System Manual
ECUcore-9G20

User Manual
Version 1.0

Ausgabe März 2010

Dokument-Nr.: L-1253d_01

SYS TEC electronic GmbH August-Bebel-Straße 29 D-07973 Greiz
Telefon: +49 (3661) 6279-0 Telefax: +49 (3661) 6279-99
Web: http://www.systec-electronic.com Mail: info@systec-electronic.com

Status/Änderungen

Status: Freigegeben

Datum/Version	Abschnitt	Änderung	Bearbeiter
2010/03/26 1.0	alle	Erstellung	R. Sieber

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 August-Bebel-Str. 29 D-07973 Greiz GERMANY	Sie finden eine Liste unserer Distributoren unter http://www.systec-electronic.com/distributors
Angebots-Hotline:	+49 (0) 36 61 / 62 79-0 info@systec-electronic.com	
Technische Hotline:	+49 (0) 36 61 / 62 79-0 support@systec-electronic.com	
Fax:	+49 (0) 36 61 / 6 79 99	
Webseite:	http://www.systec-electronic.com	

1. Auflage März 2010

Inhalt

1 Einleitung

Vielen Dank, dass Sie sich für das SYS TEC ECUcore-9G20 entschieden haben. Mit diesem Produkt verfügen Sie über einen innovativen und leistungsfähigen Einplatinenrechner mit Linux-Betriebssystem. Aufgrund seiner hohen Performance auf besonders kleiner Baugröße sowie wegen seiner geringen Leistungsaufnahme eignet er sich besonders gut als Kommunikations- und Steuerrechner für Embedded Anwendungen.

Bitte nehmen Sie sich etwas Zeit, dieses Manual aufmerksam zu lesen. Es beinhaltet wichtige Informationen zur Inbetriebnahme, Konfiguration und Programmierung des ECUcore-9G20. Es wird Ihnen helfen, sich mit dem Funktionsumfang und der Anwendung des ECUcore-9G20 vertraut zu machen. Dieses Dokument wird ergänzt durch weitere Manuals, beispielsweise zur Hardware des Moduls. Eine Auflistung der relevanten Manuals zum ECUcore-9G20 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 ECUcore-9G20

Das ECUcore-9G20 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 ECUcore-9G20 ist ESD empfindlich und darf nur an ESD geschützten Arbeitsplätzen von geschultem Fachpersonal ausgepackt und gehandhabt bzw. betrieben werden.

Das ECUcore-9G20 ist ein Modul für den Bereich Automatisierungstechnik. Durch die Programmierbarkeit unter Linux 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.

2 Übersicht / Wo finde ich was?

Das vorliegende Dokument beschreibt die Inbetriebnahme des ECUcore-9G20 auf Basis des Development Kit ECUcore-9G20 sowie die prinzipielle Vorgehensweise bei der Softwareentwicklung für dieses Modul. Für die Hardwarekomponenten wie das ECUcore-9G20 selbst, die Developmentboards sowie Referenzschaltungen existieren eigene Hardware-Manuals. Softwareseitig wird das ECUcore-9G20 mit einem vorinstallierten Embedded Linux ausgeliefert. Anwendungen, die auf diesem Modul ausgeführt werden sollen, sind also dementsprechend als Linux-Programme zu entwickeln. Das Kit beinhaltet ein komplett eingerichtetes Linux-Entwicklungssystem in Form eines VMware-Images und ermöglicht so einen leichten Einstieg in die Softwareentwicklung für das ECUcore-9G20. Das VMware-Image kann dabei unverändert unter verschiedenen Host-Systemen benutzt werden. Tabelle 1 listet die für das ECUcore-9G20 relevanten Manuals auf.

