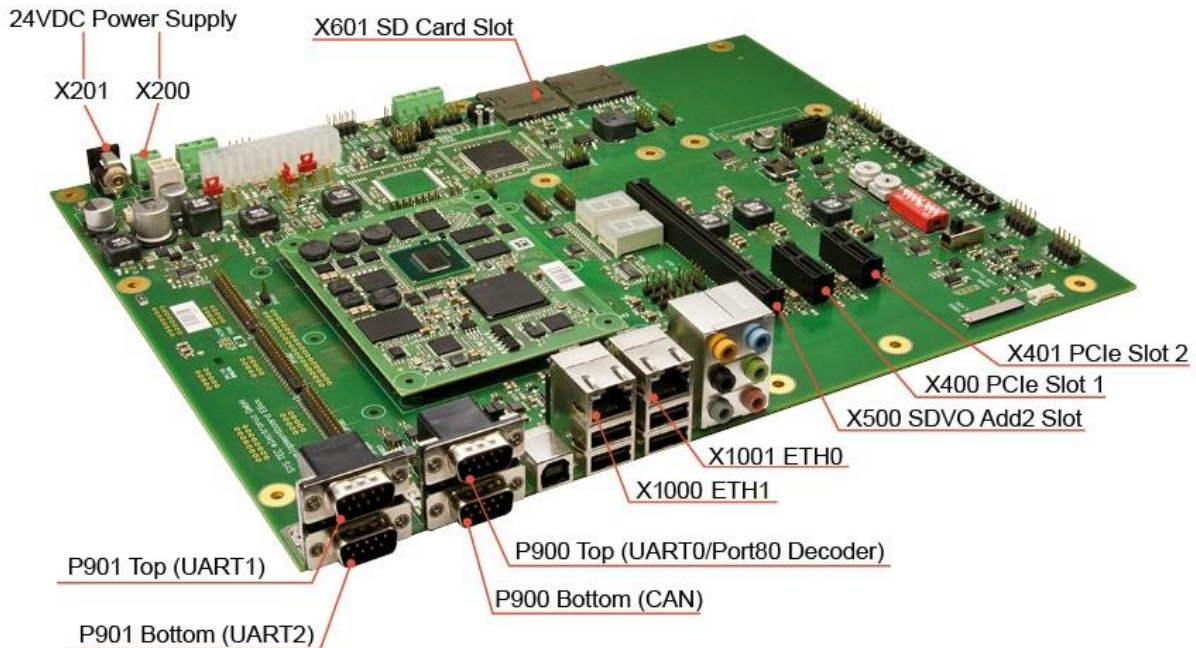


Figure 3: Positioning of most important connections on the Development Board for the ECUcore-E660

Advice: Upon commissioning, an SD card (X601) and cables for Ethernet (ETH0, X1001) and RS232 (UART1, P901 Top) must be connected prior to activating the power supply (X200 / X201).

Caution: The core module shouldn't be used without a CPU heat sink on its top. Figure 3 is not complete, but it shows the design of the core module better than a CPU heat sink hides the module below it.

4.3 Jumper configuration of the Development Kit

The Development Kit ECUcore-E660 is configured with a default jumper configuration. Table 3 lists the jumpers of the Development Board and describes their meaning and default options.

Table 3: Jumpers of the Development Board for the ECUcore-E660

Control element	Name	Default Option	Meaning
SDC_BOOT0	JP103	open	Boot option of SDC
ATX /PS_ON	JP200	closed	ATX power supply "ON"
12V power	JP201	closed (2-3)	12V power onboard regulated or delivered by ATX power supply interface (default = onboard)
Power good	JP202	closed (2-3)	Power good signal of onboard voltage regulator or onboard voltage regulator and ATX

Control element	Name	Default Option	Meaning
5V power	JP300	closed (2-3)	5V power onboard regulated or delivered by ATX power supply interface (default = onboard)
3V3 power	JP301	closed (2-3)	3V3 power onboard regulated or delivered by ATX power supply interface (default = onboard)
LCD Touch	JP500 + JP502	open	Not supported
SD card slot 2	JP600	open	SD card slot 2 available at X604 if the eMMC on the ECUcore is not assembled
LPC	JP601 + JP603	open	LPC interface available at X607
SATA0	JP604	open	Power Option for SATA0 (SATADOM) interface
Watchdog	JP700	open	Watchdog disable
I2C	JP701	open	I2C interface available at X701
ADC	JP702 + JP703	open	ADC interface is available at X700 and R721
SPI	JP704	open	SPI interface available at X603
SPI	JP705	closed	EEPROM available at SPI interface
AUDIO	JP800	closed (7-8) open (5-6) open (3-4) closed (1-2)	AUDIO interface is available at X800
CAN	JP900	open	Enable 120 ohm bus termination
CAN	JP903	open	Enable 5V output
CAN	JP909	closed	CAN is available at P900A and X900 Enable CAN Interface (EGT20H or SDC) Remark: If CAN via SDC is selected, UART0 can't be used
CAN	JP904	closed (13-14) closed (15-16)	CAN via SDC is selected Remark: If CAN via SDC is selected, UART0 can't be used
UART0	JP901	open	UART0 is available at P900B Remark: If CAN via SDC is selected, UART0 can't be used
UART1	JP904	closed (1-2) closed (7-8)	UART1 is available at P901B
UART2	JP904	closed (3-4) closed (9-10)	UART2 is available at P901A
UART3	JP902 + JP905 + JP906 + JP907	open	UART3 is available at X902. Different configuration modes (RS232 or RS485, Listen Only etc.) can be selected
SDC UART1	JP904	open (5-6) open (11-12)	SDC UART1 is available at X901 (not supported) Remark: SDC UART1 is selected, CAN via SDC can't be used

4.4 Control elements of the Development Kit ECUcore-E660

The Development Kit ECUcore-E660 allows for easy commissioning of the ECUcore-E660. It has available various control elements to configure the module and to simulate in- and outputs for the usage of the ECUcore-E660 as the main component of an industrial control. Table 4 lists the control elements of the Development Board and describes their meaning.

Table 4: Control elements of the Development Board for the ECUcore-E660

Control element	Name	Meaning
Pushbutton 0	S704	Digital Input DI0
Pushbutton 1	S705	Digital Input DI1
Pushbutton 2	S706	Digital Input DI2
Pushbutton 3	S707	Digital Input DI3
Pushbutton 4	C RES	Button for COLD RESET – POWER button
Pushbutton 5	W RES	Button for WARM RESET
Pushbutton 6	SDC RES	Button for RESET to the SDC (System Diagnostic Controller)
LED 0	D701	Digital Output DO0
LED 1	D702	Digital Output DO1
LED 2	D703	Digital Output DO2
LED 3	D704	Digital Output DO3
Potentiometer (ADC)	R721	Analog Input AI0 Potentiometer placed in neighborhood of JP702
Screw-type terminals	X700	Terminals for Analog Inputs AI1 and AI2 4 terminals in 1 block, 2 for each AI
Run/Stop Switch	3-position switch with positions named: <ul style="list-style-type: none"> • RUN • STOP • MRES 	Control element can be used freely for the user program (e.g. Run / Stop to operate the control program)
Run-LED	RUN	Control element can be used freely for the user program (e.g. Display of activity state of the control program)
Error-LED	ERR	Control element can be used freely for the user program (e.g. Display of error state of the control program)
Hex-Encoding Switches	S702/S703	Control element can be used freely for the user program (e.g. Configuration of node address CAN0) Order for the half bytes of the byte coded by this switch: <ul style="list-style-type: none"> • S703 – upper half byte • S702 – lower half byte

Control element	Name	Meaning
DIP-Switch	S700	Control element can be used freely for the user program (e.g. Configuration bitrate and master mode CAN0) Switch number 8 is the MSB of the byte coded by this switch
HEX-SWITCH	S702 or S703	Control element can be used freely for the user program (e.g. Configuration bitrate and master mode CAN0)
Beeper	L700	Control element can be used freely for the user program (e.g. activity state of the control program)
7 segment display	U1205 + U1206	Display error codes during boot-up. "00" means boot-up without faults.

4.5 Optional accessory

4.5.1 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-E660 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

4.5.2 Driver Development Kit (DDK)

The ECUcore-E660 Driver Development Kit (order number SO-1117) 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-E660

5.1 System requirements and necessary software tools

The administration of the ECUcore-E660 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 4.5.1) that provides an appropriate RS232 interface via USB port.

All examples referred to in this manual are based on an administration of the ECUcore-E660 using a Windows computer. Procedures using a Linux computer would be analogous.

To administrate the ECUcore-E660 the following software tools are necessary:

- Terminal program** A Terminal program allows the communication with the **command shell** of the ECUcore-E660 via a **serial RS232 connection to UART1 of the ECUcore-E660**. This is required for the Ethernet configuration of the ECUcore-E660 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-E660 via **Ethernet connection to ETH0 of the ECUcore-E660**. Using Telnet clients requires a completed Ethernet configuration of the ECUcore-E660 according to section 6.6.2. As alternative solution to Telnet client, all commands can be edited via a Terminal program (to UART1 of the ECUcore-E660).
- 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-E660 (ETH0) and the computer. This allows for example **editing configurations files** by transferring those from the ECUcore-E660 onto the computer where they can be edited and transferred back onto to the ECUcore-E660. Downloading files onto the ECUcore-E660 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.
- Win32DiskImager** To write the SD card image onto the SD card under Windows a free software tool named "Win32DiskImager" is available. It can be downloaded from: <https://sourceforge.net/projects/win32diskimager/> (see also section 6.8).

The settings for the serial console interface between the Windows computer and the connector for UART1 on the Development board are as follows:

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

For example, the Windows terminal "Tera Term" must be set up as shown in Figure 5.

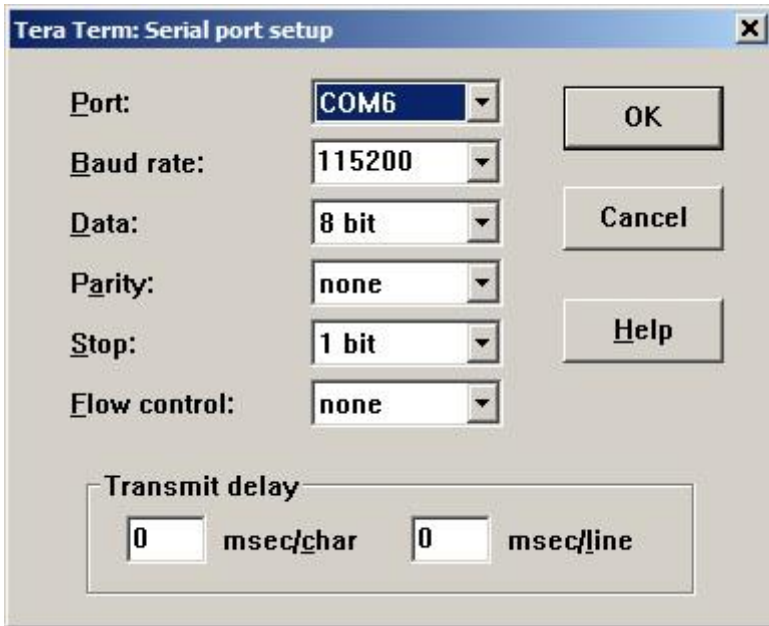


Figure 5: Configuration of the serial port for Tera Term

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-E660

5.2.1 Boot-Up sequence

In principle the Boot-Up sequence consists of 3 steps:

Step 1

After the ECUcore leaves the Reset state (conditions – see section 5.2.2 below) the UEFI Bootloader boots up. It starts with a memory test as shown in Figure 6. This memory test can be aborted manually by input of <ESC> over the console via the terminal connection.

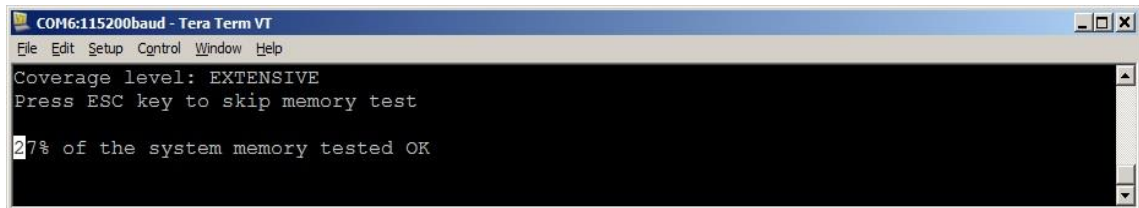


Figure 6: Memory test during Boot-Up

Step 2

Once the memory test has finished, the UEFI Bootloader waits 2 seconds for a keyboard input of a space character. If a SPACE character is received over UART1, the UEFI Bootloader shows a boot menu. The further procedure is described in section 5.3.2.

Without a Memory test abort or after leaving the UEFI boot menu by a boot command (see section 5.3.2) the operating system Linux will be booted in the boot order as configured for the UEFI Bootloader (see section 5.3.1.2). The Boot-Up sequence will be resumed with step 3.

Step 3

Once the boot procedure of the operating system Linux has finished, the user gets a login-prompt (see section 5.5.1).

Without any input over the serial console the Boot-Up sequence comes automatically to the end of step 3 as configured in the UEFI settings.

5.2.2 Reset conditions of the ECUcore-E660 Development Board

There are in principle 4 possibilities to trigger a Reset/Reboot on the ECUcore-E660 Development Board. They are described in Table 5 as follows.

Table 5: Reset management

Reset source	Explanation
Button C RES pressed for more than 4 seconds	This button functions as "Power Off". The Power-Down-Sequence on the System Diagnostic Controller (SDC) is started and the system goes down and is switched off then. In the current version the system boots up again after additional 10 seconds and performs a Cold Start.
Button W RES pressed	The system reboots with a Warm Start. The SDC doesn't initiate a Power-Down-Sequence
Command "reboot" over the console	The system reboots. The Reset cause is handled as a Cold Start.
Real Switch Off of the power	The system halts immediately. After next Power On the system reboots with a Cold Start

5.2.3 Autostart for user software

By default, the ECUCore-E660 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-E660 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 7 shows the system start in detail:

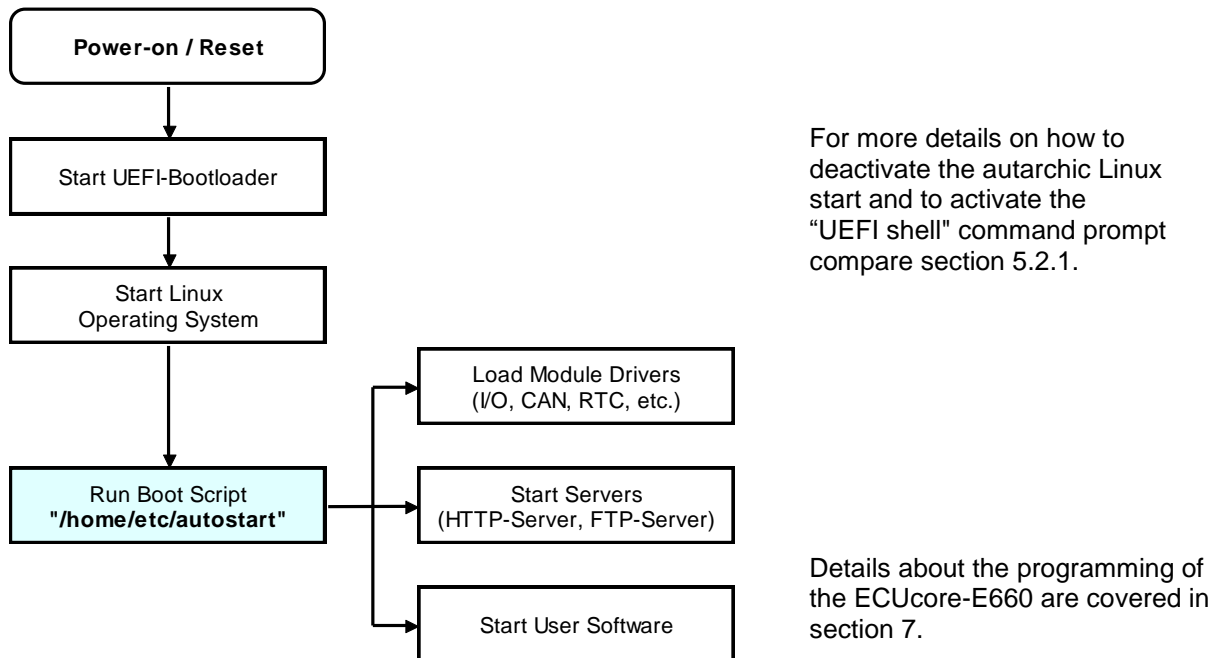


Figure 7: System start of the ECUCore-E660

It is possible to configure the ECUCore-E660 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-E660. The script can be edited directly on the ECUcore-E660 in the FTP client `"WinSCP"` (see section 5.1) using pushbuttons `"F4"` or `"F4 Edit"`.

5.3 UEFI Bootloader and EFI Shell

5.3.1 UEFI-based Bootloader for ECUcore-E660

5.3.1.1 Features

- Based on Intel Bootloader Development Kit phase II
- ACPI support including CPU frequency scaling
- Boot media:
 - o SD card, eMMC
 - o USB hard disk or thumb drive
 - o SATA disk
- Partition layouts on boot media:
 - o GUID Partition Table (GPT)
 - o Master Boot Record (MBR)
- Supported file systems on boot media
 - o FAT16, FAT32: reading and writing
 - o ext2: reading only with limited support for ext3; the EFI driver is licensed under GPL
- Integrated full UEFI Shell (see http://www.intel.com/intelpress/sum_eshl.htm)

Please refer to http://www.intel.com/intelpress/sum_efi2.htm or other introductions to UEFI compatible firmware for further information.

5.3.1.2 Default boot order

When no boot options exist or booting of all existing boot options failed, the bootloader tries to run `\EFI\BOOT\BOOTIA32.EFI` from any media in the following order.

1. USB
2. SATA
3. SD card
4. eMMC
5. EFI Shell

Note: It is recommended to always create a boot option for the intended boot media for two reasons:

1. To assure that the configured boot media is always used irrespective of any attached USB thumb drive for example
2. To increase the boot speed. To process the default boot order the bootloader has to initialize all peripheral devices. To boot a configured boot option the bootloader has to initialize only those peripheral devices necessary to process this boot option.

5.3.1.3 How to create a boot option

The Linux program *efibootmgr* can be used to create, modify and delete boot options of the UEFI boot manager.

The following example creates (option “-c”) a boot option named “Linux” (option “-L”) for the SD card on device */dev/disk/by-path/pci-0000:02:04.0* (option “-d”) partition 1 (option “-p”), where the Linux kernel is *lefi\boot\bootia32.efi* (option “-l”). The option “-w” assures that a unique signature is assigned to the partition in the MBR. The signature of all partitions on the SD card image created by Ptxdist is zero and therefore a priori identical on all SD cards. The UEFI boot manager identifies the partition on the available boot media by means of the signature. This means it is crucial to personalize the media while creating a boot option.

```
$ efibootmgr -c -d /dev/disk/by-path/pci-0000:02:04.0 -w -l
\\efi\boot\bootia32.efi -p 1 -L 'Linux'
```

Additionally, there is the script *create_bootentry*, which creates a boot option for the currently booted system, if there doesn't exist any entries yet. This script determines the disk partition which is booted from and commands *efibootmgr* to create an appropriate boot option.

create_bootentry also additionally has options for manual use. Parameter *--overwrite* (or *-o*) deletes all existing boot entries and the boot order and creates an appropriate entry for the current boot media. Parameter *--debug* (or *-d*) can be used to create a debug entry additionally which sets up a console for kernel message on serial line before any UART driver is loaded. The debug boot item also let the kernel write debug information to the console by default.

5.3.1.4 How to read out the system identification

The UEFI-based bootloader provides SMBIOS tables to the operating system. These SMBIOS tables are data structures which describe the computer in terms of

- serial numbers (system, chassis, board),
- manufacturers, product names, version,
- boards, slots, connectors (internal, external) and
- system, CPU, memory, devices, chassis.

Most of this information is stored in the SPI boot flash on the ECUcore-E660 and initially set during the factory test.

The Linux kernel presents basic information of the SMBIOS tables in the sysfs under */sys/class/dmi/id/* as attributes. Those attributes can be read by the *cat* command or any other file operation like a normal file.

Table 6 explains some of the available attributes.

