System Manual
PLCcore-E660

User Manual
Version 1.0

Edition July 2014

Document No.: L-1555e_1

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

Status/Changes

Status: released

Date/Version	Section	Changes	Editor
2014/07/28	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, xed 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.

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.systec-electronic.com/distributors
Ordering Information:	+49 (0) 3765 38600-0 info@systec-electronic.com	
Technical Support:	+49 (0) 3765 38600-0 support@systec-electronic.com	
Fax:	+49 (0) 3765 38600-4100	
Web Site:	http://www.systec-electronic.com	

1st Edition July 2014

Table of Contents

1 Introduction 6

2 Overview / Where to find what? 7

3 Product Description 9

4 Development Kit PLCcore-E660 12
4.1 Overview 12
4.2 Electric commissioning of the Development Kit ECUcore-E660 13
4.3 Jumper configuration of the Development Kit 15
4.4 Control elements of the Development Kit ECUcore-E660 17
4.5 Optional accessory 18
4.5.1 USB-RS232 Adapter Cable 18
4.5.2 Driver Development Kit (DDK) 18

5 Pinout of the PLCcore-E660 19

6 PLC Functionality of the PLCcore-E660 24
6.1 Overview 24
6.2 System start of the PLCcore-E660 24
6.2.1 Boot-Up sequence 24
6.2.2 Reset conditions of the PLCcore-E660 Development Board 24
6.2.3 Autostart for user software 25
6.3 UEFI Bootloader and EFI Shell 26
6.3.1 UEFI-based Bootloader for PLCcore-E660 26
6.3.2 Activation of EFI Shell 27
6.3.3 Requirements and settings for the communication with the EFI Shell 30
6.4 Programming the PLCcore-E660 30
6.5 Process image of the PLCcore-E660 31
6.5.1 Local In- and Outputs 31
6.5.2 In- and outputs of user-specific baseboards 32
6.6 Communication interfaces 32
6.6.1 Serial interfaces 32
6.6.2 CAN interfaces 33
6.6.3 Ethernet interfaces 33
6.7 Control and display elements 34
6.7.1 Run/Stop Switch 34
6.7.2 Run-LED (green) 34
6.7.3 Error-LED (red) 35
6.8 Local deletion of a PLC program 35
6.9 Using CANopen for CAN interfaces 36
6.9.1 CAN interface CAN0 37
6.9.2 Additional CAN interfaces 37
6.10 Integrated Target Visualization 37

7 Configuration and Administration of the PLCcore-E660 39
7.1 System requirements and necessary software tools 39
7.2 Activation/Deactivation of Linux Autostart 40
7.3 Ethernet configuration of the PLCcore-E660 40
7.4 PLC configuration of the PLCcore-E660 41
7.4.1 PLC configuration via WEB Frontend 41
7.4.2 Setup of the configuration file "plccore-E660.cfg" 43
7.5 Boot configuration of the PLCcore-E660 45
7.6 Selecting the appropriate firmware version 46
7.7 Predefined user accounts 47
7.8 Login to the PLCcore-E660 47

7.8.1 Login to the command shell 47
7.8.2 Login to the FTP server 48
7.9 Adding and deleting user accounts 50
7.10 How to change the password for user accounts 51
7.11 Setting the system time 52
7.12 File system of the PLCcore-E660 53
7.13 Software update of the PLCcore-E660 55
7.13.1 Updating the PLC firmware 55
7.13.2 How to update the Linux Image 57

8 Adaption of In-/Outputs and Process Image 58
8.1 Data exchange via shared process image 58
8.1.1 Overview of the shared process image 58
8.1.2 API of the shared process image client 61
8.1.3 Creating a user-specific client application 65
8.1.4 Example for using the shared process image 67
8.2 Driver Development Kit (DDK) for the ECUcore-E660 71
8.3 Testing the hardware connections 72

Appendix A: Firmware function scope of PLCcore-E660 74

Appendix B: GNU GENERAL PUBLIC LICENSE 78

Index 83

List of Tables

Table 1: Overview of relevant manuals for the PLCcore-E660 7
Table 2: Connections of the Development Kit PLCcore-E660 13
Table 3: Jumpers of the Development Board for the PLCcore-E660 15
Table 4: Control elements of the Development Board for the ECUcore-E660 17
Table 5: Connections of the PLCcore-E660, completely, sorted by connection pin 20
Table 6: Connections of the PLCcore-E660, only I/O, sorted by function 22
Table 7: Coding of the Run/Stop Switch 23
Table 8: Reset management 25
Table 9: EFI commands 28
Table 10: Support of Function Block classes for different types of the PLCcore 31
Table 11: Assignment of in- and outputs to the process image of the PLCcore-E660 31
Table 12: Display status of the Run-LED 34
Table 13: Display status of the Error-LED 35
Table 14: EFI shell configuration commands of the PLCcore-E660 41
Table 15: Configuration entries of the CFG file 44
Table 16: Assignment of BoardIDs and firmware versions for the PLCcore-E660 46
Table 17: Predefined user accounts of the PLCcore-E660 47
Table 18: File system configuration of the PLCcore-E660 54
Table 19: Content of the archive files "shpimgdemo.tar.gz" 66
Table 20: Firmware functions and function blocks of PLCcore-E660 74

List of figures

Figure 1: Top view of the PLCcore-E660 9
Figure 2: Development Board PLCcore-E660 12
Figure 3: Positioning of most important connections on the Development Board for the ECUcore-E660 15
Figure 4: SYS TEC USB-RS232 Adapter Cable 18
Figure 5: Pinout of the PLCcore-E660 - top view 19
Figure 6: Memory test during Boot-Up 24
Figure 7: System start of the PLCcore-E660 25
Figure 8: UEFI boot menu 27
Figure 9: EFI Shell started 28
Figure 10: Terminal configuration using the example of "TeraTerm" 40
Figure 11: User login dialog of the WEB Frontend 42
Figure 12: PLC configuration via WEB Frontend 43
Figure 13: Calling the Telnet client in Windows 47
Figure 14: Login to the PLCcore-E660 48
Figure 15: Starting the FTP server 49
Figure 16: Login settings for WinSCP 49
Figure 17: FTP client for Windows "WinSCP" 50
Figure 18: Adding a new user account 51
Figure 19: Changing the password for a user account 52
Figure 20: Setting and displaying the system time 53
Figure 21: Display of information about the file system 54
Figure 22: File transfer in FTP client "WinSCP" 55
Figure 23: Installing PLC firmware on the PLCcore-E660 56
Figure 24: Overview of the shared process image 58
Figure 25: Unzipping the archive files shpimgdemo.tar.gz in the Linux development system 66
Figure 26: Generating the demo project "shpimgdemo" in the Linux development system 67
Figure 27: Terminal outputs of the demo program "shpimgdemo" after start 70
Figure 28: Terminal outputs of the demo program "shpimgdemo" after user inputs 71
Figure 29: Overview of the Driver Development Kit for the ECUcore-E660 72
Figure 30: Testing the hardware connections using "iodrvdemo" 73

1 Introduction

Thank you that you have decided for the SYS TEC PLCcore-E660. This product provides to you an innovative and high-capacity PLC-kernel. 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.

Please take some time to read through this manual carefully. It contains important information about the commissioning, configuration and programming of the PLCcore-E660. It will assist you in getting familiar with the functional range and usage of the PLCcore-E660. This document is complemented by other manuals, e.g. for the *OpenPCS* IEC 61131 programming system and the CANopen extension for IEC 61131-3. Table 3 in section 4.1 shows a listing of relevant manuals for the PLCcore-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 PLCcore-E660 (EMC law)

The PLCcore-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 PLCcore-E660 is ESD-sensitive and may only be unpacked, handled and used by skilled personnel at ESD-protected work stations.

The PLCcore-E660 is a module for the application in automation technology. It features IEC 61131-3 programmability, uses standard CAN-bus and Ethernet network interfaces and a standardized network protocol. Consequently, development times are short and hardware costs are reasonable. PLC-functionality is created on-board through a CANopen network layer. Hence, it is not necessary for the user to create firmware.

2 Overview / Where to find what?

The PLCcore-E660 is based on SYS TEC ECUcore-E660 hardware and is extended by PLC-specific functionality (PLC firmware, target visualization). There are different hardware manuals for all hardware components such as the ECUcore-E660 and the PLCcore-E660 (the hardware of both modules is identical), development boards and reference circuitry. Software-sided, the PLCcore-E660 is programmed with IEC 61131-3-conform *OpenPCS* programming environment. There are additional manuals for *OpenPCS* that describe the handling of programming tools and SYS TEC-specific extensions. Those are part of the software package *"OpenPCS"*. Table 1 lists up all relevant manuals for the PLCcore-E660.

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

Information about…	**In which manual?**
Basic information about the PLCcore-E660 (configuration, administration, process image, connection assignment, firmware update, reference designs et cetera)	In this manual
Development of user-specific C/C++ applications for the ECUcore-E660 / PLCcore-E660, VMware-Image of the Linux development system	System Manual ECUcore-E660 (Manual no.: L-1554)
Hardware description about the ECUcore-E660 / PLCcore-E660, reference designs et cetera	Hardware Manual ECUcore-E660 (Manual no.: L-1562)
Development Board for the ECUcore-E660 / PLCcore-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 ECUcore-E660 (Manual no.: L-1561)
Basics about the *OpenPCS* IEC 61131 programming system	Brief instructions for the programming system (Entry "*OpenPCS Documentation*" in the *OpenPCS* program group of the start menu) (Manual no.: L-1005)
Complete description of the *OpenPCS* IEC 61131 programming system, basics about the PLC programming according to IEC 61131-3	Online help about the *OpenPCS* programming system
Command overview and description of standard function blocks according to IEC 61131-3	Online help about the *OpenPCS* programming system
SYS TEC extension for IEC 61131-3: - String functions - UDP function blocks - SIO function blocks - FB for RTC, Counter, EEPROM, PWM/PTO	User Manual *"SYS TEC-specific extensions for OpenPCS / IEC 61131-3"* (Manual no.: L-1054)

CANopen extension for IEC 61131-3 (Network variables, CANopen function blocks)	User Manual *"CANopen extension for IEC 61131-3"* (Manual no.: L-1008)
HMI extension for IEC 61131-3: - HMI function blocks - Basics about Spider Control	User Manual *"SYS TEC-specific HMI extensions for OpenPCS / IEC 61131-3"* (Manual no.: L-1321)
Textbook about PLC programming according to IEC 61131-3	IEC 61131-3: Programming Industrial Automation Systems John/Tiegelkamp Springer-Verlag ISBN: 3-540-67752-6 (a short version is available as PDF on the *OpenPCS* installation CD)

Section 4 of this manual explains the **commissioning of the PLCcore-E660** based on the Development Kit for the PLCcore-E660.

Section 5 describes the **connection assignment** of the PLCcore-E660.

Section 6 explains details about the **application of the PLCcore-E660**, e.g. the **setup of the process image**, the **meaning of control elements** and it provides basic information about programming the module. Moreover, information is given about the usage of CAN interfaces related to **CANopen**.

Section 7 describes **details about the configuration of the PLCcore-E660**, e.g. the configuration of Ethernet and CAN interfaces, the Linux Autostart procedure as well as choosing the firmware version. Furthermore, the **administration of the PLCcore-E660** is explained, e.g. the login to the system, the user administration and the execution of software updates.

Section 8 defines the **adaptation of in- and outputs** as well as the **process image** and it covers the data exchange between a PLC program and a user-specific C/C++ application via **shared process image**.

3 Product Description

The PLCcore-E660 as an innovative product extends the SYS TEC electronic GmbH product range within the field of control applications. In the form of an insert-ready core module, it provides to the user a complete and compact PLC. Due to CAN and Ethernet interfaces, the PLCcore-E660 is best suitable to realize custom specific HMI (**H**uman **M**achine **I**nterface) applications.

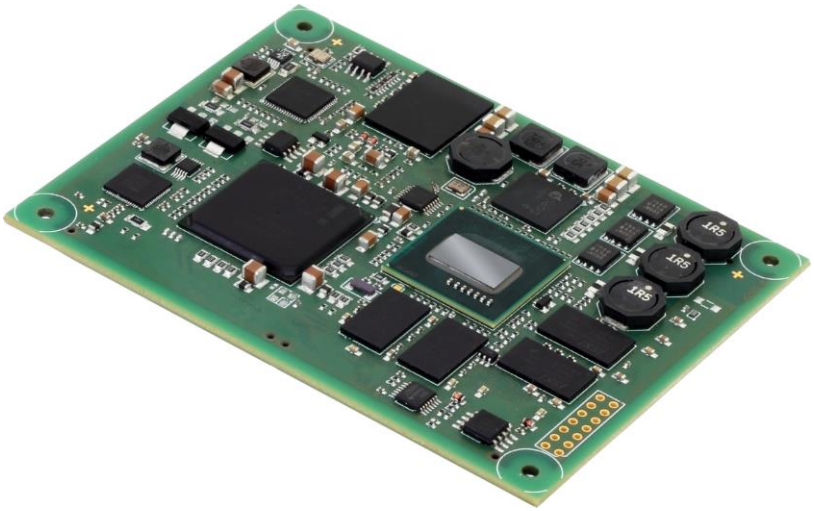


Figure 1: Top view of the PLCcore-E660

These are some significant features of the PLCcore-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/outputs, 9 digital outputs (I/Os modifiable as requested by the user)
- Externally usable SPI and I^2C
- On-board peripherals: RTC, temperature sensor, watchdog, SDC
- On-board software: Linux, PLC firmware, CANopen Master, HTTP and FTP server – **additional for HMI version only:** Target Visualization and HMI Function block library
- Programmable in IEC 61131-3 and in C/C++
- Function block libraries for communication (CANopen, Ethernet and UART)
- Support of typical PLC control elements (e.g. Run/Stop Switch, Run-LED, Error-LED)
- Linux-based (other user programs may run in parallel)
- Easy, HTML-based configuration via WEB Browser
- Remote Login via Telnet
- Small dimension (105 mm x 80 mm)

There are different types of firmware available for the PLCcore-E660. They differ regarding in the Target Visualization and in the protocol used for the communication between Programming PC and PLCcore-E660:

Art.-Nr.: 3390089/Z3:	PLCcore-E660/Z3 (RS232 without target visualization), communication with the programming PC via Siemens 3964(R) (Interface UART1)
Art.-Nr.: 3390089/Z4:	PLCcore-E660/Z4 (CANopen without target visualization), communication with the programming PC via CANopen protocol (Interface CAN0)
Art.-Nr.: 3390089/Z5:	PLCcore-E660/Z5 (Ethernet without target visualization), Kommunikation with the programming PC via UDP protocol (Interface ETH0)
Art.-Nr.: 3390090/Z3:	PLCcore-E660/Z4 (RS232 with target visualization), communication with the programming PC via Siemens 3964(R) (Interface UART1)
Art.-Nr.: 3390090/Z4:	PLCcore-E660/Z4 (CANopen with target visualization), communication with the programming PC via CANopen protocol (Interface CAN0)
Art.-Nr.: 3390090/Z5:	PLCcore-E660/Z5 (Ethernet with target visualization), communication with the programming PC via UDP-protocol (Interface ETH0)

Making PLC available as an insert-ready core module with small dimensions reduces effort and costs significantly for the development of user-specific controls. The PLCcore-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 (CANopen and UDP).

The on-board firmware of the PLCcore-E660 contains the entire Target Visualization (HMI version only, Order number 3390090) as well as the PLC runtime environment including CANopen connection with CANopen master functionality. Thus, the module is able to perform human-machine-communication as well as control tasks such as linking in- and outputs or implementing control loop algorithms on-site. Data and events can be exchanged with other nodes (e.g. superior main controller, I/O slaves and so forth) via CANopen network, Ethernet (UDP protocol) and serial interfaces (UART). Moreover, the number of in- and outputs either is locally extendable or decentralized via CANopen devices. For this purpose, the CANopen-Chip is suitable. It has also been designed as insert-ready core module for the appliance in user-specific applications.

The PLCcore-E660 provides 9 digital inputs (DI0...DI8) and 9 digital outputs (DO0...DO8). This default I/O configuration can be adapted for specific application requirements by using the Driver Development Kit (SO-1117).Saving the PLC program in the on-board Flash-Disk of the module allows an automatic restart in case of power breakdown.

Programming the PLCcore-E660 takes place according to IEC 61131-3 using the *OpenPCS* programming system of the company infoteam Software GmbH (http://www.infoteam.de). This programming system has been extended and adjusted for the PLCcore-E660 by the company SYS TEC electronic GmbH. Hence, it is possible to program the PLCcore-E660 graphically in KOP/FUB, AS and CFC or textually in IL or ST. Downloading the PLC program onto the module takes place via Ethernet or CANopen – depending on the firmware that is used. Addressing in- and outputs and creating a process image follows the SYS TEC scheme for compact control units. Like all other SYS TEC controls, the PLCcore-E660 supports backward documentation of the PLC program as well as the debug functionality including watching and setting variables, single cycles, breakpoints and single steps.

The HMI version of the PLCcore-E660 (Order number 3390085) contains an integrated Target Visualization. That is based on the *SpiderControl MicroBrowser* by the iniNet Solutions GmbH (http://www.spidercontrol.net). It enables for displaying of process values from the PLC as well as forwarding of operator actions to the PLC (e.g. mouse or keyboard).

In the standard version of the PLCcore-E660 (Order number 3390055) the display is free available for customer specific GUI applications, based on Qt.

The PLCcore-E660 is based on Embedded Linux as operating system. This allows for an execution of other user-specific programs while PLC firmware is running. If necessary, those other user-specific programs may interchange data with the PLC program via the process image. More information about this is provided in section 8.

The Embedded Linux applied to the PLCcore-E660 is licensed under GNU General Public License, version 2. Appendix B contains the license text. All sources of LinuxBSP are included in the software package **SO-1116** ("VMware-Image of the Linux development system for the ECUcore-E660"). If you require the LinuxBSP sources independently from the VMware-Image of the Linux development system, please contact our support:

support@systec-electronic.com

The PLC system and the PLC- and C/C++ programs developed by the user are **not** subject to GNU General Public License!

4 Development Kit PLCcore-E660

4.1 Overview

Due to the Development Board contained in the Kit, the Development Kit PLCcore-E660 allows for a quick commissioning of the PLCcore-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.

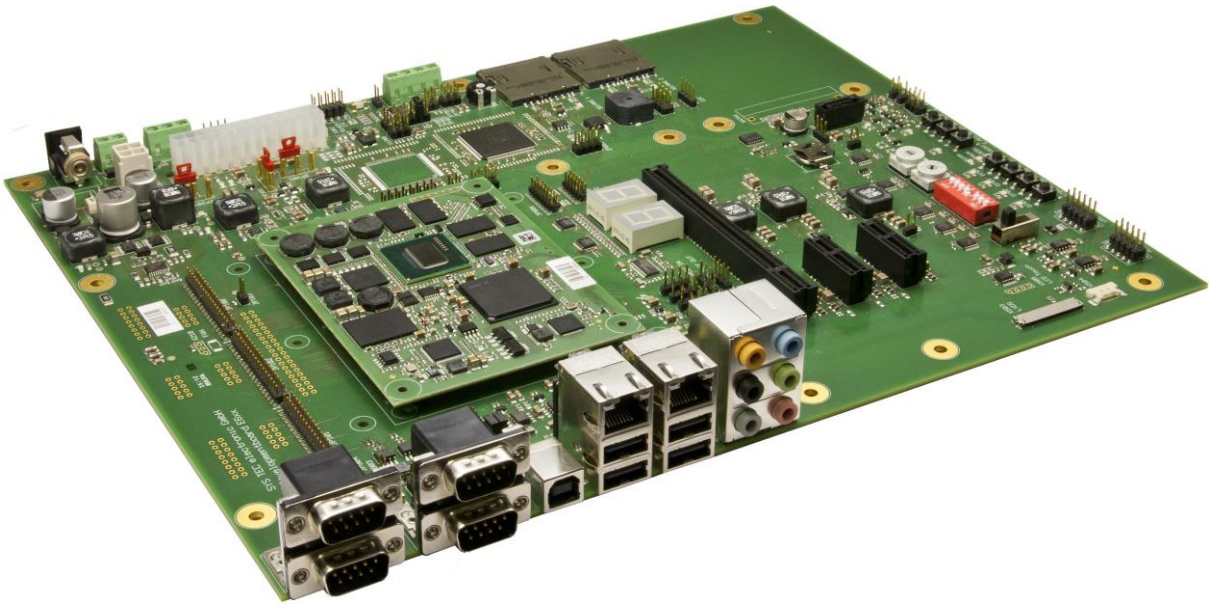


Figure 2: Development Board PLCcore-E660

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 of the document L-1554 (System Manual ECUcore-E660) 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 PLCcore-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. This alternative usage needs 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. Includes 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 document L-1554 about how to create the SD card image).
SD card slot 2	X604	Optional SD card interface. Remark: JP600 needs 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 needs to be configured.
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.

Connection	Label on the Development Board	Remark
I2C	X701	This interface can be used for user specific I2C interface. Remark: JP701 needs to be configured.
SPI	X603	This interface can be used for user specific SPI interface. Remark: JP704 needs to be configured.
AUDIO	X800	This interface can be used for user specific AUDIO interface. Remark: JP800 needs 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 needs to be configured
UART2 (RS232)	P901A Bottom	Interface can be used freely for the user program. Remark: JP909 needs 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 24 V/2 ... 4 A.

