System Manual
PLCcore-iMX35

User Manual
Version 2

Ausgabe März 2016

Dokument-Nr.: L1567d_2

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

Status/Änderungen

Status: Freigegeben

Datum/Version	Abschnitt	Änderung	Bearbeiter
2014/06/03 Version 1	alle	Erstellung	T. Volckmann
2016/03/15 Version 2	Anhang A	Referenzdokument für Dateisystem-Funktionsbausteine angepasst Funktionsbausteine für Modbus hinzugefügt	T. Volckmann

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) 37 65 / 38 600-0 info@systec-electronic.com	
Technische Hotline:	+49 (0) 37 65 / 38 600-0 support@systec-electronic.com	
Fax:	+49 (0) 37 65 / 38 600-4100	
Webseite:	http://www.systec-electronic.com	

2. Auflage März 2016

Inhalt

1 Einleitung

Vielen Dank, dass Sie sich für das SYS TEC PLCcore-iMX35 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-iMX35. Es wird Ihnen helfen, sich mit dem Funktionsumfang und der Anwendung des PLCcore-iMX35 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-iMX35 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-iMX35

Das PLCcore-iMX35 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-iMX35 ist ESD empfindlich und darf nur an ESD geschützten Arbeitsplätzen von geschultem Fachpersonal ausgepackt und gehandhabt bzw. betrieben werden.

Das PLCcore-iMX35 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-iMX35 basiert auf der Hardware des ECUcore-iMX35 und erweitert dieses um SPS-spezifische Funktionalitäten (SPS-Firmware, Target-Visualisierung). Für die Hardwarekomponenten wie das ECUcore-iMX35 bzw. das PLCcore-iMX35 selbst (die Hardware beider Module ist identisch), die Developmentboards sowie Referenzschaltungen existieren eigene Hardware-Manuals. Softwareseitig wird das PLCcore-iMX35 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-iMX35 relevanten Manuals auf.

Tabelle 1: Übersicht relevanter Manuals zum PLCcore-iMX35

Informationen über…	**In welchem Manual?**
Grundlegende Informationen zum PLCcore-iMX35 (Konfiguration, Administration, Prozessabbild, Anschlussbelegung, Firmwareupdate, Referenzdesigns usw.)	In diesem Manual
Entwicklung anwenderspezifischer C/C++ Applikationen für das ECUcore-iMX35 / PLCcore-iMX35, VMware-Image des Linux-Entwicklungssystems	System Manual ECUcore-iMX35 (Manual-Nr.: L-1569)
Hardware-Beschreibung zum ECUcore-iMX35 / PLCcore-iMX35, Referenzdesigns usw.	Hardware Manual ECUcore-iMX35 (Manual-Nr.: L-1570)
Developmentboard zum ECUcore-iMX35 / PLCcore-iMX35, Referenzdesigns usw.	Hardware Manual Developmentboard iMX35 (Manual-Nr.: L-1571)
Driver Development Kit (DDK) für das ECUcore-iMX35	Software Manual Driver Development Kit (DDK) für ECUcore-iMX35 (Manual-Nr.: L-1572)
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)

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-iMX35** auf Basis des Development Kit für das PLCcore-iMX35.

Abschnitt 5 beschreibt die **Anschlussbelegung** des PLCcore-iMX35.

Abschnitt 6 erklärt Details zur **Anwendung des PLCcore-iMX35**, 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-iMX35**, 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-iMX35** 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 des Datenaustausches zwischen einem SPS-Programm und einer anwenderspezifischen C/C++ Applikation über das **Shared Prozessabbild**.

3 Produktbeschreibung

Das PLCcore-iMX35 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-iMX35 optimal zum Aufbau von anwenderspezifischen HMI (**H**uman **M**achine **I**nterface) Applikationen geeignet.

Bild 1: Ansicht des PLCcore-iMX35

Einige herausragende Merkmale des PLCcore-iMX35 sind:

- leistungsfähiger CPU-Kern (ARM 32-Bit ARM1136JF-S, 532 MHz CPU Takt, 740 MIPS)
- 128 MByte SDRAM Memory, 128 MByte FLASH Memory
- LCD Controller mit Unterstützung für Displays bis max. 800x600 Pixel Auflösung bei 24 Bit Farbtiefe
- Unterstützung für Scrollwheel und 4x4 Matrixtastatur
- 1x 10/100 Mbps Ethernet LAN Interface (mit on–board PHY)
- 2x CAN 2.0B Interface, nutzbar als CANopen-Manager (CiA 302 konform)
- 3x Asynchronous Serial Ports (UART)
- 16 digitale Eingänge, 10 digitale Ausgänge (Standardkonfiguration, modifizierbar über DDK)
- SPI und I^2C extern nutzbar
- On-board Peripherie: RTC, Watchdog, Power-fail Eingang
- On-board Software: Linux, SPS-Firmware, CANopen-Master, HTTP- und FTP-Server
  nur HMI-Version: Target-Visualisierung und HMI Funktionsbausteinbibliothek
- 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 Abmaße (78 * 54 mm)

Für das PLCcore-iMX35 sind verschiedene Firmware-Varianten verfügbar. Diese unterscheiden sich in der Target-Visualisierung und in dem zur Kommunikation zwischen Programmier-PC und PLCcore-iMX35 verwendeten Protokoll:

Art.-Nr.: 3390065/Z4:	PLCcore-iMX35/Z4 (CANopen, ohne Target-Visualisierung) Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
Art.-Nr.: 3390065/Z5:	PLCcore-iMX35/Z5 (Ethernet, ohne Target-Visualisierung) Kommunikation mit Programmier-PC via UDP-Protokoll (Interface ETH0)
Art.-Nr.: 3390075/Z4:	PLCcore-iMX35-HMI/Z4 (CANopen, mit Target-Visualisierung) Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
Art.-Nr.: 3390075/Z5:	PLCcore-iMX35-HMI/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-iMX35 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-iMX35 beinhaltet die gesamte Target-Visualisierung (nur HMI-Version, Art.-Nr. 3390075) 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-iMX35 bietet 16 digitale Eingänge (DI0...DI15, 3.3V-Pegel), 10 digitale Ausgänge (DO0...DO9, 3.3V-Pegel) sowie Unterstützung für Scrollwheel und 4x4 Matrixtastatur. Mit Hilfe des Driver Development Kit (SO-1119) kann diese Standard-I/O-Konfiguration an die spezifischen Applikationsbedingungen adaptiert werden (siehe Abschnitte 4.4.2 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-iMX35 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-iMX35 erweitert und angepasst. Damit kann das PLCcore-iMX35 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-iMX35 die Rückdokumentation des SPS-Programms aus der Steuerung sowie als Debugfunktionalität das Beobachten und Setzen von Variablen, Einzelzyklus, Breakpoints und Einzelschritt.

Die HMI-Version des PLCcore-iMX35 (Art.-Nr. 3390075) verfügt über eine integrierte Target-Visualisierung. Diese 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).

In der Standard-Version des PLCcore-iMX35 (Art.-Nr. 4001025) steht das Display frei für anwenderspezifische, Qt-basierte GUI-Programme zur Verfügung.

Das PLCcore-iMX35 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-iMX35 eingesetzte Embedded Linux ist unter der GNU General Public License, Version 2 lizenziert. Der entsprechende Lizenztext ist im Anhang D aufgeführt. Die vollständigen Quellen des verwendeten LinuxBSP sind im Softwarepaket **SO-1121** ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-iMX35") 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-iMX35

4.1 Übersicht

Das Development Kit PLCcore-iMX35 ist ein leistungsfähiges Komplettpaket zu einem besonders günstigen Preis, um auf Basis einer Kompakt-SPS mit integrierter Target-Visualsierung eigene, anwenderspezifische HMI-Geräte zu entwickeln.

Bild 2: Development Kit PLCcore-iMX35

Das Development Kit PLCcore-iMX35 gewährleistet eine rasche und problemlose Inbetriebnahme des PLCcore-iMX35. Es vereint dazu alle notwendigen Hard- und Software-Komponenten, die zur Erstellung eigener HMI Applikationen erforderlich sind: als Kernkomponente das PLCcore-iMX35 selbst, das zugehörige Developmentboard mit QVGA LCD Display, I/O-Peripherie und zahlreichen Schnittstellen, das IEC 61131-Programmiersystems *OpenPCS*, den *SpiderControl HMI Editor* zum erstellen der Grafikseiten sowie weiters Zubehör. Damit bildet das Developmentkit die ideale Plattform zur Entwicklung anwenderspezifischer HMI-Applikationen auf Basis des PLCcore-iMX35. 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-iMX35 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-iMX35 beinhaltet folgende Komponenten:

- PLCcore-iMX35-HMI
- Developmentboard für PLCcore-iMX35, incl.:
  - 320x240 Pixel QVGA LCD Display
  - Scrollwheel (on-board)
  - 4x4 Matrix Folientastatur (extern ansteckbar)
- 12V – 1,5A Steckernetzteil
- Ethernet-Kabel
- RS232-Kabel
- RS485-Adapter
- CD mit Programmiersoftware, Beispielen, Dokumentation und weiteren Tools

Das im Kit enthaltene Developmentboard ermöglicht eine schnelle Inbetriebnahme des PLCcore-iMX35 und vereinfacht den Aufbau von Prototypen anwenderspezifischer HMI Applikationen, die auf diesem Modul basieren. Das Developmentboard besitzt u.a. verschiedene Möglichkeiten der Spannungszufuhr, ein 320x240 Pixel QVGA LCD Display, Ethernet-Schnittstelle, 2 CAN-Schnittstellen, 4 Taster und 4 LEDs als Bedienelemente für die digitalen Ein- und Ausgänge sowie Scroll Wheel und Anschluss für eine 4x4 Matrixtastatur. Die an den Steckverbindern des PLCcore-iMX35 verfügbaren Signale sind auf Stiftleisten geführt und ermöglichen so einen einfachen Anschluss eigener Peripherie-Schaltungen. Damit bildet das Developmentboard gleichzeitig eine ideale Experimentier- und Testplattform für das PLCcore-iMX35.

Als Software-Entwicklungsplattform sowie Debugumgebung für das PLCcore-iMX35 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-iMX35 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-iMX35

Das für den Betrieb des Development Kit PLCcore-iMX35 notwendige Steckernetzteil sowie die erforderlichen Ethernet- und RS232-Kabel sind bereits im Lieferumfang des Kit enthalten. Für die Inbetriebnahme des Kit sind mindestens die Anschlüsse für Versorgungsspannung (X100/X101), COM0 (X701A) und ETH0 (X702) erforderlich. Darüber hinaus wird der Anschluss von CAN0 (X801A) empfohlen. Einen vollständigen Überblick über die Anschlüsse des Development Kit PLCcore-iMX35 vermittelt Tabelle 2.

Tabelle 2: Anschlüsse des Development Kit PLCcore-iMX35

Anschluss	Bezeichnung auf Developmentboard	Bemerkung
Versorgungsspannung	X100 oder X101	Das im Lieferumfang enthaltene Steckernetzteil ist zum direkten Anschluss an X101 vorgesehen
ETH0 (Ethernet)	X702	Schnittstelle dient zur Kommunikation mit Programmier-PC sowie zum Programmdownload (PLCcore-iMX35-HMI/Z5, Bestellnummer 3390075/Z5), darüber hinaus für das Anwenderprogramm frei verfügbar
COM0 (RS232)	X701A	Schnittstelle wird zur Konfiguration der Baugruppe (z.B. Setzen der IP-Adresse) benötigt und ist im normalen Betrieb für das Anwenderprogramm frei verfügbar
COM1 (RS232)	X701B	Schnittstelle ist für das Anwenderprogramm frei verfügbar.
COM2 (RS485)	X700	Schnittstelle ist für das Anwenderprogramm frei verfügbar.

CAN0 (CAN)	X801A	Schnittstelle dient zur Kommunikation mit Programmier-PC sowie zum Programmdownload (PLCcore-iMX35-HMI/Z4, Bestellnummer 3390075/Z4), darüber hinaus für das Anwenderprogramm frei verfügbar
CAN1	X801B	Schnittstelle ist für das Anwenderprogramm frei verfügbar.

Bild 3 zeigt die Lage der wichtigsten Anschlüsse auf dem Developmentboard für das PLCcore-iMX35. Anstelle des mitgelieferten Steckernetzteiles kann die Stromversorgung des Kit optional auch über X100 mit einer externen Quelle von 12V/1,5A erfolgen.

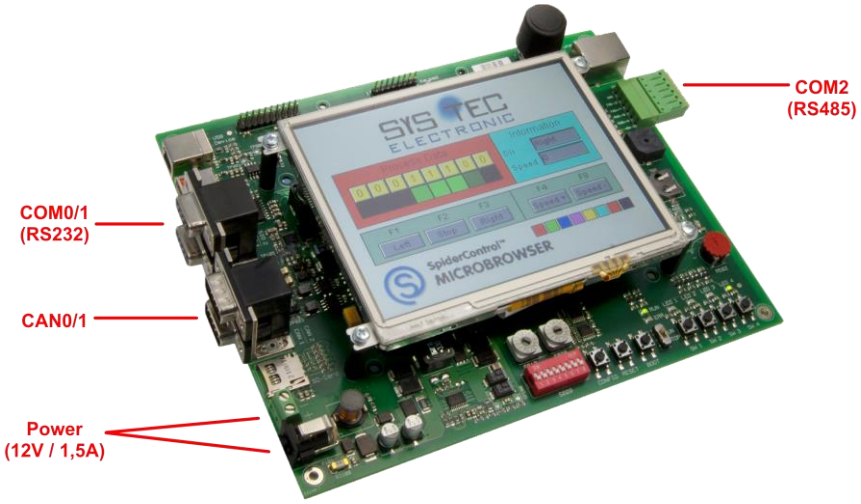


Bild 3: Lage der wichtigsten Anschlüsse auf dem Developmentboard für das PLCcore-iMX35

Hinweis: Bei der Inbetriebnahme sind zunächst die Kabel für Ethernet (ETH0, X702) und RS232 (COM0, X701A) anzuschließen, erst danach ist die Versorgungsspannung (X100 / X101) zu aktivieren.

4.3 Bedienelemente des Development Kit PLCcore-iMX35

Das Development Kit PLCcore-iMX35 ermöglicht eine einfache Inbetriebnahme des PLCcore-iMX35. Es verfügt über verschiedene Bedienelemente sowohl zur Konfiguration des Moduls als auch zur Simulation von Ein- und Ausgängen für den Einsatz des PLCcore-iMX35 als SPS-Kern. Tabelle 3 listet die einzelnen Bedienelemente des Developmentboard auf und beschreibt deren Bedeutung.

Tabelle 3: Bedienelemente des Developmentboard für das PLCcore-iMX35

Bedienelement	Bezeichnung	Bedeutung
Taster 0	S604	Digitaler Eingang DI0 (Prozessabbild: %IX0.0)
Taster 1	S605	Digitaler Eingang DI1 (Prozessabbild: %IX0.1)
Taster 2	S606	Digitaler Eingang DI2 (Prozessabbild: %IX0.2)
Taster 3	S607	Digitaler Eingang DI3 (Prozessabbild: %IX0.3)
LED 0	D602	Digitaler Ausgang DO0 (Prozessabbild: %QX0.0)
LED 1	D603	Digitaler Ausgang DO1 (Prozessabbild: %QX0.1)
LED 2	D604	Digitaler Ausgang DO2 (Prozessabbild: %QX0.2)
LED 3	D605	Digitaler Ausgang DO3 (Prozessabbild: %QX0.3)
Run/Stop-Schalter	S603	Run / Stop für Abarbeitung des SPS-Programms (siehe Abschnitt 6.6.1)
Run-LED	D600	Anzeige Aktivitätszustand der SPS (siehe Abschnitt 6.6.2)
Error-LED	D601	Anzeige Fehlerzustand der SPS (siehe Abschnitt 6.6.3)
Hexcodier-Schalter	S608/S610	Konfiguration der Knotenadresse CAN0, siehe Abschnitt 7.4.2
DIP-Schalter	S609	Konfiguration Bitrate und Master-Modus CAN0, siehe Abschnitt 7.4.2

Eine vollständige Auflistung des Prozessabbildes enthält Tabelle 7 im Abschnitt 6.4.1.

4.4 Optionales Zubehör

4.4.1 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-iMX35 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

4.4.2 Driver Development Kit (DDK)

Das Driver Development Kit für das ECUcore-iMX35 (Bestellnummer SO-1119) 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-iMX35

Die Anschlüsse des PLCcore-iMX35 sind über zwei auf der Modulunterseite montierte, doppelreihige Buchsenleisten nach außen geführt (X600A/B, siehe Bild 5). Die entsprechenden Steckverbinder als Gegenstücken zur Aufnahme des PLCcore-iMX35 sind bei der Firma "W + P" unter folgender Bezeichnung lieferbar:

W+P-Bezeichnung: SMT-Stiftleisten RM 1,27mm, stehend, 2-reihig - 1,00mm
W+P-Bestellnummer: 7072-100-10-00-10-PPST (auch in anderen Höhen lieferbar)

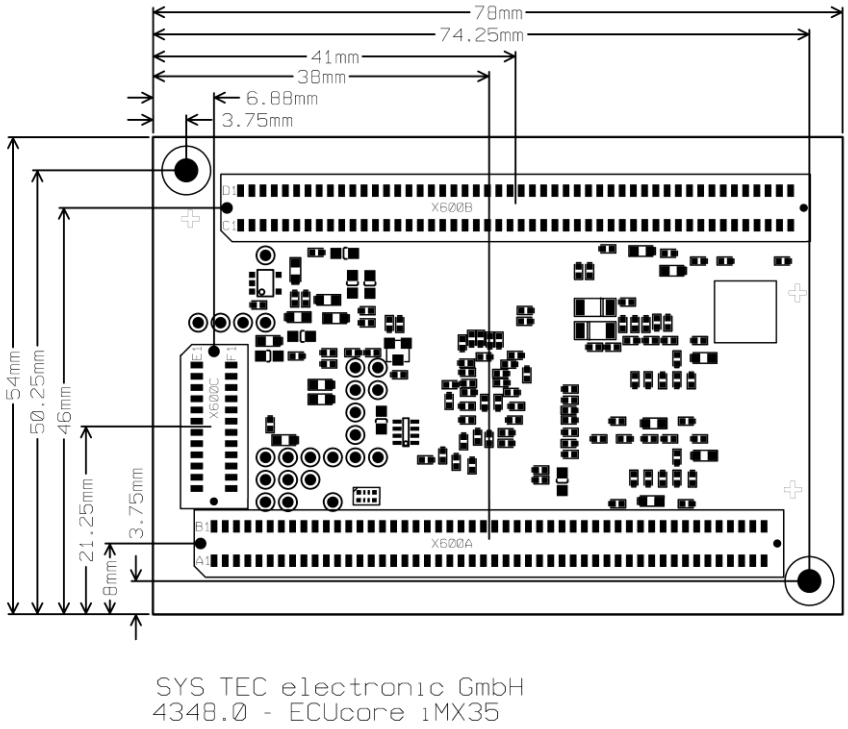


Bild 5: Pinout des PLCcore-iMX35 - top view

Bild 5 verdeutlicht die Lage der Buchsenleisten (X600A/B) auf dem PLCcore-iMX35, Tabelle 4 listet die vollständige Anschlussbelegung des Moduls auf. Die im Bild 5 zusätzlich aufgeführte Buchsenleiste X600C dient zum Anschluss eines JTAG-Interfaces und ist nur auf speziellen Entwicklungsmodulen bestückt. Für die Verwendung des PLCcore-iMX35 als SPS-Kern ist das JTAG-Interface ohne Bedeutung. Eine ausführliche Beschreibung der Modulanschlüsse beinhaltet das Hardware Manual ECUcore-iMX35 (Manual-Nr.: L-1570). Referenzdesigns zur Anwendung des PLCcore-iMX35 in anwenderspezifischen Applikationen enthält Anhang B.

Tabelle 4: Anschlussbelegung des PLCcore-iMX35, vollständig, sortiert nach Anschluss-Pin

Signal	Pin	Pin	Signal		Signal	Pin	Pin	Signal
GND	A01	B01	GND		GND	C01	D01	2V5_EPHY
/BOOT	A02	B02	/MR		Eth_Tx-	C02	D02	GND
/BOOTSTRAP_1	A03	B03	/RESET_IN		Eth_Tx+	C03	D03	Speed
VSTBY	A04	B04	/PFI		Eth_Rx+	C04	D04	Link/Act
/BOOTSTRAP_0	A05	B05	WDI		Eth_Rx-	C05	D05	GND
GND	A06	B06	/PFO		GND	C06	D06	GPIO1_6
RXD1	A07	B07	GND		GPIO1_0	C07	D07	GPIO1_5

Signal	Pin	Pin	Signal		Signal	Pin	Pin	Signal
TXD1	A08	B08	RTS2		GPIO1_1	C08	D08	GPIO1_4
RTS1	A09	B09	CTS2		GND	C09	D09	GND
CTS1	A10	B10	RTS3		SD2_DATA0	C10	D10	GPIO1_3
GPIO2_12	A11	B11	CTS3		SD2_DATA1	C11	D11	USBOTG_OC
GND	A12	B12	GND		SD2_DATA2	C12	D12	USBOTG_PWR
TXD2	A13	B13	TXD3		SD2_DATA3	C13	D13	USBPHY1_VBUS
RXD2	A14	B14	RXD3		SD2_CLK	C14	D14	SD2_CMD
NVCC_3V3	A15	B15	GPIO1_26		GND	C15	D15	GPIO1_31
NVCC_3V3	A16	B16	GPIO2_18		GPIO1_28	C16	D16	GND
GND	A17	B17	Nicht verwendet		Nicht verwendet	C17	D17	CAPTURE
USBPHY2_DP	A18	B18	Nicht verwendet		GPIO2_12	C18	D18	GPIO1_25
USBPHY2_DM	A19	B19	GND		GPIO2_13	C19	D19	COMPARE
USBPHY1_UID	A20	B20	USBPHY1_DP		GPIO2_14	C20	D20	CLKO
Nicht verwendet	A21	B21	USBPHY1_DM		GPIO2_15	C21	D21	GPIO2_26
GND	A22	B22	GND		GPIO2_17	C22	D22	GPIO2_28
I2C2_DAT	A23	B23	CAN1_TX		GPIO2_25	C23	D23	GND
I2C2_CLK	A24	B24	CAN1_RX		GND	C24	D24	BACKL_EN
GND	A25	B25	GPIO1_24		GPIO1_2	C25	D25	GPIO2_31
GPIO3_25	A26	B26	GND		LCD_CONTRAST	C26	D26	GPIO2_29
GND3_26	A27	B27	/RESET		GPIO2_30	C27	D27	GPIO2_19
Nicht verwendet	A28	B28	/PORESET		GPIO2_20	C28	D28	GND
/EN_IO3V3	A29	B29	Nicht verwendet		GND	C29	D29	GPIO2_21
Nicht verwendet	A30	B30	CAN2_RX		GPIO2_22	C30	D30	Nicht verwendet
GND	A31	B31	CAN2_TX		LCD_TXout0+	C31	D31	LCD_TXout0-
CSPI1_SS0	A32	B32	GND		LCD_TXout1+	C32	D32	LCD_TXout1-
CSPI1_SS2/PWMO	A33	B33	CSPI1_SS1		LCD_TXout2+	C33	D33	GND
CSPI1_MOSI	A34	B34	CSPI1_MISO		LCD_TXout2-	C34	D34	LCD_TXoutCLK+
CSPI1_SS3	A35	B35	CSPI1_SCLK		GND	C35	D35	LCD_TXoutCLK-
Nicht verwendet	A36	B36	Nicht verwendet		LCD_R0	C36	D36	LCD_R1
GND	A37	B37	SD1_CLK		LCD_R2	C37	D37	LCD_R3
SD1_CMD	A38	B38	GND		LCD_R4	C38	D38	LCD_R5
SD1_DATA0	A39	B39	SD1_DATA1		LCD_G0	C39	D39	GND
SD1_DATA2	A40	B40	SD1_DATA3		LCD_G1	C40	D40	LCD_G2
MATRIX_C1	A41	B41	MATRIX_C0		GND	C41	D41	LCD_G3
MATRIX_C3	A42	B42	MATRIX_C2		LCD_G4	C42	D42	LCD_G5
GND	A43	B43	MATRIX_R0		LCD_B0	C43	D43	LCD_B1
MATRIX_R1	A44	B44	GND		LCD_B2	C44	D44	LCD_B3
MATRIX_R3	A45	B45	MATRIX_R2		LCD_B4	C45	D45	GND
GPIO1_8	A46	B46	GPIO1_9		LCD_B5	C46	D46	/LVDS_PWD
GPIO1_12	A47	B47	GPIO1_13		GND	C47	D47	GPIO1_15
VBAT	A48	B48	GPIO1_14		LCD_DEN	C48	D48	LCD_DCLK
GND	A49	B49	GND		LCD_HSYNC	C49	D49	LCD_VSYNC
+3V3	A50	B50	+3V3		GND	C50	D50	GND

Tabelle 5 beinhaltet als Untermenge von Tabelle 4 nur alle Ein- und Ausgänge des PLCcore-iMX35, sortiert nach deren Funktion.