Table 6: Sysfs attributes under */sys/class/dmi/id/*

Attribute	Description
product_name	Name of the product
product_serial	Serial number of the product
bios_version	Version of the bootloader

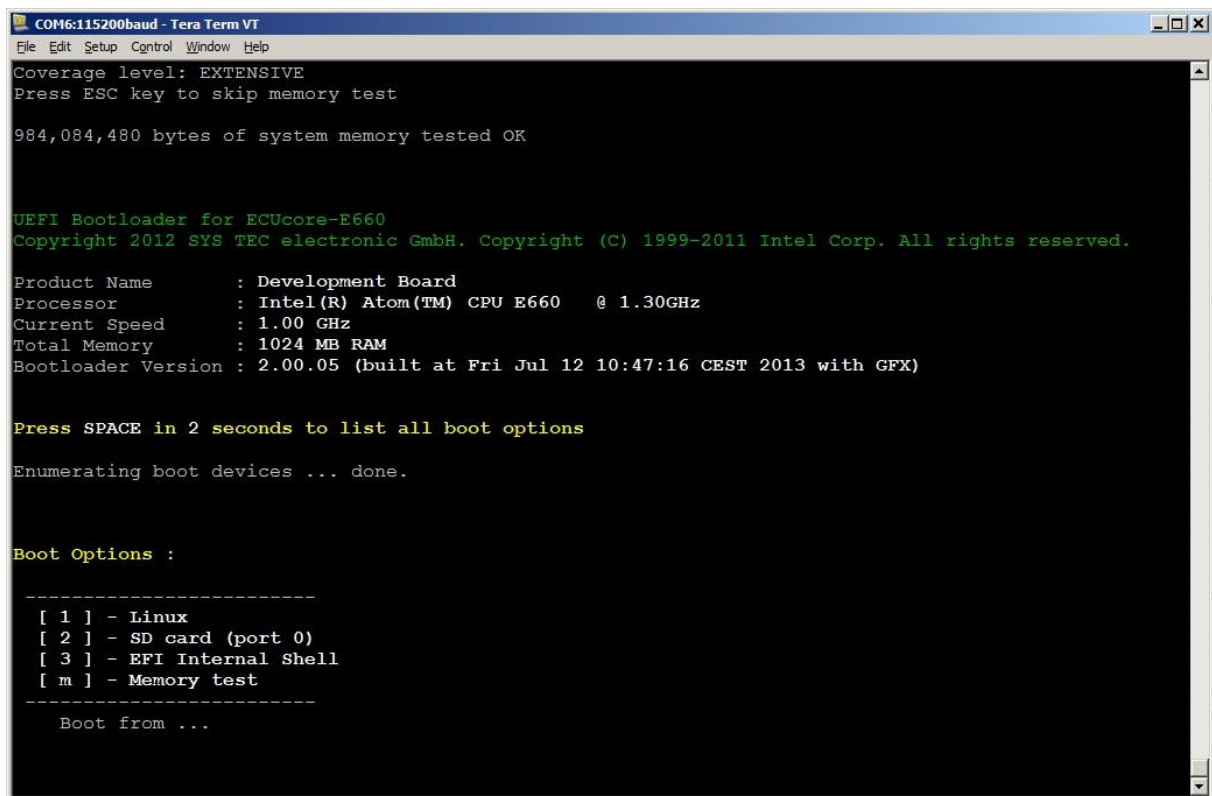
The entire SMBIOS tables can be shown with the Linux program *dmidecode*. This program can be included into the BSP optionally.

Note: If the directory */sys/class/dmi/id/* is not available, the bootloader is not UEFI-based. That means a bootloader version less than 2.00.00 is installed.

5.3.2 Activation of EFI Shell

During standard operation mode, the UEFI bootloader automatically starts the Linux operating system of the module after Reset (or Power-on). Afterwards, the operating system loads all further software components. For service purposes, such as configuring the Ethernet interface (see section 5.4), it is necessary to start the embedded EFI Shell of the UEFI bootloader on the serial interface UART1 of the ECUcore-E660.

The automatic boot of Linux operating system can be interrupted by pressing the SPACE key, when the UEFI bootloader prints "Press SPACE in 2 seconds to list all boot options" on the serial interface UART1. Then the bootloader prints the boot menu (see *Figure 8*). By pressing the key '3' the EFI Shell is started (see *Figure 9*).



```
COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help
Coverage level: EXTENSIVE
Press ESC key to skip memory test

984,084,480 bytes of system memory tested OK

UEFI Bootloader for ECUcore-E660
Copyright 2012 SYS TEC electronic GmbH. Copyright (C) 1999-2011 Intel Corp. All rights reserved.

Product Name      : Development Board
Processor         : Intel(R) Atom(TM) CPU E660 @ 1.30GHz
Current Speed    : 1.00 GHz
Total Memory     : 1024 MB RAM
Bootloader Version : 2.00.05 (built at Fri Jul 12 10:47:16 CEST 2013 with GFX)

Press SPACE in 2 seconds to list all boot options

Enumerating boot devices ... done.

Boot Options :

-----
[ 1 ] - Linux
[ 2 ] - SD card (port 0)
[ 3 ] - EFI Internal Shell
[ m ] - Memory test
-----

Boot from ...
```

Figure 8: UEFI boot menu

```

COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help
UEFI Interactive Shell v2.0. UEFI v2.30 (SYS TEC, 0x00004E20).
Mapping table
  FS0: Alias(s):HD16b;BLK1:
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)/HD(1,MBR,0x1541B300,0x12C,0x40001)
  FS1: Alias(s):HD17b;BLK3:
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x1)/HD(1,MBR,0x00000000,0x3F,0x3C4F82)
  BLK0: Alias(s):
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)
  BLK2: Alias(s):
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x1)

Press ESC in 0 seconds to skip startup.nsh or any other key to continue.
2.0 Shell>

```

Figure 9: EFI Shell started

Once the EFI Shell is active, a set of commands shown in Table 7 can be performed.

Table 7: EFI commands

EFI command	Action performed by the command
attrib	Displays or changes the attributes of files or directories.
cd	Displays or changes the current directory.
cp	Copies one or more source files or directories to a destination.
load	Loads a UEFI driver into memory.
map	Defines a mapping between a user-defined name and a device handle.
mkdir	Creates one or more new directories.
mv	Moves one or more files to a destination within a file system.
parse	Command used to retrieve a value from a particular record which was output in a standard formatted output.
reset	Resets the system.
set	Displays, changes or deletes a UEFI Shell environment variables.
ls	Lists a directory's contents or file information.
rm	Deletes one or more files or directories.
vol	Displays the volume information for the file system that is specified by fs.
date	Displays and sets the current date for the system.
time	Displays or sets the current time for the system.
timezone	Displays or sets time zone information.
stall	Stalls the operation for a specified number of microseconds.
for	Starts a loop based on for syntax.

EFI command	Action performed by the command
goto	Moves around the point of execution in a script.
if	Controls which script commands will be executed based on provided conditional expressions.
shift	Moves all in-script parameters down 1 number (allows access over 10).
exit	Exits the UEFI Shell or the current script.
else	Else identifies the portion of the code executed if the if was FALSE.
endif	Ends the block of a script controlled by an 'if' statement.
endfor	Ends a 'for' loop.
type	Sends the contents of a file to the standard output device.
touch	Updates the time and date on a file to the current time and date.
ver	Displays the version information for the UEFI Shell and the underlying UEFI firmware.
alias	Displays, creates, or deletes aliases in the UEFI Shell environment.
cls	Clears the standard output and optionally changes the background color.
echo	Controls whether script commands are displayed as they are read from the script file, and prints the given message to the display.
pause	Pauses a script and waits for an operator to press a key.
getmtc	Gets the MTC from BootServices and displays it.
help	Displays the list of commands that are built into the UEFI Shell.
connect	Binds a driver to a specific device and starts the driver.
devices	Displays the list of devices managed by UEFI drivers.
openinfo	Displays the protocols and agents associated with a handle.
disconnect	Disconnects one or more drivers from the specified devices.
reconnect	Reconnects drivers to the specific device.
unload	Unloads a driver image that was already loaded.
drvdiag	Invokes the Driver Diagnostics Protocol.
dh	Displays the device handles in the UEFI environment.
drivers	Displays a list of information for drivers that follow the UEFI Driver Model in the UEFI environment.
devtree	Displays the tree of devices compliant with the UEFI Driver Model.
drvcfg	Configures the driver using the platform's underlying configuration infrastructure.
bcfg	Manages the boot and driver options that are stored in NVRAM.
setsize	Adjusts the size of a file.
comp	Compares the contents of two files on a byte for byte basis.
mode	Displays or changes the console output device mode.
memmap	Displays the memory map maintained by the EFI environment.
eficompress	Compress a file using EFI Compression Algorithm.
efidecompress	Decompress a file using UEFI Decompression Algorithm.
dmem	Displays the contents of system or device memory.
loadpcirom	Loads a UEFI driver from a file in the format of a PCI Option ROM.
mm	Displays or modifies MEM/MMIO/IO/PCI/PCIE address space.
setvar	Changes the value of a UEFI variable.

EFI command	Action performed by the command
sermode	Sets serial port attributes.
pci	Displays PCI device list or PCI function configuration space.
smbiosview	Displays SMBIOS information.
dmpstore	Manages all UEFI NVRAM variables.
dblk	Displays the contents of one or more blocks from a block device.
edit	Full screen editor for ASCII or UCS-2 files.
hexedit	Full screen hex editor for files, block devices, or memory.
ping	Ping the target host with IPv4 stack.
ifconfig	Modify the default IP address of the UEFI IP4 Network Stack.

A help text regarding the usage of the EFI internal commands in Table 7 is printed out over the terminal in case of starting the command as follows:

```
"<EFI command> -? -v"
```

In addition to the EFI internal commands in Table 7 the EFI Shell can perform external commands (scripts designed by SYS TEC) which are placed in the directory *"/efi/tools/"*. The following external commands are available:

- **setenv** – a command for setting environment variables
- **printenv** – a command for verification of the environment variables

A help text regarding the usage of the external commands is printed out over the terminal in case of starting the commands without any parameter.

5.3.3 Requirements and settings for the communication with the EFI Shell

Communicating with the UEFI bootloader only takes place via the serial interface UART1. Regarding the configuration of the serial ports see section 5.1

5.4 Ethernet configuration of the ECUcore-E660

The main Ethernet configuration of the ECUcore-E660 takes place within the EFI Shell of the UEFI bootloader and is taken on for all software components (Linux, FTP server etc.). The Ethernet configuration is carried out via the serial interface UART1. **Therefore, the EFI Shell must be activated as described in section 5.3.2.** Table 8 lists up EFI Shell commands necessary for the Ethernet configuration of the ECUcore-E660.

Table 8: EFI Shell - Ethernet configuration commands

Configuration	Command	Remark
IP address	setenv ipaddr <xxx.xxx.xxx.xxx>	This command sets the local IP address of the ECUcore-E660. The IP address is to be defined by the network administrator. To assign a dynamic IP address to the ECUcore-E660 via DHCP, value "0.0.0.0" must be entered as address like shown in Figure 10. Compare the advice about DHCP in the text below!
Network mask	setenv netmask <xxx.xxx.xxx.xxx>	This command sets the network mask of the ECUcore-E660. 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-E660. The gateway address is set by the network administrator. Advice: If ECUcore-E660 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.
Domain Name System address	setenv dnsip <xxx.xxx.xxx.xxx>	This command defines the IP address of the DNS server which is to be used by the ECUcore-E660. The DNS IP address is set by the network administrator.

Modified configurations may be verified again by entering "*printenv*" at the EFI Shell command prompt.

Modifications are adopted upon next Linux boot of the ECUcore-E660.

```

COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help
UEFI Interactive Shell v2.0. UEFI v2.30 (SYS TEC, 0x00004E20).
Mapping table
  FS0: Alias(s):HD16b;BLK1:
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)/HD(1,MBR,0x1541B300,0x12C,0x40001)
  FS1: Alias(s):HD17b;BLK3:
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x1)/HD(1,MBR,0x00000000,0x3F,0x3C4F82)
  BLK0: Alias(s):
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)
  BLK2: Alias(s):
      PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x1)

Press ESC in 0 seconds to skip startup.nsh or any other key to continue.
2.0 Shell> setenv ipaddr 0.0.0.0
2.0 Shell> echo -off
2.0 Shell> setenv netmask 255.255.255.0
2.0 Shell> echo -off
2.0 Shell> setenv gatewayip 192.168.10.1
2.0 Shell> echo -off
2.0 Shell>

```

Figure 10: Ethernet configuration of the ECUcore-E660 by EFI Shell commands

Advice about the application of DHCP:

The Embedded Linux of the ECUcore-E660 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.
- To create a Telnet or FTP connection to the ECUcore-E660 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 **UART1** of the ECUcore-E660. Therefore, logging in to the command shell of the ECUcore-E660 must be accomplished as described in section 5.5.1. Afterwards, the current IP address may be queried using command "ifconfig".

Advice: After the configuration is finished, the serial connection between PC and ECUcore-E660 is no longer necessary.

5.5 Login to the ECUcore-E660

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-E660. 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 **UART1** of the ECUcore-E660 – analogous 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-E660.

For logging in to the ECUcore-E660 via the Windows standard Telnet client, the command "*telnet*" must be called by using the IP address provided for example in Figure 19, e.g.

```
telnet 192.168.10.248
```

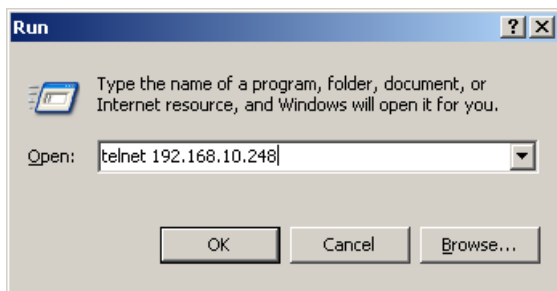


Figure 11: Calling the Telnet client in Windows

Logging in to the ECUcore-E660 is possible in the Terminal window (if connected via UART1) 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-E660 (also compare section 5.6):

User: *PlcAdmin*
Password: *Plc123*

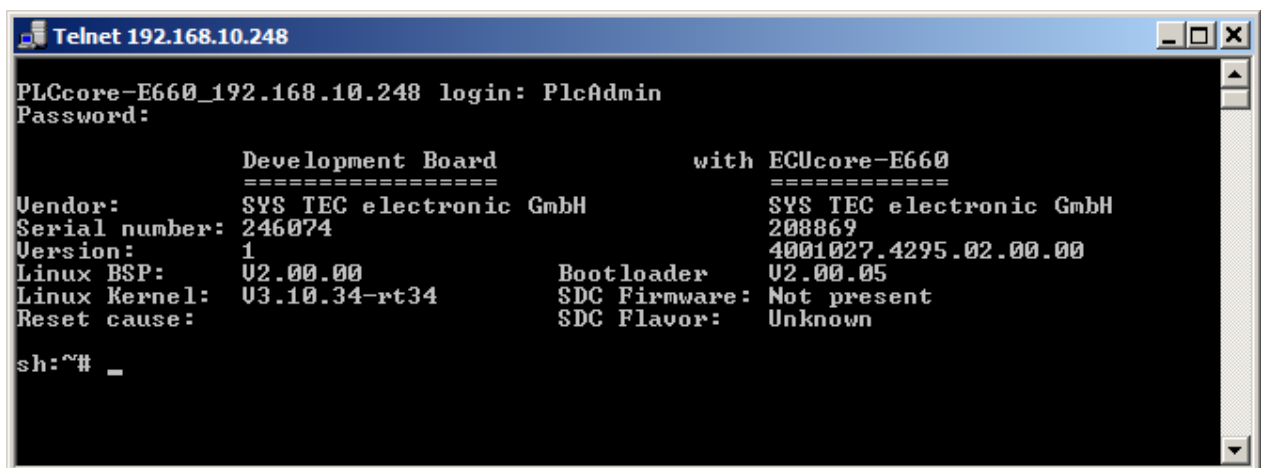


Figure 12: Login to the ECUcore-E660

Figure 12 exemplifies the login to the ECUcore-E660 using a Windows standard Telnet client.

5.5.2 Login to the FTP server

The ECUcore-E660 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-E660 following the procedures described in section 5.5.1. Afterwards, the following command must be entered in the Telnet or Terminal window:

```
pureftp
```

Figure 13 illustrates an example for starting the FTP server ("*pureftp*").

```

Telnet 192.168.10.248
PLCcore-E660_192.168.10.248 login: PlcAdmin
Password:

          Development Board                with ECUcore-E660
          =====
Vendor:    SYS TEC electronic GmbH        SYS TEC electronic GmbH
Serial number: 246074                      208869
Version:   1                               4001027.4295.02.00.00
Linux BSP:  U2.00.00                        Bootloader  U2.00.05
Linux Kernel: U3.10.34-rt34                SDC Firmware: Not present
Reset cause:                               SDC Flavor:  Unknown

sh:~# pureftp
sh:~# _

```

Figure 13: 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-E660 (see section 5.2.3).

"*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 14) and must be adjusted according to the following instructions:

File protocol: FTP
Host name: IP address of the ECUcore-E660 as shown for example in Figure 19
User name: *PlcAdmin* (for predefined user account, see section 5.6)
Password: *Plc123* (for predefined user account, see section 5.6)

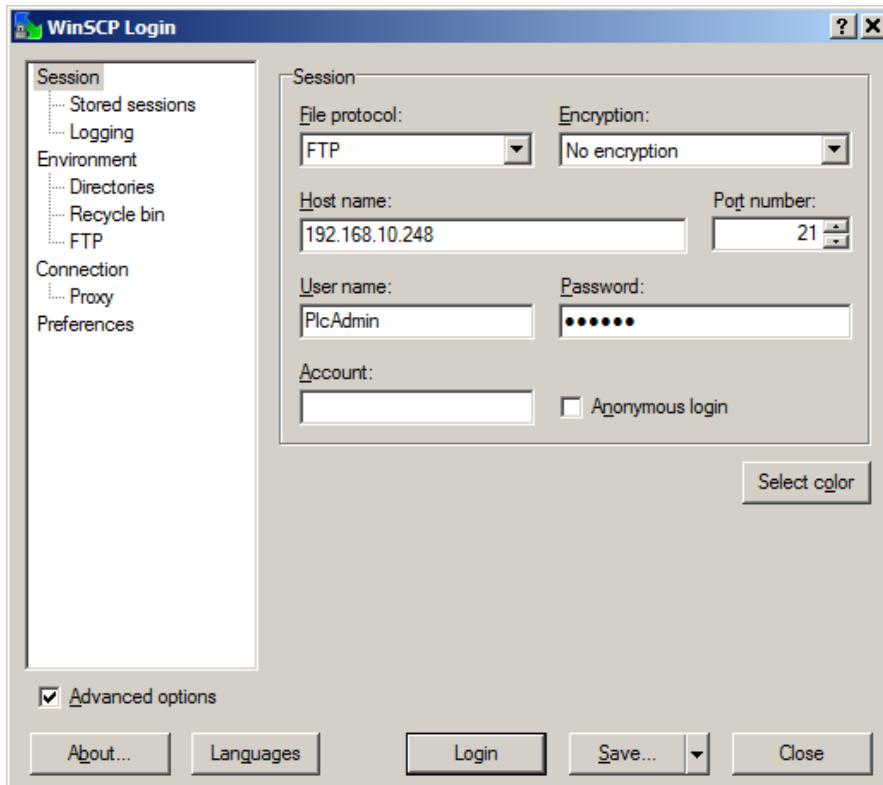


Figure 14: Login settings for WinSCP

After using pushbutton "Login", the FTP client logs in to the ECUcore-E660 and lists up the current content of directory "/home" in the right window. Figure 15 shows the FTP client "WinSCP" after successful login to the ECUcore-E660.