Tabelle 1: Übersicht relevanter Manuals zum ECUcore-9G20

Informationen über…	**In welchem Manual?**
Grundlegende Informationen zum ECUcore-9G20 (Konfiguration, Administration, Anschlussbelegung, Softwareentwicklung, Referenzdesigns usw.)	In diesem Manual
Einsatz des ECUcore/PLCcore-9G20 als SPS (Programmierung des PLCcore-9G20 als SPS gemäß IEC 61131-3, Prozessabbild, Datenaustausch über Shared Prozessabbild mit externen Applikationen usw.)	System Manual PLCcore-9G20 (Manual-Nr.: L-1254)
Hardware-Beschreibung zum ECUcore-9G20, Referenzdesigns usw.	Hardware Manual ECUcore-9G20 (Manual-Nr.: L-1255)
Developmentboard zum ECUcore-9G20, Referenzdesigns usw.	Hardware Manual Developmentboard 9G20 (Manual-Nr.: L-1256)
Driver Development Kit (DDK) für das ECUcore-9G20	Software Manual Driver Development Kit (DDK) für ECUcore-9G20 (Manual-Nr.: L-1257)
Register- und Funktionsbeschreibung zum FPGA des ECUcore-9G20 / PLCcore-9G20	FPGA Manual PLCcore-9G20 (Manual-Nr.: L-1244)
CAN-Treiber	CAN Treiber Software Manual (Manual-Nr.: L-1023)

Geeignete Nachschlagewerke zur Anwendungs-Programmierung unter Linux	• Advanced Programming in the UNIX Environment, Stevens Rago, Addison-Wesley • Linux/Unix Systemprogrammierung, Helmut Herold, Addison-Wesley • Linux-UNIX-Programmierung, Jürgen Wolf, Galileo Computing • GNU Function List: http://www.silicontao.com/ProgrammingGuide/GNU_function_list

Abschnitt 4 dieses Manuals beschreibt zunächst die **elektrische Inbetriebnahme** des ECUcore-9G20 auf Basis des Development Kit ECUcore-9G20.

Abschnitt 5 erläutert **Details zur Anwendung des ECUcore-9G20**, so z.B. die Konfiguration und Administration des Moduls, die Anmeldung am System, Ethernetkonfiguration, Startprozess und Dateisystem.

Abschnitt 6 beschreibt das **VMware-Image mit dem Linux-Entwicklungssystem**.

Abschnitt 7 behandelt die Thematik der **Softwareentwicklung** für das ECUcore-9G20 und erläutert dazu die **Einbindung des I/O-Treibers** in eigene Applikationen, die Vorgehensweise zum **Übersetzen** von Anwenderprogrammen, deren **Übertragung** auf das Modul sowie das **Debugging**.

Abschnitt 10 vermittelt **Tipps & Tricks**, die den Umgang mit Linux vereinfachen helfen. Dieser Abschnitt ist insbesondere für Quereinsteiger in Linux hilfreich.

3 Produktbeschreibung

Das ECUcore-9G20 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 einen vollständigen, unter Linux programmierbaren Einplatinenrechner mit zahlreichen Erweiterungsschnittstellen zur Verfügung. Dank seiner hohen Performance und seiner zahlreichen on-board Schnittstellen ist das ECUcore-9G20 optimal zur Realisierung dezentraler Steuerungsaufgaben geeignet.

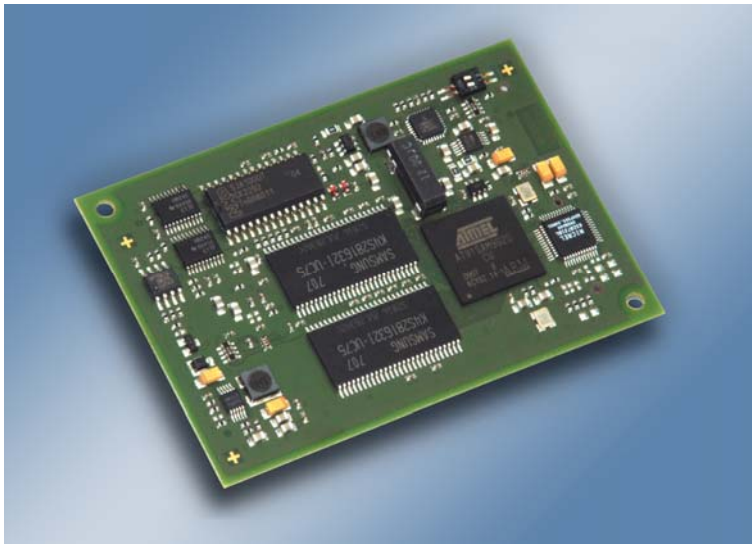


Bild 1: Ansicht des ECUcore-9G20

Einige herausragende Merkmale des ECUcore-9G20 sind:

- leistungsfähiger CPU-Kern (Atmel 32-Bit AT91SAM9G20, 400 MHz CPU Takt, 440 MIPS)
- 32 MByte SDRAM Memory, 16 MByte FLASH Memory
  (max: 64 MByte SDRAM Memory, 64 MByte FLASH Memory)
- on-board FPGA (ECP2-6)
- 1x 10/100 Mbps Ethernet LAN Interface (mit on–board PHY)
- 1x CAN 2.0B Interface
- 2x USB 2.0 Host
- 1x USB 2.0 Device
- 2x SD-Card
- 5x Asynchronous Serial Ports (UART)
- 1x SSC
- 19 digitale Eingänge, 8 digitale Ausgänge (Standardkonfiguration, modifizierbar über DDK)
- 3 analoge Eingänge (ADC)
- 4 High-speed Counter (Pulse/Dir oder A/B)
- 4 PWM-/PTO-Ausgang (Pulse/Dir)
- SPI und I^2C extern nutzbar
- On-board Peripherie: RTC, Temperatursensor, Watchdog, Power-fail Eingang
- Betriebssystem: Linux
- geringe Abmaße (78 * 54 mm),

Die Verfügbarkeit einer vollständigen Einplatinenrechners als aufsteckbares "Core“ und die damit verbundenen geringen Abmessungen reduzieren den Aufwand und die Kosten bei der Entwicklung anwenderspezifischer Industriesteuerungen enorm. Gleichzeitig eignet sich das ECUcore-9G20

besonders als intelligenter Netzwerkknoten zur dezentralen Verarbeitung von Prozesssignalen (CANopen und UDP/TCP), für Applikationen im Bereich Motion Control sowie als Basiskomponente von Spezialbaugruppen in industriellen Bereichen.

In der Standard-I/O-Konfiguration verfügt das Modul über 19 digitale Eingänge (DI0...DI18, 3.3V-Pegel), 8 digitale Ausgänge (DO0...DO7, 3.3V-Pegel), 4 High-Speed Counter-Eingänge sowie 4 PWM/PTO-Ausgänge. Mit Hilfe des Driver Development Kit (SO-1105) kann diese Standard-I/O-Konfiguration an die spezifischen Applikationsbedingungen adaptiert werden. Die Ablage des Anwenderprogramms in der on-board Flash-Disk des Moduls ermöglicht den autarken Wiederanlauf nach einer Spannungsunterbrechung.

Das ECUcore-9G20 basiert auf einem Embedded Linux als Betriebssystem. Dadurch ist es möglich, mehrere anwenderspezifische Programme simultan abzuarbeiten.

Das auf dem ECUcore-9G20 eingesetzte Embedded Linux ist unter der GNU General Public License, Version 2 lizenziert. Der entsprechende Lizenztext ist im Anhang A aufgeführt. Die vollständigen Quellen des verwendeten LinuxBSP sind im VMware-Image des Linux-Entwicklungssystems (SO-1105) 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

4 Development Kit ECUcore-9G20

4.1 Übersicht

Das Development Kit ECUcore-9G20 ermöglicht durch das im Kit enthaltene Developmentboard eine schnelle Inbetriebnahme des ECUcore-9G20 und vereinfacht den Aufbau von Prototypen anwenderspezifischer Applikationen, die auf diesem Modul basieren. Das Developmentboard besitzt u.a. verschiedene Möglichkeiten der Spannungszufuhr, Ethernet-Schnittstelle, Anbindungen für CAN-Bus, USB und SD-Card sowie 4 Taster und 4 LEDs als Bedienelemente für die digitalen Ein- und Ausgänge sowie ein Potentiometer für den analogen Eingang. Die an den Steckverbindern des ECUcore-9G20 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 ECUcore-9G20.