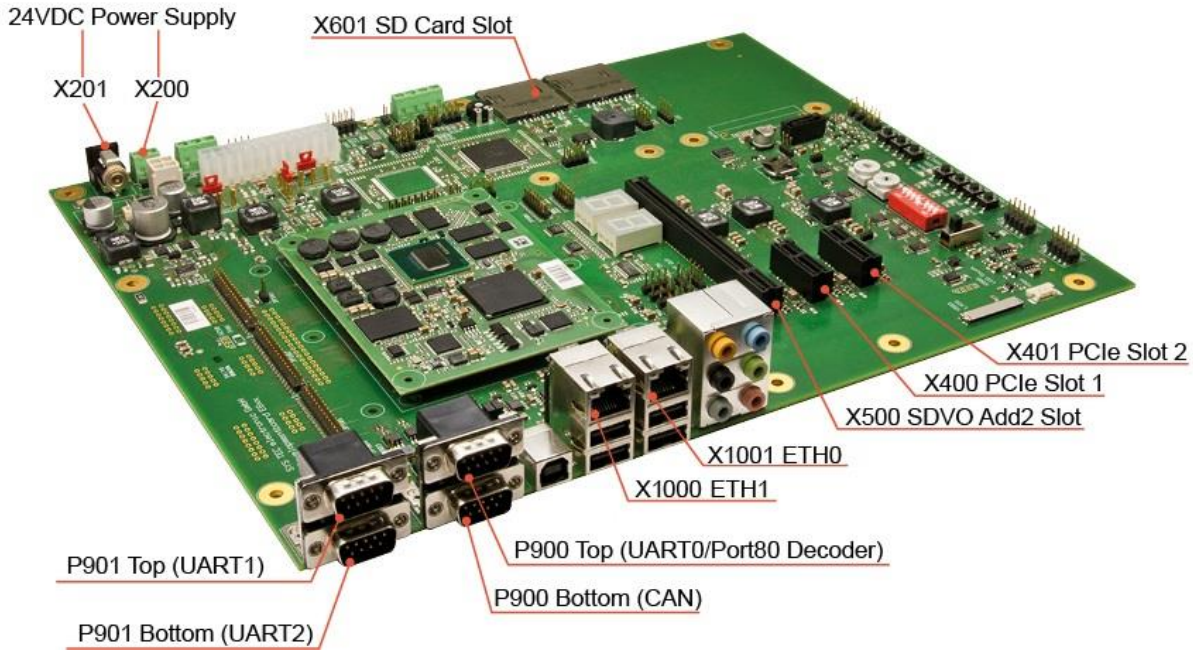


Figure 3: Positioning of most important connections on the Development Board for the ECUcore-E660

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 PLCcore-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 PLCcore-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
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)

Control element	Name	Default Option	Meaning
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: • 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
DIP-Switch	S700	Control element can be used freely for the user program Switch number 8 is the MSB of the byte coded by this switch
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 7.1).

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 8.2 provides information about the Driver Development Kit.

5 Pinout of the PLCcore-E660

Connections of the PLCcore-E660 are directed to the outside via four female headers that are double-row and mounted on the bottom of the module (X1400 bis X1403, see Figure 5).

The following types of connectors are used.

ECUcore-E660:
4 x Samtec QSE-040-01-F-D-A-K-TR (2x40pol. female)

Development Board:
4 x Samtec QTE-040-01-F-D-A-K (2x40pol. male)

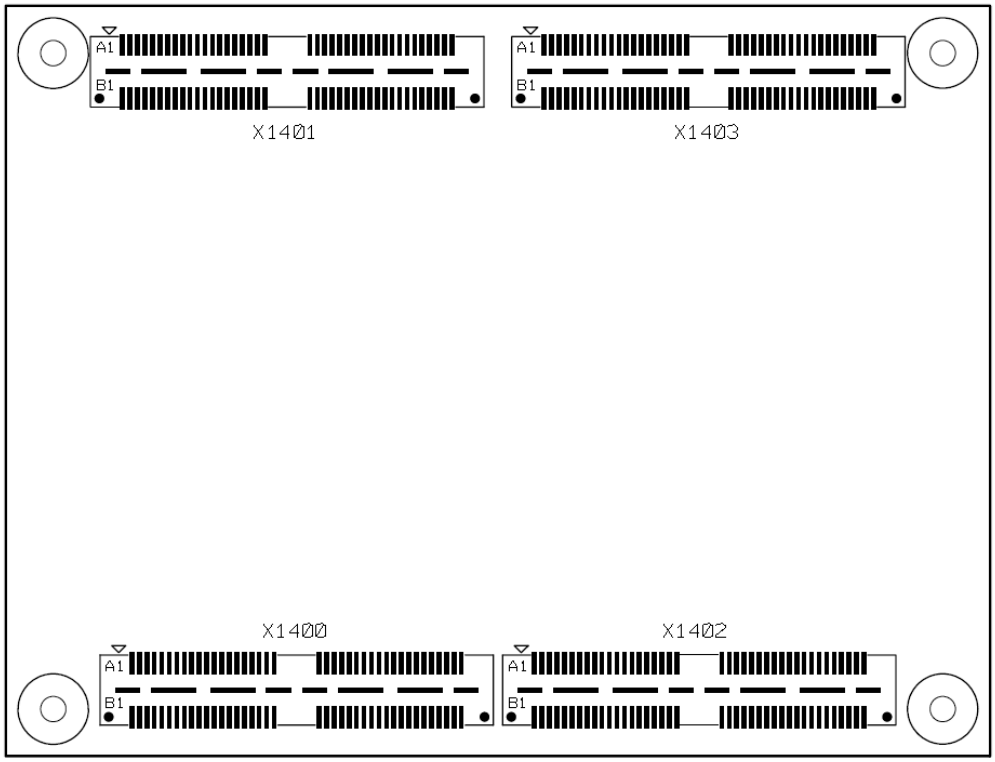


Figure 5: Pinout of the PLCcore-E660 - top view

Figure 5 exemplifies the positioning of female headers (X1400 to X1403) on the PLCcore-E660. The complete connection assignment of this module is listed up in Table 5. A detailed description of all module connectors is located in the Hardware Manual ECUcore-E660 (Manual no.: L-1562).

Table 5: Connections of the PLCcore-E660, completely, sorted by connection pin

Signal	Pin	Pin	Signal		Signal	Pin	Pin	Signal
X1400					X1401			
ETH0_MDI_3-	A1	B1	/LPC_FRAME_B		ETH1_MDI_3-	A1	B1	1V9_LAN
ETH0_MDI_3+	A2	B2	LPC_AD0		ETH1_MDI_3+	A2	B2	
GND	A3	B3	LPC_AD1		ETH1_LED0	A3	B3	CAN_RX
ETH0_MDI_2-	A4	B4	LPC_AD2		ETH1_MDI_2-	A4	B4	CAN_TX
ETH0_MDI_2+	A5	B5	LPC_AD3		ETH1_MDI_2+	A5	B5	SROM_CLK
ETH0_LED1	A6	B6	LPC_CLKOUT0		ETH1_LED1	A6	B6	/SROM_CS
ETH0_MDI_1-	A7	B7	LPC_CLKOUT1		ETH1_MDI_1-	A7	B7	SROM_DIN
ETH0_MDI_1+	A8	B8	LPC_CLKOUT2		ETH1_MDI_1+	A8	B8	SROM_DOUT
ETH0_LED2	A9	B9	LPC_SERIRQ		ETH1_LED2	A9	B9	NC_SI_CLK_IN
ETH0_MDI_0-	A10	B10	SMB_CLK_EXT		ETH1_MDI_0-	A10	B10	NC_SI_TX_EN
ETH0_MDI_0+	A11	B11	SMB_DATA_EXT		ETH1_MDI_0+	A11	B11	NC_SI_CRS_DV
GND	A12	B12	/SMB_ALERT_EXT		SDIO1_PWR_EN	A12	B12	NC_SI_RXD0
SATA0_TX_+	A13	B13	SATA1_TX_+		SDIO1_LED	A13	B13	NC_SI_TCD0
SATA0_TX_-	A14	B14	SATA1_TX_-		SDIO1_D0	A14	B14	NC_SI_RXD1
SATA0_RX_+	A15	B15	SATA1_RX_+		SDIO1_D1	A15	B15	NC_SI_TXD1
SATA0_RX_-	A16	B16	SATA1_RX_-		SDIO1_D2	A16	B16	SDIO1_
SATA0_LED	A17	B17	SATA0_LED		SDIO1_D3	A17	B17	SDIO1_
HDA_SYNC	A18	B18	HDA_SDI1		SDIO1_CLK	A18	B18	SDIO1_
/HDA_RST	A19	B19	/HDA_SDI0		SDIO1_CMD	A19	B19	SDIO1_
HDA_CLK	A20	B20	HDA_DOCKEN		SDIO1_CARD_WP	A20	B20	/SDIO1_CARD_DET
GND	GND1				GND	GND1		
HDA_SDO	A21	B21	HDA_DOCKRST		SPI2_CS	A21	B21	SPI2_CLK
USB_H4_POW_EN	A22	B22	USB_H5_POW_EN		SDC_CLK	A22	B22	SPI2_MISO
/USB_H4_OVC	A23	B23	/USB_H5_OVC		SDC_GPIO_1	A23	B23	SPI2_MOSI
USB_H4_-	A24	B24	USB_H5_-		SDC_GPIO_2	A24	B24	SDC_NJTRST
USB_H4_+	A25	B25	USB_H5_+		SDC_GPIO_3	A25	B25	SDC_TDO/TRACESWO
USB_H2_POW_EN	A26	B26	USB_H3_POW_EN		SDC_GPIO_4	A26	B26	SDC_TDI
/USB_H2_OVC	A27	B27	/USB_H3_OVC		SDC_GPIO_5	A27	B27	SDC_TMS/SWDIO
USB_H2_-	A28	B28	USB_H3_-		SDC_GPIO_6	A28	B28	SDC_TCK/SWCLK
USB_H2_+	A29	B29	USB_H3_+		SDC_GPIO_7	A29	B29	SDC_GPIO_15
USB_H0_POW_EN	A30	B30	USB_H1_POW_EN		SDC_GPIO_8	A30	B30	SDC_GPIO_16
/USB_H0_OVC	A31	B31	/USB_H1_OVC		SDC_GPIO_9	A31	B31	SDC_GPIO_17
USB_H0_-	A32	B32	USB_H1_-		SDC_GPIO_10	A32	B32	SDC_GPIO_18
USB_H0_+	A33	B33	USB_H1_+		SDC_GPIO_11	A33	B33	SDC_GPIO_19
USB_D_-	A34	B34	I2C_CLK		SDC_GPIO_12	A34	B34	SDC_GPIO_20
USB_D_+	A35	B35	I2C_DATA		SDC_GPIO_13	A35	B35	SDC_GPIO_21
/COLD_RESET	A36	B36	PWROK		SDC_GPIO_14	A36	B36	SDC_GPIO_22
ATOM_WD	A37	B37	SPEAKER		SDC_USART1_RX	A37	B37	SDC_GPIO_23
/CPU_THERMTRIP	A38	B38	/CPU_OVT		SDC_USART1_TX	A38	B38	SDC_GPIO_24
/WARM_RESET	A39	B39	/CPU_RESET		SDC_CANRX	A39	B39	SDC_GPIO_25
VBAT_RTC	A40	B40	/CPU_PROCHOT		SDC_CANTX	A40	B40	SDC_GPIO_26
GND	GND2				GND	GND2		

Signal	Pin	Pin	Signal		Signal	Pin	Pin	Signal
X1402					X1403			
PCIE_TX3_+	A1	B1	PCIE_RX3_+		SDC_BOOT0	A1	B1	SDC_GPIO_27
PCIE_TX3_-	A2	B2	PCIE_RX3_-		SDC_BOOT1	A2	B2	SDC_GPIO_28
PCIE_TX2_+	A3	B3	PCIE_RX2_+		----------	A3	B3	/CPU_WAKE
PCIE_TX2_-	A4	B4	PCIE_RX2_-		----------	A4	B4	/MR
USER_PCIE_CLK_+	A5	B5	SDIO0_PWR_EN		IO_PWROK	A5	B5	ADC_IN_2
USER_PCIE_CLK_-	A6	B6	SDIO0_LED		----------	A6	B6	ADC_IN_3
SDIO0_CARD_WP	A7	B7	/SDIO0_CARD_DET		H_GPIO0	A7	B7	A_GPIO_0
SDIO0_D0	A8	B8	SDIO0_D4		H_GPIO1	A8	B8	A_GPIO_1
SDIO0_D1	A9	B9	SDIO0_D5		H_GPIO2	A9	B9	A_GPIO_2
SDIO0_D2	A10	B10	SDIO0_D6		H_GPIO3	A10	B10	A_GPIO_3
SDIO0_D3	A11	B11	SDIO0_D7		H_GPIO4	A11	B11	A_GPIO_4
SDIO0_CLK	A12	B12	SDIO0_CMD		H_GPIO5	A12	B12	A_GPIO_SUS_0
LVDS_DATA0_+	A13	B13	SDVO0_RED_+		H_GPIO6	A13	B13	A_GPIO_SUS_1
LVDS_DATA0_-	A14	B14	SDVO0_RED_-		H_GPIO7	A14	B14	A_GPIO_SUS_2
LVDS_DATA1_+	A15	B15	SDVO0_GREEN_+		H_GPIO8	A15	B15	A_GPIO_SUS_3
LVDS_DATA1_-	A16	B16	SDVO0_GREEN_-		H_GPIO9	A16	B16	A_GPIO_SUS_4
LVDS_DATA2_+	A17	B17	SDVO0_BLUE_+		H_GPIO10	A17	B17	A_GPIO_SUS_5
LVDS_DATA2_-	A18	B18	SDVO0_BLUE_-		H_GPIO11	A18	B18	A_GPIO_SUS_6
LVDS_DATA3_+	A19	B19	SDVO0_INT_+		----------	A19	B19	A_GPIO_SUS_7
LVDS_DATA3_-	A20	B20	SDVO0_INT_-			A20	B20	A_GPIO_SUS_8
GND	GND1				GND	GND1		
LVDS_CLK_+	A21	B21	SDVO_CLK_+		HUB_TCK	A21	B21	ETH_TCK
LVDS_CLK_-	A22	B22	SDVO_CLK_-		HUB_TMS	A22	B22	ETH_TMS
SPI_MISO	A23	B23	SDVO_TVCLKIN_+		HUB_TDI	A23	B23	ETH_TDI
SPI_CLK	A24	B24	SDVO_TVCLKIN_-		HUB_TDO	A24	B24	ETH_TDO/WAKE
SPI_MOSI	A25	B25	SDVO_STALL_+		/HUB_TRST	A25	B25	IO_TCK
/SPI_CS	A26	B26	SDVO_STALL_-		HUB_RTCK	A26	B26	IO_TMS
UART0_TX	A27	B27	SDVO_CTRLCLK		CPU_TCK	A27	B27	IO_TDI
UART0_RX	A28	B28	SDVO_CTRLDATA		CPU_TMS	A28	B28	IO_TDO
UART1_TX	A29	B29	UART0_CTS		CPU_TDI	A29	B29	/IO_TRST
UART1_RX	A30	B30	UART0_DCD		CPU_TDO	A30	B30	NC_TCK
UART2_TX	A31	B31	UART0_DSR		/CPU_TRST	A31	B31	NC_TMS
UART2_RX	A32	B32	UART0_DTR		DS_TCK	A32	B32	NC_TDI
UART3_TX	A33	B33	UART0_RI		DS_TMS	A33	B33	IO_TDO
UART3_RX	A34	B34	UART0_RTS		DS_TDI	A34	B34	----------
VCC_5V0	A35	B35	VCC_5V0		DS_TDO	A35	B35	----------
	A36	B36			VCC_5V0	A36	B36	VCC_5V0
	A37	B37				A37	B37	
	A38	B38				A38	B38	
	A39	B39				A39	B39	
	A40	B40				A40	B40	
GND	GND2				GND	GND2		

Table 6 is a subset of Table 5 and only includes all in- and outputs of the PLCcore-E660 sorted by their function.

Table 6: Connections of the PLCcore-E660, only I/O, sorted by function

Name	Function
SDC_BOOTx	for selecting boot mode of System Diagnostic Controller
/MR	manual reset input of module
/CPU_RESET	reset input signal for the CPU
SATAx_TX+, SATAx_TX-, SATAx_RX+, SATAx_RX-, SATAx_LED	SATA 0, 1 with Busy-LED
ATOM_WD	watchdog input for whole core module
LPC_xyz	LPC Interface Signals . (further information at Intel® AtomE6xx Series Datasheet)
HDA_xyz	Connectors for Intel® HD Audio interfaces. (further information at Intel® AtomE6xx Series Datasheet)
USB_Hx_POW_EN	Power enable signal for USB Host (0 to 5) interfaces
/USB_Hx_ÖVC	USB Host (0 to 5) overcurrent indication
USB_Hx_+, USB_Hx_-	USB Host (0 to 5) differential bus signals
PCIE_TXx_+, PCIE_TXx_-, PCIE_RXx_+, PCIE_RXx_-	PCIExpress (lane 2 and 3) differential bus signals
USER_PCIE_CLK_+, USER_PCIE_CLK_-	PCIExpress differential clock signal
SMB_CLK_EXT, SMB_DATA_EXT, /SMB_ALERT_EXT	SMBus Interface Signals
UARTx_TX, UARTx_RX,	UART 0,1,2,3
UART0_CTS, UART0_DCD, UART0_DSR, UART0_DTR, UART0_RI, UART0_RTS	UART 0 handshake signals
SDIOx_CMD, SDIOx_CLK, SDIOx_DATA,	SD-Card 0,1 interface pins
/SDIOx_CARD_DET	SD-Card 0,1 card detect pins
SDIOx_LED	SD-Card 0,1 busy led
SDIOx_CARD_WP	SD-Card 0,1 Wake-up card signal
I2C2_DAT, I2C2_CLK	two wire interface
SDVO_xyz	Serial Digital Video Output . (further information at Intel® Atom E6xx Series Datasheet)
SDC_GPIO_x	GPIOs provided by STM32 System Diagnostic Controller (SDC)
A_GPIO_SUS_x, A_GPIO_x	GPIOs provided by Intel Atom E660T processor
H_GPIO_x	GPIOs provided by Intel EG20T hub IC
NC_SI_xyz	Pins for Network Controller Sideband Interface
SDC_USART1_RX, SDC_USART1_TX	UART of SDC
SPIx_CLK, SPIx_MISO, SPIx_MOSI	SPI 1,2 clock and data signals
SROM_DIN, SROM_DOUT, /SROM_CS, SROM_CLK	SROM Interface of EG20T Hub
VBAT	backup battery input (3,3V) for RTC
ETHx_TX-, TX+, RX-, RX+; LEDx	Ethernet-interfaces with LEDs
SPEAKER	PWM driven beeper output
HUB_TCK,HUB_TMS, HUB_TDI, HUB_TDO, /HUB_TRST, HUB_RTCK	JTAG interface for EG20T hub
ETH_TCK, ETH_TMS, ETH_TDI, ETH_TDO	JTAG interface for Intel Ethernet Controller
IO_TCK, IO_TMS, IO_TDI, IO_TDO, /IO_TRST	IO JTAG of Atom E660T
NC_TCK, NC_TMS, NC_TDI, NC_TDO	JTAG for internal network controller of Atom E660T
CPU_TCK, CPU_TMS, CPU_TDI, CPU_TDO, CPU_TRST_B	JTAG of CPU Atom E660T
SDC_TCK, SDC_TMS, SDC_TDI, SDC_TDO, SDC_NJTRST	JTAG for system diagnostic controller (SDC)
DS_TCK, DS_TMS, DS_TDI, DS_TDO	JTAG of Reset -/ Power Controller
+3V3	3,3V-supply (about 820mA)
GND	Signal ground

Table 7 defines the coding of the Run/Stop Switch. Functionality of the Run/Stop Switch for PLC firmware is explained in section 6.7.1. If no Run/Stop Switch is intended for the usage of the PLCcore-E660 on an application-specific baseboard, the coding for *"Run"* must be hard-wired at the module connections.

Table 7: Coding of the Run/Stop Switch

Mode	Run: X1403/A18 (H_GPIO11)	MRes: X1403/A10 (H_GPIO3)	Stop: X1403/A11 (H_GPIO4)
Run	1	1	0
Stop	0	1	1
MRes	0	0	0

6 PLC Functionality of the PLCcore-E660

6.1 Overview

The PLCcore-E660 realizes a complete Linux-based compact PLC as an insert-ready core ("Core"). There, the PLCcore-E660 is based on the hardware ECUcore-E660 and extends it by PLC-specific functionality (PLC firmware, Target Visualization). Both modules, the ECUcore-E660 and the PLCcore-E660, use the same Embedded Linux as operating system. Consequently, the configuration and the C/C++ programming of the PLCcore-E660 are almost identical with the ECUcore-E660.

6.2 System start of the PLCcore-E660

6.2.1 Boot-Up sequence

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

Step 1

After the PLCcore leaves the Reset state (conditions – see section 6.2.2 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 0.

Without a Memory test abort or after leaving the UEFI boot menu by a boot command (see section 0) the operating system Linux will be booted in the boot order as configured for the UEFI Bootloader (see section 6.3.1.2). 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 7.8.1).

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.

6.2.2 Reset conditions of the PLCcore-E660 Development Board

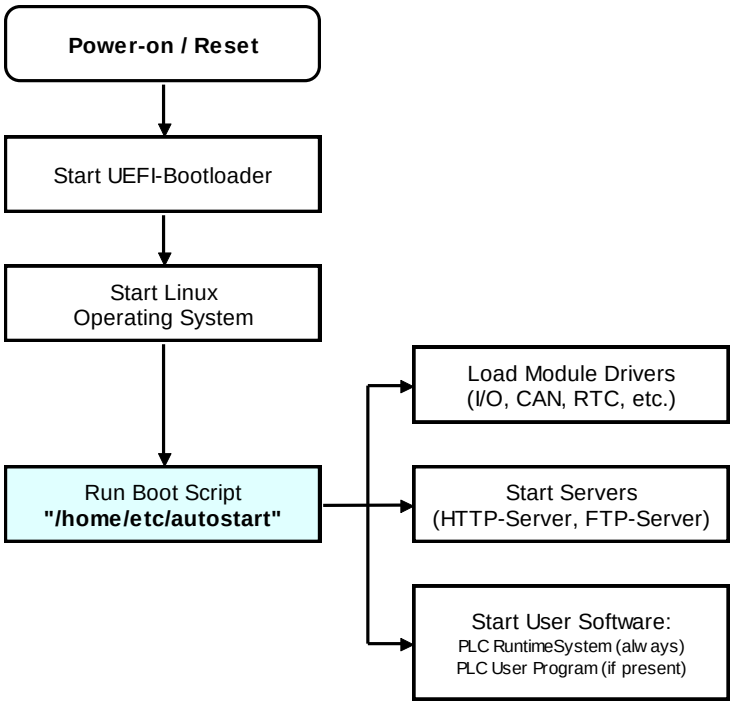
There are in principle 4 possibilities to trigger a Reset/Reboot on the PLCcore-E660 Development Board. They are described in Table 8 as follows.

Table 8: 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

6.2.3 Autostart for user software

By default, the PLCcore-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 PLCcore-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:



For more details on how to deactivate the autarchic Linux start and to activate the “UEFI shell" command prompt compare section 6.2.1.

Details about the programming of the PLCcore-E660 are covered in section 6.4.

Figure 7: System start of the PLCcore-E660

It is possible to configure the PLCcore-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 PLCcore-E660. The script can be edited directly on the PLCcore-E660 in the FTP client *"WinSCP"* (see section 7.1) using pushbuttons *"F4"* or *"F4 Edit"*.

6.3 UEFI Bootloader and EFI Shell

6.3.1 UEFI-based Bootloader for PLCcore-E660

6.3.1.1 Features

- Based on Intel Bootloader Development Kit phase II
- ACPI support including CPU frequency scaling
- Boot media:
  - SD card, eMMC
  - USB hard disk or thumb drive
  - SATA disk
- Partition layouts on boot media:
  - GUID Partition Table (GPT)
  - Master Boot Record (MBR)
- Supported file systems on boot media
  - FAT16, FAT32: reading and writing
  - 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.