Tabelle 5: Anschlussbelegung des PLCcore-iMX35, nur I/O, sortiert nach Funktion

Connector	I/O-Pin	PLC Function 1	PLC Function 2 A=alternative, S=simultaneous
D27	GPIO2_19	DI0 [Switch0]	
C28	GPIO2_20	DI1 [Switch1]	
D29	GPIO2_21	DI2 [Switch2]	
C30	GPIO2_22	DI3 [Switch3]	
B25	GPIO1_24	DI4	
D18	GPIO1_25	DI5	
B15	GPIO1_26	DI6	
D15	GPIO1_31	DI7	
D14	GPIO2_0 (SD2_CMD)	DI8	
A47	GPIO1_12	DI9	
B16	GPIO2_18	DI10	
B9	GPIO3_13	DI11	
A26	GPIO3_25	DI12	
A27	GPIO3_26	DI13	
D21	GPIO2_26	DI14	
C25	GPIO1_2	DI15	
D22	GPIO2_28	DO0 [LED0]	
D26	GPIO2_29	DO1 [LED1]	
C27	GPIO2_30	DO2 [LED2]	
D25	GPIO2_31	DO3 [LED3]	
C7	GPIO1_0	DO4	
C8	GPIO1_1	DO5	
D10	GPIO1_3	DO6	
D6	GPIO1_6	DO7	
C14	GPIO2_1 (SD2_CLK)	DO8	
C21	GPIO2_15	DO9	
B41	MATRIX_C0	MATRIX_C0	
A41	MATRIX_C1	MATRIX_C1	
B42	MATRIX_C2	MATRIX_C2	
A42	MATRIX_C3	MATRIX_C3	
B43	MATRIX_R0	MATRIX_R0	
A44	MATRIX_R1	MATRIX_R1	
B45	MATRIX_R2	MATRIX_R2	
A45	MATRIX_R3	MATRIX_R3	
D17	CAPTURE	Scrollwheel DIR	
D19	COMPARE	Scrollwheel CLK	
C22	GPIO2_17	Scrollwheel Button	
A46	GPIO1_8	Error-LED	

Die Funktion des Run/Stop-Schalters für die SPS-Firmware erläutert Abschnitt 6.6.1. Ist für den Einsatz des PLCcore-iMX35 auf einer anwender-spezifischen Basisplatine kein Run/Stop-Schalter vorgesehen, muss an den Modulanschlüssen die Codierung für *"Run"* fest verdrahtet sein (siehe dazu auch Referenzdesign im Anhang B).

6 SPS-Funktionalität des PLCcore-iMX35

6.1 Übersicht

Das PLCcore-iMX35 realisiert eine vollständige, Linux-basierte Kompakt-SPS in Form eines Aufsteckmoduls ("Core"). Das PLCcore-iMX35 basiert dabei auf der Hardware des ECUcore-iMX35 und erweitert dieses um SPS-spezifische Funktionalitäten (SPS-Firmware, Target-Visualisierung). Beide Module, sowohl ECUcore-iMX35 als auch PLCcore-iMX35, benutzt dasselbe Embedded Linux als Betriebssystem. Folglich sind auch Konfiguration und C/C++ Programmierung des PLCcore-iMX35 weitestgehend identisch zum ECUcore-iMX35.

6.2 Systemstart des PLCcore-iMX35

Standardmäßig lädt das PLCcore-iMX35 nach Power-on bzw. Reset alle notwendigen Firmware-Komponenten und startet anschließend die Abarbeitung des SPS-Programms. Damit eignet sich das PLCcore-iMX35 für den Einsatz in autarken Steuerungssystemen, die nach einer Spannungsunterbrechung selbständig und ohne Benutzeraktionen die Ausführung des SPS-Programms wieder aufnehmen. Bild 6 zeigt den Systemstart im Detail.

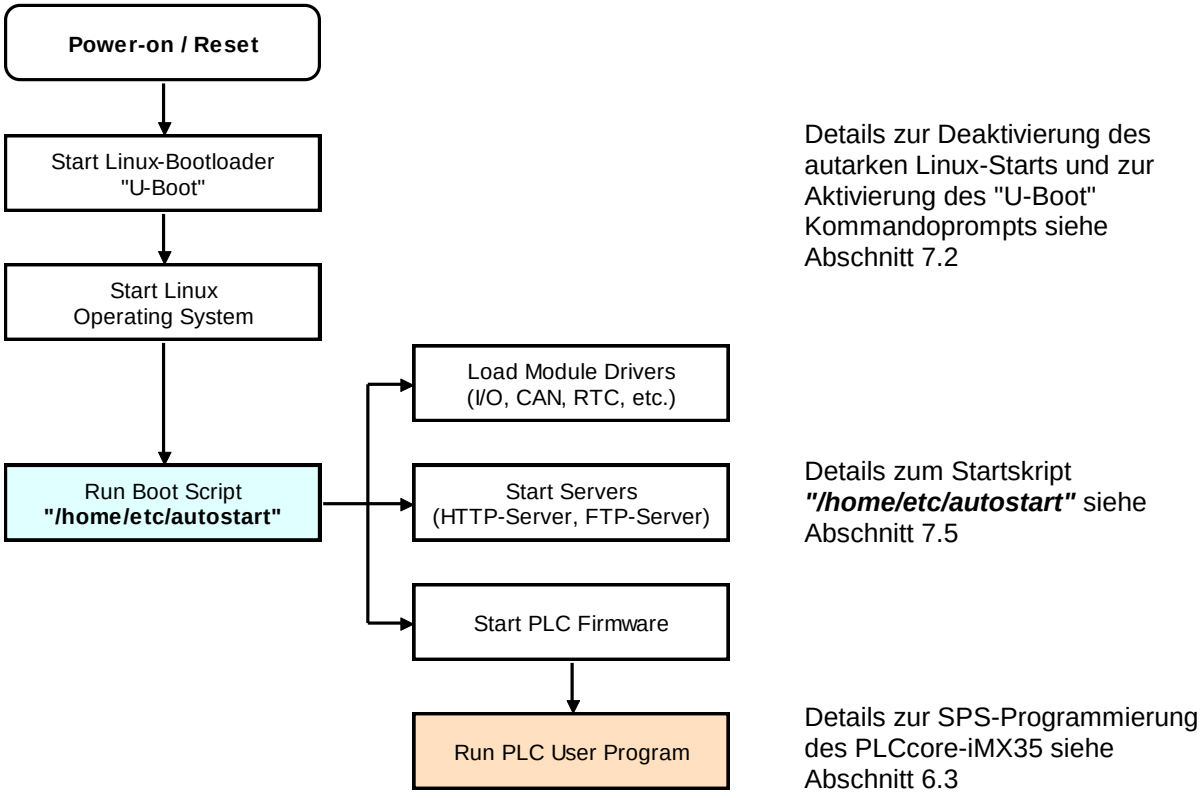


Bild 6: Systemstart des PLCcore-iMX35

6.3 Programmierung des PLCcore-iMX35

Das PLCcore-iMX35 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-iMX35 relevanten Manuals auf.

Die Firmware des PLCcore-iMX35 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 6.4) sowie die Funktionsweise der Bedienelemente (Hexcodier-Schalter, DIP-Schalter, Run/Stop-Schalter, Run-LED, Error-LED).

Die Firmware des PLCcore-iMX35 stellt dem SPS-Programmierer in Abhängigkeit der eingesetzten Firmware-Variante zahlreiche Funktionsbausteine für den Zugriff auf Kommunikationsschnittstellen bereit. Tabelle 6 listet die Verfügbarkeit der Kommunikations-FB-Klassen (SIO, CAN, UDP) für die einzelnen Firmware-Typen des PLCcore-iMX35 auf. Abschnitt 7.7 beschreibt die Auswahl der jeweils zu startenden Firmware-Variante.

Tabelle 6: Unterstützung von Funktionsbaustein-Klassen für verschiedene PLCcore Typen

Interface Typ	**PLCcore-iMX35/Z3 Art.-Nr.: 3390065/Z3 3390075/Z3**	**PLCcore-iMX35/Z4 Art.-Nr.: 3390065/Z4 3390075/Z4**	**PLCcore-iMX35/Z5 Art.-Nr.: 3390065/Z5 3390075/Z5**	**Bemerkung**
CAN	-	x	x	FB-Beschreibung siehe Manual L-1008
UDP	-	x	x	FB-Beschreibung siehe Manual L-1054
SIO	x	x	x	FB-Beschreibung siehe Manual L-1054
HMI	x (nur 3390075)	x (nur 3390075)	x (nur 3390075)	FB-Beschreibung siehe Manual L-1321

Eine vollständige Auflistung der vom PLCcore-iMX35 unterstützten Firmware-Funktionen und -Funktionsbausteine enthält Tabelle 25 im Anhang A.

Detaillierte Informationen zu Nutzung der CAN-Schnittstellen in Verbindung mit CANopen beschreibt Abschnitt 6.7.

6.4 Prozessabbild des PLCcore-iMX35

6.4.1 Lokale Ein- und Ausgänge

Das PLCcore-iMX35 besitzt ein Prozessabbild mit identischen Adressen im Vergleich zu anderen SYS TEC-Kompaktsteuerungen. Vom PLCcore-iMX35 werden die in Tabelle 7 aufgelisteten Ein- und Ausgänge unterstützt.

Tabelle 7: Zuordnung der Ein- und Ausgänge zum Prozessabbild des PLCcore-iMX35

I/O des PLCcore-iMX35	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
AIN0, siehe (1)	**%IW8.0**	15Bit + sign(0 … + 32767)
AIN1, siehe (1)	**%IW10.0**	15Bit + sign(0 … + 32767)
AIN2, siehe (1)	**%IW12.0**	15Bit + sign(0 … + 32767)
AIN3, siehe (1)	**%IW14.0**	15Bit + sign(0 … + 32767)
On-board Temperatursensor	**%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 … DO9	**%QB1.0** %QX1.0	als Byte mit DO8 … DO9 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/bin/plccore-imx35.cfg"*** direkt gesetzt werden (siehe Abschnitt 7.4.3). Die entsprechende Konfigurationseinstellung wird beim Starten der SPS-Firmware ausgewertet.

Die Ein- und Ausgänge des PLCcore-iMX35 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.

6.4.2 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-iMX35. Damit können sämtliche Ein- und Ausgänge des PLCcore-iMX35 flexibel an die jeweiligen Erfordernisse angepasst werden. Dies wiederum hat zur Folge, dass das Prozessabbild des PLCcore-iMX35 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-iMX35*" (Bestellnummer SO-1119).

6.5 Kommunikationsschnittstellen

6.5.1 Serielle Schnittstellen

Das PLCcore-iMX35 besitzt 3 serielle Schnittstellen (COM0 … COM2). COM0 und COM1 sind als RS-232 Schnittstellen, COM2 als RS-485 Schnittstelle ausgeführt. Details zur Hardwareanschaltung beschreibt das *"Hardware Manual Development Board ECUcore-iMX35"* (Manual-Nr.: L-1571).

COM0: Die Schnittstelle COM0 dient primär als Serviceschnittstelle zur Administration des PLCcore-iMX35. Sie wird standardmäßig im Boot-Skript *"/etc/inittab"* dem Linux-Prozess *"getty"* zugeordnet und als Linux-Konsole zur Administration des PLCcore-iMX35 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-1121 ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-iMX35").

COM1/2: Die Schnittstelle COM1 ist frei verfügbar und dient in der Regel zum Datenaustausch zwischen PLCcore-iMX35 und anderen Feldgeräten unter Kontrolle des SPS-Programms.

Die Schnittstelle COM1 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).

6.5.2 CAN-Schnittstellen

Das PLCcore-iMX35 besitzt 2 CAN-Schnittstellen (CAN0 … CAN1). Details zur Hardwareanschaltung beschreibt das *"Hardware Manual Development Board ECUcore-iMX35"* (Manual-Nr.: L-1571).

Die CAN-Schnittstellen ermöglichen den Datenaustausch mit anderen Geräten über Netzwerkvariablen und sind zudem aus einem SPS-Programm über die Funktionsbausteine vom Typ *"CAN_Xxx"* nutzbar (siehe Abschnitt 6.7 sowie *"User Manual CANopen-Erweiterung für IEC 61131-3"*, Manual-Nr.: L-1008).

Detaillierte Informationen zur Nutzung der CAN-Schnittstelle in Verbindung mit CANopen beschreibt Abschnitt 6.7.

6.5.3 Ethernet-Schnittstellen

Das PLCcore-iMX35 besitzt 1 Ethernet-Schnittstelle (ETH0). Details zur Hardwareanschaltung beschreibt das *"Hardware Manual Development Board ECUcore-iMX35"* (Manual-Nr.: L-1571).

Die Ethernet-Schnittstelle dient sowohl als Serviceschnittstelle zur Administration des PLCcore-iMX35 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.

6.6 Bedien- und Anzeigeelemente

6.6.1 Run/Stop-Schalter

Der Modulanschluss *"GPIO3_12"* (siehe Tabelle 5 und Referenzdesign im Anhang B) 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.

6.6.2 Run-LED (grün)

Der Modulanschluss *"GPIO1_9"* (siehe Tabelle 5 und Referenzdesign im Anhang B) ist für die Anbindung einer Run-LED vorgesehen. Diese Run-LED gibt Auskunft über den Aktivitätszustand der Steuerung, die durch verschiedene Modi dargestellt werden:

Tabelle 8: 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

6.6.3 Error-LED (rot)

Der Modulanschluss *"GPIO1_8"* (siehe Tabelle 5 und Referenzdesign im Anhang B) ist für die Anbindung einer Error-LED vorgesehen. Diese Error-LED gibt Auskunft über den Fehlerzustand der Steuerung, die durch verschiedene Modi dargestellt werden:

Tabelle 9: 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.
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.

6.7 Nutzung der CAN-Schnittstellen mit CANopen

Das PLCcore-iMX35 besitzt 2 CAN-Schnittstellen (CAN0 … CAN1), die beide als CANopen-Manager nutzbar ist (konform zum CiA Draft Standard 302). Die Konfiguration der beiden Schnittstellen (aktiv/inaktiv, Knotennummer, Bitrate, Master an/aus) beschreibt Abschnitt 7.4.

Die beiden CAN-Schnittstellen ermöglichen den Datenaustausch mit anderen Geräten über Netzwerkvariablen und sind 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-iMX35 nutzbar.

Die SDO-Kommunikation erfolgt grundsätzlich immer über Funktionsbausteine vom Typ *"CAN_SDO_Xxx"* (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-iMX35 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-iMX35 kann optional die Aufgabe des CANopen-Managers übernehmen. Die Aktivierung des Managers beschreibt Abschnitt 7.4.

Als CANopen-Manager ist das PLCcore-iMX35 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.

6.7.1 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-iMX35 geladen werden muss.

6.7.2 CAN-Schnittstelle CAN1

Im Gegensatz zur Schnittstelle CAN0 ist die Schnittstelle CAN1 als statisches Object-Dictionary realisiert. Das bedeutet, dass sowohl die Menge der Netzwerkvariablen (Kommunikationsobjekte) als auch die Anzahl der zur Verfügung stehenden PDOs fest vorgegeben sind. Die Konfiguration der PDOs ist jedoch zur Laufzeit modifizierbar, d.h. die verwendeten Kommunikationsparameter (CAN Identifier, usw.) und die Zuordnung der Netzwerkvariablen auf die einzelnen Bytes eines CAN-Telegramms (Mapping) können durch den Anwender selber vorgegeben und modifiziert werden. Im statischen Object-Dictionary ist also lediglich die Menge der Objekte fix (Anzahl Netzwerkvariablen und PDOs), deren Verwendung und Eigenschaften können aber zur Laufzeit konfiguriert werden. Damit verhält sich das PLCcore-iMX35 am Interface CAN1 wie ein CANopen I/O-Gerät.

Die Netzwerkvariablen für das SPS-Programm sind über den Merkerbereich des Prozessabbildes ansprechbar. Dabei sind jeweils fix 252 Bytes als Eingangs- sowie 252 Bytes als Ausgangsvariablen nutzbar. Um einen beliebigen Datenaustausch mit anderen CANopen I/O-Geräten zu ermöglichen, ist der Bereich der statischen Netzwerkvariablen überlappend auf verschiedene Datentypen im Object-Dictionary abgebildet (BYTE, SINT, WORD, INT, DWORD, DINT). Die Variablen der verschiedenen Datentypen liegen dabei alle im selben Speicherbereich, d.h. die Variablen repräsentieren dieselben physikalischen Speicherstellen. Eine WORD-Variable überlagert somit 2 BYTE-Variablen, eine DWORD-Variable dementsprechend 2 WORD- bzw. 4 BYTE-Variablen. Bild 7 veranschaulicht die Lage der Netzwerkvariablen für CAN1 innerhalb des Merkerbereiches.

CAN1 Input Variables

	CAN1 IN0	CAN1 IN1	CAN1 IN2	CAN1 IN3	CAN1 IN4	CAN1 IN5	CAN1 IN6	CAN1 IN7	...	CAN1 IN244	CAN1 IN245	CAN1 IN246	CAN1 IN247	CAN1 IN248	CAN1 IN249	CAN1 IN250	CAN1 IN251
BYTE / SINT, USINT	%MB 0.0 (Byte0)	%MB 1.0 (Byte1)	%MB 2.0 (Byte2)	%MB 3.0 (Byte3)	%MB 4.0 (Byte4)	%MB 5.0 (Byte5)	%MB 6.0 (Byte6)	%MB 7.0 (Byte7)		%MB 244.0 (Byte244)	%MB 245.0 (Byte245)	%MB 246.0 (Byte246)	%MB 247.0 (Byte247)	%MB 248.0 (Byte248)	%MB 249.0 (Byte249)	%MB 250.0 (Byte250)	%MB 251.0 (Byte251)
WORD / INT, UINT	%MW 0.0 (Word0)		%MW 2.0 (Word1)		%MW 4.0 (Word2)		%MW 6.0 (Word3)		...	%MW 244.0 (Word122)		%MW 246.0 (Word123)		%MW 248.0 (Word124)		%MW 250.0 (Word125)	
DWORD / DINT, UDINT	%MD 0.0 (Dword0)				%MD 4.0 (Dword1)					%MD 244.0 (Dword61)				%MD 248.0 (Dword62)			

CAN1 Output Variables

	CAN1 OUT0	CAN1 OUT1	CAN1 OUT2	CAN1 OUT3	CAN1 OUT4	CAN1 OUT5	CAN1 OUT6	CAN1 OUT7	...	CAN1 OUT244	CAN1 OUT245	CAN1 OUT246	CAN1 OUT247	CAN1 OUT248	CAN1 OUT249	CAN1 OUT250	CAN1 OUT251
BYTE / SINT, USINT	%MB 256.0 (Byte0)	%MB 257.0 (Byte1)	%MB 258.0 (Byte2)	%MB 259.0 (Byte3)	%MB 260.0 (Byte4)	%MB 261.0 (Byte5)	%MB 262.0 (Byte6)	%MB 263.0 (Byte7)		%MB 500.0 (Byte244)	%MB 501.0 (Byte245)	%MB 502.0 (Byte246)	%MB 503.0 (Byte247)	%MB 504.0 (Byte248)	%MB 505.0 (Byte249)	%MB 506.0 (Byte250)	%MB 507.0 (Byte251)
WORD / INT, UINT	%MW 256.0 (Word0)		%MW 258.0 (Word1)		%MW 260.0 (Word2)		%MW 262.0 (Word3)		...	%MW 500.0 (Word122)		%MW 502.0 (Word123)		%MW 504.0 (Word124)		%MW 506.0 (Word125)	
DWORD / DINT, UDINT	%MD 265.0 (Dword0)				%MD 260.0 (Dword1)					%MD 500.0 (Dword61)				%MD 504.0 (Dword62)			

Bild 7: Lage der Netzwerkvariablen für CAN1 innerhalb des Merkerbereiches

Tabelle 10 veranschaulicht die Repräsentation der Netzwerkvariablen durch entsprechende Einträge im Object-Dictionary des Interface CAN1.

Tabelle 10: Repräsentation der Netzwerkvariablen für CAN1 durch Einträge im Object-Dictionary

OD-Bereich	OD-Variable / EDS-Eintrag	Datentyp CANopen	Datentyp IEC 61131-3
Inputs (Eingänge aus Sicht des PLCcore-iMX35)			
Index 2000H Sub 1 … 252	CAN1InByte0 … CAN1InByte251	Unsigned8	BYTE, USINT
Index 2001H Sub 1 … 252	CAN1InSInt0 … CAN1InSInt251	Integer8	SINT
Index 2010H Sub 1 … 126	CAN1InWord0 … CAN1InWord125	Unsigned16	WORD, UINT
Index 2011H Sub 1 … 126	CAN1InInt0 … CAN1InInt125	Integer16	INT
Index 2020H Sub 1 … 63	CAN1InDword0 … CAN1InDword62	Unsigned32	DWORD, UDINT
Index 2021H Sub 1 … 63	CAN1InDInt0 … CAN1InDInt62	Integer32	DINT
Outputs (Ausgänge aus Sicht des PLCcore-iMX35)			
Index 2030H Sub 1 … 252	CAN1OutByte0 … CAN1OutByte251	Unsigned8	BYTE, USINT
Index 2031H Sub 1 … 252	CAN1OutSInt0 … CAN1OutSInt251	Integer8	SINT
Index 2040H Sub 1 … 126	CAN1OutWord0 … CAN1OutWord125	Unsigned16	WORD, UINT
Index 2041H Sub 1 … 126	CAN1OutInt0 … CAN1OutInt125	Integer16	INT
Index 2050H Sub 1 … 63	CAN1OutDword0 … CAN1OutDword62	Unsigned32	DWORD, UDINT
Index 2051H Sub 1 … 63	CAN1OutDInt0 … CAN1OutDInt62	Integer32	DINT

Im Object-Dictionary des Interface CAN1 sind insgesamt 16 TPDO und 16 RPDO verfügbar. Die ersten 4 TPDO und RPDO sind entsprechend dem Predefined Connection Set vorkonfiguriert und freigeschalten. In diese PDOs sind jeweils die ersten 32 Byte der Input- und Output-Variablen gemapped. Tabelle 11 listet die vorkonfigurierten PDOs für Interface CAN1 im Detail auf.

Tabelle 11: Vorkonfigurierte PDOs für Interface CAN1

PDO	CAN-ID	Daten
1. RPDO	0x200 + NodeID	%MB0.0 … %MB7.0
2. RPDO	0x300 + NodeID	%MB8.0 … %MB15.0
3. RPDO	0x400 + NodeID	%MB16.0 … %MB23.0
4. RPDO	0x500 + NodeID	%MB24.0 … %MB31.0
1. TPDO	0x180 + NodeID	%MB256.0 … %MB263.0
2. TPDO	0x280 + NodeID	%MB264.0 … %MB271.0
3. TPDO	0x380 + NodeID	%MB272.0 … %MB279.0
4. TPDO	0x480 + NodeID	%MB280.0 … %MB287.0

Durch die Limitierung auf jeweils 16 TPDO und 16 RPDO können von den insgesamt 504 Bytes für Netzwerkvariablen im Merkerbereich (2 252Bytes) nur 256 Bytes (2 * 16PDO * 8Byte/PDO) per PDO übertragen werden. Unabhängig davon ist aber der Zugriff auf alle Variablen per SDO möglich.

Die Konfiguration (Mapping, CAN Identifier usw.) der Schnittstelle CAN1 erfolgt typischerweise durch einen externe Configuration Manager, der das Object-Dictionary anhand einer vom CANopen-Konfigurator erstellten DCF-Datei parametriert. Das PLCcore-iMX35 unterstützt das persistente Speichern und Rückladen einer gesicherten Konfiguration über die standardmäßig dafür vorgesehenen Objekteinträge 1010H und 1011H.

Alternativ kann die Konfiguration (Mapping, CAN Identifier usw.) des statischen Object-Dictionaries für die Schnittstelle CAN1 auch vom SPS-Programm aus unter Nutzung der SDO-Funktionsbausteine erfolgen. Die beiden Eingänge *NETNUMBER* und *DEVICE* sind hierfür wie folgt zu belegen:

```
NETNUMBER  := 1;          (* Interface CAN1 *)
DEVICE     := 0;          (* local Node     *)
```

Das SPS-Programmbeispiel *"ConfigCAN1"* verdeutlicht die Konfiguration der Schnittstelle CAN0 durch ein SPS-Programm unter Anwendung der Funktionsbausteine vom Typ *"CAN_SDO_Xxx"*.