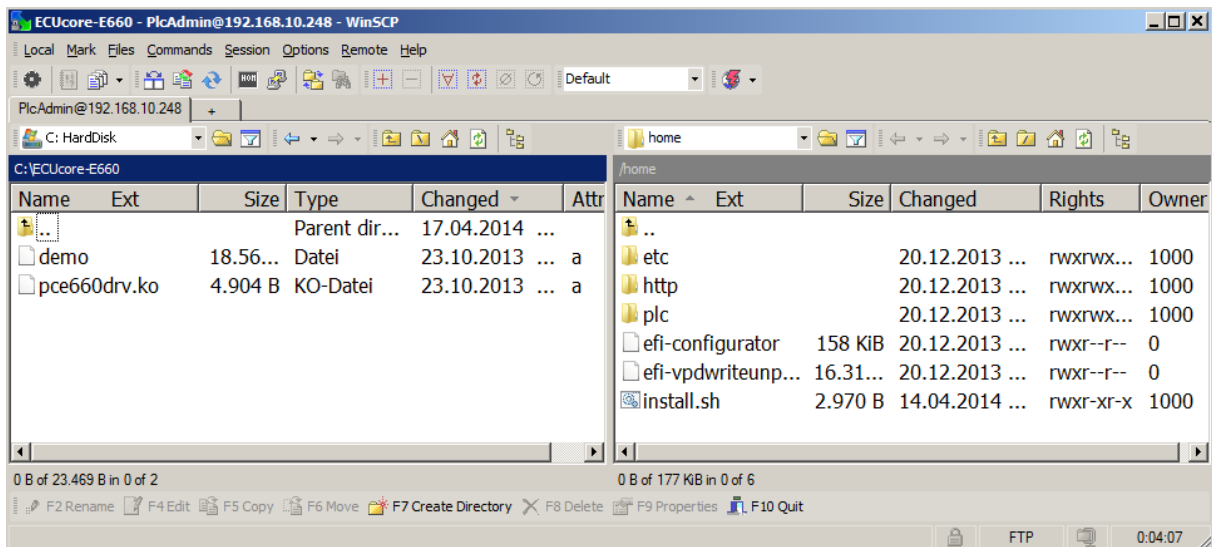


Figure 15: FTP client for Windows "WinSCP"

After successful login, configuration files on the ECUcore-E660 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-E660,

e.g. for data backups of the ECUcore-E660 or to transfer installation files for firmware updates (select transfer mode *"Binary"*).

5.6 Predefined user accounts

All user accounts in Table 9 are predefined upon delivery of the ECUcore-E660. Those allow for a login to the command shell (serial RS232 connection or Telnet) and at the FTP server of the ECUcore-E660.

Table 9: Predefined user accounts of the ECUcore-E660

User name	Password	Remark
PlcAdmin	Plc123	Predefined user account for the administration of the ECUcore-E660 (configuration, user administration, software updates etc.)
root	Sys123	Main user account ("root") of the ECUcore-E660

5.7 Adding and deleting user accounts

Adding and deleting user accounts requires the login to the ECUcore-E660 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-E660, 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-E660, Linux command *"adduser"* is to be used as follows:

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

Figure 16 exemplifies adding a new user account on the ECUcore-E660 for user *"admin2"*.

```

Telnet 192.168.10.248
PLCcore-E660_192.168.10.248 login: PlcAdmin
Password:
          Development Board          with ECUcore-E660
          =====
Vendor:      SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number: 246074                          208869
Version:     1                                4001027.4295.02.00.00
Linux BSP:   U2.00.00                          Boot loader   U2.00.05
Linux Kernel: U3.10.34-rt34                      SDC Firmware: Not present
Reset cause:                               SDC Flavor:   Unknown

sh:~# adduser -h /home -H -G users admin2
Changing password for admin2
New password:
Retype password:
Password for admin2 changed by root
sh:~# _

```

Figure 16: Adding a new user account

To **delete** an existing user account from the ECUcore-E660, 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-E660 as explained in section 5.5.1.

To change the password for an existing user account on the ECUcore-E660, Linux command *"passwd"* plus the respective user name must be entered:

```
passwd <username>
```

Figure 17 exemplifies the password change for user *"admin2"*.

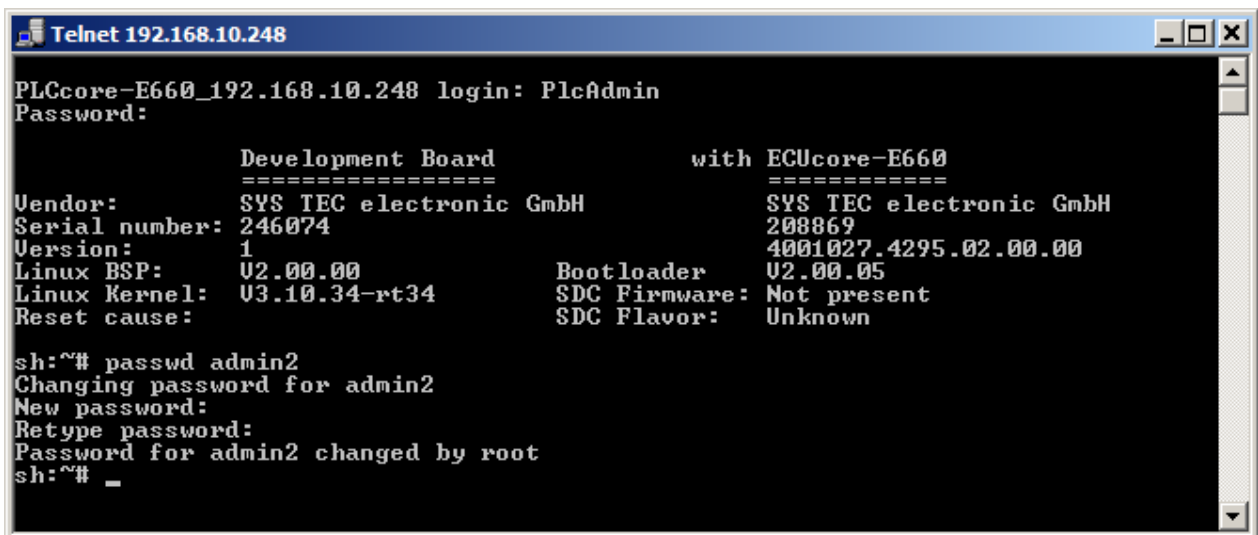


Figure 17: Changing the password for a user account

5.9 Setting the system time

Setting the system time requires login to the ECUcore-E660 as described in section 5.5.1.

There are two steps for setting the system time of the ECUcore-E660. 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-E660.

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

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

Example:

```

date    2013.02.25-11:34:55
|      |      |      |      |
|      |      |      |      +---- Second
|      |      |      | +----- Minute
|      |      | +----- Hour
|      | +----- Day
| +----- Month
+----- Year

```

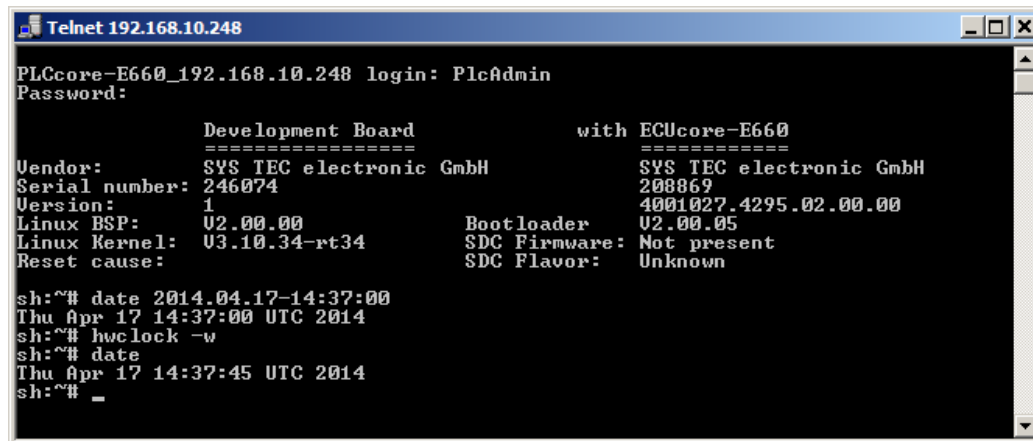
To set the system time of the ECUcore-E660 to 2013/02/25 and 11:34:55 (as shown in the example above), the following commands are necessary:

```

date 2013.02.25-11:34:55
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 18 exemplifies setting and displaying the system time.



```

Telnet 192.168.10.248
PLCcore-E660_192.168.10.248 login: PlcAdmin
Password:

          Development Board          with ECUcore-E660
          =====
Vendor:    SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number: 246074                      208869
Version:   1                               4001027.4295.02.00.00
Linux BSP: 02.00.00                        Bootloader 02.00.05
Linux Kernel: 03.10.34-rt34                SDC Firmware: Not present
Reset cause:                               SDC Flavor: Unknown

sh:~# date 2014.04.17-14:37:00
Thu Apr 17 14:37:00 UTC 2014
sh:~# hwclock -w
sh:~# date
Thu Apr 17 14:37:45 UTC 2014
sh:~# _

```

Figure 18: Setting and displaying the system time

Upon start of the ECUcore-E660, 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 EFI bootloader configuration data

Command `"fw_printenv"` under Linux enables the access to configuration data of the EFI bootloader. For example, it is used in the start script `"/etc/rc.d/S05network"` to re-use the Ethernet configuration set for the ECUcore-E660 within the EFI bootloader (see section 5.4) to parameterize interface "ETH0" under Linux.

```

Telnet 192.168.10.248
PLCcore-E660_192.168.10.248 login: PlcAdmin
Password:

          Development Board          with ECUcore-E660
          =====
Vendor:    SYS TEC electronic GmbH    SYS TEC electronic GmbH
Serial number: 246074                208869
Version:   1                          4001027.4295.02.00.00
Linux BSP: 02.00.00                  Bootloader 02.00.05
Linux Kernel: 03.10.34-rt34          SDC Firmware: Not present
Reset cause:                          SDC Flavor: Unknown

sh:~# fw_printenv
sernum=208869
ordernum=4002016
ipaddr=192.168.10.248
netmask=255.255.255.0
sh:~# _

```

Figure 19: Displaying EFI bootloader configuration data under Linux using "fw_printenv"

By calling "fw_printenv" without indicating parameters, all accessible EFI configuration data is shown (see Figure 19). A direct query of specific configuration information is possible by calling "fw_printenv <paramname>" (e.g. "fw_printenv ipaddr"). Figure 19 illustrates the usage of "fw_printenv", demonstrates the assignment of requested information to environment variables and shows the analysis of information in the Shell script.

5.11 Showing the installed Linux-Version

The Linux-Version installed on the ECUcore-E660 is shown by calling command "version" (see Figure 20).

```

Telnet 192.168.10.248
PLCcore-E660_192.168.10.248 login: PlcAdmin
Password:

          Development Board          with ECUcore-E660
          =====
Vendor:    SYS TEC electronic GmbH    SYS TEC electronic GmbH
Serial number: 246074                208869
Version:   1                          4001027.4295.02.00.00
Linux BSP: 02.00.00                  Bootloader 02.00.05
Linux Kernel: 03.10.34-rt34          SDC Firmware: Not present
Reset cause:                          SDC Flavor: Unknown

sh:~# version
Linux 03.10.34-rt34 / LinuxBSP 02.00.00
sh:~# _

```

Figure 20: Showing the installed Linux-Version

5.12 File system of the ECUcore-E660

The Embedded Linux which is preinstalled on the ECUcore-E660 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-E660. 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 10 lists up writable paths of the file system during runtime. Path "/" comprises a flash disk that provides part of the on-board flash memory of the ECUcore-E660 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 10: File system configuration of the ECUcore-E660

Path	Size	Description
/	129983 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	496616 kByte ¹	RAM disk, suitable as intermediate buffer for FTP downloads, but no data preservation in case of power breakdown
/var/log	496616 kByte ¹	RAM disk which is used by the system to store temporary files, no data preservation in case of power breakdown
/var/run	496616 kByte ¹	RAM disk which is used by the system to store temporary files, no data preservation in case of power breakdown
/var/lock	496616 kByte ¹	RAM disk which is used by the system to store temporary files, no data preservation in case of power breakdown
/var/tmp	496616 kByte ¹	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

¹⁾ All RAM disks share the same RAM. This means, that the really available size of a special RAM disk can vary dependent on the usage of the other RAM disks.

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 21.


```

COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help

Devboard_0050c2393f21 login: PlcAdmin
Password:

Development Board with ECUcore-E660
=====
Vendor: SYS TEC electronic GmbH SYS TEC electronic GmbH
Serial number: 246074 208869
Version: 1 4001027.4295.02.00.00
Linux BSP: V2.00.02 Bootloader V2.00.06
Linux Kernel: V3.10.34-rt34 SDC Firmware: V1.00.16
Reset cause: Cold reset SDC Flavor: Generic

sh:~# df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/rootfs
none            129983         57431    65999  47% /
none            496616          4    496612   0% /tmp
none            496616         88    496528   0% /var/log
none            496616         32    496584   0% /var/run
none            496616          0    496616   0% /var/lock
none            496616          0    496616   0% /var/tmp
tmpfs           496616         208    496408   0% /dev
sh:~#

```

Figure 21: Display of information about the file system

Particular information about the system login and handling the Linux command shell of the ECUcore-E660 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-E660 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-E660 includes the following files in the directory `"/home"`:

/home	
+-- etc	
+- autostart	Script that automatically starts user software (see section 5.2.3)
+- rc.usr	Optional user-specific configuration script
+- hotplug.sh	Scripts for response of connecting/disconnecting USB Sticks (see section 9.1)
+- diskadded.sh	
+- diskremoved.sh	
+-- bin	
+- pce660drv.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-E660 (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-E660.sh"*. The setup script *"setup-ECUcore-E660.sh"* is located in the directory *"SetupFlashdisk_ECUcore-E660"* of the DVD "SO-1116". Section 7.5 describes several possibilities to transfer this file onto the ECUcore-E660.

5.14 Using the HTTP server

HTTP server *"lighttpd"* is installed on the ECUcore-E660. 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-E660 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 configuration file. Therefore, the user must first be logged in to the command shell of the ECUcore-E660 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 22 exemplifies starting the HTTP server *"lighttpd"* for the Demo configuration included in the delivery status of the ECUcore-E660.

```
COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help

Devboard_0050c2393f21 login: PlcAdmin
Password:

Development Board with ECUcore-E660
=====
Vendor: SYS TEC electronic GmbH SYS TEC electronic GmbH
Serial number: 246074 208869
Version: 1 4001027.4295.02.00.00
Linux BSP: V2.00.02 Bootloader V2.00.06
Linux Kernel: V3.10.34-rt34 SDC Firmware: V1.00.16
Reset cause: Cold reset SDC Flavor: Generic

sh:~# lighttpd -f /home/http/lighttpd.conf
sh:~#
```

Figure 22: Starting the HTTP server "lighttpd"

To call the pages provided by the HTTP server, prefix "http://" plus the IP address of the ECUcore-E660 as set in section 5.4 (e.g. "http://192.168.10.65") must be entered in the address bar of the WEB-Browser. Figure 23 shows HTML pages for the ECUcore-E660 in the WEB-Browser.



Figure 23: Display of HTML pages for the ECUcore-E660 in the WEB-Browser

Advice: By entering the start call of the HTTP server (e.g. `lighttpd -c /home/http/lighttpd.conf`) into the start script `"/home/etc/autostart"`, calling the HTTP server upon boot of the ECUcore-E660 may be automated (see section 5.2.3).

5.15 HMI Components

5.15.1 Supported HMI Devices

The ECUcore-E660 can be used 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.

Displays

The ECUcore-E660 contains an integrated LCD Controller with support for displays with maximum 800x600 pixel resolution at 16 bit color depth. The Linux-Image installed on the ECUcore-E660 for delivery is supporting the display contained in the Development Kit ECUcore-E660:

Manufacturer:	AU-Optronics
Manufacturer Article No.:	G070VW01 V0
SYSTEC Article-No.:	Available upon customer request
Type/Size:	TFT-Display 7" WVGA LVDS with Backlight
Resolution:	WVGA (800x480 Pixel)

Alternatively, the connection of another display with maximum SVGA-resolution (800x600 Pixel) to the ECUcore-E660 is possible. For further information see section 5.15.2. A self-recognition of the connected display type is integrated in the Linux-Image by default. Please contact our Support Department in case of any further questions:

support@systec-electronic.com

Input Devices

The ECUcore-E660 supports the input devices listed in Table 11. The corresponding device drivers are aligned to the periphery contained in the Development Kit ECUcore-E660 but can be adjusted if needed. The necessary sources are included in the LinuxBSP of the ECUcore-E660 (Software package **SO-1116**, "VMware-Image of the Linux-Development System for the ECUcore-E660").

Table 11: Input Devices of ECUcore-E660

Device File	Input Device	Input Events
/dev/input/event<X>	Keyboard	<p>Event Type: 1 (Key)</p> <p>Event Codes: Keycodes</p> <p>Event Values: 1 – Key just pressed once 2 – Key permanently pressed 0 – Key just released</p>
/dev/input/event<Y>	Mouse	<p>Event Type: 2 (Relative)</p> <p>Event Codes: 0 – X direction movement value n 1 – Y direction movement value n 8 – Wheel movement value n</p> <p>Explanation for the value n: The value n represents the amount of - increments (positive) or - decrements (negative)</p>
		<p>Event Type: 1 (Key)</p> <p>Event Codes: 272 – LeftBtn 273 – RightBtn 274 – MiddleBtn</p> <p>Event Values: 1 – Down 0 – Up</p>

Which device file is linked to which input device can be determined in the following way as shown in Figure 24 below.

```

192.168.10.248:23 - Tera Term VT
File Edit Setup Control Window Help
sh:/dev/input# ls -l
drwxr-xr-x  2 root  root    100 Jan  1  2001 by-id
drwxr-xr-x  2 root  root    100 Jan  1  2001 by-path
crw-r----- 1 root  root     13,  64 Jan  1  2001 event0
crw-r----- 1 root  root     13,  65 Jan  1  2001 event1
crw-r----- 1 root  root     13,  66 Jan  1  2001 event2
crw-r----- 1 root  root     13,  67 Jan  1  2001 event3
crw-r----- 1 root  root     13,  68 Jan  1  2001 event4
crw-r----- 1 root  root     13,  69 Jan  1  2001 event5
crw-r----- 1 root  root     13,  70 Jan  1  2001 event6
crw-r----- 1 root  root     13,  71 Jan  1  2001 event7
crw-r----- 1 root  root     13,  63 Jan  1  2001 mice
crw-r----- 1 root  root     13,  32 Jan  1  2001 mouse0
sh:/dev/input# ls -l by-id/
lrwxrwxrwx  1 root  root         9 Jan  1  2001 usb-Logitech_USB_Receiver-event-kbd -> ../event2
lrwxrwxrwx  1 root  root         9 Jan  1  2001 usb-Logitech_USB_Receiver-event-mouse -> ../event3
lrwxrwxrwx  1 root  root         9 Jan  1  2001 usb-Logitech_USB_Receiver-mouse -> ../mouse0
sh:/dev/input# ls -l by-path/
lrwxrwxrwx  1 root  root         9 Jan  1  2001 pci-0000:02:08.0-usb-0:1:1.0-event-kbd -> ../event2
lrwxrwxrwx  1 root  root         9 Jan  1  2001 pci-0000:02:08.0-usb-0:1:1.1-event-mouse -> ../event3
lrwxrwxrwx  1 root  root         9 Jan  1  2001 pci-0000:02:08.0-usb-0:1:1.1-mouse -> ../mouse0
sh:/dev/input#

```

Figure 24: How to determine the device filename of input devices

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>
```

The links as shown in Figure 24 depend on the given configuration and can vary. Thus the number "X" in the term "event<X>" above can vary too.

Figure 25 demonstrates the command "evtest" on the example of the USB keyboard (/dev/input/event2 – for comparison see Figure 24) and shows the outputs generated when pressing the keys.

```

192.168.10.248:23 - Tera Term VT
File Edit Setup Control Window Help
Event: time 1355987029.054822, ----- Report Sync -----
Event: time 1355987029.304542, type 1 (Key), code 57 (Space), value 2
Event: time 1355987029.304542, ----- Report Sync -----
Event: time 1355987029.337540, type 1 (Key), code 57 (Space), value 2
Event: time 1355987029.337540, ----- Report Sync -----
Event: time 1355987029.370540, type 1 (Key), code 57 (Space), value 2
Event: time 1355987029.370540, ----- Report Sync -----
Event: time 1355987029.382796, type 4 (Misc), code 4 (ScanCode), value 7002c
Event: time 1355987029.382796, type 1 (Key), code 57 (Space), value 0
Event: time 1355987029.382796, ----- Report Sync -----
Event: time 1355987036.270799, type 4 (Misc), code 4 (ScanCode), value 70059
Event: time 1355987036.270799, type 1 (Key), code 79 (KP1), value 1
Event: time 1355987036.270799, ----- Report Sync -----
Event: time 1355987036.310784, type 4 (Misc), code 4 (ScanCode), value 70059
Event: time 1355987036.310784, type 1 (Key), code 79 (KP1), value 0
Event: time 1355987036.310784, ----- Report Sync -----
Event: time 1355987039.326797, type 4 (Misc), code 4 (ScanCode), value 7005a
Event: time 1355987039.326797, type 1 (Key), code 80 (KP2), value 1
Event: time 1355987039.326797, ----- Report Sync -----
Event: time 1355987039.422782, type 4 (Misc), code 4 (ScanCode), value 7005a
Event: time 1355987039.422782, type 1 (Key), code 80 (KP2), value 0
Event: time 1355987039.422782, ----- Report Sync -----
Event: time 1355987043.462790, type 4 (Misc), code 4 (ScanCode), value 70029
Event: time 1355987043.462790, type 1 (Key), code 1 (Esc), value 1
Event: time 1355987043.462790, ----- Report Sync -----
Event: time 1355987043.534778, type 4 (Misc), code 4 (ScanCode), value 70029
Event: time 1355987043.534778, type 1 (Key), code 1 (Esc), value 0
Event: time 1355987043.534778, ----- Report Sync -----

```

Figure 25: Diagnosis of Input Devices by means of "evtest"

5.15.2 Connections of Keyboard, Mouse and Display

A USB keyboard, a USB mouse and a display can be connected to the Development Board for HMI interaction to the ECUcore-E660. The schematic of the Development Board serves as reference design. Figure 26 shows the connectors for keyboard, mouse and display to the Development Board.



