System Manual
PLCcore-E660

User Manual
Version 1.0

Ausgabe Juli 2014

Dokument-Nr.: L-1555d_1

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

Status/Änderungen

Status: Freigegeben

Datum/Version	Abschnitt	Änderung	Bearbeiter
2014/07/28 1.0	alle	Erstellung	Dr. Norbert Prang

Im Buch verwendete Bezeichnungen für Erzeugnisse, die zugleich ein eingetragenes Warenzeichen darstellen, wurden nicht besonders gekennzeichnet. Das Fehlen der © Markierung ist demzufolge nicht gleichbedeutend mit der Tatsache, dass die Bezeichnung als freier Warenname gilt. Ebenso wenig kann anhand der verwendeten Bezeichnung auf eventuell vorliegende Patente oder einen Gebrauchsmusterschutz geschlossen werden.

Die Informationen in diesem Handbuch wurden sorgfältig überprüft und können als zutreffend angenommen werden. Dennoch sei ausdrücklich darauf verwiesen, dass die Firma SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf den Gebrauch oder den Inhalt dieses Handbuches zurückzuführen sind. Die in diesem Handbuch enthaltenen Angaben können ohne vorherige Ankündigung geändert werden. Die Firma SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Ferner sei ausdrücklich darauf verwiesen, dass SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf falschen Gebrauch oder falschen Einsatz der Hard- bzw. Software zurückzuführen sind. Ebenso können ohne vorherige Ankündigung Layout oder Design der Hardware geändert werden. SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Informieren Sie sich:

Kontakt	**Direkt**	**Ihr Lokaler Distributor**
Adresse:	SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund GERMANY	Sie finden eine Liste unserer Distributoren unter http://www.systec-electronic.com/distributors
Angebots-Hotline:	+49 (0) 3765 38600-0 info@systec-electronic.com	
Technische Hotline:	+49 (0) 3765 38600-0 support@systec-electronic.com	
Fax:	+49 (0) 3765 38600-4100	
Webseite:	http://www.systec-electronic.com	

1. Auflage Juli 2014

Inhalt

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Vielen Dank, dass Sie sich für das SYS TEC PLCcore-E660 entschieden haben. Mit diesem Produkt verfügen Sie über einen innovativen und leistungsfähigen SPS-Kern. Aufgrund seiner integrierten Target-Visualisierung, hohen Performance sowie wegen seiner umfangreichen on-board Peripherie eignet er sich besonders gut als Kommunikations- und Steuerrechner für HMI Anwendungen.

Bitte nehmen Sie sich etwas Zeit, dieses Manual aufmerksam zu lesen. Es beinhaltet wichtige Informationen zur Inbetriebnahme, Konfiguration und Programmierung des PLCcore-E660. Es wird Ihnen helfen, sich mit dem Funktionsumfang und der Anwendung des PLCcore-E660 vertraut zu machen. Dieses Dokument wird ergänzt durch weitere Manuals, wie beispielsweise zum IEC 61131-Programmiersystem *OpenPCS* und zur CANopen-Erweiterung für IEC 61131-3. Eine Auflistung der relevanten Manuals zum PLCcore-E660 beinhaltet Tabelle 1 im Abschnitt 2. Bitte beachten Sie auch diese ergänzenden Dokumentationen.

Für weiter führende Informationen, Zusatzprodukte, Updates usw. empfehlen wir den Besuch unserer Website unter: http://www.systec-electronic.com. Der Inhalt dieser Webseite wird periodisch aktualisiert und stellt Ihnen stets die neuesten Software-Releases und Manual-Versionen zum Download bereit.

Anmerkungen zum EMV-Gesetz für das PLCcore-E660

Das PLCcore-E660 ist als Zulieferteil für den Einbau in ein Gerät (Weiterverarbeitung durch Industrie) bzw. als Entwicklungsboard für den Laborbetrieb (zur Hardware- und Softwareentwicklung) bestimmt.

Nach dem Einbau in ein Gerät oder bei Änderungen/Erweiterungen an diesem Produkt muss die Konformität nach dem EMV-Gesetz neu festgestellt und bescheinigt werden. Erst danach dürfen solche Geräte in Verkehr gebracht werden.

Die CE-Konformität gilt nur für den hier beschriebenen Anwendungsbereich unter Einhaltung der im folgenden Handbuch gegebenen Hinweise zur Inbetriebnahme! Das PLCcore-E660 ist ESD empfindlich und darf nur an ESD geschützten Arbeitsplätzen von geschultem Fachpersonal ausgepackt und gehandhabt bzw. betrieben werden.

Das PLCcore-E660 ist ein Modul für den Bereich Automatisierungstechnik. Durch die Programmierbarkeit nach IEC 61131-3 und die Verwendung von CAN-Bus und Ethernet-Standard-Netzwerkschnittstelle für verschiedenste Automatisierungslösungen, sowie dem standardisierten Netzwerkprotokoll CANopen ergeben sich geringere Entwicklungszeiten bei günstigen Kosten der Hardware. Durch die on-board Realisierung der SPS-Funktionalität mit optionaler CANopen-Netzwerkschicht ist die Erstellung einer Firmware durch den Anwender überflüssig geworden.

2 Übersicht / Wo finde ich was?

Das PLCcore-E660 basiert auf der Hardware des ECUcore-E660 und erweitert dieses um SPS-spezifische Funktionalitäten (SPS-Firmware, Target-Visualisierung). Für die Hardwarekomponenten wie das ECUcore-E660 bzw. das PLCcore-E660 selbst (die Hardware beider Module ist identisch), die Developmentboards sowie Referenzschaltungen existieren eigene Hardware-Manuals. Softwareseitig wird das PLCcore-E660 mit der IEC 61131-3 konformen Programmierumgebung *OpenPCS* programmiert. Zu *OpenPCS* existieren ebenfalls eigene Handbücher, die den Umgang mit dem Programmierwerkzeug und SYS TEC-spezifische Erweiterungen dazu beschreiben. Diese sind Bestandteil des Software-Paketes *"OpenPCS"*. Tabelle 1 listet die für das PLCcore-E660 relevanten Manuals auf.

Tabelle 1: Übersicht relevanter Manuals zum PLCcore-E660

Informationen über…	**In welchem Manual?**
Grundlegende Informationen zum PLCcore-E660 (Konfiguration, Administration, Prozessabbild, Anschlussbelegung, Firmwareupdate, Referenzdesigns usw.)	In diesem Manual
Entwicklung anwenderspezifischer C/C++ Applikationen für das ECUcore-E660 / PLCcore-E660, VMware-Image des Linux-Entwicklungssystems	System Manual ECUcore-E660 (Manual-Nr.: L-1554)
Hardware-Beschreibung zum ECUcore-E660 / PLCcore-E660, Referenzdesigns usw.	Hardware Manual ECUcore-E660 (Manual-Nr.: L-1562)
Developmentboard zum ECUcore-E660 / PLCcore-E660, Referenzdesigns usw.	Hardware Manual Development Board E660 (Manual-Nr.: L-1563)
Driver Development Kit (DDK) für das ECUcore-E660	Software Manual Driver Development Kit (DDK) für ECUcore-E660 (Manual-Nr.: L-1561)
Grundlagen zum IEC 61131-Programmiersystem *OpenPCS*	Kurzanleitung Programmiersystem (Eintrag "*OpenPCS Dokumentation*" in der *OpenPCS*-Programmgruppe des Startmenüs) (Manual-Nr.: L-1005)
Vollständige Beschreibung zum IEC 61131-Programmiersystem *OpenPCS*, Grundlagen der SPS-Programmierung nach IEC 61131-3	Online-Hilfe zum *OpenPCS*-Programmiersystem
Befehlsübersicht und Beschreibung der Standard-Funktionsbausteine nach IEC 61131-3	Online-Hilfe zum *OpenPCS*-Programmiersystem
SYS TEC-Erweiterung für IEC 61131-3: - Stringfunktionen - UDP-Funktionsbausteine - SIO-Funktionsbausteine - FB für RTC, Counter, EEPROM, PWM/PTO	User Manual *"SYS TEC spezifische Erweiterungen für OpenPCS / IEC 61131-3"* (Manual-Nr.: L-1054)

Informationen über…	In welchem Manual?
CANopen-Erweiterung für IEC 61131-3 (Netzwerkvariablen, CANopen-Funktionsbausteine)	User Manual *"CANopen-Erweiterung für IEC 61131-3"* (Manual-Nr.: L-1008)
HMI Erweiterungen für IEC 61131-3: - HMI Funktionsbausteine - Grundlagen zu Spider Control	User Manual *"SYS TEC spezifische HMI Erweiterungen für OpenPCS / IEC 61131-3"* (Manual-Nr.: L-1231)
Lehrbuch zur SPS-Programmierung nach IEC 61131-3	SPS-Programmierung mit IEC 61131-3 John/Tiegelkamp Springer-Verlag ISBN: 3-540-66445-9 (in gekürzter Form auch als PDF auf *OpenPCS* Installations-CD enthalten)

Abschnitt 4 dieses Manuals erläutert die **Inbetriebnahme des PLCcore-E660** auf Basis des Development Kit für das PLCcore-E660.

Abschnitt 5 beschreibt die **Anschlussbelegung** des PLCcore-E660.

Abschnitt 6 erklärt Details zur **Anwendung des PLCcore-E660**, so z.B. den **Aufbau des Prozessabbildes**, die **Bedeutung der Bedienelemente** und vermittelt grundlegende Informationen zur Programmierung des Moduls. Weiterhin werden hier Informationen zur Nutzung der CAN-Schnittstellen in Verbindung mit **CANopen** gegeben.

Abschnitt 7 beschreibt **Details zur Konfiguration des PLCcore-E660**, so z.B. die Konfiguration von Ethernet- und CAN-Schnittstellen, den Linux-Autostartvorgang sowie die Auswahl der Firmwarevariante. Weiterhin wird hier die **Administration des PLCcore-E660** erläutert, so z.B. die Anmeldung am System, die Nutzerverwaltung und die Durchführung von Softwareupdates.

Abschnitt 8 erläutert die **Adaption von Ein- und Ausgängen** sowie des **Prozessabbildes** und behandelt schwerpunktmäßig den Datenaustausch zwischen einem SPS-Programm und einer anwenderspezifischen C/C++ Applikation über das **Shared Prozessabbild**.

3 Produktbeschreibung

Das PLCcore-E660 erweitert die Produktpalette der Firma SYS TEC electronic GmbH im Steuerungsbereich um ein weiteres innovatives Produkt. In Form eines Aufsteckmoduls ("Core") stellt es dem Anwender eine vollständige Kompakt-SPS mit integrierter Target-Visualisierung zur Verfügung. Dank der CAN- und Ethernet-Schnittstellen ist das PLCcore-E660 optimal zum Aufbau von anwenderspezifischen HMI (**H**uman **M**achine **I**nterface) Applikationen geeignet.

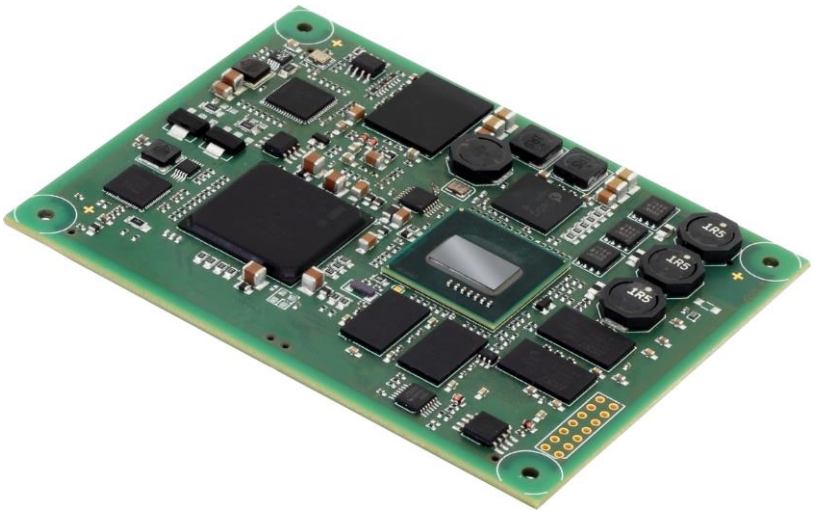


Bild 1: Ansicht des PLCcore-E660

Einige herausragende Merkmale des PLCcore-E660 sind:

- leistungsfähiger CPU-Kern (Intel Atom E660 1.3 GHz CPU, CPU clock 1.0 GHz)
- 4 GByte SDRAM Memory, 4 GByte FLASH Memory
- 1 MiB dedicated Video RAM
- LVDS LCD Controller mit Unterstützung für Displays bis max. 800x600 Pixel Auflösung bei 16 Bit Farbtiefe
- 4 USB Host interfaces
- 2 USB Device interfaces
- Anschlussmöglichkeit von HMI devices wie USB Keyboard, USB Mouse und DVI Graphic Cards
- 2x 10/100/1000 Mbps Ethernet LAN interface
- 2x CAN 2.0B Interface, nutzbar als CANopen-Manager (CiA 302 konform)
- 4x Asynchronous Serial Ports (UART)
- Auf dem Development Board:
  8 digitale Eingänge, 9 digitale Ausgänge
- Auf dem Core-Modul:
  9 digitale Eingänge, 9 digitale Ausgänge (Standardkonfiguration, modifizierbar über DDK)
- SPI und I^2C extern nutzbar
- On-board Peripherie: RTC, Watchdog, Temperatursensor, SDC
- On-board Software: Linux, SPS-Firmware, CANopen-Master, HTTP- und FTP-Server – zusätzlich in der HMI-Version Target-Visualisierung und HMI Funktionsbaustein-Bibliothek
- Programmierbar nach IEC 61131-3 und in C/C++
- Funktionsbausteinbibliotheken für Kommunikation (CANopen, Ethernet und UART)
- Support SPS-typischer Bedienelemente (z.B. Run/Stop-Schalter, Run-LED, Error-LED)
- Linux-basiert, dadurch weitere beliebige Anwenderprogramme parallel ausführbar
- Einfache, HTML-basierte Konfiguration über WEB-Browser
- Remote Login über Telnet
- Geringe Abmessungen (105 mm x 80 mm)

Für das PLCcore-E660 sind verschiedene Firmware-Varianten verfügbar, die sich in dem zur Kommunikation zwischen Programmier-PC und PLCcore-E660 verwendeten Protokoll unterscheiden:

Art.-Nr.: 3390089/Z3:	PLCcore-E660/Z3 (RS232 ohne Target-Visualisierung) Kommunikation mit Programmier-PC via Siemens 3964(R) (Interface UART1)
Art.-Nr.: 3390089/Z4:	PLCcore-E660/Z4 (CANopen ohne Target-Visualisierung) Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
Art.-Nr.: 3390089/Z5:	PLCcore-E660/Z5 (Ethernet ohne Target-Visualisierung) Kommunikation mit Programmier-PC via UDP-Protokoll (Interface ETH0)
Art.-Nr.: 3390090/Z3:	PLCcore-E660/Z4 (RS232 mit Target-Visualisierung) Kommunikation mit Programmier-PC via Siemens 3964(R) (Interface UART1)
Art.-Nr.: 3390090/Z4:	PLCcore-E660/Z4 (CANopen mit Target-Visualisierung) Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
Art.-Nr.: 3390090/Z5:	PLCcore-E660/Z5 (Ethernet mit Target-Visualisierung) Kommunikation mit Programmier-PC via UDP-Protokoll (Interface ETH0)

Die Verfügbarkeit einer SPS als aufsteckbares "Core“ und die damit verbundenen geringen Abmessungen reduzieren den Aufwand und die Kosten bei der Entwicklung anwenderspezifischer Steuerungen enorm. Gleichzeitig eignet sich das PLCcore-E660 besonders als Basiskomponente für

anwenderspezifische HMI Baugruppen sowie als intelligenter Netzwerkknoten zur dezentralen Verarbeitung von Prozesssignalen (CANopen und UDP)

Die On-Board-Firmware des PLCcore-E660 beinhaltet die gesamte Target-Visualisierung sowie die SPS-Laufzeitumgebung einschließlich der CANopen-Anbindung mit CANopen-Masterfunktionalität. Dadurch ist das Modul in der Lage, sowohl Aufgaben der Mensch-Maschine-Kommunikation als auch Steuerungsaufgaben wie die Verknüpfung von Ein- und Ausgängen oder die Umsetzung von Regelalgorithmen vor Ort durchzuführen. Über das CANopen-Netzwerk, über Ethernet (UDP-Protokoll) sowie über die seriellen Schnittstellen (UART) können Daten und Ereignisse mit anderen Knoten (z.B. übergeordnete Zentralsteuerung, I/O-Slaves usw.) ausgetauscht werden. Darüber hinaus ist die Anzahl der Ein- und Ausgänge sowohl lokal als auch dezentral mit Hilfe von CANopen-Geräten erweiterbar. Ideal geeignet ist hierfür der CANopen-Chip, der ebenfalls als Aufsteckmodul für den Einsatz in anwenderspezifischen Applikationen konzipiert wurde.

Das PLCcore-E660 bietet 9 digitale Eingänge (DI0...DI8) und 9 digitale Ausgänge (DO0...DO8). Mit Hilfe des Driver Development Kit (SO-1103) kann diese Standard-I/O-Konfiguration an die spezifischen Applikationsbedingungen adaptiert werden (siehe Abschnitte 4.5.2 und 8.2). Die Ablage des SPS-Programms in der On-board Flash-Disk des Moduls ermöglicht den autarken Wiederanlauf nach einer Spannungsunterbrechung.

Die Programmierung des PLCcore-E660 erfolgt nach IEC 61131-3 mit dem Programmiersystem *OpenPCS* der Firma infoteam Software GmbH (http://www.infoteam.de). Dieses Programmiersystem wurde von der Firma SYS TEC electronic GmbH für das PLCcore-E660 erweitert und angepasst. Damit kann das PLCcore-E660 sowohl grafisch in KOP/FUB, AS und CFC als auch textuell in AWL oder ST programmiert werden. Der Download des SPS-Programms auf das Modul erfolgt in Abhängigkeit von der verwendeten Firmware über Ethernet oder CANopen. Die Adressierung der Ein- und Ausgänge – und damit der Aufbau des Prozessabbildes – wurde an das Schema der SYS TEC-Kompaktsteuerungen angelehnt. Analog zu allen anderen SYS TEC-Steuerungen unterstützt auch das PLCcore-E660 die Rückdokumentation des SPS-Programms aus der Steuerung sowie als Debugfunktionalität das Beobachten und Setzen von Variablen, Einzelzyklus, Breakpoints und Einzelschritt.

Die integrierte Target-Visualisierung des PLCcore-E660 basiert auf dem *SpiderControl MicroBrowser* der Firma iniNet Solutions GmbH (http://www.spidercontrol.net). Sie ermöglicht sowohl die Anzeige von Prozesswerten aus der SPS als auch die Weiterleitung von Benutzeraktionen an die SPS (z.B. Eingaben über Maus oder Tastatur).

Das PLCcore-E660 basiert auf einem Embedded Linux als Betriebssystem. Dadurch ist es möglich, simultan zur SPS-Firmware noch weitere, anwenderspezifische Programme abzuarbeiten. Falls erforderlich, können diese anwenderspezifischen Programme unter Nutzung des Prozessabbildes Daten mit dem SPS-Programm austauschen. Weitere Informationen hierzu beinhaltet der Abschnitt 8.

Das auf dem PLCcore-E660 eingesetzte Embedded Linux ist unter der GNU General Public License, Version 2 lizenziert. Der entsprechende Lizenztext ist im Anhang B aufgeführt. Die vollständigen Quellen des verwendeten LinuxBSP sind im Softwarepaket **SO-1116** ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-E660") enthalten. Sollten Sie die Quellen des LinuxBSP unabhängig vom VMware-Image des Linux-Entwicklungssystems benötigen, setzen Sie sich bitte mit unserem Support in Verbindung:

support@systec-electronic.com

Das verwendete SPS-System sowie die vom Anwender entwickelten SPS- und C/C++ Programme unterliegen **nicht** der GNU General Public License!

4 Development Kit PLCcore-E660

4.1 Übersicht

Auf Grund des im Kit enthaltenen Development Boards erlaubt das Development Kit PLCcore-E660 eine schnelle Inbetriebnahme des PLCcore-E660 und vereinfacht das Design von Prototypen für nutzerspezifische Applikationen auf diesem Modul.

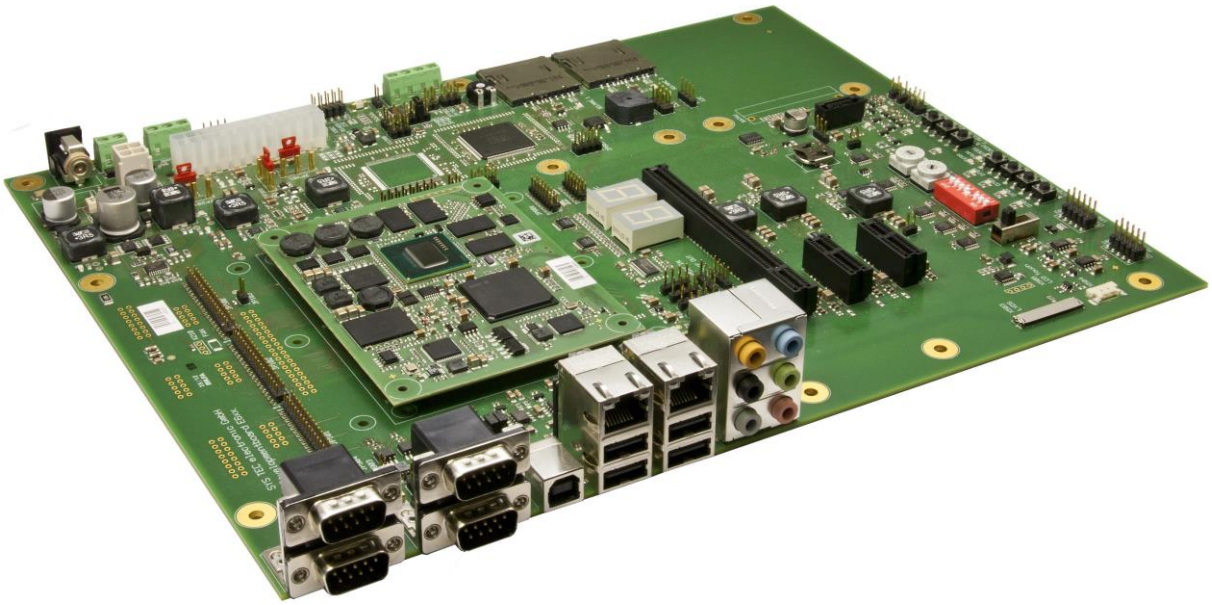


Bild 2: Development Kit PLCcore-E660

Das Development Kit PLCcore-E660 gewährleistet eine rasche und problemlose Inbetriebnahme des PLCcore-E660. Es vereint dazu alle notwendigen Hard- und Software-Komponenten, die zur Erstellung eigener HMI Applikationen erforderlich sind: als Kernkomponente das PLCcore-E660 selbst, das zugehörige Development Board mit der Anschlussmöglichkeit von HMI-Devices (Tastatur, Maus, DVI-Monitor an einer SDVO-Grafikkarte), verschiedenen Möglichkeiten der Stromversorgung, Ethernet-Interfaces, CAN-Bus-Anschlüssen, USB-Anschlüssen,, SD-Karten-Slots, 4 Tastern und 4 LEDs als Steuerelemente für digitale Ein- und Ausgänge. Weiterhin gehören zum Kit das IEC 61131-Programmiersystems *OpenPCS*, der *SpiderControl HMI Editor* zum Erstellen der Grafikseiten sowie weiteres Zubehör.

Damit bildet das Developmentkit die ideale Plattform zur Entwicklung anwenderspezifischer HMI-Applikationen auf Basis des PLCcore-E660. Ebenso ermöglicht das Kit den kostengünstigen Einstieg in die Welt der dezentralen Automatisierungstechnik. Bereits die im Kit enthaltenen Komponenten ermöglichen die Erweiterung der Ein- und Ausgänge des PLCcore-E660 durch CANopen-I/O-Baugruppen. Damit kann das Development Kit auch für Projekte eingesetzt werden, die eine SPS mit Netzwerkanbindung erfordern.

Das Development Kit PLCcore-E660 beinhaltet folgende Komponenten:

- PLCcore-E660
- Developmentboard für PLCcore-E660
- 24V Steckernetzteil
- Ethernet-Kabel
- RS232-Kabel
- DVD mit Programmiersoftware, Beispielen, Dokumentation und weiteren Tools (SO-1116)

Als Software-Entwicklungsplattform sowie Debug-Umgebung für das PLCcore-E660 dient das im Kit enthaltene IEC 61131-Programmiersystem *OpenPCS*. Dadurch kann das Modul sowohl grafisch in KOP/FUB, AS und CFC als auch textuell in AWL oder ST programmiert werden. Der Download des SPS-Programms auf das PLCcore-E660 erfolgt in Abhängigkeit von der verwendeten Firmware über Ethernet oder CANopen. Leistungsfähige Debug-Funktionalitäten wie das Beobachten und Setzen von Variablen, Einzelzyklus, Breakpoints und Einzelschritt erleichtern die Entwicklung und Inbetriebnahme von Anwendersoftware für das Modul.

4.2 Elektrische Inbetriebnahme des Development Kit PLCcore-E660

Das für den Betrieb des Development Kit PLCcore-E660 notwendige Steckernetzteil sowie die erforderlichen Ethernet- und RS232-Kabel sind bereits im Lieferumfang des Kits enthalten. Für die Inbetriebnahme des Kits sind mindestens die Anschlüsse für Versorgungsspannung (X200/X201), UART1 (P901 Top) und ETH0 (X1001) erforderlich. Einen vollständigen Überblick über die Anschlüsse des Development Kit PLCcore-E660 vermittelt Tabelle 2.

Tabelle 2: Anschlüsse des Development Kit PLCcore-E660

Anschluss	Bezeichnung auf dem Development Board	Bemerkung
Haupt-Stromversorgung	X200 or X201 + X217	Stromversorgungsanschlüsse mit folgenden Parametern: 24 VDC, 2 – 4 A (abhängig von der verwendeten Peripherie). X217 ist ein zusätzlicher Anschluss für eine Front-Panel-LED.
Alternative Stromversorgung	X208 und X209	Stromversorgungsanschluss für ATX2.2 Standard-interface – muss konfiguriert werden mit den Jumpern JP201, JP202, JP300 and JP301
PCIe0 (PCIexpress)	X400	Slot für eine PCIexpress-Card Bezeichnet als "PCIe Slot 1" in Bild 3
PCIe1 (PCIexpress)	X401	Slot für eine PCIexpress-Card Bezeichnet als "PCIe Slot 2" in Bild 3
SDVO-Interface	X501	Slot für eine SDVO-Grafikkarte (Grafikkarte ist im Kit enthalten).
LCD-Interface	X503 und X504	Anschlüsse für ein LCD-Display, LVDS – X503 und Backlight – X504. Display AOU G070VW01V0 wird unterstützt.
LCD-Touchscreen-Interface	X503	Nicht unterstützt
SD-Card-Slot 1	X601	Die SD-Card enthält das Embedded-Linux-Betriebssystem (siehe Dokument L-1554, zur Erstellung eines SD-Card-Images).
SD-Card-Slot 2	X604	Optionales SD-Card-Interface. Remark: JP600 muss konfiguriert sein. Das SD-Card-Interface ist auch verbunden mit einem eMMC auf dem ECUcore. Deshalb kann das externe SD-Card-Interface nur verwendet werden mit einer speziellen Variante des ECUcore, bei dem kein eMMC bestückt ist!
SATA0	X600	Dieses Interface kann benutzt werden in Kombination mit einem SATADOM-Device. Bemerkung: JP604 muss konfiguriert sein.
SATA1	X602 + X605	Dieses Interface (X602) wird benutzt für ein SATA-Device (2,5'' Hard Disk). X605 ist ein zusätzlicher Anschluss für eine Front-Panel-LED.