6.8 Integrierte Target-Visualisierung

Das PLCcore-iMX35-HMI **(nur Art.-Nr. 3390075)** 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-iMX35 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.

6.8.1 LCD und Touchscreen

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 Touchscreen arbeitet unmittelbar mit der Target-Visualisierung des PLCcore-iMX35-HMI zusammen, d.h. Touch-Events werden direkt vom *SpiderControl MicroBrowser* verarbeitet. Eine Weiterleitung von Touch-Events an das SPS-Programm ist nicht vorgesehen, da eine sinnvolle Interpretation dieser Daten (X- und Y-Koordinate, Anpressdruck) ohnehin nicht möglich ist.

Tochscreen und Toch Controller müssen vor der ersten Verwendung aufeinander abgestimmt – kalibriert – werden. Ohne Kalibrierung arbeitet der Touchscreen extrem ungenau, normalerweise ist eine korrekte Bedienung unmöglich. Die Geräte-Firmware kann beim Booten des SPS Systems prüfen, ob die erforderliche Kalibrierung des Touchscreens durchgeführt wurde. Falls nicht, wird noch vor dem Start der SPS-Firmware das entsprechende Kalibrierprogramm aufgerufen. Diese automatische Prüfung kann durch entsprechende Konfigurationseinstellungen des Modules wahlweise aktiviert bzw. deaktiviert werden. Ebenso lässt sich der Touchscreen, falls notwendig, jederzeit manuell nachkalibrieren. Details hierzu beschreibt Abschnitt 7.14.

6.8.2 Scrollwheel und Matrixtastatur

Die Modulanschlüsse *"MATRIX_C0 … MATRIX _C3"* und *„MATRIX_R0 … MATRIX_R3“* sind für die Anbindung einer 4x4 Matrixtastatur vorgesehen. Darüber hinaus erlauben die Modulanschlüsse *"SW_DIR"*, *"SW_S"* und *"SW_CLK"* den Anschluss eines Scrollwheels mit Push-Button (siehe Tabelle 5 und Referenzdesign im Anhang B). Bild 8 Zeigt den Anschluss der im Development Kit PLCcore-iMX35 enthaltenen Folientastatur an das Developmentboard.

Bild 8: Anschluss der Folientastatur an das Developmentboard

Die Standardbelegung der im Development Kit PLCcore-iMX35 enthaltenen Folientastatur sowie die Tastenzuordnungen für das Scrollwheel zeigt Bild 9. Beschriftungskarten im Maßstab 1:1 mit der Standardbelegung zum Einstecken in die Folientatstur enthält Bild 43 im Anhang C.

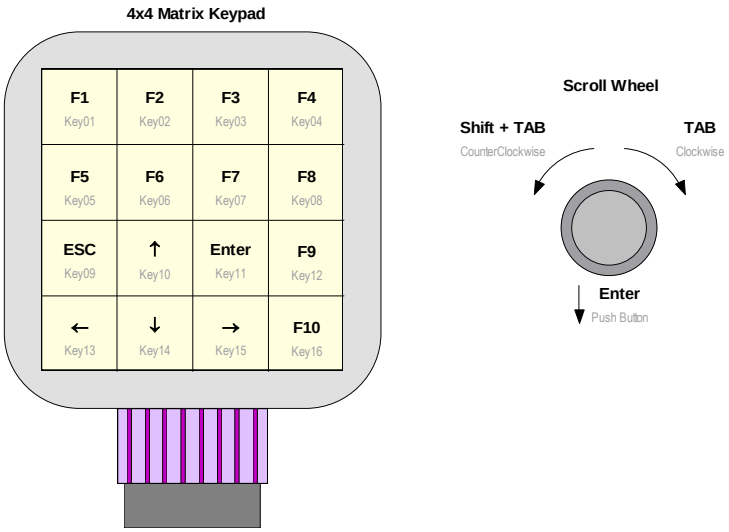


Bild 9: Standardbelegung von Folientastatur und Scrollwheel

Mit Hilfe der beiden Firmware-Funktionsbausteine *HMI_REG_KEY_FUNCTION_TAB* und *HMI_SEL_KEY_FUNCTION_TAB* können durch das SPS-Programm bis zu 4 Tastaturtabellen definiert und wechselweise aktiviert werden (Details zu diesen Funktionsbausteinen siehe "*SYS TEC spezifische HMI Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L-1231). Die Tastaturtabellen beinhalten neben den 16 Einträgen für die Tasten der 4x4 Matrixtastatur noch 3 weitere Einträge für das Scrollwhell (Drehung links, Drehung rechts und Push-Button), so dass sich auch dessen Funktionen flexibel anpassen lassen. Tabelle 12 verdeutlicht Aufbau und Standardbelegung der Tastaturtabelle durch die SPS-Firmware.

Hinweis: Bei der im Development Kit PLCcore-iMX35 enthaltenen Folientastatur kann die Beschriftung durch Austausch der auf der Rückseite eingesteckten Beschriftungskarten flexibel an die reale Tastaturbelegung angepasst werden. Als Vorlage können die Beschriftungskarten mit Maßangabe im Anhang C (Bild 43) verwendet werden.

Tabelle 12: Standard-Tastaturtabelle der SPS-Firmware

Tabellen-Index	Device	Funktion	Belegung
0	Matrixtastatur	Key01	'FKEY_1'
1		Key02	'FKEY_2'
2		Key03	'FKEY_3'
3		Key04	'FKEY_4'
4		Key05	'FKEY_5'
5		Key06	'FKEY_6'
6		Key07	'FKEY_7'
7		Key08	'FKEY_8'
8		Key09	'ESC'
9		Key10	'UP'
10		Key11	'ENTER'
11		Key12	'FKEY_9'
12		Key13	'LEFT'
13		Key14	'DOWN'
14		Key15	'RIGHT'
15		Key16	'FKEY_10'
16	Scrollwheel	Push Button	'ENTER'
17		Clockwise	'TAB'
18		CounterClockwise	'SHIFT-TAB'

Die von der Matrix-Tastatur und vom Scrollwheel generierten Events werden standardmäßig direkt an den *SpiderControl MicroBrowser* gesendet und dort verarbeitet. Alternativ ist es aber auch möglich, diese Events entweder selektiv nur für einzelne Steuer-Elemente oder global für alle Input-Events an das SPS-Programm umzuleiten und dort auszuwerten. Die hierfür erforderlichen Firmware-Funktionsbausteine *"HMI_REG_EDIT_CONTROL_TAB"* sowie *"HMI_SEL_EVENT_HANDLER"* und *"HMI_GET_INPUT_EVENT"* beschreibt das Manual "*SYS TEC spezifische HMI Erweiterungen für OpenPCS / IEC 61131-3"* (Manual-Nr.: L-1231).

6.8.3 Steuerung der Display-Helligkeit

Die Steuerung der Display-Helligkeit am PLCcore-iMX35 erfolgt über Schreib- und Lesezugriffe auf Einträge des Display-Treibers im Dateisystem der SPS. Alle Einträge des Display-Treibers befinden sich im Verzeichnis:

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

Hier sind folgende Treiber-Einträge relevant:

`bl_power`: 1 = Display wird mit max. Helligkeit betrieben, unabhängig vom Wert *"brightness"*
0 = Display-Helligkeit wird durch den in *"brightness"* eingetragenen Wert bestimmt

`brightness`: Helligkeitswert (1=min ... 255=max), nur wirksam bei *"bl_power = 0"*

`actual_brightness`: aktuell wirksamer Helligkeitswert
bl_power := 0 -> Kopie des Wertes *"brightness"*
bl_power := 1 -> konstant 0

`max_brightness`: Konstante für max. Helligkeitswert (= 255)

Der Funktionsbaustein ***HMI_SET_DISPLAY_BRIGHTNESS*** ermöglicht die Steuerung der Display-Helligkeit durch das SPS-Programm. Details zu diesem Baustein beschreibt das Manual *"SYS TEC spezifische HMI-Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L-1321).

Der Funktionsbaustein *HMI_SET_DISPLAY_BRIGHTNESS* unterstützt zwei alternative Betriebsarten. Standardmäßig schreibt der Baustein den am Eingang *BRIGHTNESS* übergebenen Helligkeitswert direkt auf den Treibereintrag *"/sys/devices/platform/mx3_sdc_fb/backlight/mx3fb-bl/brightness"* im Dateisystem der Steuerung. Da der Wert dabei unverändert an den Treiber weiter gegeben wird, repräsentiert *BRIGHTNESS := 1* den minimalen sowie *BRIGHTNESS := 255* den maximalen Wert für die Helligkeit (siehe oben).

Alternativ kann es möglich sein, bei Anbindung spezieller Displays den Standard-Treiber durch einen dedizierten Treiber für den entsprechenden Display-Typ ersetzen zu müssen. Um auch in diesem Fall den Baustein *HMI_SET_DISPLAY_BRIGHTNESS* nutzen zu können, kann die SPS-Firmware so konfiguriert werden, dass sie ein externes Shell-Skript zum Setzen der Display-Helligkeit benutzt. Das aufzurufende Skript ist in der Konfigurationsdatei ***"/home/plc/bin/plccore-imx35.cfg"*** einzutragen (Eintrag *"CmdSetDispBr="* in Sektion *"[Visu]"*, siehe Abschnitt 7.4.3):

```
[Visu]
CmdSetDispBr=<script_name>
```

Im Verzeichnis ***"/home/plc/bin"*** befindet sich neben der SPS-Firmware bereits das als Vorlage geeignete Shell-Skript ***"set_disp_br.sh"***. Um dieses Skript zu aktivieren, ist die Konfigurationsdatei ***"plccore-imx35.cfg"*** wie folgt anzupassen:

```
[Visu]
CmdSetDispBr=./set_disp_br.sh
```

Neben dem im Lieferumfang bereits enthaltenen Skript *"set_disp_br.sh"* kann auch ein beliebiges anderes Skript verwendet werden. Dessen Pfad ist entweder absolut oder relativ zum Verzeichnis *"/home/plc/bin"* anzugeben.

Der am Eingang *BRIGHTNESS* des Funktionsbaustein *HMI_SET_DISPLAY_BRIGHTNESS* angegebene Helligkeitswert wird dem Skript bei seinem Aufruf als Parameter *"$1"* übergeben. Die Standardimplementierung des Skripts schreibt diesen als *"$1"* übergebenen Helligkeitswert ebenfalls direkt auf den Treibereintrag "*/sys/devices/platform/mx3_sdc_fb/backlight/mx3fb-bl/brightness"* und entspricht damit funktionell der internen Implementierung des Funktionsbausteins innerhalb der SPS-Firmware:

```
echo  $1  >  /sys/devices/platform/mx3_sdc_fb/backlight/mx3fb-bl/brightness
```

Nach Löschen oder Auskommentieren des Eintrages *"CmdSetDispBr="* benutzt die SPS-Firmware wieder die interne Standardimplementierung des Bausteins *HMI_SET_DISPLAY_BRIGHTNESS*.

Hinweis: Das Setzen der Display-Helligkeit mittels *HMI_SET_DISPLAY_BRIGHTNESS* ist nur möglich, wenn der Treibereintrag *"bl_power"* auf 0 gesetzt ist (siehe oben).

6.9 Impulsgenerator

Das PLCcore-iMX35 besitzt 1 Pulsausgang (PWMO) zur Ausgabe von PWM-Signalfolgen. Der Pulsausgang muss vor der Nutzung zunächst mit Hilfe des Funktionsbausteins *"PTO_PWM"* parametriert werden (siehe *"SYS TEC spezifische Erweiterungen für OpenPCS / IEC 61131-3"*, Manual-Nr.: L 1054).

6.9.1 PWM Signal Generierung

Nach dem Starten des Impulsgenerators übernimmt dieser die Steuerung des zugehörigen Ausgangs. Nach der Deaktivierung des Impulsgenerators nimmt der Ausgang den Wert an, den er vor der Deaktivierung hatte. Wird der Impulsgenerator deaktiviert während am PWM-Ausgang ein High-Pegel anliegt, verbleibt der PWM-Ausgang nach der Deaktivierung ebenfalls auf High. Liegt dagegen zum Zeitpunkt der Deaktivierung ein Low-Pegel an, verbleibt der PWM-Ausgang auf Low.

Tabelle 13: Zuordnung zwischen Impulskanälen und Ausgängen

Impulskanal	Impulsausgang
P0	PWMO

Der Impulsgenerator besitzt einen 16 bit Counter. Die minimale Zykluszeit beträgt 100 us (10 kHz), die maximale Zykluszeit 65 ms (16 Hz).

6.9.2 PWM Sound Generierung

Das PLCcore-iMX35 besitzt einen für die Sounderzeugung optimierten PWM-Ausgang. Sounds können mit Hilfe des Funktionsbausteins "PTO_PWM" erzeugt und über den Beeper des Baseboards ausgegeben werden.

Bei der Verwendung von "PTO_PWM" für die Sounderzeugung muss ein Tastverhältnis von 50% verwendet werden. Dem Funktionsbaustein wird beim Aufruf daher eine Pulslänge von 50% der Periodendauer übergeben.

Die folgende Abbildung zeigt exemplarisch die Verwendung von "PTO_PWM" zur Sounderzeugung unter OpenPCS.

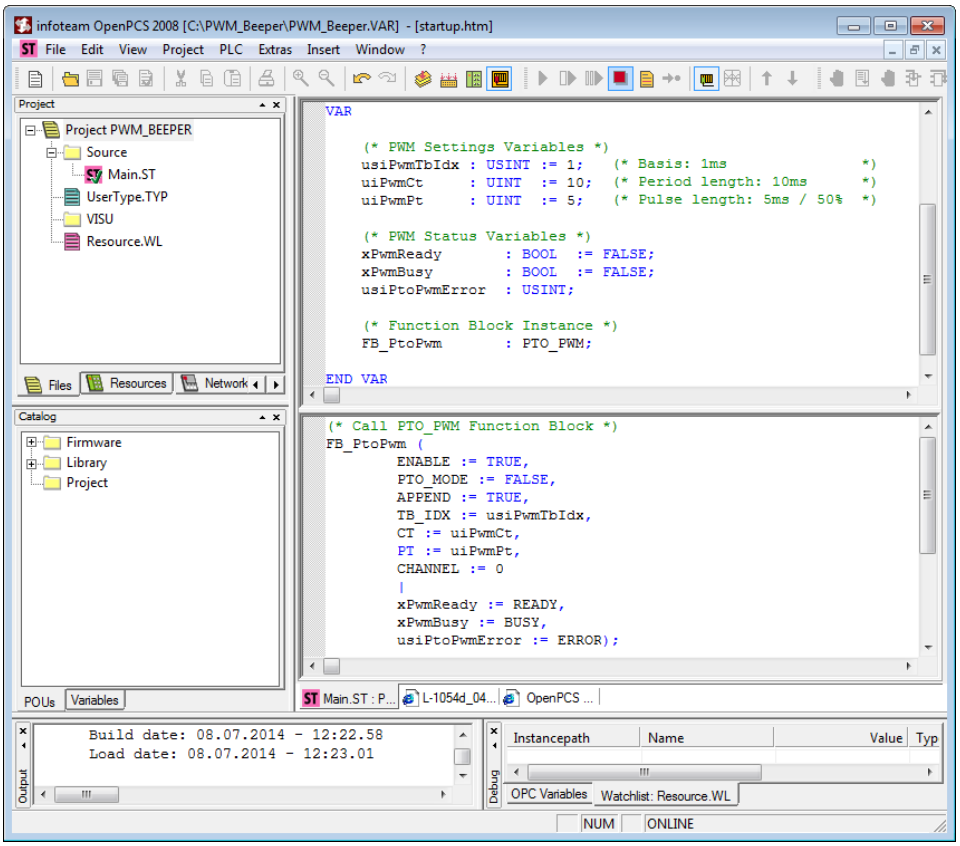


Bild 10: Sounderzeugung mit "PTO_PWM"

7 Konfiguration und Administration des PLCcore-iMX35

7.1 Systemvoraussetzungen und erforderliche Softwaretools

Zur Administration des PLCcore-iMX35 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 angebotene USB-RS232 Adapter Kabel (Bestellnummer 3234000, siehe Abschnitt 4.4.1), 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-iMX35 von einem Windows-PC aus. Das Vorgehen auf einem Linux-PC ist analog.

Zur Administration des PLCcore-iMX35 sind folgende Softwaretools erforderlich:

Terminalprogramm

Ein Terminalprogramm ermöglicht die Kommunikation mit der **Kommando-Shell** des PLCcore-iMX35 über eine **serielle RS232-Verbindung an COM0 des PLCcore-iMX35**. Diese ist Voraussetzung für die im Abschnitt 7.3 beschriebene Ethernet-Konfiguration des PLCcore-iMX35. 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-iMX35 über eine **Ethernet-Verbindung an ETH0 des PLCcore-iMX35**. Voraussetzung für die Verwendung eines Telnet-Clients ist eine abgeschlossene Ethernet-Konfiguration des PLCcore-iMX35 gemäß Abschnitt 7.3. Alternativ zum Telnet-Client können auch sämtliche Kommandos über ein Terminalprogramm (an COM0 des PLCcore-iMX35) 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-iMX35 (ETH0) und dem PC. Dies erlaubt beispielsweise das **Editieren von Konfigurationsdateien**, indem diese zunächst vom PLCcore-iMX35 auf den PC übertragen, dort mit einem Editor bearbeitet und anschließend wieder zurück auf das PLCcore-iMX35 geschrieben werden. Der Download von Dateien auf das PLCcore-iMX35 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.

TFTP-Server

Der TFTP-Server ist lediglich zum Update des Linux-Images auf dem PLCcore-iMX35 erforderlich. Als TFTP-Server eignet sich die Freeware *"TFTPD32"* (Download unter: http://tftpd32.jounin.net). Das Programm besteht lediglich aus einer einzelnen EXE-Datei, die keine Installation erfordert und sofort gestartet werden kann.

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

Im Standardbetriebsmodus startet der Bootloader "U-Boot" bei Reset (bzw. Power-on) autark das Linux-Betriebssystem des Moduls, das dann wiederum das Laden aller weiteren Softwarekomponenten bis hin zur Ausführung des SPS-Programms übernimmt (siehe Abschnitt 6.2). Für Servicezwecke wie beispielsweise die Konfiguration der Ethernet-Schnittstelle (siehe Abschnitt 7.3) oder zum Update des Linux-Images (siehe Abschnitt 7.15.2) ist es erforderlich, diesen Autostart-Mechanismus zu unterbinden und stattdessen zum "U-Boot" Kommandoprompt zu wechseln (Konfigurationsmodus).

Der automatische Start des Linux-Betriebssystems ist an die **gleichzeitige Erfüllung** verschiedener Bedingungen geknüpft ("UND-Verknüpfung"). Dementsprechend ist es zur Unterdrückung autarken Linux-Starts ausreichend, eine dieser Bedingungen **nicht zu erfüllen**.

Im Detail werden durch den Bootloader "U-Boot" die in Tabelle 14 aufgelisteten Voraussetzungen geprüft, von denen alle Bedingungen für ein autarkes Booten des Linux-Images erfüllt sein müssen.

Tabelle 14: Voraussetzungen zum Booten von Linux

Nr.	Bedingung	Bemerkung
1	Anschluss "/BOOT" = High (Taster S602 am Development-board **nicht** gedrückt)	Der Linux-Autostart wird nur freigegeben, wenn Signal "/BOOT" am PLCcore-iMX35 auf dem H-Pegel ("/BOOT“ ist nicht aktiv) liegt. Die Position des Anschlusses "/BOOT" am Steckverbinder des Moduls ist im Hardware Manual ECUcore-iMX35 (Manual-Nr.: L-1570) definiert.
2	Kein Abbruch der Autoboot-Prozedur über COM0 des PLCcore-iMX35	Sind die vorangegangenen Bedingungen erfüllt, überprüft "U-Boot" nach Reset für ca. 1 Sekunde die serielle Schnittstelle COM0 des PLCcore-iMX35 auf den Empfang eines SPACE-Zeichens (ASCII 20H). Wird innerhalb dieser Zeit ein entsprechendes Zeichen empfangen, unterbindet "U-Boot" den Linux-Bootvorgang und aktiviert stattdessen seinen eigenen Kommandoprompt.

Gemäß Tabelle 14 wird unter folgenden Voraussetzungen der Linux-Bootvorgang nach Reset (z.B. Taster S601 am Developmentboard) unterbunden und stattdessen der "U-Boot"-Kommandoprompt aktiviert:

(1) /BOOT = "Low" Developmentboard: "/BOOT" = Taster S602

- ODER -

(2) Empfang eines SPACE-Zeichens (ASCII 20H) innerhalb 1 Sekunde nach Reset

Nach dem Betätigen des Reset-Tasters (z.B. Taster S601 am Developmentboard) meldet sich der "U-Boot" Kommandoprompt.

Die Kommunikation mit dem Bootloader "U-Boot" erfolgt ausschließlich über die serielle Schnittstelle COM0 des PLCcore-iMX35. 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 11):

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

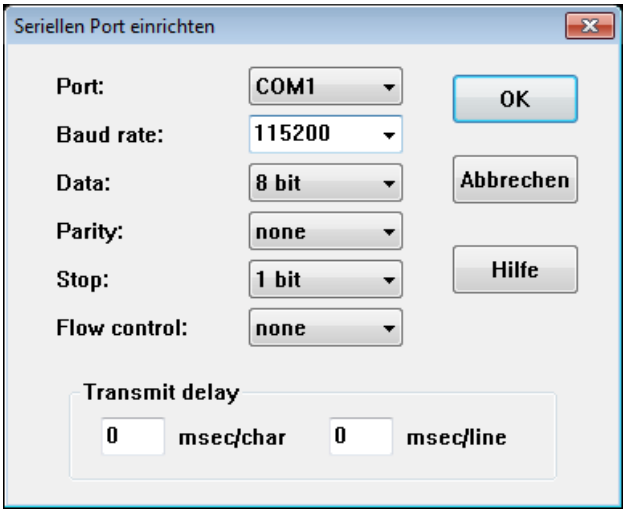


Bild 11: Terminaleinstellungen am Beispiel von "TeraTerm"

7.3 Ethernet-Konfiguration des PLCcore-iMX35

Die zentrale Ethernet-Konfiguration des PLCcore-iMX35 erfolgt im Bootloader "U-Boot" und wird von hier für alle anderen Softwarekomponenten übernommen (Linux, SPS-Firmware, HTTP-Server usw.) Die Ethernet-Konfiguration erfolgt über die serielle Schnittstelle COM0. **Dazu ist wie im Abschnitt 7.2 beschrieben der "U-Boot"-Kommandoprompt zu aktivieren.** Tabelle 15 listet die zur Ethernet-Konfiguration des PLCcore-iMX35 notwendigen U-Boot-Kommandos auf.

Tabelle 15: "U-Boot" Kommandos zur Konfiguration des PLCcore-iMX35

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-iMX35. Die IP-Adresse ist vom Netzwerkadministrator festzulegen.
Netzwerkmaske	setenv netmask <xxx.xxx.xxx.xxx>	Dieses Kommando setzt die Netzwerkmaske des PLCcore-iMX35. Die Netzwerkmaske ist vom Netzwerkadministrator festzulegen.
Gateway-Adresse	setenv gatewayip <xxx.xxx.xxx.xxx>	Dieses Kommando definiert die IP-Adresse des vom PLCcore-iMX35 zu benutzenden Gateways. Die Gateway-Adresse ist vom Netzwerkadministrator festzulegen. **Hinweis:** Befinden sich PLCcore-iMX35 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.
Speichern der Konfiguration	saveenv	Dieses Kommando speichert die aktuellen Einstellungen im Flash des PLCcore-iMX35.

Die modifizierten Einstellungen können durch die Eingabe von *"printenv"* am "U-Boot" Kommandoprompt nochmals überprüft werden. Die aktuellen Einstellungen werden durch das Kommando

saveenv

persistent im Flash des PLCcore-iMX35 gespeichert. Die Änderungen werden mit dem nächsten Reset des PLCcore-iMX35 übernommen.