6.3.1.2 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.

6.3.1.3 How to create a boot option

Please refer to the respective sections in the document L-1554.

6.3.1.4 How to read out the system identification

Please refer to the respective sections in the document L-1554.

6.3.2 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 7.3), it is necessary to start the embedded EFI Shell of the UEFI bootloader on the serial interface UART1 of the PLCcore-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*).

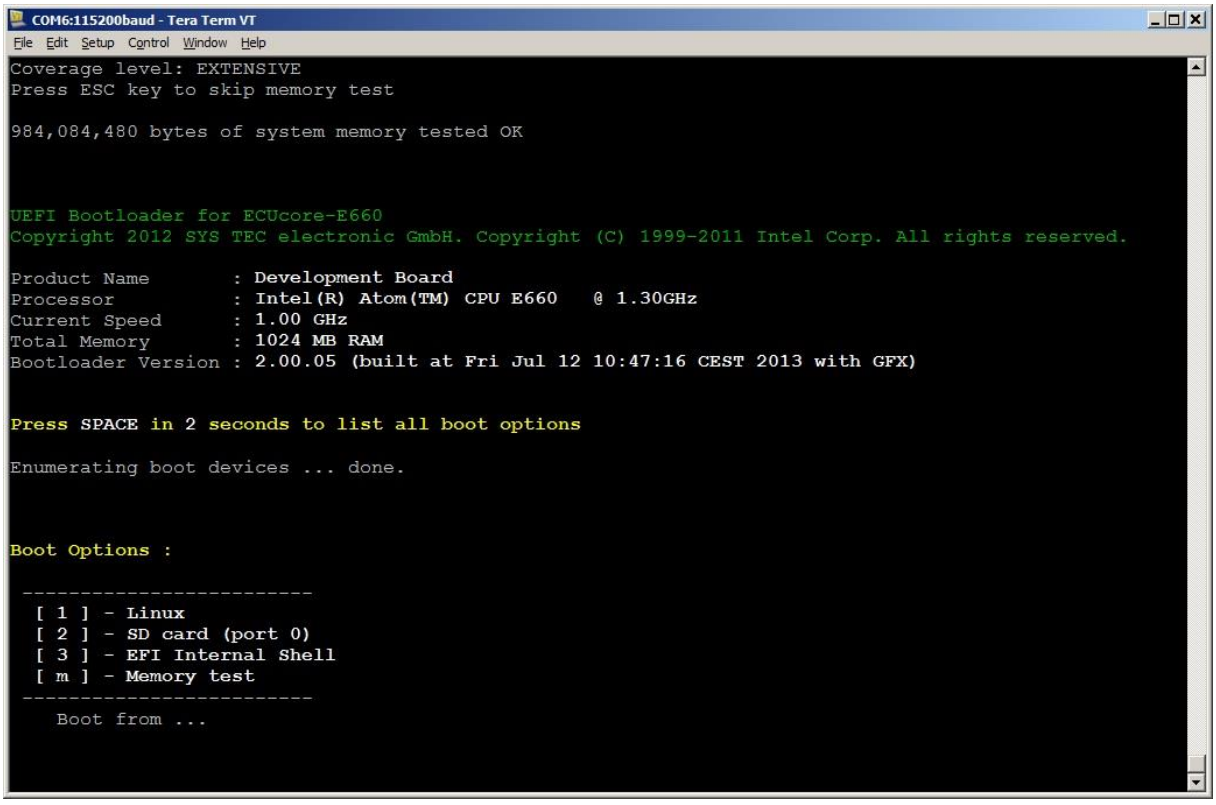


Figure 8: UEFI boot menu

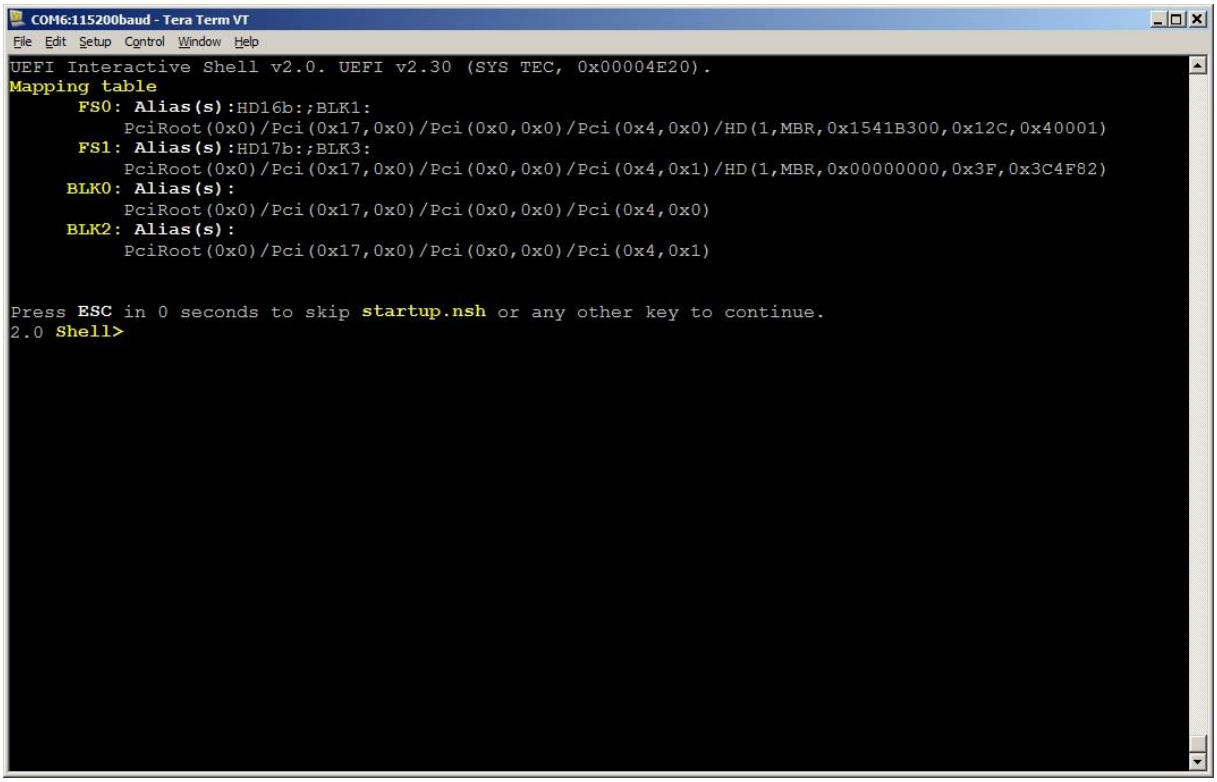


Figure 9: EFI Shell started

Once the EFI Shell is active, a set of commands shown in Table 9 can be performed.

Table 9: 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.
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.
sermode	Sets serial port attributes.

EFI command	Action performed by the command
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.

A help text regarding the usage of the EFI internal commands in Table 9 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 9 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.

6.3.3 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 7.2

6.4 Programming the PLCcore-E660

The PLCcore-E660 is programmed with IEC 61131-3-conform *OpenPCS* programming environment. There exist additional manuals about *OpenPCS* that describe the handling of this programming tool. Those are part of the software package *"OpenPCS"*. All manuals relevant for the PLCcore-E660 are listed in Table 1.

PLCcore-E660 firmware is based on standard firmware for SYS TEC`s compact control units. Consequently, it shows identical properties like other SYS TEC control systems. This affects especially the process image setup (see section 6.5) as well as the functionality of control elements (Hex-Encoding switch, DIP-Switch, Run/Stop Switch, Run-LED, Error-LED).

Depending on the firmware version used, PLCcore-E660 firmware provides numerous function blocks to the user to access communication interfaces. Table 10 specifies the availability of FB communication classes (SIO, CAN, UDP) for different PLCcore-E660 firmware versions. Section 7.6 describes the selection of the appropriate firmware version.

Table 10: Support of Function Block classes for different types of the PLCcore

Type of Interface	PLCcore-E660/Z3 Art. no: 3390089/Z3 3390090/Z3	PLCcore-E660/Z4 Art. no: 3390089/Z4 3390090/Z4	PLCcore-E660/Z5 Art. no: 3390089/Z5 3390090/Z5	Remark
CAN	-	x	x	FB description see manual L-1008
UDP	-	x	x	FB description see manual L-1054
SIO	x	x	x	FB description see manual L-1054
HMI	x (only 3390085)	x (only 3390085)	x (only 3390085)	FB description see manual L-1321

6.5 Process image of the PLCcore-E660

6.5.1 Local In- and Outputs

Compared to other SYS TEC compact control systems, the PLCcore-E660 obtains a process image with identical addresses. All in- and outputs listed in Table 11 are supported by the PLCcore-E660.

Table 11: Assignment of in- and outputs to the process image of the PLCcore-E660

I/O of the PLCcore-E660	Address and Data type in the Process Image	
DI0 ... DI7	**%IB0.0**	as Byte with DI0 … DI7
	%IX0.0 … %IX0.7	as single Bit for each input
DI8 ... DI15	**%IB1.0**	as Byte with DI8 … DI15
	%IX1.0 … %IX1.7	as single Bit for each input
DI16 … DI23 (as user specific extension only)	**%IB2.0**	as Byte with DI16 … DI23
	%IX2.0 … %IX2.7	as single Bit for each input
DI24 ... DI31 (as user specific extension only)	**%IB3.0**	as Byte with DI24 … DI31
	%IX3.0 … %IX3.7	as single Bit for each input
DI32 ... DI39 (as user specific extension only)	**%IB4.0**	as Byte with DI32 … DI139
	%IX4.0 … %IX4.7	as single Bit for each input
DI40 … DI47 (as user specific extension only)	**%IB5.0**	as Byte with DI40 … DI47
	%IX5.0 … %IX5.7	as single Bit for each input
On-board Temperature Sensor, see (1)	**%ID72.0**	31Bit + sign as 1/10000 °C

DO0 ... DO7	**%QB0.0** %QX0.0 … %QX0.7	as Byte with DO0 … DO7 as single Bit for each output
DO8 … DO15 (as user specific extension only)	**%QB1.0** %QX1.0 … %QX1.7	as Byte with DO8 … DO15 as single Bit for each output
DO16 … DO23 (as user specific extension only)	**%QB2.0** %QX2.0 … %QX2.7	as Byte with DO16 … DO23 as single Bit for each output
DO24 … DO31 (as user specific extension only)	**%QB3.0** %QX3.0 … %QX3.7	as Byte with DO24 … DO31 as single Bit for each output
DO32 … DO39 (as user specific extension only)	**%QB4.0** %QX4.0 … %QX4.7	as Byte with DO32 … DO39 as single Bit for each output
DO40 … DO47 (as user specific extension only)	**%QB5.0** %QX5.0 … %QX5.7	as Byte with DO40 … DO47 as single Bit for each output

(1) This marked components are only available in the process image, if the **Option *"Enable extended I/Os"*** is activated within the PLC configuration (see section 7.4.1). Alternatively, entry *"EnableExtIo="* can directly be set within section *"[ProcImg]"* of the configuration file ***"/home/plc/plccore-e660.cfg"*** (see section 7.4.2). The appropriate configuration setting is evaluated upon start of the PLC firmware.

In- and outputs of thePLCcore-E660 are not negated in the process image. Hence, the H-level at one input leads to value "1" at the corresponding address in the process image. Contrariwise, value "1" in the process image leads to an H-level at the appropriate output.

6.5.2 In- and outputs of user-specific baseboards

The connection lines leading towards the outside provides to the user most effective degrees of freedom for designing the in-/output circuit of the PLCcore-E660. Therewith, all in- and outputs of the PLCcore-E660 can be flexibly adjusted to respective requirements. This implicates that the process image of PLCcore-E660 is significantly conditioned by the particular, user-specific in-/output circuit. Including the software for in-/output components into the process image requires the "*Driver Development Kit for ECUcore-E660*" (order number SO-1117).

6.6 Communication interfaces

6.6.1 Serial interfaces

The PLCcore-E660 features 4 serial interfaces (UART0 … UART3) that function as RS-232. The interface UART3 can be used alternatively in RS-485 mode. Details about hardware activation are included in the *"Hardware Manual Development Board ECUcore-E660"* (Manual no.: L-1562).

UART0: The interface UART0 normally cannot be used . (see Table 2)

UART1: The interface UART1 primarily serves as service interface to administer the PLCcore-E660. By default, in boot script *"/etc/inittab"* it is assigned to the Linux process *"getty"* and is used as Linux console to administer the PLCcore-E660. Even though interface UART1 may be used from a PLC program via function blocks of type *"SIO_Xxx"* (see manual *"SYS TEC-specific Extensions for OpenPCS / IEC 61131-3"*, Manual no.: L-1054), only signs should be output in this regard. The module tries to interpret and to execute signs that it receives as Linux commands.

To freely use an interface from a PLC program, boot script *"/etc/inittab"* must be adjusted appropriately which is only possible by modifying the Linux image. This requires software package SO-1116 ("VMware-Image of the Linux Development System for the ECUcore-E660").

UART2: The interface UART2 may be used from a PLC program via function blocks of type *"SIO_Xxx"* (see manual *"SYS TEC-specific Extensions for OpenPCS / IEC 61131-3"*, Manual no.: L-1054).

To freely use an interface from a PLC program, boot script *"/etc/inittab"* must be adjusted appropriately which is only possible by modifying the Linux image. This requires software package SO-1116 ("VMware-Image of the Linux Development System for the ECUcore-E660").

UART3: The interface UART3 may be used from a PLC program via function blocks of type *"SIO_Xxx"* (see manual *"SYS TEC-specific Extensions for OpenPCS / IEC 61131-3"*, Manual no.: L-1054).

To freely use an interface from a PLC program, boot script *"/etc/inittab"* must be adjusted appropriately which is only possible by modifying the Linux image. This requires software package SO-1116 ("VMware-Image of the Linux Development System for the ECUcore-E660").

6.6.2 CAN interfaces

The PLCcore-E660 features 2 CAN interfaces (CAN0, CAN1). Details about the hardware activation are included in the *"Hardware Manual Development Board ECUcore-E660"* (Manual no.: L-1562).

The CAN interface CAN0 allows for data exchange with other devices via network variables and they are accessible from a PLC program via function blocks of type *"CAN_Xxx"* (see section 6.9 and *"User Manual CANopen Extension for IEC 61131-3"*, Manual no.: L-1008).

Section 6.9 provides detailed information about the usage of the CAN interface in connection with CANopen.

6.6.3 Ethernet interfaces

The PLCcore-E660 features 2 Ethernet interfaces (ETH0, ETH1). Details about the hardware activation are included in the *"Hardware Manual Development Board ECUcore-E660"* (Manual no.: L-1562).

The Ethernet interface ETH0 serves as service interface to administer the PLCcore-E660 and it enables data exchange with other devices. The interface is accessible from a PLC program via function blocks of type *"LAN_Xxx"* (see manual *"SYS TEC-specific Extensions for OpenPCS / IEC 61131-3"*, Manual no.: L-1054).

The exemplary PLC program *"UdpRemoteCtrl"* illustrates the usage of function blocks of type *"LAN_Xxx"* within a PLC program.

The interface ETH1 is unused under OpenPCS on the PLCcore-E660.

6.7 Control and display elements

6.7.1 Run/Stop Switch

The Module connections *"X1403/A10", " X1403/A11" and " X1403/A18"* (see Table 7) are designed to connect a Run/Stop Switch. Using this Run/Stop Switch makes it possible to start and interrupt the execution of the PLC program. Together with start and stop pushbuttons of the *OpenPCS* programming environment, the Run/Stop Switch represents a "logical" AND-relation. This means that the PLC program will not start the execution until the local Run/Stop Switch is positioned to *"Run"* **AND** additionally the start command (cold, warm or hot start) is given by the *OpenPCS* user interface. The order hereby is not relevant. A run command given by *OpenPCS* while at the same time the Run/Stop Switch is positioned to *"Stop"* is visible through quick flashing of the Run-LED (green).

Positioned to *"MRes"* (*"Modul Reset"*), the Run/Stop Switch allows for local deletion of a PLC program from the PLCcore-E660. This might for example be necessary if an error occurs and the PLC program is running an infinite loop and consequently, accessing the *OpenPCS* programming environment is no longer possible. The procedure for deleting a PLC program is described in section 6.8.

6.7.2 Run-LED (green)

The Run-LED on the Development Board provides information about the activity state of the control system. The activity state is shown through different modes:

Table 12: Display status of the Run-LED

LED Mode	PLC Activity State
Off	The PLC is in state *"Stop"*: • the PLC does not have a valid program, • the PLC has received a stop command from the *OpenPCS* programming environment or • the execution of the program has been canceled due to an internal error
Quick flashing in relation 1:8 to pulse	The PLC is on standby but is not yet executing: • The PLC has received a start command from the *OpenPCS* programming environment but the local Run/Stop Switch is still positioned to *"Stop"*
Slow flashing in relation 1:1 to pulse	The PLC is in state *"Run"* and executes the PLC program.
Quick flashing in relation 1:1 to pulse	The PLC is in mode *"Reset"*, compare section 6.8

6.7.3 Error-LED (red)

The Error-LED on the Development Board provides information about the error state of the control system. The error state is represented through different modes:

Table 13: Display status of the Error-LED

LED Mode	PLC Error State
Off	No error has occurred; the PLC is in normal state.
Permanent light	A severe error has occurred: • The PLC was started using an invalid configuration (e.g. CAN node address 0x00) and had to be stopped or • A severe error occurred during the execution of the program and caused the PLC to independently stop its state *"Run"* (division by zero, invalid Array access, ...), see below
Slow flashing in relation 1:1 to pulse	A network error occurred during communication to the programming system; the execution of a running program is continued. This error state will be reset independently by the PLC as soon as further communication to the programming system is successful.
Quick flashing in relation 1:1 to pulse	The PLC is in mode *"Reset"*, compare section 6.8.
Quick flashing in relation 1:8 to pulse	The PLC is on standby, but is not yet running: • The PLC has received a start command from the *OpenPCS* programming environment but the local Run/Stop Switch is positioned to *"Stop"*

In case of severe system errors such as division by zero of invalid Array access, the control system passes itself from state *"Run"* into state *"Stop"*. This is recognizable by the permanent light of the Error-LED (red). In this case, the error cause is saved by the PLC and is transferred to the computer and shown upon next power-on.

6.8 Local deletion of a PLC program

If the Run/Stop Switch is positioned to *"MRes"* (*"Modul Reset"*) (see section 6.7.1), it is possible to delete a program from the PLCcore-E660. This might for example be necessary if an error occurs and the PLC program is running an infinite loop and consequently, accessing the *OpenPCS* programming environment is no longer possible. To prevent deleting a PLC program by mistake, it is necessary to keep to the following order:

(1) Position the Run/Stop Switch to *"MRes"*

(2) Reset the PLCcore-E660 (by pressing the reset pushbutton of the Development Board or through temporary power interrupt)
⇨ Run-LED (green) is flashing quickly in relation 1:1 to the pulse

(3) Position the Run/Stop Switch to *"Run"*
⇨ Error-LED (red) is flashing quickly in relation 1:1 to the pulse

(4) Reposition Run/Stop Switch back to *"MRes"* **within 2 seconds**
   ⇨ PLCcore-E660 is deleting PLC program
   ⇨ Run-LED (green) and Error-LED (red) are both flashing alternately

(5) Reposition Run/Stop Switch to *"Stop"* or *"Run"* and reset again to start the PLCcore-E660 and bring it into normal working state

If Reset of the PLCcore-E660 is activated (e.g. through temporary power interrupt) while at the same time the Run/Stop Switch is positioned to *"MRes"*, the module recognizes a reset requirement. This is visible through quick flashing of the Run-LED (green). This mode can be stopped without risk. Therefore, the Run/Stop Switch must be positioned to *"Run"* or *"Stop"* (Error-LED is flashing) and it must be waited for 2 seconds. The PLCcore-E660 independently stops the reset process after 2 seconds and starts a normal working state with the PLC program which was saved last.

6.9 Using CANopen for CAN interfaces

The PLCcore-E660 manages 2 CAN interfaces
- CAN0 as USB-CAN-Modul of the sysWORXX series (SYS TEC order number 3204000)
- CAN1 on-board on the Core-Modul – connector P900A Bottom or X900 on the Development Board

The PLCcore-E660 features 1 CAN interface (CAN0), usable as CANopen Manager (conform to CiA Draft Standard 302). The configuration of this interface (active/inactive, node number, Bitrate, Master on/off) is described in section 7.4.

The CAN interface allow for data exchange with other devices via network variables and is usable from a PLC program via function blocks of type *"CAN_Xxx"*. More details are included in *"User Manual CANopen Extension for IEC 61131-3"*, Manual no.: L-1008.

The CANopen services **PDO** (***P***rocess ***D***ata ***O***bjects) and **SDO** (***S***ervice ***D***ata ***O***bjects) are two separate mechanisms for data exchange between single field bus devices. Process data sent from a node (**PDO**) are available as broadcast to interested receivers. PDOs are limited to 1 CAN telegram and therewith to 8 Byte user data maximum because PDOs are executed as non-receipt broadcast messages. On the contrary, **SDO** transfers are based on logical point-to-point connections ("Peer to Peer") between two nodes and allow the receipted exchange of data packages that may be larger than 8 Bytes. Those data packages are transferred internally via an appropriate amount of CAN telegrams. Both services are applicable for interface CAN0 as well as for CAN1 of the PLCcore-E660.

SDO communication basically takes place via function blocks of type *"CAN_SDO_Xxx"* (see *"User Manual CANopen Extension for IEC 61131-3"*, Manual no.: L-1008). Function blocks are also available for PDOs (*"CAN_PDO_Xxx"*). Those should only be used for particular cases in order to also activate non-CANopen-conform devices. For the application of PDO function blocks, the CANopen configuration must be known in detail. The reason for this is that the PDO function blocks only use 8 Bytes as input/output parameter, but the assignment of those Bytes to process data is subject to the user.

Instead of PDO function blocks, network variables should mainly be used for PDO-based data exchange. Network variables represent the easiest way of data exchange with other CANopen nodes. Accessing network variables within a PLC program takes place in the same way as accessing internal, local variables of the PLC. Hence, for PLC programmers it is not of importance if e.g. an input variable is allocated to a local input of the control or if it represents the input of a decentralized extension module. The application of network variables is based on the integration of DCF files that are generated by an appropriate CANopen configurator. On the one hand, DCF files describe communication parameters of any device (CAN Identifier, etc.) and on the other hand, they allocate network variables to the Bytes of a CAN telegram (mapping). The application of network variables only requires basic knowledge about CANopen.