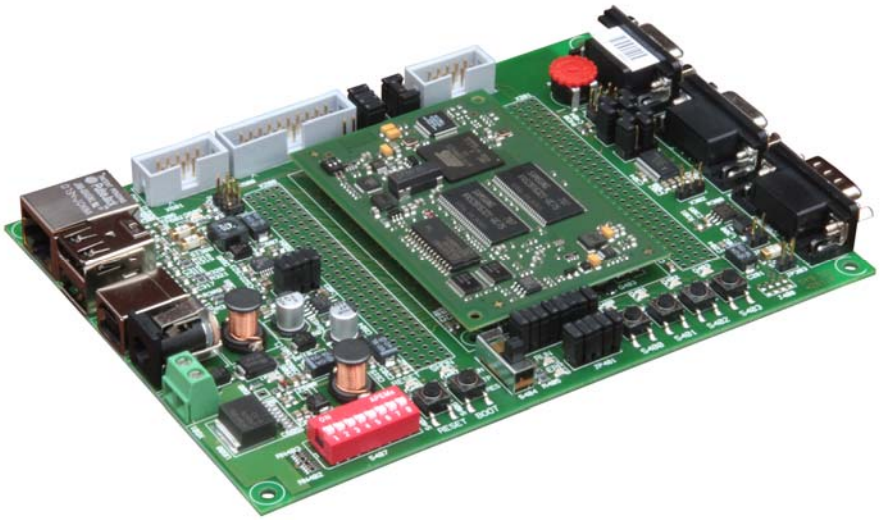


Bild 2: Development Kit ECUcore-9G20

Das Development Kit ECUcore-9G20 gewährleistet eine rasche und problemlose Inbetriebnahme des ECUcore-9G20. Es vereint dazu alle notwendigen Hard- und Software-Komponenten, die zur Erstellung eigener Applikationen erforderlich sind: als Kernkomponente das ECUcore-9G20 selbst, das zugehörige Developmentboard mit I/O-Peripherie und zahlreichen Schnittstellen, das Linux-Entwicklungssystem sowie weiteres Zubehör. Damit bildet das Developmentkit die ideale Plattform zur Entwicklung anwenderspezifischer Applikationen auf Basis des ECUcore-9G20.

Das Development Kit ECUcore-9G20 beinhaltet folgende Komponenten:

- ECUcore-9G20
- Developmentboard für ECUcore-9G20
- 24V Steckernetzteil
- Ethernet-Kabel
- RS232-Kabel
- DVD mit Linux-Entwicklungssystem, Beispielen, Dokumentation und weiteren Tools (SO-1105)

Als Software-Entwicklungsplattform sowie Debugumgebung für das ECUcore-9G20 dient das im Kit enthaltene Linux-Entwicklungssystem. In Form eines VMware-Images kann das Entwicklungssystem unverändert unter verschiedenen Host-Systemen verwendet werden. Abschnitt 6 beschreibt exemplarisch den Umgang mit dem VMware-Image unter Windows.

4.2 Elektrische Inbetriebnahme des Development Kit ECUcore-9G20

Das für den Betrieb des Development Kit ECUcore-9G20 notwendige Steckernetzteil sowie die erforderlichen Ethernet- und RS232-Kabel sind bereits im Lieferumfang des Kits enthalten. Für die Inbetriebnahme des Kit sind mindestens die Anschlüsse für Versorgungsspannung (X600/X601), COM0 (X301) und ETH0 (X304) erforderlich. Einen vollständigen Überblick über die Anschlüsse des Development Kit ECUcore-9G20 vermittelt Tabelle 2.

Tabelle 2: Anschlüsse des Development Kit ECUcore-9G20

Anschluss	Bezeichnung auf Developmentboard	Bemerkung
Versorgungsspannung	X600 oder X601	Das im Lieferumfang enthaltene Steckernetzteil ist zum direkten Anschluss an X600 vorgesehen
ETH0 (Ethernet)	X304	Schnittstelle zur Kommunikation mit Linux-Entwicklungssystem (Programmier-PC) sowie zur freien Verwendung durch das Anwenderprogramm
COM0 (RS232)	X301	Schnittstelle wird zur Konfiguration der Baugruppe (z.B. Setzen der IP-Adresse) und für Diagnose bzw. Debugging benötigt, ist im normalen Betrieb für das Anwenderprogramm frei verfügbar
COM1 (RS232)	X300	Schnittstelle ist für das Anwenderprogramm frei verfügbar
COM2 (RS232)	X302	Schnittstelle ist für das Anwenderprogramm frei verfügbar
CAN0 (CAN)	X303	Schnittstelle ist für das Anwenderprogramm frei verfügbar

Bild 3 zeigt die Lage der wichtigsten Anschlüsse auf dem Developmentboard für das ECUcore-9G20. Anstelle des mitgelieferten Steckernetzteiles kann die Stromversorgung des Kit optional auch über X601 mit einer externen Quelle von 24V/1A erfolgen.