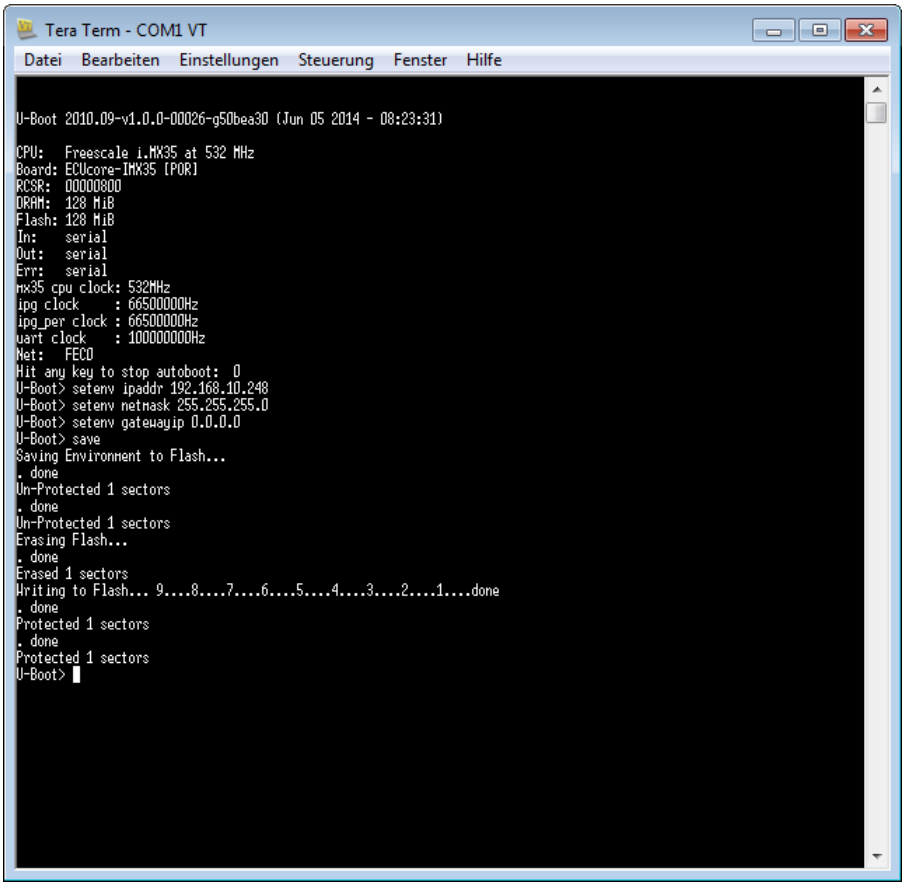


Bild 12: Ethernet-Konfiguration des PLCcore-iMX35

Nach Abschluss der Konfiguration sind gemäß Abschnitt 7.2 die Voraussetzungen für einen Linux-Autostart wieder herzustellen.

Nach Reset (z.B. Taster S601 am Developmentboard) startet das Modul mit den aktuellen Einstellungen.

Hinweis: Nach Abschluss der Konfiguration ist die serielle Verbindung zwischen PC und PLCcore-iMX35 nicht mehr erforderlich.

7.4 SPS-Konfiguration des PLCcore-iMX35

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-iMX35 erfolgen. Beim Einsatz des PLCcore-iMX35 auf dem Development Kit sind grundlegende Einstellungen alternativ auch über lokale Bedienelemente möglich (siehe Abschnitt 7.4.2).

Für die Konfiguration des PLCcore-iMX35 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-iMX35 einzugeben, z.B. *"http://192.168.10.248"*. Bild 13 verdeutlicht den Aufruf der Konfigurationsseite für das PLCcore-iMX35 im WEB-Browser.

In der Standardeinstellung (Werkseinstellungen) erfordert die Konfiguration des PLCcore-iMX35 über das WEB-Frontend eine Benutzeranmeldung, um unbefugte Zugriffe zu unterbinden. Dazu sind Nutzername und Passwort in dem entsprechenden Anmeldedialog einzugeben (siehe Bild 13). Bei Auslieferung des Moduls ist folgendes Nutzerkonto vorkonfiguriert (siehe auch Abschnitt 7.8):

User: PlcAdmin
Passwort: Plc123

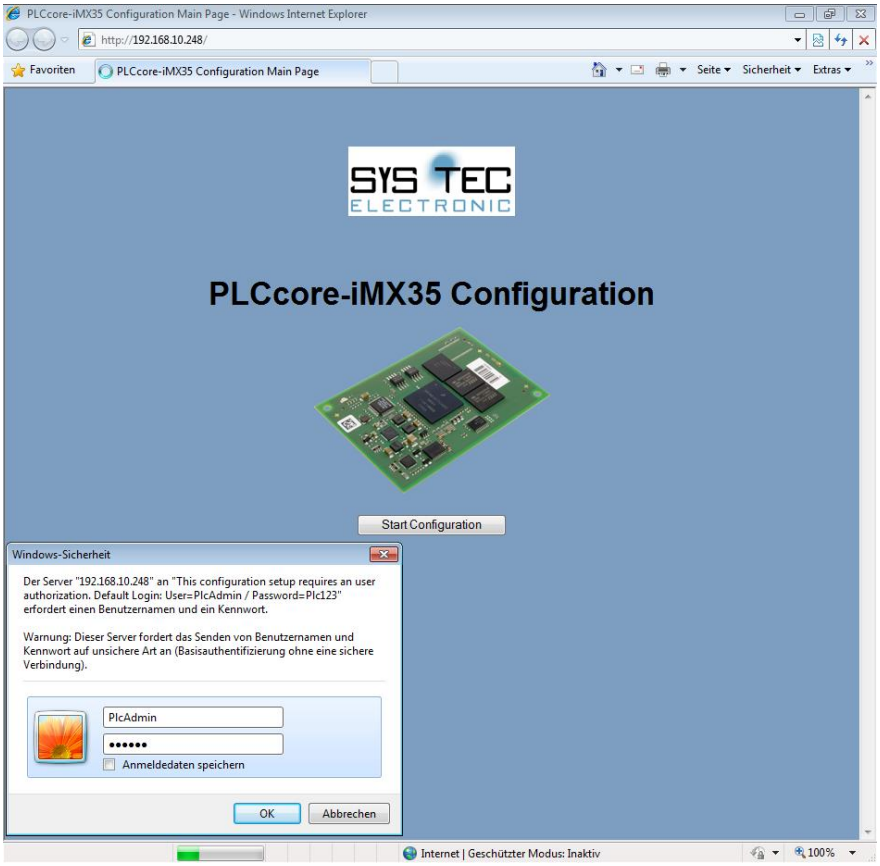


Bild 13: Anmeldedialog des WEB-Frontend

Sämtliche Konfigurationseinstellungen für das PLCcore-iMX35 erfolgen dialogbasiert und werden durch Betätigen der Schaltfläche "Save Configuration" in die Datei ***"/home/plc/bin/plccore-imx35.cfg"*** des PLCcore-iMX35 übernommen (siehe auch Abschnitt 7.4.3). Nach Reset (z.B. Taster S601 am Developmentboard) startet das PLCcore-iMX35 mit den aktuellen Einstellungen. Bild 14 zeigt die Konfiguration des PLCcore-iMX35 über das WEB-Frontend.

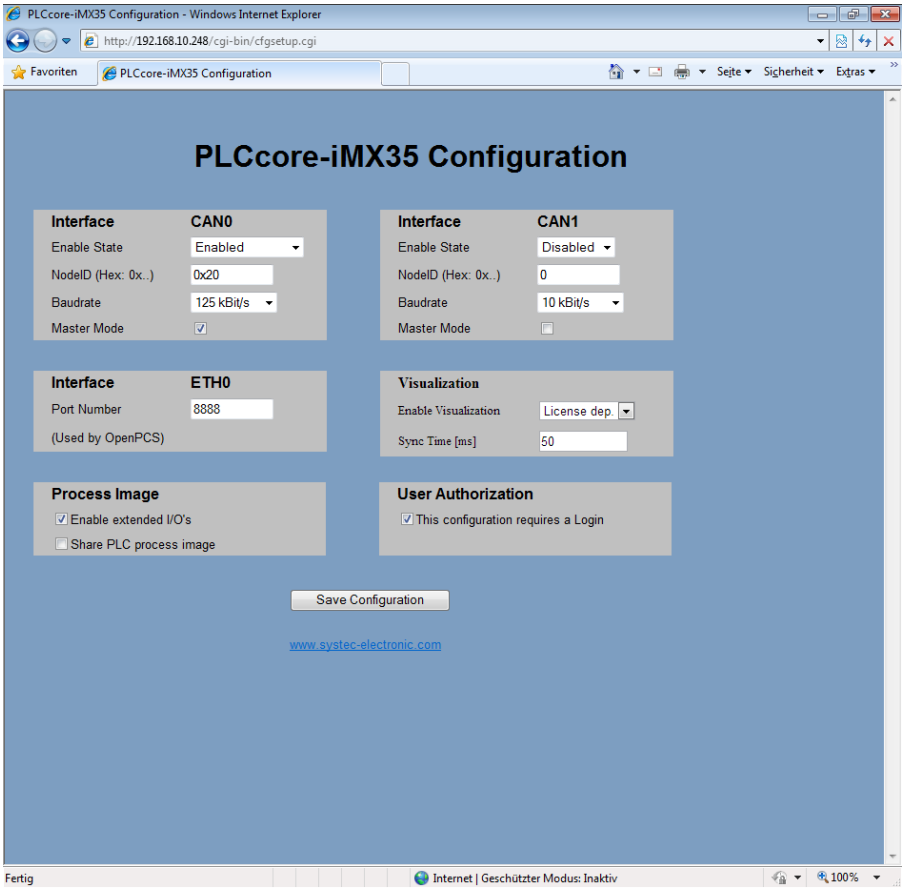


Bild 14: SPS-Konfiguration über WEB-Frontend

Bei der Auswahl von "DIP/Hex-Switch" für den Enable State des Interfaces CAN0 erfolgt die Konfiguration dieser Schnittstelle über die lokalen Bedienelemente des Development Kit PLCcore-iMX35 (siehe Abschnitt 7.4.2).

In der Standardeinstellung (Werkseinstellungen) ist das PLCcore-iMX35 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-iMX35.cfg"*** angegebene Benutzername akzeptiert (Eintrag *"User="* in Sektion *"[Login]"*, siehe Abschnitt 7.4.3). Die Vorgehensweise zum Ändern des Passwortes für die Nutzeranmeldung beschreibt Abschnitt 7.11. Um einem anderen Benutzer die Konfiguration des Moduls zu ermöglichen, ist zunächst das entsprechende Nutzerkonto wie im Abschnitt 7.10 beschrieben anzulegen. Anschließend ist der neue Benutzername in der Konfigurationsdatei ***"/home/plc/bin/plccore-imx35.cfg"*** einzutragen. Durch Löschen des Eintrages *"User="* in Sektion *"[Login]"* (siehe 7.4.3) 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 14) wird ein freier Zugriff auf die Konfiguration ohne vorherige Nutzeranmeldung ermöglicht.

7.4.2 SPS-Konfiguration über Bedienelemente des Development Kit PLCcore-iMX35

Die **Konfiguration über die Bedienelemente** des Developmentboards für das PLCcore-iMX35 **ist die Voreinstellung bei Auslieferung** des Development Kit PLCcore-iMX35 und ermöglicht eine einfache Inbetriebnahme des Moduls mit Nutzung der CAN-Schnittstelle CAN0. Aufgrund der begrenzten Anzahl an vorhandenen Schalterelementen ist jedoch nur eine eingeschränkte Grundkonfiguration für CAN0 möglich. Die Nutzung der Schnittstelle CAN1 erfordert die im Abschnitt

7.4.1 beschriebene Konfiguration über das WEB-Frontend, dass darüber hinaus noch weitergehende Einstellungen erlaubt.

Hinweis: Die Konfiguration der Schnittstelle CAN0 ist nur dann mit Hilfe lokaler Bedienelemente möglich, wenn über das WEB-Forntend als Enable State für CAN0 die Einstellung "DIP/Hex-Switch" aktiviert ist (Werkseinstellung). Andernfalls haben die im WEB-Frontend vorgenommenen Einstellungen Vorrang gegenüber den Bedienelementen.

Knotenadresse CAN0: Die Knotenadresse für das Interface CAN0 wird über die beiden Hexcodier-Schalter S608 und S610 auf dem Developmentboard für das PLCcore-iMX35 eingestellt:

S608: High-Teil der Knotenadresse
S610: Low-Teil der Knotenadresse

Beispiel: S608=2 / S610=0 → resultierende Knotenadresse = 20 Hex.

Bitrate CAN0: Die Bitrate für das Interface CAN0 wird über die Bitpositionen 1-3 des DIP-Schalters S609 auf dem Developmentboard für das PLCcore-iMX35 eingestellt. Tabelle 16 listet die Kodierung der unterstützten Bitraten auf.

Tabelle 16: Einstellung der Bitrate für CAN0 über DIP-Schalter

Bitrate [kBit/s]	DIP1	DIP2	DIP3
10	OFF	OFF	ON
20	ON	OFF	OFF
50	ON	OFF	ON
125	OFF	OFF	OFF
250	OFF	ON	ON
500	OFF	ON	OFF
800	ON	ON	ON
1000	ON	ON	OFF

Master-Modus CAN0: Der Master-Modus wird über die Bitpositionen 4 des DIP-Schalter S609 auf dem Developmentboard für das PLCcore-iMX35 aktiviert:

DIP4 = OFF: SPS ist NMT-Slave
DIP4 = ON: SPS ist NMT-Master

7.4.3 Aufbau der Konfigurationsdatei "plccore-imx35.cfg"

Die Konfigurationsdatei ***"/home/plc/bin/plccore-imx35.cfg"*** ermöglicht eine umfassende Konfiguration des PLCcore-iMX35. 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 17 enthält eine Auflistung der Konfigurationseinträge. Die Einträge der Sektion "*[CAN0]*" haben Vorrang gegenüber den Einstellungen an den Bedienelementen (siehe Abschnitt 7.4.2).

Tabelle 17: Konfigurationseinträge der CFG-Datei

Section	Entry	Value	Bedeutung
[CAN0]	Enabled	-1, 0, 1	-1: Interface CAN0 ist aktiviert, Konfiguration erfolgt über Bedienelemente des Developmentboards (Werkseinstellung, siehe Abschnitt 7.4.2) 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	0, 1	0: Interface CAN1 ist deaktiviert 1: Interface CAN1 ist aktiviert, Konfiguration erfolgt über nachstehende Einträge dieser Konfigurationsdatei
	NodeID	1 … 127 bzw. 0x01 … 0x7F	Knotennummer für Interface CAN1 (dezimal oder hexadezimal mit Präfix "0x")
	Baudrate	10, 20, 50, 125, 250, 500, 800, 1000	Bitrate für Interface CAN1
	MasterMode	0, 1	1: Master-Modus ist aktiviert 0: Master-Modus ist deaktiviert
[ETH0]	PortNum	Default Portnummer: 8888	Portnummer zur Kommunikation mit dem Programmier-PC sowie zum Programmdownload (nur für PLCcore-iMX35/Z5, Bestellnummer 4001025/Z5 oder 3390085/Z5)
[ProcImg]	EnableExtIo	0, 1	0: nur on-board I/O's des PLCcore-iMX35 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	1: Visualisierung ist aktiviert – erfordert einen gültigen Lizenzschlüssel). 0: Visualisierung ist deaktiviert -1: Visualisierung wird aktiviert, wenn sie von der Lizenz unterstützt wird, andernfalls ist die Visualisierung

			deaktiviert (Auto-Mode)
	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
	CmdSetDispBr	Preset but disabled: ./set_disp_br.sh	Optionales Shell-Skript zum Setzen der Display-Helligkeit (siehe Abschnitt 6.8.3)
[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-iMX35 angelegte Benutzer (siehe Abschnitt 7.10) zur Konfiguration über das WEB-Frontend anmelden.

Die Konfigurationsdatei ***"/home/plc/bin/plccore-imx35.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 Konfiguration des A/D-Wandlers

Der A/D-Wandler des PLCcore-iMX35 verfügt über 4 Kanäle. Die zuletzt gewandelten Werte können über den sysfs-Eintrag ***"/sys/bus/spi/devices/spi0.1/channelX"*** (X steht hierbei für den jeweiligen Kanal) abgefragt werden.

Im SPS-Programm werden die gewandelten Werte über den I/O-Treiber ermittelt. Üblicherweise entspricht die im SPS-Programm verwendete Kanalnummer dem Kanal am A/D-Wandler. Kanal 0 im SPS-Programm entspricht folglich Kanal 0 am A/D-Wandler.
Dieses Mapping kann mit Hilfe einer Konfigurationsdatei an die eigenen Bedürfnisse angepasst werden. So kann z.B. ein SPS-Programm eines anderen Moduls ohne größere Anpassungen auf das PLCcore-iMX35 portiert werden.

Die Konfiguration des Kanal-Mapping erfolgt über die Datei ***"/home/etc/ADCMapping.conf"***. Soll ein anderer Pfad verwendet werden, so kann dieser über die Umgebungsvariable **"ADC_MAPPING_FILE"** definiert werden. Existiert weder die Konfigurationsdatei im o.g. Pfad, noch die Umgebungsvariable bzw. die darüber referenzierte Konfigurationsdatei, so erfolgt kein Mapping durch den I/O-Treiber. Damit entspricht die im SPS-Programm verwendete Kanalnummer der des A/D-Wandlers.

Tabelle 18 beschreibt den Aufbau der Konfigurationsdatei. Das dabei verwendete Mapping entspricht dem Auslieferungszustand des PLCcore-iMX35.

Tabelle 18: Kanal-Mapping des A/D-Wandlers

Kanal	Wert	Konfigurationsdatei	Bedeutung
0	2	AIN0=2	Kanal 0 im SPS-Programm entspricht Kanal 2 am ADC
1	3	AIN1=3	Kanal 1 im SPS-Programm entspricht Kanal 3 am ADC
2	0	AIN2=0	Kanal 2 im SPS-Programm entspricht Kanal 0 am ADC
3	1	AIN3=1	Kanal 3 im SPS-Programm entspricht Kanal 1 am ADC

Existiert sowohl die Konfigurationsdatei im Standardpfad, als auch die über "ADC_MAPPING_FILE" referenzierte Datei, so erfolgt das Mapping entsprechend der Datei im "ADC_MAPPING_FILE".

7.6 Boot-Konfiguration des PLCcore-iMX35

Das PLCcore-iMX35 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-iMX35 automatisiert werden. Das Skript lässt sich im FTP-Client *"WinSCP"* (siehe Abschnitt 7.1) mit Hilfe der Taste *"F4"* bzw. der Schaltfläche *"F4 Edit"* direkt auf dem PLCcore-iMX34 bearbeiten.

7.7 Auswahl der zu startenden Firmware-Variante

Das PLCcore-iMX35 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 6.3). Die Auswahl der zu verwendenden Firmware-Varianten erfolgt im Startskript ***"/home/etc/autostart"***. Standardmäßig wird hierzu die im Bootloader "U-Boot" festgelegte *"BoardID"* des Moduls ausgewertet. Tabelle 19 listet die Zuordnung von Firmware-Variante und BoardID auf.

Tabelle 19: Zuordnung von BoardID und Firmware-Variante für das PLCcore-iMX35

BoardID	Firmware-Variante	Bemerkung
1010004	plccore-imx35-z4	**PLCcore-iMX35/Z4 (CANopen, ohne Target-Visualisierung)** Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
1010005	plccore-imx35-z5	**PLCcore-iMX35/Z5 (Ethernet, ohne Target-Visualisierung)** Kommunikation mit Programmier-PC via UDP-Protokoll (Interface ETH0)
1010014	plccore-imx35-hmi-z4	**PLCcore-iMX35-HMI/Z4 (CANopen, mit Target-Visualisierung)** Kommunikation mit Programmier-PC via CANopen-Protokoll (Interface CAN0)
1010015	plccore-imx35-hmi-z5	**PLCcore-iMX35-HMI/Z5 (Ethernet, mit Target-Visualisierung)** Kommunikation mit Programmier-PC via UDP-Protokoll (Interface ETH0)

Die Konfiguration der BoardID erfolgt über die serielle Schnittstelle COM0. **Dazu ist wie im Abschnitt 7.2 beschrieben der "U-Boot"-Kommandoprompt zu aktivieren.** Das Festlegen der jeweiligen BoardID erfolgt mit dem "U-Boot"-Kommando *"set boardid"* unter Angabe der in Tabelle 19 aufgeführten, zugehörigen Nummer, z.B.:

```
setenv  boardid  1010005
```

Die modifizierte Einstellung kann durch die Eingabe von "*printenv*" am "U-Boot" Kommandoprompt nochmals überprüft werden. Die aktuelle Auswahl wird durch das Kommando

saveenv

persistent im Flash des PLCcore-iMX35 gespeichert. Bild 15 veranschaulicht die Konfiguration der BoardID.

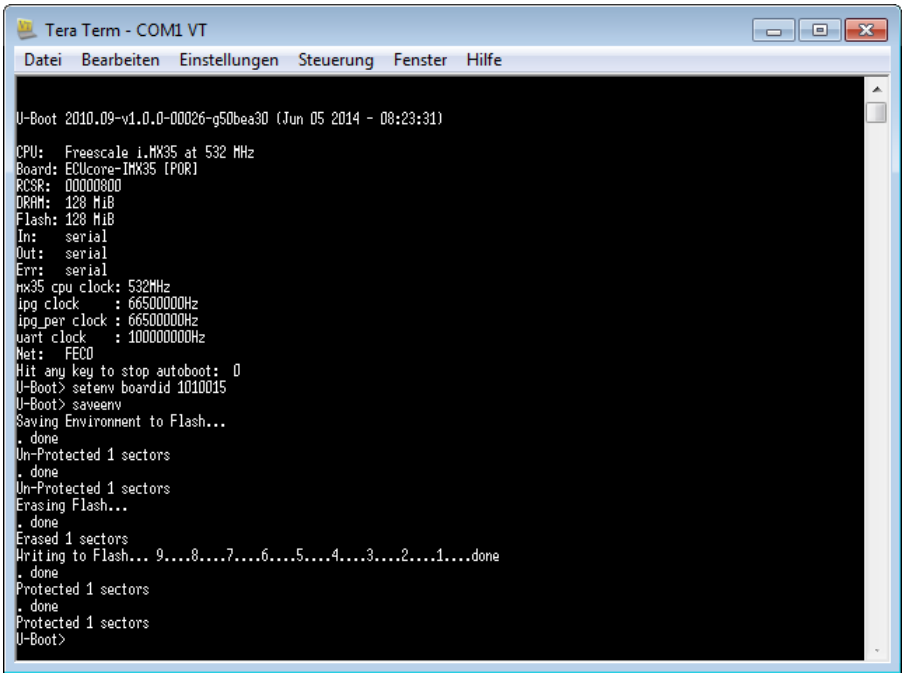


Bild 15: Auswahl der zu startenden Firmware-Variante für das PLCcore-iMX35

Nach Abschluss der Konfiguration sind gemäß Abschnitt 7.2 die 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=plccore-iMX35-z5
```

7.8 Vordefinierte Nutzerkonten

Bei der Auslieferung des PLCcore-iMX35 sind die in Tabelle 20 aufgelisteten Benutzerkonten vordefiniert. Diese erlauben eine Anmeldung an der Kommando-Shell (serielle RS232-Verbindung oder Telnet) und am FTP-Server des PLCcore-iMX35.

Tabelle 20: Vordefinierte Benutzerkonten des PLCcore-iMX35

Benutzername	Passwort	Bemerkung
PlcAdmin	Plc123	vordefiniertes Benutzerkonto zur Administration des PLCcore-iMX35 (Konfiguration, Nutzerverwaltung, Softwareupdates usw.)
root	Sys123	Haupt-Benutzerkonto ("root") des PLCcore-iMX35

7.9 Anmeldung am PLCcore-iMX35

7.9.1 Anmeldung an der Kommando-Shell

Die Administration des PLCcore-iMX35 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 **COM0** des PLCcore-iMX35, 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-iMX35 möglich.

Um sich über den in Windows standardmäßig enthaltenen Telnet-Client am PLCcore-iMX35 anzumelden, ist der Befehl "*telnet*" unter Angabe der in Abschnitt 7.2 festgelegten IP-Adresse für das PLCcore-iMX35 aufzurufen, z.B.