In a CANopen network, exchanging PDOs only takes place in status *"OPERATIONAL"*. If the PLCcore-E660 is not in this status, it does not process PDOs (neither for send-site nor for receive-site) and consequently, it does not update the content of network variables. The CANopen Manager is in charge of setting the operational status *"OPERATIONAL"*, *"PRE-OPERATIONAL"* etc. (mostly also called "CANopen Master"). In typical CANopen networks, a programmable node in the form of a PLC is used as CANopen-Manager. The PLCcore-E660 is optionally able to take over tasks of the CANopen Manager. How the Manager is activated is described in section 7.4.

As CANopen Manager, the PLCcore-E660 is able to parameterize the CANopen I/O devices ("CANopen-Slaves") that are connected to the CAN bus. Therefore, upon system start via SDO it transfers DCF files generated by the CANopen configurator to the respective nodes.

6.9.1 CAN interface CAN0

Interface CAN0 features a dynamic object dictionary. This implicates that after activating the PLC, the interface does not provide communication objects for data exchange with other devices. After downloading a PLC program (or its reload from the non-volatile storage after power-on), the required communication objects are dynamically generated according to the DCF file which is integrated in the PLC project. Thus, CAN interface CAN0 is extremely flexible and also applicable for larger amount of data.

For the PLC program, all network variables are declared as *"VAR_EXTERNAL"* according to IEC61131-3. Hence, they are marked as "outside of the control", e.g.:

```
VAR_EXTERNAL
    NetVar1 : BYTE ;
    NetVar2 : UINT ;
END_VAR
```

A detailed procedure about the integration of DCF files into the PLC project and about the declaration of network variables is provided in manual *"User Manual CANopen Extension for IEC 61131-3"* (Manual no.: L-1008).

When using CAN interface CAN0 it must be paid attention that the generation of required objects takes place upon each system start. This is due to the dynamic object directory. "Design instructions" are included in the DCF file that is integrated in the PLC project. **Hence, changes to the configuration can only be made by modifying the DCF file.** This implies that after the network configuration is changed (modification of DCF file), the PLC project must again be translated and loaded onto the PLCcore-E660.

6.9.2 Additional CAN interfaces

In general, the PLC firmware used for PLCcore-E660 is able to simultaneously operate several CAN interfaces (like other PLC types such as the PLCcore-5484 or PLCmodule-C32).

If necessary, more CAN interfaces can be connected to the module externally. Please contact our support employee if you are interested in this option:

support@systec-electronic.com

6.10 Integrated Target Visualization

The PLCcore-9363-HMI **(Order number 3390090 only)** represents a Compact PLC with integrated Target Visualization and thereby optimal for generating user-specific HMI (**H**uman **M**achine **I**nterface)

applications. The integrated Target Visualization of the PLCcore-E660 is based on the *SpiderControl MicroBrowser* by the iniNet Solutions GmbH (http://www.spidercontrol.net). It allows for displaying process values from the PLC as well as forwarding of user actions to the PLC (e.g. entries via Touchscreen, Scrollwheel and Matrix keyboard). The creation of pages shown on the display occurs through the *SpiderControl PLC Editor*, which is installed as additional component together with the programming system *OpenPCS*.

The data exchange between the Target Visualization and the PLC-Program occurs through variables of the PLC-program. It is therewith for example possible to exchange process information in both directions (passing of process variable to display from the PLC to the visualization, passing of a parameter that has been entered into the process visualization to the PLC). Operator events may also be used, e.g. pressing a special button to change values of variables in the PLC-program (e.g. when pressing a button the value of the linked variable changes from 0 to 1). The necessary steps for the creation of visualization pages with the *SpiderControl PLC Editor* as well as linking of graphical elements with variables of the PLC-program is described by the manual "*SYS TEC specific HMI extensions for OpenPCS / IEC 61131-3"* (Manual-No.: L-1231).

Display, mouse and keyboard work directly with the Target Visualization of the PLCcore-E660-HMI, i.e. mouse and keyboard events are processed directly from the *SpiderControl MicroBrowser*. Forwarding of mouse and keyboard events to the PLC-program is not intended, as a purposefull analysis of those data (X- and Y-coordinate, contact pressure) is impossible anyway.

7 Configuration and Administration of the PLCcore-E660

7.1 System requirements and necessary software tools

The administration of the PLCcore-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 PLCcore-E660 using a Windows computer. Procedures using a Linux computer would be analogous.

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

Terminal program

A Terminal program allows the communication with the **command shell** of the PLCcore-E660 via a **serial RS232 connection to UART1 of the PLCcore-E660**. This is required for the Ethernet configuration of the PLCcore-E660 as described in section 7.3. 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://ttssh2.sourceforge.jp).

Telnet client

Telnet-Client allows the communication with **command shell** of the PLCcore-E660 via **Ethernet connection to ETH0 of the PLCcore-E660**. Using Telnet clients requires a completed Ethernet configuration of the PLCcore-E660 according to section 7.3. As alternative solution to Telnet client, all commands can be edited via a Terminal program (to UART1 of the PLCcore-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 PLCcore-E660 (ETH0) and the computer. This allows for example **editing configuration files** by transferring those from the PLCcore-E660 onto the computer where they can be edited and get transferred back to the PLCcore-E660. Downloading files onto the PLCcore-E660 is also necessary to **update the PLC firmware**. (Advice: The update of *PLC firmware* is not identical with the update of the *PLC user program*. The PLC program is directly transferred to the module from the *OpenPCS* programming environment. No additional software is needed for that.)

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.

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.

7.2 Activation/Deactivation of Linux Autostart

In order to allow or prevent the boot procedure of Linux the UEFI bootloader is used (details about system start and bootloader: see sections 6.2 and 6.3).

Communicating with the UEFI bootloader only takes place via the serial interface UART1 of the PLCcore-E660. As receiver on the computer one of the terminal programs must be started (e.g. HyperTerminal or TeraTerm, see section 7.1) and must be configured as follows (see Figure 10):

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

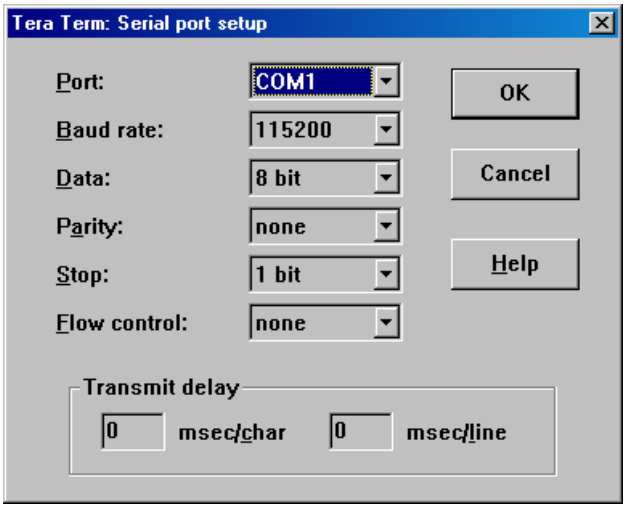


Figure 10: Terminal configuration using the example of "TeraTerm"

7.3 Ethernet configuration of the PLCcore-E660

The main Ethernet configuration of the PLCcore-E660 takes place within the UEFI bootloader and is taken on for all software components (Linux, PLC firmware, HTTP server etc.). The Ethernet configuration is carried out via the serial interface UART1. **Therefore, the EFI shell command prompt must be activated as described in section 6.3.2**. Table 14 lists up EFI shell commands necessary for the Ethernet configuration of the PLCcore-E660.

Table 14: EFI shell configuration commands of the PLCcore-E660

Configuration	Command	Remark
MAC address	setenv ethaddr <xx:xx:xx:xx:xx:xx>	The MAC address worldwide is a clear identification oft he module and is assigned by the producer. **It should not be modified by the user.**
IP address	setenv ipaddr <xxx.xxx.xxx.xxx>	This command sets the local IP address of the PLCcore-E660. The IP address is to be defined by the network administrator.
Network mask	setenv netmask <xxx.xxx.xxx.xxx>	This command sets the network mask of the PLCcore-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 PLCcore-E660. The gateway address is set by the network administrator. **Advice:** If PLCcore-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.

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

After the configuration is finished and according to section 6.3.2, all conditions for a Linux Autostart must be re-established.

Upon Reset (e.g. pushbutton "C RES" on the Development Board) the module starts using the active configurations.

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

7.4 PLC configuration of the PLCcore-E660

7.4.1 PLC configuration via WEB Frontend

After finishing the Ethernet configuration (see section 7.3), all further adjustments can take place via the integrated WEB Frontend of the PLCcore-E660.

To configure the PLCcore-E660 via WEB Frontend it needs a WEB-Browser on the PC (e.g. Microsoft Internet Explorer, Mozilla Firefox etc.). To call the configuration page, prefix *"http://"* must be entered into the address bar of the WEB-Browser prior to entering the IP address of the PLCcore-E660 as set in section 7.2, e.g. *"http://192.168.10.130"*. Figure 11 exemplifies calling the PLCcore-E660 configuration page in the WEB-Browser.

The standard setting (factory setting) requires a user login to configure the PLCcore-E660 via WEB Frontend. This is to prevent unauthorized access. Therefore, user name and password must be entered (see Figure 11). On delivery of the module, the following user account is preconfigured (see section 7.7):

User: PlcAdmin
Password: Plc123

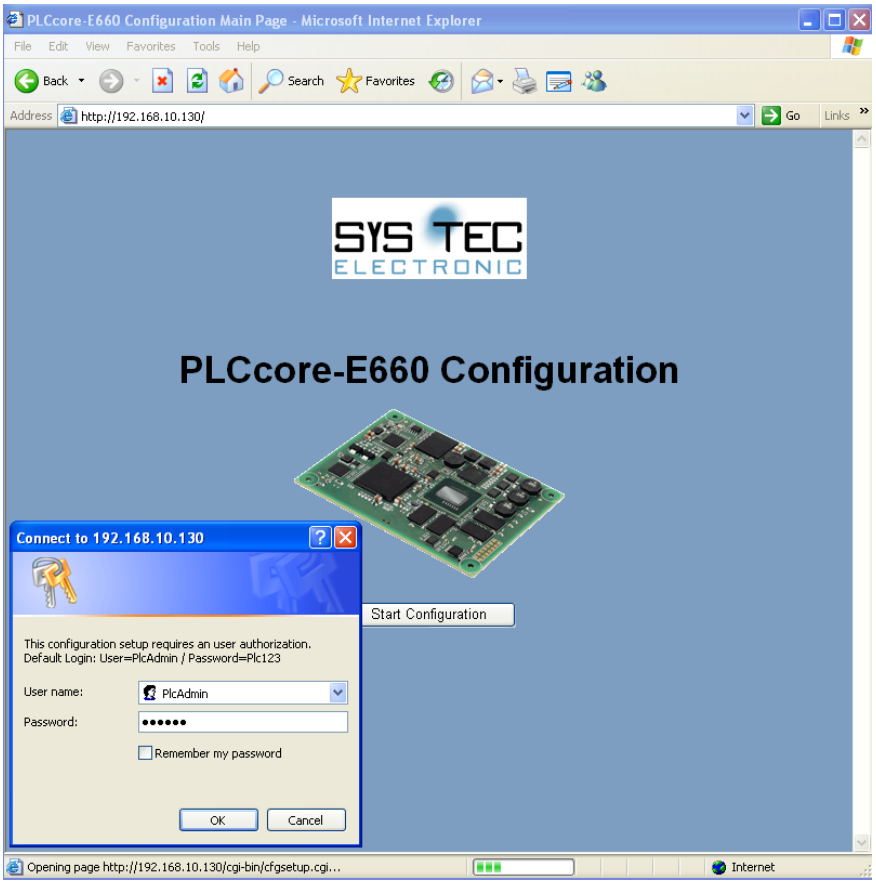


Figure 11: User login dialog of the WEB Frontend

All configuration adjustments for the PLCcore-E660 are based on dialogs. They are adopted into the file ***"/home/plc/bin/plccore-E660.cfg"*** of the PLCcore-E660 by activating the pushbutton "Save Configuration" (also compare section 7.4.2). After activating Reset (e.g. pushbutton "C RES" on the Development Board), the PLCcore-E660 starts automatically using the active configuration. Figure 12 shows the configuration of the PLCcore-E660 via WEB Frontend.

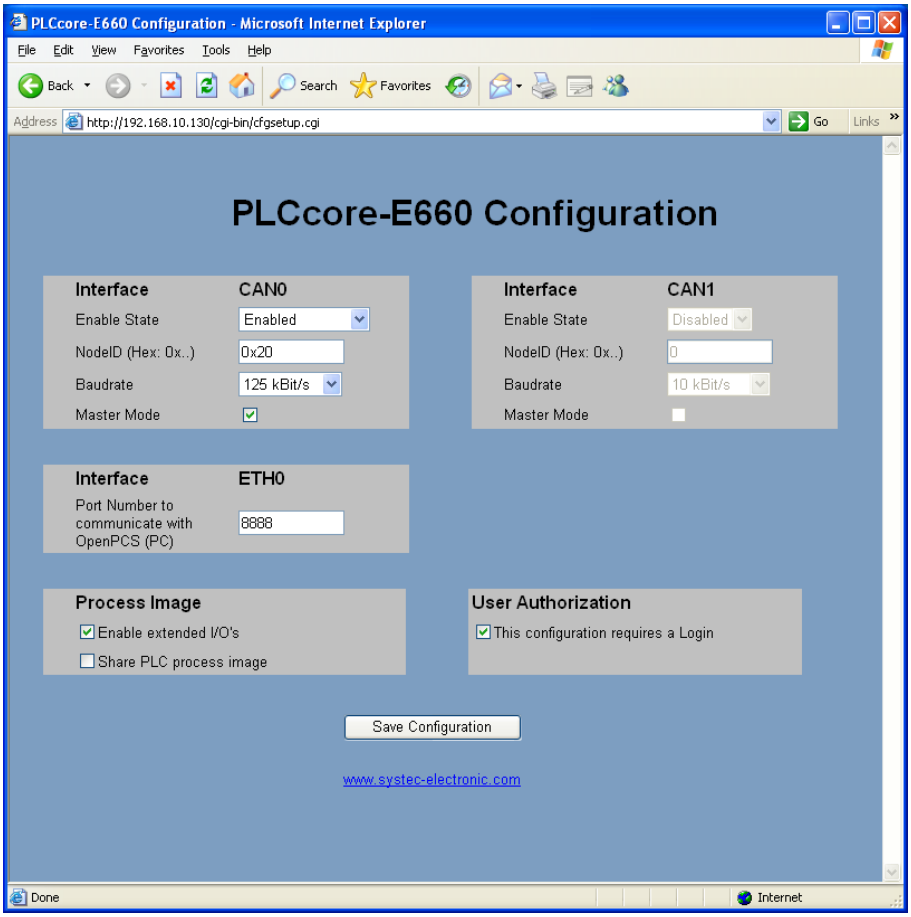


Figure 12: PLC configuration via WEB Frontend

The standard setting (factory setting) of the PLCcore-E660 requires a user login to access the WEB-Frontend. Therefore, only the user name indicated in configuration file ***"/home/plc/bin/plccore-E660.cfg"*** is valid (entry *"User="* in section *"[Login]"*, see section 7.4.2). Procedures to modify the user login password are described in section 7.10. To allow module configuration to another user, an appropriate user account is to be opened as described in section 7.9. Afterwards, the new user name must be entered into the configuration file ***"/home/plc/bin/plccore-E660.cfg"***. Limiting the user login to one user account is cancelled by deleting the entry *"User="* in section *"[Login]"* (see 7.4.2). Thus, any user account may be used to configure the module. By deactivating control box *"This configuration requires a Login"* in the field *"User Authorization"* of the configuration page (see Figure 12) free access to the module configuration is made available without previous user login.

7.4.2 Setup of the configuration file "plccore-E660.cfg"

The configuration file ***"/home/plc/bin/plccore-E660.cfg"*** allows for comprehensive configuration of the PLCcore-E660. Although, working in it manually does not always make sense, because most of the adjustments may easily be edited via WEB Frontend (compare section 7.4.1). The setup of the configuration file is similar to the file format "Windows INI-File". It is divided into "*[Sections]*" which include different entries "*Entry=*". Table 15 shows all configuration entries.

Table 15: Configuration entries of the CFG file

Section	Entry	Value	Meaning
[CAN0]	Enabled	-1, 0, 1	0: Interface CAN0 is deactivated 1: Interface CAN0 is activated, configuration takes place via entries of the configuration file below
	NodeID	1 … 127 or 0x01 … 0x7F	Node number for interface CAN0 (decimal or hexadecimal with prefix "0x")
	Baudrate	10, 20, 50, 125, 250, 500, 800, 1000	Bitrate for interface CAN0
	MasterMode	0, 1	1: Master mode is activated 0: Master mode is deactivated
[CAN1]	Enabled		By default, section "[CAN1]" is not evaluated, but if necessary it enables the extension of PLCcore-E660 by an additional CAN interface (see section 6.9.2).
	NodeID		
	Baudrate		
	MasterMode		
[ETH0]	PortNum	Default Port no: 8888	Port number for the communication with the Programming-PC and for program download (only for PLCcore-E660/Z5, order number 3390055/Z5 or 3390085/Z5)
[ProcImg]	EnableExtIo	0, 1	0: Only on-board I/Os of the PLCcore-E660 are used for the process image (except Temperature Sensor) 1: All I/Os supported by driver are used for the process image (incl. Temperature Sensor and external ADC of Developmentboard) (for adaptation of process image see section 8.2)
	EnableSharing	0, 1	0: No sharing of process image 1: Sharing of process image is enabled (see section 8.1)
[Visu]	Enable	0, 1	1: Visualization is activated 0: Visualization is deactivated
	SyncTime	0, 1…n	0: Synchronization of data between PLC and Visualization after each PLC Cycle >0: Synchronization of data between PLC and Visualization after <SyncTime> ms

Section	Entry	Value	Meaning
[Login]	Authorization	0, 1	0: Configuration via WEB Frontend is possible without user login 1: Configuration via WEB Frontend requires user login
	User	Default Name: PlcAdmin	If entry *"User="* is available, only the user name defined is accepted for the login to configure via WEB Frontend. If the entry is not available, any user registered on the PLCcore-E660 (see section 7.9) may login via WEB Frontend.

The configuration file ***"/home/plc/bin/plccore-E660.cfg"*** includes the following factory settings:

```
[Login]
Authorization=1
User=PlcAdmin

[CAN0]
Enabled=-1
NodeID=0x20
Baudrate=125
MasterMode=1

[CAN1]
Enabled=0
NodeID=0
Baudrate=0
MasterMode=0

[ETH0]
PortNum=8888

[ProcImg]
EnableExtIo=1
EnableSharing=0

[Visu]
Enable=1
SyncTime=50
```

7.5 Boot configuration of the PLCcore-E660

The PLCcore-E660 is configured so that after Reset the PLC firmware starts automatically. Therefore, all necessary commands are provided by the start script ***"/home/etc/autostart"***. Hence, the required environment variables are set and drivers are booted.

If required, the start script ***"/home/etc/autostart"*** may be complemented by further entries. For example, by entering command *"pureftp"*, the FTP server is called automatically when the PLCcore-E660 is booted. The script can be edited directly on the PLCcore-E660 in the FTP client *"WinSCP"* (compare section 7.8.2) using pushbutton *"F4"* or *"F4 Edit"*.

7.6 Selecting the appropriate firmware version

The PLCcore-E660 is delivered with different firmware versions. Those vary in the communication protocol for the data exchange with the programming PC and they differ from each other regarding the availability of FB communication classes (see section 6.4). The selection of the appropriate firmware version takes place in the start script ***"/home/etc/autostart"***. By default, the *"BoardID"* of the module as set in the EFI bootloader is analyzed. Table 16 lists up the assignments of firmware versions and BoardIDs.

Table 16: Assignment of BoardIDs and firmware versions for the PLCcore-E660

BoardID	Firmware Version	Remark
1011003	plccore-E660-z3	**PLCcore-E660/Z3 (RS232, without Target Visualization)** communication with the programming PC via protocol Siemens 3964(R) (Interface UART1)
1011004	plccore-E660-z4	**PLCcore-E660/Z4 (CANopen, without Target Visualization)** communication with the programming PC via CANopen protocol (Interface CAN0)
1011005	plccore-E660-z5	**PLCcore-E660/Z5 (Ethernet, without Target Visualization)** communication with the programming PC via UDP protocol (Interface ETH0)
1011013	plccore-E660-hmi-z3	**PLCcore-E660-HMI/Z3 (RS232, with Target Visualization)** communication with the programming PC via protocol Siemens 3964(R) (Interface UART1)
1011014	plccore-E660-hmi-z4	**PLCcore-E660-HMI/Z4 (CANopen, with Target Visualization)** communication with the programming PC via CANopen protocol (Interface CAN0)
1011015	plccore-E660-hmi-z5	**PLCcore-E660-HMI/Z5 (Ethernet, with Target Visualization)** communication with the programming PC via UDP protocol (Interface ETH0)

The configuration of BoardIDs takes place via the serial interface UART1. **Therefore, the EFI shell command prompt must be activated as described in section 6.3.2.** Setting BoardIDs is carried out via the EFI shell command *"setenv boardid"* by entering the corresponding number listed in Table 16, e.g.:

```
setenv  boardid  1011005
```

The modified setting can be verified by entering "*printenv*" at the EFI shell command prompt. Command

After completing the configuration, all preconditions for a Linux Autostart must be reestablished according to section 6.3.2.

Alternatively, the appropriate firmware version may be selected directly in the start script ***"/home/etc/autostart"***. Therefore, delete part *"Select PLC Type"* and insert the appropriate firmware instead, e.g.:

```
PLC_FIRMWARE=plccore-E660-z5
```

7.7 Predefined user accounts

All user accounts listed in Table 17 are predefined upon delivery of the PLCcore-E660. Those allow for a login to the command shell (serial RS232 connection or Telnet) and at the FTP server of the PLCcore-E660.

Table 17: Predefined user accounts of the PLCcore-E660

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

7.8 Login to the PLCcore-E660

7.8.1 Login to the command shell

In some cases the administration of the PLCcore-E660 requires the entry of Linux commands in the command shell. Therefore, the user must be directly logged in at the module. There are two different possibilities:

- Logging in is possible with the help of a **Terminal program** (e.g. HyperTerminal or TeraTerm, see section 7.1) via the serial interface **UART1** of the PLCcore-E660 – analog to the procedure described for the Ethernet configuration in section 7.2. **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 PLCcore-E660.