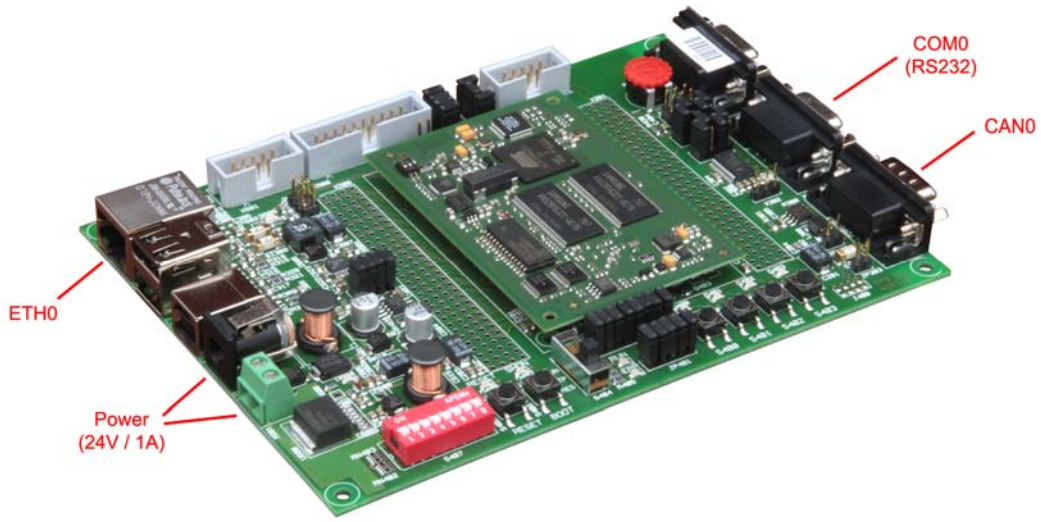


Bild 3: Lage der wichtigsten Anschlüsse auf dem Developmentboard für das ECUcore-9G20

Hinweis: Bei der Inbetriebnahme sind zunächst die Kabel für Ethernet (ETH0, X304) und RS232 (COM0, X301) anzuschließen, erst danach ist die Versorgungsspannung (X600 / X601) zu aktivieren.

4.3 Bedienelemente des Development Kit ECUcore-9G20

Das Development Kit ECUcore-9G20 ermöglicht eine einfache Inbetriebnahme des ECUcore-9G20. Es verfügt über verschiedene Bedienelemente sowohl zur Konfiguration des Moduls als auch zur Simulation von Ein- und Ausgängen für den Einsatz des ECUcore-9G20 als Kernkomponente einer Industriesteuerung. Tabelle 3 listet die einzelnen Bedienelemente des Developmentboard auf und beschreibt deren Bedeutung.

Tabelle 3: Bedienelemente des Developmentboard für das ECUcore-9G20

Bedienelement	Bezeichnung	Bedeutung
Taster 0	S400	Digitaler Eingang DI0
Taster 1	S401	Digitaler Eingang DI1
Taster 2	S402	Digitaler Eingang DI2
Taster 3	S403	Digitaler Eingang DI3
LED 0	D400	Digitaler Ausgang DO0
LED 1	D401	Digitaler Ausgang DO1
LED 2	D402	Digitaler Ausgang DO2
LED 3	D403	Digitaler Ausgang DO3
Poti (ADC)	R429	Analoger Eingang AI0
Run/Stop-Schalter	S404	Bedienelement ist für das Anwenderprogramm frei verfügbar (z.B. Run / Stop für Abarbeitung des Steuerprogramms)
Run-LED	D405	Bedienelement ist für das Anwenderprogramm frei verfügbar (z.B. Anzeige Aktivitätszustand des Steuerprogramms)
Error-LED	D406	Bedienelement ist für das Anwenderprogramm frei verfügbar (z.B. Anzeige Fehlerzustand des Steuerprogramms)
DIP-Schalter	S407	Bedienelement ist für das Anwenderprogramm frei verfügbar (z.B. Konfiguration Bitrate und Master-Modus CAN0)

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 und Diagnose des ECUcore-9G20 von PCs, wie z.B. Laptops, die selber keine physikalische RS232-Schnittstelle mehr besitzen (siehe Abschnitt 5).

Bild 4: SYS TEC USB-RS232 Adapter Kabel

4.4.2 Driver Development Kit

Das Driver Development Kit für das ECUcore-9G20 (Bestellnummer SO-1106) ermöglicht dem Anwender die eigenständige und flexible Anpassung der I/O-Ebene an ein selbst entwickeltes Baseboard. Einen Überblick zum Driver Development Kit vermittelt Abschnitt 8.1.