Figure 26: Connections for Keyboard, Mouse and Display to the Development Board

Explanation of HMI connectors

Table 12 lists the possibilities for the connection of the HMI devices to the Development Board. They all can be connected in different way (alternatively).

Table 12: Possible connections of HMI devices to the Development Board

Device	Connector	Explanation
USB keyboard	One of the USB connectors below X1000 or X1001	Direct connection of the USB keyboard
	Alternatively a USB hub on one of these USB connectors	The hub can be the same which the mouse is connected to.
USB mouse	One of the USB connectors below X1000 or X1001	Direct connection of the USB mouse
	Alternatively a USB hub on one of these USB connectors	The hub can be the same which the keyboard is connected to.
Display width: 800 Pixels height: 600 Pixels	X503 as LVDS signal connector	LVDS display
	X504 as backlight supply connector for the display	
	DVI graphics adaptor on slot X500	DVI monitor

5.16 The Use of Qt on the ECUcore-E660

Qt is a platform-independent C++-class library for programming graphical user interfaces. On the ECUcore-E660, 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-E660.

5.16.1 Overview of the Qt-Components for the ECUcore-E660

Together with the Linux-Image, the Qt-libraries and demo programs are generated for the ECUcore-E660 (see section 7.7). After completion of the build process, the needed software components are located in Linux Image

Table 13 lists the available Qt-Components for the ECUcore-E660.

Table 13: Qt-Components for the ECUcore-E660

File	Category	Meaning
qtdemo	Optional	Qt Demo program for execution on the ECUcore-E660 (see hint below, for program call see section 5.16.4)

Note: Through execution of the sample program "qtdemo" it can be checked easily whether the Qt Runtime-libraries are installed error-free and whether the requested environment variables are set correctly.

5.16.2 Preinstalled Qt-Libraries on the ECUcore-E660

A special installation of the Qt-components on the ECUcore-E660 is unnecessary. All required libraries are already preinstalled in the Linux image.

5.16.3 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. The environment variables for the input devices can be set automatically (by the firmware) as well as manually.

5.16.3.1 Automatic Configuration

The ECUcore-E660 firmware performs an automatic configuration of the input devices Mouse and Keyboard during startup and after hot plug. The Qt environment variables are set to the configured values of the input devices.

5.16.3.2 Manual Configuration

For manual configuration, those environment variables have to be set as follows:


```
QWS_KEYBOARD="LinuxInput:/dev/input/event<X>"
QWS_MOUSE_PROTO="LinuxInput:/dev/input/event<Y>"
```

On the ECUcore-E660 the values <X> for the keyboard and <Y> for the mouse depend on the given configuration and can vary (see also section 5.15.1).

If the keyboard input device events are mapped to "/dev/input/event2", the value "<X>" in the environment variable above must be set to 2.

If the mouse input device events are mapped to "/dev/input/event3", the value "<Y>" in the environment variable above must be set to 3.

I.e., in this case the given configuration requires the variables as follows:

```
QWS_KEYBOARD="LinuxInput:/dev/input/event2"
QWS_MOUSE_PROTO="LinuxInput:/dev/input/event3"
```

5.16.4 Starting Qt-Programs

The execution of Qt-programs on the ECUcore-E660 requires the installation of the Qt Runtime-libraries and the correct setting of needed environment variables (see section 5.16.3).

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-E660, in the given configuration (see section 5.16.3) the following command line sequence is needed:

In case of manual configuration of the environment variables at first the following two commands must be given over the console:

```
export QWS_KEYBOARD="LinuxInput:/dev/input/event2"
export QWS_MOUSE_PROTO="LinuxInput:/dev/input/event3"
```

In case of automatically configured QT environment variables only the last command must be given:

```
qtdemo -qws
```



Figure 27: Output of "qtdemo" on the ECUcore-E660

Figure 27 shows the display output of "qtdemo" generated on the ECUcore-E660. By clicking the button "Exit" with the left button of the mouse, the program is closed. It is also closed, when the button <SPACE> on the keyboard is pressed.

5.17 Updating the Linux-Image

An update of the Linux-Image must be performed by creating a new SD card image (see section 7.7).

This SD card image must be written to the SD card that must be placed in SD card slot 1 of the Development Board. This procedure is described in section 6.8.

6 VMware-Image with Linux Development System

6.1 Overview

The ECUcore-E660 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-E660. 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 Atom-Processors
- Linux-Sourcecode for the ECUcore-E660 (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-E660 and the development computer)
- NFS server (enables the integration of the development computer into the local file system of the ECUcore-E660)

6.2 Installing the Linux VMware-Image

The Development Kit ECUcore-E660 contains the DVD "SO-1116" which includes the VMware-Image for the Linux development system of the ECUcore-E660 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-E660 could possibly be defective.

The VMware-Image is included on the DVD as self-extracting archive "**SO-1116.exe**". If "**SO-1116.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-ECUcoreE660.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.

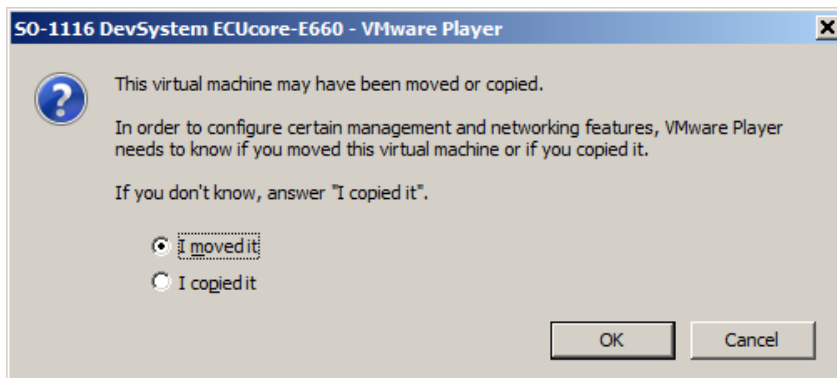


Figure 29: VMware selection dialog to generate or remain the MAC address

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

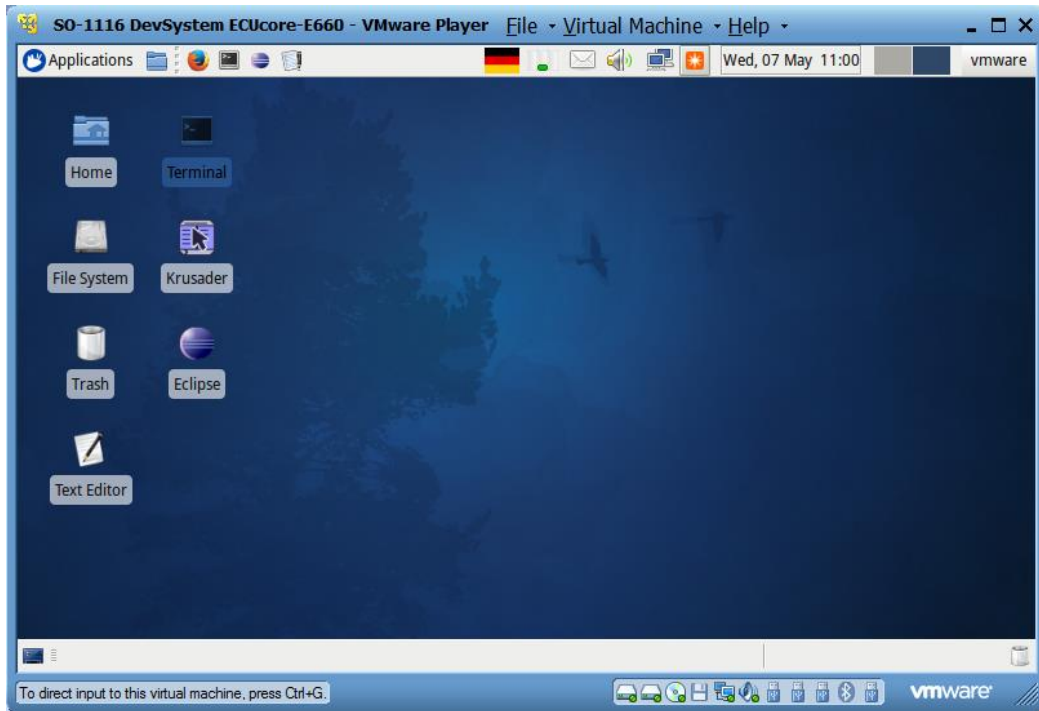


Figure 30: Desktop of the Linux development system

6.4 User accounts to log in to the Linux development system

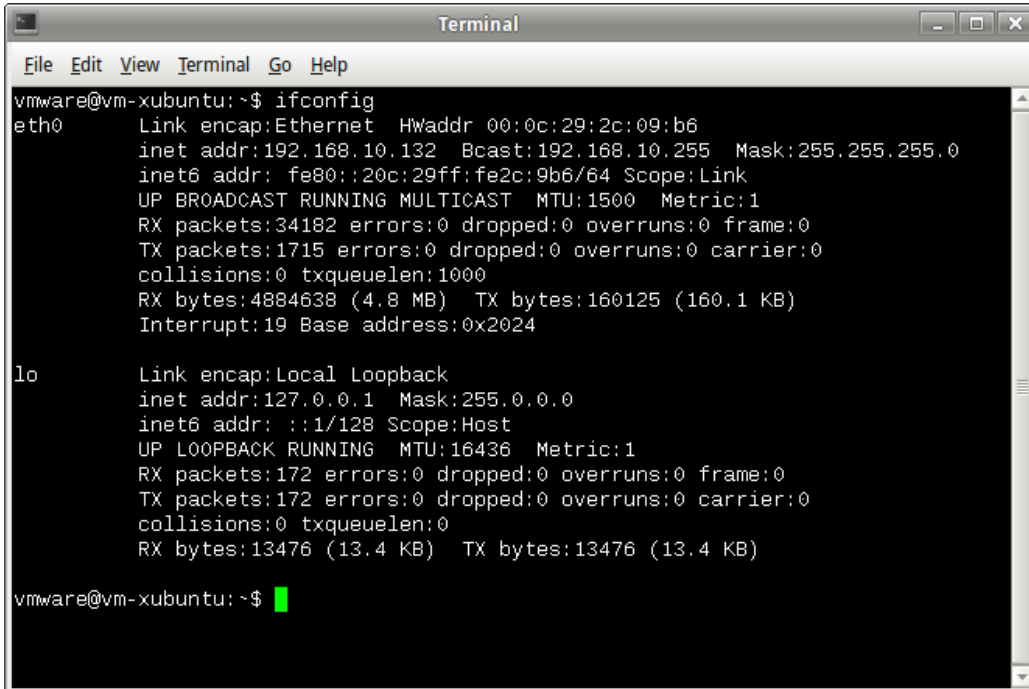
Table 14 lists up all predefined user accounts for logging in to the Linux development system.

Table 14: 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 00:0c:29:2c:09:b6
          inet addr:192.168.10.132  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe2c:9b6/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34182 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1715 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4884638 (4.8 MB)  TX bytes:160125 (160.1 KB)
          Interrupt:19 Base address:0x2024

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:172 errors:0 dropped:0 overruns:0 frame:0
          TX packets:172 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13476 (13.4 KB)  TX bytes:13476 (13.4 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 14 in section 6.4).

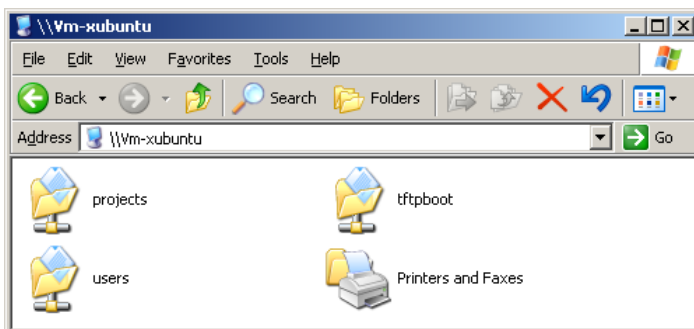


Figure 32: Linux development system in Windows network environment

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.

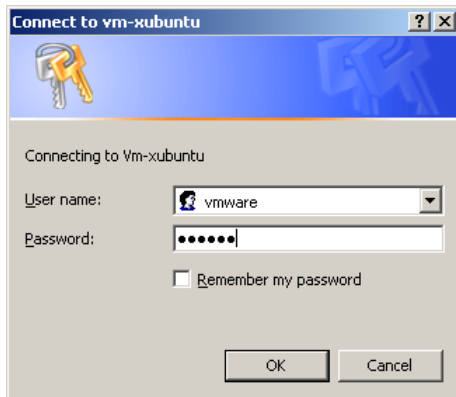


Figure 33: Login to "Vm-xubuntu"

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: \\192.168.10.82\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-E660 (compare Figure 11 in section 5.5.1), e.g.

```
telnet 192.168.10.82
```

Login as user "vmware" with the corresponding password "vmware" is possible in the Telnet window (also see Table 14 in section 6.4):

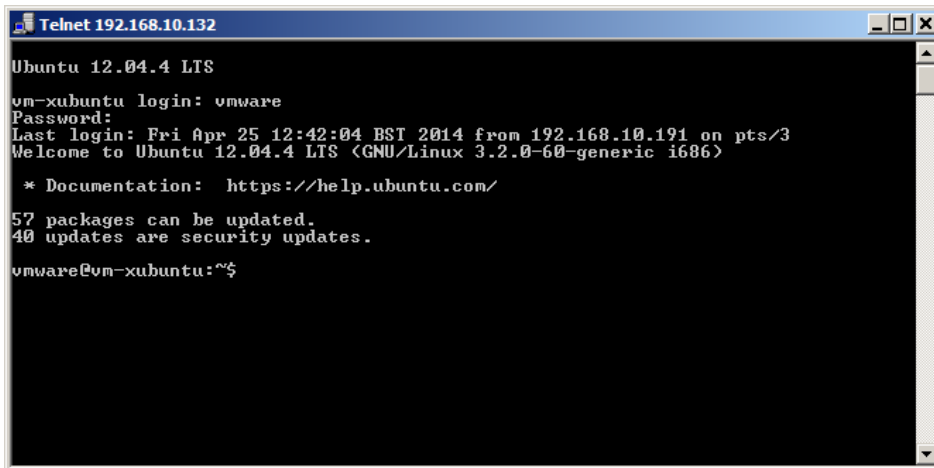


Figure 34: Access to the Linux development system via Telnet client

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).

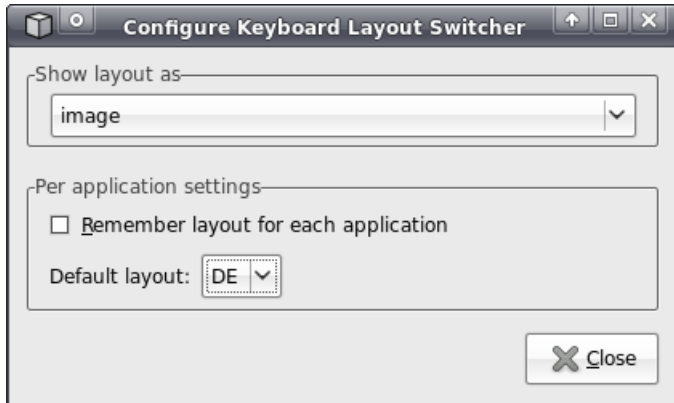


Figure 36: Choosing a permanent keyboard layout

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).

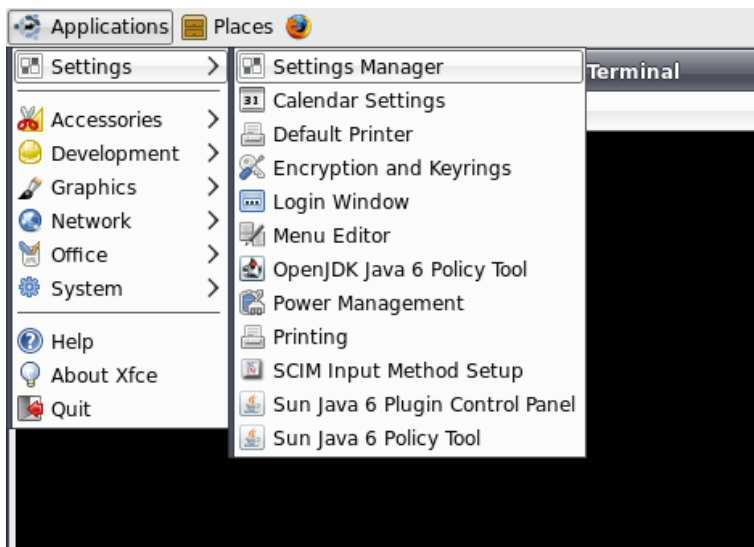


Figure 37: Calling "Xfce Settings Manager" from the start menu

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

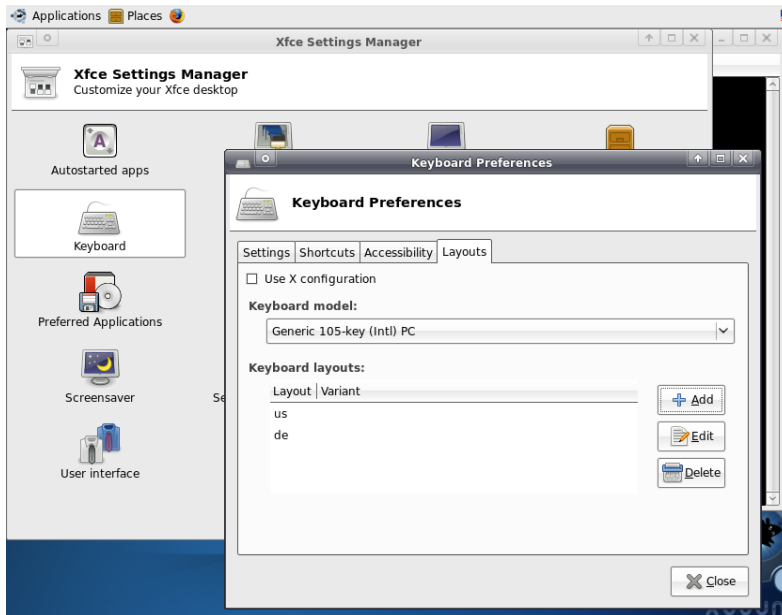


Figure 38: Adding keyboard layouts in the "Xfce Settings Manager"

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 14 in section 6.4). Afterwards, the capital and the corresponding time zone can be chosen via path "*Time zone*".

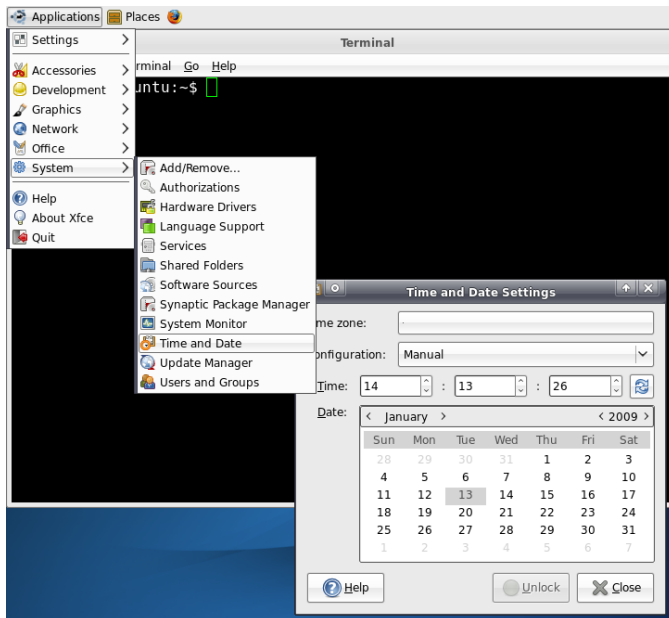


Figure 39: Adjusting the time zone

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-1116Xubuntu-ECUcoreE660.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-E660 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).