For logging in to the PLCcore-E660 via the Windows standard Telnet client, the command "*telnet*" must be called by using the IP address provided in section 7.3, e.g.

```
telnet  192.168.10.248
```

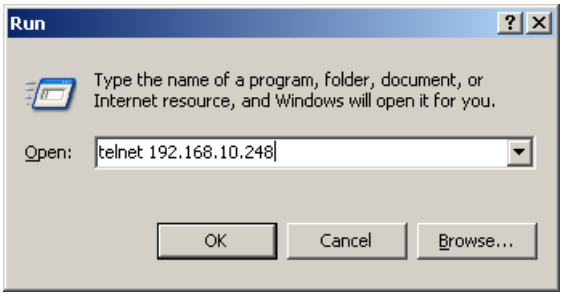


Figure 13: Calling the Telnet client in Windows

Logging in to the PLCcore-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 PLCcore-E660 (also compare section 7.7):

User: *PlcAdmin*
Password: *Plc123*

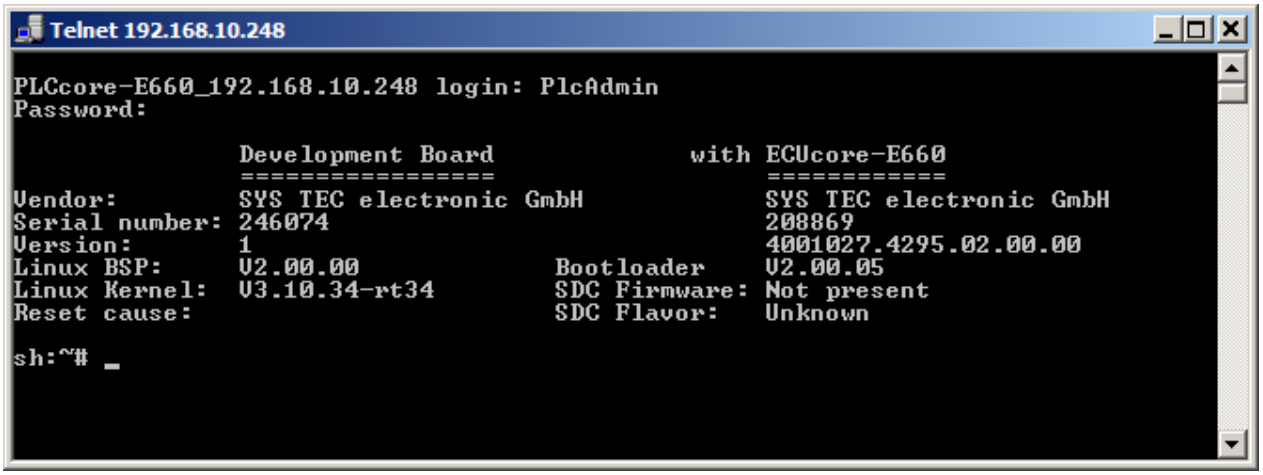


Figure 14: Login to the PLCcore-E660

Figure 14 exemplifies the login to the PLCcore-E660 using a Windows standard Telnet client.

7.8.2 Login to the FTP server

The PLCcore-E660 has available a FTP server (FTP Daemon) that allows file exchange with any computer (up- and download of files). 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 PLCcore-E660 following the procedures described in section 7.8.1. Afterwards, the following command must be entered in the Telnet or Terminal window:

```
pureftp
```

Figure 15 illustrates an example for starting the FTP server.

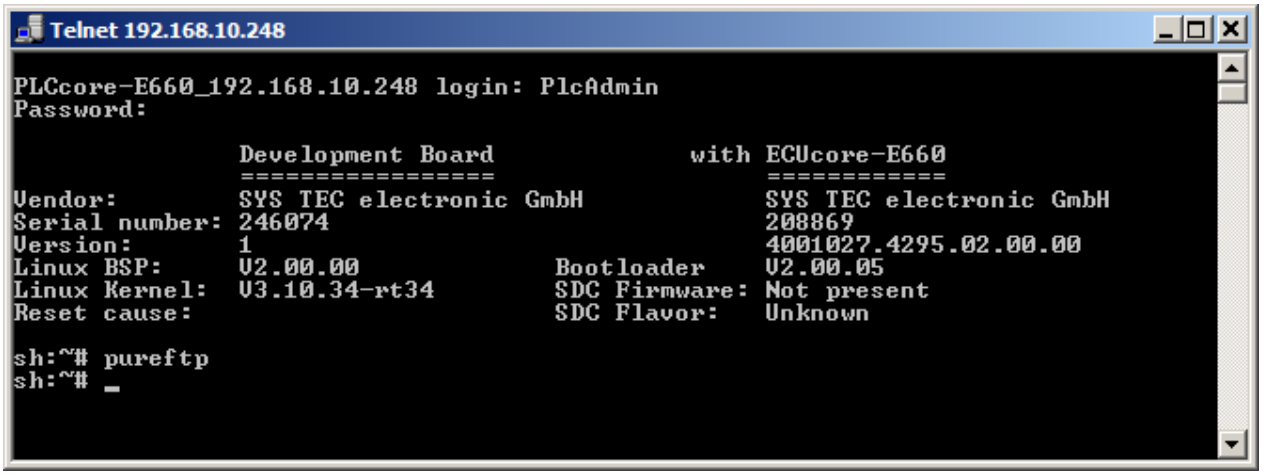


Figure 15: Starting the FTP server

Advice: By entering command *"pureftp"* in the start script ***"/home/etc/autostart"***, the FTP server may be called automatically upon boot of the PLCcore-E660 (see section 7.5).

"WinSCP" - which is available as open source - would be suitable as FTP client for the computer (see section 7.1). It consists of only one EXE file, needs no installation and may be started immediately. After program start, dialog *"WinSCP Login"* appears (see Figure 16) and must be adjusted according to the following configurations:

File protocol: FTP
Host name: IP address for the PLCcore-E660 as set in section 7.3
User name: *PlcAdmin* (for predefined user account, see section 7.7)
Password: *Plc123* (for predefined user account, see section 7.7)

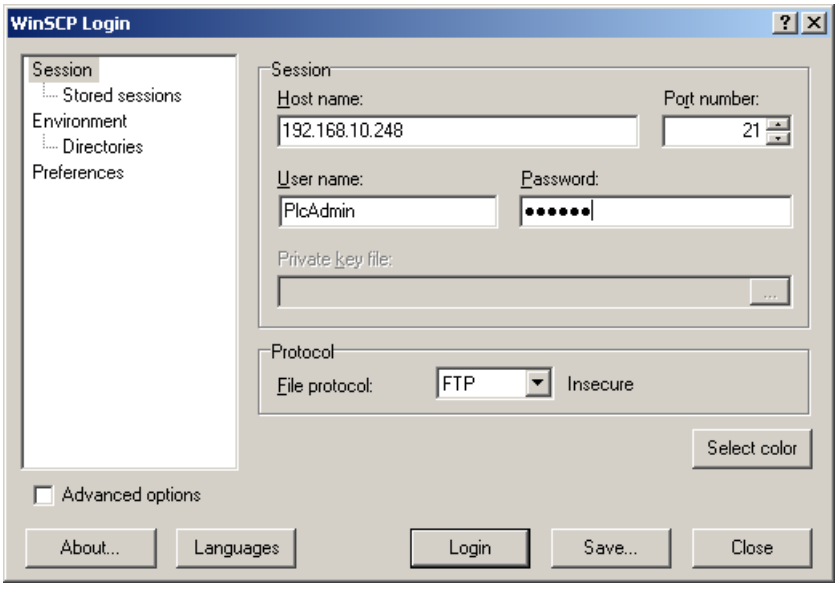


Figure 16: Login settings for WinSCP

After using pushbutton *"Login"*, the FTP client logs in to the PLCcore-E660 and lists up the active content of directory *"/home"* in the right window. Figure 17 shows FTP client *"WinSCP"* after successful login to the PLCcore-E660.

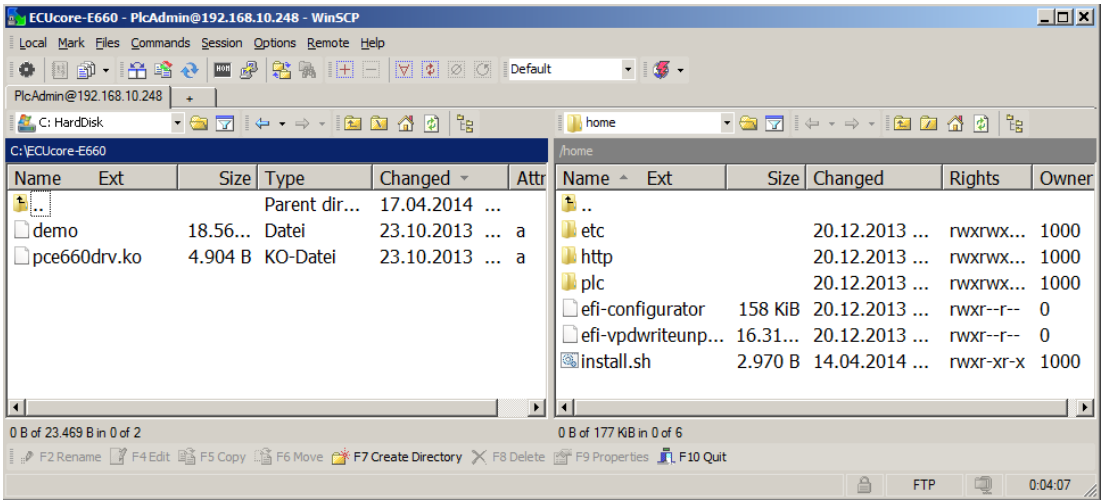


Figure 17: FTP client for Windows "WinSCP"

After successful login, configuration files on the PLCcore-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 PLCcore-E660, e.g. for data backups of the PLCcore-E660 or to transfer installation files for firmware updates (select transfer mode *"Binary"*).

7.9 Adding and deleting user accounts

Adding and deleting user accounts requires the login to the PLCcore-E660 as described in section 7.8.1.

Adding a new user account takes place via Linux command *"adduser"*. In embedded systems such as the PLCcore-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 PLCcore-E660, Linux command *"adduser"* is to be used as follows:

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

Figure 18 exemplifies adding a new account on the PLCcore-E660 for user *"admin2"*.

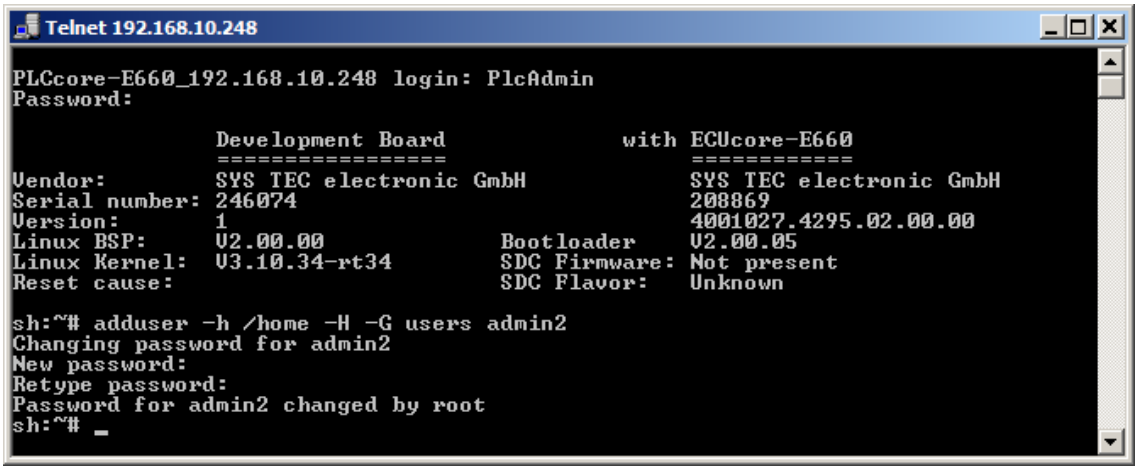


Figure 18: Adding a new user account

Advice:	**If the new user account shall be used to access WEB Frontend, the user name must be entered into the configuration file *"plccore-E660.cfg"* (for details about logging in to WEB Frontend please compare section 7.4.1 and 7.4.2).**

To **delete** an existing user account from the PLCcore-E660, Linux command *"deluser"* plus the respective user name must be used:

```
deluser  <username>
```

7.10 How to change the password for user accounts

Changing the password for user accounts requires login to the PLCcore-E660 as described in section 7.8.1.

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

```
passwd <username>
```

Figure 19 exemplifies the password change for user *"PlcAdmin"*.

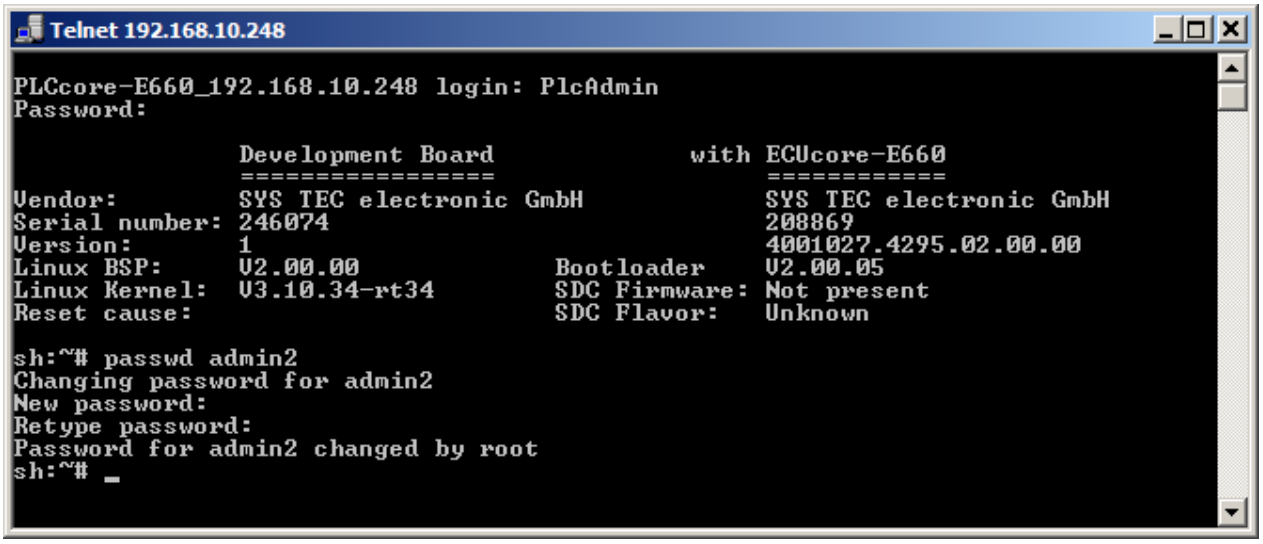


Figure 19: Changing the password for a user account

7.11 Setting the system time

Setting the system time requires login to the PLCcore-E660 as described in section 7.8.1.

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

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

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

Example:

```
date   2011.02.25-11:34:55
          |  |  |  |  |  |
          |  |  |  |  |  +--- Second
          |  |  |  |  +------ Minute
          |  |  |  +--------- Hour
          |  |  +------------ Day
          |  +--------------- Month
          +------------------ Year
```

To set the system time of the PLCcore-E660 to 2011/02/25 and 11:34:55 (as shown in the example above), the following commands are necessary:

```
date  2011.02.25-11:34:55
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 20 exemplifies setting and displaying the system time.

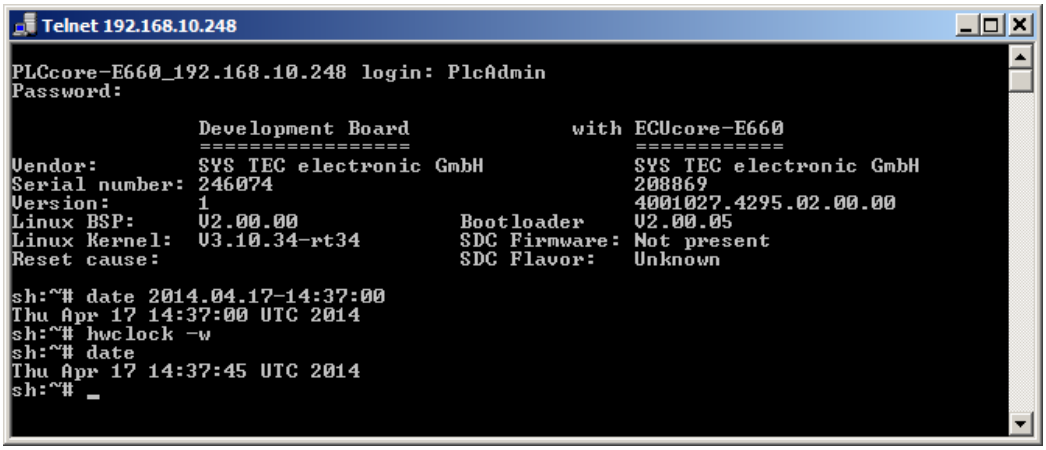


Figure 20: Setting and displaying the system time

Upon start of the PLCcore-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"***.

7.12 File system of the PLCcore-E660

Pre-installed Embedded Linux on the PLCcore-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 PLCcore-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 18 lists up writable paths of the file system during runtime. Path **"/home"** comprises a flash disk that provides part of the on-board flash memory of the PLCcore-E660 as file system. This path is used to store all files modifiable and updatable by the user, e.g. configuration files, PLC firmware and PLC program files that have been loaded onto the module. Directory **"/tmp"** is appropriately sized to function as temporary buffer for FTP downloads of firmware archives for PLC software updates (see section 7.13.1).

Table 18: File system configuration of the PLCcore-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

[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.

Sizes of file system paths that are configured or still available can be identified by using the Linux command "df" ("DiskFree") – see Figure 21.

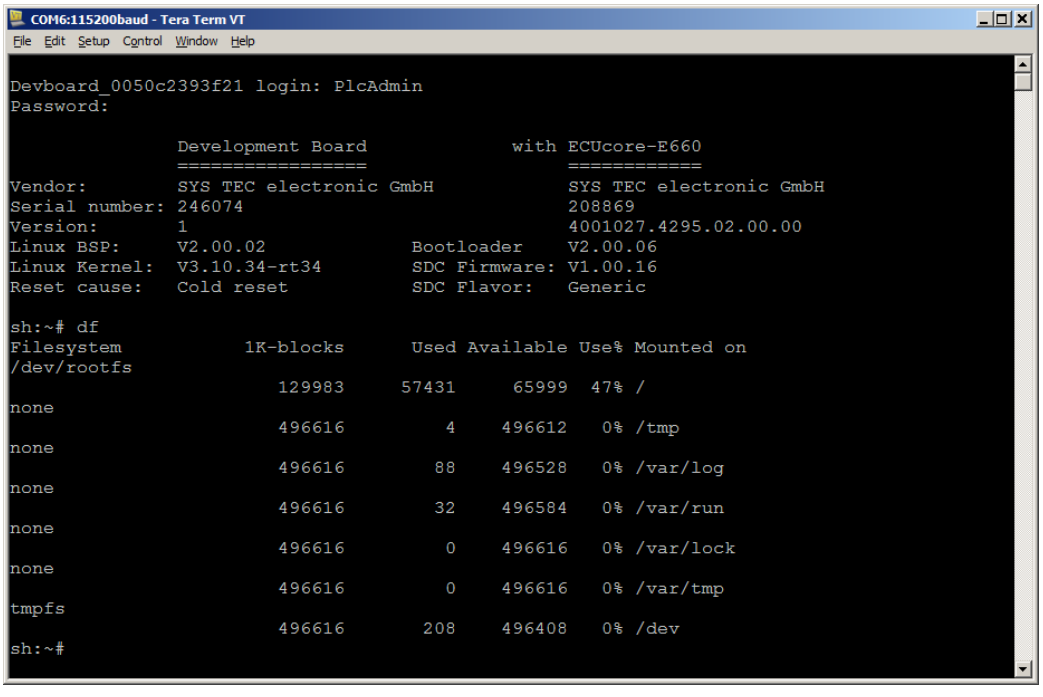


Figure 21: Display of information about the file system

Particular information about the system login and handling the Linux command shell of the PLCcore-E660 is given attention in section 7.8.

7.13 Software update of the PLCcore-E660

All necessary firmware components to run the PLCcore-E660 are already installed on the module upon delivery. Hence, firmware updates should only be required in exceptional cases, e.g. to input new software that includes new functionality.

7.13.1 Updating the PLC firmware

PLC firmware indicates the run time environment of the PLC. ***PLC firmware*** can only be generated and modified by the producer; **it is not identical with the PLC user program** which is created by the PLC user. The PLC user program is directly transferred from the *OpenPCS* programming environment onto the module. No additional software is needed.

Updating the PLC firmware requires login to the command shell of the PLCcore-E660 as described in section 7.8.1 and login to the FTP server as described in section 7.8.2.

Updating the PLC firmware takes place via a self-extracting firmware archive that is transferred onto the PLCcore-E660 via FTP. After starting the FTP server on the PLCcore-E660 (command *"pureftp"*, see section 7.8.2), the respective firmware archive can be transferred into directory *"/tmp"* of the PLCcore-E660 (see Figure 22).

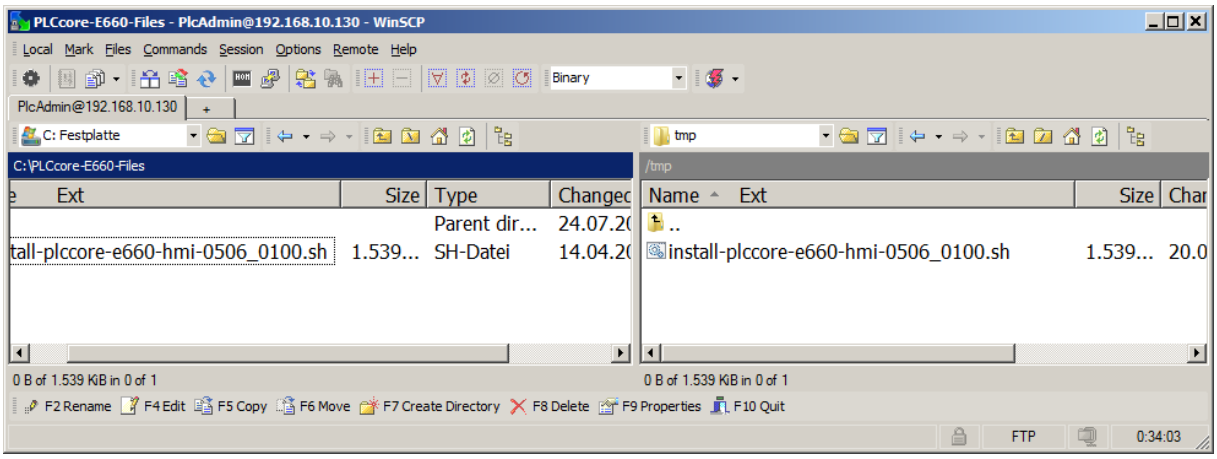


Figure 22: File transfer in FTP client "WinSCP"

Important: To transfer the firmware archive via FTP, transfer type *"Binary"* must be chosen. If FTP client *"WinSCP"* is used, the appropriate transfer mode is to be chosen from the menu bar. After downloading the firmware archive, it must be checked if the file transferred to the PLCcore-E660 has the exact same size as the original file on the computer (compare Figure 22). Any differences in that would indicate a mistaken transfer mode (e.g. *"Text"*). In that case the transfer must be repeated using transfer type *"Binary"*.

After downloading the self-extracting archive, the PLC firmware must be installed on the PLCcore-E660. Therefore, the following commands are to be entered in the Telnet window. It must be considered that the file name for the firmware archive is labeled with a version identifier (e.g. *"install-plccore-E660-0506_0100.sh"* for version 5.06.01.00). This number must be adjusted when commands are entered:

```
cd  /tmp
chmod  +x  install-plccore-E660-0506_0100.sh
./install-plccore-E660-0506_0100.sh
```

Advice: The command shell of the PLCcore-E660 is able to automatically complete names if the Tab key is used ("tab completion"). Hence, it should be sufficient to enter the first letters of each file name and the system will complement it automatically. For example, *"./ins"* is completed to *"./install-plccore-E660-0506_0100.sh"* if the Tab key is used.

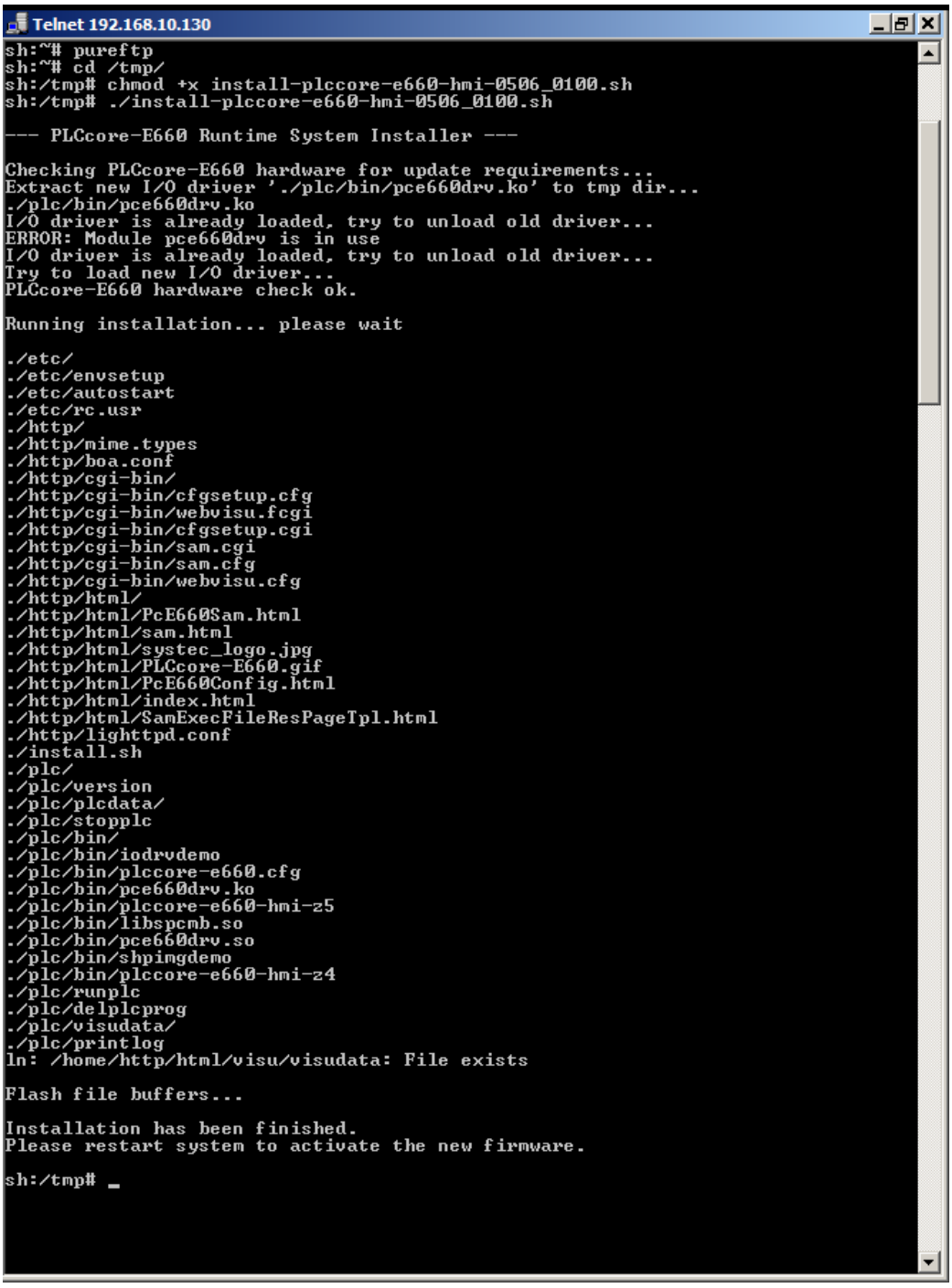


Figure 23: Installing PLC firmware on the PLCcore-E660

Figure 23 exemplifies the installation of PLC firmware on the PLCcore-E660. After Reset the module is started using the updated firmware.

Advice:	**If the PLC firmware is updated, the configuration file *"/home/plc/bin/plccore-E660.cfg"* is overwritten. This results in a reset of the PLC configuration to default settings. Consequently, after an update, the configuration described in section 7.4 should be checked and if necessary it should be reset.**

7.13.2 How to update the Linux Image

Normally, an update of the Linux image is unnecessary. In cases, that an update would be needed, please refer to the respective explanations in sections 7.7.and 6.8 in the document L-1554 System Manual ECUcore-E660.

Advice:	**If a new SD card image is written to the SD card, all data in the directory "home/" and the PCL firmware data is lost. Please repeat then the respective installations (see section 7.13.1).**

8 Adaption of In-/Outputs and Process Image

8.1 Data exchange via shared process image

8.1.1 Overview of the shared process image

The PLCcore-E660 is based on the operating system Embedded Linux. Thus, it is possible to execute other user-specific programs simultaneously to running the PLC firmware. The PLC program and a user-specific C/C++ application can exchange data by using the same process image (shared process image). Implementing user-specific C/C++ applications **is based on the Software package SO-1116 ("VMware-Image of the Linux development system for the ECUcore-E660")**.

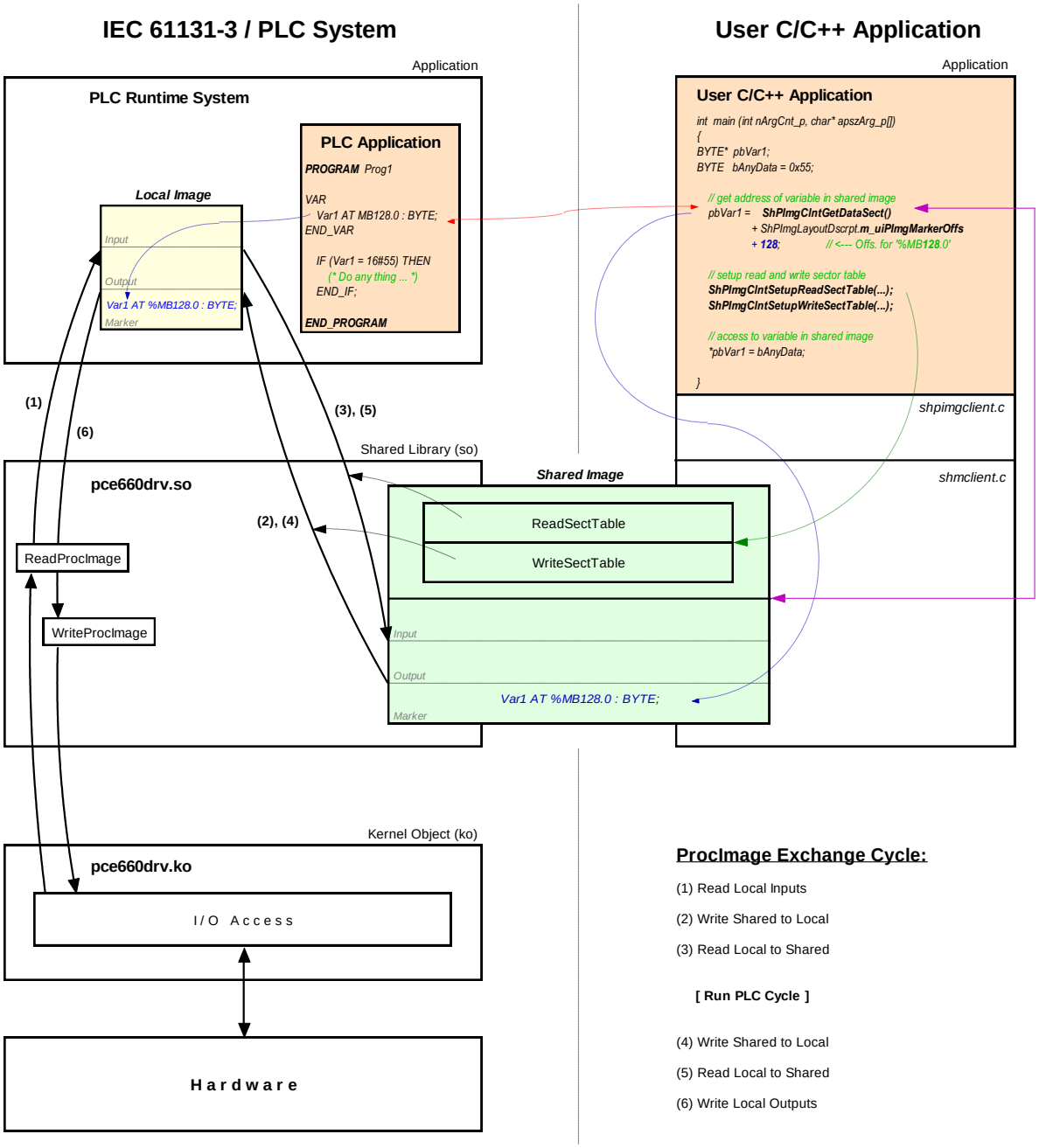


Figure 24: Overview of the shared process image

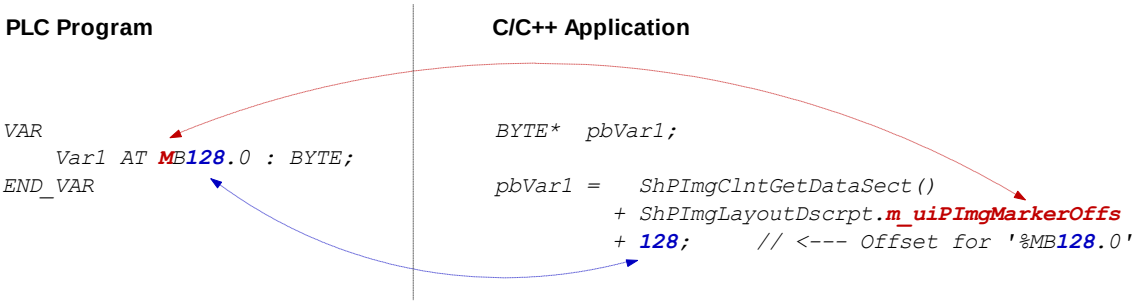
Not all variables are utilizable via the shared process image within a C/C++ application. Only those directly addressed variables that the PLC program generates within the process image. As shown in Figure 24, two separate process images are used for the data exchange with an external application inside of the PLC runtime system. This is necessary to meet the IEC 61131-3 requirement that the initial PLC process image may not be modified during the entire execution of one PLC program cycle. Thereby, the PLC program always operates with the internal process image that is locally generated within the PLC runtime system ("Local Image" in Figure 24). This is integrated within the PLC runtime system and is protected against direct accesses from the outside. On the contrary, the user-specific, external C/C++ application always uses the shared process image ("Shared Image" in Figure 24). This separation of two process images enables isolation between accesses to the PLC program and the external application. Those two in parallel and independently running processes now must only be synchronized for a short period of time to copy the process data.

An activation of **option *"Share PLC process image"*** within the PLC configuration enables data exchange with external applications (see section 7.4.1). Alternatively, entry *"EnableSharing="* can directly be set within section *"[ProcImg]"* of the configuration file ***"/home/plc/bin/plccore-E660.cfg"*** (see section 7.4.2). The appropriate configuration setting is evaluated upon start of the PLC firmware. By activating option *"Share PLC process image"*, the PLC firmware creates a second process image as Shared Memory ("Shared Image" in Figure 24). Its task is to exchange data with external applications. Hereby, the PLC firmware functions as Server and the external, user-specific C/C++ application functions as Client.

ReadSectorTable and ***WriteSectorTable*** both control the copying of data between the two process images. Both tables are filled by the Client (external, user-specific C/C++ application) and are executed by Server (PLC runtime system). The Client defines ranges of the PLC process image from which it will read data (*ReadSectorTable*) or in which it will write data (*WriteSectorTable*). Hence, the terms *"Read"* and *"Write"* refer to data tranfer directions from the viewpoint of the Client.

Sections to read and write may comprise all sections of the entire process image – input, output as well as marker sections. This allows for example that a Client application writes data into the input section of the PLC process image and reads data from the output section. Figure 24 shows the sequence of single read and write operations. Prior to the execution of a PLC program cycle, the physical inputs are imported into the local process image of the PLC (1). Afterwards, all sections defined in *WriteSectorTable* are taken over from the shared process image into the local process image (2). By following this sequence, a Client application for example is able to overwrite the value of a physical input. This may be used for simulation purposes as well as for setting input data to constant values (*"Forcen"*). Similarly, prior to writing the process image onto the physical outputs (6), sections defined in *WriteSectorTable* are taken over from the shared process image into the local process image. (4). Thus, a Client application is able to overwrite output information generated by the PLC program.

The PLC firmware provides the **setup of the process image**. The Client application receives information about the setup of the process image via function ***ShPImgCIntSetup()***. This function enters start offsets and values of the input, output and marker sections into the structure of type *tShPImgLayoutDscrpt*. Function ***ShPImgCIntGetDataSect()*** provides the start address of the shared process image. Upon defining a variable within the PLC program, its absolute position within the process image is determined through sections (%I = Input, %Q = Output, %M = Marker) and offset (e.g. %MB**128**.0). In each section the offset starts at zero, so that for example creating a new variable in the marker section would be independent of values in the input and output section. Creating a corresponding **pair of variables** in the PLC program as well as in the C/C++ application allows for data exchange between the PLC program and the external application. Therefore, both sides must refer to the same address. Structure *tShPImgLayoutDscrpt* reflects the physical setup of the process image in the PLC firmware including input, output and marker sections. This is to use an addressing procedure for defining appropriate variables in the C/C++ application that is comparable to the PLC program. Hence, also in the C/C++ program a variable is defined in the shared process image by indicating the respective section and its offset. The following example illustrates the creation of a corresponding variable pair in the PLC program and C/C++ application:



As described above, ***ReadSectorTable*** and ***WriteSectorTable*** manage the copy process to exchange variable contents between the PLC and the C/C++ program. Following the example illustrated, the Client (C/C++ application) must enter an appropriate value into the *WriteSectorTable* to transfer the value of a variable from the C/C++ application to the PLC program (*WriteSectorTable*, because the Client "writes" the variable to the Server):

```
// specify offset and size of 'Var1' and define sync type (always or on demand?)
WriteSectTab[0].m_uiPImgDataSectOffs = ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 128;
WriteSectTab[0].m_uiPImgDataSectSize = sizeof(BYTE);
WriteSectTab[0].m_SyncType           = kShPImgSyncOnDemand;

// define the WriteSectorTable with the size of 1 entry
ShPImgClntSetupWriteSectTable (WriteSectTab, 1);
```

If several variable pairs are generated within the same transfer direction for the data exchange between the PLC program and the C/C++ application, they should possibly all be defined in one coherent address range. Thus, it is possible to list them as one entry in the appropriate SectorTable. The address of the first variable must be set as the *SectorOffset* and the sum of the variable sizes as *SectorSize*. Combining the variables improves the efficiency and the performance of the copy processes.

For each entry of the *WriteSectorTable* an appropriate *SyncType* must be defined. It determines whether the section is generally taken over from the shared process image into the local image whenever there are two successive PLC cycles (***kShPImgSyncAlways***) or whether it is taken over on demand (***kShPImgSyncOnDemand***). If classified as *SyncOnDemand*, the data only is copied if the respective section before was explicitly marked as updated. This takes places by calling function ***ShPImgClntWriteSectMarkNewData()*** and entering the corresponding *WriteSectorTable*-Index (e.g. 0 for *WriteSectTab[0]* etc.).

kShPImgSyncAlways is provided as *SyncType* for the *ReadSectorTable* (the value of the member element *m_SyncType* is ignored). The PLC firmware is not able to identify which variables were changed by the PLC program of the cycle before. Hence, all sections defined in *ReadSectorTable* are always taken over from the local image into the shared process image. Thus, the respective variables in the shared process image always hold the actual values.

The PLC firmware and the C/C++ application both use the shared process image. To prevent conflicts due to accesses from both of those in parallel running processes at the same time, the shared process image is internally protected by a semaphore. If one process requires access to the shared process image, this process enters a critical section by setting the semaphore first and receiving exclusive access to the shared process image second. If the other process requires access to the shared process image at the same time, it also must enter a critical section by trying to set the semaphore. In this case, the operating system identifies that the shared process image is already being used. It blocks the second process until the first process leaves the critical section and releases the semaphore. Thereby, the operating system assures that only one of the two in parallel running processes (PLC runtime system and C/C++ application) may enter the critical section and receives access to the shared process image. To ensure that both processes do not interfere with each other too much, they should enter the critical section as less as possible and only as long as necessary. Otherwise, the PLC cycle time may be extended and runtime variations (Jitter) may occur.

The client application has available two functions to set the semaphore and to block exclusive access to the shared process image. Function ***ShPImgClntLockSegment()*** is necessary to enter the critical section and function ***ShPImgClntUnlockSegment()*** to leave it. The segment between both functions is called protected section, because in this segment the client application holds access to the shared process image without competition. The consistency of read or written data is only guaranteed within such a protected section. Outside the protected section, the shared process image may anytime be manipulated by the PLC runtime system. The following example shows the exclusive access to the shared process image in the C/C++ application:

```
ShPImgClntLockSegment();
{
    // write new data value into Var1
    *pbVar1 = bAnyData;

    // mark new data for WriteSectorTable entry number 0
    ShPImgClntWriteSectMarkNewData (0);
}
ShPImgClntUnlockSegment();
```

For the example above, *kShPImgSyncOnDemand* was defined as *SyncType* upon generating entry *WriteSectorTable*. Hence, taking over variable *Var1* from the shared process image into the local image can only take place if the respective section was beforehand explicitly marked as updated. Therefore, it is necessary to call function ***ShPImgClntWriteSectMarkNewData()***. Since function *ShPImgClntWriteSectMarkNewData()* does not modify the semaphore, it may only be used within a protected section (see example) – such as the code section between *ShPImgClntLockSegment()* and *ShPImgClntUnlockSegment()*.

The synchronization between local image and shared process image by the PLC runtime system only takes place in-between two successive PLC cycles. A client application (user-specific C/C++ program) is not directly informed about this point of time, but it can get information about the update of the shared process image from the PLC runtime system. Therefore, the client application must define a callback handler of the type *tShPImgAppNewDataSigHandler*, e.g.:

```
static  void  AppSigHandlerNewData (void)
{
    fNewDataSignaled_1 = TRUE;
}
```

This callback handler must be registered with the help of function ***ShPImgClntSetNewDataSigHandler()***. The handler is selected subsequent to a synchronization of the two images.

The **callback handler of the client application is called within the context of a Linux signal handler** (the PLC runtime system informs the client using Linux function *kill()*). Accordingly, all common **restrictions** for the Linux signal handler also apply to the callback handler of the client application. In particular, it is only allowed to call a few operating system functions that are explicitly marked as reentrant-proof. Please pay attention to not make reentrant calls of local functions within the client application. As shown in the example, only a global flag should be set for the signaling within the callback handler. This flag will later on be evaluated and processed in the main loop of the client application.

8.1.2 API of the shared process image client

As illustrated in Figure 24, the user-specific C/C++ application exclusively uses the API (Application Programming Interface) provided by the s*hared process image client*. This API is declared in the

header file *shpimgclient.h* and implemented in the source file *shpimgclient.c*. It contains the following types (partly defined in *shpimg.h*) and functions:

Structure *tShPImgLayoutDscrpt*

```
typedef struct
{
    // definition of process image sections
    unsigned int        m_uiPImgInputOffs;      // start offset of input section
    unsigned int        m_uiPImgInputSize;      // size of input section
    unsigned int        m_uiPImgOutputOffs;     // start offset of output section
    unsigned int        m_uiPImgOutputSize;     // size of output section
    unsigned int        m_uiPImgMarkerOffs;     // start offset of marker section
    unsigned int        m_uiPImgMarkerSize;     // size of marker section

} tShPImgLayoutDscrpt;
```

Structure ***tShPImgLayoutDscrpt*** describes the setup of the process image given by the PLC firmware. The client application receives the information about the setup of the process image via function *ShPImgClntSetup()*. This function enters start offsets and values of input, output and marker sections into the structure provided upon function calling.

Structure *tShPImgSectDscrpt*

```
typedef struct
{
    // definition of data exchange section
    unsigned int        m_uiPImgDataSectOffs;
    unsigned int        m_uiPImgDataSectSize;
    tShPImgSyncType     m_SyncType;              // only used for WriteSectTab
    BOOL                m_fNewData;

} tShPImgSectDscrpt;
```

Structure ***tShPImgSectDscrpt*** describes the setup of a *ReadSectorTable* or *WriteSectorTable* entry that must be defined by the client. Both tables support the synchronization between the local image of the PLC runtime system and the shared process image (see section 8.1.1). Member element *m_uiPImgDataSectOffs* defines the absolute start offset of the section within the shared process images. The respective start offsets of the input, output and marker sections can be determined through structure *tShPImgLayoutDscrpt*. Member element *m_uiPImgDataSectSize* determines the size of the section which may include one or more variables. Member element *m_SyncType* only applies to entries of the WriteSectorTable. It determines whether the section is generally taken over from the shared process image into the local image whenever there are two successive PLC cycles (***kShPImgSyncAlways***) or whether it is taken over on demand (***kShPImgSyncOnDemand***). If classified as *SyncOnDemand*, the data must be marked as modified by calling function *ShPImgClntWriteSectMarkNewData()*. It sets the member element *m_fNewData* to TRUE. The client application should never directly modify this member element.

Function *ShPImgClntSetup*