Anschluss	Bezeichnung auf dem Development Board	Bemerkung
LPC	X607	Dieses Interface dient als LPC-Interface. Bemerkung: JP601 und JP603 müssen konfiguriert sein.
I2C	X701	Dieses Interface kann benutzt werden für ein nutzerspezifisches I2C -Interface. Bemerkung: JP701 muss konfiguriert sein.
SPI	X603	Dieses Interface kann benutzt werden für ein nutzerspezifisches SPI-Interface. Bemerkung: JP704 muss konfiguriert sein.
AUDIO	X800	Dieses Interface kann benutzt werden für ein nutzerspezifisches AUDIO-Interface. Bemerkung: JP800 muss konfiguriert sein.
CAN1 (CAN)	P900A Bottom oder X900	Dieses Interface kann vom Nutzerprogramm frei benutzt werden. Bemerkung: JP900 + JP903 + JP904 + JP909 müssen konfiguriert sein
UART0 (RS232)	P900B Top	Dieses Interface kann vom Nutzerprogramm frei benutzt werden. Bemerkung: JP904 + JP909 müssen konfiguriert sein
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. Bemerkung: JP909 muss konfiguriert sein
UART2 (RS232)	P901A Bottom	Dieses Interface kann vom Nutzerprogramm frei benutzt werden. Bemerkung: JP909 muss konfiguriert sein
UART3 (RS232 or RS485)	X902	Dieses Interface kann vom Nutzerprogramm frei benutzt werden. Bemerkung: JP902 + JP905 + JP906 + JP907 müssen konfiguriert sein
SDC UART1 (RS232)	X901	Nicht unterstützt
USB0 (HOST)	X1000 Bottom	Dieses Interface dient als USB HOST Interface.
USB1 (HOST)	X1000 Middle	Dieses Interface dient als USB HOST Interface.
USB2 (HOST)	X1001 Bottom	Dieses Interface dient als USB HOST Interface.
USB3 (HOST)	X1001 Middle	Dieses Interface dient als USB HOST Interface.
USB4 (HOST)	X1003	Dieses Interface dient als USB HOST Interface.
USB5 (HOST)	X1003	Dieses Interface dient als USB HOST Interface.
USB (Device)	X1002	Nicht unterstützt
ETH0 (Ethernet)	X1001 Top	Dieses Interface dient als Kommunikationsinterface mit dem Linux-Entwicklungssystem (Programmier-PC) und kann ebenso vom Nutzerprogramm frei benutzt werden.

Anschluss	Bezeichnung auf dem Development Board	Bemerkung
ETH1 (Ethernet)	X1000 Top	Dieses Interface kann vom Nutzerprogramm frei benutzt werden.

Bild 3 zeigt die Lage der wichtigsten Anschlüsse auf dem Developmentboard für das PLCcore-E660. Anstelle des mitgelieferten Steckernetzteiles kann die Stromversorgung des Kit optional auch über X200 mit einer externen Quelle von 24 V/2 ... 4 A erfolgen.

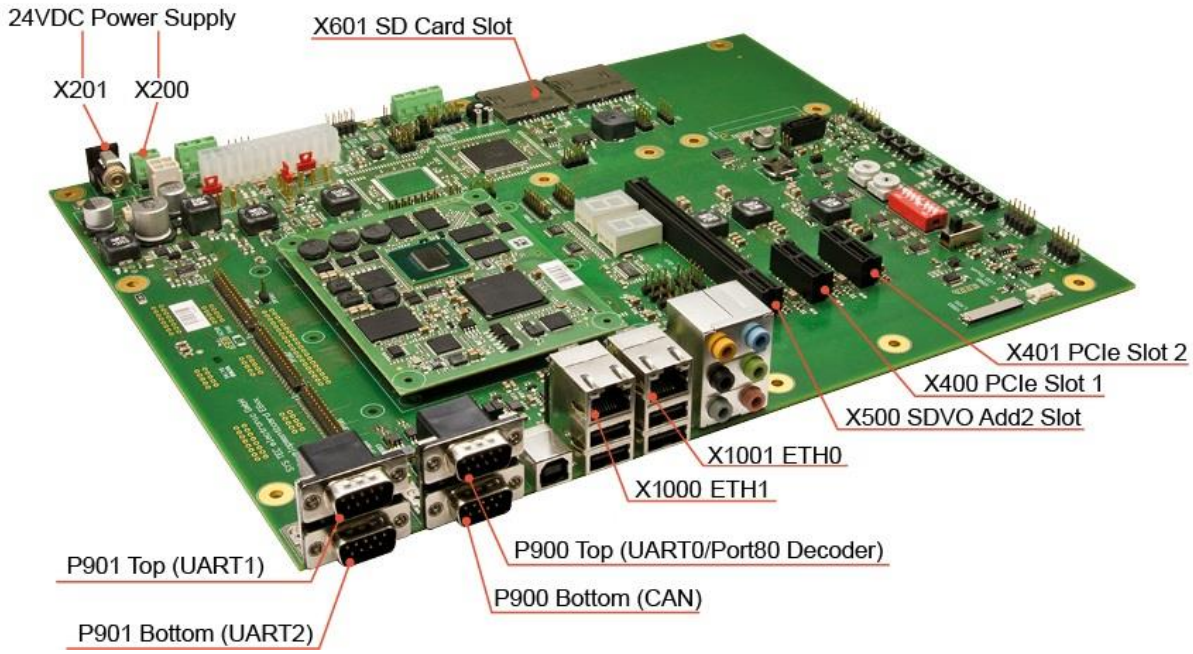


Bild 3: Lage der wichtigsten Anschlüsse auf dem Developmentboard für das PLCcore-E660

Hinweis:	**Bei der Inbetriebnahme sind zunächst eine SD-Karte (X601), die Kabel für Ethernet (ETH0, X1001) und RS232 (UART1, P901 Top) anzuschließen, erst danach ist die Versorgungsspannung (X200 / X201) zu aktivieren.**

Wichtig:	**Das Core-Modul sollte nicht ohne CPU-Kühlkörper betrieben werden. Bild 3 ist nicht vollständig, aber es zeigt das Design des Core-Moduls besser, als wenn ein Kühlkörper das Modul unter ihm verbirgt.**

4.3 Jumper-Konfiguration des Development Kits

Das Development Kit PLCcore-E660 konfiguriert mit einer Default-Jumper-Konfiguration. Tabelle 3 enthält die Jumper des Development-Boards und beschreibt deren Bedeutung und ihre Default-Optionen.

Tabelle 3: Jumper des Development Boards für den ECUcore-E660

Steuerelement	Name	Default-Option	Bedeutung
SDC_BOOT0	JP103	offen	Boot-Option des SDC
ATX /PS_ON	JP200	geschlossen	ATX power supply “ON”
12V power	JP201	geschlossen (2-3)	12V Power onboard-geregelt oder geliefert vom ATX Power-Supply-Interface (Default = Onboard)
Power good	JP202	geschlossen (2-3)	Power-Good-Signal vom Onboard-Spannungsregler oder vom Onboard-Spannungsregler und ATX
5V power	JP300	geschlossen (2-3)	5V Power onboard-geregelt oder geliefert vom ATX Power-Supply-Interface (Default = Onboard)
3V3 power	JP301	geschlossen (2-3)	3V3 Power onboard-geregelt oder geliefert vom ATX Power-Supply-Interface (Default = Onboard)
LCD Touch	JP500 + JP502	offen	Nicht unterstützt
SD card slot 2	JP600	offen	SD-Card-Slot 2 verfügbar an X604, wenn der eMMC auf dem ECUcore nicht bestückt ist
LPC	JP601 + JP603	offen	LPC Interface verfügbar an X607
SATA0	JP604	offen	Power-Option für SATA0 (SATADOM)-Interface
Watchdog	JP700	offen	Watchdog disable
I2C	JP701	offen	I2C Interface verfügbar an X701
ADC	JP702 + JP703	offen	ADC Interface verfügbar an X700 und R721
SPI	JP704	offen	SPI Interface verfügbar an X603
SPI	JP705	geschlossen	EEPROM verfügbar am SPI-Interface
AUDIO	JP800	geschlossen (7-8) offen (5-6) offen (3-4) geschlossen (1-2)	AUDIO Interface verfügbar an X800
CAN	JP900	offen	Enable 120 Ohm Bus-Abschlusswiderstand
CAN	JP903	offen	Enable 5V Ausgang
CAN	JP909	geschlossen	CAN ist verfügbar an P900A und X900 Enable CAN-Interface (EGT20H oder SDC) Bemerkung: Wenn CAN via SDC ausgewählt ist, kann UART0 nicht benutzt werden.

Steuerelement	Name	Default-Option	Bedeutung
CAN	JP904	geschlossen (13-14) geschlossen (15-16)	CAN via SDC ist ausgewählt. Bemerkung: Wenn CAN via SDC ausgewählt ist, kann UART0 nicht benutzt werden.
UART0	JP901	offen	UART0 ist verfügbar an P900B. Bemerkung: Wenn CAN via SDC ausgewählt ist, kann UART0 nicht benutzt werden.
UART1	JP904	geschlossen (1-2) geschlossen (7-8)	UART1 ist verfügbar an P901B
UART2	JP904	geschlossen (3-4) geschlossen (9-10)	UART2 ist verfügbar an P901A
UART3	JP902 + JP905 + JP906 + JP907	offen	UART3 ist verfügbar an X902. Verschiedene Konfigurationsmodi (RS232 oder RS485, "Nur Mitlesen" usw.) können ausgewählt werden.
SDC UART1	JP904	offen (5-6) offen (11-12)	SDC UART1 ist verfügbar an X901 (nicht unterstützt) Bemerkung: SDC UART1 ist ausgewählt, CAN via SDC kann nicht benutzt werden.

4.4 Bedienelemente des Development Kit PLCcore-E660

Das Development Kit PLCcore-E660 ermöglicht eine einfache Inbetriebnahme des PLCcore-E660. Es verfügt über verschiedene Bedienelemente sowohl zur Konfiguration des Moduls als auch zur Simulation von Ein- und Ausgängen für den Einsatz des PLCcore-E660 als SPS-Kern. Tabelle 4 listet die einzelnen Bedienelemente des Developmentboard auf und beschreibt deren Bedeutung.

Tabelle 4: Bedienelemente des Developmentboard für das PLCcore-E660

Bedienelement	Bezeichnung	Bedeutung
Taster 0	S704	Digitaler Eingang DI0 (Prozessabbild: %IX0.0)
Taster 1	S705	Digitaler Eingang DI1 (Prozessabbild: %IX0.1)
Taster 2	S706	Digitaler Eingang DI2 (Prozessabbild: %IX0.2)
Taster 3	S707	Digitaler Eingang DI3 (Prozessabbild: %IX0.3)
Taster 4	C RES	Taster für Kaltstart – Power-Button
Taster 5	W RES	Taster für Warmstart
Taster6	SDC RES	Reset-Taster für den SDC (System Diagnostic Controller)
LED 0	D701	Digitaler Ausgang DO0 (Prozessabbild: %QX0.0)
LED 1	D702	Digitaler Ausgang DO1 (Prozessabbild: %QX0.1)
LED 2	D703	Digitaler Ausgang DO2 (Prozessabbild: %QX0.2)
LED 3	D704	Digitaler Ausgang DO3 (Prozessabbild: %QX0.3)
Potentiometer (ADC)	R721	Analogeingang AI0 – auf dem Development-Board bestückt in der Nähe von JP702
Schraubklemmen	X700	Klemmen für die Analogeingänge AI1 und AI2 – 4 Klemmen in einem Block, je 2 für jeden AI
Run/Stop-Schalter	3-Positionen-Schalter mit der Bezeichnung • RUN • STOP • MRES	Run / Stop für Abarbeitung des SPS-Programms, Rücksetzen der Steuerung (siehe Abschnitt 6.7.1)
Run-LED	RUN	Anzeige Aktivitätszustand der SPS (siehe Abschnitt 6.7.2)
Error-LED	ERR	Anzeige Fehlerzustand der SPS (siehe Abschnitt 6.7.3)
Hexcodier-Schalter	S702/S703	Dieses Steuerelement kann vom nutzerspezifischen SPS-Steuerprogramm frei benutzt werden
DIP-Schalter	S700	Dieses Steuerelement kann vom nutzerspezifischen SPS-Steuerprogramm frei benutzt werden
Beeper	L700	Dieses Steuerelement kann vom nutzerspezifischen SPS-Steuerprogramm frei benutzt werden (z. B. zur Aktivitätsanzeige usw.)
7-Segment-Anzeige	U1205 + U1206	Anzeige verschiedener Codes während des Boot-Up-Vorgangs. "00" bedeutet "Boot-Up ohne Fehler".

Eine vollständige Auflistung des Prozessabbildes enthält Tabelle 11 im Abschnitt 6.5.1.

4.5 Optionales Zubehör

4.5.1 USB-RS232 Adapter Kabel

Das SYS TEC USB-RS232 Adapter Kabel (Bestellnummer 3234000) stellt eine RS232-Schnittstelle über einen USB-Port des PC zur Verfügung. In Verbindung mit einem Terminalprogramm ermöglicht es die Konfiguration des PLCcore-E660 von PCs, wie z.B. Laptops, die selber keine physikalische RS232-Schnittstelle mehr besitzen (siehe Abschnitt 7.1).

Bild 4: SYS TEC USB-RS232 Adapter Kabel

4.5.2 Driver Development Kit (DDK)

Das Driver Development Kit für das ECUcore-E660 (Bestellnummer SO-1117) ermöglicht dem Anwender die eigenständige Anpassung der I/O-Ebene an ein selbst entwickeltes Baseboard. Einen Überblick zum Driver Development Kit vermittelt Abschnitt 8.2.

5 Anschlussbelegung des PLCcore-E660

Die Anschlüsse des PLCcore-E660 sind über vier auf der Modulunterseite montierte, doppelreihige Buchsenleisten nach außen geführt (X1400 bis X1403, siehe Bild 5).

Folgend genannte Steckverbindertypen kommen zum Einsatz.
ECUcore-E660:
4 x Samtec QSE-040-01-F-D-A-K-TR (2x40pol. Buchse)

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

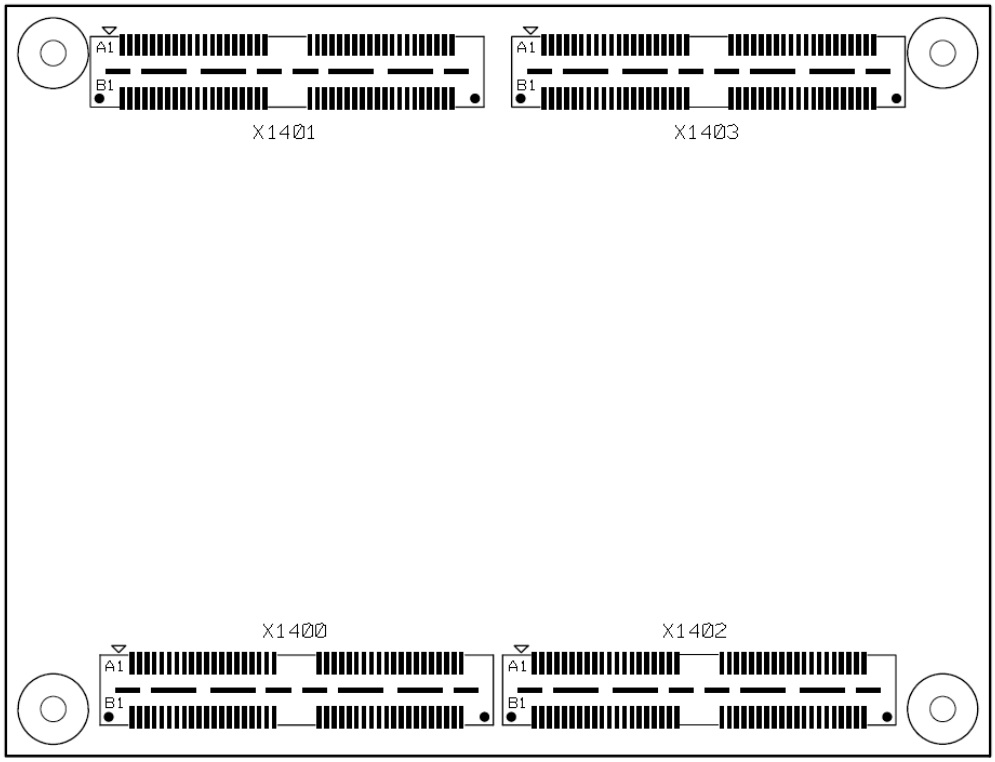


Bild 5: Pinout des PLCcore-E660 - top view

Bild 5 verdeutlicht die Lage der Buchsenleisten (X1400 bis X1403) auf dem PLCcore-E660, Tabelle 5 listet die vollständige Anschlussbelegung des Moduls auf. Eine ausführliche Beschreibung der Modulanschlüsse beinhaltet das Hardware Manual ECUcore-E660 (Manual-Nr.: L-1562).

Tabelle 5: Anschlussbelegung des PLCcore-E660, vollständig, sortiert nach Anschluss-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		

Tabelle 6 beinhaltet als Untermenge von Tabelle 5 nur alle Ein- und Ausgänge des PLCcore-E660, sortiert nach deren Funktion.

Tabelle 6: Anschlussbelegung des PLCcore-E660, nur I/O, sortiert nach Funktion

Name	Funktion
SDC_BOOTx	Zur Auswahl des Boot-Modes SDC (System Diagnostic Controller)
/MR	Eingang für manuellen Reset des Moduls
/CPU_RESET	Reset-Eingangssignal für die CPU
SATAx_TX+, SATAx_TX-, SATAx_RX+, SATAx_RX-, SATAx_LED	SATA 0, 1 mit Busy-LED
ATOM_WD	Watchdog-Eingang für das gesamte Core-Modul
LPC_xyz	LPC-Interface Signale . (weitere Informationen im Intel® AtomE6xx Series Datasheet)
HDA_xyz	Konnektoren für Intel® HD Audio-Interfaces. (weitere Informationen im Intel® AtomE6xx Series Datasheet)
USB_Hx_POW_EN	Power-Enable-Signal für USB Host (0 bis 5) Interfaces
/USB_Hx_ÖVC	USB Host (0 bis 5) Überstrom-Indikation
USB_Hx_+, USB_Hx_-	USB Host (0 bis 5) differenzielle Bus-Signale
PCIE_TXx_+, PCIE_TXx_-, PCIE_RXx_+, PCIE_RXx_-	PCIExpress (2 und 3) differenzielle Bus-Signale
USER_PCIE_CLK_+, USER_PCIE_CLK_-	PCIExpress differenzielles Clock-Signal
SMB_CLK_EXT, SMB_DATA_EXT, /SMB_ALERT_EXT	SMBus Interface-Signale
UARTx_TX, UARTx_RX,	UART 0,1,2,3
UART0_CTS, UART0_DCD, UART0_DSR, UART0_DTR, UART0_RI, UART0_RTS	UART 0 Handshake-Signale
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 . (weitere Informationen im Intel® AtomE6xx Series Datasheet)
SDC_GPIO_x	GPIOs, die vom STM32 System Diagnostic Controller (SDC) unterstützt werden
A_GPIO_SUS_x, A_GPIO_x	GPIOs, die vom Intel Atom-E660T-Prozessor unterstützt werden
H_GPIO_x	GPIOs, die vom Intel EG20T Hub-IC unterstützt werden
NC_SI_xyz	Pins für das Network Controller Sideband Interface
SDC_USART1_RX, SDC_USART1_TX	UART des SDC
SPIx_CLK, SPIx_MISO, SPIx_MOSI	SPI 1,2 Clock- und Daten-Signale
SROM_DIN, SROM_DOUT, /SROM_CS, SROM_CLK	SROM-Interface des EG20T Hub
VBAT	Backup-Batterie-Eingang (3,3V) für den RTC
ETHx_TX-, TX+, RX-, RX+; LEDx	Ethernet-Interfaces mit LEDs
SPEAKER	PWM getriebener Beeper-Ausgang
HUB_TCK,HUB_TMS, HUB_TDI, HUB_TDO, /HUB_TRST, HUB_RTCK	JTAG-Interface für den EG20T Hub
ETH_TCK, ETH_TMS, ETH_TDI, ETH_TDO	JTAG-Interface für den Intel Ethernet Controller
IO_TCK, IO_TMS, IO_TDI, IO_TDO, /IO_TRST	IO JTAG des Atom E660T
NC_TCK, NC_TMS, NC_TDI, NC_TDO	JTAG für den internen Netzwerk-Controller des Atom E660T
CPU_TCK, CPU_TMS, CPU_TDI, CPU_TDO, CPU_TRST_B	JTAG der CPU Atom E660T
SDC_TCK, SDC_TMS, SDC_TDI, SDC_TDO, SDC_NJTRST	JTAG für den System Diagnostic Controller (SDC)
DS_TCK, DS_TMS, DS_TDI, DS_TDO	JTAG des Reset -/ Power-Controllers
+3V3	3,3V-Stromversorgung (ca. 820mA)
GND	Signal-Ground

Tabelle 7 definiert die Codierung des Run/Stop-Schalters. Die Funktion des Run/Stop-Schalters für die SPS-Firmware erläutert Abschnitt 6.7.1. Ist für den Einsatz des PLCcore-E660 auf einer anwenderspezifischen Basisplatine kein Run/Stop-Schalter vorgesehen, muss an den Modulanschlüssen die Codierung für *"Run"* fest verdrahtet sein.

Tabelle 7: Codierung des Run/Stop-Schalters

Modus	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 SPS-Funktionalität des PLCcore-E660

6.1 Übersicht

Das PLCcore-E660 realisiert eine vollständige, Linux-basierte Kompakt-SPS in Form eines Aufsteckmoduls ("Core"). Das PLCcore-E660 basiert dabei auf der Hardware des ECUcore-E660 und erweitert dieses um SPS-spezifische Funktionalitäten (SPS-Firmware, Target-Visualisierung). Beide Module, sowohl ECUcore-E660 als auch PLCcore-E660, benutzt dasselbe Embedded Linux als Betriebssystem. Folglich sind auch Konfiguration und C/C++ Programmierung des PLCcore-E660 weitestgehend identisch zum ECUcore-E660.

6.2 Systemstart des PLCcore-E660

6.2.1 Boot-Up-Sequenz

Prinzipiell besteht die Boot-Up-Sequenz aus 3 Schritten:

Schritt 1

Nachdem der PLCcore den Reset-Status verlassen hat (Bedingungen – siehe Abschnitt 6.2.2 unten) fährt der UEFI-Bootloader hoch. Er startet mit einem Speichertest wie in Bild 6 gezeigt. Dieser Speichertest kann manuell abgebrochen werden durch Eingabe von <ESC> über die Konsole mittels Terminalverbindung an UART1.

Bild 6: Speichertest während des Boot-Up

Schritt 2

Sobald der Speichertest beendet ist, wartet der UEFI-Bootloader 2 Sekunden auf die Eingabe eines SPACE-Zeichens. Wird ein SPACE-Zeichen empfangen über UART1, zeigt der UEFI-Bootloader ein Boot-Menü. Die weitere Prozedur ist in Abschnitt 0 beschrieben.

Ohne Abbruch des Speichertests oder nach Verlassen des UEFI-Boot-Menüs durch ein Boot-Kommando (siehe Abschnitt 0) wird das Betriebssystem Linux gebootet in der Boot-Reihenfolge, wie sie für den UEFI-Bootloader konfiguriert ist (siehe Abschnitt 6.3.1.2). Die Boot-Up-Sequenz wird dann fortgesetzt mit Schritt 3.

Schritt 3

Sobald der Boot-Vorgang des Betriebssystems Linux beendet ist, erhält der User einen Login-Prompt (siehe Abschnitt 7.8.1).

Ohne jegliche Eingaben an der seriellen Konsole kommt die Boot-Up-Sequenz automatisch zum Ende von Schritt 3, wie es in den UEFI-Settings konfiguriert ist.

6.2.2 Reset-Bedingungen des PLCcore-E660 Development-Boards

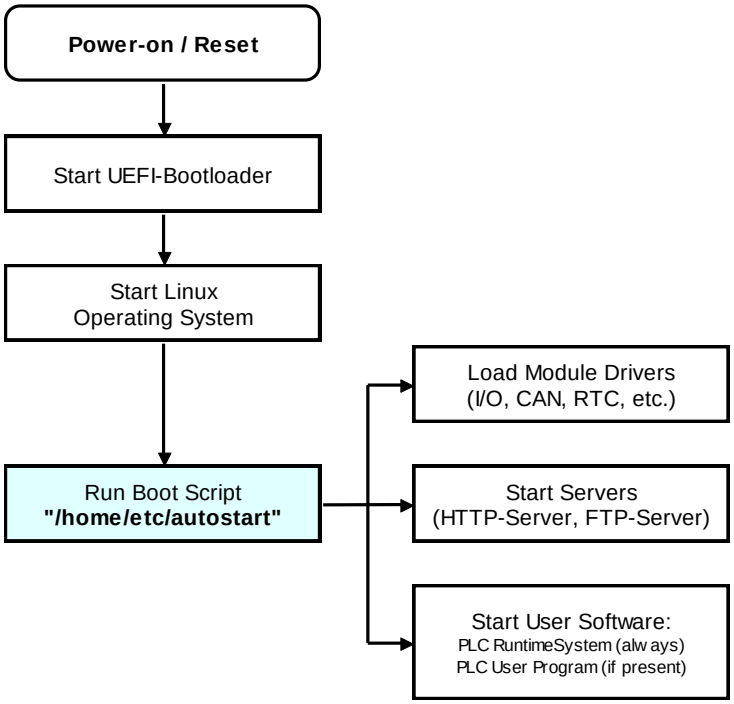
Es gibt prinzipiell 4 Möglichkeiten, einen Reset/Reboot am PLCcore-E660 Development-Board auszulösen. Sie sind in Tabelle 8 beschrieben.

Tabelle 8: Reset-Management

Reset-Quelle	Erklärung
Taster C RES gedrückt für mehr als 4 Sekunden	Dieser Taster wirkt als "Power Off". Die Power-Down-Sequenz des System Diagnostic Controllers (SDC) wird gestartet, das System fährt herunter und wird dann abgeschaltet. In der gegenwärtigen Version startet das System nach 10 Sekunden neu und führt einen Kaltstart aus.
Taster W RES gedrückt	Das System startet neu mit einem Warmstart. Der SDC initiiert keine Power-Down-Sequenz
Kommando "reboot" über die Konsole	Das System startet neu. Der Reset-Grund wird behandelt wie bei einem Kaltstart.
Ausschalten der Stromversorgung	Das System hält sofort an. Nach dem nächsten Einschalten startet das System neu mit einem Kaltstart.

6.2.3 Autostart der Nutzer-Software

Standardmäßig lädt das PLCcore-E660 nach Power-on bzw. Reset alle notwendigen Firmware-Komponenten und startet anschließend die Abarbeitung des SPS-Programms. Damit eignet sich das PLCcore-E660 für den Einsatz in autarken Steuerungssystemen, die nach einer Spannungsunterbrechung selbständig und ohne Benutzeraktionen die Ausführung des SPS-Programms wieder aufnehmen. Bild 7 zeigt den Systemstart im Detail.



Bezüglich Details zur Deaktivierung des autarken Linux-Starts und zur Aktivierung des “UEFI-Shell" Kommando-Prompts siehe Abschnitt 6.2.1.

Details über die Programmierung des PLCcore-E660 sind enthalten in Abschnitt 6.4.

Bild 7: Systemstart des PLCcore-E660

Es ist möglich, den PLCcore-E660 so zu konfigurieren, dass die Nutzer-Software nach einem Reset automatisch startet. Dazu müssen alle wesentlichen Kommandos im Startskript **"/home/etc/autostart"** enthalten sein. Wenn nötig, können alle benötigten Umgebungsvariablen gesetzt und ebenso die benötigten Treiber geladen werden.

Der Startskript **"/home/etc/autostart"** muss entsprechend der angestrebten Funktionalität angepasst werden. Zum Beispiel ist es durch Hinzufügen des Kommandos *"pureftp"* zu diesem Skript möglich, dass der FTP-Server automatisch beim Hochfahren des PLCcore-E660 gestartet wird. Der Skript kann direkt im PLCcore-E660 editiert werden, indem im FTP-Client *"WinSCP"* (siehe Abschnitt 7.8.2) entweder die Funktionstaste *"F4"* oder der Pushbutton *"F4 Edit"* betätigt werden.

6.3 UEFI Bootloader and EFI Shell

6.3.1 UEFI-basierter Bootloader für den PLCcore-E660

6.3.1.1 Features

- Basierend auf Intel Bootloader Development Kit phase II
- ACPI support einschließlich CPU Frequenz-Skalierung
- Boot-Medien:
    - SD card, eMMC

  - USB Hard Disk oder Stick
  - SATA Disk
- Partition-Layouts auf den Boot-Medien:
  - GUID Partition Table (GPT)
  - Master Boot Record (MBR)
- Unterstützte Filesysteme auf den Boot-Medien
  - FAT16, FAT32: Lesen und Schreiben
  - ext2: Read Only mit eingeschränkter Unterstützung für ext3; Der EFI-Treiber ist lizenziert unter GPL
- Integrierte vollständige UEFI-Shell (siehe http://www.intel.com/intelpress/sum_eshl.htm)

Bitte beachten Sie http://www.intel.com/intelpress/sum_efi2.htm oder andere Instruktionen zu UEFI-kompatibler Firmware.