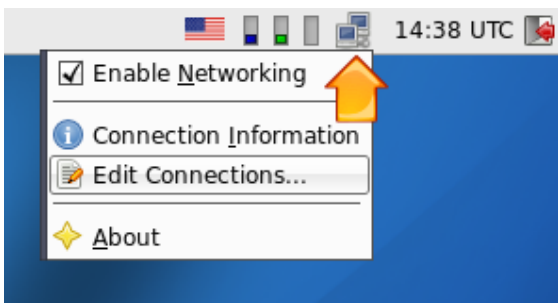


Figure 40: "Network Manager" for the configuration of the Ethernet interface

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).

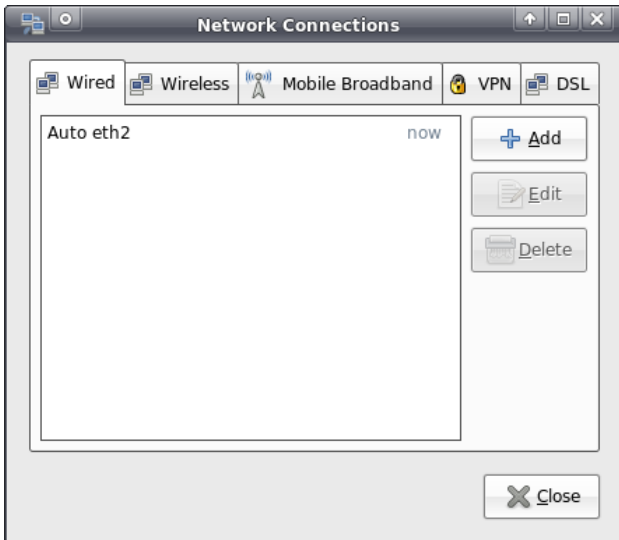


Figure 41: Adding a network connection

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".

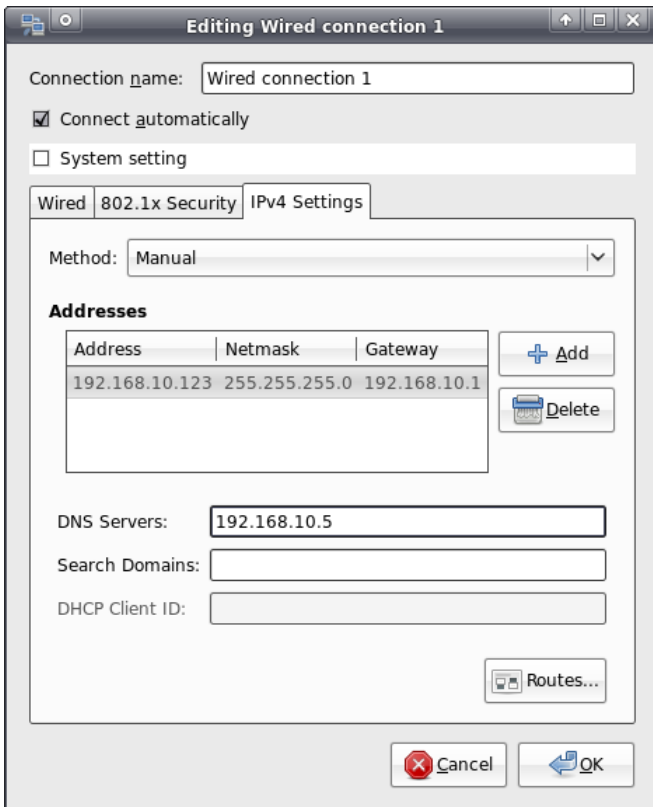


Figure 42: Configuration of the network connection

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.

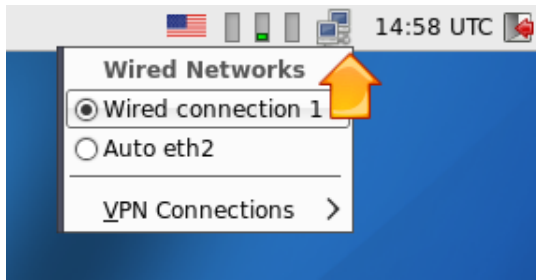


Figure 43: Changing the network configuration in the "Network Manager"

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 cannot be guaranteed that the Linux development system of the ECUcore-E660 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 14 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*".

6.8 How to write the SD card image to a SD card

Procedure inside the VMware:

Insert the SD card into an SD card reader. Connect the SD card reader to the USB.

```
$ sudo dd if=hd.img of=<SDcardDevice> bs=1M
```

<SDcardDevice> corresponds to the block device of the SD card. Normally, this is /dev/sdb or following. Assure to not overwrite any local hard disk.

Procedure outside the VMware on the Windows computer:

To write the SD card image onto the SD card under Windows a free software tool named "win32diskimager" is available. It can be downloaded from:

<https://sourceforge.net/projects/win32diskimager/>.

The SD card image "hd.img" must be available over a local directory on the Windows computer or a directory on a Windows network volume (e.g. over WinSCP) which is located inside the VMware.

6.9 How to write the SD card image to the internal eMMC

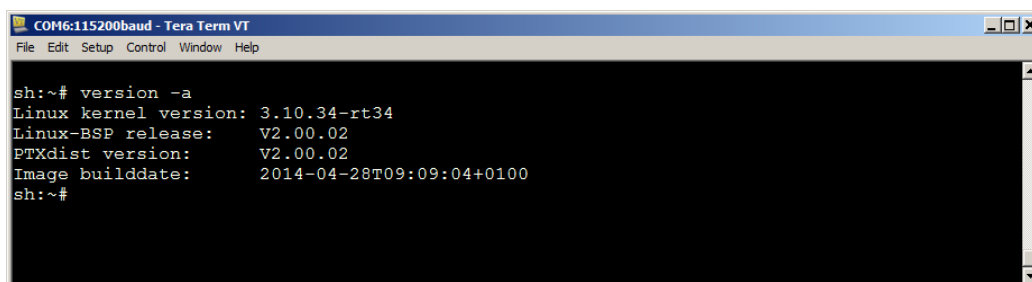
- Boot the target system from SD card.
- Copy the SD card image "hd.img" to the temp file system of the target system (e.g. via SCP or USB thumb drive).
- Copy the SD card image to the eMMC:

```
$ dd if=/tmp/hd.img of=<eMMCDevice> bs=1M
```

<eMMCDevice> corresponds to the block device of the eMMC. Usually, this is /dev/mmcblk0 or /dev/mmcblk1. The number depends on the order the eMMC and SD card are detected by Linux. Starting with version 1.02.00 of the BSP /dev/disk/by-path/pci-0000:02:04.1 points to eMMC irrespective if the SD card is present or not.

6.10 How to read out the BSP version from the target system

There is the script *version*, which prints out the BSP version, build date and Ptxdist version. Therefore the command must be started with option "-a" as shown below in Figure 44: Version output over Tera Term terminal console with option "-a".



```
COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help
sh:~# version -a
Linux kernel version: 3.10.34-rt34
Linux-BSP release: v2.00.02
PTXdist version: v2.00.02
Image builddate: 2014-04-28T09:09:04+0100
sh:~#
```

Figure 44: Version output over Tera Term terminal console with option "-a"

7 Software Development for the ECUcore-E660

7.1 Software structure of the ECUcore-E660

All components necessary to develop software for the ECUcore-E660 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 45 illustrates the directory structure.

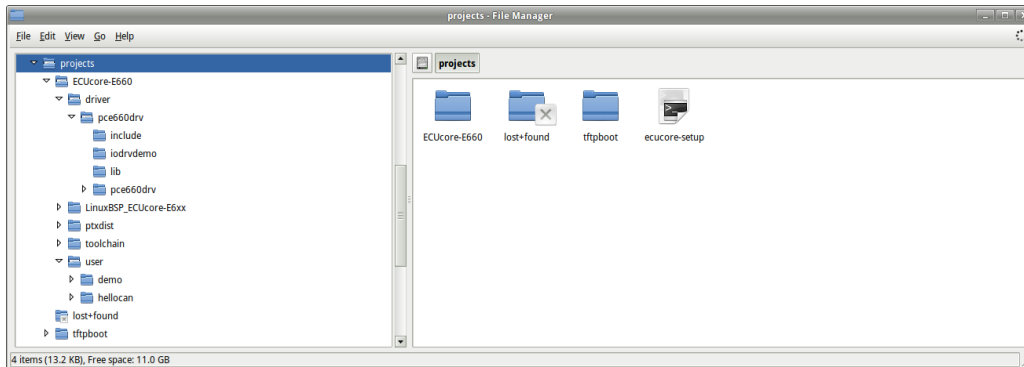


Figure 45: Structure of directory `"/projects"` in the Linux development system

<code>/projects</code>	
-- ECUcore-E660	
-- driver	Driver Libraries for the ECUcore-E660
-- pce660drv	I/O Driver for the integration into own user projects (including source code and testing application)
-- LinuxBSP_ECUcore-E6xx	Source code of the Linux-Images for the ECUcore-E660 (Linux kernel and user programs)
-- ptxdist	User programs and Build System for the ECUcore-E660
-- toolchain	GCC-Toolchain for the ECUcore-E660
-- user	Path to file user projects
-- demo	Reference and I/O Demo project for the ECUcore-E660
-- hellocan	Reference and CAN Demo project for the ECUcore-E660
-- tftpboot	NFS directory for the integration by the ECUcore-E660
-- ecucore-setup	Shell script to set environment variables

Path `"/projects/ECUcore-E660/LinuxBSP-2.6"` contains all Linux sources for the ECUcore-E660.

Directory `"/projects/ECUcore-E660/driver/pce660drv"` includes the source code of the I/O Driver of the ECUcore-E660 (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-E660/user/demo"` provides a Demo program that describes access to in- and outputs of the ECUcore-E660 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-E660/user/hellocan"*** includes a Demo program that describes access to the CAN interface of the ECUcore-E660 on the one hand and serves as template for own projects on the other.

Path ***"/projects/tftpboot"*** is the root directory for the integration of the Linux development system into the local file system of the ECUcore-E660 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-E660 requires the usage of GNU-Crosscompiler Toolchain for Intel x86 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:

```
I586E660_BASE_PATH=/projects/ECUcore-E660
I586E660_LINUX_BSP_PATH=$I586E660_BASE_PATH/LinuxBSP_ECUcore-E6xx
I586E660_LINUX_KDIR_PATH=$I586E660_LINUX_BSP_PATH/platform/build-target/linux-
3.10.34

I586E660_CC_PATH=$I586E660_BASE_PATH/toolchain/OSELAS.Toolchain-2011.03.0/i686-
unknown-linux-gnu/gcc-4.5.2-glibc-2.13-binutils-2.21-kernel-2.6.36-sanitized/bin

I586E660_CC_PREFIX=$I586E660_CC_PATH/i686-unknown-linux-gnu-
I586E660_CFLAGS=-D_FILE_OFFSET_BITS=64
I586E660_GDB_PATH=$I586E660_CC_PATH

export I586E660_BASE_PATH
export I586E660_LINUX_BSP_PATH
export I586E660_LINUX_KDIR_PATH
export I586E660_CC_PATH
export I586E660_CC_PREFIX
export I586E660_CFLAGS
export I586E660_GDB_PATH

# Path to sysroot for target SquashFS
export TARGET_SYSROOT=$I586E660_LINUX_BSP_PATH/platform/sysroot-target

# add toolchain to PATH
export PATH=$I586E660_CC_PATH:$PATH

# add ptxdist to PATH
export PATH=$I586E660_BASE_PATH/ptxdist/bin:$PATH

export SD_CARD_TARGET_IMAGE=/dev/sdc
```

The template in path ***"/projects/ECUcore-E660/user/demo"*** should be the initial point to develop own programs (or ***"\\Vm-xubuntu\projects\ECUcore-E660\user\demo"*** in the Windows network environment). Hereby, variables defined in Makefile (***demo/source/Makefile***) and references to GNU-Crosscompiler Toolchain for Intel x86 processors (based on definitions in ***"/projects/ecucore-setup"***) are especially relevant.

```
# C-Compiler settings
CDEFS = $(I586E660_CFLAGS) -D$(DBG_MODE)

# Toolchain command line settings
```

```
CFLAGS      += -O0 -g -Wall -Wno-pointer-sign -Wno-enum-compare $(INCLUDES)
$(CDEFS)
LDFLAGS     +=

CROSS       = $(I586E660_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:

```
# ----- Compile single Source -----

demo.o:    Makefile demo.c
           @echo "Compiling '$(notdir $*.c)'"
           @$$(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-E660 without other intermediate steps (also compare section 7.5.1).

7.3 I/O Driver for the ECUcore-E660

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

Directory *"/projects/ECUcore-E660/driver/pce660drv"* of the Linux development system contains the source code of the I/O Driver for the ECUcore-E660 (including testing application). Moreover, it contains Header files and complete libraries of the Driver for the integration into own user projects.

Figure 46 illustrates the structure of the I/O Driver for the ECUcore-E660. The Driver is divided into a Kernelspace module (*pce660drv.ko*) which is in charge of accesses to the hardware (Port pins) and a User space library (*pce660drv.a* as static library or *pce660drv.so* as dynamically loadable library). Both components, the Kernelspace module and the User space library (static or dynamic) are necessary to accomplish I/O accesses.

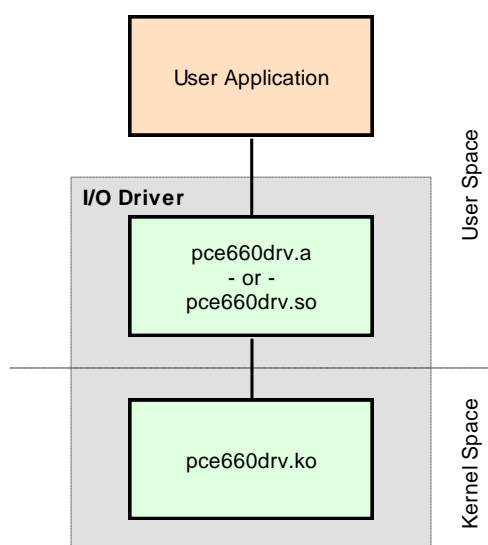


Figure 46: Structure of the I/O Driver for the ECUcore-E660

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

pce660drv.h: /*projects/ECUcore-E660/driver/pce660drv/include/pce660drv.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*.

pce660drv.a: /*projects/ECUcore-E660/driver/pce660drv/lib/pce660drv.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-E660.

pce660drv.so: /*projects/ECUcore-E660/driver/pce660drv/lib/pce660drv.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-E660, e.g.:

```
export LD_LIBRARY_PATH=.
```

pce660drv.ko: /*projects/ECUcore-E660/driver/pce660drv/lib/pce660drv.ko*
Kernel space 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 pce660drv.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 *pce660drv.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 = -lrt ../lib/pce660drv.a
```

```
@$(CC) $(LD_FLAGS) -o $@ $(OBJS) $(LIBS) $(LDLIBS)
```

Functions provided by the I/O Driver are listed in Header file "*pce660drv.h*", their usage is kept record of in the Demo program "*/projects/ECUcore-E660/user/demo*" (also see section 7.6).

The delivery status of the ECUcore-E660 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/pce660drv.ko
```

The Demo project in "*/projects/ECUcore-E660/user/demo*" shows an example for the application of the I/O Driver on the ECUcore-E660 (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-E660/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-E660/driver/pce660drv"`. Section 7.6 in much detail describes the Demo project as reference project for software development for the ECUcore-E660.

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 pce660drv.ko
./demo
```

Figure 52 in section 7.6.1. exemplifies the execution of the Demo project on the ECUcore-E660. The Demo project can be closed either by setting the Run/Stop Switch to position "MRes" or by pressing "Ctrl+C".

7.4 CAN Driver for the ECUcore-E660

7.4.1 Preinstalled CAN Driver in the Linux image

For the integration of own user projects using the CAN interface a Socket CAN driver is preinstalled in the ECUcore-E660. To activate it, the hash sign `"#"` at the beginning of each command line in the section

```
#-----
#  Setup CAN interface
#-----
```

of the shell script `"/home/etc/autostart"` must be removed by editing this shell script.

Demo project `"/projects/ECUcore-E660/user/hellocan"` is an example for the usage of the CAN Driver on the ECUcore-E660 (see section 7.4.2).

7.4.2 CAN Driver Demo project

The Demo project for the CAN Driver is filed in directory `"/projects/ECUcore-E660/user/hellocan"` of the Linux development system. It illustrates the access to the CAN interfaces of the module. Therefore, the Demo program needs the CAN Driver that is filed in `"/projects/ECUcore-E660/driver/candrv"`.

To demonstrate information exchange via the CAN-Bus, an appropriate receiver is needed. Therefore, the CAN analysis tool `"CAN-REport"` in combination with an USB-CANmodul is well suitable. By using `"CAN-REport"`, any CAN messages can be sent and received (see Figure 48).

Advice: The USB-CANmodul and CAN-REport are not included on the scope of delivery of Development Kit ECUcore-E660. 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 47 shows the execution of Demo project *"hellocan"* on the ECUcore-E660. The Demo project can be closed by pressing "Ctrl+C".

```

Telnet 192.168.10.248
Deuboard_0050c2393f21 login: PlcAdmin
Password:
=====
Development Board          with ECUcore-E660
=====
Vendor:   SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number: 246074                    208869
Version:  1                               4001027.4295.02.00.00
Linux BSP: 02.00.02                       Bootloader 02.00.06
Linux Kernel: 03.10.34-rt34              SDC Firmware: 01.00.16
Reset cause: Power failure (shutdown by power management controller) SDC Flavo
r: Generic

sh:~# ./mountnfs.sh 192.168.10.132
Check reachability of nfs server '192.168.10.132'... server is online
mount /mnt/nfs... mount.nfs: /mnt/nfs is busy or already mounted
done.

sh:~# cd /mnt/nfs/hellocan/
sh:/mnt/nfs/hellocan# ls
hellocan
sh:/mnt/nfs/hellocan# ./hellocan
*****
CAN demo application for SVSTEC ECUcore-E660
(C) 2010-2014 SVSTEC electronic GmbH
*****
Version: 1.10
Build: Apr 24 2014, 13:23:42
*****
Runtime configuration:
Supported instances: 1
DevNumber: 0
Bitrate: 125kBaud
Initialize CAN Device Number 0 ... ok
Register Rx CAN-ID pool:
Register Rx CAN-ID = 0x100 ... ok
Register Rx CAN-ID = 0x101 ... ok
Register Rx CAN-ID = 0x102 ... ok
Register Rx CAN-ID = 0x103 ... ok
Register Rx CAN-ID = 0x104 ... ok
Register Rx CAN-ID = 0x105 ... ok
Register Rx CAN-ID = 0x106 ... ok
Register Rx CAN-ID = 0x107 ... ok
Register Rx CAN-ID = 0x108 ... ok
Register Rx CAN-ID = 0x109 ... ok
Register Tx CAN-ID pool:
Register Tx CAN-ID = 0x200 ... ok
Register Tx CAN-ID = 0x201 ... ok
Register Tx CAN-ID = 0x202 ... ok
Register Tx CAN-ID = 0x203 ... ok
Register Tx CAN-ID = 0x204 ... ok
Register Tx CAN-ID = 0x205 ... ok
Register Tx CAN-ID = 0x206 ... ok
Register Tx CAN-ID = 0x207 ... ok
Register Tx CAN-ID = 0x208 ... ok
Register Tx CAN-ID = 0x209 ... ok
Register Tx CAN-ID for "hello" message:
Register Tx CAN-ID = 0x400 ... ok
Send "hello" message ... ok

Enter Main Loop ...