```
BOOL  ShPImgClntSetup  (tShPImgLayoutDscrpt* pShPImgLayoutDscrpt_p);
```

Function ***ShPImgClntSetup()*** initializes the *shared process image client* and connects itself with the storage segment for the shared process image which is generated by the PLC runtime system. Afterwards, it enters the start offsets and values of the input, output and marker sections into the structure of type *tShPImgLayoutDscrpt* provided upon function call. Hence, the

client application receives notice about the process image setup managed by the PLC firmware.

If the PLC runtime system is not active when the function is called or if it has not generated a shared process image (option *"Share PLC process image"* in the PLC configuration deactivated, see section 8.1.1), the function will return with the return value FALSE. If the initialization was successful, the return value will be TRUE.

Function *ShPImgClntRelease*

```
BOOL  ShPImgClntRelease  (void);
```

Function ***ShPImgClntRelease()*** shuts down the s*hared process image client* and disconnects the connection to the storage segment generated for the shared process image by the PLC runtime system.

If executed successfully, the function delivers return value TRUE. If an error occurs, it will deliver return value FALSE.

Function *ShPImgClntSetNewDataSigHandler*

```
BOOL  ShPImgClntSetNewDataSigHandler  (
         tShPImgAppNewDataSigHandler pfnShPImgAppNewDataSigHandler_p);
```

Function ***ShPImgClntSetNewDataSigHandler()*** registers a user-specific callback handler. This callback handler is called after a synchronization of both images. Registered callback handlers are cleared by the parameter NULL.

The **callback handler is called within the context of a Linux signal handler**. Accordingly, all common **restrictions** for the Linux signal handler also apply to the callback handler (see section 8.1.1).

If executed successfully, the function delivers return value TRUE. If an error occurs, it will deliver return value FALSE.

Function *ShPImgClntGetHeader*

```
tShPImgHeader*  ShPImgClntGetHeader  (void);
```

Function ***ShPImgClntGetHeader()*** provides a pointer to the internally used structure type *tShPImgHeader* to manage the shared process image. The client application does usually not need this structure, because all data that it includes can be read and written through functions of the API provided by the *shared process image client*.

Function *ShPImgClntGetDataSect*

```
BYTE*  ShPImgClntGetDataSect  (void);
```

Function ***ShPImgClntGetDataSect()*** provides a pointer to the beginning of the shared process image. This pointer represents the basic address for all accesses to the shared process image; including the definition of sections *ReadSectorTable* and *WriteSectorTable* (see section 8.1.1).

Functions *ShPImgCIntLockSegment* and *ShPImgCIntUnlockSegment*

```
BOOL  ShPImgClntLockSegment  (void);
BOOL  ShPImgClntUnlockSegment  (void);
```

To exclusively access the shared process image, the client application has available two functions - function ***ShPImgCIntLockSegment()*** to enter the critical section and function ***ShPImgCIntUnlockSegment()*** to leave it. The segment between both functions is called protected section, because in this segment the client application holds unrivaled access to the shared process image (see section 8.1.1). The consistency of read or written data is only guaranteed within such a protected section. Outside the protected section, the shared process image may anytime be manipulated by the PLC runtime system. To ensure that the client application does not interfere with the PLC runtime system too much, the critical sections should be set as less as possible and only as long as necessary. Otherwise, the PLC cycle time may be extended and runtime variations (Jitter) may occur.

If executed successfully, the function delivers return value TRUE. If an error occurs, it will deliver return value FALSE.

Function *ShPImgCIntSetupReadSectTable*

```
BOOL  ShPImgClntSetupReadSectTable  (
         tShPImgSectDscrpt* paShPImgReadSectTab_p,
         unsigned int uiNumOfReadDscrptUsed_p);
```

Function ***ShPImgCIntSetupReadSectTable()*** initializes the *ReadSectorTable* with the values defined by the client. The client hereby determines those sections of the PLC process image from which it wants to read data (see section 8.1.1). Parameter *paShPImgReadSectTab_p* holds elements of the structure *tShPImgSectDscrpt* and must be transferred as start address of a section. Parameter *uiNumOfReadDscrptUsed_p* indicates how many elements the section has.

kShPImgSyncAlways is provided as *SyncType* for the *ReadSectorTable*.

The maximum amount of possible elements for the *ReadSectorTable* is defined by the constant *SHPIMG_READ_SECT_TAB_ENTRIES* and can only be modified if the shared library *"pce660drv.so"* is generated again and at the time (this requires SO-1117 - "Driver Development Kit for the ECUcore-E660", see section 8.2).

If executed successfully, the function delivers return value TRUE. If an error occurs, it will deliver return value FALSE.

Function *ShPImgCIntSetupWriteSectTable*

```
BOOL  ShPImgClntSetupWriteSectTable  (
         tShPImgSectDscrpt* paShPImgWriteSectTab_p,
         unsigned int uiNumOfWriteDscrptUsed_p);
```

Function ***ShPImgCIntSetupWriteSectTable***() initializes the *WriteSectorTable* with the values defined by the client. The client hereby determines those sections of the PLC process image from which it wants to write data (see section 8.1.1). Parameter *paShPImgWriteSectTab_p* holds elements of structure *tShPImgSectDscrpt* and must be transferred as start address of a section. Parameter *uiNumOfWriteDscrptUsed_p* indicates how many elements the section has.

For each entry in the *WriteSectorTable* the *SyncType* must be defined. This *SyncType* defines whether the section is always taken over into the local image between two PLC cycles (***kShPImgSyncAlways***) or only on demand (***kShPImgSyncOnDemand***). If taken over on demand, the respective section is explicitly marked as updated by calling

ShPImgClntWriteSectMarkNewData().

The maximum amount of possible elements for the *WriteSectorTable* is defined by the constant *SHPIMG_WRITE_SECT_TAB_ENTRIES* and can only be modified if the shared library *"pcE660drv.so"* is generated again and at the same time (this requires SO-1103 - "Driver Development Kit for the ECUcore-E660", see section 8.2).

If executed successfully, the function delivers return value TRUE. If an error occurs, it will deliver return value FALSE.

Function *ShPImgClntWriteSectMarkNewData*

```
BOOL  ShPImgClntWriteSectMarkNewData  (unsigned int uiWriteDscrptIdx_p);
```

For the content of a section that is held by the *WriteSectorTable*, function ***ShPImgClntWriteSectMarkNewData()*** marks this content as modified. This function is used (for sections with *SyncType* ***kShPImgSyncOnDemand***) to initiate the copy process of data from the shared process image into the local image of the PLC.

Function *ShPImgClntWriteSectMarkNewData()* directly accesses the header of the shared process image without setting a semaphore before. Hence, it may only be used within the protected section – in the code section between *ShPImgClntLockSegment()* and *ShPImgClntUnlockSegment()*.

If executed successfully, the function delivers return value TRUE. If an error occurs, it will deliver return value FALSE.

8.1.3 Creating a user-specific client application

Software package SO-1116 ("VMware image of the Linux Development System") is the precondition for the implementation of user-specific C/C++ applications. It contains a complete Linux development system in the form of a VMware image. Hence, it allows for an easy introduction into the C/C++ software development for the PLCcore-E660. Thus, the VMware image is the ideal basis to develop Linux-based user programs on the same host PC that already has the *OpenPCS* IEC 61131 programming system installed on it. The VMware image of the Linux development system includes the GNU-Crosscompiler Toolchain for x86 processors. Additionally, it includes essential server services that are preconfigured and usable for effective software development. Details about the VMware image of the Linux development system and instructions for its usage are described in the *"System Manual ECUcore-E660"* (Manual no: L-1554).

As illustrated in Figure 24, the user-specific C/C++ application uses the API (files *shpimgclient.c* and *shpimgclient.h*) which is provided by the *shared process image client*. The *shared process image client* is based on services provided by the *shared memory client* (files *shmclient.c* and *shmclient.c*). Both client implementations are necessary to generate a user-specific C/C++ application. The archive of the *shared process image demos* (***shpimgdemo.tar.gz***) contains the respective files. To create own user-specific client applications, it is recommended to use this demo project as the basis for own adaptations and extensions. Moreover, this demo project contains a Makefile with all relevant configuration adjustments that are necessary to create a Linux application for the PLCcore-E660. Table 19 lists all files of the archive *"shpimgdemo.tar.gz"* and classifies those as general part of the C/C++ application or as specific component for the demo project *"shpimgdemo"*.

Table 19: Content of the archive files "shpimgdemo.tar.gz"

File	Necessary for all C/C++ applications	In particular for demo ***"shpimgdemo"***
shpimgclient.c	x	
shpimgclient.h	x	
shmclient.c	x	
shmclient.h	x	
shpimg.h	x	
global.h	x	
Makefile	draft, to be adjusted	
shpimgdemo.c		x
trmio.c		x
trmio.h		x
trace.c		x

The archive file ***"shpimgdemo.tar.gz"*** including the *shared process image demo* must be unzipped into any subdirectory following the path *"/projects/ECUcore-E660/user"* within the Linux development system. Therefore, command *"tar"* must be called:

```
tar  xzvf  shpimgdemo.tar.gz
```

During the unzipping process, command *"tar"* independently generates the subdirectory *"shpimgdemo"*. For example, if the command is called in directory *"/projects/ECUcore-E660/user"*, all archive files will be unzipped into the path *"/projects/ECUcore-E660/user/**shpimgdemo**"*. Figure 25 exemplifies the unzipping process of *"shpimgdemo.tar.gz"* within the Linux development system.

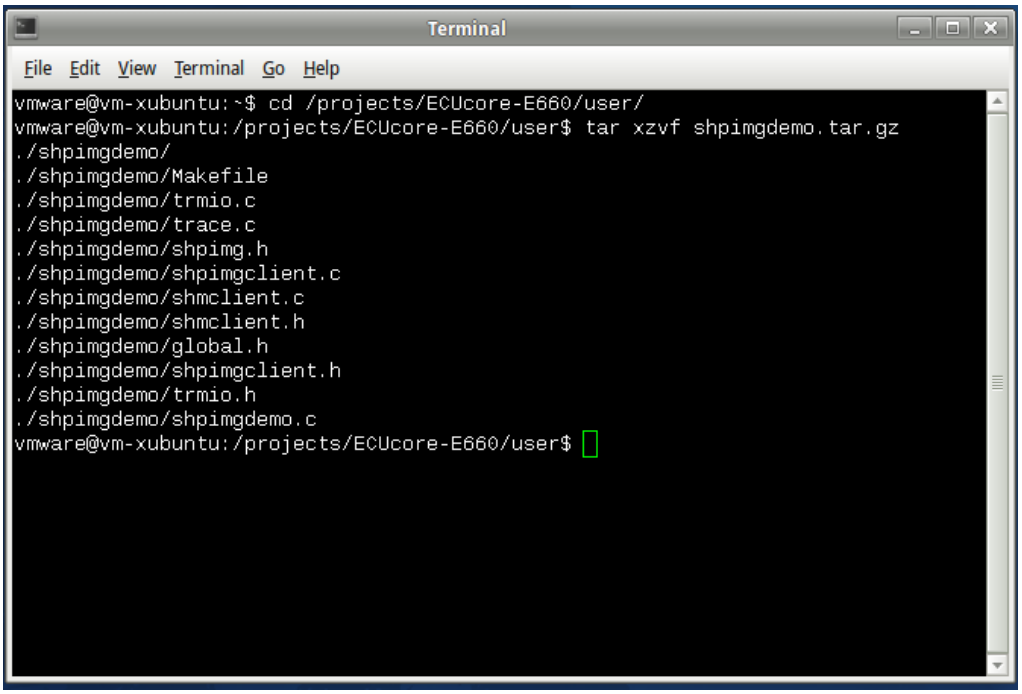


Figure 25: Unzipping the archive files shpimgdemo.tar.gz in the Linux development system

After unzipping and switching into subdirectory *"shpimgdemo"*, the demo project can be created by calling command *"make"*:

```
cd  shpimgdemo
make
```

Figure 26 shows how the demo project *"shpimgdemo"* is generated in the Linux development system.

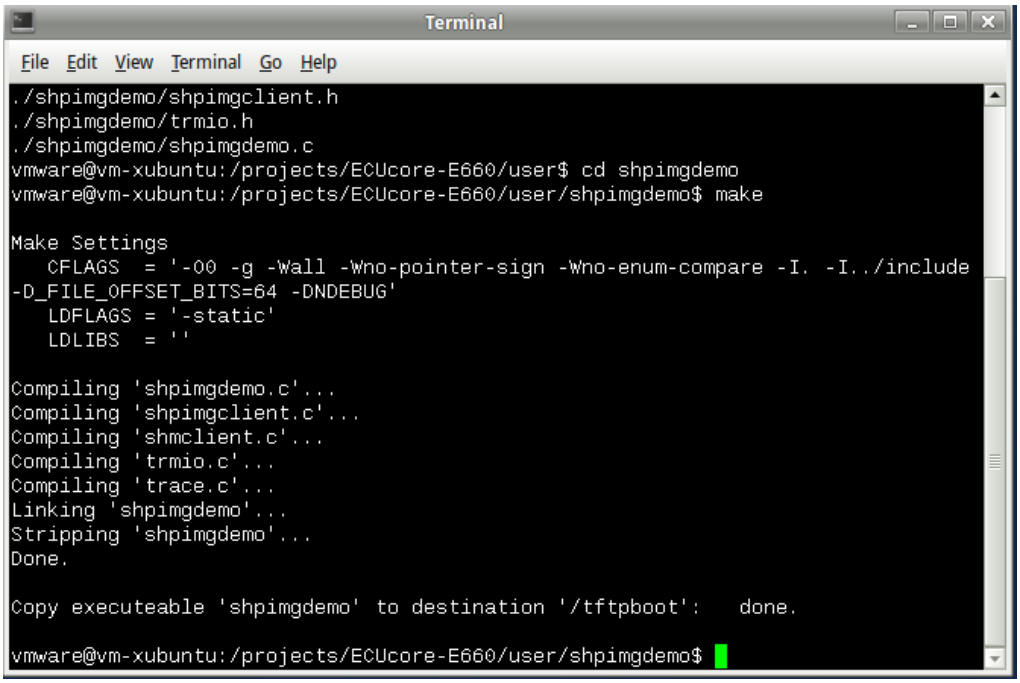


Figure 26: Generating the demo project "shpimgdemo" in the Linux development system

Section 8.1.4 describes the usage and handling of the demo project *"shpimgdemo"* on the PLCcore-E660.

8.1.4 Example for using the shared process image

The demo project *"shpimgdemo"* (described in section 8.1.3) in connection with the PLC program example *"RunLight"* both exemplify the data exchange between a PLC program and a user-specific C/C++ application.

Technical background

The PLC program generates some variables in the process image as directly addressable variables. In a C/C++ application, all those variables are usable via the shared process image. For the PLC program example *"RunLight"* those are the following variables:

```
(* variables for local control via on-board I/Os *)
bButtonGroup        AT %IB0.0   : BYTE;
iAnalogValue        AT %IW8.0   : INT;
bLEDGroup0          AT %QB0.0   : BYTE;
bLEDGroup1          AT %QB1.0   : BYTE;

(* variables for remote control via shared process image  *)
uiRemoteSlidbarLen  AT %MW512.0 : UINT;     (* out: length of slidebar         *)
bRemoteStatus       AT %MB514.0 : BYTE;     (* out: Bit0: RemoteControl=on/off *)
bRemoteDirCtrl      AT %MB515.0 : BYTE;     (* in:  direction left/right       *)
iRemoteSpeedCtrl    AT %MW516.0 : INT;      (* in:  speed                      *)
```

Variables of the PLC program are accessible from a C/C++ application via the shared process image. Therefore, sections must be generated for the *ReadSectorTable* and *WriteSectorTable* on the one hand and on the other hand, pointers must be defined for accessing the variables. The following program extract shows this using the example *"shpimgdemo.c"*. Function *ShPImgClntSetup()* inserts the start offsets of input, output and marker sections into the structure *ShPImgLayoutDscrpt*. Hence, on the basis of the initial address provided by *ShPImgClntGetDataSect()*, the absolute initial addresses of each section in the shared process image can be determined. To identify the address of a variable, the variable's offset within the particular section must be added. For example, the absolute address to access the variable *"bRemoteDirCtrl AT %MB515.0 : BYTE;"* results from the sum of the initial address of the shared process image (pabShPImgDataSect), the start offset of the marker section (*ShPImgLayoutDscrpt.m_uiPImgMarkerOffs für "%M…"*) as well as the direct address within the marker section which was defined in the PLC program (515 for "%MB515.0"):

```
    pbPImgVar_61131_bDirCtrl = (BYTE*) (pabShPImgDataSect
               + ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 515);
```

The following code extract shows the complete definition of all variables in the demo project used for exchanging data with the PLC program:

```
// ---- Setup shared process image client ----
fRes = ShPImgClntSetup (&ShPImgLayoutDscrpt);
if ( !fRes )
{
    printf ("\n*** ERROR *** Init of shared process image client failed");
}

pabShPImgDataSect = ShPImgClntGetDataSect();


// ---- Read Sector Table ----
// Input Section:       bButtonGroup AT %IB0.0
{
    ShPImgReadSectTab[0].m_uiPImgDataSectOffs =
                ShPImgLayoutDscrpt.m_uiPImgInputOffs + 0;
    ShPImgReadSectTab[0].m_uiPImgDataSectSize = sizeof(BYTE);
    ShPImgReadSectTab[0].m_SyncType           = kShPImgSyncAlways;

    pbPImgVar_61131_bButtonGroup = (BYTE*) (pabShPImgDataSect
               + ShPImgLayoutDscrpt.m_uiPImgInputOffs + 0);
}
```

```
// Output Section:      bLEDGroup0 AT %QB0.0
//                      bLEDGroup1 AT %QB1.0
{
    ShPImgReadSectTab[1].m_uiPImgDataSectOffs =
                ShPImgLayoutDscrpt.m_uiPImgOutputOffs + 0;
    ShPImgReadSectTab[1].m_uiPImgDataSectSize = sizeof(BYTE) + sizeof(BYTE);
    ShPImgReadSectTab[1].m_SyncType           = kShPImgSyncAlways;

    pbPImgVar_61131_bLEDGroup0 = (BYTE*) (pabShPImgDataSect
                + ShPImgLayoutDscrpt.m_uiPImgOutputOffs + 0);
    pbPImgVar_61131_bLEDGroup1 = (BYTE*) (pabShPImgDataSect
                + ShPImgLayoutDscrpt.m_uiPImgOutputOffs + 1);
}

// Marker Section:      uiSlidbarLen AT %MW512.0
//                      bStatus      AT %MB514.0
{
    ShPImgReadSectTab[2].m_uiPImgDataSectOffs =
                ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 512;
    ShPImgReadSectTab[2].m_uiPImgDataSectSize =   sizeof(unsigned short int)
                                                + sizeof(BYTE);
    ShPImgReadSectTab[2].m_SyncType           = kShPImgSyncAlways;

    pbPImgVar_61131_usiSlidbarLen = (unsigned short int*) (pabShPImgDataSect
                + ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 512);
    pbPImgVar_61131_bStatus = (BYTE*) (pabShPImgDataSect
                + ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 514);
}

fRes = ShPImgClntSetupReadSectTable (ShPImgReadSectTab, 3);
if ( !fRes )
{
    printf ("\n*** ERROR *** Initialization of read sector table failed");
}

// ---- Write Sector Table ----
// Marker Section:      bDirCtrl   AT %MB515.0
//                      iSpeedCtrl AT %MB516.0
{
    ShPImgWriteSectTab[0].m_uiPImgDataSectOffs =
                ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 515;
    ShPImgWriteSectTab[0].m_uiPImgDataSectSize = sizeof(BYTE) + sizeof(WORD);
    ShPImgWriteSectTab[0].m_SyncType           = kShPImgSyncOnDemand;

    pbPImgVar_61131_bDirCtrl = (BYTE*) (pabShPImgDataSect
                + ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 515);
    psiPImgVar_61131_iSpeedCtrl = (short int*) (pabShPImgDataSect
                + ShPImgLayoutDscrpt.m_uiPImgMarkerOffs + 516);
}

fRes = ShPImgClntSetupWriteSectTable (ShPImgWriteSectTab, 1);
if ( !fRes )
{
    printf ("\n*** ERROR *** Initialization of write sector table failed");
}
```

Realization on the PLCcore-E660

To enable the execution of the *shared process image demo* without previous introduction into the Linux-based C/C++ programming for the PLCcore-E660, the module comes with a preinstalled, translated and ready-to-run program version and PLC firmware (*"/home/plc/bin/shpimgdemo"*). The following description refers to this program version. Alternatively, the demo project can be newly-generated from the corresponding source files (see section 8.1.3) and can be started afterwards.

The following steps are necessary to run the *shared process image demo* on the PLCcore-E660:

1. **Activate option *"Share PLC process image"*** in the PLC configuration (see sections 8.1.1, 7.4.1 and 7.4.2).

2. Open the PLC program example *"RunLight"* in the *OpenPCS* IEC 61131 programming system und build the project for a target hardware of the type *"SYSTEC - PLCcore-E660"*.

3. Select the network connection to the PLCcore-E660 und download the program.

4. Start the PLC program on the PLCcore-E660.

5. Login to the command shell of the PLCcore-E660 as described in section 7.8.1.

6. Switch to the directory *"/home/plc/bin"* and call the demo program *"shpimgdemo"*:

```
cd  /home/plc/bin
./shpimgdemo
```

   The digital outputs of the PLCcore-E660 are selected as runlight. With the help of pushbuttons S704 (DI0) and S705 (DI1), the running direction can be changed. After starting the demo program *"shpimgdemo"* on the PLCcore-E660, actual status information about the runlight is indicated cyclically in the terminal (see Figure 27).

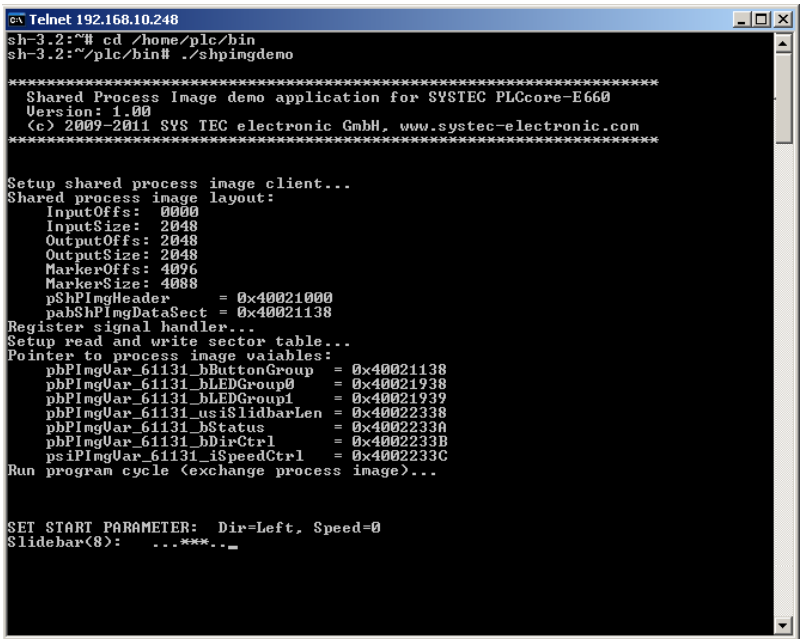


Figure 27: Terminal outputs of the demo program "shpimgdemo" after start

7. By pressing pushbutton S707 (DI3), the control of the runlight direction and speed is handed over to the demo program *"shpimgdemo"*. Afterwards, the running direction may be set by the C application by using the cursor pushbuttons left and right (← und →) in the terminal window and the speed may be changed by using cursor pushbuttons up and down (↑ und ↓).

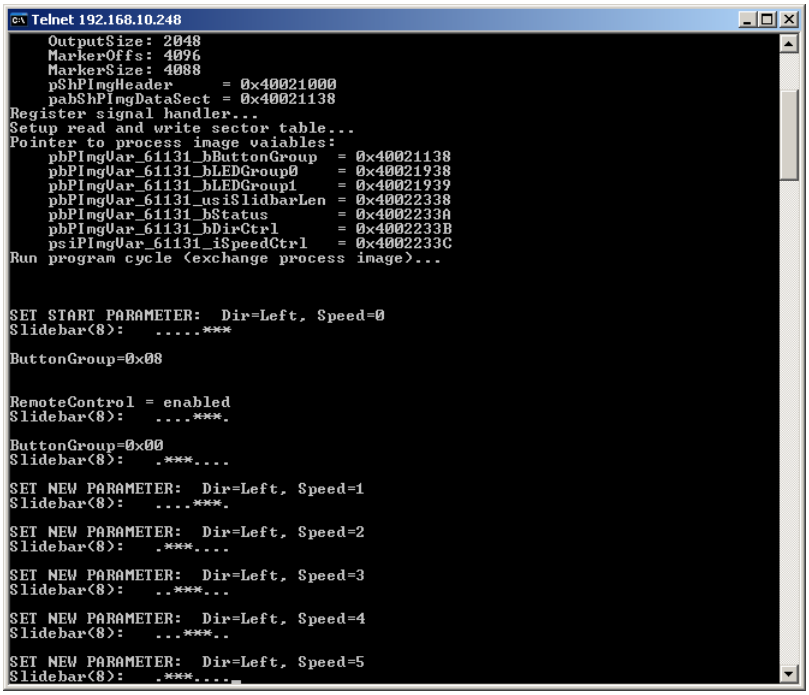


Figure 28: Terminal outputs of the demo program "shpimgdemo" after user inputs

Figure 28 shows the terminal outputs of the demo program *"shpimgdemo"* in answer to activating the cursor pushbuttons.

The demo program *"shpimgdemo"* may be terminated by pressing "Ctrl+C" in the terminal window.

8.2 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 I^2C-Bus
- All other resources provided by the operating system, e.g. file system and TCP/IP

Figure 29 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*).

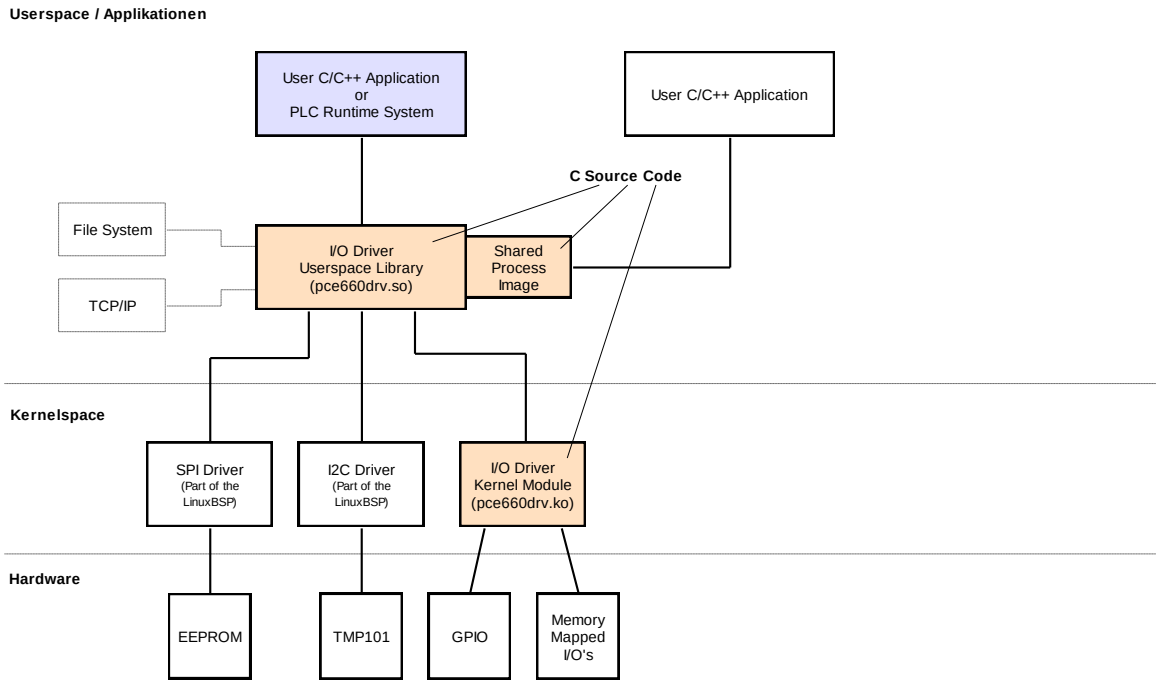


Figure 29: Overview of the Driver Development Kit for the ECUcore-E660

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 29); 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 29); 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 x86 processors.

8.3 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.).

In order to give the testing prgramm unlimited access to all I/O recources, at first the PLC runtime system must be terminated by the command sequence as follows:

```
cd  /home/plc
./stopplc
```

Then the testing program *"iodrvdemo"* can be started as follows:

```
cd  /home/bin
insmod  pce660drv.ko
./iodrvdemo
```

Figure 30 illustrates testing the hardware connection by using *"iodrvdemo"*.

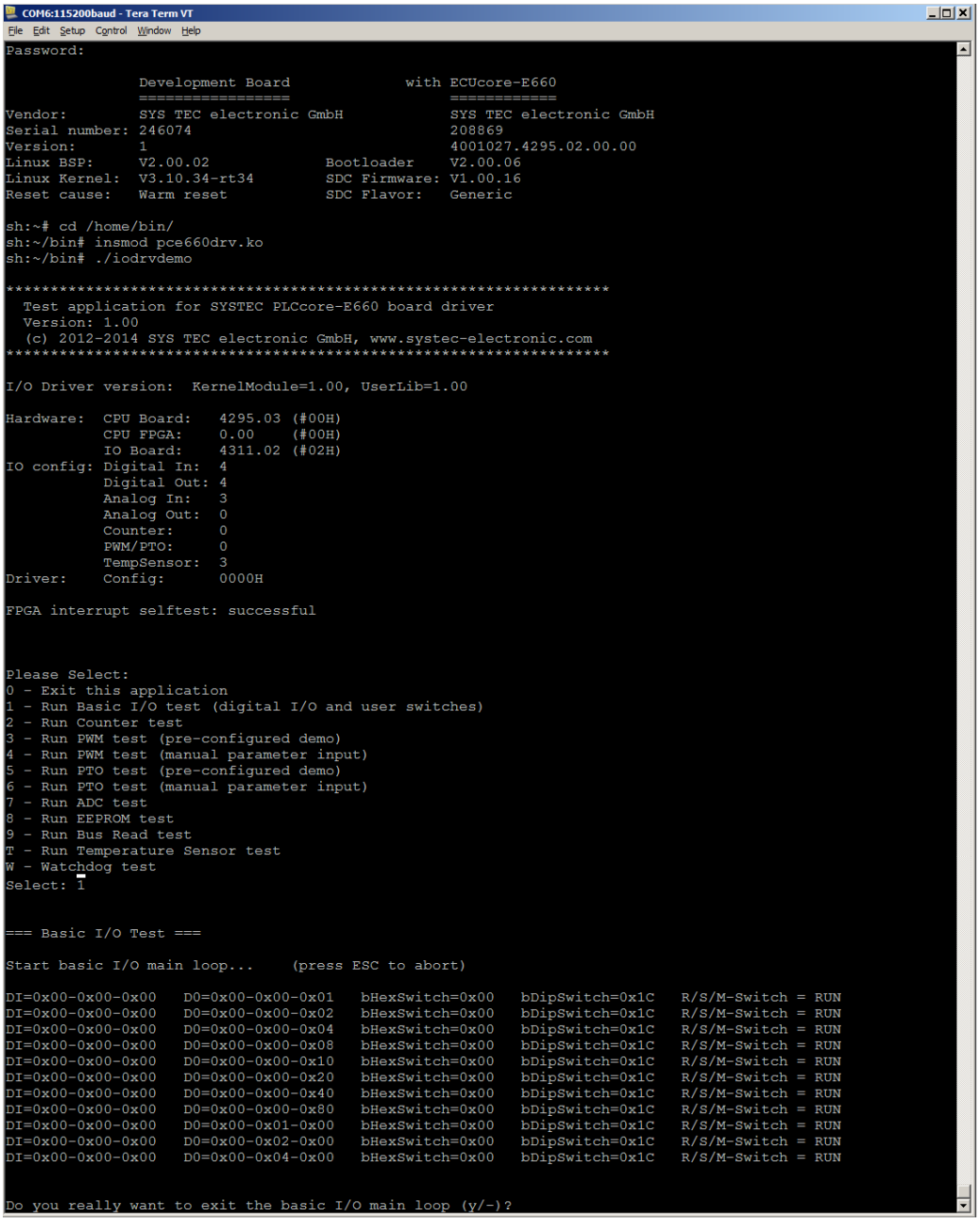


Figure 30: Testing the hardware connections using "iodrvdemo"

Appendix A: Firmware function scope of PLCcore-E660

Table 20 lists all firmware functions and function blocks available on the PLCcore-E660.

Sign explanation:

FB	Function block
FUN	Function
Online Help	*OpenPCS* online help
L-1054	Manual *"SYS TEC-specific extensions for OpenPCS / IEC 61131-3"*, Manual no.: L-1054)
PARAM:={0,1,2}	values 0, 1 and 2 are valid for the given parameter

Table 20: Firmware functions and function blocks of PLCcore-E660

Name	Type	Reference	Remark
PLC standard Functions and Function Blocks			
SR	FB	Online Help	
RS	FB	Online Help	
R_TRIG	FB	Online Help	
F_TRIG	FB	Online Help	
CTU	FB	Online Help	
CTD	FB	Online Help	
CTUD	FB	Online Help	
TP	FB	Online Help	
TON	FB	Online Help	
TOF	FB	Online Help	
Functions and Function Blocks for string manipulation			
LEN	FUN	L-1054	
LEFT	FUN	L-1054	
RIGHT	FUN	L-1054	
MID	FUN	L-1054	
CONCAT	FUN	L-1054	
INSERT	FUN	L-1054	
DELETE	FUN	L-1054	
REPLACE	FUN	L-1054	
FIND	FUN	L-1054	
GETSTRINFO	FB	L-1054	
CHR	FUN	L-1054	
ASC	FUN	L-1054	
STR	FUN	L-1054	
VAL	FUN	L-1054	
Functions and Function Blocks for OpenPCS specific task controlling			
ETRC	FB	L-1054	
PTRC	FB	L-1054	
GETVARDATA	FB	Online Help	
GETVARFLATADDRESS	FB	Online Help	
GETTASKINFO	FB	Online Help	

Functions and Function Blocks for handling of non-volatile data			
NVDATA_BIT	FB	L-1054	DEVICE:={0,1} see (1)
NVDATA_INT	FB	L-1054	DEVICE:={0,1} see (1)
NVDATA_STR	FB	L-1054	DEVICE:={0,1} see (1)
NVDATA_BIN	FB	L-1054	DEVICE:={0,1} see (1)
Functions and Function Blocks for handling of time			
GetTime	FUN	Online Help	
GetTimeCS	FUN	Online Help	
DT_CLOCK	FB	L-1054	
DT_ABS_TO_REL	FB	L-1054	
DT_REL_TO_ABS	FB	L-1054	
Functions and Function Blocks for Serial interfaces			
SIO_INIT	FB	L-1054	PORT:={0,1} see (2)
SIO_STATE	FB	L-1054	PORT:={0,1} see (2)
SIO_READ_CHR	FB	L-1054	PORT:={0,1} see (2)
SIO_WRITE_CHR	FB	L-1054	PORT:={0,1} see (2)
SIO_READ_STR	FB	L-1054	PORT:={0,1} see (2)
SIO_WRITE_STR	FB	L-1054	PORT:={0,1} see (2)
SIO_READ_BIN	FB	L-1054	PORT:={0,1} see (2)
SIO_WRITE_BIN	FB	L-1054	PORT:={0,1} see (2)
Functions and Function Blocks for CAN interfaces / CANopen			
CAN_GET_LOCALNODE_ID	FB	L-1008	NETNUMBER:={0}
CAN_CANOPEN_KERNEL_STATE	FB	L-1008	NETNUMBER:={0}
CAN_REGISTER_COBID	FB	L-1008	NETNUMBER:={0}
CAN_PDO_READ8	FB	L-1008	NETNUMBER:={0}
CAN_PDO_WRITE8	FB	L-1008	NETNUMBER:={0}
CAN_SDO_READ8	FB	L-1008	NETNUMBER:={0}
CAN_SDO_WRITE8	FB	L-1008	NETNUMBER:={0}
CAN_SDO_READ_STR	FB	L-1008	NETNUMBER:={0}
CAN_SDO_WRITE_STR	FB	L-1008	NETNUMBER:={0}
CAN_SDO_READ_BIN	FB	L-1008	NETNUMBER:={0}
CAN_SDO_WRITE_BIN	FB	L-1008	NETNUMBER:={0}
CAN_GET_STATE	FB	L-1008	NETNUMBER:={0}
CAN_NMT	FB	L-1008	NETNUMBER:={0}
CAN_RECV_EMCY_DEV	FB	L-1008	NETNUMBER:={0}
CAN_RECV_EMCY	FB	L-1008	NETNUMBER:={0}
CAN_WRITE_EMCY	FB	L-1008	NETNUMBER:={0}
CAN_RECV_BOOTUP_DEV	FB	L-1008	NETNUMBER:={0}
CAN_RECV_BOOTUP	FB	L-1008	NETNUMBER:={0}
CAN_ENABLE_CYCLIC_SYNC	FB	L-1008	NETNUMBER:={0}
CAN_SEND_SYNC	FB	L-1008	NETNUMBER:={0}

CANL2_INIT	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_SHUTDOWN	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_RESET	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_GET_STATUS	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_DEFINE_CANID	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_DEFINE_CANID_RANGE	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_UNDEFINE_CANID	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_UNDEFINE_CANID_RANGE	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_MESSAGE_READ8	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_MESSAGE_READ_BIN	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_MESSAGE_WRITE8	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_MESSAGE_WRITE_BIN	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_MESSAGE_UPDATE8	FB	L-1008	NETNUMBER:={0}	see (3)
CANL2_MESSAGE_UPDATE_BIN	FB	L-1008	NETNUMBER:={0}	see (3)
Functions and Function Blocks for Ethernet interfaces / UDP				
LAN_GET_HOST_CONFIG	FB	L-1054	NETNUMBER:={0}	
LAN_ASCII_TO_INET	FB	L-1054	NETNUMBER:={0}	
LAN_INET_TO_ASCII	FB	L-1054	NETNUMBER:={0}	
LAN_GET_HOST_BY_NAME	FB	L-1054	NETNUMBER:={0}	
LAN_GET_HOST_BY_ADDR	FB	L-1054	NETNUMBER:={0}	
LAN_UDP_CREATE_SOCKET	FB	L-1054	NETNUMBER:={0}	
LAN_UDP_CLOSE_SOCKET	FB	L-1054	NETNUMBER:={0}	
LAN_UDP_RECVFROM_STR	FB	L-1054	NETNUMBER:={0}	
LAN_UDP_SENDTO_STR	FB	L-1054	NETNUMBER:={0}	
LAN_UDP_RECVFROM_BIN	FB	L-1054	NETNUMBER:={0}	
LAN_UDP_SENDTO_BIN	FB	L-1054	NETNUMBER:={0}	
Functions and Function Blocks for Target Visualization				
HMI_REG_KEY_FUNCTION_TAB	FB	L-1321	HMI Version only	see (4)
HMI_SEL_KEY_FUNCTION_TAB	FB	L-1321	HMI Version only	see (4)
HMI_REG_EDIT_CONTROL_TAB	FB	L-1321	HMI Version only	see (4)
HMI_SEL_EVENT_HANDLER	FB	L-1321	HMI Version only	see (4)
HMI_GET_INPUT_EVENT	FB	L-1321	HMI Version only	see (4)
HMI_CLR_INPUT_EVENT_QUEUE	FB	L-1321	HMI Version only	see (4)
HMI_SEND_KEY_TO_BROWSER	FB	L-1321	HMI Version only	see (4)

(1) The PLCcore-E660 supports the following devices for saving of nonvolatile data:
DEVICE:=0: Nonvolatile data are written into file *"/home/plc/plcdata/PlcPData.bin"*. This file has a fix size of 32 kByte. By calling function blocks of type *NVDATA_Xxx* in a writing mode, the modified data is directly stored into file *"/home/plc/plcdata/PlcPData.bin"* ("*flush*"). Thus, unsecured data is not getting lost in case of power interruption.
DEVICE:=1: Nonvolatile data are written into EEPROM on PLCcore-E660. This EEPROM has a fix size of 2 kByte.

(2) Interface UART1 (PORT:=1) primarily serves as service interface to administer the PLCcore-E660. Hence, this interface should only be used for sign output. The module always tries to interpret and execute sign inputs as Linux commands (see section 6.6.1).

(3) The usage of Function Blocks from type *CANL2_Xxx* is only possible, if the according CAN interface is not used already by CANopen. Due to its necessary to disable the according CAN interface in the PLC configuration (see section 7.4.1), otherwise the Function Blocks from type *CANL2_Xxx* can't be used. Alternatively, entry *"Enable="* can directly be set to 0 within section *"[CANx]"* of the configuration file ***"/home/plc/bin/plccore-E660.cfg"*** (see section 7.4.2).

(4) The Function Blocks from type *HMI_Xxx* are only available for the HMI version of the PLCcore-E660 (Order number 3390085).

Appendix B: GNU GENERAL PUBLIC LICENSE

The Embedded Linux used on the PLCcore-E660 is licensed under GNU General Public License, version 2. The entire license text is specified below.

The PLC system used and the PLC and C/C++ programs developed by the user are **not** subject to the GNU General Public License!

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

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

/

/home 53
/home/etc/autostart 26, 45
/home/plc/bin/plccore-E660.cfg 42
/home/plc/plcdata/PlcPData.bin 76
/tmp 53, 55

A

Accessory 18
adduser 50
Administration
  System Requirements 39
autostart 26
Autostart 25, 45

B

Bitrate 43
Boot configuration 26, 45

C

CAN0 ..... 33, 37
CAN1 14
CANopen 10, 36
CANopen Chip 10
CANopen Master 10
CE conformity 6
CFG File 43
COM 32
Communication FB 30
Communication interfaces
  CAN 33
  COM 32
  ETH 33
Configuration
  Command 40
  PLC 41
Configuration Mode 27
Control Elements
  Error-LED 35
  Run/Stop Switch 34
  Run-LED 34

D

date 52
Deletion of PLC Program 35
deluser 51
Development Board
  Connections 13
  Control elements 17
  Jumper configuration 15
Development Kit 12
df (command) 54
Dimension 9
Display 38
Driver Development Kit 18, 71

E

EFI 40
EFI Shell
  Activation 27
EFI shell Command
  BoardID configuration 46
Embedded Linux 11
EMC law 6
Error-LED 35
ETH0 ..... 14, 33
  PLC program example 33
ETH1 ..... 14, 33

F

File System 53
Firmware version
  Selection 46
FTP
  Login to the PLCcore-E660 48
FTP Client 39
FUB 10

G

GNU 11
GPL 78
GUI
  Qt 11

H

hwclock 52

I

IL 10
iodrvdemo 72

K

Keyboard 11
KOP 10

L

Linux 11

M

Manuals
  Overview 7
Master Mode 43
Mouse 11

N

Node Address 43

O

OpenPCS 10

P

passwd....51
Pinout....19
PLC program example
ETH0....33
plccore-E660.cfg....42, 43, 57
PlcPData.bin....76
Predefined User Accounts....47
Process Image
Layout and Addressing....31
Programming....30

Q

Qt....11

R

ReadSectorTable....59
RTC setting....52
Run/Stop Switch....34
Deletion of PLC Program....35
Encoding....23
Run-LED....34

S

Selecting the firmware version....46
Setting the System Time....52
Shared Process Image
Activation....59
API Description....62
Example....67
Overview....58
signaling....61
Variable Pairs....59
ShPImgClntGetDataSect....63
ShPImgClntGetHeader....63
ShPImgClntLockSegment....64
ShPImgClntRelease....63
ShPImgClntSetNewDataSigHandler....63
ShPImgClntSetup....62
ShPImgClntSetupReadSectTable....64
ShPImgClntSetupWriteSectTable....64
ShPImgClntUnlockSegment....64
ShPImgClntWriteSectMarkNewData....65
shpimgdemo....65
shpimgdemo.tar.gz....65
SO-1116....65
SO-1117....71
Software Update
PLC Firmware....55
SpiderControl....11, 38
ST....10
System start....25

T

Telnet
Login to the PLCcore-E660....47
Telnet Client....39
Terminal Configuration....40
Terminal Program....39
Testing Hardware Connections....72
tShPImgLayoutDscrpt....62
tShPImgSectDscrp....62

U

UART0....32
UART1....14, 33
UART2....14, 33
UART3....33
UdpRemoteCtrl....33
UEFI Boot Command Prompt
Terminal Configuration....40
USB-RS232 Adapter Cable....18
User Accounts
Adding and deleting....50
Changing Passwords....51
Predefined....47

W

WEB Frontend....41
WinSCP....49
WriteSectorTable....59

Document:	System Manual PLCcore-E660
Document number:	L-1555e_1, July 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
Am Windrad 2
D-08468 Heinsdorfergrund
GERMANY
Fax: +49 (0) 3765 38600-4100
Email: info@systec-electronic.com