```
telnet  192.168.10.248
```

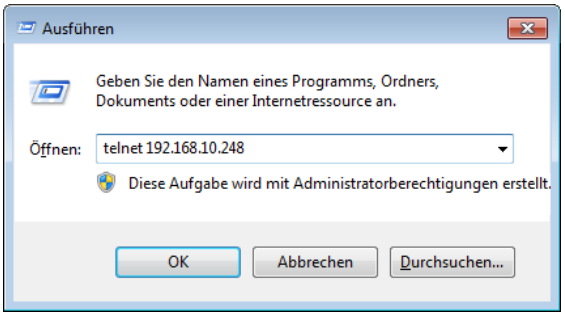


Bild 16: Aufruf des Telnet-Clients unter Windows

Innerhalb des Terminal-Fensters (bei Verbindung über COM0) bzw. des Telnet-Fensters (bei Verwendung von ETH0) ist die Anmeldung am PLCcore-iMX35 möglich. Bei Auslieferung des Moduls ist zur Administration des PLCcore-iMX35 folgendes Nutzerkonto vorkonfiguriert (siehe auch Abschnitt 7.8):

User: *PlcAdmin*
Passwort: *Plc123*

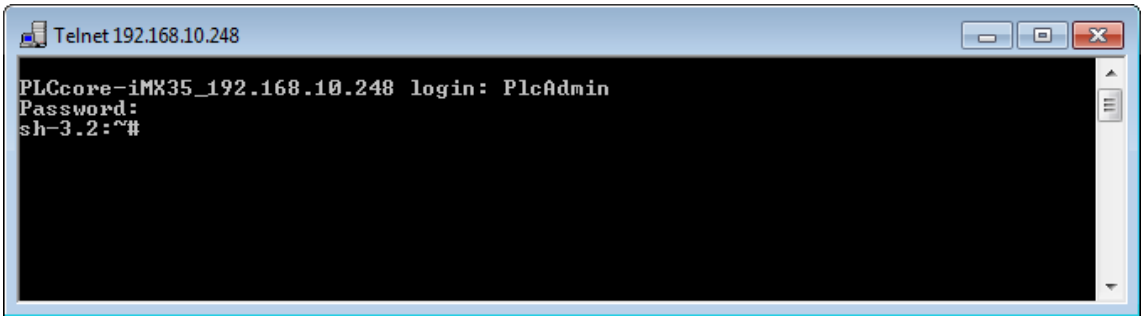


Bild 17: Anmeldung am PLCcore-iMX35

Bild 17 verdeutlicht am Beispiel die Anmeldung am PLCcore-iMX35 mit Hilfe des in Windows standardmäßig enthaltenen Telnet-Clients.

7.9.2 Anmeldung am FTP-Server

Das PLCcore-iMX35 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 7.9.1 beschriebenen Anmeldung an der Kommando-Shell des PLCcore-iMX35 durchzuführen. Anschließend ist im Telnet- bzw. Terminal-Fenster folgendes Kommando einzugeben:

```
pureftp
```

Bild 18 verdeutlicht am Beispiel das Starten des FTP-Servers.

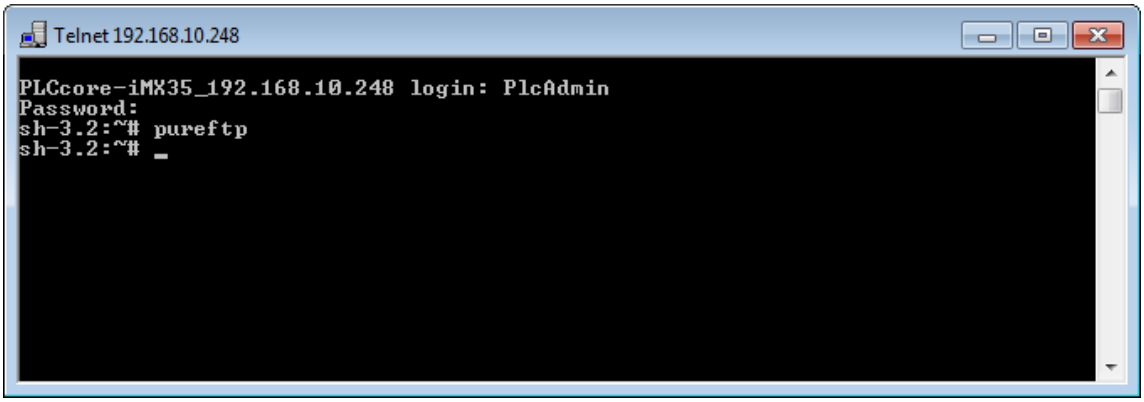


Bild 18: Starten des FTP-Servers

Hinweis: Durch einfügen des Kommandos *"pureftp"* in das Startskript ***"/home/etc/autostart"*** lässt sich der Aufruf des FTP-Servers beim Booten des PLCcore-iMX35 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 19), in dem folgende Einstellungen vorzunehmen sind:

File protocol: FTP
Host name: die im Abschnitt 7.3 festgelegte IP-Adresse für das PLCcore-iMX35
User name: *PlcAdmin* (für vorkonfiguriertes Nutzerkonto, siehe Abschnitt 7.8)
Password: *Plc123* (für vorkonfiguriertes Nutzerkonto, siehe Abschnitt 7.8)

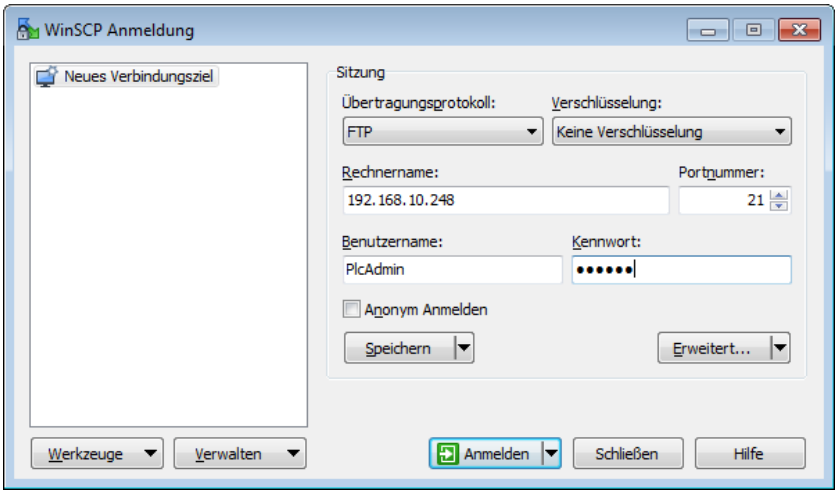


Bild 19: Login-Einstellungen für WinSCP

Nach dem Betätigen der Schaltfläche *"Login"* meldet sich der FTP-Client am PLCcore-iMX35 an und listet im rechten Fenster den aktuellen Inhalt des Verzeichnisses *"/home"* auf. Bild 20 zeigt den FTP-Client *"WinSCP"* nach der erfolgreichen Anmeldung am PLCcore-iMX35.

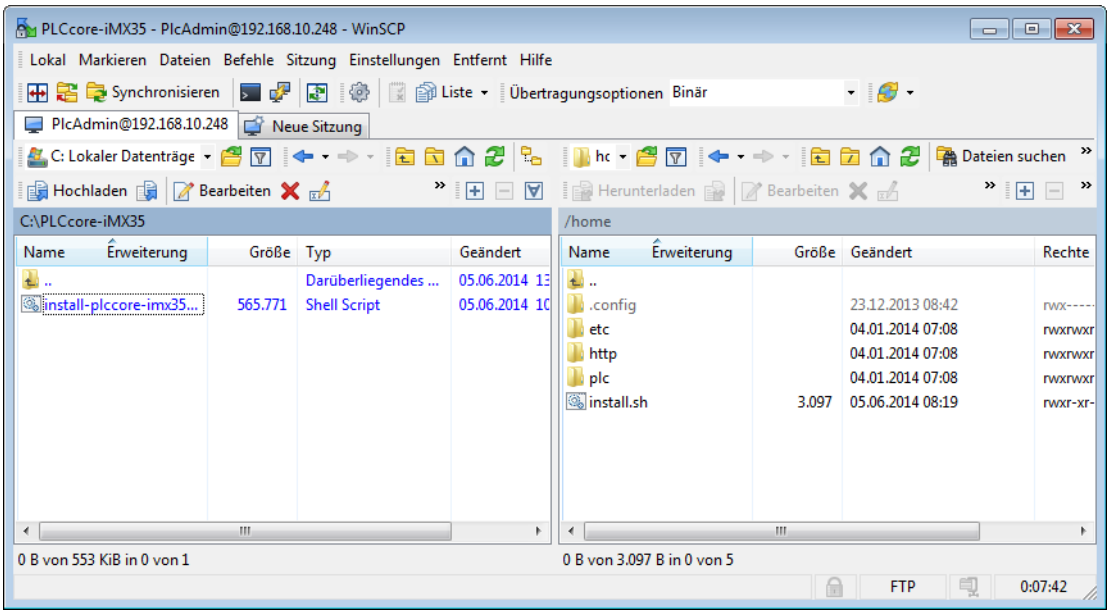


Bild 20: FTP-Client für Windows "WinSCP"

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-iMX35 bearbeiten (dazu Transfermodus *"Text"* selektieren). Mit Hilfe der Taste *"F5"* bzw. der Schaltfläche *"F5 Copy"* können Dateien zwischen dem PC und dem PLCcore-iMX35 transferiert werden, um beispielsweise Datensicherungen des PLCcore-iMX35 durchzuführen oder um Installationsdateien für Firmware-Updates auf das Modul zu übertragen (dazu Transfermodus *"Binary"* selektieren).

7.10 Anlegen und Löschen von Nutzerkonten

Das Anlegen und Löschen von Nutzerkonten erfordert zunächst die Anmeldung am PLCcore-iMX35 wie in Abschnitt 7.9.1 beschrieben.

Das **Anlegen** eines neuen Nutzerkontos erfolgt mit Hilfe des Linux-Kommandos *"adduser"*. Da es bei einem Embedded System wie dem PLCcore-iMX35 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-iMX35 ist das Linux-Kommandos *"adduser"* wie folgt anzuwenden:

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

Bild 21 verdeutlicht am Beispiel das Anlegen eines neuen Kontos auf dem PLCcore-iMX35 für den Nutzer *"admin2"*.

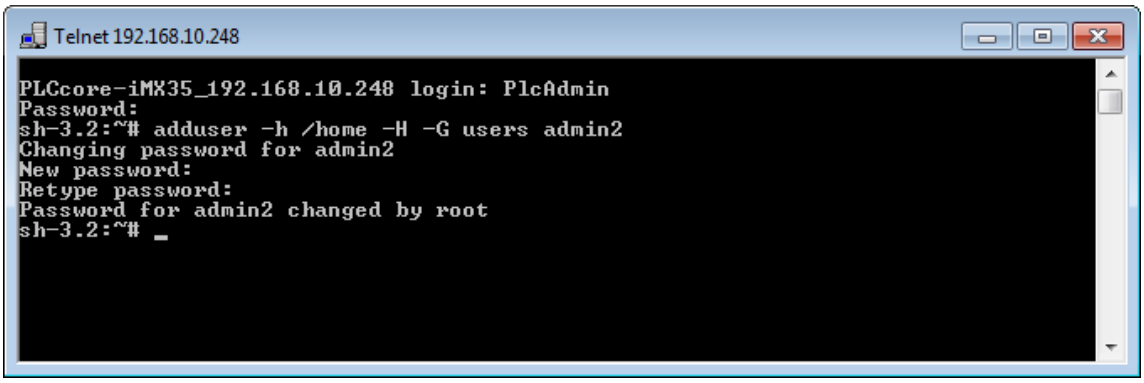


Bild 21: Anlegen eines neuen Nutzerkontos

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-imx35.cfg"* einzutragen (für Details zur Anmeldung an das WEB-Frontend siehe Abschnitte 7.4.1 und 7.4.3).

Zum **Löschen** eines vorhandenen Nutzerkontos vom PLCcore-iMX35 ist das Linux-Kommando *"deluser"* unter Angabe des Benutzernamens zu verwenden:

```
deluser  <username>
```

7.11 Passwort eines Nutzerkontos ändern

Das Ändern von Passwörtern erfordert zunächst die Anmeldung am PLCcore-iMX35 wie in Abschnitt 7.9.1 beschrieben.

Zum Ändern des Passwortes für ein auf dem PLCcore-iMX35 vorhandenen Nutzerkonto ist das Linux-Kommando *"passwd"* unter Angabe des Benutzernamens zu verwenden:

```
passwd <username>
```

Bild 22 verdeutlicht am Beispiel das Ändern des Passwortes für den Nutzer *"PlcAdmin"*.

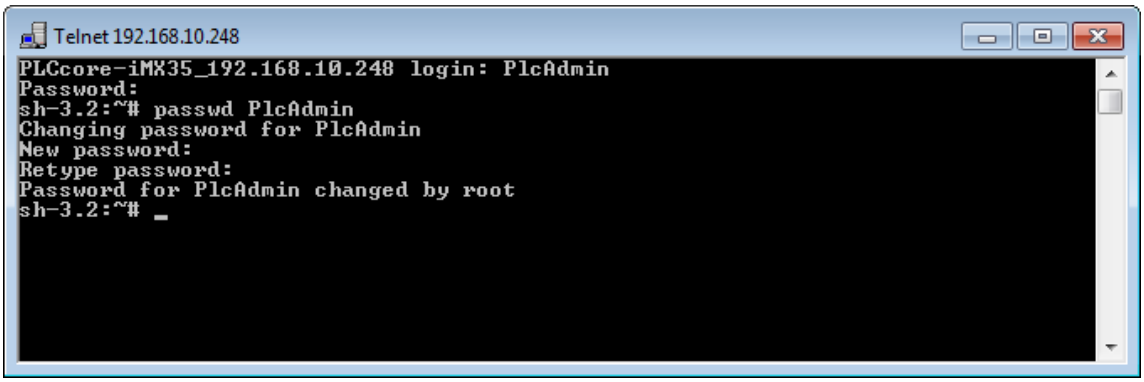


Bild 22: Passwort eines Nutzerkontos ändern

7.12 Setzen der Systemzeit

Das Setzen der Systemzeit erfordert zunächst die Anmeldung am PLCcore-iMX35 wie in Abschnitt 7.9.1 beschrieben.

Das Setzen der Systemzeit für das PLCcore-iMX35 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-iMX35 übernommen.

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

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

Beispiel:

```
date   2014.06.05-14:00:35
          |  |  |  |  |  |
          |  |  |  |  |  +--- Sekunde
          |  |  |  |  +------ Minute
          |  |  |  +--------- Stunde
          |  |  +------------ Tag
          |  +--------------- Monat
          +------------------ Jahr
```

Um die aktuelle Systemzeit des PLCcore-iMX35 wie im obigen Beispiel dargestellt auf den 2014/06/05 um 14:00:35 zu setzen, ist folgende Befehlssequenz notwendig:

```
date  2014.06.05-14:00:35
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 23 verdeutlicht das Setzen und Anzeigen der Systemzeit am Beispiel.

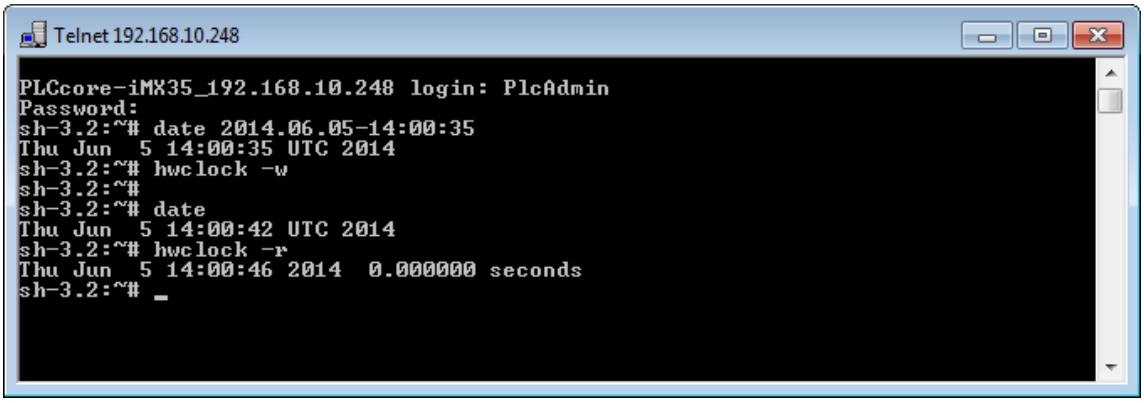


Bild 23: Setzen und Anzeigen der Systemzeit

Beim Starten des PLCcore-iMX35 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.

7.13 Dateisystem des PLCcore-iMX35

Das auf dem PLCcore-iMX35 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 nur durch Neuerstellung des Linux-Images für das PLCcore-iMX35 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 ausgeschalten werden, ohne vorherigen Shutdown.

Die zur Laufzeit beschreibbaren Zweige des Dateisystems listet Tabelle 21 auf. Hinter dem Verzeichnis **"/home"** verbirgt sich eine Flash-Disk, die einen Teil des on-baord Flash-Speichers des PLCcore-iMX35 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 7.15.1).

Tabelle 21: Dateisystemkonfiguration des PLCcore-iMX35

Zweig	Nutzbare Größe	Beschreibung
/home	95516 kByte	Flash-Disk, zur persistenten Ablage vom Anwender modifizierbarer und updatefähiger Files (z.B. Konfigurations-dateien, SPS-Firmware, SPS-Programm, Dateien für Target-Visualisierung), Datenerhalt bei Spannungsausfall ist gegeben
/tmp	8192 kByte	RAM-Disk, geeignet als Zwischenpuffer für FTP-Downloads, aber kein Datenerhalt bei Spannungsausfall
/var/log	62640 kByte	RAM-Disk, wird vom System selbst zur Ablage temporärer Dateien benutzt, aber kein Datenerhalt bei Spannungsausfall
/mnt		Ziel für die Einbindung von Remote-Verzeichnissen, wird auf dem PLCcore-iMX35 in der Standardfunktionalität nicht verwendet

Die konfigurierten sowie jeweils aktuell noch verfügbaren Größen der einzelnen Dateisystemzweige können mit Hilfe des Linux-Kommandos "df" ("DiskFree") ermittelt werden (siehe Bild 24).

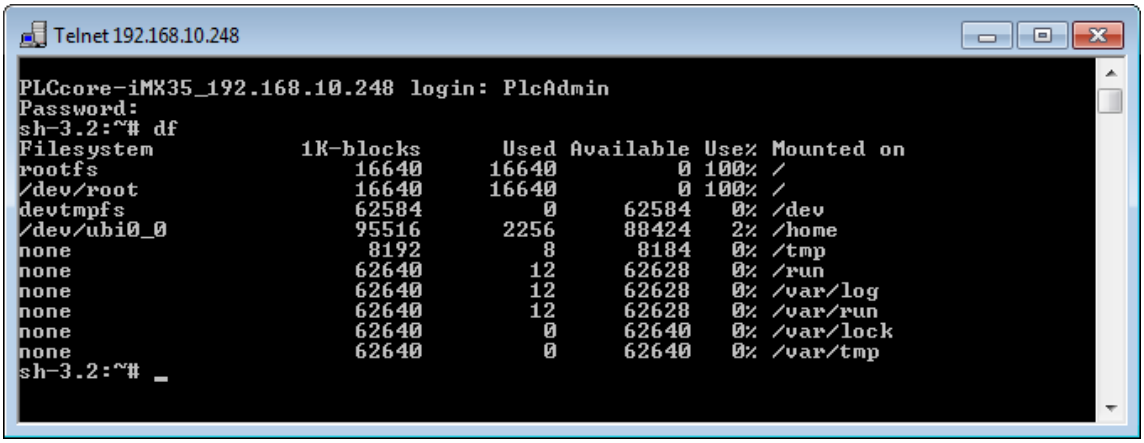


Bild 24: Anzeige von Informationen zum Dateisystem

Einzelheiten zur Anmeldung am System und zum Umgang mit der Linux-Kommando-Shell des PLCcore-iMX35 behandelt Abschnitt 7.9.

7.14 Kalibrierung des Touchscreen

Das PLCcore-iMX35 besitzt keinen eigenen on-board Touch-Controller. Für den Anschluss resistiver Touchscreens wird dementsprechend ein externer Touch Controller benötigt. Touchscreen und Touch Controller müssen vor der ersten Verwendung aufeinander abgestimmt – kalibriert – werden. Ohne Kalibrierung arbeitet der Touchscreen extrem ungenau, normalerweise ist eine korrekte Bedienung unmöglich.

7.14.1 Automatische Überprüfung der Touchscreen-Kalibrierung

Eine erfolgreiche Kalibrierung ist Voraussetzung für die Benutzung des Touchscreens. Die Geräte-Firmware kann beim Booten des SPS Systems prüfen, ob die erforderliche Kalibrierung des

Touchscreens bereits durchgeführt wurde. Als Entscheidungskriterium hierfür wird untersucht, ob die Datei *"/home/etc/pointercal"* existiert und größer als 0 Byte ist. Ist diese Bedingung nicht erfüllt, wird noch vor dem Start der SPS-Firmware das entsprechende Kalibrierprogramm *"ts_calibrate"* aufgerufen (siehe auch Abschnitt 7.14.2).

Da das PLCcore-iMX35 sowohl Displays mit als auch ohne Touchscreen unterstützt, lässt sich die automatische Überprüfung der Touchscreen-Kalibrierung durch Konfigurationseinstellungen des Module wahlweise aktivieren bzw. deaktivieren. Die entsprechende Konfiguration erfolgt über die Umgebungsvariable *"check_tscalibfile"* des Bootloaders "U-Boot". Um diese Variable zu setzen, ist zunächst wie im Abschnitt 7.2 beschrieben der "U-Boot"-Kommandoprompt zu aktivieren. Tabelle 22 listet die Kommandos zum ein- bzw. ausschalten der automatische Überprüfung auf.

Tabelle 22: Konfiguration der automatischen Überprüfung der Touchscreen-Kalibrierung

Kommando	Einstellung
setenv check_tscalibfile on saveenv	Automatische Überprüfung der Touchscreen-Kalibrierung aktiviert, falls die Datei *"/home/etc/pointercal"* nicht existiert (oder nur eine Größe von 0 Byte hat), wird automatisch das Kalibrierprogramm *"ts_calibrate"* aufgerufen
setenv check_tscalibfile off saveenv	Automatische Überprüfung der Touchscreen-Kalibrierung deaktiviert, das Vorhandensein der Datei *"/home/etc/pointercal"* wird nicht geprüft

Hinweis: Das in Tabelle 22 jeweils mit aufgeführte Kommando *"saveenv"* ist erforderlich, um die modifizierte Konfiguration persistent im Flash des PLCcore-iMX35 abzuspeichern.

7.14.2 Manuelle Kalibrierung des Touchscreen

Das manuelle Kalibrieren des Touchscreens erfolgt interaktiv, indem der Anwender auf dem Display vorgegebene Markierungen ("Fadenkreuze") anklickt. Das dazu notwendige Kalibrierungs-Programm wird über die Kommandozeile gestartet, daher ist zunächst wie in Abschnitt 7.9.1 beschrieben die Anmeldung an der Kommando-Shell des PLCcore-iMX35 notwendig. Anschließend ist im Telnet- bzw. Terminal-Fenster folgendes Kommando einzugeben:

```
ts_calibrate
```

Im Verlaufe der Kalibrierungssequenz werden auf dem Display nacheinander 5 Markierungen ("Fadenkreuze" in allen 4 Ecken sowie in der Mitte) angezeigt, die vom Anwender anzuklicken sind. Je exakter dabei die dargestellten Markierungen angeklickt werden, umso größer ist die erreichbare Genauigkeit bei der späteren Bedienung des Touchscreen. Es wird daher empfohlen, zur Kalibrierung einen Eingabestift (umgangssprachlich als "Touchpen" oder "Stylus" bezeichnet) zu verwenden, wie er auch für Handhelds, PDAs oder Grafiktabletts üblich ist.

Nach Abschluss der Kalibrierung werden die Kalibrierdaten in der Datei *"/home/etc/pointercal"* gespeichert. Geht diese Datei verloren (z.B. durch Neuformatierung der Flash-Disk, ist die Kalibrierung erneut auszuführen.

Hinweis: Das Development Kit PLCcore-iMX35 wird bereits fertig kalibrieret ausgeliefert. Eine Neukalibrierung ist nur in Ausnahmefällen (z.B. nach Wechsel des Displays mit integriertem Touchscreen) notwendig.

7.15 Software-Update des PLCcore-iMX35

Bei Auslieferung ab Werk sind bereits sämtliche für den Betrieb des PLCcore-iMX35 notwendigen Firmwarekomponenten auf dem Modul installiert. Ein Firmwareupdate ist daher nur in Ausnahmefällen erforderlich, um beispielsweise eine aktuellere Software mit neuen Eigenschaften einzuspielen.

7.15.1 Update der SPS-Firmware

Als SPS-Firmware wird 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-iMX35 wie in Abschnitt 7.9.1 beschrieben als auch die Anmeldung am FTP-Server gemäß Abschnitt 7.9.2.

Das Update der SPS-Firmware erfolgt durch ein selbst-extrahierendes Firmware-Archiv, das per FTP auf das PLCcore-iMX35 übertragen wird. Nach dem Start des FTP-Servers auf dem PLCcore-iMX35 (Kommando *"pureftp"*, siehe Abschnitt 7.9.2) kann das entsprechende Firmware-Archiv in das Verzeichnis *"/tmp"* des PLCcore-iMX35 übertragen werden (siehe Bild 25).