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

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

```

Figure 47: Execution of Demo project *"hellocan"* on the ECUcore-E660

Figure 48 shows data exchange with the Demo program *"hellocan"* in the CAN-Bus analysis tool *"CAN-Report"*.

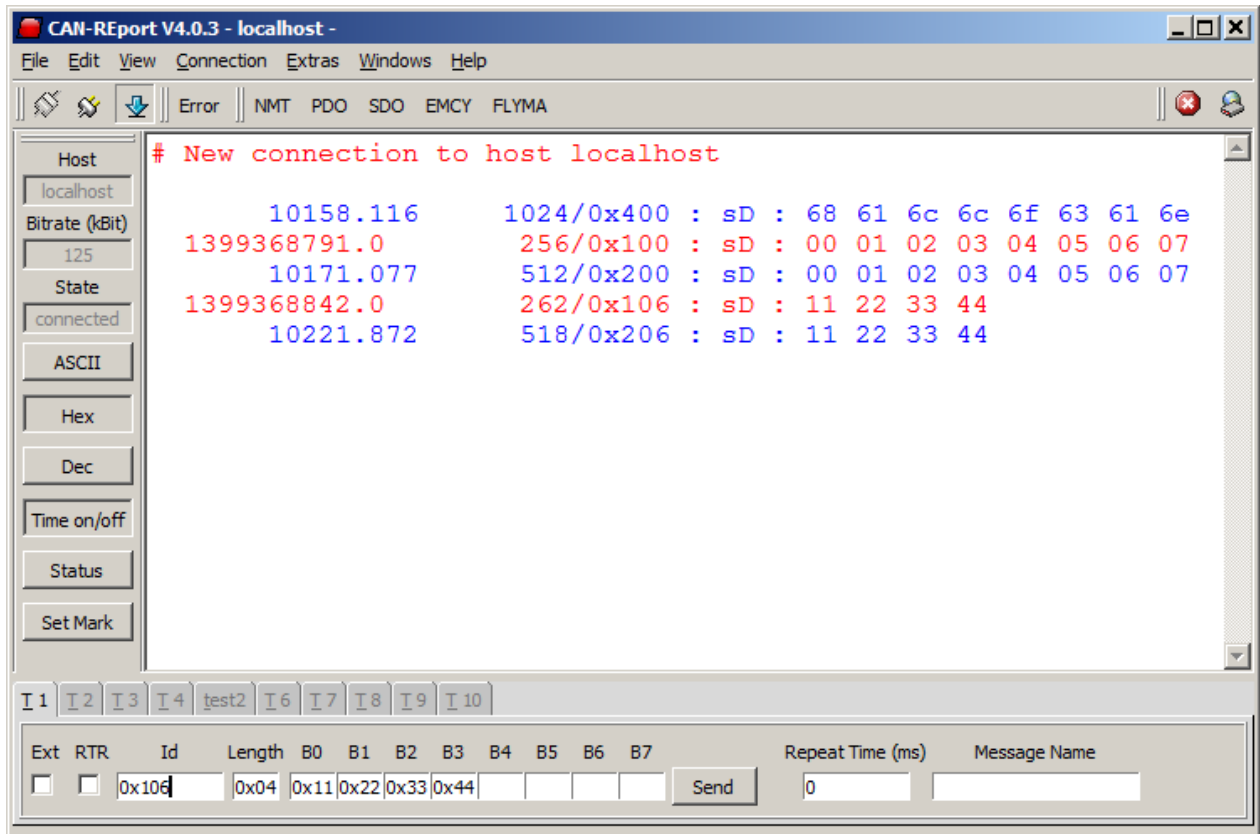


Figure 48: CAN analysis tool "CAN-REport"

7.5 Transferring programs to the ECUcore-E660

The configuration of the module as described in section 5 is one requirement for transferring – as well as starting – programs on the ECUcore-E660. Afterwards, login to the command shell of the ECUcore-E660 according to section 5.5.1 is necessary.

There are two possibilities for the transmission of programs on the ECUcore-E660 or data exchange between development system (VMware Linux-Image) and the ECUcore-E660 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-E660. To do so, from the VMware-Image of the Linux development system a directory is mounted into the local file system of the ECUcore-E660 ("mounted"). Enter the appropriate command on the ECUcore-E660 to start the program. Required data transfer from the development system to the ECUcore-E660 implicitly takes place via NFS – no further commands from the user are necessary. Consequently, it is assured that the ECUcore-E660 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 rudimental 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-E660 (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-E660 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-E660 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-E660. 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-E660:

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-E660

To mount directory *"tftpboot"* of the Linux-Image into the local file system of the ECUcore-E660, command *"mount"* must be used as follows:

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

For example, to attach the ECUcore-E660 to the Linux development system via the IP address defined in section 6.5, the following command must be entered on the ECUcore-E660:

```
mount -t nfs -o nolock 192.168.10.132:/tftpboot /mnt/nfs
```

Script *"mountnfs.sh"* that is at first located in the directory *"projects/tftpboot/"* of the VM is able to simplify the usage of the mount command. In case of a NFS-mounted *"projects/tftpboot/"* it must be copied before to the directory *"home/"* of the ECUcore-E660 by the command

```
cp /mnt/nfs/mountnfs.sh /home/
```

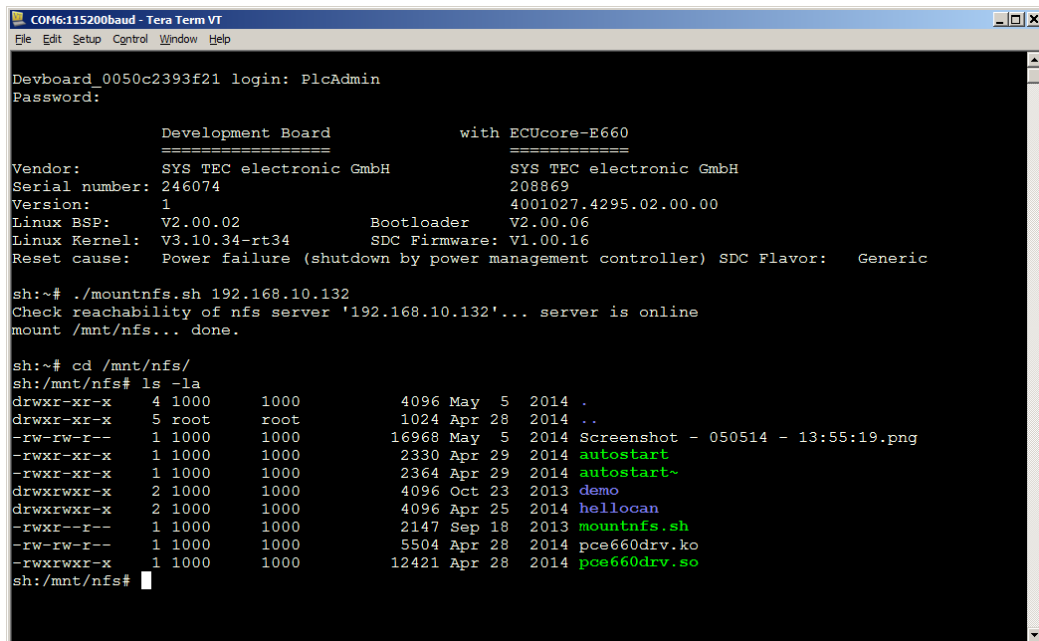
for usage as shown in Figure 49. The user is automatically located in that home directory after login to the command prompt. The IP address of the VM must be given to that script as parameter. Hence, for the example above this would lead to the following call:

```
./mountnfs.sh 192.168.10.132
```

It is unnecessary to set the "x" permission right (executable) for "mountnfs.sh" again after copy procedure by NFS access. It is already set in the VMware image in the directory */tftpboot/* and remains.

After the mount command (at first by command "mount" – after copying the shell script to the directory *"home/"* of the ECUcore-E660 by using the shell script as shown above) is executed, the entire content of directory *"tftpboot/"* of the VM (including potential sub-directories) is available in the local directory *"mnt/nfs/"* of the ECUcore-E660.

Figure 49 summarizes all necessary steps to mount the directory `"/fttboot/"` of the VM into the local file system of the ECUcore-E660.



```

COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help

Devboard_0050c2393f21 login: PlcAdmin
Password:

Development Board          with ECUcore-E660
=====
Vendor:      SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number: 246074                          208869
Version:     1                                4001027.4295.02.00.00
Linux BSP:   V2.00.02                          Bootloader V2.00.06
Linux Kernel: V3.10.34-rt34                      SDC Firmware: V1.00.16
Reset cause: Power failure (shutdown by power management controller) SDC Flavor: Generic

sh:~# ./mountnfs.sh 192.168.10.132
Check reachability of nfs server '192.168.10.132'... server is online
mount /mnt/nfs... done.

sh:~# cd /mnt/nfs/
sh:/mnt/nfs# ls -la
drwxr-xr-x  4 1000    1000    4096 May  5  2014 .
drwxr-xr-x  5 root    root    1024 Apr 28  2014 ..
-rw-rw-r--  1 1000    1000    16968 May  5  2014 Screenshot - 050514 - 13:55:19.png
-rwxr-xr-x  1 1000    1000    2330 Apr 29  2014 autostart
-rwxr-xr-x  1 1000    1000    2364 Apr 29  2014 autostart~
drwxrwxr-x  2 1000    1000    4096 Oct 23  2013 demo
drwxrwxr-x  2 1000    1000    4096 Apr 25  2014 hellocan
-rwxr--r--  1 1000    1000    2147 Sep 18  2013 mountnfs.sh
-rw-rw-r--  1 1000    1000    5504 Apr 28  2014 pce660drv.ko
-rwxrwxr-x  1 1000    1000    12421 Apr 28  2014 pce660drv.so
sh:/mnt/nfs#

```

Figure 49: Mounting the directory `"/fttboot/"` of the VM into the local file system of the ECUcore-E660

7.5.2 Using FTP

As an alternative to the integration of the Linux-Image into the local file system of the ECUcore-E660 via NFS, data may be transferred into both directions between the development computer and the ECUcore-E660 via FTP. There, the ECUcore-E660 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 ("eXecutable") 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-E660 as FTP client

If the ECUcore-E660 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-E660 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-E660 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-E660:

```
ftpget -u vmware -p vmware 192.168.10.132 /tmp/demo /tftpboot/demo/demo
```

FTP Upload

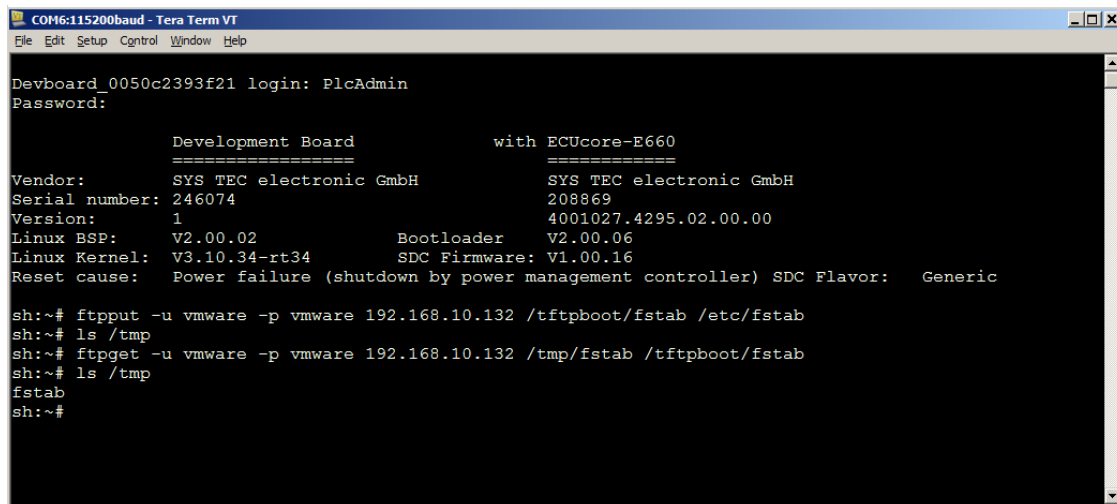
Uploading files of the ECUcore-E660 into the Linux-Image happens via command "ftpput". Parameters "-u" for username and "-p" for password are needed to authenticate at the host system. Command "ftpput" is written as follows:

```
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-E660 into directory "/tftpboot" of the host system:

```
ftpput -u vmware -p vmware 192.168.10.132 /tftpboot/autostart  
/home/etc/autostart
```

Figure 50 exemplifies the usage of commands "ftpget" and "ftpput" to download and upload files via FTP.



```
COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help

Devboard_0050c2393f21 login: PlcAdmin
Password:

Development Board          with ECUcore-E660
=====
Vendor:      SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number: 246074                          208869
Version:     1                                4001027.4295.02.00.00
Linux BSP:   V2.00.02                          Bootloader V2.00.06
Linux Kernel: V3.10.34-rt34                    SDC Firmware: V1.00.16
Reset cause: Power failure (shutdown by power management controller) SDC Flavor: Generic

sh:~# ftpput -u vmware -p vmware 192.168.10.132 /tftpboot/fstab /etc/fstab
sh:~# ls /tmp
sh:~# ftpget -u vmware -p vmware 192.168.10.132 /tmp/fstab /tftpboot/fstab
sh:~# ls /tmp
fstab
sh:~#
```

Figure 50: Download and upload via FTP

7.5.2.2 ECUcore-E660 as FTP server

The ECUcore-E660 provides a FTP server (FTP Deamon) 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-E660 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-E660 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-E660 and the login to the module via a FTP client. Section 5.1 specifies suitable FTP client programs.

7.6 Compilation and execution of demo project "demo"

7.6.1 Usage of "make"

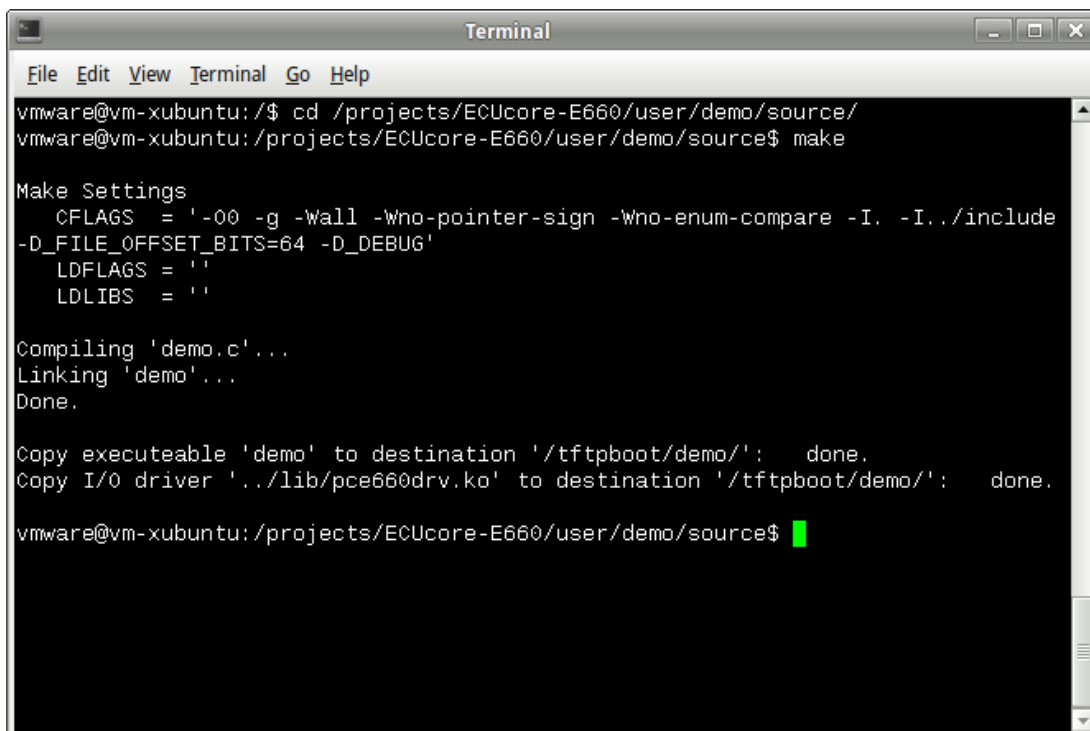
Directory `"/projects/ECUcore-E660/user/demo"` in the VMware-Image of the Linux development system contains a demo program that illustrates accesses to in- and outputs. The I/O driver filed in `"/projects/ECUcore-E660/driver/pce660drv"` is used by the demo program to access the I/O. It is recommended 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-E660/user/demo/source"`:

```
cd /projects/ECUcore-E660/user/demo/source
```

Afterwards, command `"make"` must be started. Figure 51 shows the usage of commands `"cd"` and `"make"` for the exemplary demo program contained in the VMware-Image.

A screenshot of a terminal window titled "Terminal". The terminal shows the following commands and output:

```
vmware@vm-xubuntu:/$ cd /projects/ECUcore-E660/user/demo/source/
vmware@vm-xubuntu:/projects/ECUcore-E660/user/demo/source$ make

Make Settings
  CFLAGS = '-O0 -g -Wall -Wno-pointer-sign -Wno-enum-compare -I. -I../include
-D_FILE_OFFSET_BITS=64 -D_DEBUG'
  LDFLAGS = ''
  LDLIBS = ''

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

Copy executable 'demo' to destination '/tftpboot/demo/': done.
Copy I/O driver '../lib/pce660drv.ko' to destination '/tftpboot/demo/': done.

vmware@vm-xubuntu:/projects/ECUcore-E660/user/demo/source$
```

Figure 51: Translating the demo project in the VMware Linux development system

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 *"pce660drv.ko"*) are copied to directory *"tftpboot/demo"* (compare screenshot in Figure 51). All required steps are completed after the translation and copying is finished.

All further steps exclusively take place on the ECUcore-E660. 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-E660 (see section 7.5.1):

```
mount -t nfs -o nolock 192.168.10.132:/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-E660 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-E660 in directory *"mnt/nfs/demo"* of its file system. It is not necessary to explicitly download executable binary files to the ECUcore-E660.

3. Starting the demo program on the ECUcore-E660.

Since the demo program accesses in- and outputs of the ECUcore-E660, it requires I/O driver *"pce660drv"* to do so. Consequently, the I/O driver must be loaded using command *"insmod"*. Afterwards, the demo program can be started:

```
insmod pce660drv.ko
./demo
```

```

Telnet 192.168.10.248
Devboard_0050c2393f21 login: PlcAdmin
Password:

                Development Board                with ECUcore-E660
                =====                =====
Vendor:          SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number:  246074                            200869
Version:        1                                4001027.4295.02.00.00
Linux BSP:      U2.00.02                          Boot loader   U2.00.06
Linux Kernel:   U3.10.34-rt34                    SDC Firmware: U1.00.16
Reset cause:    Warm reset                        SDC Flavor:   Generic

sh:~# mount -t nfs -o nolock 192.168.10.132:/tftpboot /mnt/nfs/
sh:~# cd /mnt/nfs/demo/
sh:/mnt/nfs/demo# ls
demo                pce660drv.ko
sh:/mnt/nfs/demo# insmod pce660drv.ko
sh:/mnt/nfs/demo# ./demo

*****
I/O demo application for SYSTEC ECUcore-E660
(c) 2012-2014 SYSTEC electronic GmbH
-----
Version: 1.00
Build:   May 5 2014, 13:39:48
*****

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

Hardware:  CPU Board: 4295.03 (<#00H>)
           IO Board: 4311.02 (<#02H>)
Driver:    Config:   0000H

Start basic I/O main loop...