6.3.1.2 Default-Boot-Reihenfolge

Wenn keine Boot-Optionen existieren oder das Booten aller existierenden Boot-Optionen fehlschlägt, versucht der Bootloader \EFI\BOOT\BOOTIA32.EFI von irgendeinem Medium in der folgenden Reihenfolge auszuführen.

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

Anmerkung:
Es wird aus zweierlei Gründen empfohlen, immer eine Boot-Option für das ausgewählte Boot-Medium zu kreieren. Diese Gründe sind:

1. Sicherstellen, dass das konfigurierte Boot-Medium immer benutzt wird, ungeachtet dessen, ob irgendein anderes Boot-Medium (z. B. ein USB-Stick) angeschlossen ist

2. Erhöhen der Boot-Geschwindigkeit – bei der Default-Boot-Reihenfolge initialisiert der Bootloader im ungünstigsten Fall alle Boot-Medien, bei Vorhandensein einer Boot-Option jedoch nur das in der Boot-Option konfigurierte Medium.

6.3.1.3 Kreieren einer Boot-Option

Bitte beachten Sie die betreffenden Abschnitte im Dokument L-1554.

6.3.1.4 Auslesen der System-Identifikation

Bitte beachten Sie die betreffenden Abschnitte im Dokument L-1554.

6.3.2 Aktivierung der EFI-Shell

Standardmäßig, startet der UEFI-Bootloader nach Reset oder Power On das Betriebssystem Linux des Moduls automatisch. Danach lädt das Betriebssystem alle weiteren Software-Komponenten. Für Servicezwecke wie z. B. die Ethernet-Konfiguration (siehe Abschnitt 7.3) ist es notwendig, die EFI-Shell des UEFI-Bootloaders am seriellen Interface UART1 des PLCcore-E660 zu starten.

Das automatische Booten des Betriebssystems Linux kann unterbrochen werden durch Drücken der SPACE-Taste, wenn der UEFI-Bootloader die Meldung “Press SPACE in 2 seconds to list all boot options” am seriellen Interface UART1 ausgibt. Dann gibt der Bootloader das Boot-Menü aus (siehe *Bild 8*). Durch Drücken der Taste ‘3’ wird die EFI-Shell gestartet (siehe *Bild 9*).

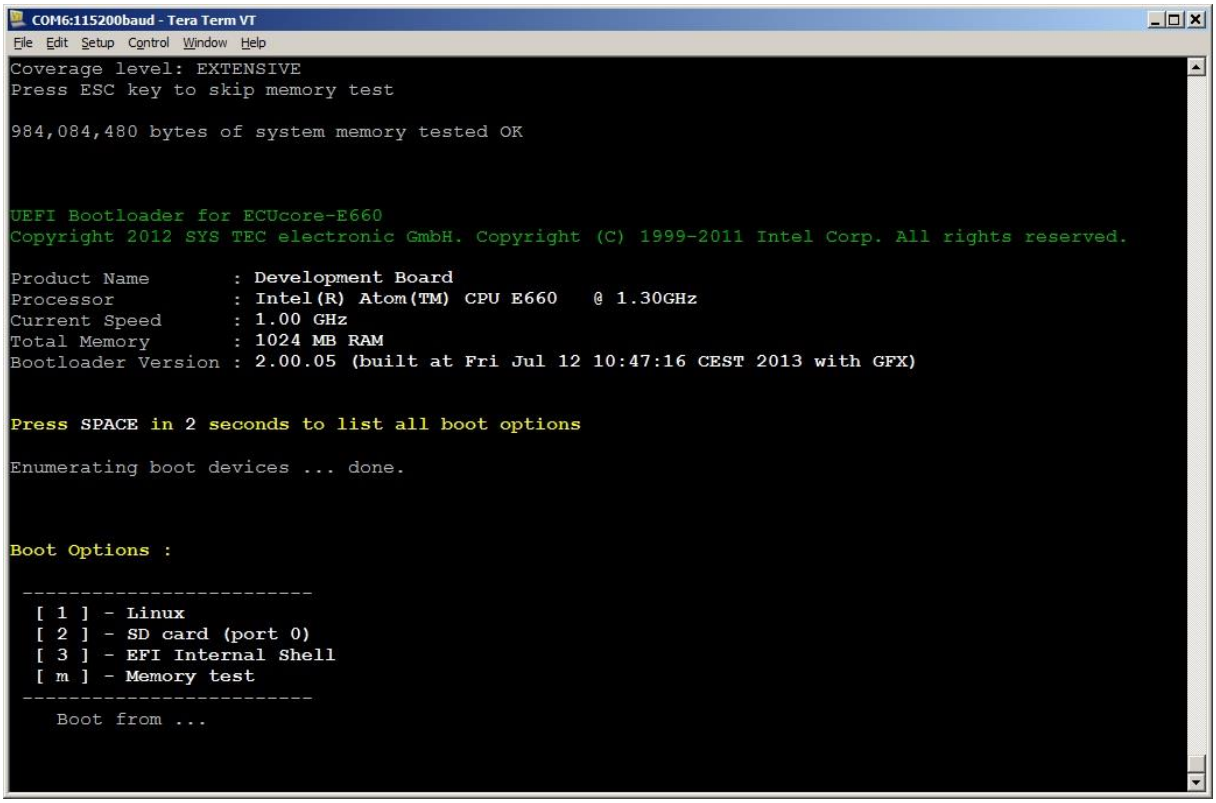


Bild 8: UEFI-Boot-Menü

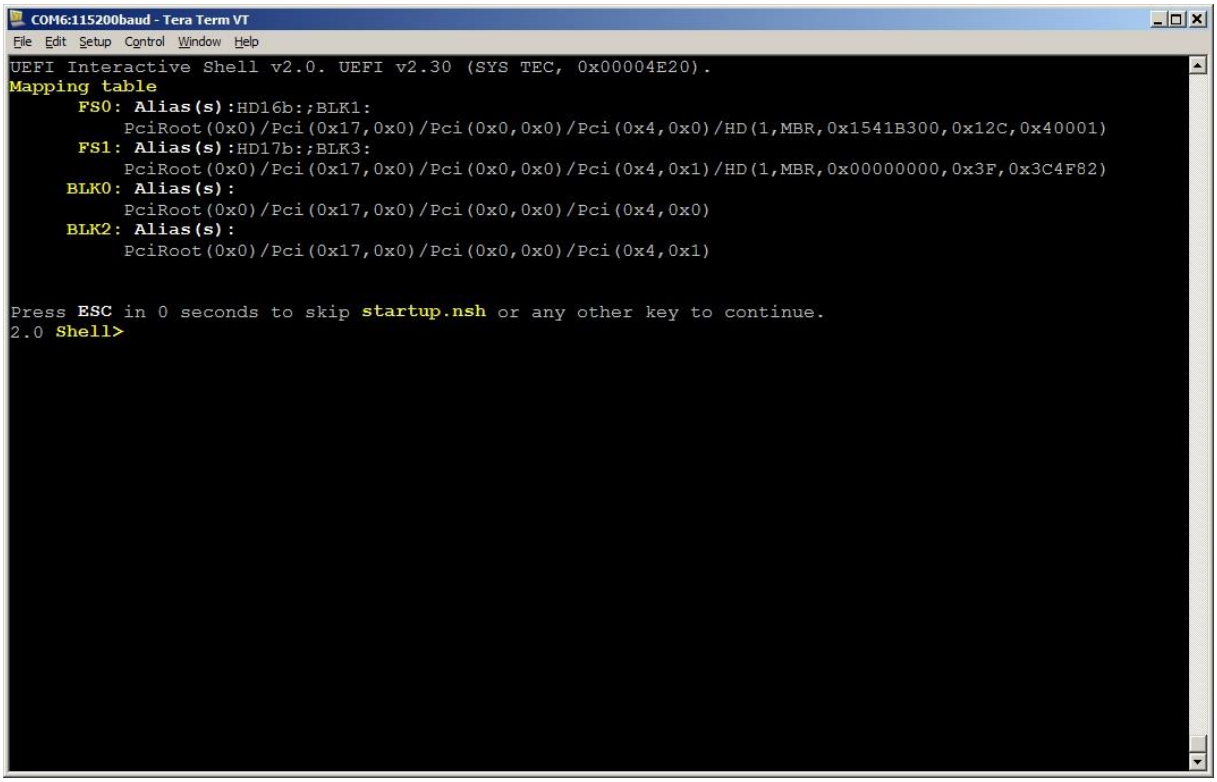


Bild 9: EFI-Shell gestartet

Sobald die EFI-Shell aktiv ist, können die in Tabelle 9 aufgeführten Kommandos ausgeführt werden.

Tabelle 9: EFI-Kommandos

EFI-Kommando	Mit dem Kommando ausgeführte Aktion
attrib	Attribute von Dateien oder Verzeichnissen anzeigen oder ändern
cd	Aktuelles Verzeichnis anzeigen oder ändern
cp	Kopieren einer oder mehrerer Dateien oder Verzeichnisse in eine Zieldatei bzw. in ein Zielverzeichnis
load	Laden eines UEFI-Treibers in den Speicher
map	Ein Mapping zwischen einem benutzerdefinierten Namen und einem Device-Handle
mkdir	Anlegen eines oder mehrerer Verzeichnisse
mv	Verschieben einer oder mehrerer Dateien auf ein Zielverzeichnis innerhalb des Dateisystems
parse	Auffinden des Wertes eines bestimmten Records in einer standardformatierten Ausgabe
reset	Reset des Systems
set	UEFI-Shell-Umgebungsvariablen anzeigen, verändern oder löschen
ls	Verzeichnisinhalte oder Dateiinformationen anzeigen
rm	Löschen eines oder mehrerer Dateien oder Verzeichnisse
vol	Anzeigen der Volume-Informationen des Dateisystems, das durch ***fs*** spezifiziert ist
date	Anzeigen oder Setzen des aktuellen Systemdatums
time	Anzeigen oder Setzen der aktuellen Systemzeit

EFI-Kommando	Mit dem Kommando ausgeführte Aktion
timezone	Zeitzonen-Informationen anzeigen oder setzen
stall	Operation anhalten für eine spezifizierte Anzahl Mikrosekunden
for	Schleife starten in ***for***-Syntax
goto	Sprung zu einem spezifizierten Ausführungspunkt eines Skripts
if	Steuern, welche Skript-Kommandos ausgeführt werden in Abhängigkeit vom Bedingungs-Ausdruck
shift	Schieben der Skript-Aufrufparameter um 1 Stelle (ermöglicht Zugriff auf mehr als 10 Skript-Parameter)
exit	Austritt aus der EFI-Shell oder aus dem aktuellen Skript
else	Der ***else***-Block bezeichnet die Menge des auszuführenden Codes falls der Bedingungs-Ausdruck für ***if*** FALSE ist.
endif	Ende des Blocks einer ***if***-Anweisung
endfor	Ende einer ***for***-Schleife
type	Ausgeben des Inhalts einer Datei zur Standardausgabe
touch	Zeit- und Datumsinformationen einer Datei auf die aktuelle Zeit und das aktuelle Datum setzen
ver	Anzeige der Versions-Informationen von UEFI-Shell darunter liegender UEFI-Firmware
alias	Alias-Namen der UEFI-Shell-Umgebung anzeigen, kreieren oder löschen
cls	Standardausgabe löschen und optional die Hintergrundfarbe ändern
echo	Steuern, ob Skript-Kommandos angezeigt werden, wenn sie vom Skript gelesen werden, bzw. Ausgabe einer Nachricht auf dem Display
pause	Skript pausieren lassen bis zu einer Tastatureingabe durch den Bediener
help	Anzeige aller in der UEFI-Shell enthaltenen Kommandos
connect	Binden eines Treibers an ein spezielles Device und Starten des Treibers.
devices	Anzeigen der Liste der von UEFI-Treibern verwalteten Devices
openinfo	Anzeigen der mit einem Handle verbundenen Protokolle und Agents
disconnect	Trennen eines oder mehrerer Treiber vom spezifizierten Device
reconnect	Wieder Verbinden von Treibern mit dem spezifizierten Device
unload	Entladen eines bereits geladenen Treiber-Images
drvdiag	Aufruf des Treiber-Diagnoseprotokolls
dh	Anzeigen der Device-Handles der UEFI-Umgebung
drivers	Anzeigen einer Liste von Informationen für Treiber, die dem UEFI-Treibermodell in der UEFI-Umgebung entsprechen
devtree	Anzeigen des Baums UEFI-Treibermodell-konformer Devices
drvcfg	Konfigurieren der der Plattform unterlegten Konfigurations-Infrastruktur
bcfg	Verwalten der im NVRAM gespeicherten Boot- und Treiber-Optionen
setsize	Anpassen der Größe einer Datei
comp	Byteweiser Vergleich zweier Dateien
mode	Output-Device-Mode der Konsole anzeigen oder ändern
memmap	Anzeigen der von der UEFI-Umgebung unterstützten Memory-Map
eficompress	Datei mit dem EFI-Kompressionsalgorithmus komprimieren

EFI-Kommando	Mit dem Kommando ausgeführte Aktion
efidecompress	Datei mit dem EFI-Dekompressionsalgorithmus dekomprimieren
dmem	Inhalt von System- oder Device-Speicher anzeigen
loadpcirom	Laden eines UEFI-Treibers aus einer Datei im Format "PCI Option ROM"
mm	Anzeigen oder Modifizieren des MEM/MMIO/IO/PCI/PCIE-Adressraums
setvar	Wert einer UEFI-Variablen ändern
sermode	Setzen von Attributen serieller Ports
pci	Anzeigen der PCI-Device-Liste oder des PCI function configuration space
smbiosview	Anzeigen der SMBIOS-Informationen
dmpstore	Verwaltung aller UEFI-NVRAM-Variablen
dblk	Anzeigen des Inhalts eines oder mehrerer Blöcke eines Block-Devices
edit	Full-Screen-Editor für ASCII- oder UCS-2-Dateien
hexedit	Full-Screen-Hex-Editor für Dateien, Block-Devices oder Speicher

Ein Hilfetext zur Benutzung der EFI-internen Kommandos in Tabelle 9 wird angezeigt, wenn das Kommando wie folgt gegeben wird:

"<EFI command> -? -v"

In Ergänzung zu den EFI-internen Kommandos in Tabelle 9 kann die EFI-Shell externe Kommandos ausführen (Skripte von SYS TEC). Diese befinden sich im Verzeichnis *"/efi/tools/".* Folgende externe Kommandos sind verfügbar:

- ***setenv*** – Kommando zum Setzen von Umgebungsvariablen
- ***printenv*** – Kommando zur Verifikation der Umgebungsvariablen

Ein Hilfetext zur Benutzung der externen Kommandos wird angezeigt, wenn die Kommandos ohne Parameter gegeben werden.

6.3.3 Anforderungen und Einstellungen für die Kommunikation mit der EFI-Shell

Die Kommunikation mit dem UEFI-Bootloader erfolgt nur über das serielle Interface UART1 (bzgl. Konfiguration des seriellen Ports siehe Abschnitt 7.2).

6.4 Programmierung des PLCcore-E660

Das PLCcore-E660 wird mit der IEC 61131-3 konformen Programmierumgebung *OpenPCS* programmiert. Zu *OpenPCS* existieren eigene Handbücher, die den Umgang mit diesem Programmierwerkzeug beschreiben. Diese sind Bestandteil des Software-Paketes *"OpenPCS"*. Tabelle 1 listet die für das PLCcore-E660 relevanten Manuals auf.

Die Firmware des PLCcore-E660 basiert auf der Standardfirmware für SYS TEC-Kompaktsteuerungen und hat daher ein identisches Verhalten zu anderen Steuerungen der Firma SYS TEC. Dies betrifft insbesondere den Aufbau des Prozessabbildes (siehe Abschnitt 6.5) sowie die Funktionsweise der Bedienelemente (Hexcodier-Schalter, DIP-Schalter, Run/Stop-Schalter, Run-LED, Error-LED).

Die Firmware des PLCcore-E660 stellt dem SPS-Programmierer in Abhängigkeit der eingesetzten Firmware-Variante zahlreiche Funktionsbausteine für den Zugriff auf Kommunikationsschnittstellen bereit. Tabelle 10 listet die Verfügbarkeit der Kommunikations-FB-Klassen (SIO, CAN, UDP) für die einzelnen Firmware-Typen des PLCcore-E660 auf. Abschnitt 7.6 beschreibt die Auswahl der jeweils zu startenden Firmware-Variante.

Tabelle 10: Unterstützung von Kommunikations-FB-Klassen für verschiedene PLCcore Typen

Interface Typ	**PLCcore-E660/Z3 Art.-Nr.: 3390089/Z3 3390090/Z3**	**PLCcore-E660/Z4 Art.-Nr.: 3390089/Z4 3390090/Z4**	**PLCcore-E660/Z5 Art.-Nr.: 3390089/Z5 3390090/Z5**	**Bemerkung**
CAN	-	x	x	FB-Beschreibung siehe Manual L-1008
UDP	-	x	x	FB-Beschreibung siehe Manual L-1054
SIO	X (nur 3390090)	x (nur 3390090)	x (nur 3390090)	FB-Beschreibung siehe Manual L-1054

Eine vollständige Auflistung der vom PLCcore-E660 unterstützten Firmware-Funktionen und -Funktionsbausteine enthält Tabelle 20 im Anhang A.

Detaillierte Informationen zu Nutzung der CAN-Schnittstellen in Verbindung mit CANopen beschreibt Abschnitt 6.9.

6.5 Prozessabbild des PLCcore-E660

6.5.1 Lokale Ein- und Ausgänge

Das PLCcore-E660 besitzt ein Prozessabbild mit identischen Adressen im Vergleich zu anderen SYS TEC-Kompaktsteuerungen. Vom PLCcore-E660 werden die in Tabelle 11 aufgelisteten Ein- und Ausgänge unterstützt.

Tabelle 11: Zuordnung der Ein- und Ausgänge zum Prozessabbild des PLCcore-E660

I/O des PLCcore-E660	Adresse und Datenformat im Prozessabbild	
DI0 ... DI7	**%IB0.0** %IX0.0 … %IX0.7	als Byte mit DI0 … DI7 als einzelne Bits für jeden Eingang
DI8 ... DI15	**%IB1.0** %IX1.0 … %IX1.7	als Byte mit DI8 … DI15 als einzelne Bits für jeden Eingang
DI16 … DI23 (nur als anwenderspezifische Erweiterung)	**%IB2.0** %IX2.0 … %IX2.7	als Byte mit DI16 … DI23 als einzelne Bits für jeden Eingang
DI24 ... DI31 (nur als anwenderspezifische Erweiterung)	**%IB3.0** %IX3.0 … %IX3.7	als Byte mit DI24 … DI31 als einzelne Bits für jeden Eingang
DI32 ... DI39 (nur als anwenderspezifische Erweiterung)	**%IB4.0** %IX4.0 … %IX4.7	als Byte mit DI32 … DI139 als einzelne Bits für jeden Eingang
DI40 … DI47 (nur als anwenderspezifische Erweiterung)	**%IB5.0** %IX5.0 … %IX5.7	als Byte mit DI40 … DI47 als einzelne Bits für jeden Eingang
On-board Temperatursensor, siehe (1)	**%ID72.0**	31Bit + Vorzeichen als 1/10000 °C
DO0 ... DO7	**%QB0.0** %QX0.0 … %QX0.7	als Byte mit DO0 … DO7 als einzelne Bits für jeden Ausgang
DO8 … DO15 (nur als anwenderspezifische Erweiterung)	**%QB1.0** %QX1.0 … %QX1.7	als Byte mit DO8 … DO15 als einzelne Bits für jeden Ausgang
DO16 … DO23 (nur als anwenderspezifische Erweiterung)	**%QB2.0** %QX2.0 … %QX2.7	als Byte mit DO16 … DO23 als einzelne Bits für jeden Ausgang
DO24 … DO31 (nur als anwenderspezifische Erweiterung)	**%QB3.0** %QX3.0 … %QX3.7	als Byte mit DO24 … DO31 als einzelne Bits für jeden Ausgang
DO32 … DO39 (nur als anwenderspezifische Erweiterung)	**%QB4.0** %QX4.0 … %QX4.7	als Byte mit DO32 … DO39 als einzelne Bits für jeden Ausgang
DO40 … DO47 (nur als anwenderspezifische Erweiterung)	**%QB5.0** %QX5.0 … %QX5.7	als Byte mit DO40 … DO47 als einzelne Bits für jeden Ausgang

(1) Die entsprechend gekennzeichneten Komponenten sind nur im Prozessabbild verfügbar, wenn die Option ***"Enable extended I/O's"*** in der SPS-Konfiguration aktiviert ist (siehe Abschnitt 7.4.1). Alternativ kann auch der Eintrag *"EnableExtIo="* in der Sektion *"[ProcImg]"* innerhalb der

Konfigurationsdatei ***"/home/plc/plccore-E660.cfg"*** direkt gesetzt werden (siehe Abschnitt 7.4.2). Die entsprechende Konfigurationseinstellung wird beim Starten der SPS-Firmware ausgewertet.

Die Ein- und Ausgänge des PLCcore-E660 werden nicht negiert im Prozessabbild verwaltet, d.h. ein H-Pegel an einem Eingang führt zum Wert "1" an der entsprechenden Adresse im Prozessabbild und umgekehrt führt der Wert "1" im Prozessabbild zu einem H-Pegel am zugehörigen Ausgang.

6.5.2 Ein- und Ausgänge anwenderspezifischer Baseboards

Die nach außen geführten Anschlussleitungen des Moduls bietet dem Anwender größtmögliche Freiheitsgrade bei der Gestaltung der Ein-/Ausgangsbeschaltung des PLCcore-E660. Damit können sämtliche Ein- und Ausgänge des PLCcore-E660 flexibel an die jeweiligen Erfordernisse angepasst werden. Dies wiederum hat zur Folge, dass das Prozessabbild des PLCcore-E660 maßgeblich von der konkreten Realisierung der anwenderspezifischen Außenbeschaltung definiert wird. Die softwareseitige Einbindung der Ein-/Ausgangskomponenten in das Prozessabbild erfordert das "*Driver Development Kit für das ECUcore-E660*" (Bestellnummer SO-1117).

6.6 Kommunikationsschnittstellen

6.6.1 Serielle Schnittstellen

Das PLCcore-E660 besitzt 4 serielle Schnittstellen (UART0 … UART3), die als RS-232 ausgeführt sind. Die Schnittstelle UART3 kann alternativ auch im RS-485 Modus betrieben werden. Details zur Hardwareanschaltung beschreibt das *"Hardware Manual Development Board ECUcore-E660"* (Manual-Nr.: L-1562).

UART0: Die Schnittstelle UART0 ist in der Regel nicht nutzbar. (siehe hierzu Tabelle 2)

UART1: Die Schnittstelle UART1 dient primär als Serviceschnittstelle zur Administration des PLCcore-E660. Sie wird standardmäßig im Boot-Skript *"/etc/inittab"* dem Linux-Prozess *"getty"* zugeordnet und als Linux-Konsole zur Administration des PLCcore-E660 benutzt. Die Schnittstelle COM0 ist zwar prinzipiell aus einem SPS-Programm über die Funktionsbausteine vom Typ *"SIO_Xxx"* nutzbar (siehe Manual *"SYS TEC spezifische Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L-1054), allerdings sollten hier nur Zeichen ausgegeben werden. Empfangene Zeichen versucht das Modul stets als Linux-Kommandos zu interpretieren und auszuführen.

Um die Schnittstelle aus einem SPS-Programm frei verwenden zu können, ist das Boot-Skript *"/etc/inittab"* entsprechend anzupassen, was nur durch eine Modifikation des Linux-Images möglich ist. Voraussetzung hierfür ist das Softwarepaket SO-1116 ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-E660").

UART2: Die Schnittstelle UART2 ist einem SPS-Programm über die Funktionsbausteine vom Typ *"SIO_Xxx"* nutzbar (siehe Manual *"SYS TEC spezifische Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L-1054).

Um die Schnittstelle aus einem SPS-Programm frei verwenden zu können, ist das Boot-Skript *"/etc/inittab"* entsprechend anzupassen, was nur durch eine Modifikation des Linux-Images möglich ist. Voraussetzung hierfür ist das Softwarepaket SO-1116 ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-E660").

UART3: Die Schnittstelle UART3 ist aus einem SPS-Programm über die Funktionsbausteine vom Typ *"SIO_Xxx"* nutzbar (siehe Manual *"SYS TEC spezifische Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L-1054).

Um die Schnittstelle aus einem SPS-Programm frei verwenden zu können, ist das Boot-Skript *"/etc/inittab"* entsprechend anzupassen, was nur durch eine Modifikation des Linux-Images möglich ist. Voraussetzung hierfür ist das Softwarepaket SO-1116 ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-E660").

6.6.2 CAN-Schnittstellen

Das PLCcore-E660 verwaltet 2 CAN-Schnittstellen (CAN0, CAN1). Details zur Hardwareanschaltung beschreibt das *"Hardware Manual Development Board ECUcore-E660"* (Manual-Nr.: L-1562).

Die CAN-Schnittstelle CAN0 ermöglicht den Datenaustausch mit anderen Geräten über Netzwerkvariablen und ist zudem aus einem SPS-Programm über die Funktionsbausteine vom Typ *"CAN_Xxx"* nutzbar (siehe Abschnitt 6.9 sowie *"User Manual CANopen-Erweiterung für IEC 61131-3"*, Manual-Nr.: L-1008).

Detaillierte Informationen zu Nutzung der CAN-Schnittstellen in Verbindung mit CANopen beschreibt Abschnitt 6.9.

6.6.3 Ethernet-Schnittstellen

Das PLCcore-E660 besitzt 2 Ethernet-Schnittstellen (ETH0, ETH1). Details zur Hardwareanschaltung beschreibt das *"Hardware Manual Development Board ECUcore-E660"* (Manual-Nr.: L-1562).

Die Ethernet-Schnittstelle ETH0 dient sowohl als Serviceschnittstelle zur Administration des PLCcore-E660 als auch zum Datenaustausch mit beliebigen anderen Geräten. Aus einem SPS-Programm ist die Schnittstelle über Funktionsbausteine vom Typ *"LAN_Xxx"* nutzbar (siehe Manual *"SYS TEC spezifische Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L-1054).

Das SPS-Programmbeispiel *"UdpRemoteCtrl"* verdeutlicht die Nutzung der Funktionsbausteine vom Typ *"LAN_Xxx"* innerhalb eines SPS-Programms.

Die Schnittstelle ETH1 wird im PLCcore-E660 mit OpenPCS nicht benutzt.

6.7 Bedien- und Anzeigeelemente

6.7.1 Run/Stop-Schalter

Die Modulanschlüsse *"X1403/A10"*, *" X1403/A11"* und *" X1403/A18"* (siehe Tabelle 7) sind für die Anbindung eines Run/Stop-Schalters vorgesehen. Mit Hilfe dieses Run/Stop-Schalters ist es möglich, die Abarbeitung eines SPS-Programms zu starten und zu unterbrechen. Der Run/Stop-Schalter bildet mit den Start- und Stop-Schaltflächen der *OpenPCS*-Programmierumgebung eine "logische" UND-Verknüpfung. Das bedeutet, dass die SPS erst dann mit der Programm-Abarbeitung beginnt, wenn sich der lokale Run/Stop-Schalter in Stellung *"Run"* befindet **UND** außerdem ein Startkommando (Kalt-, Warm- oder Heißstart) durch die *OpenPCS*-Oberfläche erteilt wurde. Die Reihenfolge ist dabei irrelevant. Ein von *OpenPCS* veranlasster Run-Befehl bei gleichzeitiger Stellung des Run/Stop-Schalters in Stellung *"Stop"* ist an dem kurzen Aufblinken der Run-LED (grün) erkennbar.

In der Position *"MRes"* (*"Modul Reset"*) ermöglicht der Run/Stop-Schalter das lokale Löschen eines auf dem PLCcore-E660 befindlichen SPS-Programms. Dies ist beispielsweise dann notwendig, wenn sich das SPS-Programm durch einen Programmierfehler in einer Endlosschleife "verklemmt" und ein

Zugriff von der *OpenPCS*-Programmierumgebung dadurch nicht mehr möglich ist. Die zum Löschen des SPS-Programms notwendige Vorgehensweise beschreibt Abschnitt 6.8.