5 Anwendung und Administration des ECUcore-9G20

5.1 Systemvoraussetzungen und erforderliche Softwaretools

Zur Administration des ECUcore-9G20 ist ein beliebiger Windows- oder Linux-PC erforderlich, der über eine Ethernet-Schnittstelle sowie eine serielle Schnittstelle (RS232) verfügt. Als Alternative zur seriellen on-board Schnittstelle eignet sich auch das von SYS TEC angebotenen USB-RS232 Adapter Kabel (Bestellnummer 3234000, siehe Abschnitt 4.4.1), das eine entsprechende RS232-Schnittstelle über einen USB-Port zur Verfügung stellt.

Alle in diesem Manual aufgeführten Beispiele beziehen sich auf die Administration des ECUcore-9G20 von einem Windows-PC aus. Das Vorgehen auf einem Linux-PC ist analog.

Zur Administration des ECUcore-9G20 sind folgende Softwaretools erforderlich:

Terminalprogramm

Ein Terminalprogramm ermöglicht die Kommunikation mit der **Kommando-Shell** des ECUcore-9G20 über eine **serielle RS232-Verbindung an COM0 des ECUcore-9G20**. Diese ist Voraussetzung für die im Abschnitt 5.4 beschriebene Ethernet-Konfiguration des ECUcore-9G20. 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 ECUcore-9G20 über eine **Ethernet-Verbindung an ETH0 des ECUcore-9G20**. Voraussetzung für die Verwendung eines Telnet-Clients ist eine abgeschlossene Ethernet-Konfiguration des ECUcore-9G20 gemäß Abschnitt 5.4. Alternativ zum Telnet-Client können auch sämtliche Kommandos über ein Terminalprogramm (an COM0 des ECUcore-9G20) 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 ECUcore-9G20 (ETH0) und dem PC. Dies erlaubt beispielsweise das **Editieren von Konfigurationsdateien**, indem diese zunächst vom ECUcore-9G20 auf den PC übertragen, dort mit einem Editor bearbeitet und anschließend wieder zurück auf das ECUcore-9G20 geschrieben werden. Der Download von Dateien auf das ECUcore-9G20 ist aber auch zum **Update von Softwarekomponenten** 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 zum Update des Linux-Images auf dem ECUcore-9G20 erforderlich. Standardmäßig ist bereits ein entsprechend konfigurierter TFTP-Server im VMware-Image des Linux-Entwicklungssystems enthalten. Um alternativ ein Update des Linux-Images von einem Windows-PC durchführen zu können, 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.

5.2 Systemstart des ECUcore-9G20

5.2.1 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 der Anwenderprogramme übernimmt (siehe Abschnitt 5.2.2). Für Servicezwecke wie beispielsweise die Konfiguration der Ethernet-Schnittstelle (siehe Abschnitt 5.4) oder zum Update des Linux-Images (siehe Abschnitt 5.15) 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 4 aufgelisteten Voraussetzungen geprüft, von denen alle Bedingungen für ein autarkes Booten des Linux-Images erfüllt sein müssen.

Tabelle 4: Voraussetzungen zum Booten von Linux

Nr.	Bedingung	Bemerkung
1	DIP1 auf ECUcore-9G20 = "Off" UND Anschluss "/BOOT" = High (Taster S406 am Developmentboard **nicht** gedrückt)	Der DIP-Schalter 1 auf dem ECUcore-9G20 und der Modulanschluss "/BOOT" sind elektrisch parallel geschalten. Nur wenn beide Elemente nicht aktiv sind (DIP-Schalter 1 offen, Modulanschluss "/BOOT" nicht aktiv) liegt das Signal "/BOOT" am ECUcore-9G20 auf H-Pegel und gibt damit den Linux-Autostart frei. Die Lage des DIP-Schalter 1 auf dem ECUcore-9G20 zeigt Bild 5, die Position des Anschlusses "/BOOT" am Steckverbinder des Moduls ist im Hardware Manual ECUcore-9G20 (Manual-Nr.: L-1255) definiert.
2	**Kein** Abbruch der Autoboot-Prozedur über COM0 des ECUcore-9G20	Sind die vorangegangenen Bedingungen erfüllt, überprüft "U-Boot" nach Reset für ca. 1 Sekunde die serielle Schnittstelle COM0 des ECUcore-9G20 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 4 wird unter folgenden Voraussetzungen der Linux-Bootvorgang nach Reset (z.B. Taster S405 am Developmentboard) unterbunden und stattdessen der "U-Boot"-Kommandoprompt aktiviert:

(1) DIP1 = "On" oder /BOOT = "Low" DIP1: siehe Bild 5, /BOOT: siehe Manual L-1255
Developmentboard: "/BOOT" = Taster S406
DIP1 und "/BOOT" sind parallel geschalten

- ODER -

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

Nach dem Betätigen des Reset-Tasters (z.B. Taster S405 am Developmentboard) meldet sich der "U-Boot" Kommandoprompt.

Bild 5 zeigt die Lage und Bedeutung des DIP-Schalters 1 auf dem ECUcore-9G20. Da dieser DIP-Schalter im eingebauten Zustand des Moduls unter Umständen schwer zugänglich sein kann, ist das damit verbundene Portpin des Prozessors parallel dazu auch als Anschluss "/BOOT" am Steckverbinder des ECUcore-9G20 verfügbar (siehe Hardware Manual ECUcore-9G20, Manual-Nr.: L-1255).

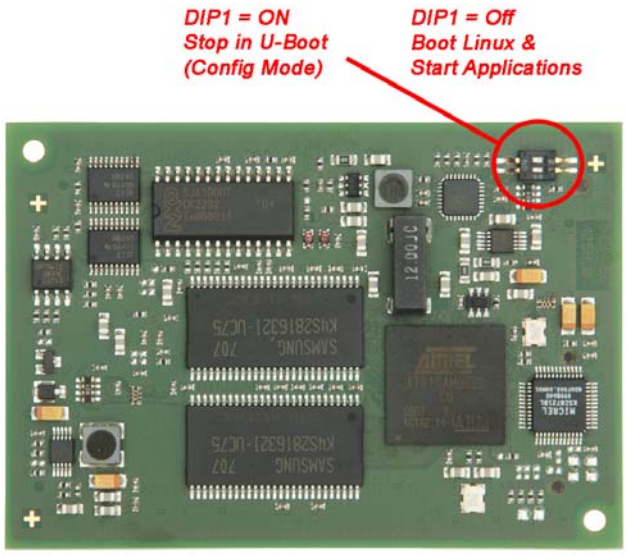


Bild 5: Lage und Bedeutung des DIP-Schalter 1 auf dem ECUcore-9G20

Die Kommunikation mit dem Bootloader "U-Boot" erfolgt ausschließlich über die serielle Schnittstelle COM0 des ECUcore-9G20. Als Gegenstelle ist auf dem PC ein beliebiges Terminalprogramm zu starten (z.B. HyperTerminal oder TeraTerm, siehe Abschnitt 5.1) und wie folgt zu konfigurieren (siehe Bild 6):

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

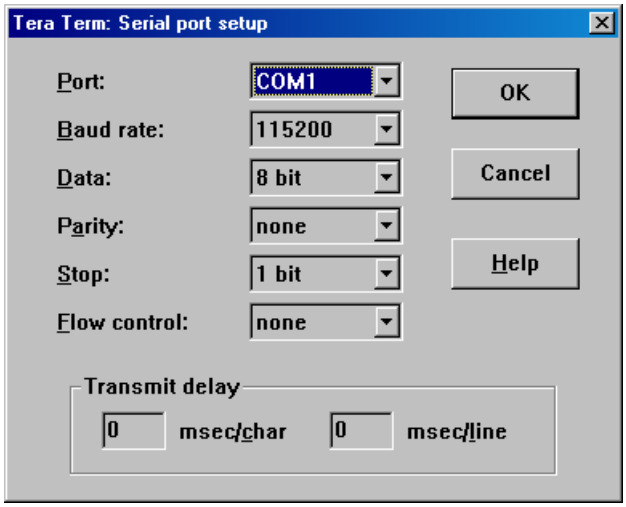
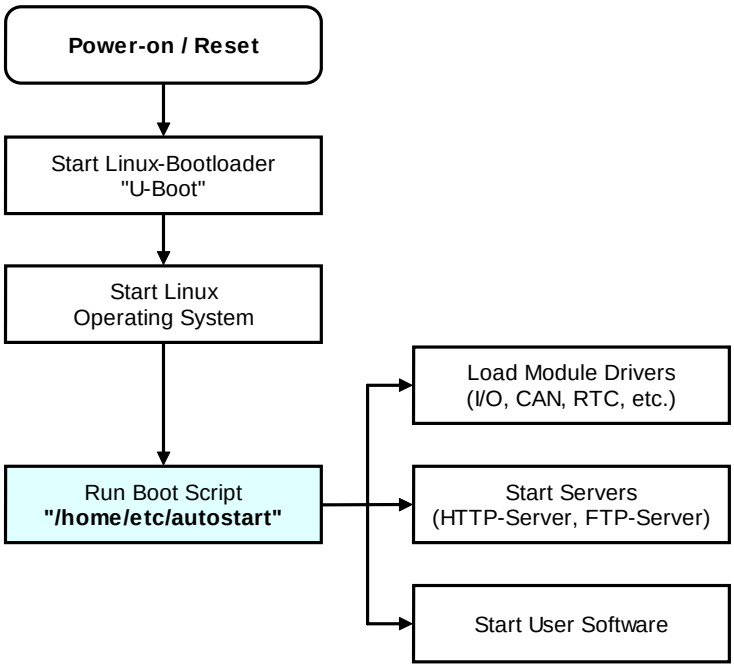