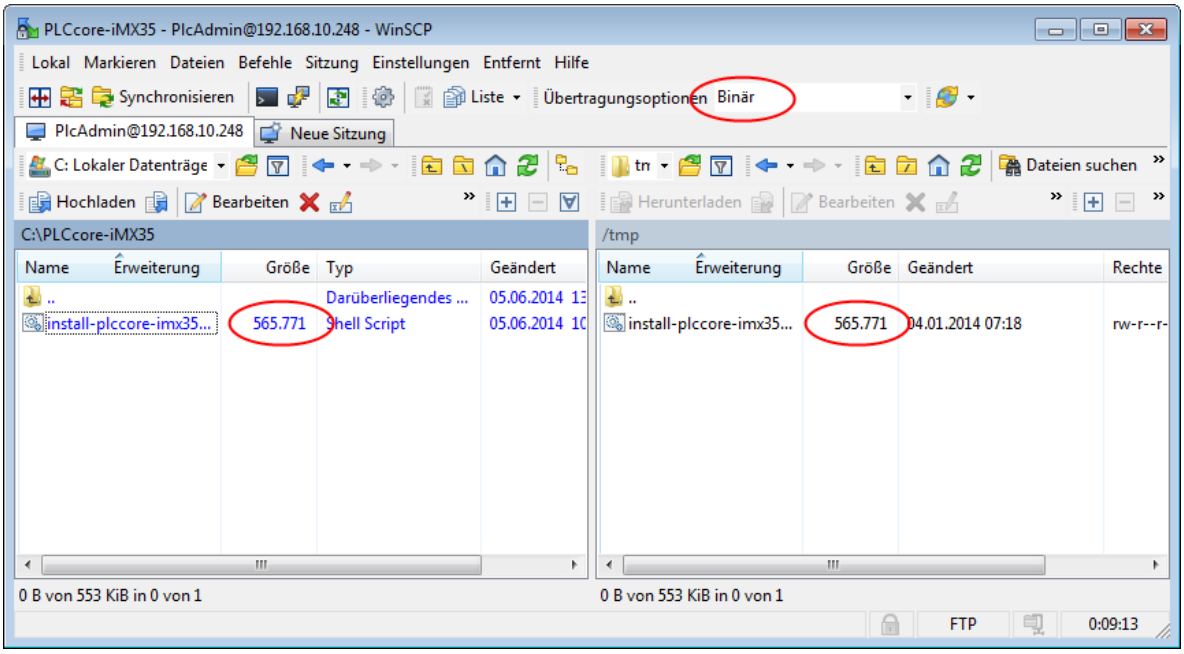


Bild 25: Dateitransfer im FTP-Client "WinSCP"

Wichtig: Für den FTP-Transfer des Firmware-Archives ist der Übertragungstyp *"Binär"* 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-iMX35 exakt dieselbe Größe besitzt wie die Originaldatei auf dem PC (siehe Bild 25). Eine abweichende Größe deutet auf einen falschen Übertragungsmodus hin (z.B. *"Text"*). In diesem Fall ist der Download mit dem Übertragungstyp *"Binär"* zu wiederholen.

Nach dem Download des selbst-extrahierenden Archives muss die SPS-Firmware auf dem PLCcore-iMX35 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-imx35-0506_0100.sh"* für die Version 5.06.01.00). Diese Nummer ist bei der Befehlseingabe entsprechend anzupassen:

```
cd  /tmp
chmod  +x  install-plccore-imx35-0506_0100.sh
./ install-plccore-imx35-0506_0100.sh
```

Tipp: Die Kommando-Shell auf dem PLCcore-iMX35 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-imx35-0506_0100.sh"* erweitert.

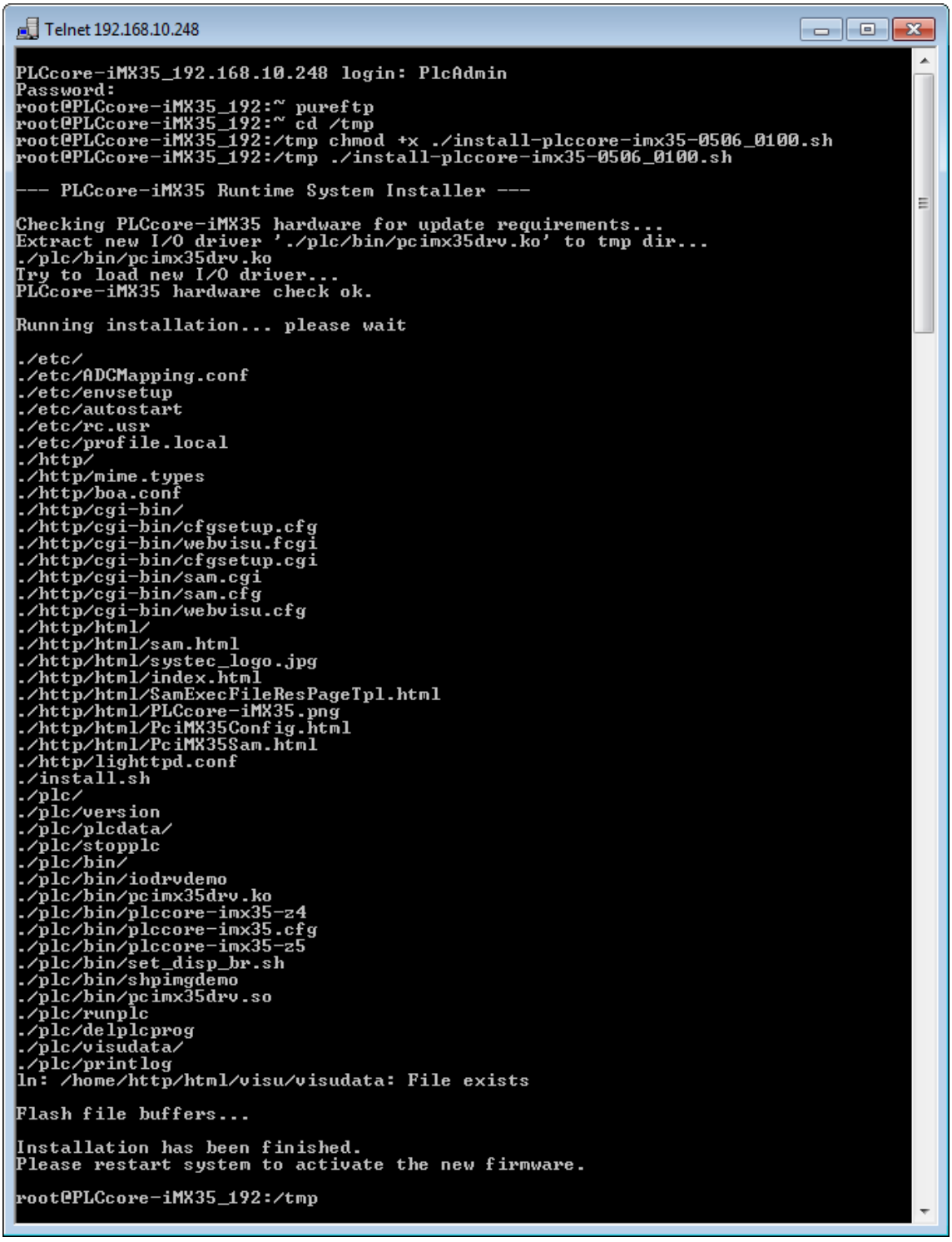


Bild 26: Installation der SPS-Firmware auf dem PLCcore-iMX35

Bild 26 verdeutlicht die Installation der SPS-Firmware auf dem PLCcore-iMX35. 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-imx35.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.**

7.15.2 Update des Linux-Images

Der Update des Linux-Images erfolgt via TFTP (**T**rivial **FTP**) innerhalb des Linux-Bootloaders *"U-Boot"*. Auf dem PC ist hierfür ein entsprechender TFTP-Server notwendig, beispielsweise die Freeware *"TFTPD32"* (siehe Abschnitt 7.1). Das Programm besteht lediglich aus einer einzelnen EXE-Datei, die keine Installation erfordert und sofort gestartet werden kann. Nach dem Programmstart sollte ein geeignetes Arbeitsverzeichnis ("Current Directory") durch Anklicken der Schaltfläche *"Browse"* eingestellt werden (z.B. *"C:\PLCcore-iMX35"*). In diesem Verzeichnis müssen sich Image-Dateien für das PLCcore-iMX35 befinden (*"linuximage"* und *"root.squashfs"*).

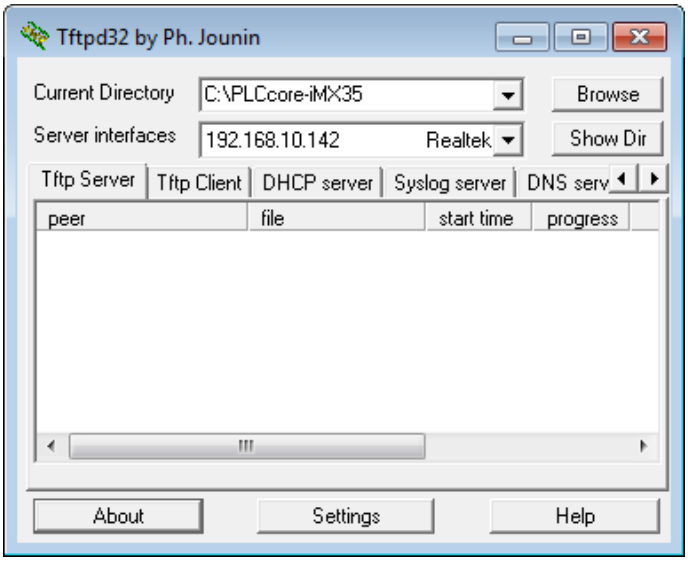


Bild 27: TFTP-Server für Windows "TFTPD32"

Voraussetzung für den TFTP-Download der Image-Dateien **ist die abgeschlossene Ethernet-Konfiguration** des PLCcore-iMX35 **gemäß Abschnitt 7.3**. Für das Update des Linux-Images ist neben der Ethernet-Verbindung noch eine serielle Verbindung zum PLCcore-iMX35 erforderlich. Hier gelten ebenfalls die im Abschnitt 7.2 beschriebenen Einstellungen für das Terminalprogramm (115200 Baud, 8 Datenbits, 1 Stopbit, keine Parität, keine Flusskontrolle).

Ein Update des Linux-Images auf dem PLCcore-iMX35 ist nur möglich, wenn Linux noch nicht gestartet ist. Daher ist vor dem Update der Linux-Autostart zu unterbinden und stattdessen zum "U-Boot" Kommandoprompt zu wechseln. Die dazu notwendigen Schritte beschreibt Abschnitt 7.2.

Nach Reset (z.B. Taster S601 am Developmentboard) meldet sich der "U-Boot" Kommandoprompt. Hier sind zum Update des Linux-Images die im Folgenden beschriebenen Kommandos in der aufgeführten Reihenfolge einzugeben:

Tabelle 23: Kommando-Sequenz zum Update des Linux-Images auf dem PLCcore-iMX35

Kommando	**Bedeutung**
setenv serverip <host_ip_addr>	Setzen der IP-Adresse des TFTP-Servers. Bei Verwendung von *"TFTPD32"* wird diese auf dem PC im Feld "Server Interface" angezeigt
tftp linuximage	Download des Linux-Images vom Entwicklungs-PC auf das PLCcore-iMX35
erase nor0,4	Löschen des vom Linux-Image benötigten Flash-Bereiches
cp.b ${fileaddr} 0xa00e0000 ${filesize}	Speichern des Linux-Images im Flash des PLCcore-iMX35
tftp root.squashfs	Download des Root-Filesystems vom Entwicklungs-PC auf das PLCcore-iMX35
erase nor0,5	Löschen des vom Root-Filesystems benötigten Flash-Bereiches
cp.b ${fileaddr} 0xa04e0000 ${filesize}	Speichern des Root-Filesystems im Flash des PLCcore-iMX35

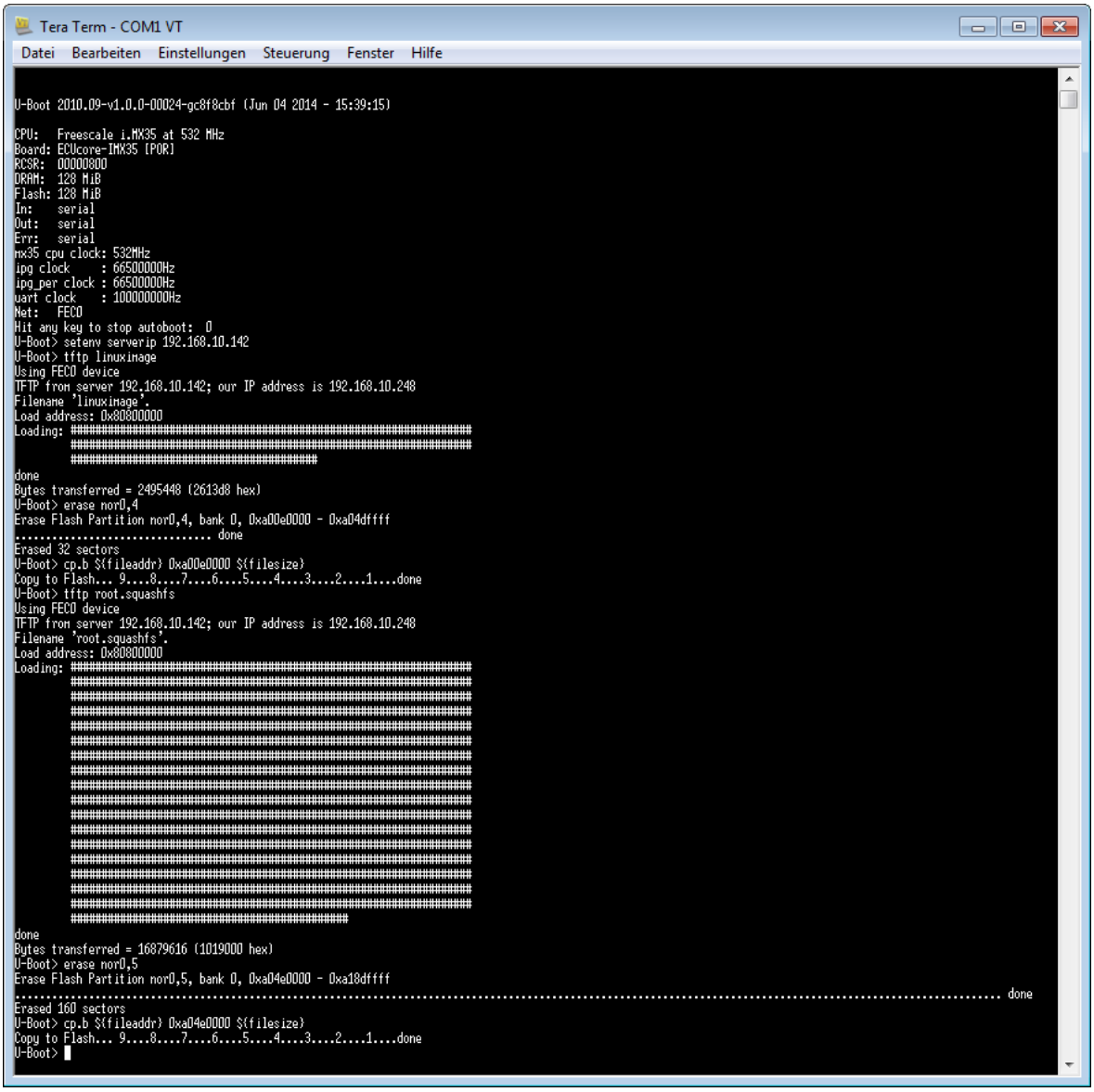


Bild 28: Download des Linux-Images auf das PLCcore-iMX35

Nach Abschluss der Konfiguration sind gemäß Abschnitt 7.2 die Voraussetzungen für einen Linux-Autostart wieder herzustellen.

Nach Reset (z.B. Taster S601 am Developmentboard) startet das PLCcore-iMX35 mit dem aktuellen Linux-Image.

Hinweis: Nach Abschluss der Konfiguration ist die serielle Verbindung zwischen PC und PLCcore-iMX35 nicht mehr erforderlich.

8 Adaption von Ein-/Ausgängen sowie Prozessabbild

8.1 Datenaustausch über Shared Prozessabbild

8.1.1 Übersicht zum Shared Prozessabbild

Das PLCcore-iMX35 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-1121 ("VMware-Image des Linux-Entwicklungssystems für das ECUcore-iMX35")**.

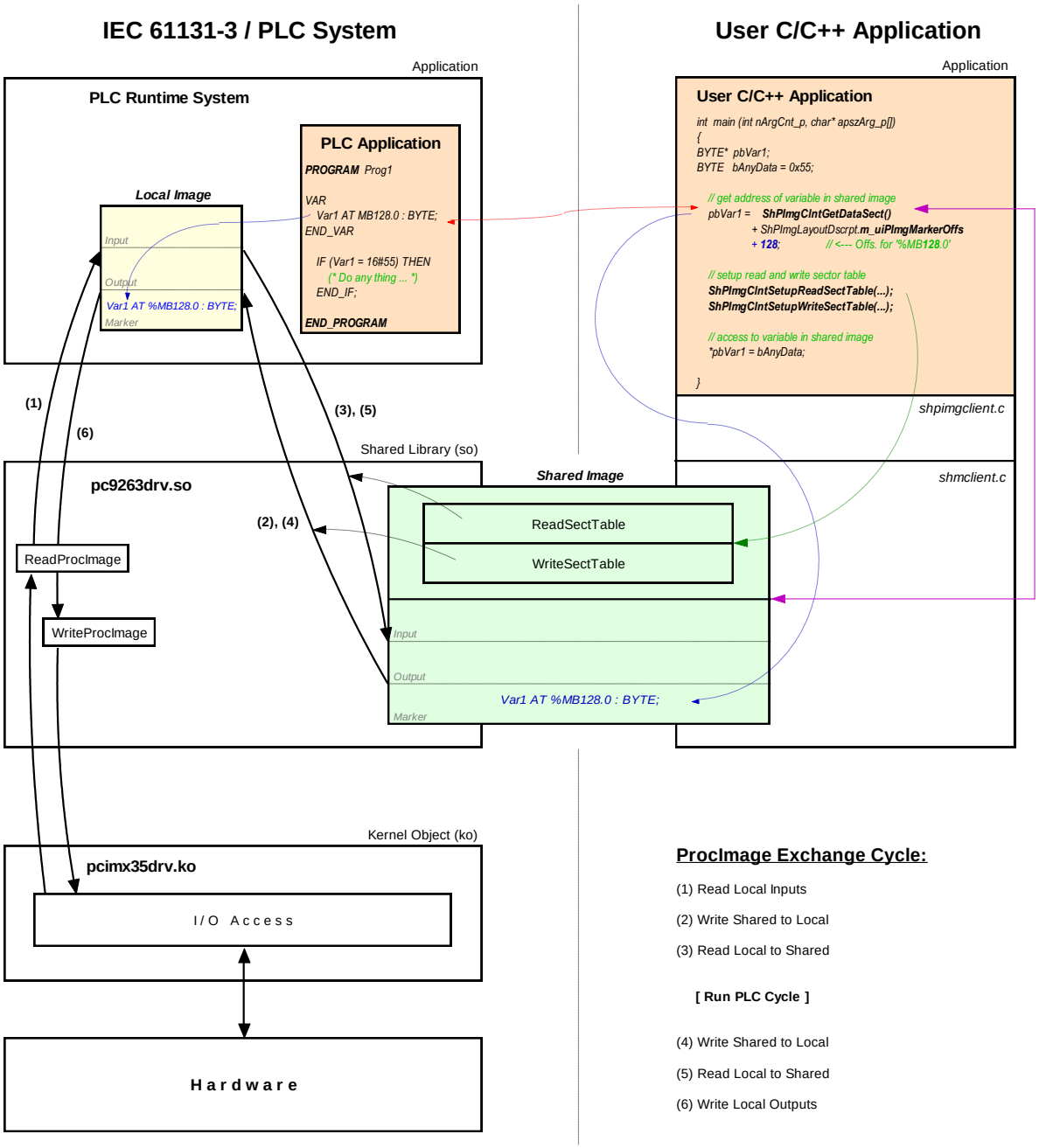


Bild 29: Übersicht Shared Prozessabbild

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 29 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 29). 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 29). 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.

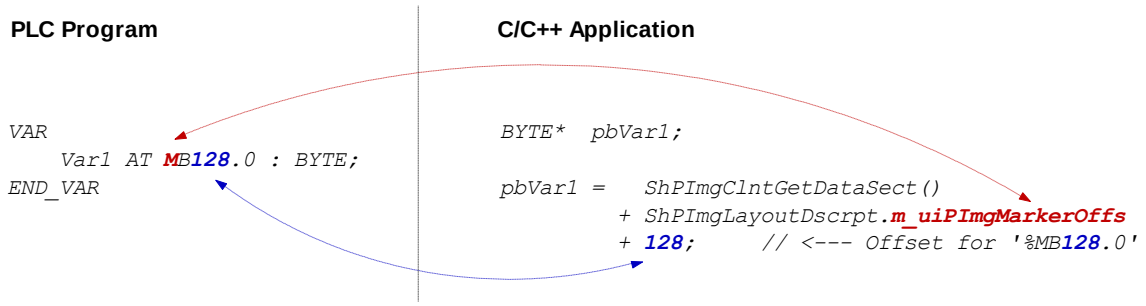
Um den Datenaustausch mit externen Applikationen zu ermöglichen, muss die **Option *"Share PLC process image"*** in der SPS-Konfiguration aktiviert sein (siehe Abschnitt 7.4.1). Alternativ kann auch der Eintrag *"EnableSharing="* in der Sektion *"[ProcImg]"* innerhalb der Konfigurationsdatei ***"/home/plc/bin/plccore-imx35.cfg"*** direkt gesetzt werden (siehe Abschnitt 7.4.3). 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 29), 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 29 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 Eingangsdaten 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 der 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 ein 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, dass dann später in der Hauptschleife der Client-Applikation ausgewertet und verarbeitet wird.

8.1.2 API des Shared Prozessabbild Client

Wie im Bild 29 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 *"pcimx35drv.so"* neu generiert wird (setzt SO-1119 - "Driver Development Kit für das ECUcore-iMX35" 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 *"pcimx35drv.so"* neu generiert wird (setzt SO-1119 - "Driver Development Kit für das ECUcore-iMX35" 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-1121 ("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-iMX35. 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 ARM11-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-iMX35"* (Manual-Nr.: L-1570).

Wie im Bild 29 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.h*). 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 anwenderspezifischer 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-iMX35 lauffähigen Linux-Applikation notwendig sind. Tabelle 24 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 24: 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-iMX35/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-iMX35/user"* aufgerufen, so entpackt es die im Archiv enthaltenen Files in den Pfad *"/projects/ECUcore-iMX35/user/**shpimgdemo**"*. Bild 30 veranschaulicht das Entpacken von *"shpimgdemo.tar.gz"* innerhalb des Linux-Entwicklungssystems.

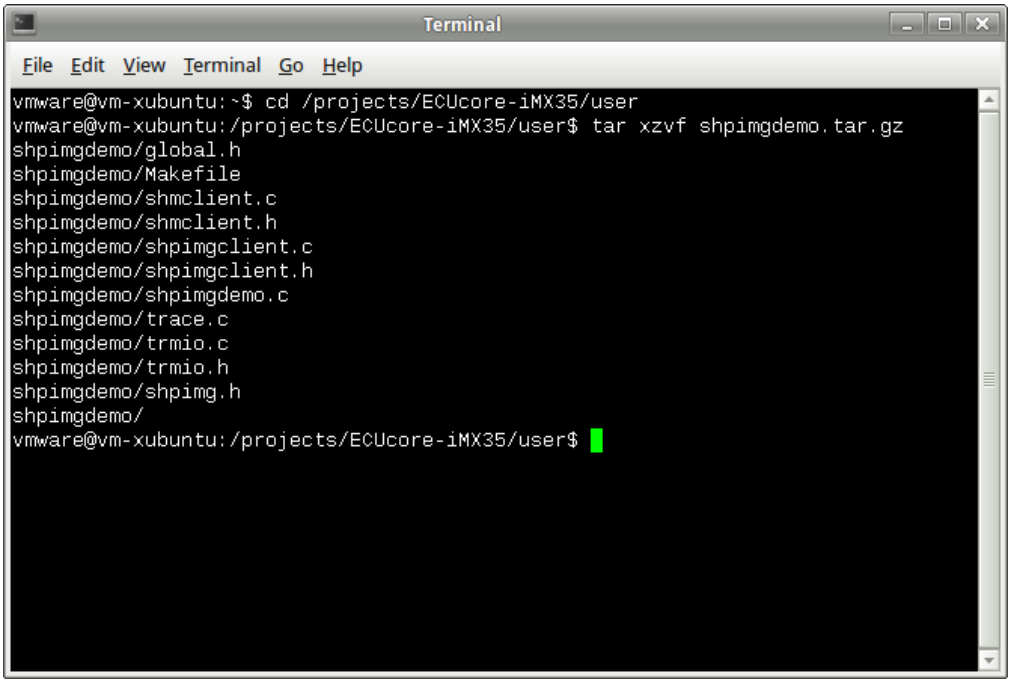


Bild 30: Entpacken des Archiv-Files shpimgdemo.tar.gz im Linux-Entwicklungssystem

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

```
cd  shpimgdemo
make
```

Bild 31 zeigt die Generierung des Demoprojektes *"shpimgdemo"* im Linux-Entwicklungssystem.