6.7.2 Run-LED (grün)

Die auf dem Development-Board vorhandene Run-LED gibt Auskunft über den Aktivitätszustand der Steuerung, die durch verschiedene Modi dargestellt werden:

Tabelle 12: Anzeigezustände Run-LED

LED-Modus	SPS-Aktivitätszustand
Aus	Die SPS befindet sich im Zustand *"Stop"*: • die SPS besitzt kein gültiges Programm, • die SPS hat ein Stop-Kommando von der *OpenPCS*-Programmierumgebung erhalten oder • die Programm-Abarbeitung wurde aufgrund eines internen Fehlers abgebrochen
Kurzes Aufblinken im Puls-Verhältnis 1:8	Die SPS befindet sich in Bereitschaft, führt jedoch noch keine Verarbeitung aus: • Die SPS hat ein Start-Kommando von der *OpenPCS*-Programmierumgebung erhalten, der lokale Run/Stop-Schalter befindet sich jedoch noch in Stellung *"Stop"*
Langsames Blinken im Puls-Verhältnis 1:1	Die SPS befindet sich im Zustand *"Run"* und arbeitet das SPS-Programm ab
Schnelles Blinken im Puls-Verhältnis 1:1	Die SPS befindet sich im Modus *"Rücksetzen"*, siehe Abschnitt 6.8

6.7.3 Error-LED (rot)

Die auf dem Development-Board vorhandene Error-LED gibt Auskunft über den Fehlerzustand der Steuerung, die durch verschiedene Modi dargestellt werden:

Tabelle 13: Anzeigezustände Error-LED

LED-Modus	SPS-Fehlerzustand
Aus	Es ist kein Fehler aufgetreten, die SPS befindet sich im Normalzustand
Dauerlicht	Es ist ein schwerwiegender Fehler aufgetreten: • Die SPS wurde mit einer ungültigen Konfiguration gestartet (z.B. CAN-Knotenadresse 0x00) und musste beendet werden oder • Bei der Abarbeitung eines Programms trat ein schwerwiegender Fehler auf, der die SPS veranlasste, den Zustand *"Run"* selbständig zu beenden (Division durch Null, ungültiger Array-Zugriff, ...), siehe unten
Langsames Blinken im Puls-Verhältnis 1:1	Bei der Kommunikation mit dem Programmiersystem trat ein Netzwerkfehler auf, ein evtl. gestartetes Programm wird jedoch weiter abgearbeitet. Dieser Fehlerzustand wird von der SPS bei der nächsten erfolgreichen Kommunikation mit dem Programmiersystem selbständig zurückgesetzt.
Schnelles Blinken im Puls-Verhältnis 1:1	Die SPS befindet sich im Modus *"Rücksetzen"*, siehe Abschnitt 6.8
Kurzes Aufblinken im Puls-Verhältnis 1:8	Die SPS befindet sich in Bereitschaft, führt jedoch noch keine Verarbeitung aus: • Die SPS hat ein Start-Kommando von der *OpenPCS*-Programmierumgebung erhalten, der lokale Run/Stop-Schalter befindet sich jedoch noch in Stellung *"Stop"*

Beim Auftreten eines schwerwiegenden Systemfehlers wie z.B. Division durch Null oder einem ungültiger Array-Zugriff geht die Steuerung selbständig vom Zustand *"Run"* in den Zustand *"Stop"* über. Erkennbar ist dies am Dauerlicht der Error-LED (rot). In einem solchen Fall wird die Fehlerursache jedoch von der SPS gespeichert und beim nächsten Anmelden des Programmiersystems zum PC übermittelt und dort angezeigt.

6.8 Lokales Löschen des SPS-Programms

Mit Hilfe der Schalterstellung *"MRes"* (*"Modul Reset"*) des Run/Stop-Schalters (siehe Abschnitt 6.7.1) kann ein auf dem PLCcore-E660 befindliches Programm gelöscht werden. Dies ist beispielsweise dann notwendig, wenn sich das SPS-Programm durch einen Programmierfehler in einer Endlosschleife "verklemmt" und ein Zugriff von der *OpenPCS*-Programmierumgebung dadurch nicht mehr möglich ist. Um ein versehentliches Löschen des SPS-Programms zu vermeiden, ist folgende Bedienreihenfolge einzuhalten:

(1) Run/Stop-Schalter in Position *"MRes"* stellen

(2) Reset am PLCcore-E660 auslösen (durch Reset-Taster des Developmentboards oder kurzzeitige Spannungsunterbrechung)
⇨ die Run-LED (grün) blinkt schnell mit einem Puls-Verhältnis von 1:1

(3) Run/Stop-Schalter in Position *"Run"* stellen
⇨ die Error-LED (rot) blinkt schnell mit einem Puls-Verhältnis von 1:1

(4) Run/Stop-Schalter **innerhalb von 2 Sekunden** wieder zurück in Stellung *"MRes"* bringen
⇨ das PLCcore-E660 löscht sein SPS-Programm
⇨ Run-LED (grün) und Error-LED (rot) blinken wechselseitig

(5) Run/Stop-Schalter wieder in zurück in Position *"Stop"* oder *"Run"* bringen und einen erneuten Reset auslösen, um das PLCcore-E660 im normalen Arbeitsmodus zu starten

Wird am PLCcore-E660 ein Reset ausgelöst (z.B. durch kurzzeitige Spannungsunterbrechung), während sich der Run/Stop-Schalter in Position *"MRes"* befindet, erkennt das Modul eine Rücksetzanforderung. Signalisiert wird dies durch ein schnelles Blinken der Run-LED (grün). Dieser Modus kann jedoch wieder gefahrlos beenden werden. Dazu ist der Run/Stop-Schalter in Stellung *"Run"* oder *"Stop"* zu bringen (jetzt blinkt die Error-LED) und mindestens 2 Sekunden warten. Das PLCcore-E660 bricht nach dieser Zeit den Rücksetzvorgang selbständig ab und startet mit dem zuletzt gespeicherten SPS-Programm im normalen Betriebsmodus.

6.9 Nutzung der CAN-Schnittstellen mit CANopen

Das PLCcore-E660 verwaltet 2 CAN-Schnittstellen
- CAN0 als USB-CAN-Modul aus der sysWORXX-Reihe (SYS TEC-Bestellnummer 3204000)
- CAN1 on-board auf dem Core-Modul – Anschluss P900A Bottom oder X900 am Development-Board

CAN0 ist als CANopen-Manager nutzbar (konform zum CiA Draft Standard 302). Die Konfiguration der Schnittstelle (aktiv/inaktiv, Knotennummer, Bitrate, Master an/aus) beschreibt Abschnitt 7.4.

Die CAN-Schnittstelle ermöglicht den Datenaustausch mit anderen Geräten über Netzwerkvariablen und ist zudem aus einem SPS-Programm über die Funktionsbausteine vom Typ *"CAN_Xxx"* nutzbar. Ausführliche Details hierzu beschreibt das *"User Manual CANopen-Erweiterung für IEC 61131-3"*, Manual-Nr.: L-1008.

CANopen stellt mit den beiden Diensten **PDO** (***P***rocess ***D***ata ***O***bjects) und **SDO** (***S***ervice ***D***ata ***O***bjects) zwei unterschiedliche Mechanismen für den Datenaustausch zwischen den einzelnen Feldbusgeräten zur Verfügung. Die von einem Knoten gesendeten Prozessdaten (**PDO**) stehen als Broadcast gleichzeitig allen interessierten Empfängern zur Verfügung. Durch die Realisierung als quittungslose Broadcast-Nachrichten sind PDOs auf 1 CAN-Telegramm und damit auf maximal 8 Byte Nutzdaten limitiert. Im Gegensatz dazu basieren **SDO**-Transfers auf logischen Punkt-zu-Punkt-Verbindungen ("Peer to Peer") zwischen zwei Knoten und ermöglichen den quittierten Austausch von Datenpaketen, die auch größer als 8 Bytes sein können. Diese Datenpakete werden intern durch eine entsprechende Anzahl quittierter CAN-Telegramme übertragen. Beide Dienste sind sowohl für die Schnittstelle CAN0 als auch CAN1 des PLCcore-E660 nutzbar.

Die SDO-Kommunikation erfolgt grundsätzlich immer über Funktionsbausteine vom Typ *"CAN_SDO_Xxx"* erfolgt (siehe *"User Manual CANopen-Erweiterung für IEC 61131-3"*, Manual-Nr.: L-1008). Für PDOs stehen ebenfalls Funktionsbausteine bereit (*"CAN_PDO_Xxx"*), diese sollten jedoch nur in besonderen Fällen verwendet werden, um auch nicht CANopen-konforme Geräte ansprechen zu können. Für die Anwendung der PDO-Bausteine muss die CANopen-Konfiguration im

Detail bekannt sein, da die Bausteine lediglich 8 Bytes als Übergabeparameter benutzen, die Zuordnung der Bytes zu den Prozessdaten jedoch Aufgabe des Anwenders ist.

Statt PDO-Bausteinen sollten vorrangig Netzwerkvariablen für den PDO-basierten Datenaustausch verwendet werden. Netzwerkvariablen stellen die einfachste Form des Datenaustausches mit anderen CANopen-Knoten dar. In einem SPS-Programm erfolgt der Zugriff auf Netzwerkvariablen in derselben Form wie auf interne, lokale Variablen der SPS. Aus Sicht des SPS-Programmierers ist es somit völlig unbedeutend, ob z.B. eine Input-Variable einem lokalen Eingang der Steuerung zugeordnet ist oder einen Eingang eines dezentralen Erweiterungsmoduls repräsentiert. Die Verwendung von Netzwerkvariablen basiert auf der Einbindung von DCF-Dateien, die durch einen entsprechenden CANopen-Konfigurator erstellt wurden. Die DCF-Dateien beschreiben zum einen die Kommunikationsparameter eines Gerätes (CAN Identifier, usw.) und zum anderen beinhalten sie eine Zuordnung der Netzwerkvariablen auf die einzelnen Bytes eines CAN-Telegramms (Mapping). Die Anwendung von Netzwerkvariablen erfordert nur allgemeine Grundkenntnisse über CANopen.

Der Austausch von PDOs erfolgt in einem CANopen-Netzwerk nur im Zustand *"OPERATIONAL"*. Befindet sich das PLCcore-E660 nicht in diesem Zustand, verarbeitet es keine PDOs (weder sende- noch empfangsseitig) und aktualisiert folglich auch nicht den Inhalt der Netzwerkvariablen. Das Setzen der Betriebszustände *"OPERATIONAL"*, *"PRE-OPERATIONAL"* usw. ist Aufgabe des CANopen-Managers (meist auch als "CANopen-Master" bezeichnet). In typischen CANopen-Netzwerken wird für den CANopen-Manager ein programmierbarer Knoten - meist in Form einer SPS - verwendet. Das PLCcore-E660 kann optional die Aufgabe des CANopen-Managers übernehmen. Die Aktivierung des Managers beschreibt Abschnitt 7.4.

Als CANopen-Manager ist das PLCcore-E660 zudem in der Lage, die am CAN-Bus angeschlossenen CANopen I/O-Geräte ("CANopen-Slaves") zu parametrieren, indem es die vom CANopen-Konfigurators erstellten DCF-Dateien beim Systemstart per SDO auf die jeweiligen Knoten überträgt.

6.9.1 CAN-Schnittstelle CAN0

Die Schnittstelle CAN0 besitzt ein dynamisches Object-Dictionary. Das bedeutet, dass diese Schnittstelle nach dem Einschalten der SPS zunächst überhaupt keine Kommunikationsobjekte für den Datenaustausch mit anderen Geräten zur Verfügung stellt. Erst nach dem Download eines SPS-Programms (bzw. dessen Rückladen aus dem nichtflüchtigen Speicher nach Power-on) werden die benötigten Kommunikationsobjekte anhand der in das SPS-Projekt eingebundenen DCF-Datei dynamisch angelegt. Dadurch ist die CAN-Schnittstelle CAN0 extrem flexibel auch für große Datenmengen nutzbar.

Auf Ebene des SPS-Programms werden die Netzwerkvariablen entsprechend der Norm IEC61131-3 als *"VAR_EXTERNAL"* deklariert und somit als "außerhalb der Steuerung" gekennzeichnet, z.B.:

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

Die detaillierte Vorgehensweise zum Einbinden von DCF-Dateien in das SPS-Projekt und zur Deklaration von Netzwerkvariablen beschreibt das *"User Manual CANopen-Erweiterung für IEC 61131-3"* (Manual-Nr.: L-1008).

Bei der Nutzung der CAN-Schnittstelle CAN0 ist zu beachten, dass aufgrund des dynamischen Objekt-Verzeichnisses das Anlegen der benötigten Objekte nach jedem Systemstart erneut erfolgt. Die "Aufbauvorschrift" dazu beinhaltet die in das SPS-Projekt eingebundene DCF-Datei. **Änderungen an der Konfiguration können daher nur durch entsprechende Modifikationen der DCF-Datei erfolgen.** Das bedeutet, dass nach einer Änderung der Netzwerkkonfiguration (Modifikation der DCF-Datei) das SPS-Projekt neu übersetzt und auf das PLCcore-E660 geladen werden muss.

6.9.2 Zusätzliche CAN-Schnittstellen

Die auf dem PLCcore-E660 eingesetzte SPS-Firmware ist prinzipiell in der Lage, mehrere CAN-Schnittstellen simultan zu bedienen (analog zu anderen SPS-Typen wie beispielsweise PLCcore-5484 oder PLCmodule-C32).

Bei Bedarf können weitere CAN-Schnittstellen extern an das Modul angeschlossen werden. Bitte setzen Sie sich bei Interesse dazu mit unserem Support in Verbindung:

support@systec-electronic.com

6.10 Integrierte Target-Visualisierung

Das PLCcore-E660-HMI **(nur Betsellnummer 3390090)** ist eine Kompakt-SPS mit integrierter Target-Visualisierung und damit optimal zum Aufbau von anwenderspezifischen HMI (**H**uman **M**achine **I**nterface) Applikationen geeignet. Die integrierte Target-Visualisierung des PLCcore-E660 basiert auf dem *SpiderControl MicroBrowser* der Firma iniNet Solutions GmbH (http://www.spidercontrol.net). Sie ermöglicht sowohl die Anzeige von Prozesswerten aus der SPS als auch die Weiterleitung von Benutzeraktionen an die SPS (z.B. Eingaben über Tochscreen, Scrollwheel und Matrixtastatur). Die Erstellung der auf dem Display angezeigten Seiten erfolgt mit dem *SpiderControl PLC Editor*, der als Zusatzkomponente zusammen mit dem Programmiersystem *OpenPCS* installiert wird.

Der Datenaustausch zwischen Target-Visualisierung und SPS-Programm erfolgt über Variablen des SPS-Programms. So lassen sich beispielsweise Prozessinformationen in beiden Richtungen austauschen (Übergabe einer anzuzeigenden Prozessgröße von der SPS an die Visualisierung, Übergabe eines an der Prozessvisualisierung eingegebenen Parameters an das SPS-Programm). Ebenso können aber auch Bedienerereignisse wie z.B. das Betätigen einer bestimmten Schaltfläche benutzt werden, um Werte von Variablen im SPS-Programm zu ändern (z.B. beim Betätigen einer Schaltfläche ändert sich der Wert der damit verknüpften Variable von 0 auf 1). Die Erstellung der Visualisierungs-Seiten mit dem *SpiderControl PLC Editor* notwenigen Schritte sowie die Verknüpfung von grafischen Elementen mit Variablen des SPS-Programms beschreibt das Manual "*SYS TEC spezifische HMI Erweiterungen für OpenPCS / IEC 61131-3"* (Manual-Nr.: L-1231).

Der Monitor, die Maus und die Tastatur arbeiten unmittelbar mit der Target-Visualisierung des PLCcore-E660 zusammen, d.h. Tastatur- und Maus-Events werden direkt vom *SpiderControl MicroBrowser* verarbeitet. Eine Weiterleitung von Tastatur- und Maus-Events an das SPS-Programm ist nicht vorgesehen, da eine sinnvolle Interpretation dieser Daten (X- und Y-Koordinate usw.) ohnehin nicht möglich ist.

7 Konfiguration und Administration des PLCcore-E660

7.1 Systemvoraussetzungen und erforderliche Softwaretools

Zur Administration des PLCcore-E660 ist ein beliebiger Windows- oder Linux-PC erforderlich, der über eine Ethernet-Schnittstelle sowie eine serielle Schnittstelle (RS232) verfügt. Als Alternative zur seriellen on-board Schnittstelle eignet sich auch das von SYS TEC angebotenen USB-RS232 Adapter Kabel (Bestellnummer 3234000, siehe Abschnitt 4.5.1), das eine entsprechende RS232-Schnittstelle über einen USB-Port zur Verfügung stellt.

Alle in diesem Manual aufgeführten Beispiele beziehen sich auf die Administration des PLCcore-E660 von einem Windows-PC aus. Das Vorgehen auf einem Linux-PC ist analog.

Zur Administration des PLCcore-E660 sind folgende Softwaretools erforderlich:

Terminalprogramm

Ein Terminalprogramm ermöglicht die Kommunikation mit der **Kommando-Shell** des PLCcore-E660 über eine **serielle RS232-Verbindung an UART1 des PLCcore-E660**. Diese ist Voraussetzung für die im Abschnitt 7.3 beschriebene Ethernet-Konfiguration des PLCcore-E660. Nach Abschluss der Ethernet-Konfiguration können alle weiteren Kommandos wahlweise entweder auch weiterhin im Terminalprogramm eingegeben werden oder alternativ dazu in einem Telnet-Client (siehe unten).

Als Terminalprogramm eignen sich z.B. das im Lieferumfang von Windows bereits enthaltenen *"HyperTerminal"* oder für gehobenere Ansprüche das als Open-Source verfügbare *"TeraTerm"* (Download unter: http://ttssh2.sourceforge.jp).

Telnet-Client

Ein Telnet-Client ermöglicht die Kommunikation mit der **Kommando-Shell** des PLCcore-E660 über eine **Ethernet-Verbindung an ETH0 des PLCcore-E660**. Voraussetzung für die Verwendung eines Telnet-Clients ist eine abgeschlossene Ethernet-Konfiguration des PLCcore-E660 gemäß Abschnitt 7.3. Alternativ zum Telnet-Client können auch sämtliche Kommandos über ein Terminalprogramm (an COM0 des PLCcore-E660) eingegeben werden.

Als Telnet-Client eignet sich z.B. das im Lieferumfang von Windows bereits enthaltene *"Telnet"* oder ebenfalls *"TeraTerm"*, das gleichzeitig auch als Terminalprogramm eingesetzt werden kann (siehe oben).

FTP-Client

Ein FTP-Client ermöglicht den Austausch von Dateien zwischen dem PLCcore-E660 (ETH0) und dem PC. Dies erlaubt beispielsweise das **Editieren von Konfigurationsdateien**, indem diese zunächst vom PLCcore-E660 auf den PC übertragen, dort mit einem Editor bearbeitet und anschließend wieder zurück auf das PLCcore-E660 geschrieben werden. Der Download von Dateien auf das PLCcore-E660 ist aber auch zum **Update der SPS-Firmware** erforderlich. (Hinweis: Der Update der *SPS-Firmware* ist nicht identisch mit dem Update des *SPS-Anwenderprogramms*. Das SPS-Programm wird direkt aus der OpenPCS-Programmierumgebung heraus auf das Modul übertragen, hierzu ist keinerlei Zusatzsoftware erforderlich.)

Als FTP-Client für den PC eignen sich beispielsweise das als Open-Source verfügbare *"WinSCP"* (Download unter: http://winscp.net), das lediglich aus einer einzelnen EXE-Datei besteht, die keine Installation erfordert und sofort

gestartet werden kann. Ebenso geeignet sind aber auch die Freeware *"Core FTP LE"* (Download unter: http://www.coreftp.com) oder der bereits im Dateimanager *"Total Commander"* integrierte FTP-Client.

Bei Programmen die über die Ethernet-Schnittstelle kommunizieren, wie beispielsweise FTP-Client oder TFTP-Server, ist darauf zu achten, dass die entsprechenden Rechte in der Windows-Firewall freigegeben sind. In der Regel melden Firewalls, dass ein Programm Zugriff auf das Netzwerk erlangen möchte und fragen, ob dieser Zugriff erlaubt oder abgeblockt werden soll. Hier ist in jedem Fall der entsprechende Zugriff zu gestatten.

7.2 Linux-Autostart aktivieren bzw. deaktivieren

Um beim Systemstart das Booten von Linux zu gestatten bzw. zu verhindern wird der UEFI-Bootloader eingesetzt (Einzelheiten zum Systemstart und zum Bootloader: siehe Abschnitte 6.2 und 6.3)

Die Kommunikation mit dem UEFI-Bootloader erfolgt ausschließlich über die serielle Schnittstelle UART1 des PLCcore-E660. Als Gegenstelle ist auf dem PC ein beliebiges Terminalprogramm zu starten (z.B. HyperTerminal oder TeraTerm, siehe Abschnitt 7.1) und wie folgt zu konfigurieren (siehe Bild 10):

- 115200 Baud
- 8 Datenbits
- 1 Stopbit
- keine Parität
- keine Flusskontrolle

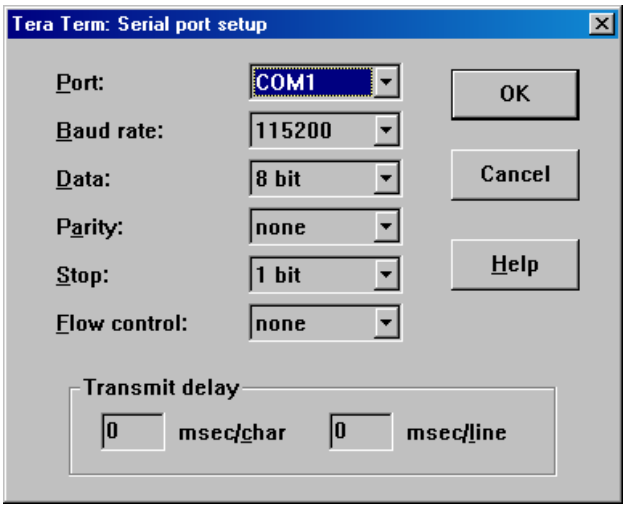


Bild 10: Terminaleinstellungen am Beispiel von "TeraTerm"

7.3 Ethernet-Konfiguration des PLCcore-E660

Die zentrale Ethernet-Konfiguration des PLCcore-E660 erfolgt im UEFI-Bootloader und wird von hier für alle anderen Softwarekomponenten übernommen (Linux, SPS-Firmware, HTTP-Server usw.) Die Ethernet-Konfiguration erfolgt über die serielle Schnittstelle UART1. **Dazu ist wie im Abschnitt 6.3.2**

beschrieben der EFI-Shell-Kommandoprompt zu aktivieren. Tabelle 14 listet die zur Ethernet-Konfiguration des PLCcore-E660 notwendigen EFI-Shell-Kommandos auf.

Tabelle 14: EFI-Shell-Kommandos zur Konfiguration des PLCcore-E660

Einstellung	Kommando	Bemerkung
MAC-Adresse	setenv ethaddr <xx:xx:xx:xx:xx:xx>	Die MAC-Adresse ist eine weltweit eindeutige Kennung des Moduls, die bereits vom Hersteller vergeben wird. **Diese sollte vom Anwender normalerweise nicht verändert werden.**
IP-Adresse	setenv ipaddr <xxx.xxx.xxx.xxx>	Dieses Kommando setzt die lokale IP-Adresse des PLCcore-E660. Die IP-Adresse ist vom Netzwerkadministrator festzulegen.
Netzwerkmaske	setenv netmask <xxx.xxx.xxx.xxx>	Dieses Kommando setzt die Netzwerkmaske des PLCcore-E660. Die Netzwerkmaske ist vom Netzwerkadministrator festzulegen.
Gateway-Adresse	setenv gatewayip <xxx.xxx.xxx.xxx>	Dieses Kommando definiert die IP-Adresse des vom PLCcore-E660 zu benutzenden Gateways. Die Gateway-Adresse ist vom Netzwerkadministrator festzulegen. **Hinweis:** Befinden sich PLCcore-E660 und Programmier-PC im selben Sub-Netz, dann kann die Definition der Gateway-Adresse entfallen und stattdessen der Wert "0.0.0.0" verwendet werden.

Die modifizierten Einstellungen können durch die Eingabe von *"printenv"* am EFI-Shell-Kommandoprompt nochmals überprüft werden.

Nach Abschluss der Konfiguration sind gemäß Abschnitt 6.3.2 die Voraussetzungen für einen Linux-Autostart wieder herzustellen.

Nach Reset (z.B. Taster "C RES" am Developmentboard) startet das Modul mit den aktuellen Einstellungen.

Hinweis: Nach Abschluss der Konfiguration ist die serielle Verbindung zwischen PC und PLCcore-E660 nicht mehr erforderlich.

7.4 SPS-Konfiguration des PLCcore-E660

7.4.1 SPS-Konfiguration über WEB-Frontend

Nach dem Abschluss der Ethernet-Konfiguration (siehe Abschnitt 7.3) können alle weiteren Einstellungen über das integrierte WEB-Frontend des PLCcore-E660 erfolgen.

Für die Konfiguration des PLCcore-E660 über das WEB-Frontend ist auf dem PC lediglich ein WEB-Browser erforderlich (z.B. Microsoft Internet Explorer, Mozilla Firefox usw.). Zum Aufruf der Konfigurationsseite ist in der Adressleiste des WEB-Browser der Präfix *"http://"* gefolgt von der im

Abschnitt 7.2 festgelegten IP-Adresse des PLCcore-E660 einzugeben, z.B. *"http://192.168.10.130"*. Bild 11 verdeutlicht den Aufruf der Konfigurationsseite für das PLCcore-E660 im WEB-Browser.

In der Standardeinstellung (Werkseinstellungen) erfordert die Konfiguration des PLCcore-E660 über das WEB-Frontend eine Benutzeranmeldung, um unbefugte Zugriffe zu unterbinden. Dazu sind Nutzername und Passwort in dem entsprechenden Anmeldedialog einzugeben (siehe Bild 11). Bei Auslieferung des Moduls ist folgendes Nutzerkonto vorkonfiguriert (siehe auch Abschnitt 7.7):

User: PlcAdmin
Passwort: Plc123

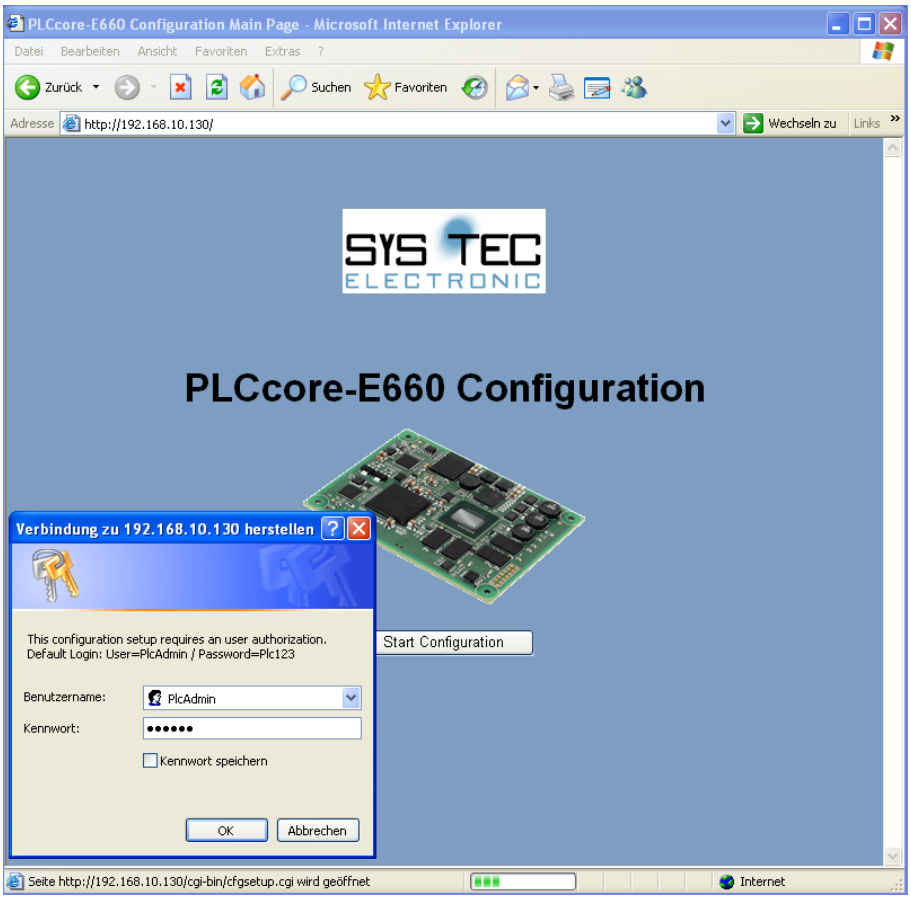


Bild 11: Anmeldedialog des WEB-Frontend

Sämtliche Konfigurationseinstellungen für das PLCcore-E660 erfolgen dialogbasiert und werden durch Betätigen der Schaltfläche "Save Configuration" in die Datei ***"/home/plc/bin/plccore-E660.cfg"*** des PLCcore-E660 übernommen (siehe auch Abschnitt 7.4.2). Nach Reset (z.B. Taster "C RES" am Developmentboard) startet das PLCcore-E660 mit den aktuellen Einstellungen. Bild 12 zeigt die Konfiguration des PLCcore-E660 über das WEB-Frontend.