DI=00-00-00  D0=00-00-01  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-02  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-04  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-08  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-10  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-20  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-40  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-80  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-01-00  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-02-00  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-04-00  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-08-00  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-10-00  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
^C

sh:/mnt/nfs/demo#

```

Figure 52: Executing the demo project "demo" on the ECUcore-E660

Figure 52 demonstrates the execution of the demo project on the ECUcore-E660. To finish the demo project either set Run/Stop Switch to position "MRes" or 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-E660/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 53. 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-E660/user/demo/workspace"`.

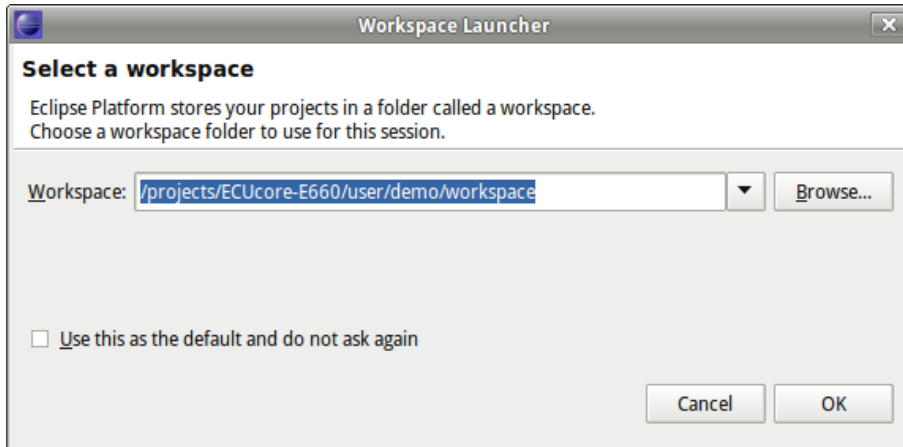


Figure 53: Eclipse dialog "Workspace Launcher"

After activating pushbutton "OK" the graphical surface of the IDE starts automatically and loads the workspace entered. Figure 54 provides an overview of "Eclipse" after it started.

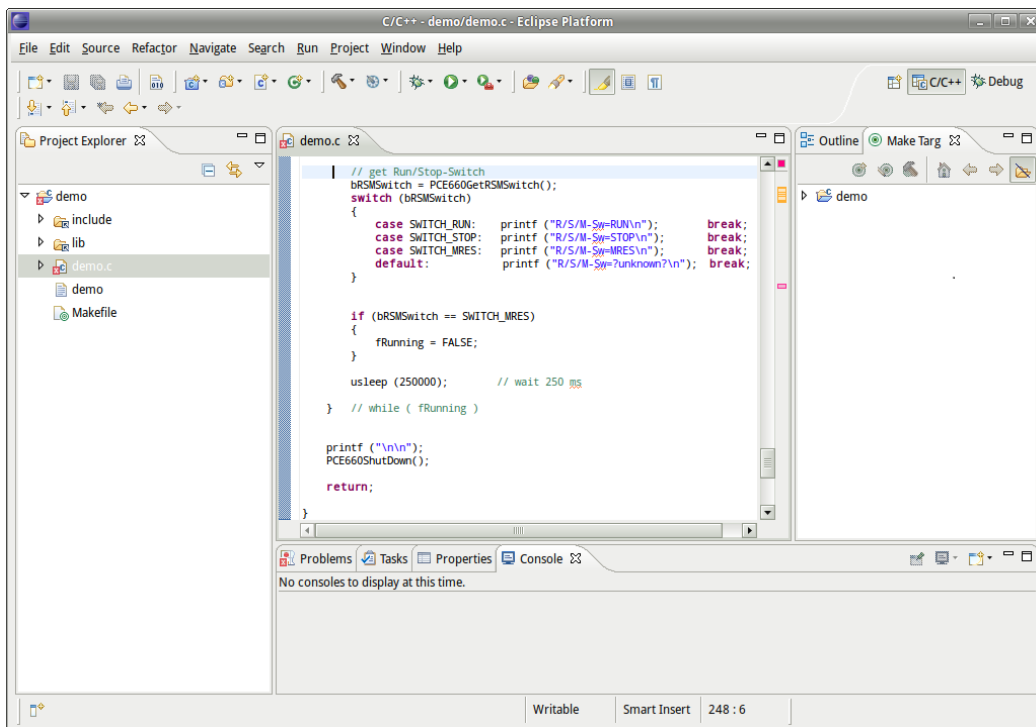


Figure 54: The graphical IDE "Eclipse"

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 55). Hence, Eclipse executes the Makefile of the demo project in directory *"source"* (*"/projects/ECUcore-E660/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 51. In the same way, the demo program (file *"demo"*) and the I/O drivers needed to execute it (file *"pce660drv.ko"*) are copied into directory *"/tftpboot/demo"*. Consequently, the demo program can be started on the ECUcore-E660 after successful completion of the Build process as explained in section 7.6.1.

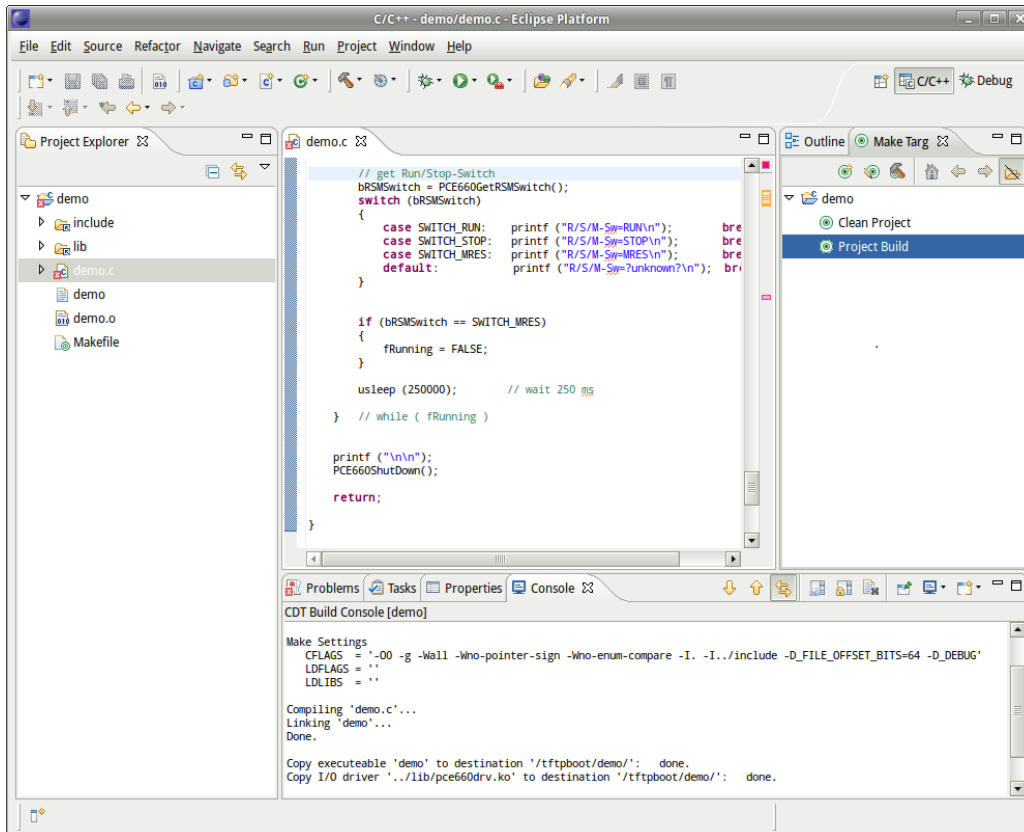


Figure 55: Translating the demo project in Eclipse

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 55) 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-E660 – 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-E660 (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-E660

For debug simply start the demo program on the ECUcore-E660 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-E660 via NFS (see section 7.5.1):

```
mount -t nfs -o nolock 192.168.10.132:/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-E660 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-E660 in directory `"/mnt/nfs/demo"` of its file system. It is not necessary to explicitly download executable binary files to the ECUcore-E660.

- 1.3 The demo program needs the I/O driver `"pce660drv"` to access the in- and outputs on the ECUcore-E660. Consequently, command `"insmod"` must be used to explicitly load the driver:

```
insmod pce660drv.ko
```

- 1.4 Starting the demo program on the ECUcore-E660 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 192.168.10.132:10000 ./demo
```

Use `"192.168.10.132"` 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 59).

Figure 56 illustrates the steps to be accomplished on the ECUcore-E660.

```

COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help

Devboard_0050c2393f21 login: PlcAdmin
Password:

          Development Board          with ECUcore-E660
          =====
Vendor:    SYS TEC electronic GmbH    SYS TEC electronic GmbH
Serial number: 246074                208869
Version:   1                          4001027.4295.02.00.00
Linux BSP: V2.00.02                    Bootloader V2.00.06
Linux Kernel: V3.10.34-rt34            SDC Firmware: V1.00.16
Reset cause: Warm reset                SDC Flavor: Generic

sh:~# mount -t nfs -o nolock 192.168.10.132:/tftpboot /mnt/nfs/
sh:~# cd /mnt/nfs/demo/
sh:/mnt/nfs/demo# ls
demo      pce660drv.ko
sh:/mnt/nfs/demo# insmod pce660drv.ko
sh:/mnt/nfs/demo# gdbserver 192.168.10.132:10000 ./demo
Process ./demo created; pid = 1397
Listening on port 10000

```

Figure 56: Starting the Debug server on the ECUcore-E660

To **simplify** this process, the delivery status of the ECUcore-E660 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 Configurations..."**. Thus, a configuration dialog opens as shown in Figure 57. Hence, follow the steps listed below:

2.1 Determining the program that is to be executed in the Debugger (see Figure 57)

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).

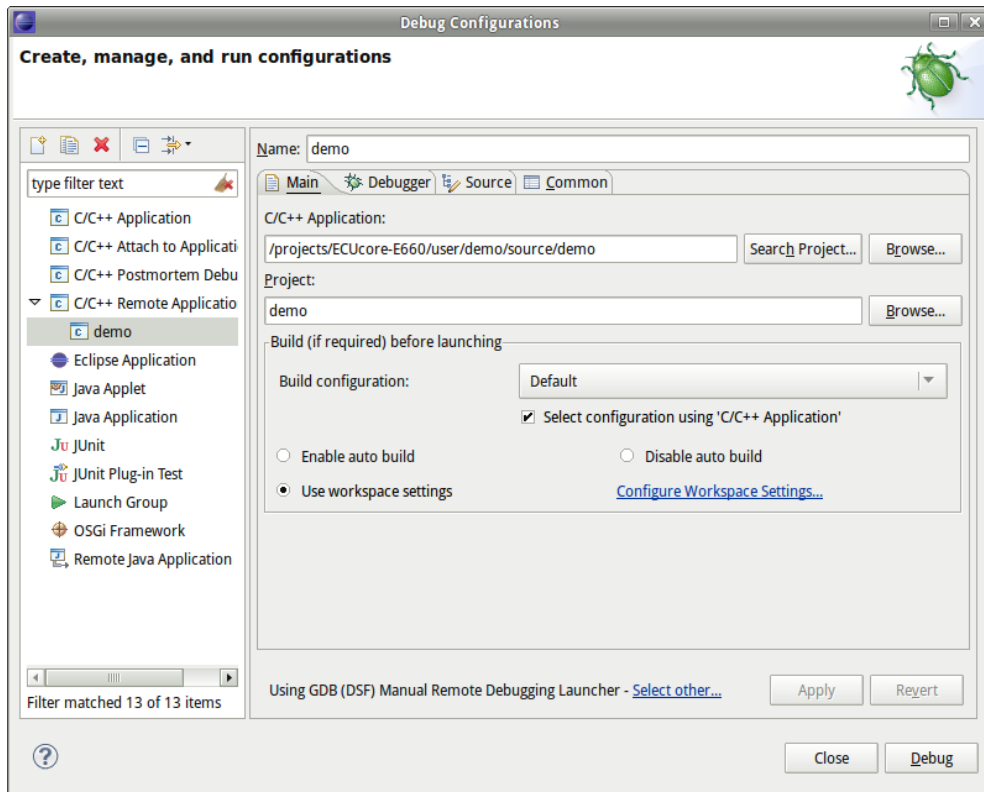


Figure 57: Determining the application that is to be debugged

2.2 Selecting the GDB Debugger to be used (see Figure 58)

Host-sided use the GDB Debugger that is included in the GNU-Crosscompiler Toolchain for ARM9 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 58; for the above example this would be: **`"/projects/ECUcore-E660/toolchain/OSELAS.Toolchain-2011.03.0/i686-unknown-linux-gnu/gcc-4.5.2-glibc-2.13-binutils-2.21-kernel-2.6.36-sanitized/bin/i686-unknown-gnu-gdb"`**).

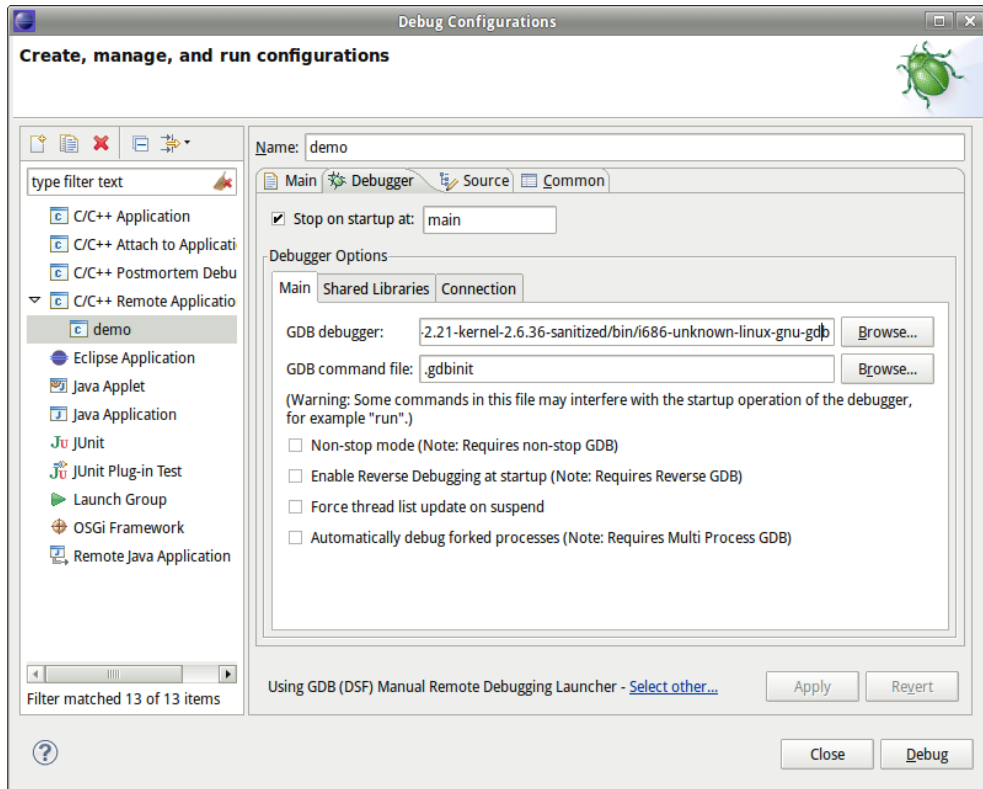


Figure 58: Selecting the GDB Debugger

2.3 Configuring the connection to the Target (see Figure 59)

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 59). In area *"Host name or IP address"* the IP address of the ECUcore-E660 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"*).

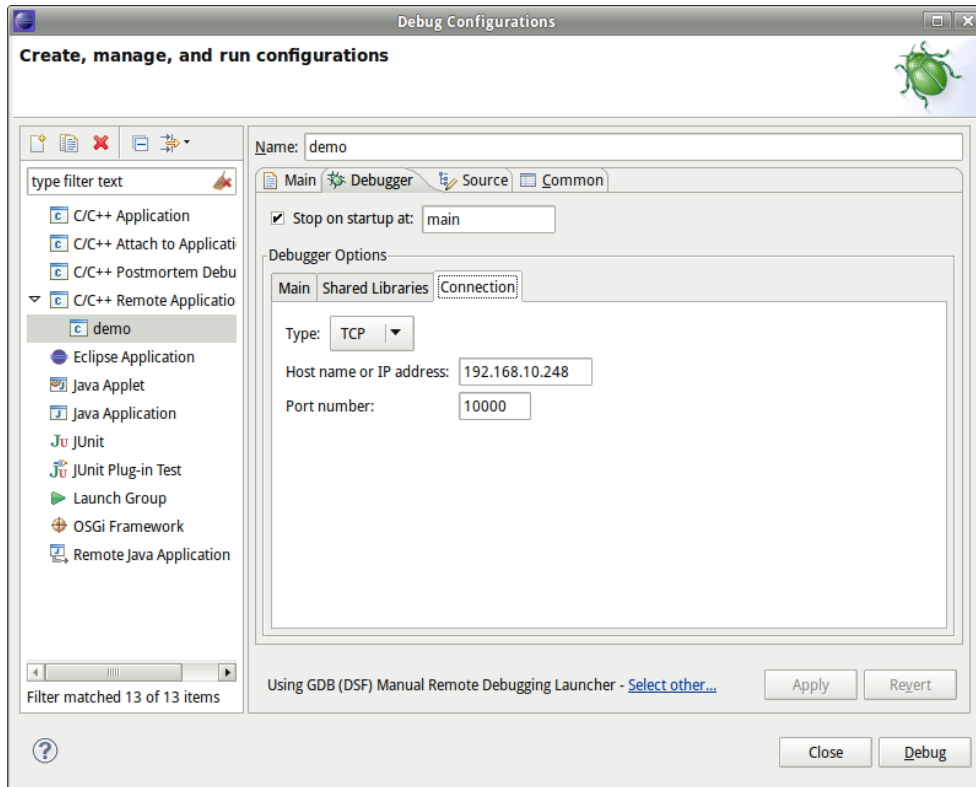


Figure 59: Configuring the connection to the Target

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 60). Table 15 lists up the most important Debugger commands.

Table 15: 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 60 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.

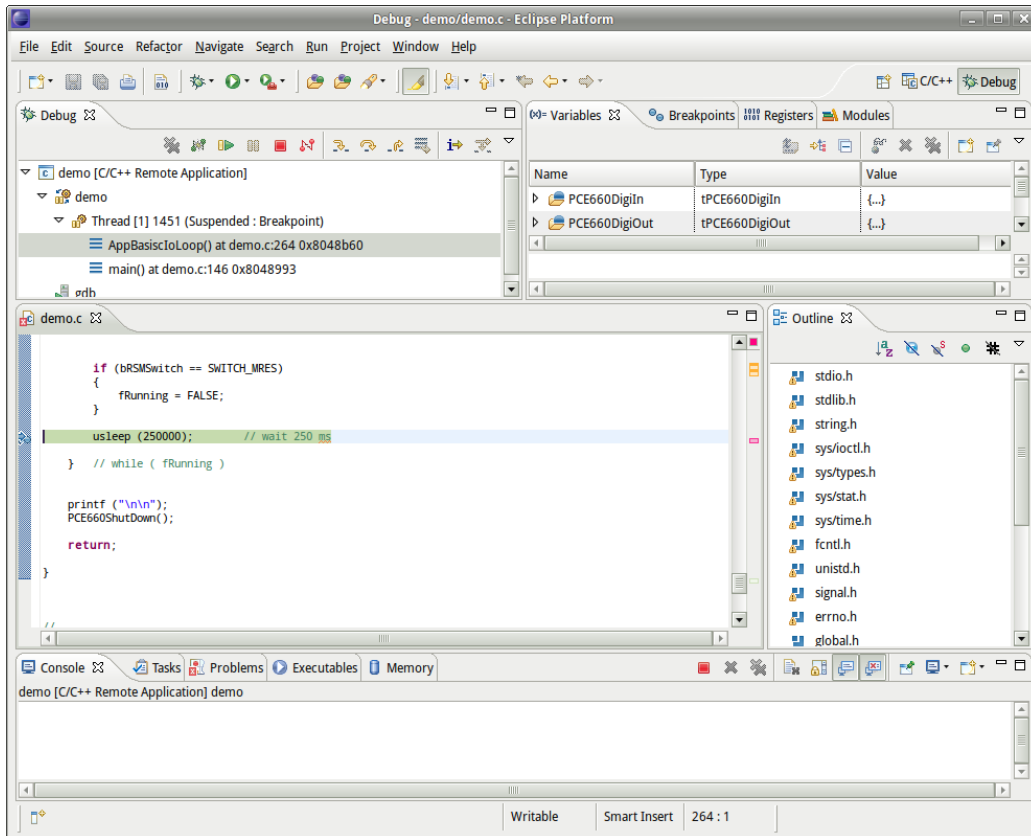


Figure 60: Debugging the Demo project in Eclipse

Figure 61 illustrates Terminal outputs that are generated on the ECUcore-E660 during the debugging process.

```

COM6:115200baud - Tera Term VT
File Edit Setup Control Window Help

Devboard_0050c2393f21 login: PlcAdmin
Password:

Development Board          with ECUcore-E660
=====
Vendor:      SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number: 246074                          208869
Version:     1                                4001027.4295.02.00.00
Linux BSP:   V2.00.02                          Bootloader V2.00.06
Linux Kernel: V3.10.34-rt34                    SDC Firmware: V1.00.16
Reset cause: Warm reset                        SDC Flavor: Generic

sh:~# mount -t nfs -o nolock 192.168.10.132:/tftpboot /mnt/nfs/
sh:~# cd /mnt/nfs/demo/
sh:/mnt/nfs/demo# ls
demo      pce660drv.ko
sh:/mnt/nfs/demo# insmod pce660drv.ko
sh:/mnt/nfs/demo# gdbserver 192.168.10.132:10000 ./demo
Process ./demo created; pid = 1451
Listening on port 10000
Remote debugging from host 192.168.10.132

*****
I/O demo application for SYSTEC ECUcore-E660
(c) 2012-2014 SYSTEC electronic GmbH
-----
Version: 1.00
Build:   May 6 2014, 15:14:46
*****

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

Hardware: CPU Board: 4295.03 (#00H)
          IO Board:  4311.02 (#02H)
Driver:   Config:   0000H

Start basic I/O main loop...