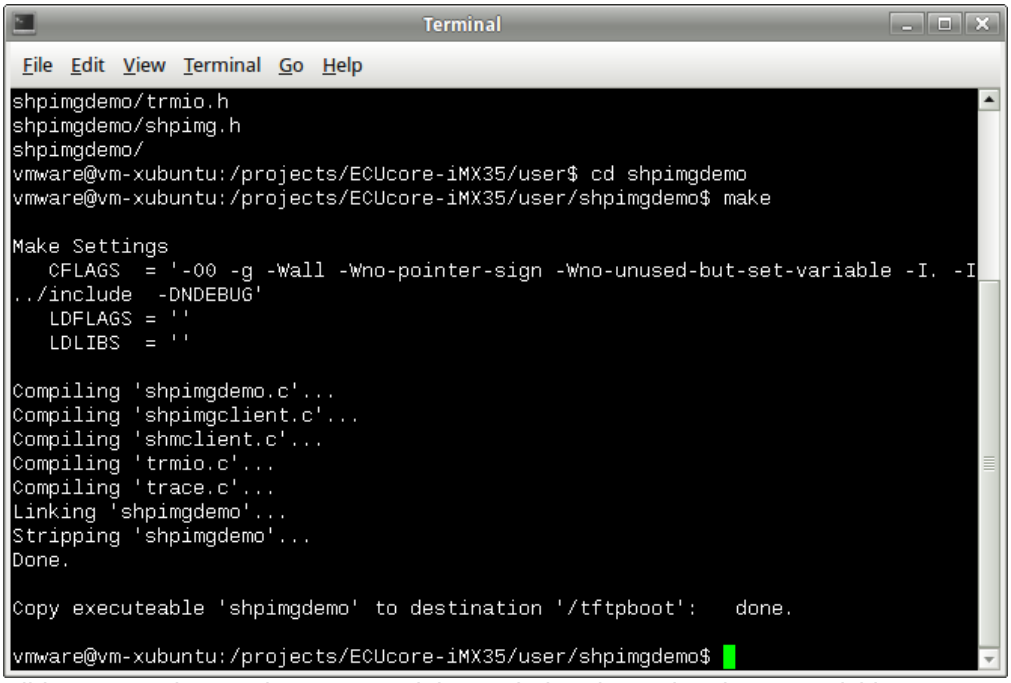


Bild 31: Generierung des Demoprojektes "shpimgdemo" im Linux-Entwicklungssystem

Die Anwendung und Bedienung des Demoprojektes *"shpimgdemo"* auf dem PLCcore-iMX35 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-iMX35

Um das *Shared Process Image Demo* auch ohne vorherige Einarbeitung in die Linux-basierte C/C++ Programmierung für das PLCcore-iMX35 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-iMX35 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 7.4.3).

2. Öffnen des SPS-Programmbeispiels *"RunLight"* im IEC 61131-Programmiersystem *OpenPCS* und Übersetzen des Projektes für eine Zielhardware vom Typ *"SYSTEC - PLCcore-iMX35"*

3. Auswahl der Netzwerkverbindung zum PLCcore-iMX35 und Download des Programms

4. Starten des SPS-Programms auf dem PLCcore-iMX35

5. Anmeldung an der Kommando-Shell des PLCcore-iMX35 wie in Abschnitt 7.9.1 beschrieben

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

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

   Auf dem PLCcore-iMX35 werden die digitalen Ausgänge als Lauflicht angesteuert. Mit Hilfe der Taster S604 (DI0) und S605 (DI1) kann die Laufrichtung umgeschaltet werden. Nach dem Start des Demoprogramms *"shpimgdemo"* auf dem PLCcore-iMX35 werden im Terminal zyklisch die aktuellen Statusinformationen zum Lauflicht angezeigt (siehe Bild 32).

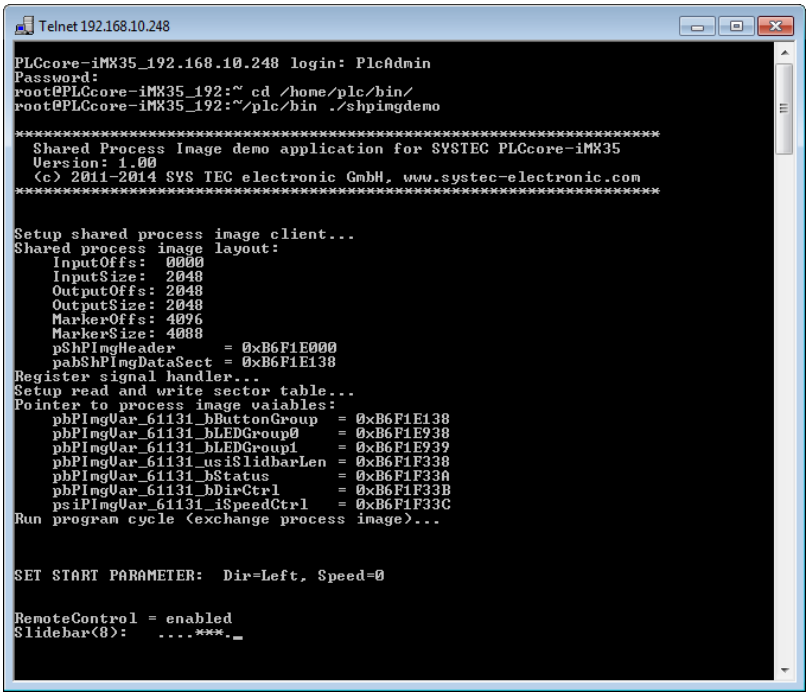


Bild 32: Terminalausgaben des Demoprogramms "shpimgdemo" nach dem Start

7. Nach Drücken des Tasters S607 (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.

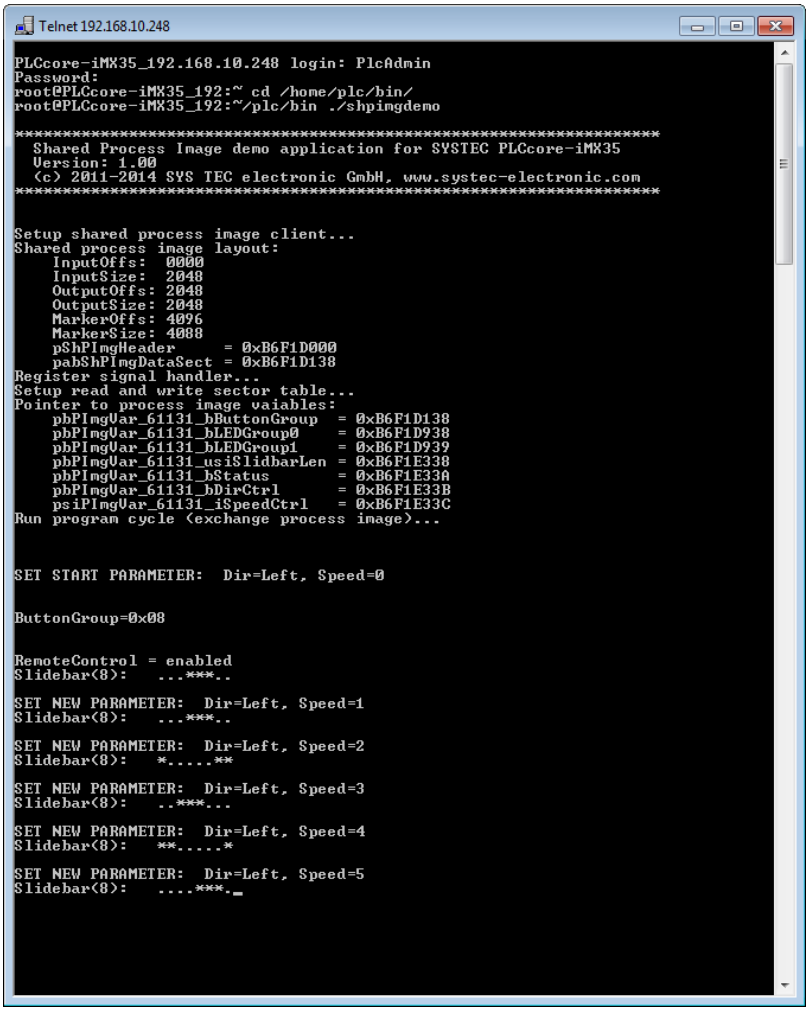


Bild 33: Terminalausgaben des Demoprogramms "shpimgdemo" nach Benutzereingaben

Bild 33 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 Driver Development Kit (DDK) für das PLCcore-iMX35

Das Driver Development Kit (DDK) für das ECUcore-iMX35 (bzw. PLCcore-iMX35) wird als zusätzliches Softwarepaket mit der Artikelnummer SO-1119 vertrieben. Es ist nicht im Lieferumfang des PLCcore-iMX35 bzw. dem Development Kit PLCcore-iMX35 enthalten. Details zum DDK beschreibt das "Software Manual Driver Development Kit für ECUcore-iMX35" (Manual-Nr.: L-1572).

Das Driver Development Kit für das ECUcore-iMX35 (bzw. PLCcore-iMX35) ermöglicht dem Anwender die eigenständige Anpassung der I/O-Ebene an ein selbst entwickeltes Baseboard. Das auf dem PLCcore-iMX35 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 ARM1136JF-S
- 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 34 vermittelt einen Überblick über die DDK-Struktur und die enthaltenen Komponenten. Das DDK beinhaltet unter anderem die Sourcen des Linux Kernel-Treibers (*pcimx35drv.ko*) und der Linux User-Bibliothek (*pcimx35drv.so*).

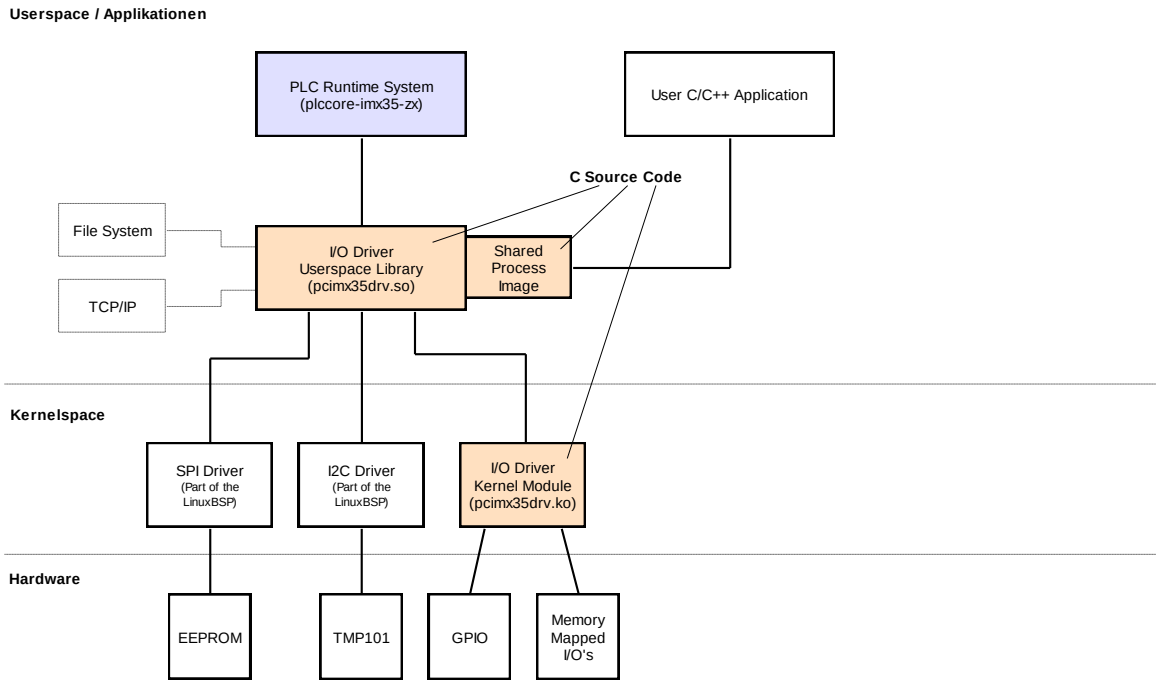


Bild 34: Übersicht zum Driver Development Kit für das PLCcore-iMX35

Lieferumfang / Komponenten des DDK:

Das DDK beinhaltet folgende Komponenten:

1. Sourcecode für Linux Kernel-Treiber (*pcimx35drv.ko*, siehe Bild 34), beinhaltet alle notwendigen Files um Kernel-Treiber neu zu erstellen (C- und H-Files, Makefile usw.)

2. Sourcecode für Linux User-Bibliothek (*pcimx35drv.so*, siehe Bild 34), 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-1121** ("VMware-Image des Linux-Entwicklungssystems") voraus, das sowohl die Quellen des verwendeten LinuxBSP als auch die erforderliche GNU-Crosscompiler Toolchain für ARM11-Prozessoren enthält.