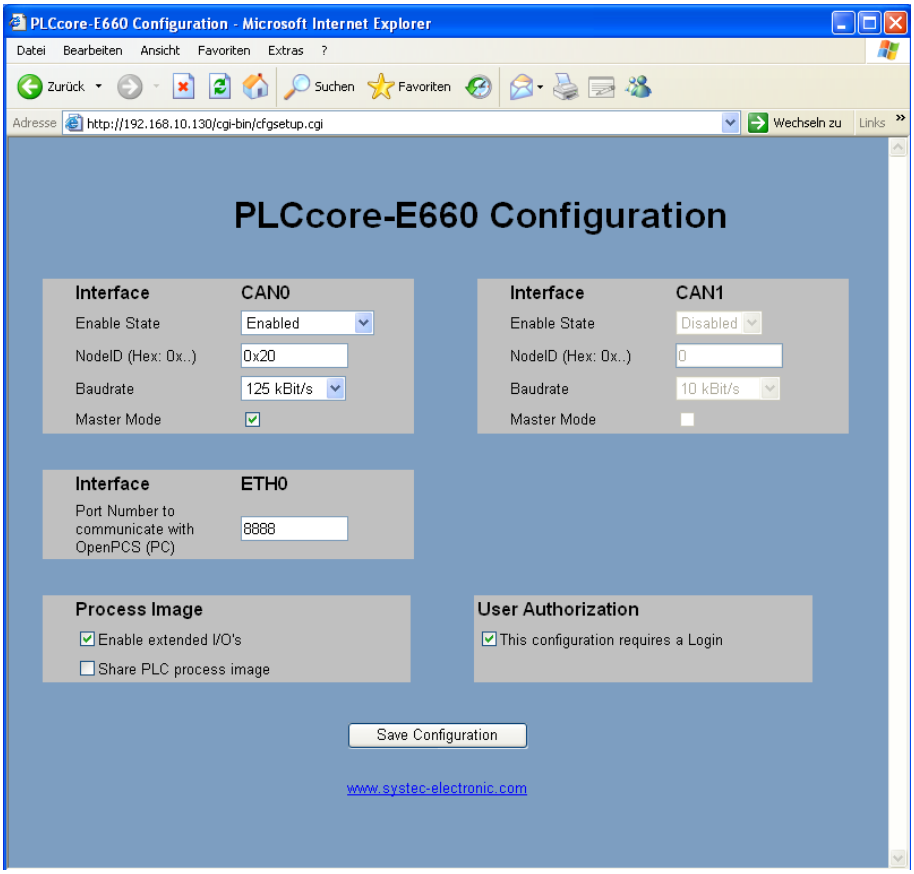


Bild 12: SPS-Konfiguration über WEB-Frontend

In der Standardeinstellung (Werkseinstellungen) ist das PLCcore-E660 so konfiguriert, dass für den Zugriff auf das WEB-Frontend eine Benutzeranmeldung erforderlich ist. Dabei wird nur der in der Konfigurationsdatei ***"/home/plc/bin/plccore-E660.cfg"*** angegebene Benutzername akzeptiert (Eintrag *"User="* in Sektion *"[Login]"*, siehe Abschnitt 7.4.2). Die Vorgehensweise zum Ändern des Passwortes für die Nutzeranmeldung beschreibt Abschnitt 7.10. Um einem anderen Benutzer die Konfiguration des Moduls zu ermöglichen, ist zunächst das entsprechende Nutzerkonto wie im Abschnitt 7.9 beschrieben anzulegen. Anschließend ist der neue Benutzername in der Konfigurationsdatei ***"/home/plc/bin/plccore-E660.cfg"*** einzutragen. Durch Löschen des Eintrages *"User="* in Sektion *"[Login]"* (siehe 7.4.2) wird die Limitierung auf einen Benutzer aufgehoben, so dass für die Konfiguration jeder auf dem Modul angelegte Nutzer-Account verwendet werden kann. Durch deaktivieren des Kontrollkästchens *"This configuration requires a Login"* im Feld *"User Authorization"* der Konfigurationsseite (siehe Bild 12) wird ein freier Zugriff auf die Konfiguration ohne vorherige Nutzeranmeldung ermöglicht.

7.4.2 Aufbau der Konfigurationsdatei "plccore-E660.cfg"

Die Konfigurationsdatei ***"/home/plc/bin/plccore-E660.cfg"*** ermöglicht eine umfassende Konfiguration des PLCcore-E660. Ihre manuelle Bearbeitung ist jedoch nur in Ausnahmefällen sinnvoll, da die meisten der darin enthaltenen Einstellungen bequem über das WEB-Frontend editiert werden können (siehe Abschnitt 7.4.1). Der Aufbau der Konfigurationsdatei entspricht dem als "Windows INI-File" bekannten Datenformat, sie untergliedert sich in "*[Sections]*", die dann wiederum verschiedene "*Entry=*" Einträge enthalten. Tabelle 15 enthält eine Auflistung der Konfigurationseinträge.

Tabelle 15: Konfigurationseinträge der CFG-Datei

Section	Entry	Value	Bedeutung
[CAN0]	Enabled	-1, 0, 1	0: Interface CAN0 ist deaktiviert 1: Interface CAN0 ist aktiviert, Konfiguration erfolgt über nachstehende Einträge dieser Konfigurationsdatei
	NodeID	1 … 127 bzw. 0x01 … 0x7F	Knotennummer für Interface CAN0 (dezimal oder hexadezimal mit Präfix "0x")
	Baudrate	10, 20, 50, 125, 250, 500, 800, 1000	Bitrate für Interface CAN0
	MasterMode	0, 1	1: Master-Modus ist aktiviert 0: Master-Modus ist deaktiviert
[CAN1]	Enabled		Die Sektion "[CAN1]" wird standardmäßig nicht ausgewertet, ermöglicht aber bei Bedarf eine Erweiterung des PLCcore-E660 um eine zusätzliche CAN-Schnittstelle (siehe Abschnitt 6.9.2)
	NodeID		
	Baudrate		
	MasterMode		
[ETH0]	PortNum	Default Portnummer: 8888	Portnummer zur Kommunikation mit dem Programmier-PC sowie zum Programmdownload (nur für PLCcore-E660/Z5, Bestellnummer 3390055)
[ProcImg]	EnableExtIo	0, 1	0: nur on-board I/O's des PLCcore-E660 für Prozessabbild verwenden (jedoch ohne Temperatursensor) 1: alle vom Treiber unterstützten I/O's für Prozessabbild verwenden (incl. Temperatursensor und externer ADC des Developmentboard) (zur Anpassung des Prozessabbildes siehe Abschnitt 8.2)
	EnableSharing	0, 1	0: kein Sharing für Prozessabbild 1: Sharing für Prozessabbild ist freigegeben (siehe Abschnitt 8.1)
[Visu]	Enable	0, 1	1: Visualisierung ist aktiviert 0: Visualisierung ist deaktiviert
	SyncTime	0, 1…n	0: Synchronisation der Daten zwischen SPS und Visualisierung nach jedem SPS-Zyklus >0: Synchronisation der Daten zwischen SPS und Visualisierung nach <SyncTime> ms

Section	Entry	Value	Bedeutung
[Login]	Authorization	0, 1	0: Konfiguration über WEB-Frontend ist ohne Benutzeranmeldung möglich 1: Konfiguration über WEB-Frontend erfordert Benutzeranmeldung
	User	Default Name: PlcAdmin	Ist der Eintrag *"User="* vorhanden, wird nur der darin definierte Benutzername bei der Anmeldung für die Konfiguration über das WEB-Frontend akzeptiert. Ist der Eintrag nicht vorhanden, kann sich jeder auf dem PLCcore-E660 angelegte Benutzer (siehe Abschnitt 7.9) zur Konfiguration über das WEB-Frontend anmelden.

Die Konfigurationsdatei ***"/home/plc/bin/plccore-E660.cfg"*** besitzt folgende Werkseinstellungen:

```
[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-Konfiguration des PLCcore-E660

Das PLCcore-E660 ist so konfiguriert, dass nach einem Reset die SPS-Firmware automatisch gestartet wird. Die dazu notwendigen Kommandos sind in dem Startskript ***"/home/etc/autostart"*** hinterlegt. Hier werden u.a. die notwendigen Umgebungsvariablen gesetzt sowie die erforderlichen Treiber geladen.

Bei Bedarf kann das Startskript ***"/home/etc/autostart"*** um weitere Einträge ergänzt werden. Hier kann z.B. durch einfügen des Kommandos *"pureftp"* der Aufruf des FTP-Servers beim Booten des PLCcore-E660 automatisiert werden. Das Skript lässt sich im FTP-Client *"WinSCP"* (siehe Abschnitt 7.8.2) mit Hilfe der Taste *"F4"* bzw. der Schaltfläche *"F4 Edit"* direkt auf dem PLCcore-E660 bearbeiten.

7.6 Auswahl der zu startenden Firmware-Variante

Das PLCcore-E660 wird mit verschiedenen Firmware-Varianten ausgeliefert. Diese unterscheiden sich im verwendeten Kommunikationsprotokoll für den Datenaustausch mit dem Programmier-PC und in der Verfügbarkeit der Kommunikations-FB-Klassen (siehe Abschnitt 6.4). Die Auswahl der zu verwendenden Firmware-Varianten erfolgt im Startskript ***"/home/etc/autostart"***. Standardmäßig wird hierzu die im EFI-Bootloader festgelegte *"BoardID"* des Moduls ausgewertet. Tabelle 16 listet die Zuordnung von Firmware-Variante und BoardID auf.

Tabelle 16: Zuordnung von BoardID und Firmware-Variante für das PLCcore-E660

BoardID	Firmware-Variante	Bemerkung
1011003	plccore-E660-z3	**PLCcore-E660/Z3 (RS232)** Kommunikation mit Programmier-PC via Siemens Protokoll 3964(R) (Interface UART1)
1011004	plccore-E660-z4	**PLCcore-E660/Z4 (CANopen)** Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
1011005	plccore-E660-z5	**PLCcore-E660/Z5 (Ethernet)** Kommunikation mit Programmier-PC via UDP-Protokoll (Interface ETH0)
1011013	plccore-E660-hmi-z3	**PLCcore-E660-HMI/Z3 (RS232 mit Target-Visualisierung)** Kommunikation mit Programmier-PC via Siemens Protokoll 3964(R) (Interface UART1)
1011014	plccore-E660-hmi-z4	**PLCcore-E660-HMI/Z4 (CANopen mit Target-Visualisierung)** Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
1011015	plccore-E660-hmi-z5	**PLCcore-E660-HMI/Z5 (Ethernet mit Target-Visualisierung)** Kommunikation mit Programmier-PC via UDP-Protokoll (Interface ETH0)

Die Konfiguration der BoardID erfolgt über die serielle Schnittstelle UART1. **Dazu ist wie im Abschnitt 6.3.2 beschrieben der EFI-Shell-Kommandoprompt zu aktivieren.** Das Festlegen der jeweiligen BoardID erfolgt mit dem EFI-Shell-Kommando *"setenv boardid"* unter Angabe der in Tabelle 16 aufgeführten, zugehörigen Nummer, z.B.:

```
setenv  boardid  1011005
```

Die modifizierte Einstellung kann durch die Eingabe von "*printenv*" am EFI-Shell Kommandoprompt nochmals überprüft werden.

Nach Abschluss der Konfiguration sind gemäß Abschnitt 6.3.2die Voraussetzungen für einen Linux-Autostart wieder herzustellen.

Alternativ kann die zu startende Firmware auch direkt im Startskript ***"/home/etc/autostart"*** festgelegt werden. Dazu ist der Abschnitt "Select PLC Type" zu löschen und stattdessen die entsprechende Firmware fest vorzugeben, z.B.:

```
PLC_FIRMWARE=$PLC_DIR/plccore-E660-z5
```

7.7 Vordefinierte Nutzerkonten

Bei der Auslieferung des PLCcore-E660 sind die in Tabelle 17 aufgelisteten Benutzerkonten vordefiniert. Diese erlauben eine Anmeldung an der Kommando-Shell (serielle RS232-Verbindung oder Telnet) und am FTP-Server des PLCcore-E660.

Tabelle 17: Vordefinierte Benutzerkonten des PLCcore-E660

Benutzername	Passwort	Bemerkung
PlcAdmin	Plc123	vordefiniertes Benutzerkonto zur Administration des PLCcore-E660 (Konfiguration, Nutzerverwaltung, Softwareupdates usw.)
root	Sys123	Haupt-Benutzerkonto ("root") des PLCcore-E660

7.8 Anmeldung am PLCcore-E660

7.8.1 Anmeldung an der Kommando-Shell

Die Administration des PLCcore-E660 erfordert in einigen Fällen die Eingabe von Linux-Kommandos in der Kommando-Shell. Dazu ist eine direkte Anmeldung am Modul notwendig. Dies kann auf zwei alternativen Wegen erfolgen:

- Mit Hilfe eines **Terminalprogramms** (z.B. HyperTerminal oder TeraTerm, siehe Abschnitt 7.1) über die serielle Schnittstelle **UART1** des PLCcore-E660, analog zum Vorgehen bei der im Abschnitt 7.2 beschriebenen Ethernet-Konfiguration. **Bei der Konfiguration der Terminaleinstellungen ist darauf zu achten, dass als Zeilenendezeichen nur "CR" (carriage return) verwendet wird.** Bei "CR+LF" (carriage return + line feed) ist keine Anmeldung mit Nutzername und Passwort möglich!

- Alternativ ist die Anmeldung mit Hilfe eines **Telnet-Clients** (z.B. Telnet oder ebenfalls TeraTerm) über die Ethernet-Schnittstelle **ETH0** des PLCcore-E660 möglich.

Um sich über den in Windows standardmäßig enthaltenen Telnet-Client am PLCcore-E660 anzumelden, ist der Befehl "*telnet*" unter Angabe der in Abschnitt 7.3 festgelegten IP-Adresse für das PLCcore-E660 aufzurufen, z.B.

```
telnet  192.168.10.248
```

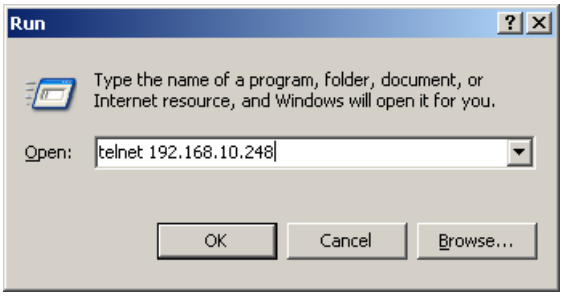


Bild 13: Aufruf des Telnet-Clients unter Windows

Innerhalb des Terminal-Fensters (bei Verbindung über UART1) bzw. des Telnet-Fensters (bei Verwendung von ETH0) ist die Anmeldung am PLCcore-E660 möglich. Bei Auslieferung des Moduls ist zur Administration des PLCcore-E660 folgendes Nutzerkonto vorkonfiguriert (siehe auch Abschnitt 7.7):

User: *PlcAdmin*
Passwort: *Plc123*

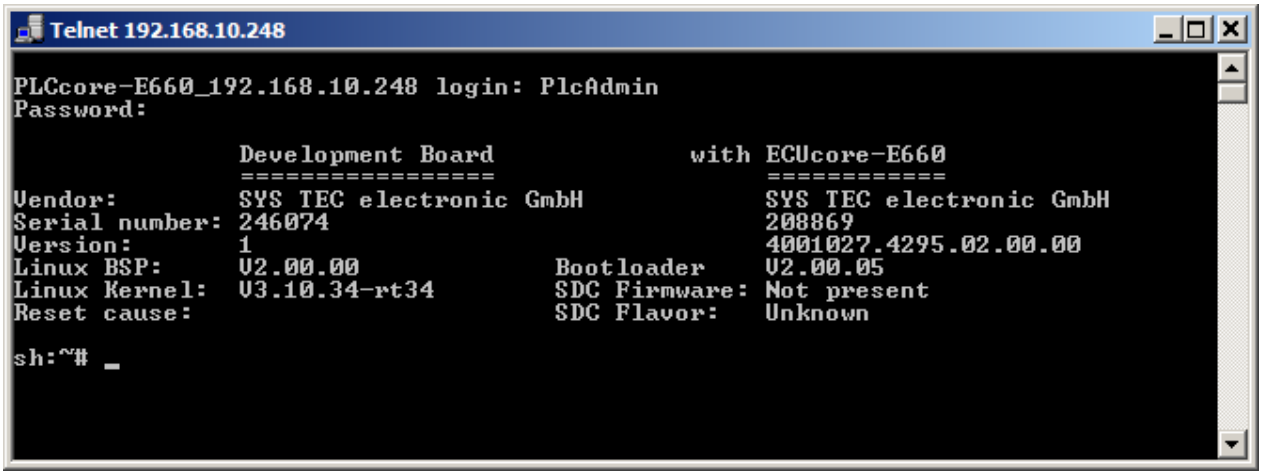


Bild 14: Anmeldung am PLCcore-E660

Bild 14 verdeutlicht am Beispiel die Anmeldung am PLCcore-E660 mit Hilfe des in Windows standardmäßig enthaltenen Telnet-Clients.

7.8.2 Anmeldung am FTP-Server

Das PLCcore-E660 verfügt über einen FTP-Server (FTP Daemon), der den Austausch von Dateien mit einem PC ermöglicht (Up- und Download von Dateien). Aus Sicherheits- und Performance-Gründen ist dieser FTP-Server jedoch standardmäßig deaktiviert und muss bei Bedarf zunächst manuell gestartet werden. Hierzu ist zunächst die im Abschnitt 7.8.1 beschriebenen Anmeldung an der Kommando-Shell des PLCcore-E660 durchzuführen. Anschließend ist im Telnet- bzw. Terminal-Fenster folgendes Kommando einzugeben:

```
pureftp
```

Bild 15 verdeutlicht am Beispiel das Starten des FTP-Servers.

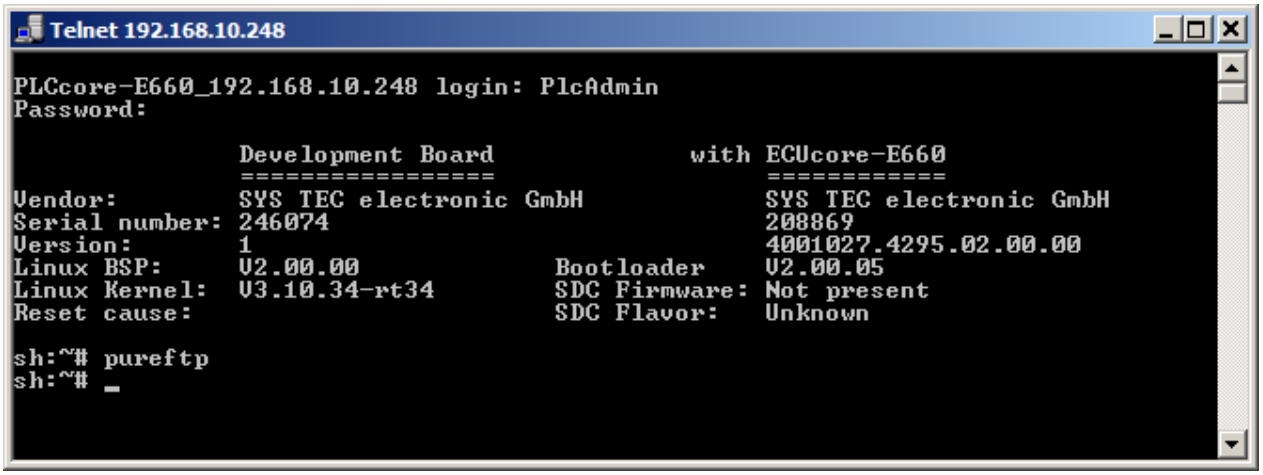


Bild 15: Starten des FTP-Servers

Hinweis: Durch einfügen des Kommandos *"pureftp"* in das Startskript ***"/home/etc/autostart"*** lässt sich der Aufruf des FTP-Servers beim Booten des PLCcore-E660 automatisieren (siehe dazu Abschnitt 7.5).

Als FTP-Client für den PC eignet sich beispielsweise das als Open-Source verfügbare *"WinSCP"* (siehe Abschnitt 7.1), das lediglich aus einer einzelnen EXE-Datei besteht, die keine Installation erfordert und sofort gestartet werden kann. Nach dem Programmstart erscheint zunächst der Dialog *"WinSCP Login"* (siehe Bild 16), in dem folgende Einstellungen vorzunehmen sind:

File protocol: FTP
Host name: die im Abschnitt 7.3 festgelegte IP-Adresse für das PLCcore-E660
User name: *PlcAdmin* (für vorkonfiguriertes Nutzerkonto, siehe Abschnitt 7.7)
Password: *Plc123* (für vorkonfiguriertes Nutzerkonto, siehe Abschnitt 7.7)

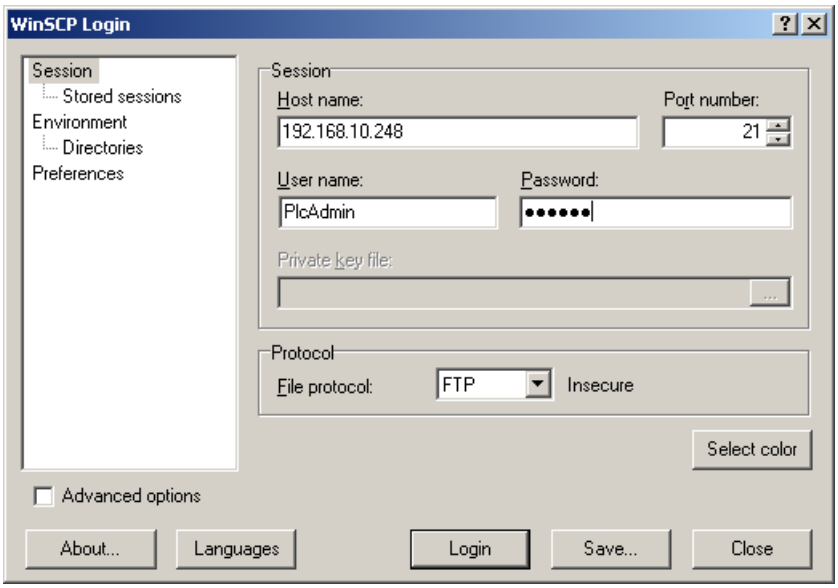


Bild 16: Login-Einstellungen für WinSCP

Nach dem Betätigen der Schaltfläche *"Login"* meldet sich der FTP-Client am PLCcore-E660 an und listet im rechten Fenster den aktuellen Inhalt des Verzeichnisses *"/home"* auf. Bild 17 zeigt den FTP-Client *"WinSCP"* nach der erfolgreichen Anmeldung am PLCcore-E660.

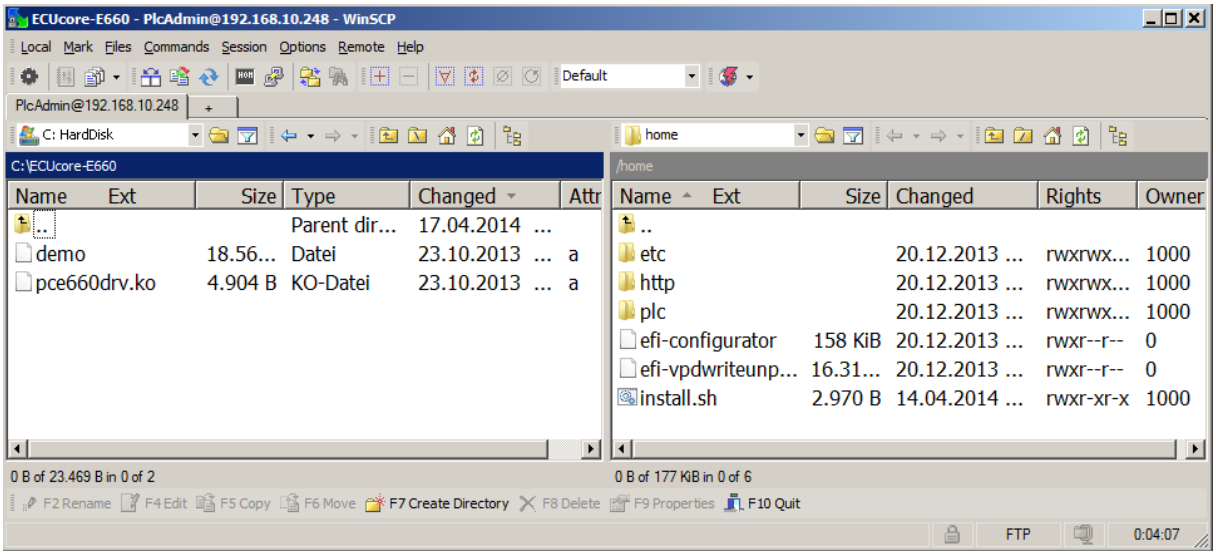


Bild 17: FTP-Client für Windows "WinSCP"

Nach einer erfolgreichen Anmeldung lassen sich innerhalb des FTP-Clients *"WinSCP"* mit Hilfe der Taste *"F4"* bzw. der Schaltfläche *"F4 Edit"* Konfigurationsdateien auf dem PLCcore-E660 bearbeiten (dazu Transfermodus *"Text"* selektieren). Mit Hilfe der Taste *"F5"* bzw. der Schaltfläche *"F5 Copy"* können Dateien zwischen dem PC und dem PLCcore-E660 transferieren werden, um beispielsweise Datensicherungen des PLCcore-E660 durchzuführen oder um Installationsdateien für Firmware-Updates auf das Modul zu übertragen (dazu Transfermodus *"Binary"* selektieren).

7.9 Anlegen und Löschen von Nutzerkonten

Das Anlegen und Löschen von Nutzerkonten erfordert zunächst die Anmeldung am PLCcore-E660 wie in Abschnitt 7.8.1 beschrieben.

Das **Anlegen** eines neuen Nutzerkontos erfolgt mit Hilfe des Linux-Kommandos *"adduser"*. Da es bei einem Embedded System wie dem PLCcore-E660 nicht sinnvoll ist, für jeden Benutzer ein eigenes Verzeichnis anzulegen, ist mit dem Parameter *"-H"* das Erstellen eines neuen Verzeichnisses zu unterbinden. Stattdessen wird dem neuen Benutzer über den Parameter *"-h /home"* das bereits vorhandene Verzeichnis *"/home"* zugeordnet. Zum Anlegen eines neuen Nutzerkontos auf dem PLCcore-E660 ist das Linux-Kommandos *"adduser"* wie folgt anzuwenden:

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

Bild 18 verdeutlicht am Beispiel das Anlegen eines neuen Kontos auf dem PLCcore-E660 für den Nutzer *"admin2"*.

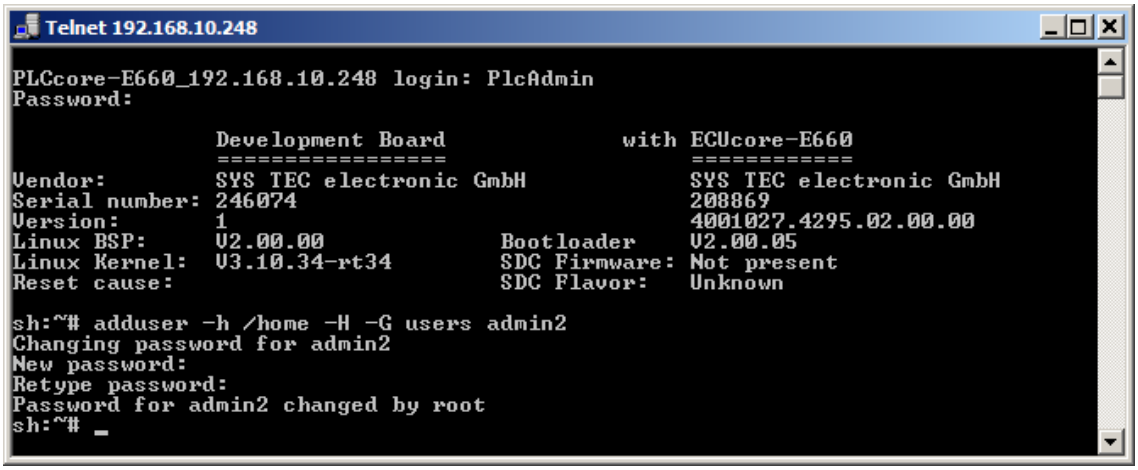


Bild 18: Anlegen eines neuen Nutzerkontos

Hinweis: **Falls das neu angelegte Benutzerkonto für den Zugriff auf das WEB-Frontend genutzt werden soll, ist der Benutzername gegebenenfalls noch in die Konfigurationsdatei *"plccore-E660.cfg"* einzutragen (für Details zur Anmeldung an das WEB-Frontend siehe Abschnitte 7.4.1 und 7.4.2).**

Zum **Löschen** eines vorhandenen Nutzerkontos vom PLCcore-E660 ist das Linux-Kommando *"deluser"* unter Angabe des Benutzernamens zu verwenden:

```
deluser  <username>
```

7.10 Passwort eines Nutzerkontos ändern

Das Ändern von Passwörtern erfordert zunächst die Anmeldung am PLCcore-E660 wie in Abschnitt 7.8.1 beschrieben.

Zum Ändern des Passwortes für ein auf dem PLCcore-E660 vorhandenen Nutzerkonto ist das Linux-Kommando *"passwd"* unter Angabe des Benutzernamens zu verwenden:

```
passwd <username>
```

Bild 19 verdeutlicht am Beispiel das Ändern des Passwortes für den Nutzer *"PlcAdmin"*.

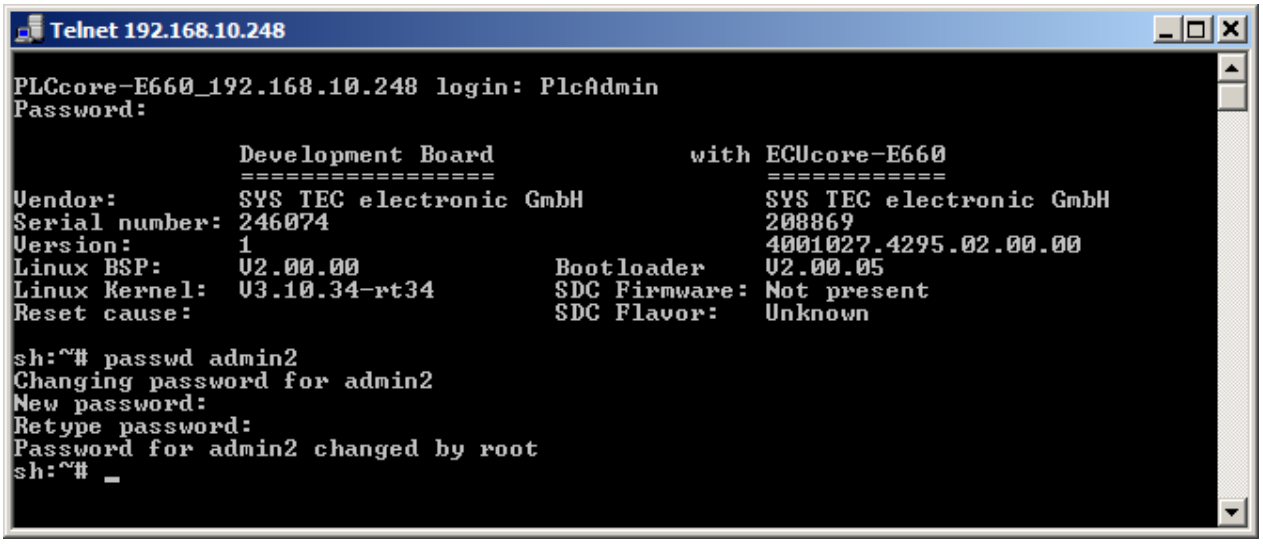


Bild 19: Passwort eines Nutzerkontos ändern

7.11 Setzen der Systemzeit

Das Setzen der Systemzeit erfordert zunächst die Anmeldung am PLCcore-E660 wie in Abschnitt 7.8.1 beschrieben.

Das Setzen der Systemzeit für das PLCcore-E660 erfolgt in zwei Schritten. Zuerst sind Datum und Uhrzeit mit Hilfe des Linux-Kommandos *"date"* auf die aktuellen Werte zu setzen. Anschließend wird die Systemzeit durch das Linux-Kommando *"hwclock -w"* in den RTC-Baustein des PLCcore-E660 übernommen.

Das Linux-Kommando *"date"* besitzt folgenden Aufbau:

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

Beispiel:

```
date   2014.02.25-11:34:55
          |  |  |  |  |  |
          |  |  |  |  |  +--- Sekunde
          |  |  |  |  +------ Minute
          |  |  |  +--------- Stunde
          |  |  +------------ Tag
          |  +--------------- Monat
          +------------------ Jahr
```

Um die aktuelle Systemzeit des PLCcore-E660 wie im obigen Beispiel dargestellt auf den 2014/02/25 um 11:34:55 zu setzen, ist folgende Befehlssequenz notwendig:

```
date  2014.02.25-11:34:55
hwclock  -w
```

Die aktuelle Systemzeit wird durch Eingabe des Linux-Kommandos *"date"* (ohne Parameter) angezeigt, die aktuellen Werte der RTC lassen sich durch das Linux-Kommando *"hwclock -r"* abfragen. Mit *"hwclock -s"* werden die aktuellen Werte der RTC als Systemzeit für Linux übernommen (Synchronisation des Kernels auf die RTC). Bild 20 verdeutlicht das Setzen und Anzeigen der Systemzeit am Beispiel.

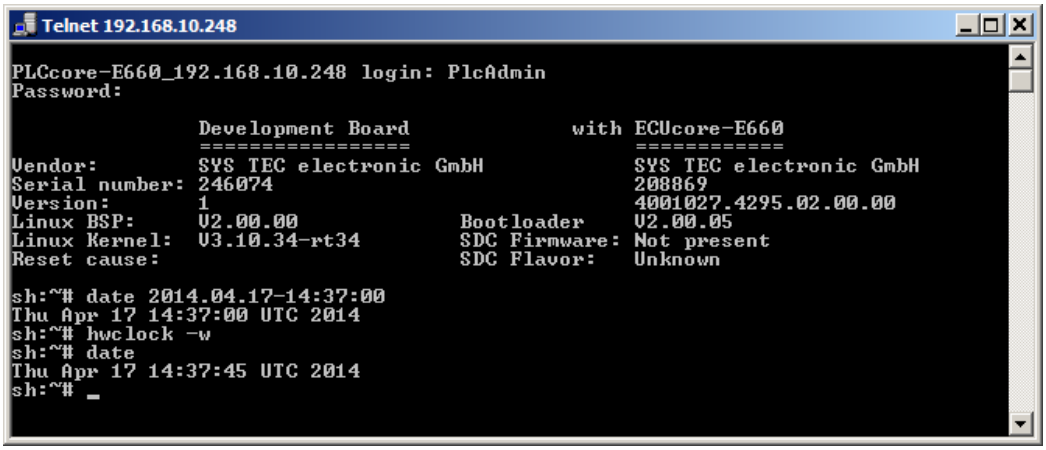


Bild 20: Setzen und Anzeigen der Systemzeit

Beim Starten des PLCcore-E660 werden Datum und Uhrzeit der RTC als aktuelle Systemzeit für das Modul übernommen. Das dazu notwendige Linux-Kommando *"hwclock -s"* ist bereits im Startskript ***"/etc/init.d/hwclock"*** enthalten.

7.12 Dateisystem des PLCcore-E660

Das auf dem PLCcore-E660 vorinstallierte Embedded Linux stellt Teile des Systemspeichers in Form eines Dateisystems zur Verfügung. Wie bei Embedded Systemen üblich ist dabei der überwiegende Teil des Dateisystems read/only, das bedeutet, dass Änderungen an diesem Teil können nur durch Neuerstellung des Linux-Images für das PLCcore-E660 vorgenommen werden können. Der große Vorteil besteht hier in der Resistenz des read/only Dateisystems gegen Beschädigungen beim Ausfall der Spannungsversorgung. Diese treten bei Embedded Systemen relativ häufig auf, da Embedded Systeme in der Regel einfach nur ausgeschaltet werden, ohne vorherigen Shutdown.

Die zur Laufzeit beschreibbaren Zweige des Dateisystems listet Tabelle 18 auf. Hinter dem Verzeichnis **"/home"** verbirgt sich eine Flash-Disk, die einen Teil des on-baord Flash-Speichers des PLCcore-E660 als Dateisystem zur Verfügung stellt. In diesem Pfad werden sämtliche vom Anwender modifizierbaren und updatefähigen Files wie beispielsweise Konfigurationsdateien, SPS-Firmware sowie die auf das Modul geladenen SPS-Programm-Dateien abgelegt. Das Verzeichnis **"/tmp"** ist in seiner Größe so dimensioniert, dass es als Zwischenpuffer für den FTP-Download des Firmware-Archives beim Update der SPS-Firmware benutzt werden kann (siehe Abschnitt 7.13.1).

Tabelle 18: Dateisystemkonfiguration des PLCcore-E660

Zweig	Nutzbare Größe	Beschreibung
/	129983 kByte	Flash-Disk für dauerhaftes Speichern, änderbar vom Benutzer (z. B. Nutzersoftware und Konfigurationsdateien); Datenerhalt bei Spannungsausfall
/tmp	496616 kByte[1]	RAM-Disk, geeignet als Zwischenpuffer für FTP-Downloads, aber kein Datenerhalt bei Spannungsausfall
/var/log	496616 kByte[1]	RAM-Disk, vom System genutzt zur Speicherung temporärer Dateien, kein Datenerhalt bei Spannungsausfall
/var/run	496616 kByte[1]	RAM-Disk, vom System genutzt zur Speicherung temporärer Dateien, kein Datenerhalt bei Spannungsausfall
/var/lock	496616 kByte[1]	RAM-Disk, vom System genutzt zur Speicherung temporärer Dateien, kein Datenerhalt bei Spannungsausfall
/var/tmp	496616 kByte[1]	RAM-Disk, vom System genutzt zur Speicherung temporärer Dateien, kein Datenerhalt bei Spannungsausfall
/mnt		Target zur Integration von Remote-Verzeichnissen anderer Systeme via NFS

[1]) Alle RAM-Disks teilen sich denselben RAM. Das bedeutet, dass die wirklich verfügbare Größe einer speziellen RAM-Disk abhängig von der Nutzung durch andere RAM-Disks variieren kann.

Die konfigurierten sowie jeweils aktuell noch verfügbaren Größen der einzelnen Dateisystemzweige können mit Hilfe des Linux-Kommandos "df" ("DiskFree") ermittelt werden (siehe Bild 21).

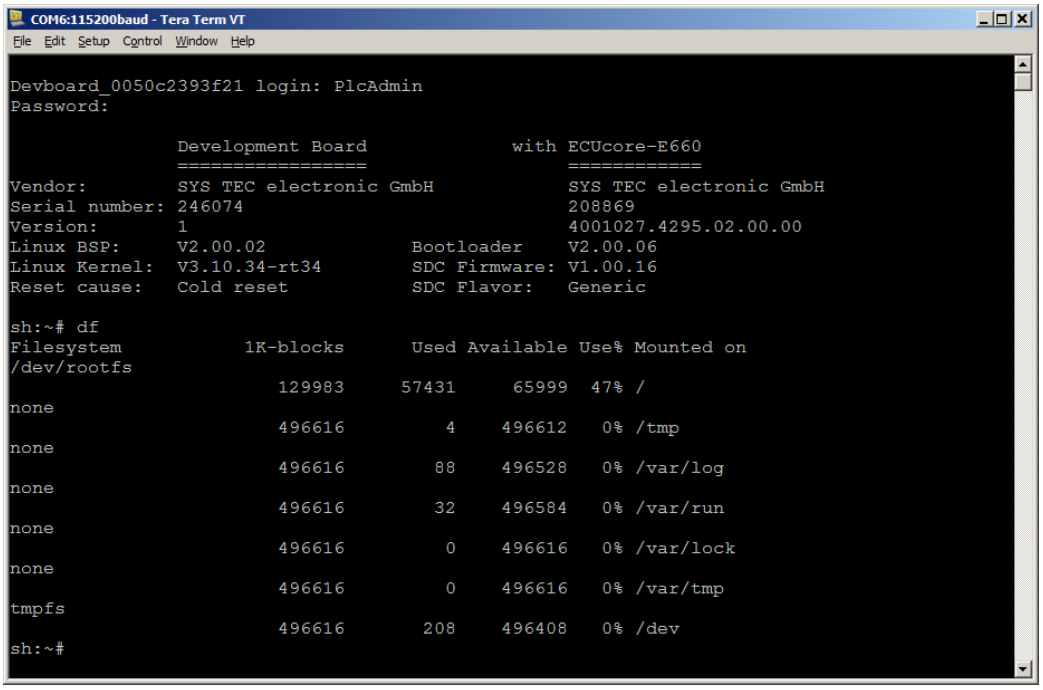


Bild 21: Anzeige von Informationen zum Dateisystem

Einzelheiten zur Anmeldung am System und zum Umgang mit der Linux-Kommando-Shell des PLCcore-E660 behandelt Abschnitt 7.8.

7.13 Software-Update des PLCcore-E660

Bei Auslieferung ab Werk sind bereits sämtliche für den Betrieb des PLCcore-E660 notwendigen Firmwarekomponenten auf dem Modul installiert. Ein Firmwareupdate ist daher nur in Ausnahmefällen erforderlich, um beispielsweise eine aktuellere Software mit neuen Eigenschaften einzuspielen.

7.13.1 Update der SPS-Firmware

Als SPS-Firmware wir die Laufzeitumgebung der SPS bezeichnet. Die ***SPS-Firmware*** kann nur vom Hersteller generiert und modifiziert werden, sie ist nicht identisch mit dem ***SPS-Anwenderprogramm***, das vom Nutzer der SPS erstellt wird. Das SPS-Anwenderprogramm wird direkt aus der OpenPCS-Programmierumgebung heraus auf das Modul übertragen, hierzu ist keinerlei externe Zusatzsoftware erforderlich.

Ein Update der SPS-Firmware erfordert sowohl die Anmeldung an der Kommando-Shell des PLCcore-E660 wie in Abschnitt 7.8.1 beschrieben als auch die Anmeldung am FTP-Server gemäß Abschnitt 7.8.2.

Das Update der SPS-Firmware erfolgt durch ein selbst-extrahierendes Firmware-Archiv, das per FTP auf das PLCcore-E660 übertragen wird. Nach dem Start des FTP-Servers auf dem PLCcore-E660 (Kommando *"pureftp"*, siehe Abschnitt 7.8.2) kann das entsprechende Firmware-Archiv in das Verzeichnis *"/tmp"* des PLCcore-E660 übertragen werden (siehe Bild 22).

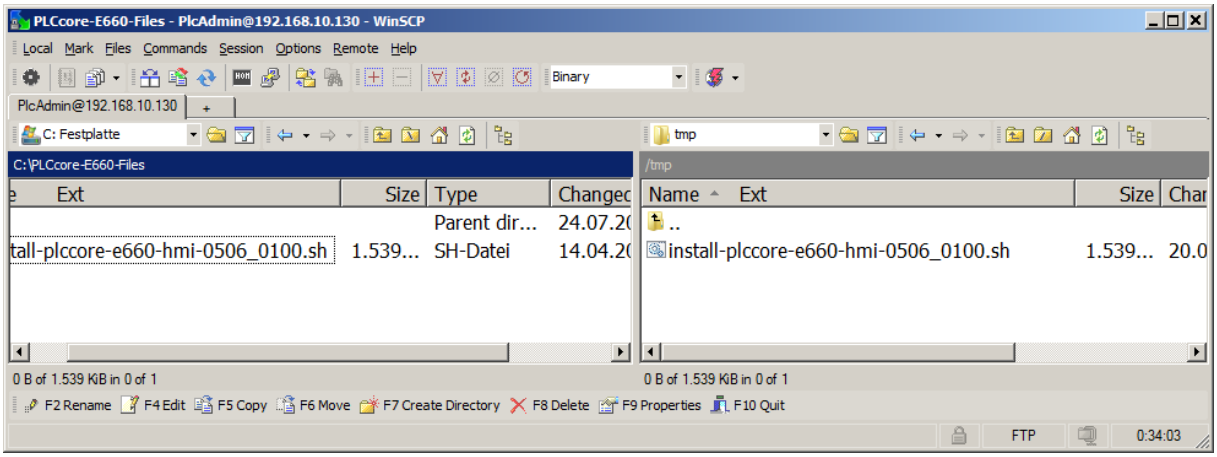


Bild 22: Dateitransfer im FTP-Client "WinSCP"

Wichtig: Für den FTP-Transfer des Firmware-Archives ist der Übertragungstyp *"Binary"* auszuwählen. Bei Verwendung des FTP-Clients *"WinSCP"* ist hierzu in der Menüleiste der entsprechende Transfermodus zu selektieren. Nach dem Download des Firmware-Archives ist zu kontrollieren, dass die übertragene Datei auf dem PLCcore-E660 exakt dieselbe Größe besitzt wie die Originaldatei auf dem PC (siehe Bild 22). Eine abweichende Größe deutet auf einen falschen Übertragungsmodus hin (z.B. *"Text"*). In diesem Fall ist der Download mit dem Übertragungstyp *"Binary"* zu wiederholen.

Nach dem Download des selbstextrahierenden Archives muss die SPS-Firmware auf dem PLCcore-E660 installiert werden. Dazu sind im Telnet-Fenster nachfolgende Befehle einzugeben. Hier ist zu beachten, dass der Dateiname für das Firmware-Archiv eine Versionskennung beinhaltet (z.B. *"install-plccore-E660-0506_0100.sh"* für die Version 5.06.01.00). Diese Nummer ist bei der Befehlseingabe entsprechend anzupassen:

```
cd  /tmp
chmod  +x  install-plccore-E660-0506_0100.sh
./install-plccore-e660-0506_0100.sh
```

Tipp: Die Kommando-Shell auf dem PLCcore-E660 kann angefangene Namen durch drücken der Tab-Taste automatisch vervollständigen ("tab completion"), so dass es in der Regel genügt, nur den Anfang des Dateinamens einzugeben und dann vom System automatisch ergänzen zu lassen. So wird z.B. *"./ins"* durch Drücken der Tab-Taste automatisch zu *"./install-plccore-E660-0506_0100.sh"* erweitert.

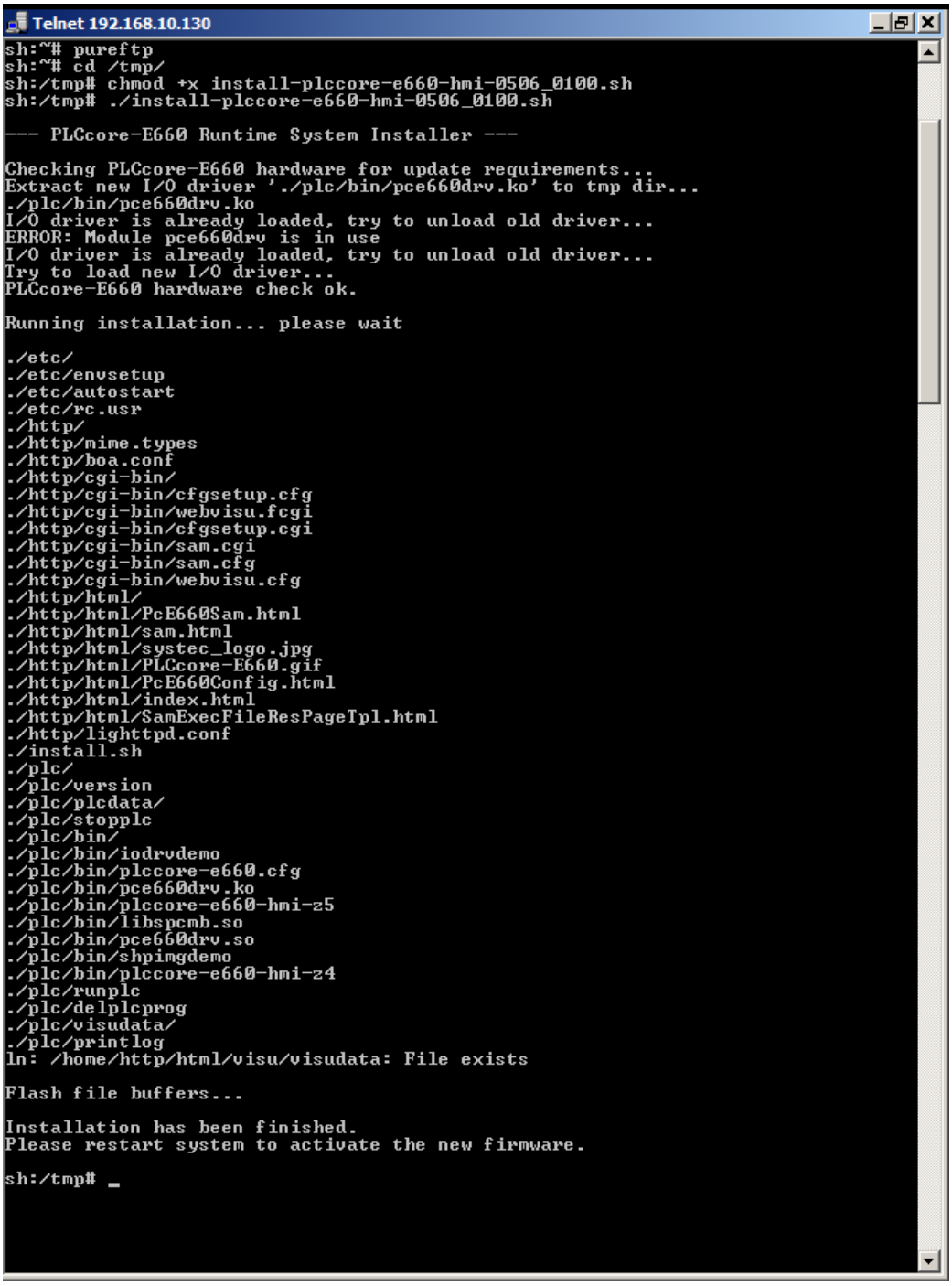


Bild 23: Installation der SPS-Firmware auf dem PLCcore-E660

Bild 23 verdeutlicht die Installation der SPS-Firmware auf dem PLCcore-E660. Nach einem Reset startet das Modul anschließend mit der aktualisierten Firmware.

Hinweis: **Bei einem Update der SPS-Firmware wird auch die Konfigurationsdatei *"/home/plc/bin/plccore-E660.cfg"* überschrieben, was zu einem Rücksetzen der SPS-Konfiguration auf Standardeinstellungen führt. Daher sollte nach einem Update die Konfiguration wie im Abschnitt 7.4 beschrieben kontrolliert und ggf. neu gesetzt werden.**

7.13.2 Update des Linux-Images

Der Update des Linux-Images ist in der Regel nicht notwendig. Sollte es doch notwendig werden, sei auf die Abschnitte 7.7 und 6.8 des Dokuments L-1554 System Manual ECUcore-E660 verwiesen.

Hinweis: **Wird ein neues SD-Karten-Image auf die SD-Karte geschrieben, gehen sowohl die SPS-Firmware als auch alle Daten im Verzeichnis /home/ verloren. Diese sind dann über die entsprechenden Installationen (siehe Abschnitt 7.13.1) zu restaurieren.**

8 Adaption von Ein-/Ausgängen sowie Prozessabbild

8.1 Datenaustausch über Shared Prozessabbild

8.1.1 Übersicht zum Shared Prozessabbild

Das PLCcore-E660 basiert auf Embedded Linux als Betriebssystem. Dadurch ist es möglich, simultan zur SPS-Firmware noch weitere, anwenderspezifische Programme abzuarbeiten. Über ein gemeinsam genutztes Prozessabbild (Shared Prozessabbild) können das SPS-Programm und eine anwenderspezifische C/C++ Applikation Daten miteinander austauschen. Voraussetzung für die Implementierung anwenderspezifischer C/C++ Applikationen ist das **Softwarepaket SO-1116 ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-E660")**.

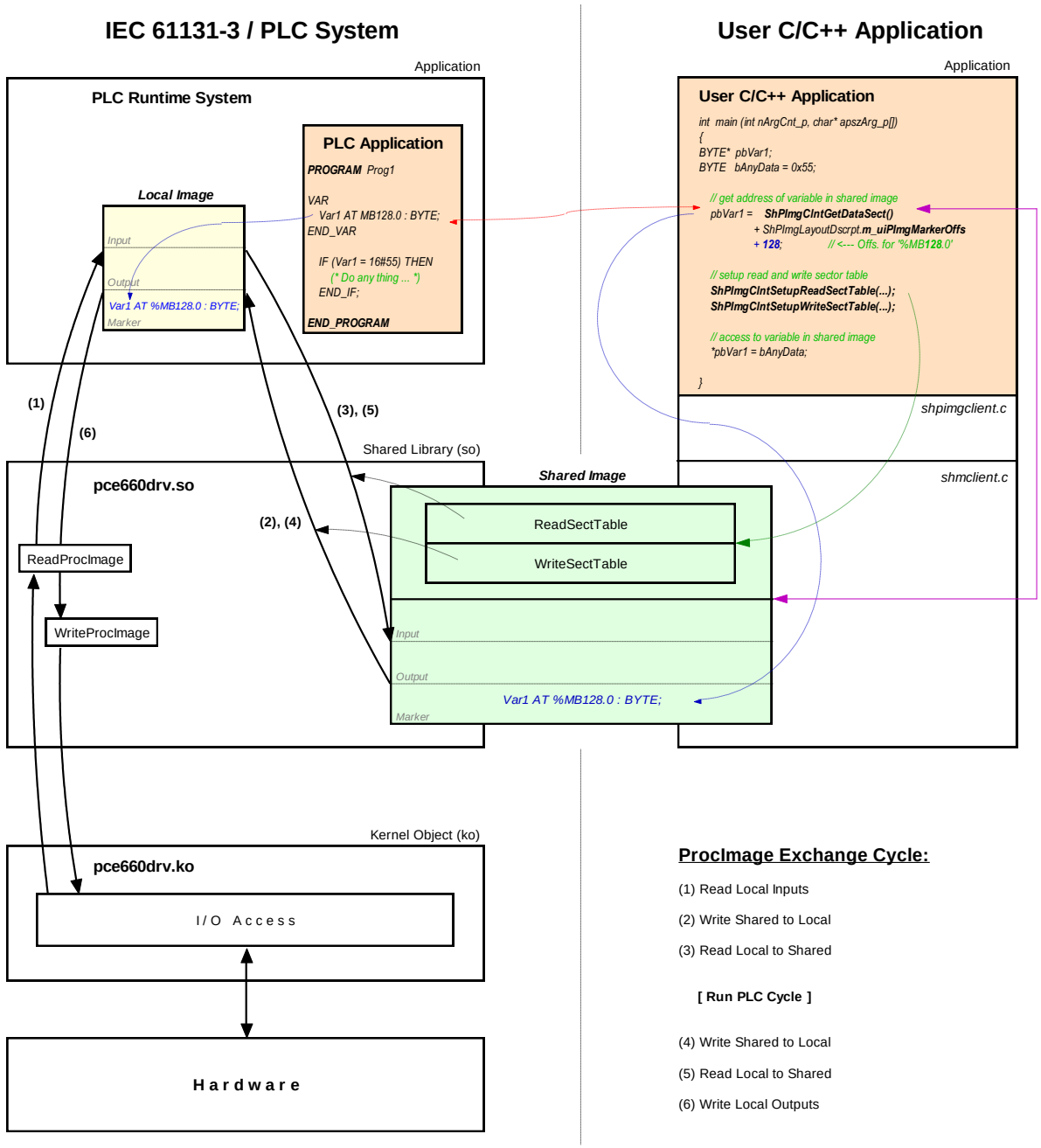


Bild 24: Übersicht Shared Prozessabbild

Für eine C/C++ Applikation sind alle Variablen über das Shared Prozessabbild nutzbar, die das SPS-Programm als direkt adressierte Variablen im Prozessabbild anlegt. Wie in Bild 24 dargestellt, werden für den Datenaustausch mit einer externen Applikation innerhalb des SPS-Laufzeitsystems zwei getrennte Prozessabbilder benutzt. Das ist notwendig, um die in der IEC 61131-3 festgelegte Forderung zu erfüllen, nach der sich das Eingangs-Prozessabbild einer SPS während der gesamten Ausführungszeit eines SPS-Programmzyklus nicht verändern darf. Das SPS-Programm operiert dabei stets mit dem internen, innerhalb des SPS-Laufzeitsystems lokal angelegten Prozessabbild ("Local Image" in Bild 24). Dieses ist durch das SPS-Laufzeitsystem gekapselt und somit vor direkten Zugriffen von außen geschützt. Die anwenderspezifische, externe C/C++ Applikation benutzt dagegen stets das Shared Prozessabbild ("Shared Image" in Bild 24). Durch die Aufsplittung in zwei getrennte Prozessabbilder wird zudem eine Entkopplung der Zugriffe zwischen SPS-Programm und externer Applikation erreicht. Eine Synchronisation der beiden parallel laufenden, unabhängigen Prozesse ist nur noch für die kurze Zeitspanne des Umkopierens der Prozessdaten notwendig.

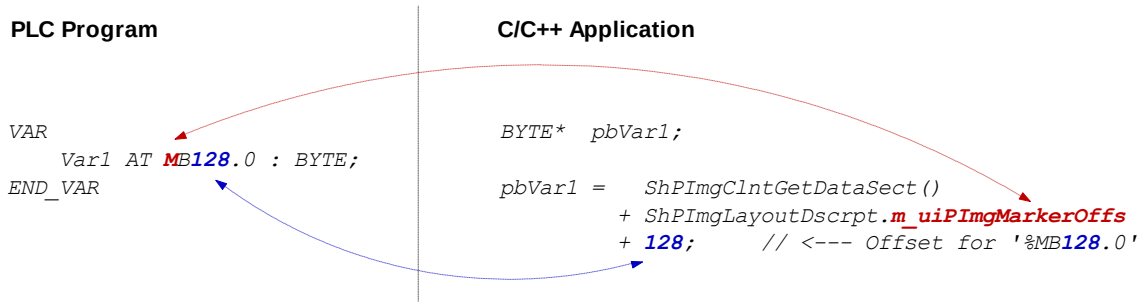
Um den Datenaustausch mit externen Applikationen zu ermöglichen, muss die **Option *"Share PLC process image"*** in der SPS-Konfiguration aktiviert sein (siehe Abschnitt 7.4.1). Alternativ kann auch der Eintrag *"EnableSharing="* in der Sektion *"[ProcImg]"* innerhalb der Konfigurationsdatei ***"/home/plc/bin/plccore-E660.cfg"*** direkt gesetzt werden (siehe Abschnitt 7.4.2). Die entsprechende Konfigurationseinstellung wird beim Starten der SPS-Firmware ausgewertet. Ist die Option *"Share PLC process image"* aktiviert, legt die SPS-Firmware ein zweites Prozessabbild als Shared Memory an ("Shared Image" in Bild 24), das explizit für den Datenaustausch mit externen Applikationen vorgesehen ist. Die SPS-Firmware agiert hier als Server, die externe, anwenderspezifische C/C++ Applikation übernimmt die Rolle des Clients.

Das Kopieren der Daten zwischen beiden Prozessabbildern wird über die ***ReadSectorTable*** und die ***WriteSectorTable*** gesteuert. Beide Tabellen werden durch den Client (externe, anwenderspezifische C/C++ Applikation) befüllt und vom Server (SPS-Laufzeitsystem) abgearbeitet. Der Client definiert hier die Bereiche des SPS-Prozessabbildes, aus denen er Daten lesen (*ReadSectorTable*) bzw. in die er Daten schreiben möchte (*WriteSectorTable*). Die Begriffe *"Read"* und *"Write"* beziehen sich somit auf die Datentransferrichtungen aus Sicht des Clients.

Die zu lesenden und zu schreibenden Sektionen können alle Bereiche des gesamten Prozessabbildes umfassen, also sowohl Input-, Output- als auch Merkerbereich. Damit ist es für eine Client-Applikation beispielsweise auch möglich, in den Input-Bereich des SPS-Prozessabbildes zu schreiben und Daten aus dem Output-Bereich zu lesen. Die Reihenfolge der einzelnen Lese- und Schreiboperationen ist in Bild 24 dargestellt. So werden vor der Ausführung eines SPS-Programmzyklus zunächst die physikalischen Eingänge in das lokale Prozessabbild der SPS eingelesen (1), anschließend erfolgt die Übernahme der in der *WriteSectorTable* definierten Bereiche aus dem Shared Prozessabbild in das lokale Prozessabbild (2). Unter Ausnutzung dieser Reihenfolge ist es einer Client-Applikation z.B. auch möglich, den Wert eines physikalischen Eingangs zu überschreiben, was sich sowohl für Simulationszwecke ausnutzen lässt als auch zum Festsetzen von Eingansdaten auf konstante Werte (*"Forcen"*). Analog dazu werden vor dem Schreiben des Prozessabbildes auf die physikalischen Ausgänge (6) zunächst wiederum die in der *WriteSectorTable* definierten Bereiche aus dem Shared Prozessabbild in das lokale Prozessabbild übernommen (4). Damit ist eine Client-Applikation gleichfalls in der Lage, auch die vom SPS-Programm generierten Ausgangsinformationen zu überschreiben.

Der **Aufbau des Prozessabbildes** wird von der jeweiligen SPS-Firmware fest vorgegeben. Die Client-Applikation erhält die Informationen zum Aufbau des Prozessabbildes über die Funktion ***ShPImgClntSetup()***. Diese trägt die Startoffsets und Größen von Input-, Output- und Merkerbereich in die beim Aufruf übergebene Struktur vom Typ *tShPImgLayoutDscrpt* ein. Die Startadresse des Shared Prozessabbildes liefert die Funktion ***ShPImgClntGetDataSect()***. Bei der Definition einer Variablen im SPS-Programm wird deren absolute Position im Prozessabbild durch Bereich (%I = Input, %Q = Output, %M = Merker) und Offset (z.B. %MB**128**.0) bestimmt. Dabei beginnt der Offset in jedem Bereich wieder mit Null, so dass beispielsweise das Anlegen einer Variablen im Merkerbereich unabhängig von den Größen der davor liegenden Input- und Output-Bereiche erfolgt. Für den Austausch von Daten zwischen SPS-Programm und externer Applikation ist ein korrespondierendes **Variablen-Paar** jeweils im SPS-Programm und in der C/C++ Applikation anzulegen. Dazu ist auf beiden Seiten dieselbe Adresse zu referenzieren. Um bei der Definition der entsprechenden Variablen

im C/C++ Programm ein zum SPS-Programm vergleichbares Adressierungsschema anwenden zu können, spiegelt die Struktur *tShPImgLayoutDscrpt* den physikalischen Aufbau des Prozessabbildes in der SPS-Firmware mit Input-, Output- und Merkerbereich wider. Damit erfolgt auch im C/C++ Programm die Definition einer Variablen im Shared Prozessabbild unter Angabe des jeweiligen Bereiches und des darauf bezogenen Offsets. Das nachfolgende Beispiel verdeutlich das Anlegen eines korrespondierenden Variablen-Paares in SPS-Programm und C/C++ Applikation:



Wie bereits weiter oben beschrieben wird der Kopiervorgang zum Austausch des Variableninhaltes zwischen SPS- und C/C++ Programm über ***ReadSectorTable*** und ***WriteSectorTable*** gesteuert. Um im dargestellten Beispiel den Wert der Variable von der C/C++ Applikation zum SPS-Programm zu übertragen, ist vom Client (C/C++ Applikation) ein entsprechender Eintrag in der *WriteSectorTable* zu definieren (*WriteSectorTable* da Client die Variable zum Server "schreibt"):