DI=00-00-00  D0=00-00-01  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-02  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-04  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-08  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN
DI=00-00-00  D0=00-00-10  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=STOP
DI=00-00-00  D0=00-00-20  bHexSw=0x00  bDipSw=0x1C  R/S/M-Sw=RUN

```

Figure 61: Terminal outputs of the ECUcore-E660 during debugging

7.7 Configuration and Creation or Update of the ECUcore E660 Linux-Image

All Linux sources for the ECUcore-E660 are filed in the VMware-Image of the Linux development system in directory `"/projects/ECUcore-E660/LinuxBSP_ECUcore-E6xx"`. To modify the configuration of the Linux kernel, use command `"cd"` in a console window to switch to the directory `"/projects/ECUcore-E660/LinuxBSP_ECUcore-E6xx"`. Afterwards, call command `"ptxdist kernelconfig"`:

```
cd /projects/ECUcore-E660/LinuxBSP_ECUcore-E6xx
ptxdist kernelconfig
```

Figure 62 shows the typical surface for the configuration of Linux kernel.

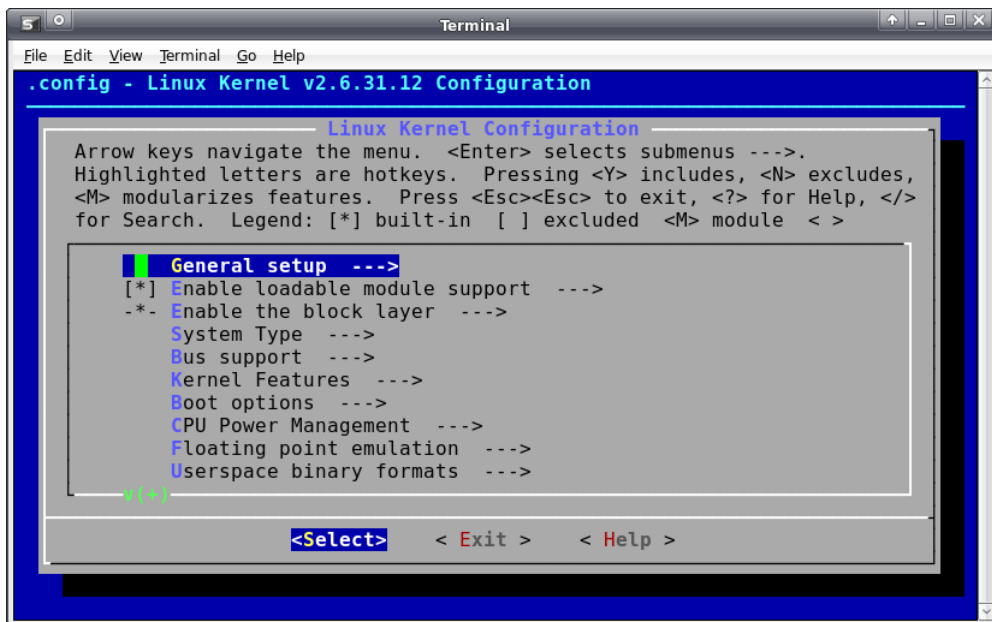


Figure 62: User surface for the configuration of the Linux kernel

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-E660/LinuxBSP_ECUcore-E6xx
ptxdist menuconfig
```

Figure 63 shows the typical user interface to configure user applications including BusyBox.

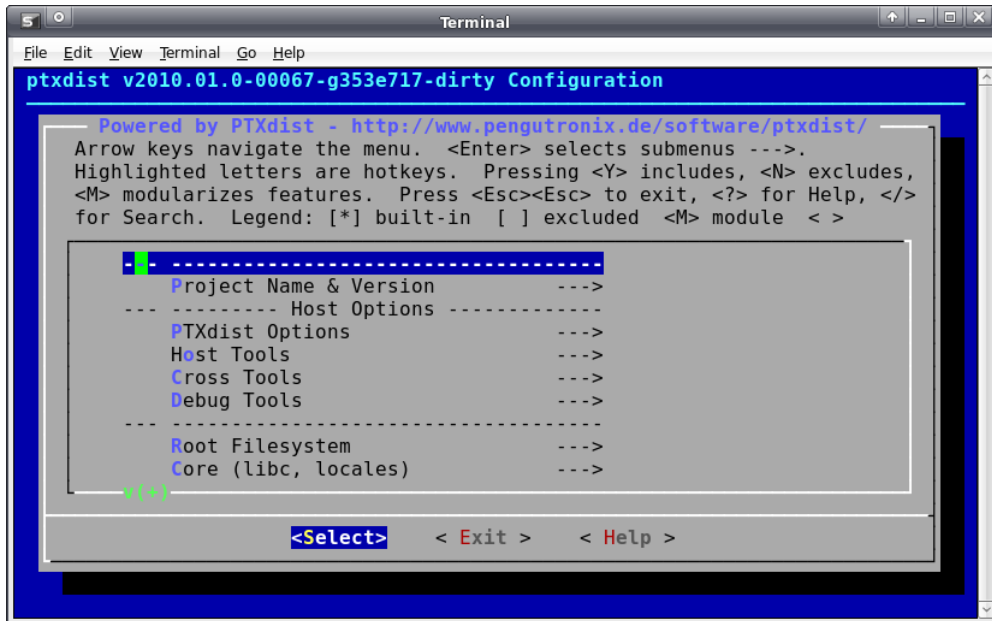


Figure 63: User interface to configure user applications including BusyBox

After configuring the Linux kernel, user applications and BusyBox, switch to superior directory **"/projects/ECUcore-E660/LinuxBSP-2.6"**. By calling **"ptxdist images"**, all software components are translated and summarized to a new image:

```
cd /projects/ECUcore-E660/LinuxBSP_ECUcore-E6xx
ptxdist images
```

The Linux image for the ECUcore-E660 is now available as a SD card Image named **"hd.img"**. It is accessible in the file system of the VM as file **"/projects/ECUcore-E660/LinuxBSP_ECUcoreE6xx/platform/images/hd.img"**. How to write this image onto the SD card is described in section 6.8.

8 Adaptation and Testing of the hardware connections

8.1 Driver Development Kit (DDK) for the ECUcore-E660

The Driver Development Kit (DDK) for the ECUcore-E660 is distributed as additional software package with the order number SO-1117. It is not included in the delivery of the Development Kit ECUcore-E660. The "Software Manual Driver Development Kit for the ECUcore-E660" (Manual no.: L-1561) provides details about the DDK.

The Driver Development Kit for the ECUcore-E660 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 Intel Atom processor
- Address-/Data Bus (memory-mapped periphery)
- SPI-Bus and I²C-Bus
- All other resources provided by the operating system, e.g. file system and TCP/IP

Figure 64 provides an overview of the DDK structure and its components. The DDK contains amongst others the source code of the Linux kernel driver (*pce660drv.ko*) and the Linux user library (*pce660drv.so*).

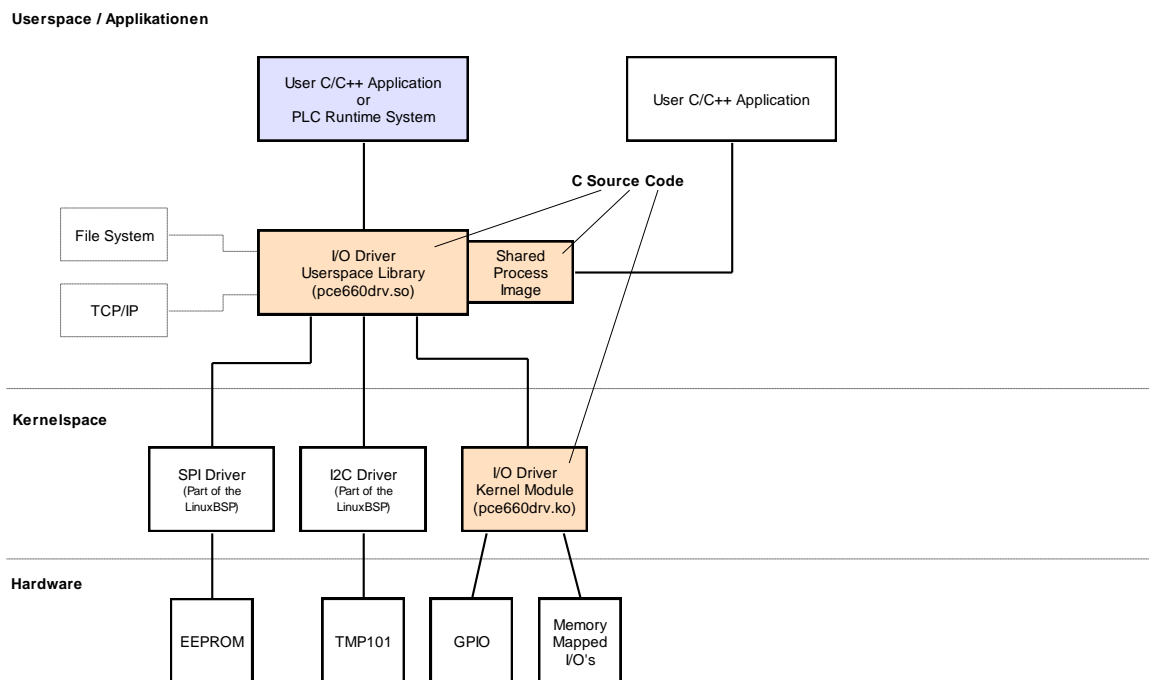


Figure 64: Overview of the Driver Development Kit for the ECUcore-E660

Scope of delivery / components of the DDK:

The DDK contains the following components:

1. Source code for the Linux kernel driver (*pce660drv.ko*, see Figure 64); includes all files necessary to regenerate kernel drivers (C and H files, Make file etc.)
2. Source code for the Linux user library (*pce660drv.so*, see Figure 64); 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-1116** ("VMware-Image of the Linux development system"). It contains sources of the LinuxBSP used and it includes the necessary GNU-Crosscompiler Toolchain for Intel processors.

8.2 Testing the hardware connections

The ECUcore-E660 primarily is designed as vendor part for the application in industrial controls. Hence, the ECUcore-E660 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-E660. 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 pce660drv.ko
./iodrvdemo
```

Figure 65 illustrates testing the hardware connection by using "*iodrvdemo*".

```

COM6:115200baud - Tera Term V1
File Edit Setup Control Window Help
Password:
Development Board          with ECUcore-E660
=====
Vendor:      SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number: 246074                          208869
Version:     1                                4001027.4295.02.00.00
Linux BSP:   V2.00.02                          Bootloader V2.00.06
Linux Kernel: V3.10.34-rt34                    SDC Firmware: V1.00.16
Reset cause: Warm reset                        SDC Flavor: Generic

sh:~# cd /home/bin/
sh:~/bin# insmod pce660drv.ko
sh:~/bin# ./iodrvdemo

*****
Test application for SYSTEC PLCcore-E660 board driver
Version: 1.00
(c) 2012-2014 SYS TEC electronic GmbH, www.systec-electronic.com
*****

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

Hardware:  CPU Board:    4295.03 (#00H)
           CPU FPGA:    0.00 (#00H)
           IO Board:    4311.02 (#02H)

IO config: Digital In:  4
           Digital Out: 4
           Analog In:   3
           Analog Out:  0
           Counter:     0
           PWM/PTO:     0
           TempSensor:  3
Driver:    Config:      0000H

FPGA interrupt selftest: successful

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
9 - Run Bus Read 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=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x00-0x02  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x00-0x04  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x00-0x08  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x00-0x10  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x00-0x20  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x00-0x40  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x00-0x80  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x01-0x00  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x02-0x00  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN
DI=0x00-0x00-0x00  D0=0x00-0x04-0x00  bHexSwitch=0x00  bDipSwitch=0x1C  R/S/M-Switch = RUN

Do you really want to exit the basic I/O main loop (y/-)?

```

Figure 65: Testing the hardware connections using "iodrvdemo"

9 Using the USB interface

9.1 Using the USB interface

The Embedded Linux used on ECUcore-E660 supports the usage of USB devices via "Hot Plug&Play". The ECUcore-E660 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-E660.

Responses to plugging and removing USB memory sticks can be flexibly adjusted by the user. "udev" 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 "/lib/udev/rules.d/81-usb.rules". The default implementation of "hotplug.sh" which is included in the scope of delivery of ECUcore-E660 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 66 illustrates the above descriptions.

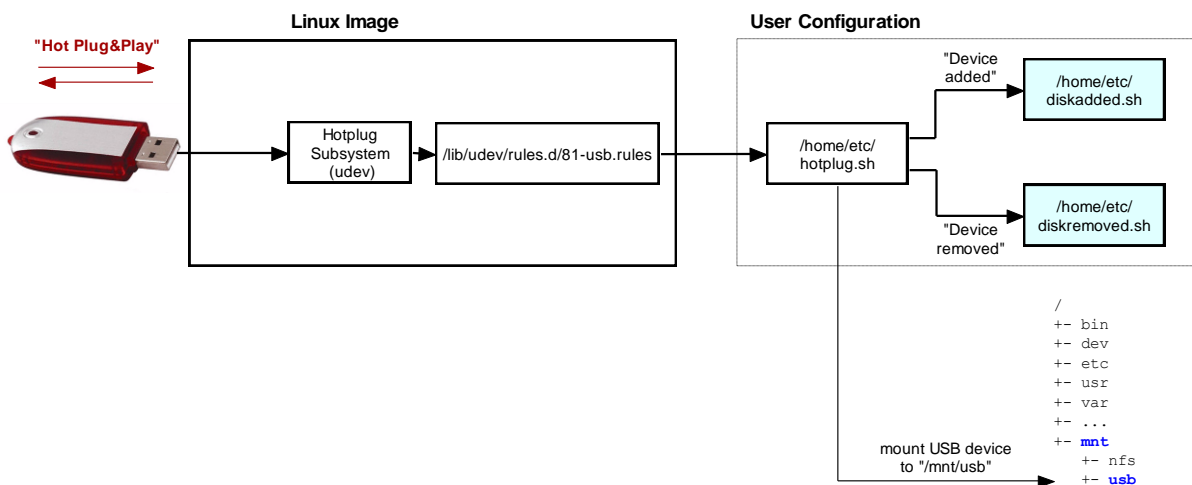


Figure 66: Internal procedures for plugging or removing USB memory sticks

The following example shows a version for the Shell script *"diskadded.sh"* which automatically lists the files of an USB memory stick when it is plugged. It performs also automatically a Shell script *"autostart"* when any is present in the root directory of the stick.

```
#!/bin/sh

echo
echo "-----"
echo "New disk drive added: $1"
echo "-----"
echo

ls -la $1

echo

if [ -f "$1/autostart" ]; then
    if [ -x "$1/autostart" ]; then
        echo "*** Running $1/autostart ***"
        cd $1
        ./autostart
    fi
fi

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

```

Telnet 192.168.10.248
Devboard_0050c2393f21 login: PlcAdmin
Password:

                Development Board                with ECUcore-E660
                =====                =====
Vendor:          SYS TEC electronic GmbH          SYS TEC electronic GmbH
Serial number:  246074                            208869
Version:        1                                 4001027.4295.02.00.00
Linux BSP:      U2.00.02                          Bootloader          U2.00.05
Linux Kernel:   U3.10.34-rt34                     SDC Firmware:      Not present
Reset cause:    SDC Flavor:                        SDC Flavor:        Unknown

sh:~# cat /var/log/hotplug.log
Tue, 29 Apr 2014 12:17:22 +0000
Event: Disk added

-----
New disk drive added: /mnt/usb
-----

drwxr-xr-x   9 root   root           16384 Jan  1  1970 .
drwxr-xr-x   5 root   root           1024 Apr 28  08:31 ..
drwxr-xr-x   3 root   root           8192 Jun 11  2013 CANopen
drwxr-xr-x   2 root   root           8192 Sep  2  2010 NiosII
-rwxr-xr-x   1 root   root              76 Apr 26  2011 autorun.inf
-rwxr-xr-x   1 root   root       1474560 Jul  8  2010 balder.img
drwxr-xr-x   5 root   root           8192 Feb  2  2013 boot
drwxr-xr-x   3 root   root           8192 Feb 14  2013 dfsee
drwxr-xr-x   2 root   root        106496 Jan 28  2013 emmc
-rwxr-xr-x   1 root   root           3158 Apr 17  2008 license.txt
-rwxr-xr-x   1 root   root       9176636 Jun 10  2009 msgothic.ttc
drwxr-xr-x   4 root   root           8192 Feb  2  2013 pmagic
-rwxr-xr-x   1 root   root       104913 May 15  2013 tek000000.png
drwxr-xr-x   8 root   root           8192 Feb  2  2013 ubcd

-----

Tue, 29 Apr 2014 12:30:58 +0000
Event: Disk removed

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

sh:~#

```

Figure 67: 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-E660 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-E660.

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-E660 and afterwards transferred onto the stick (in the background). In C/C++ programs, it is not sure, that the writing process is finished after calling "close" for closing a written file. Therefore, the syscall "fsync" is necessary in the program before the file is closed. Command "sync" must be used in case of Shell scripts.

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 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-E660. 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 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 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-E660**

Setting the time zone for the ECUcore-E660 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-E660 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	39
/home	39, 40
/home/etc/autostart	23
/home/etc/diskadded.sh	90
/home/etc/diskremoved.sh	90
/home/etc/hotplug.sh	90
/projects/ecucore-setup	64
/tmp	39
/var/	39
/var/log/hotplug.log	91
I	
\Vm-xubuntu	55
A	
Accessory	19
adduser	35
Administration	
System requirements	20
autostart	23
Autostart	23
B	
Boot configuration	23
C	
CAN driver	67
CAN1	15
chmod	71, 94
Configuration	
Command	29
Readout and display	37
Configuration Mode	26
Connection network drive	55
D	
date	36
Debugger commands	82
deluser	36
Demo project	73
CAN driver	67
I/O Driver	67
Development Board	
Connections	14
Control elements	18
Jumper configuration	16
Development Kit	13
df	39
DHCP	52
ECUcore-E660	30
Display	43, 46
Driver Development Kit	19, 87
E	
Eclipse	75
Configuring Debugger	79
Debugger commands	82
Debugging	78
Open project	76
Translating project	77
ecucore-setup	64
EFI Shell	
Activation	26
EFI Shell Command	
Ethernet Configuration	29
Embedded Linux	12
EMC law	8
Environment variables	64
Display	94
Setting	95
ETH0	15
ETH1	15
evtest	45
Execution rights	94
F	
File system	38
Files	
preinstalled	40
fstab	39
FTP	71
Login to ECUcore-E660	33
FTP client	20
FTP server	72
ftpget	71
ftpput	72
fw_printenv	37
G	
gdbserver	78
GNU	12
H	
hellocan	67
HMI Input Device	
Keyboard	43
Mouse	43
HTTP server	41
lighttpd	41
hwclock	36
I	
ifconfig	54
Input Devices	43
insmod	66
iodrvdemo	88

K

Keyboard43, 46

L

lighttpd41
 Linux12
 Linux VMware-Image.....52
 LinuxBSP85

M

make73
 Makefile64
 Manuals
 Overview9
 mount.....70
 Mouse43, 46

N

net use55
 Network environment.....54
 NFS
 mount70

P

PATH94
 Predefined user accounts
 (development system).....53
 ECUcore-E66035
 ptxdist kernelconfig85
 ptxdist menuconfig.....85
 pureftp.....33, 72

Q

Qt.....47
 Environment variables47
 Overview of Components.....47
 Program call48
 qtdemo.....47, 48
 -qws48

S

Search path94
 Setting RTC36
 Setting system time36
 setup-ECUcore-E660.sh.....41

SO-1116.exe 51
 SO-1117 87
 Socket CAN 67
 Software structure 63
 System start 23

T

TAB completion..... 94
 Telnet
 Login to ECUcore-E660 32
 Login to Linux development system..... 55
 Telnet client.....20
 Terminal program.....20
 Testing Hardware Connections 88
 Time zone95
 Toolchain 64
 TZ..... 95

U

UART1 15
 UART2 15
 USB interface.....90
 USB-RS232 Adapter Cable 19
 User accounts
 Addition and deletion..... 35
 Changing password..... 36
 predefined (development system)..... 53
 Predefined (ECUcore-E660) 35

V

VMware-Image
 Determining IP address..... 54
 Determining static IP address 59
 Installation 51
 Keyboard layout 56
 Overview 51
 Shrinking 61
 Start 52
 System update 61
 Time zone..... 58
 VMware-Player
 Installation 51

W

Windows network environment..... 54
 WinSCP 33

Document: System Manual ECUcore-E660
Document number: L-1554e_1, 1st Edition May 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: SYS TEC electronic GmbH
August-Bebel-Str. 29
D - 07973 Greiz
GERMANY
Fax : +49 (0) 36 61 / 6279-99
Email: info@systec-electronic.com