8.3 Testen der Hardwareanschaltung

Das PLCcore-iMX35 ist primär als Zulieferteil für den Einbau in industriellen Steuerungen bestimmt. Dazu wird das PLCcore-iMX35 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  pcimx35drv.ko
./iodrvdemo
```

Bild 35 veranschaulicht das Testen der Hardwareanschaltung mit *"iodrvdemo"*.

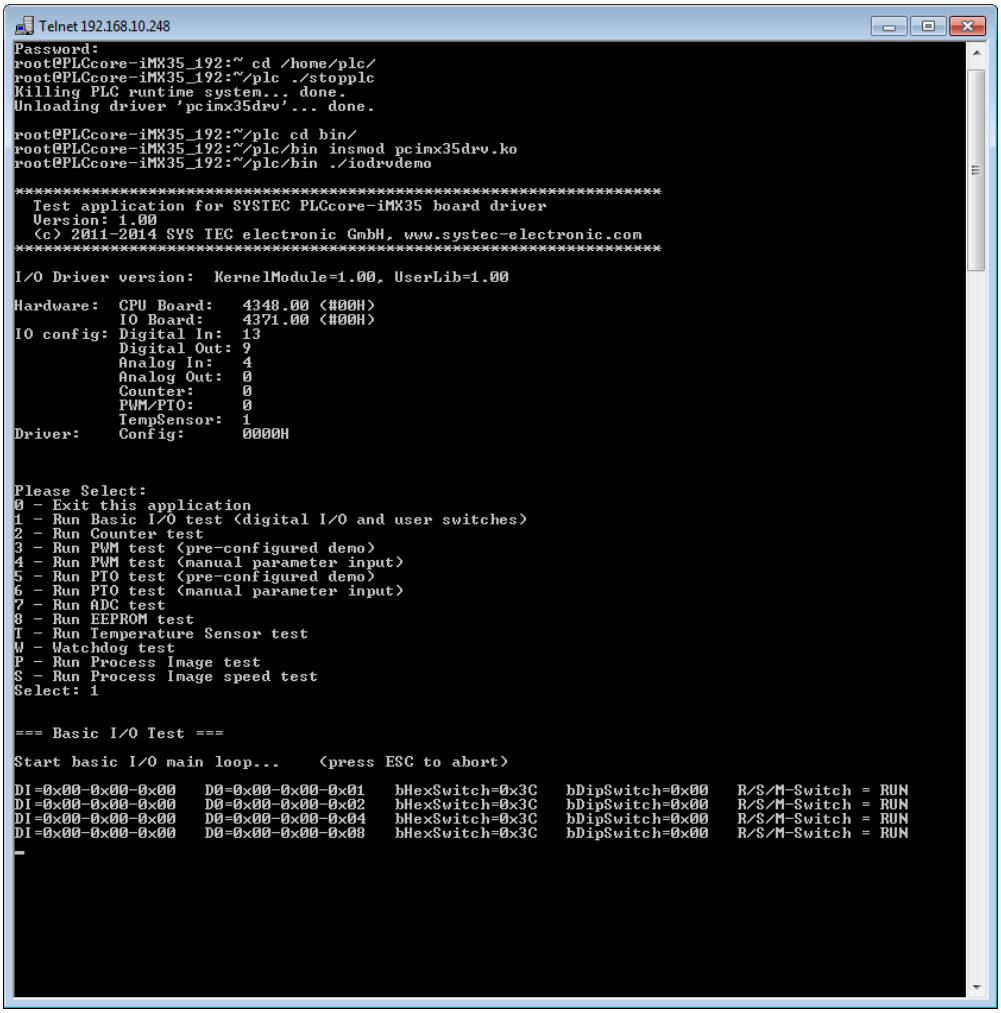


Bild 35: Testen der Hardwareanschaltung mit "iodrvdemo"

Anhang A: Firmware-Funktionsumfang des PLCcore-iMX35

Tabelle 25 listet die auf dem PLCcore-iMX35 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 25: Firmware-Funktionen und -Funktionsbausteine des PLCcore-iMX35

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	

Name	Type	Reference	Remark
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}

Name	Type	Reference	Remark
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)
HMI_SET_DISPLAY_BRIGHTNESS	FB	L-1321	HMI Version only see (4)
Functions and Function Blocks for File System			
FILE_OPEN	FB	L-1828	
FILE_CLOSE	FB	L-1828	
FILE_READ	FB	L-1828	
FILE_READ_LINE	FB	L-1828	
FILE_WRITE	FB	L-1828	
FILE_SEEK	FB	L-1828	
FILE_SYNC	FB	L-1828	
FILE_STAT	FB	L-1828	
FILE_CHMOD	FB	L-1828	
FILE_TOUCH	FB	L-1828	
FILE_DELETE	FB	L-1828	
FILE_RENAME	FB	L-1828	
FILE_COPY	FB	L-1828	
FILE_SPLIT_PATH	FB	L-1828	

Name	Type	Reference	Remark
FILE_DIR_OPEN	FB	L-1828	
FILE_DIR_CLOSE	FB	L-1828	
FILE_DIR_READ	FB	L-1828	
FILE_GET_DIR	FB	L-1828	
FILE_SET_DIR	FB	L-1828	
FILE_MKDIR	FB	L-1828	
FILE_RMDIR	FB	L-1828	
FILE_MKFIFO	FB	L-1828	
FILE_EXEC_SYS_CMD	FB	L-1828	
FTYPE_TO_INT	FUN	L-1828	
FSEEK_TO_UINT	FUN	L-1828	
FPERM_TO_STRING	FUN	L-1828	
SYSERR_TO_STRING	FUN	L-1828	
Functions and Function Blocks for Modbus			
MODBUS_OPEN_INSTANCE	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_CLOSE_INSTANCE	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_REGISTER_VAR_LIST	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_WRITE_SINGLE_REG	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_WRITE_MULTI_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_WRITE_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_INPUT_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_DISCR_INPUTS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_COILS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_WRITE_SINGLE_COIL	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_RAW_PDU_REQUEST	FB	L-1829	Since Firmware Version 5.08.02.00

(1) Das PLCcore-iMX35 unterstützt folgende Geräte zur Ablage nicht-flüchtiger 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-iMX35. Der EEPROM hat eine fixe Größe von 32 kByte.

(2) Die Schnittstelle COM0 (PORT:=0) dient primär als Serviceschnittstelle zur Administration des PLCcore-iMX35. Daher sollten über diese Schnittstelle nur Zeichen ausgegeben werden. Empfangene Zeichen versucht das Modul stets als Linux-Kommandos zu interpretieren und auszuführen (siehe Abschnitt 6.5.1).

(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-imx35.cfg"*** direkt auf 0 gesetzt werden (siehe Abschnitt 7.4.3).

(4) Die Funktionsbausteine vom Typ *HMI_Xxx* sind nur in der HMI-Version des PLCcore-iMX35 (Art.-Nr. 3390075) verfügbar.

Anhang B: Referenzdesigns zum PLCcore-iMX35

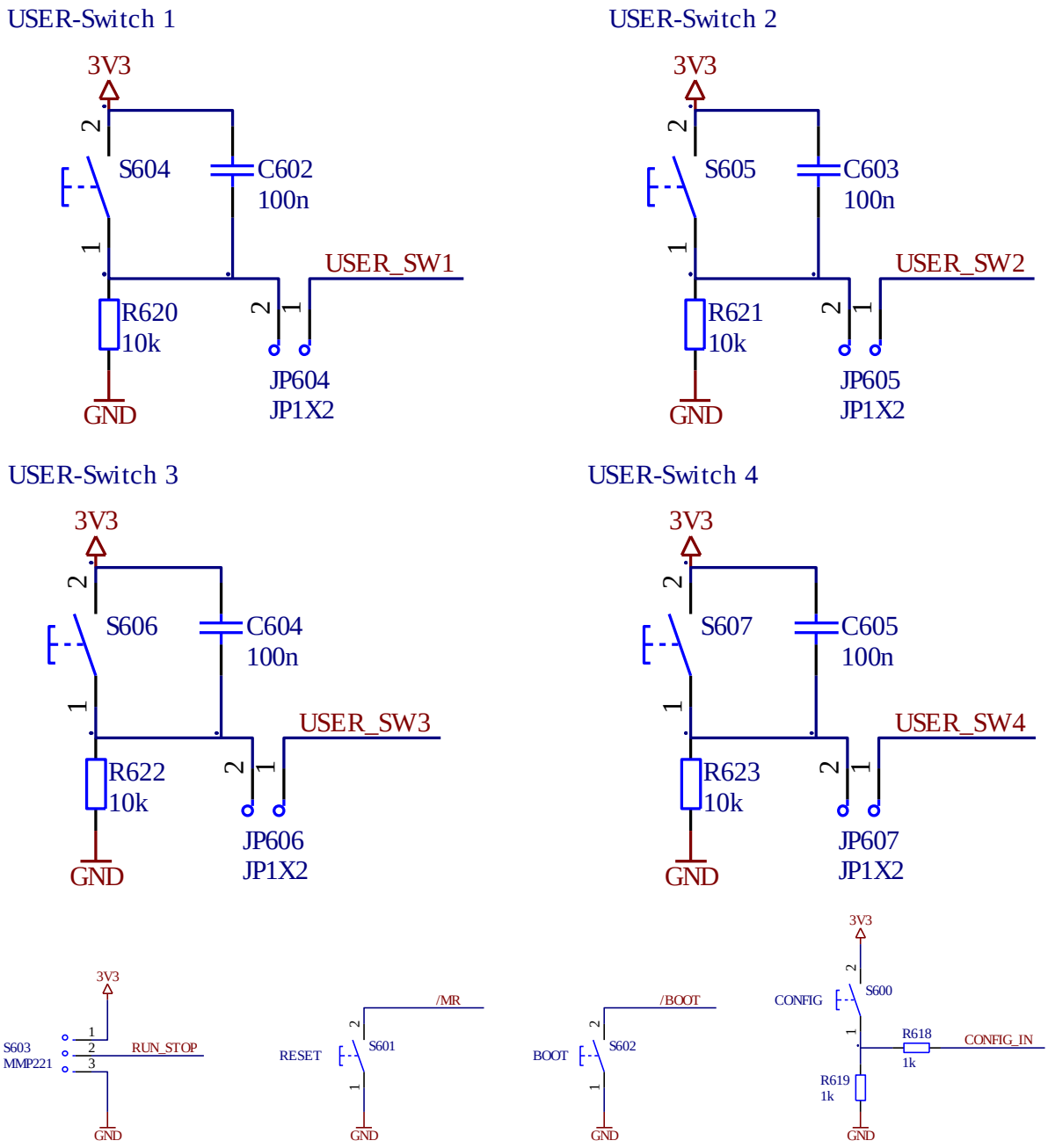


Bild 36: Referenzdesign Bedienelemente

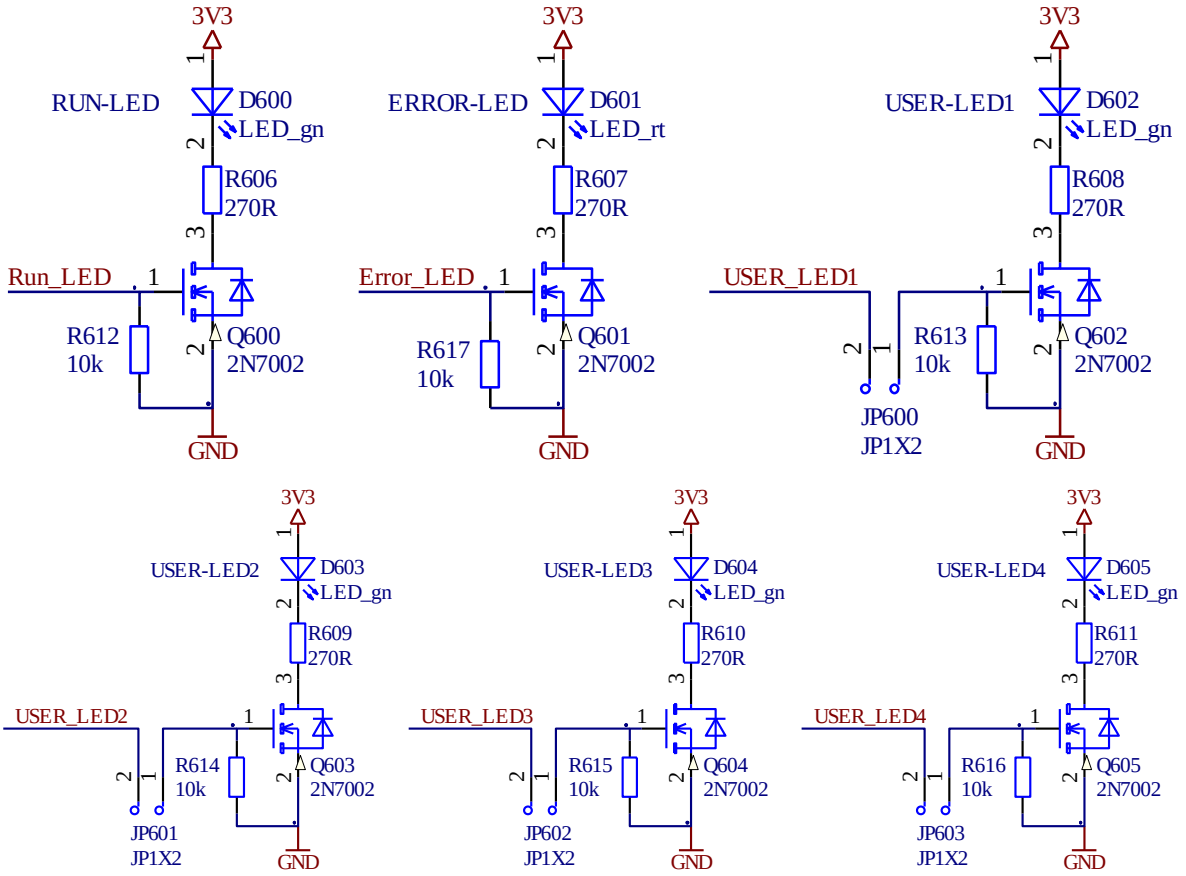


Bild 37: Referenzdesign LEDs

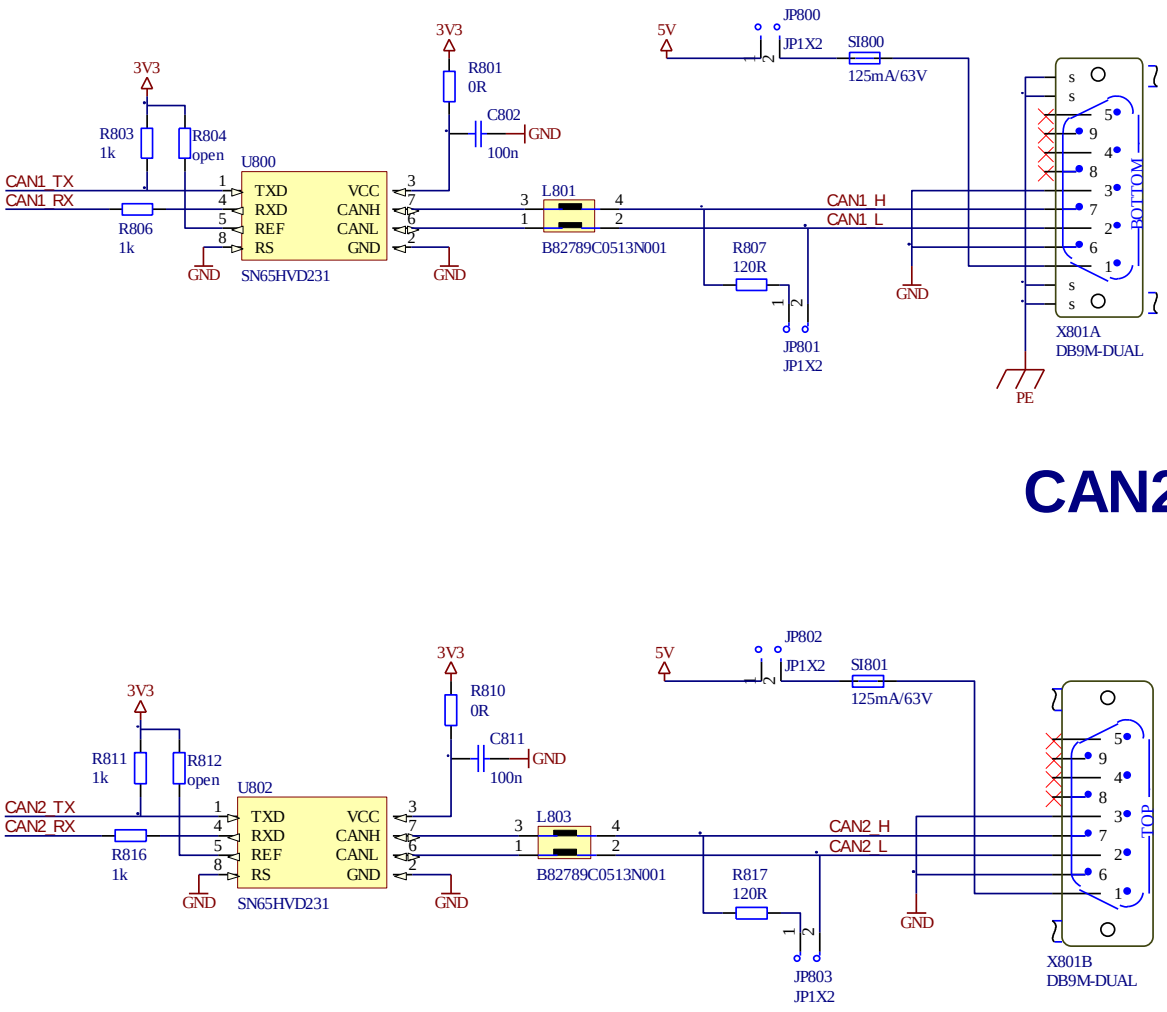
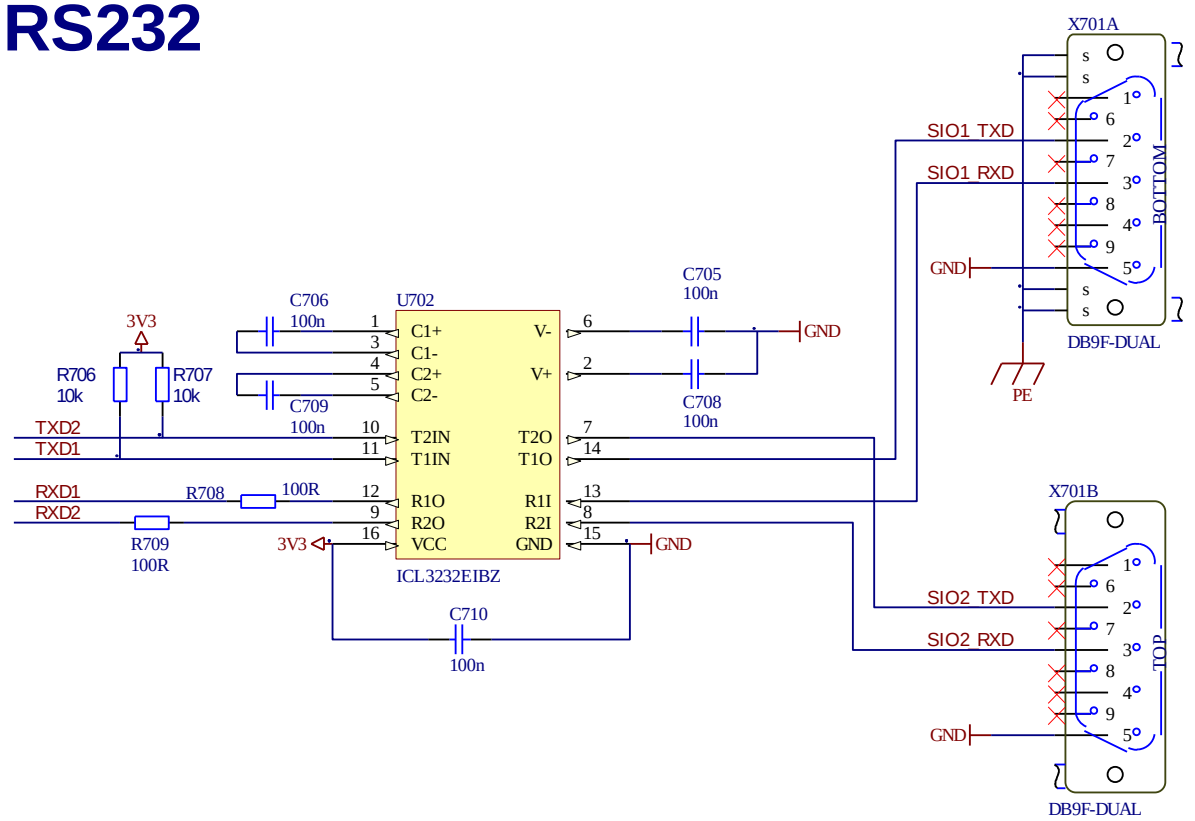


Bild 38: Referenzdesign zur Interface-Beschaltung CAN

RS232



RS485

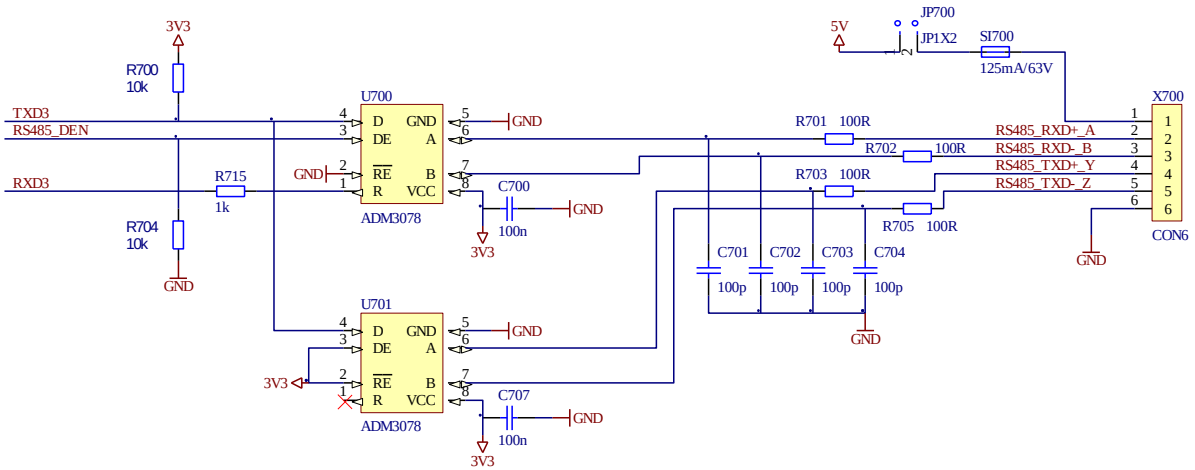


Bild 39:Referenzdesign zur Interface-Beschaltung RS232/485

LAN

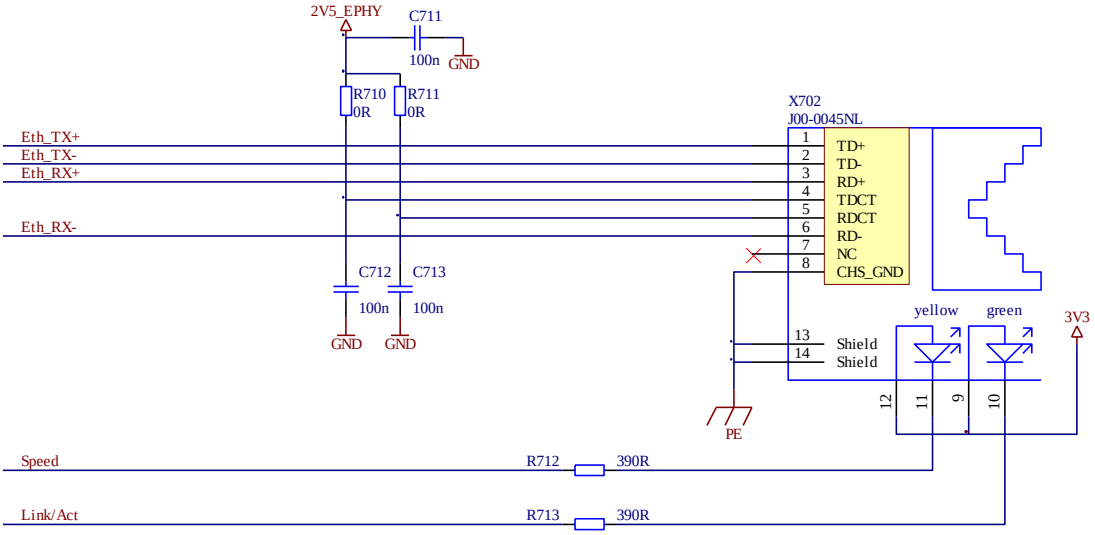
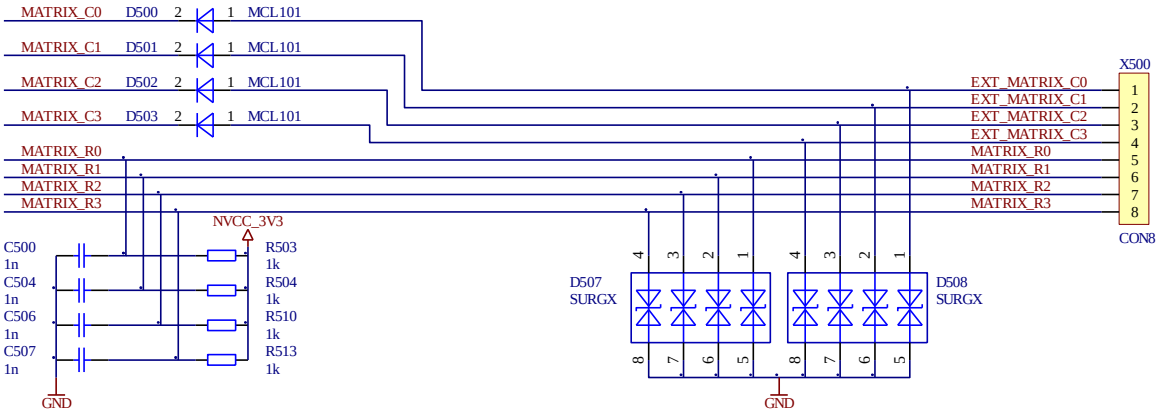


Bild 40: Referenzdesign zur Interface-Beschaltung Ethernet

Keypad



Scroll wheel

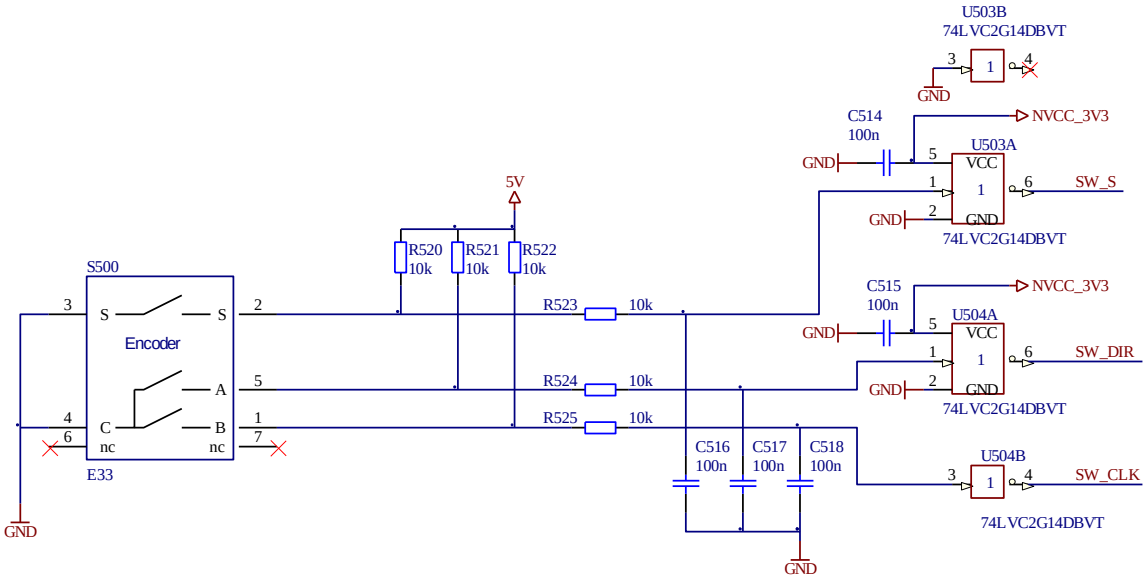


Bild 41:Referenzdesign Matrixtastatur und Scroll Wheel

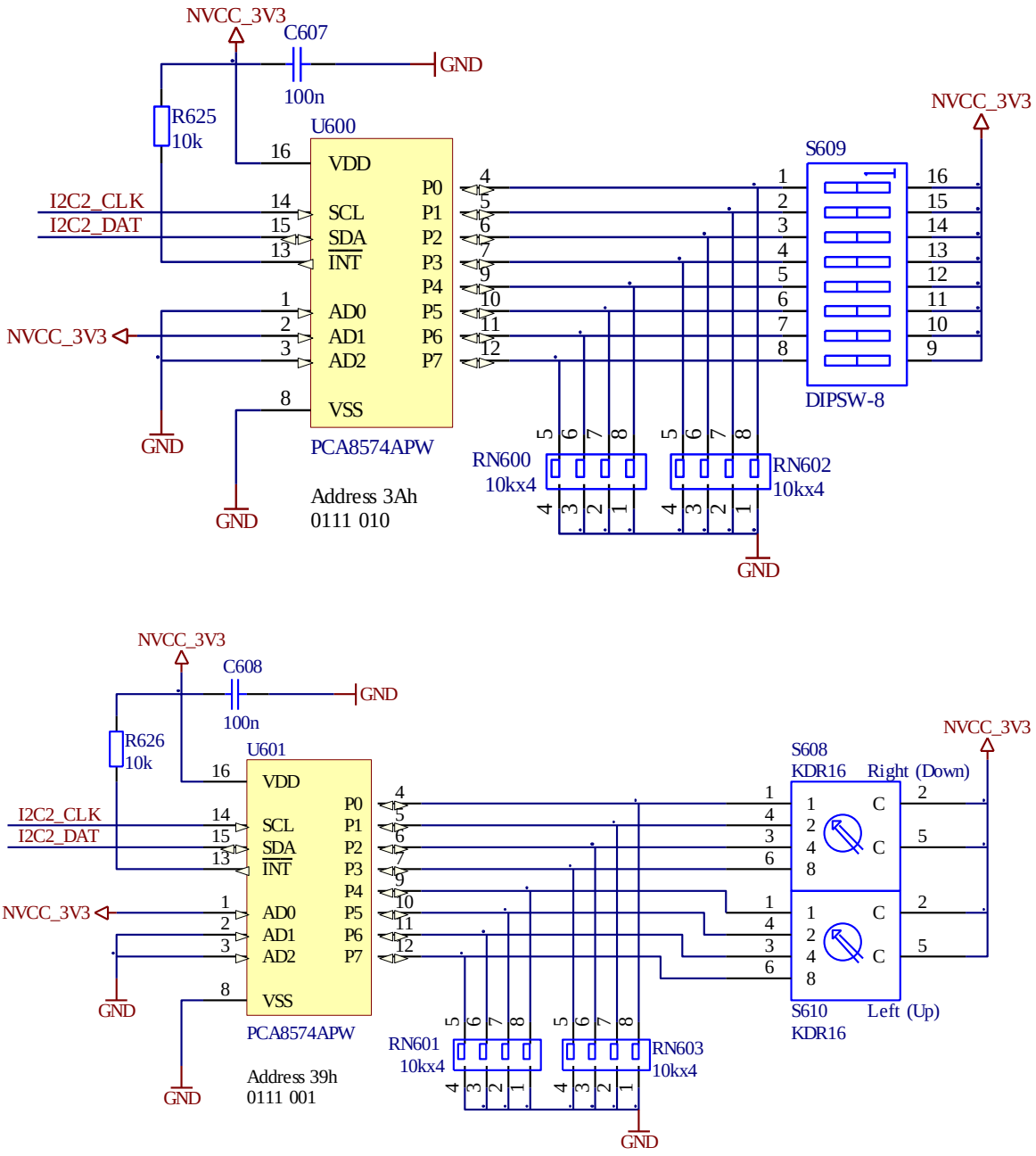


Bild 42:Referenzdesign HEX- und DIP-Switches

Anhang C: Beschriftungskarten für Matrix Folientastatur

Bild 43 enthält Beschriftungskarten im Maßstab 1:1 für die Standardbelegung der im Development Kit PLCcore-iMX35 verwendeten Folientastatur. Die Tastaturbelegung kann mit Hilfe von Funktionsbausteinen durch das SPS-Programm beliebig umdefiniert werden. Durch Austausch der auf der Rückseite eingesteckten Beschriftungskarten lässt sich Beschriftung flexibel an die reale Tastaturbelegung anpassen.

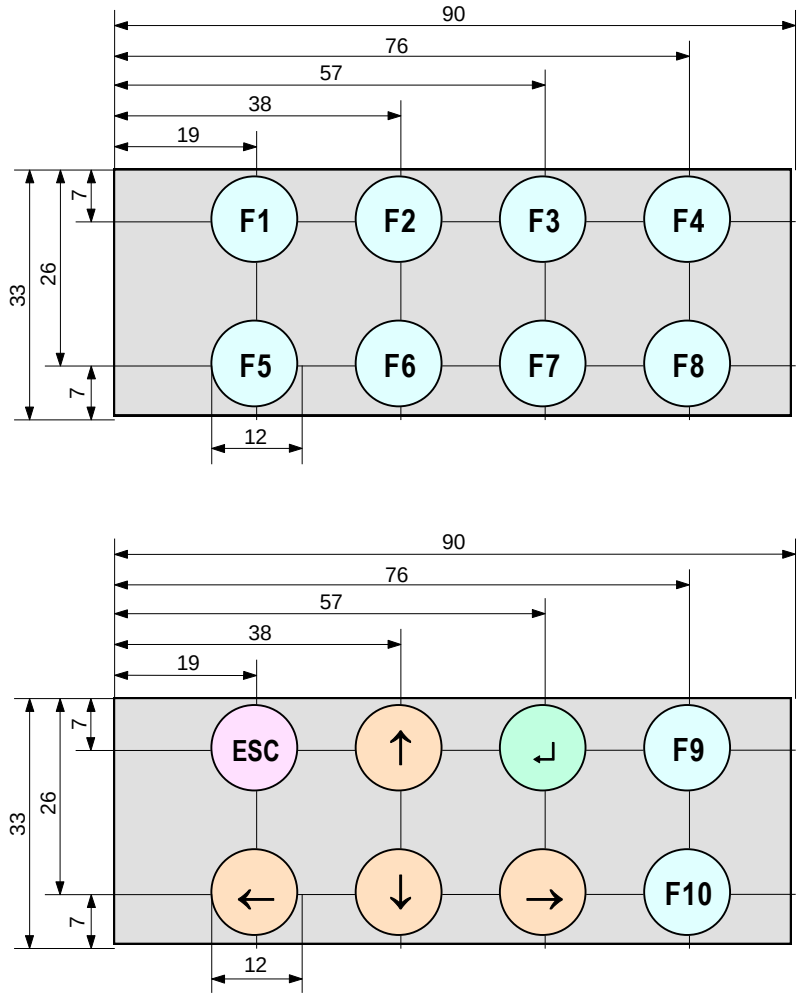


Bild 43: Beschriftungskarten für Matrix Folientastatur

Anhang D: GNU GENERAL PUBLIC LICENSE

Das auf dem PLCcore-iMX35 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-iMX35
Dokumentnummer:	L1567d_2, März 2016

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) 37 65 / 38600-4100
Email: info@systec-electronic.com