```
// 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);
```

Werden in der gleichen Transferrichtung mehrere Variablen-Paare für den Datenaustausch zwischen SPS-Programm und C/C++ Applikation angelegt, dann sollten diese nach Möglichkeit in einem zusammenhängenden Adressbereich definiert werden. Dadurch lassen sie sich in einem gemeinsamen Eintrag der entsprechenden SectorTable zusammenfassen. Als *SectorOffset* ist die Adresse der ersten Variable und als *SectorSize* die Summe der Variablengrößen anzugeben. Durch das Zusammenfassen wird die Effizienz und Performanz der Kopiervorgänge verbessert.

Für jeden Eintrag der *WriteSectorTable* ist außerdem ein entsprechender *SyncType* zu definieren. Dieser legt fest, ob die Sektion generell immer zwischen zwei aufeinander folgenden SPS-Zyklen vom Shared Prozessabbild in das lokale Abbild übernommen wird (***kShPImgSyncAlways***) oder nur bei Bedarf (***kShPImgSyncOnDemand***). Bei der Klassifizierung als *SyncOnDemand* werden die Daten nur dann kopiert, wenn die betreffende Sektion zuvor explizit als aktualisiert markiert wurde. Dies erfolgt durch den Aufruf der Funktion ***ShPImgClntWriteSectMarkNewData()*** unter Angabe des zugehörigen *WriteSectorTable*-Index (z.B. 0 für *WriteSectTab[0]* usw.).

Für die *ReadSectorTable* ist der *SyncType* fest als *kShPImgSyncAlways* vorgegeben (der Wert im Memberelement *m_SyncType* wird ignoriert). Da die SPS-Firmware nicht feststellen kann, welche Variablen das SPS-Programm im vorangegangenen Zyklus verändert hat, werden stets alle in der *ReadSectorTable* definierten Sektionen vom lokalen Abbild in das Shared Prozessabbild übernommen. Die betreffenden Variablen enthalten somit im Shared Prozessabbild immer die aktuellen Werte.

Das Shared Prozessabbild ist eine von der SPS-Firmware und der C/C++ Applikation gemeinsam benutzte Ressource. Um hier Konflikte durch einen zeitgleichen Zugriff beider parallel laufender Prozesse zu verhindern, ist das Shared Prozessabbild intern über eine Semaphore geschützt. Benötigt ein Prozess Zugriff auf das Shared Prozessabbild, betritt er einen kritischen Abschnitt, indem

er zunächst die Semaphore setzt und so den exklusiven Zugriff auf diese Ressource erhält. Möchte der andere Prozess zur selben Zeit ebenfalls auf das Shared Prozessabbild zugreifen, muss er genauso wie der erste Prozess auch einen kritischen Abschnitt betreten, indem er wiederum zunächst versucht, die Semaphore zu setzen. In diesem Fall erkennt das Betriebssystem, dass die Ressource bereits in Benutzung ist. Es blockiert den zweiten Prozess nun so lange, bis der erste Prozess den kritischen Abschnitt wieder verlassen hat und die Semaphore freigibt. Das Betriebssystem stellt dadurch sicher, dass sich immer nur einer der beiden parallel laufenden Prozesse SPS-Laufzeitsystem und C/C++ Applikation im kritischen Abschnitt befinden kann und Zugriff auf das Shared Prozessabbild erhält. Damit sich die beiden Prozesse gegenseitig möglichst wenig in ihrer Abarbeitung behindern, sollten die kritischen Abschnitte so selten wie möglich und dann auch nur so lange wie unbedingt nötig durchlaufen werden. Andernfalls kann es zur Verlängerung der SPS-Zykluszeit sowie zu Laufzeitschwankungen (Jitter) kommen.

Zum Setzen der Semaphore und damit zum Sperren des Shared Prozessabbildes für den exklusiven Zugriff stehen der Client-Applikation die beiden Funktionen ***ShPImgClntLockSegment()*** zum Betreten sowie ***ShPImgClntUnlockSegment()*** zum Verlassen des kritischen Abschnitts zur Verfügung. Der Abschnitt zwischen den beiden Funktionen wird auch als geschützter Bereich bezeichnet, in dem die Client-Applikation konkurrenzfreien Zugriff auf das Shared Prozessabbild besitzt. Nur innerhalb eines solchen geschützten Bereiches ist die Konsistenz gelesener bzw. geschriebener Daten gewährleistet. Außerhalb davon kann jederzeit eine Manipulation des Prozessabbildes durch das SPS-Laufzeitsystem erfolgen. Das nachfolgende Beispiel verdeutlicht den exklusiven Zugriff auf das Shared Prozessabbild in der C/C++ Applikation:

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

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

Im angegebenen Beispiel wurde beim Anlegen des *WriteSectorTable*-Eintrages der *SyncType* als *kShPImgSyncOnDemand* definiert. Somit erfolgt eine Übernahme der Variable *Var1* nur dann vom Shared Prozessabbild in das lokale Abbild, wenn die betreffende Sektion zuvor explizit als aktualisiert markiert wurde. Dazu ist der Aufruf der Funktion ***ShPImgClntWriteSectMarkNewData()*** erforderlich. Da die Funktion *ShPImgClntWriteSectMarkNewData()* die Semaphore nicht verändert, darf sie, wie hier im Beispiel dargestellt, nur in einem geschützten Bereich, also im Code-Abschnitt zwischen *ShPImgClntLockSegment()* und *ShPImgClntUnlockSegment()*, benutzt werden.

Die Synchronisation zwischen lokalem Abbild und Shared Prozessabbild erfolgt durch das SPS-Laufzeitsystem nur zwischen zwei aufeinander folgenden SPS-Zyklen. Einer Client-Applikation (anwenderspezifisches C/C++ Programm) ist dieser Zeitpunkt nicht unmittelbar bekannt, sie kann sich jedoch vom SPS-Laufzeitsystem über die Aktualisierung des Shared Prozessabbild informieren lassen. Dazu muss sie einen Callback-Handler vom Typ *tShPImgAppNewDataSigHandler* definieren, z.B.:

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

Dieser Callback-Handler ist mit Hilfe der Funktion ***ShPImgClntSetNewDataSigHandler()*** zu registrieren. Er wird jeweils im Anschluss an eine Synchronisation der beiden Abbilder aufgerufen.

Der **Callback-Handler der Client-Applikation wird im Kontext eines Linux Signal-Handlers gerufen** (das SPS-Laufzeitsystem informiert den Client mit Hilfe der Linux-Funktion *kill()*).

Dementsprechend gelten für den Callback-Handler der Client-Applikation die üblichen **Restriktionen** für Linux Signal-Handler. Insbesondere ist hier nur der Aufruf einiger weniger, explizit als reentranz-fest gekennzeichneter Betriebsystem-Funktionen erlaubt. Innerhalb der Client-Applikation muss ebenfalls darauf geachtet werden, dass es nicht zum reentranten Aufruf lokaler Funktionen kommt. Daher sollte, wie im Beispiel gezeigt, innerhalb des Callback-Handlers lediglich ein globales Flag zur Signalisierung gesetzt werden, das dann später in der Hauptschleife der Client-Applikation ausgewertet und verarbeitet wird.

8.1.2 API des Shared Prozessabbild Client

Wie im Bild 24 dargestellt, benutzt eine anwenderspezifische C/C++ Applikation ausschließlich das vom *Shared Process Image Client* zur Verfügung gestellte API (Application Programming Interface). Dieses API ist im Headerfile *shpimgclient.h* deklariert und im Sourcefile *shpimgclient.c* implementiert. Es umfasst folgende Typen (teilweise auch in *shpimg.h* definiert) und Funktionen:

Struktur *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;
```

Die Struktur ***tShPImgLayoutDscrpt*** beschreibt den von der SPS-Firmware vorgegebenen Aufbau des Prozessabbildes. Die Client-Applikation erhält die Informationen zum Aufbau des Prozessabbildes über die Funktion *ShPImgClntSetup()*. Diese trägt die Startoffsets und Größen von Input-, Output- und Merkerbereich in die beim Aufruf übergebene Struktur ein.

Struktur *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;
```

Die Struktur ***tShPImgSectDscrpt*** beschreibt den vom Client zu definierenden Aufbau eines *ReadSectorTable*- oder *WriteSectorTable*-Eintrages. Beide Tabellen dienen zur Synchronisation zwischen lokalem Abbild des SPS-Laufzeitsystems und Shared Prozessabbild (siehe Abschnitt 8.1.1). Das Memberelement *m_uiPImgDataSectOffs* definiert den absoluten Startoffset der Sektion innerhalb des Shared Prozessabbildes. Die jeweiligen Startoffsets von Input-, Output- und Merkerbereich können dazu aus der Struktur *tShPImgLayoutDscrpt* ermittelt werden. Das Memberelement *m_uiPImgDataSectSize* legt die Größe der Sektion fest, die eine oder mehrere Variablen enthalten kann. Das Memberelement *m_SyncType* ist nur für Einträge der WriteSectorTable relevant und legt fest, ob die Sektion generell immer zwischen zwei aufeinander folgenden SPS-Zyklen vom Shared Prozessabbild in das lokale Abbild übernommen wird (***kShPImgSyncAlways***) oder nur bei Bedarf

(***kShPImgSyncOnDemand***). Bei der Klassifizierung als *SyncOnDemand* müssen die Daten durch den Aufruf der Funktion *ShPImgClntWriteSectMarkNewData()* als modifiziert gekennzeichnet werden. Sie setzt dazu das Memberelement *m_fNewData* auf TRUE. Die Client-Applikation sollte dieses Memberelement niemals direkt manipulieren.

Funktion *ShPImgClntSetup*

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

Die Funktion ***ShPImgClntSetup()*** initialisiert den *Shared Process Image Client* und verbindet sich mit dem vom SPS-Laufzeitsystem angelegten Speichersegment für das Shared Prozessabbild. Anschließend trägt sie die Startoffsets und Größen von Input-, Output- und Merkerbereich in die beim Aufruf übergebene Struktur vom Typ *tShPImgLayoutDscrpt* ein. Damit erhält die Client-Applikation Kenntnis über den Aufbau des von der SPS-Firmware verwalteten Prozessabbildes.

Ist zum Aufrufzeitpunkt der Funktion kein SPS-Laufzeitsystem aktiv oder hat dieses kein Shared Prozessabbild angelegt (Option *"Share PLC process image"* in der SPS-Konfiguration deaktiviert, siehe Abschnitt 8.1.1), kehrt die Funktion mit dem Returnwert FALSE zurück. Bei erfolgreichem Abschluss der Initialisierung ist der Rückgabewert TRUE.

Funktion *ShPImgClntRelease*

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

Die Funktion ***ShPImgClntRelease()*** beendet den *Shared Process Image Client* und trennt die Verbindung zu dem vom SPS-Laufzeitsystem angelegten Speichersegment für das Shared Prozessabbild.

Nach erfolgreicher Ausführung liefert die Funktion den Rückgabewert TRUE, im Fehlerfall ist der Returnwert auf FALSE gesetzt.

Funktion *ShPImgClntSetNewDataSigHandler*

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

Die Funktion ***ShPImgClntSetNewDataSigHandler()*** registriert einen applikationsspezifischen Callback-Handler. Dieser Callback-Handler wird im Anschluss an eine Synchronisation der beiden Abbilder aufgerufen. Durch Übergabe von NULL wird ein registrierter Callback-Handler wieder abgemeldet.

Der **Callback-Handler wird im Kontext eines Linux Signal-Handlers gerufen**. Dementsprechend gelten die üblichen **Restriktionen** für Linux Signal-Handler (siehe Abschnitt 8.1.1).

Nach erfolgreicher Ausführung liefert die Funktion den Rückgabewert TRUE, im Fehlerfall ist der Returnwert auf FALSE gesetzt.

Funktion *ShPImgCIntGetHeader*

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

Die Funktion ***ShPImgCIntGetHeader()*** liefert einen Pointer auf die zur Verwaltung des Shared Prozessabbild intern verwendete Struktur vom Typ *tShPImgHeader*. Diese Struktur wird von der Client-Applikation normalerweise nicht benötigt, da alle darin enthaltenen Daten über Funktionen des vom *Shared Process Image Client* zur Verfügung gestellten API gelesen und geschrieben werden können.

Funktion *ShPImgCIntGetDataSect*

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

Die Funktion ***ShPImgCIntGetDataSect()*** liefert einen Pointer auf den Anfang des Shared Prozessabbildes. Dieser Pointer stellt die Basisadresse für alle Zugriffe auf das Shared Prozessabbild, einschließlich der Definition von Sektionen der *ReadSectorTable* und *WriteSectorTable* dar (siehe Abschnitt 8.1.1).

Funktionen *ShPImgCIntLockSegment* und *ShPImgCIntUnlockSegment*

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

Für den exklusiven Zugriff auf das Shared Prozessabbild stehen der Client-Applikation die beiden Funktionen ***ShPImgCIntLockSegment()*** zum Betreten sowie als Komplement ***ShPImgCIntUnlockSegment()*** zum Verlassen des kritischen Abschnitts zur Verfügung. Der Abschnitt zwischen den beiden Funktionen ist ein geschützter Bereich, in dem die Client-Applikation konkurrenzfreien Zugriff auf das Shared Prozessabbild besitzt (siehe Abschnitt 8.1.1). Nur innerhalb eines solchen geschützten Bereiches ist die Konsistenz gelesener bzw. geschriebener Daten gewährleistet. Außerhalb davon kann jederzeit eine Manipulation des Prozessabbildes durch das SPS-Laufzeitsystem erfolgen. Damit die Client-Applikation das SPS-Laufzeitsystem in seiner Abarbeitung so wenig wie möglich behindert, sollten die kritischen Abschnitte so selten wie möglich und dann auch nur so lange wie unbedingt nötig gesetzt werden. Andernfalls kann es zur Verlängerung der SPS-Zykluszeit sowie zu Laufzeitschwankungen (Jitter) kommen.

Nach erfolgreicher Ausführung liefert die Funktion den Rückgabewert TRUE, im Fehlerfall ist der Returnwert auf FALSE gesetzt.

Funktion *ShPImgCIntSetupReadSectTable*

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

Die Funktion ***ShPImgCIntSetupReadSectTable()*** initialisiert die *ReadSectorTable* mit den vom Client definierten Werten. Der Client legt hier die Bereiche des SPS-Prozessabbildes fest, aus denen er Daten lesen möchte (siehe Abschnitt 8.1.1). Als Parameter *paShPImgReadSectTab_p* ist die Anfangsadresse eines Feldes aus Elementen der Struktur *tShPImgSectDscrpt* zu übergeben. Der Parameter *uiNumOfReadDscrptUsed_p* gibt an, wie viele Elemente das Feld besitzt.

Für die *ReadSectorTable* ist der *SyncType* fest als *kShPImgSyncAlways* vorgegeben.

Die maximale Anzahl möglicher Elemente für die *ReadSectorTable* ist durch die Konstante

SHPIMG_READ_SECT_TAB_ENTRIES definiert und kann nur verändert werden, wenn gleichzeitig auch die Shared Library *"pce660drv.so"* neu generiert wird (setzt SO-1117 - "Driver Development Kit für das ECUcore-E660" voraus, siehe Abschnitt 8.2).

Nach erfolgreicher Ausführung liefert die Funktion den Rückgabewert TRUE, im Fehlerfall ist der Returnwert auf FALSE gesetzt.

Funktion *ShPImgClntSetupWriteSectTable*

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

Die Funktion ***ShPImgClntSetupWriteSectTable***() initialisiert die *WriteSectorTable* mit den vom Client definierten Werten. Der Client legt hier die Bereiche des SPS-Prozessabbildes fest, in die er Daten schreiben möchte (siehe Abschnitt 8.1.1). Als Parameter *paShPImgWriteSectTab_p* ist die Anfangsadresse eines Feldes aus Elementen der Struktur *tShPImgSectDscrpt* zu übergeben. Der Parameter *uiNumOfWriteDscrptUsed_p* gibt an, wie viele Elemente das Feld besitzt.

In der *WriteSectorTable* ist für jeden Eintrag der *SyncType* zu definieren. Dieser legt fest, ob die Sektion immer zwischen zwei SPS-Zyklen vom Shared Prozessabbild in das lokale Abbild übernommen wird (***kShPImgSyncAlways***) oder nur bei Bedarf (***kShPImgSyncOnDemand***), wenn die betreffende Sektion durch den Aufruf von *ShPImgClntWriteSectMarkNewData()* explizit als aktualisiert markiert wurde.

Die maximale Anzahl möglicher Elemente für die *WriteSectorTable* ist durch die Konstante *SHPIMG_WRITE_SECT_TAB_ENTRIES* definiert und kann nur verändert werden, wenn gleichzeitig auch die Shared Library *"pce660drv.so"* neu generiert wird (setzt SO-1103 - "Driver Development Kit für das ECUcore-E660" voraus, siehe Abschnitt 8.2).

Nach erfolgreicher Ausführung liefert die Funktion den Rückgabewert TRUE, im Fehlerfall ist der Returnwert auf FALSE gesetzt.

Funktion *ShPImgClntWriteSectMarkNewData*

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

Die Funktion ***ShPImgClntWriteSectMarkNewData()*** markiert den Inhalt einer durch die *WriteSectorTable* verwalteten Sektion als modifiziert. Sie wird benutzt, um für Sektionen mit *SyncType* ***kShPImgSyncOnDemand*** das Kopieren der Daten vom Shared Prozessabbild in das lokale Abbild der SPS zu veranlassen.

Da die Funktion *ShPImgClntWriteSectMarkNewData()* direkt auf den Header des Shared Prozessabbildes zugreift, ohne zuvor die Semaphore zu setzen, darf sie nur innerhalb eines geschützten Bereiches, also im Code-Abschnitt zwischen *ShPImgClntLockSegment()* und *ShPImgClntUnlockSegment()*, benutzt werden.

Nach erfolgreicher Ausführung liefert die Funktion den Rückgabewert TRUE, im Fehlerfall ist der Returnwert auf FALSE gesetzt.

8.1.3 Erstellen einer anwenderspezifischen Client Applikation

Voraussetzung für die Implementierung anwenderspezifischer C/C++ Applikationen ist das **Softwarepaket SO-1116 ("VMware-Image des Linux-Entwicklungssystems")**. Diese beinhaltet ein

komplett eingerichtetes Linux-Entwicklungssystem in Form eines VMware-Images und ermöglicht so einen leichten Einstieg in die Softwareentwicklung unter C/C++ für das PLCcore-E660. Das VMware-Image ist damit die ideale Voraussetzung, um Linux-basierte Anwenderprogramme auf demselben Host-PC zu entwickeln, auf dem auch schon das IEC 61131-Programmiersystem *OpenPCS* installiert ist. Neben der GNU-Crosscompiler Toolchain für x86-Prozessoren sind im VMware-Image des Linux-Entwicklungssystems bereits verschiede, für eine effektive Softwareentwicklung essentielle Server-Dienste vorkonfiguriert und sofort nutzbar. Details zum VMware-Image des Linux-Entwicklungssystems sowie dessen Nutzung beschreibt das *"System Manual ECUcore-E660"* (Manual-Nr.: L-1554).

Wie im Bild 24 dargestellt, benutzt eine anwenderspezifische C/C++ Applikation das vom *Shared Process Image Client* zur Verfügung gestellte API (Files *shpimgclient.c* und *shpimgclient.h*). Der *Shared Process Image Client* wiederum setzt auf den vom *Shared Memory Client* bereitgestellten Diensten auf (Files *shmclient.c* und *shmclient.c*). Beide Client-Implementierungen sind zur Generierung einer anwenderspezifischen C/C++ Applikation notwendig. Die entsprechenden Files sind im Archiv des *Shared Process Image Demos* (***shpimgdemo.tar.gz***) enthalten. Zur Erstellung eigener anwenderspezifischen Client Applikationen wird dringend empfohlen, dieses Demoprojekt als Ausgangspunkt für eigene Anpassungen und Erweiterungen zu verwenden. Des Weiteren beinhaltet das Demoprojekt ein Makefile mit allen relevanten Konfigurationseinstellungen, die zum Erstellen einer auf dem PLCcore-E660 lauffähigen Linux-Applikation notwendig sind. Tabelle 19 listet die im Archiv *"shpimgdemo.tar.gz"* enthaltenen Files auf und klassifiziert diese als allgemeingültigen Bestandteil einer C/C++ Applikation bzw. als spezifische Komponente für das Demoprojekt *"shpimgdemo"*.

Tabelle 19: Inhalt des Archiv-Files "shpimgdemo.tar.gz"

File	Erforderlich für alle C/C++ Applikationen	Spezifisch für Demo ***"shpimgdemo"***
shpimgclient.c	x	
shpimgclient.h	x	
shmclient.c	x	
shmclient.h	x	
shpimg.h	x	
global.h	x	
Makefile	als Vorlage, ist anzupassen	
shpimgdemo.c		x
trmio.c		x
trmio.h		x
trace.c		x

Das Archiv-File ***"shpimgdemo.tar.gz"*** mit dem *Shared Process Image Demo* ist innerhalb des Linux-Entwicklungssystems in ein beliebiges Unterverzeichnis im Pfad *"/projects/ECUcore-E660/user"* zu entpacken. Dazu ist das Kommando *"tar"* wie folgt aufzurufen:

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

Das Kommando *"tar"* legt beim Entpacken selbständig das Unterverzeichnis *"shpimgdemo"* an. Wird es beispielsweise im Verzeichnis *"/projects/ECUcore-E660/user"* aufgerufen, so entpackt es die im Archiv enthaltenen Files in den Pfad *"/projects/ECUcore-E660/user/**shpimgdemo**"*. Bild 25 veranschaulicht das Entpacken von *"shpimgdemo.tar.gz"* innerhalb des Linux-Entwicklungssystems.

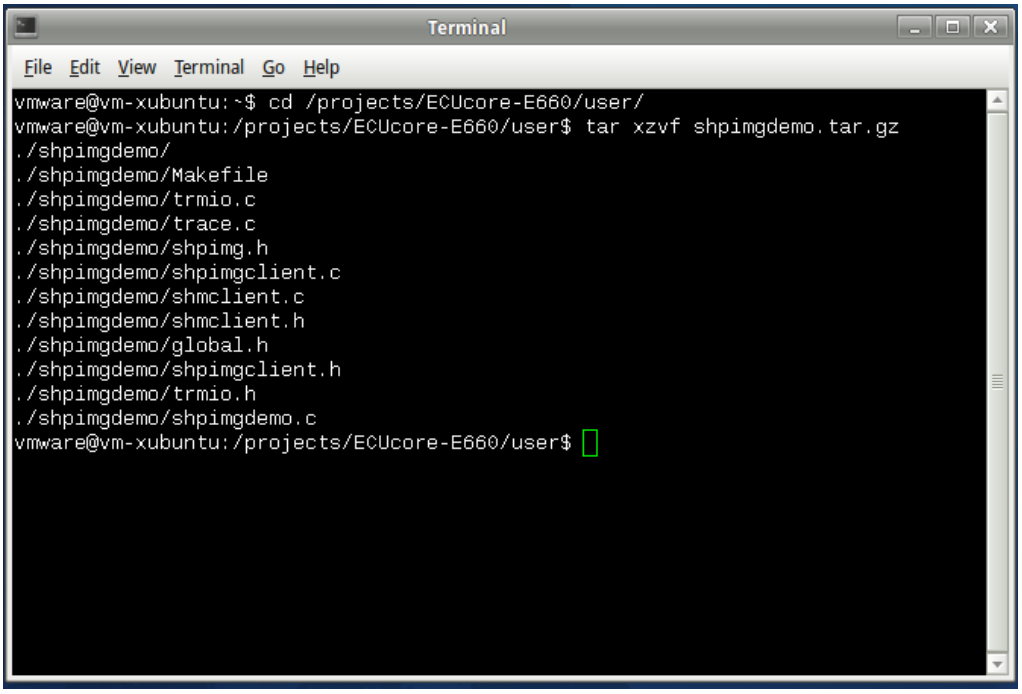


Bild 25: Entpacken des Archiv-Files shpimgdemo.tar.gz im Linux-Entwicklungssystem

Nach dem Entpacken und Wechseln in das Unterverzeichnis *"shpimgdemo"* kann das Demoprojekt durch Aufruf des Kommandos *"make"* erstellt werden:

```
cd  shpimgdemo
make
```

Bild 26 zeigt die Generierung des Demoprojektes *"shpimgdemo"* im Linux-Entwicklungssystem.

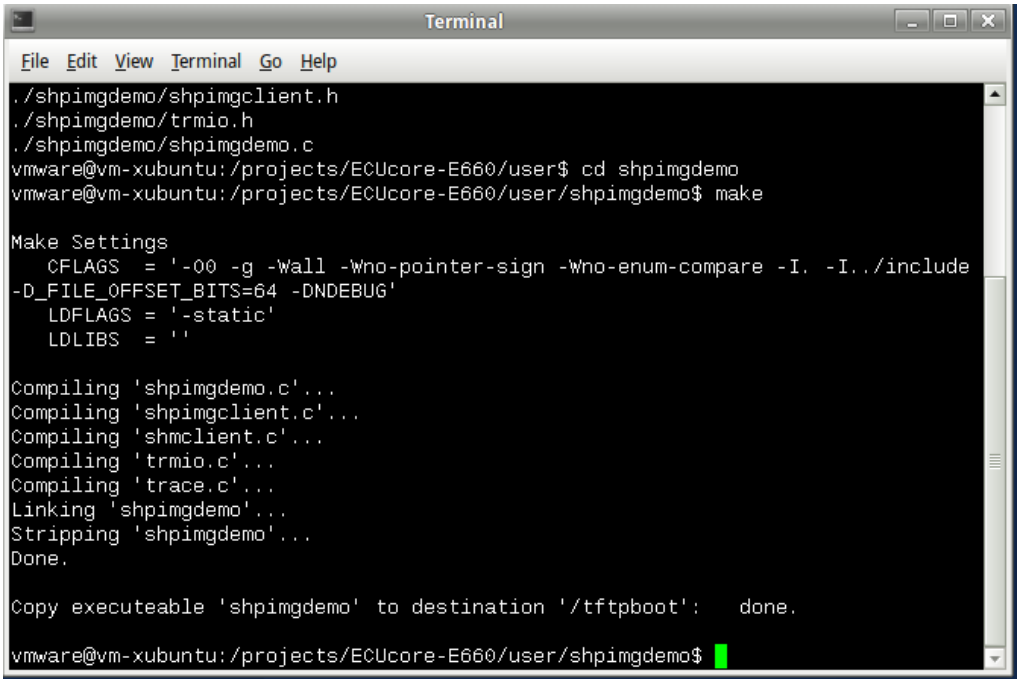


Bild 26: Generierung des Demoprojektes "shpimgdemo" im Linux-Entwicklungssystem

Die Anwendung und Bedienung des Demoprojektes *"shpimgdemo"* auf dem PLCcore-E660 beschreibt Abschnitt 8.1.4.

8.1.4 Beispiel zur Nutzung des Shared Prozessabbild

Als Beispiel zum Datenaustausch zwischen einem SPS-Programm und einer anwenderspezifischen C/C++ Applikation dient das im Abschnitt 8.1.3 beschriebene Demoprojekt *"shpimgdemo"* in Verbindung mit dem SPS-Programmbeispiel *"RunLight"*.

Technischer Hintergrund

Für eine C/C++ Applikation sind alle Variablen über das Shared Prozessabbild nutzbar, die das SPS-Programm als direkt adressierte Variablen im Prozessabbild anlegt. Im SPS-Programmbeispiel *"RunLight"* sind dies folgende Variablen:

```
(* variables for local control via on-board I/O's *)
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                      *)
```

Um von einer C/C++ Applikation über das Shared Prozessabbild auf die Variablen des SPS-Programms zugreifen zu können, müssen zum Einen entsprechende Sektionen für die *ReadSectorTable* und die *WriteSectorTable* angelegt werden und zum anderen ist die Definition von Pointern für den Zugriff auf die Variablen notwendig. Der nachfolgende Programmausschnitt verdeutlicht dies am Beispiel von *"shpimgdemo.c"*. Die Funktion *ShPImgClntSetup()* trägt die Startoffsets von Input-, Output- und Merkerbereich in die übergebene Struktur *ShPImgLayoutDscrpt* ein. Zusammen mit der von *ShPImgClntGetDataSect()* gelieferten Anfangsadresse lassen sich so die absoluten Anfangsadressen der einzelnen Bereiche innerhalb des Shared Prozessabbildes bestimmen. Um die Adresse einer bestimmten Variablen zu ermitteln, ist noch ihr entsprechender Offset innerhalb der jeweiligen Sektion zu addieren. So ergibt sich beispielsweise die absolute Adresse für den Zugriff auf die Variable *"bRemoteDirCtrl AT %MB515.0 : BYTE;"* als Summe aus der Anfangsadresse des Shared Prozessabbildes (pabShPImgDataSect), dem Startoffset des Merkerbereiches (*ShPImgLayoutDscrpt.m_uiPImgMarkerOffs* für *"%M…"*) sowie der im SPS-Programm definierten direkten Adresse innerhalb des Merkerbereiches (515 für "%MB515.0"):

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

Der nachfolgende Code-Ausschnitt zeigt die vollständige Definition aller im Demoprojekt verwendeten Variablen für den Datenaustausch mit dem SPS-Programm:

```
// ---- 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");
}
```

Ausführung auf dem PLCcore-E660

Um das *Shared Process Image Demo* auch ohne vorherige Einarbeitung in die Linux-basierte C/C++ Programmierung für das PLCcore-E660 ausprobieren zu können, wurde eine bereits fertig übersetzte und lauffähige Version des Programms zusammen mit der SPS-Firmware auf dem Modul installiert (*"/home/plc/bin/shpimgdemo"*). Die folgende Beschreibung bezieht sich auf diese Programmversion. Alternativ kann das Demoprojekt auch aus den entsprechenden Source-Files neu erstellt (siehe Abschnitt 8.1.3) und anschließend gestartet werden.

Zur Ausführung des *Shared Process Image Demos* auf dem PLCcore-E660 sind folgende Schritte notwendig:

1. **Aktivieren der Option *"Share PLC process image"*** in der SPS-Konfiguration (siehe Abschnitte 8.1.1 bzw. 7.4.1 und 7.4.2).

2. Öffnen des SPS-Programmbeispiels *"RunLight"* im IEC 61131-Programmiersystem *OpenPCS* und Übersetzen des Projektes für eine Zielhardware vom Typ *"SYSTEC - PLCcore-E660"*

3. Auswahl der Netzwerkverbindung zum PLCcore-E660 und Download des Programms

4. Starten des SPS-Programms auf dem PLCcore-E660

5. Anmeldung an der Kommando-Shell des PLCcore-E660 wie in Abschnitt 7.8.1 beschrieben

6. Wechseln in das Verzeichnis *"/home/plc/bin"* und Aufruf des Demoprogramms *"shpimgdemo"*:

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

   Auf dem PLCcore-E660 werden die digitalen Ausgänge als Lauflicht angesteuert. Mit Hilfe der Taster S704 (DI0) und S705 (DI1) kann die Laufrichtung umgeschaltet werden. Nach dem Start des Demoprogramms *"shpimgdemo"* auf dem PLCcore-E660 werden im Terminal zyklisch die aktuellen Statusinformationen zum Lauflicht angezeigt (siehe Bild 27).

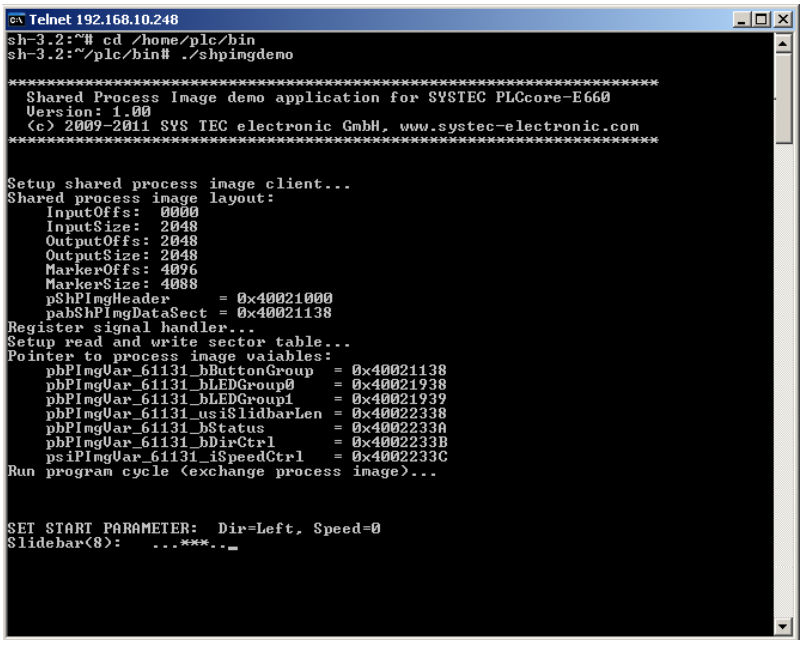


Bild 27: Terminalausgaben des Demoprogramms "shpimgdemo" nach dem Start

7. Nach Drücken des Tasters S707 (DI3) wird die Richtungs- und Geschwindigkeitskontrolle des Lauflichtes an das Demoprogramm *"shpimgdemo"* abgegeben. Im Terminalfenster kann nun mit

den Cursor-Tasten links und rechts (← und →) die Laufrichtung sowie mit den Cursor-Tasten auf und ab (↑ und ↓) die Geschwindigkeit des Lauflichtes durch die C-Applikation festgelegt werden.

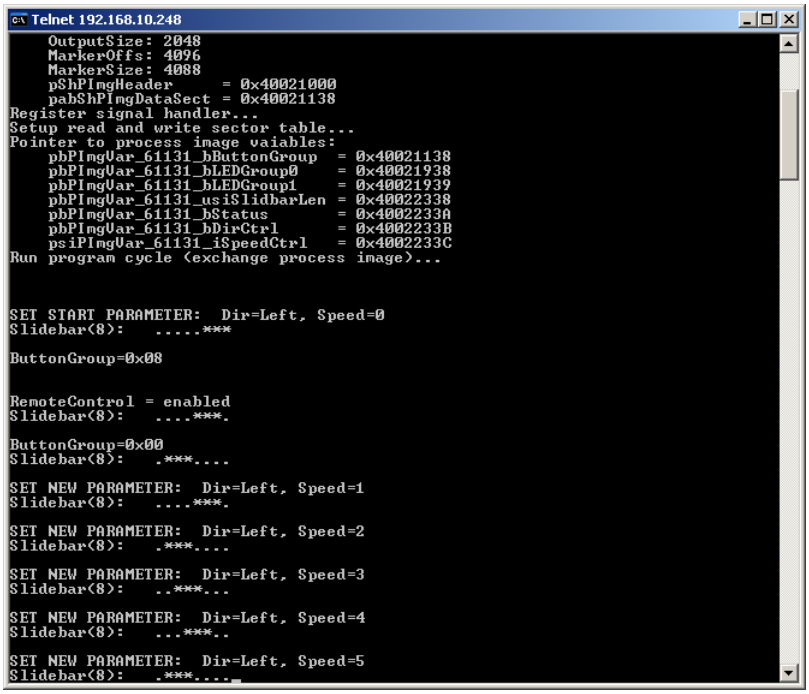


Bild 28: Terminalausgaben des Demoprogramms "shpimgdemo" nach Benutzereingaben

Bild 28 zeigt die Terminalausgaben des Demoprogramms *"shpimgdemo"* als Reaktion auf die Betätigung der Cursor-Tasten.

Das Demoprogramm *"shpimgdemo"* lässt sich durch Drücken von "Ctrl+C" im Terminalfenster beenden.

8.2 PlDriver Development Kit (DDK) für das PLCcore-E660

Das Driver Development Kit (DDK) für das ECUcore-E660 (bzw. PLCcore-E660) wird als zusätzliches Softwarepaket mit der Artikelnummer SO-1117 vertrieben. Es ist nicht im Lieferumfang des PLCcore-E660 bzw. dem Development Kit PLCcore-E660 enthalten. Details zum DDK beschreibt das "Software Manual Driver Development Kit für ECUcore-E660" (Manual-Nr.: L-1561).

Das Driver Development Kit für das ECUcore-E660 (bzw. PLCcore-E660) ermöglicht dem Anwender die eigenständige Anpassung der I/O-Ebene an ein selbst entwickeltes Baseboard. Das auf dem PLCcore-E660 eingesetzte Embedded Linux unterstützt das dynamische Laden von Treibern zur Laufzeit und ermöglicht so die Trennung von SPS-Laufzeitsystem und I/O-Treiber. Damit ist der Anwender in der Lage, den I/O-Treiber komplett an die eigenen Bedürfnisse anzupassen, ohne dazu das SPS-Laufzeitsystem selbst verändern zu müssen.

Mit Hilfe des DDK können folgende Ressourcen in die I/O-Ebene einbezogen werden:

- Peripherie (in der Regel GPIO) des Intel Atom Prozessors
- Adress-/Datenbus (memory-mapped Peripherie)
- SPI-Bus und I^2C-Bus
- alle anderen vom Betriebssystem bereitgestellten Ressourcen wie z.B. Filesystem und TCP/IP

Bild 29 vermittelt einen Überblick über die DDK-Struktur und die enthaltenen Komponenten. Das DDK beinhaltet unter anderem die Sourcen des Linux Kernel-Treibers (*pce660drv.ko*) und der Linux User-Bibliothek (*pce660drv.so*).

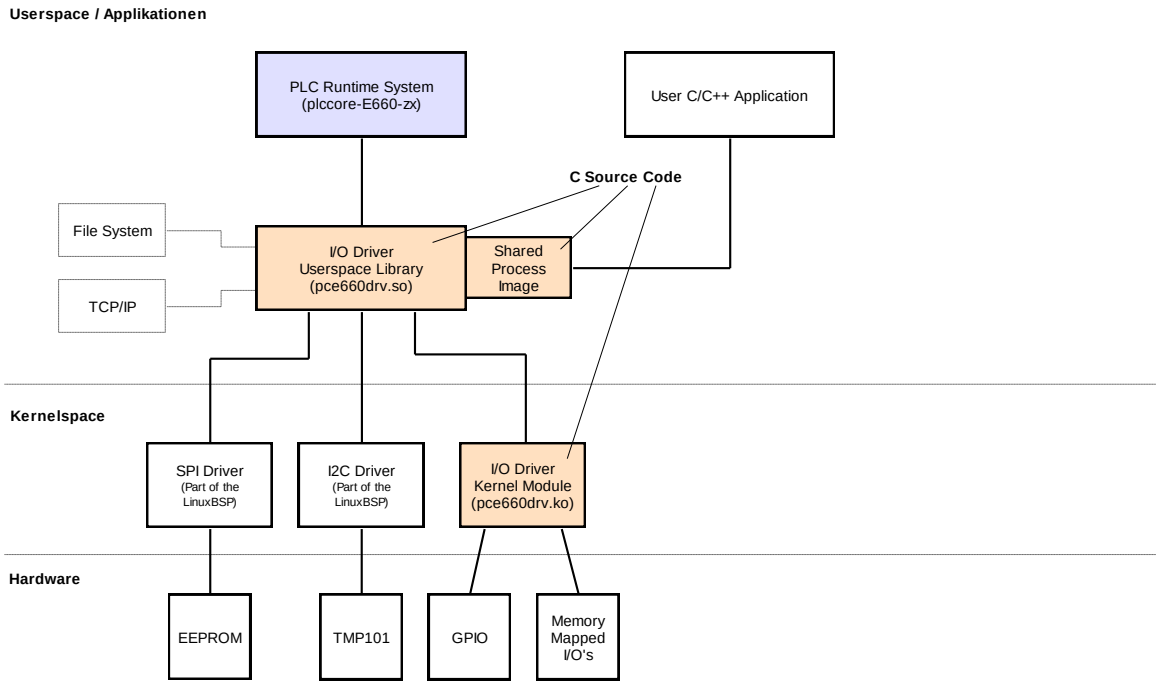


Bild 29: Übersicht zum Driver Development Kit für das PLCcore-E660

Lieferumfang / Komponenten des DDK:

Das DDK beinhaltet folgende Komponenten:

1. Sourcecode für Linux Kernel-Treiber (*pce660drv.ko*, siehe Bild 29), beinhaltet alle notwendigen Files um Kernel-Treiber neu zu erstellen (C- und H-Files, Makefile usw.)

2. Sourcecode für Linux User-Bibliothek (*pce660drv.so*, siehe Bild 29), beinhaltet alle notwendigen Files um User-Bibliothek (incl. der Implementierung des Shared Process Image) neu zu erstellen (C- und H-Files, Makefile usw.)

3. I/O-Treiber Demoapplikation (*iodrvdemo*) im Sourcecode, ermöglicht den schnellen und einfachen Test des I/O-Treibers

4. Dokumentation

Das Driver Development Kit setzt das Softwarepaket **SO-1116** ("VMware-Image des Linux-Entwicklungssystems") voraus, das sowohl die Quellen des verwendeten LinuxBSP als auch die erforderliche GNU-Crosscompiler Toolchain für x86-Prozessoren enthält.

8.3 Testen der Hardwareanschaltung

Das PLCcore-E660 ist primär als Zulieferteil für den Einbau in industriellen Steuerungen bestimmt. Dazu wird das PLCcore-E660 typischerweise auf einer anwenderspezifischen Basisplatine betrieben. Um hier eine einfache Kontrolle der korrekten I/O-Anschaltung zu ermöglichen, wird das

Testprogramm *"iodrvdemo"* zusammen mit der SPS-Firmware auf dem Modul installiert. Dieses Testprogramm setzt direkt auf dem I/O-Treiber auf und ermöglicht so den unmittelbaren Peripheriezugriff.

Damit das Testprogramm *"iodrvdemo"* exklusiven Zugriff auf alle I/O-Ressourcen erhält, muss ein evtl. laufendes SPS-Laufzeitsystem zunächst beendet werden. Dies ist am einfachsten durch den Aufruf des Skriptes *"stopplc"* möglich:

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

Anschließend kann der I/O-Treiber erneut geladen und das Testprogramm *"iodrvdemo"* gestartet werden:

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

Bild 30 veranschaulicht das Testen der Hardwareanschaltung mit *"iodrvdemo"*.

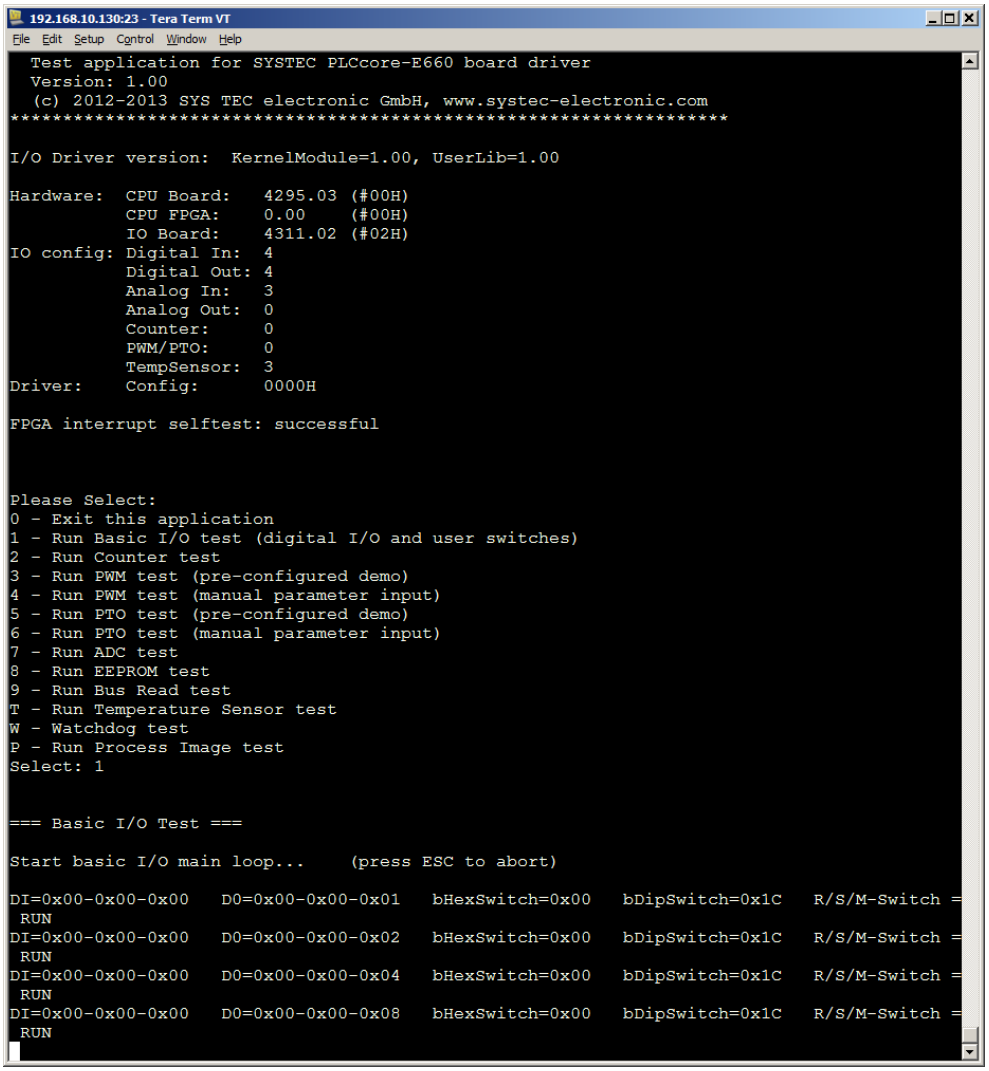


Bild 30: Testen der Hardwareanschaltung mit "iodrvdemo"

Anhang A: Firmware-Funktionsumfang des PLCcore-E660

Tabelle 20 listet die auf dem PLCcore-E660 verfügbaren Firmware-Funktionen und -Funktionsbausteine auf.

Zeichenerklärung:

FB	Funktionsbaustein
FUN	Funktion
Online Help	*OpenPCS* Online-Hilfe
L-1054	Manual *"SYS TEC spezifische Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L-1054)
PARAM:={0,1,2}	für den angegebenen Parameter sind die Werte 0,1 und 2 zulässig

Tabelle 20: Firmware-Funktionen und -Funktionsbausteine des 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) Das PLCcore-E660 unterstützt folgende Geräte zur Ablage nicht-flüchtigen Daten:
DEVICE:=0: Ablage der nicht-flüchtigen Daten in der Datei *"/home/plc/plcdata/PlcPData.bin"*. Diese Datei hat eine fixe Größe von 32 kByte. Beim Aufruf der Funktionsbausteine vom Typ *NVDATA_Xxx* in einem Schreib-Modus werden die modifizierten Daten unmittelbar in die Datei *"/home/plc/plcdata/PlcPData.bin"* geschrieben ("*flush*"). Dadurch gehen bei einer Spannungsunterbrechung keine ungesicherten Daten verloren.
DEVICE:=1: Ablage der nicht-flüchtigen Daten im EEPROM des Das PLCcore-E660. Der EEPROM hat eine fixe Größe von 2 kByte.

(2) Die Schnittstelle UART1 (PORT:=1) dient primär als Serviceschnittstelle zur Administration des PLCcore-E660. Daher sollten über diese Schnittstelle nur Zeichen ausgegeben werden. Empfangene Zeichen versucht das Modul stets als Linux-Kommandos zu interpretieren und auszuführen (siehe Abschnitt 6.6.1).

(3) Die Verwendung der Funktionsbausteine vom Typ CANL2_Xxx ist nur möglich, wenn die betreffende CAN-Schnittstelle nicht bereits für CANopen verwendet wird. Um die Funktionsbausteine vom Typ CANL2_Xxx nutzen zu können, muss daher die betreffende CAN-Schnittstelle in der SPS-Konfiguration deaktiviert sein (siehe Abschnitt 7.4.1). Alternativ kann auch der Eintrag *"Enable="* in der zugehörigen Sektion *"[CANx]"* innerhalb der Konfigurationsdatei ***"/home/plc/bin/plccore-E660.cfg"*** direkt auf 0 gesetzt werden (siehe Abschnitt 7.4.2).

(4) Dir Funktionsbausteine vom Typ HMI_Xxx sind nur verfügbar für die HMI-Version des PLCcore-E660 (Bestellnummer 3390090).

Anhang B: GNU GENERAL PUBLIC LICENSE

Das auf dem PLCcore-E660 eingesetzte Embedded Linux ist unter der GNU General Public License, Version 2 lizenziert. Der vollständige Text dieser Lizenz ist nachfolgend aufgeführt. Eine deutsche Übersetzung ist unter http://www.gnu.de/documents/gpl-2.0.de.html zu finden. Es handelt sich jedoch dabei nicht um eine offizielle oder im rechtlichen Sinne anerkannte Übersetzung.

Das verwendete SPS-System sowie die vom Anwender entwickelten SPS- und C/C++ Programme unterliegen **nicht** der 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

Dokument:	System Manual PLCcore-E660
Dokumentnummer:	L-1555d_1, 1. Auflage Juli 2014

Wie würden Sie dieses Handbuch verbessern?

Haben Sie in diesem Handbuch Fehler entdeckt? Seite

Eingesandt von:

Kundennummer: _____

Name: _____

Firma: _____

Adresse: _____

Einsenden an:

SYS TEC electronic GmbH
Am Windrad 2
D-08468 Heinsdorfergrund
GERMANY
Fax: +49 (0) 3765 38600-4100
Email: info@systec-electronic.com