Bild 6: Terminaleinstellungen am Beispiel von "TeraTerm"

5.2.2 Autostart für Anwendersoftware

Standardmäßig startet das ECUcore-9G20 nach Power-on bzw. Reset das Linux-Betriebssystem und dieses lädt wiederum alle notwendigen Softwarekomponenten zur Ausführung der Anwendersoftware.

Damit eignet sich das ECUcore-9G20 für den Einsatz in autarken Steuerungssystemen, die nach einer Spannungsunterbrechung selbständig und ohne Benutzeraktionen die Ausführung des Steuerprogramms wieder aufnehmen. Bild 7 zeigt den Systemstart im Detail.



Details zur Deaktivierung des autarken Linux-Starts und zum Aktivierung des "U-Boot" Kommandoprompts siehe Abschnitt 5.2.1

Details zur Programmierung des ECUcore-9G20 siehe Abschnitt 7

Bild 7: Systemstart des ECUcore-9G20

Das ECUcore-9G20 kann so konfiguriert werden, dass nach einem Reset die Anwendersoftware automatisch startet. Die dazu notwendigen Kommandos sind in dem Startskript **"/home/etc/autostart"** zu hinterlegen. Ebenso können hier bei Bedarf die notwendigen Umgebungsvariablen gesetzt sowie erforderliche Treiber geladen werden.

Das Startskript **"/home/etc/autostart"** ist entsprechend der gewünschten Funktionalität anzupassen. Hier kann z.B. durch Einfügen des Kommandos *"pureftp"* der Aufruf des FTP-Servers beim Booten des ECUcore-9G20 automatisiert werden. Das Skript lässt sich im FTP-Client *"WinSCP"* (siehe Abschnitt 5.1) mit Hilfe der Taste *"F4"* bzw. der Schaltfläche *"F4 Edit"* direkt auf dem ECUcore-9G20 bearbeiten.

5.3 Kommandoprompt des Bootloaders "U-Boot"

Der Bootloader "U-Boot" wird als erste Softwarekomponente unmittelbar nach einem Hardware-Reset des ECUcore-9G20 gestartet. Im Standardbetriebsmodus lädt der Bootloader autark das Linux-Betriebssystem, das dann wiederum das Ausführen aller weiteren Anwenderapplikationen veranlasst. Innerhalb des Bootloader "U-Boot" werden zentrale Konfigurationseinstellungen für das ECUcore-9G20 festgelegt. Primär wird der Kommandoprompt des Bootloaders "U-Boot" für folgende Aufgaben benötigt:

- Ethernet-Konfiguration des ECUcore-9G20 (siehe Abschnitt 5.4) sowie
- Update des Linux-Images (siehe Abschnitt 5.15)

Die Vorgehensweise zum Aktivieren des "U-Boot" Kommandoprompts beschreibt Abschnitt 5.2.1. Durch Eingabe von "?" wird eine Hilfe zu allen verfügbaren Kommandos aufgelistet, das Kommando *"printenv"* zeigt die derzeitige Modul-Konfiguration an (siehe Bild 8).

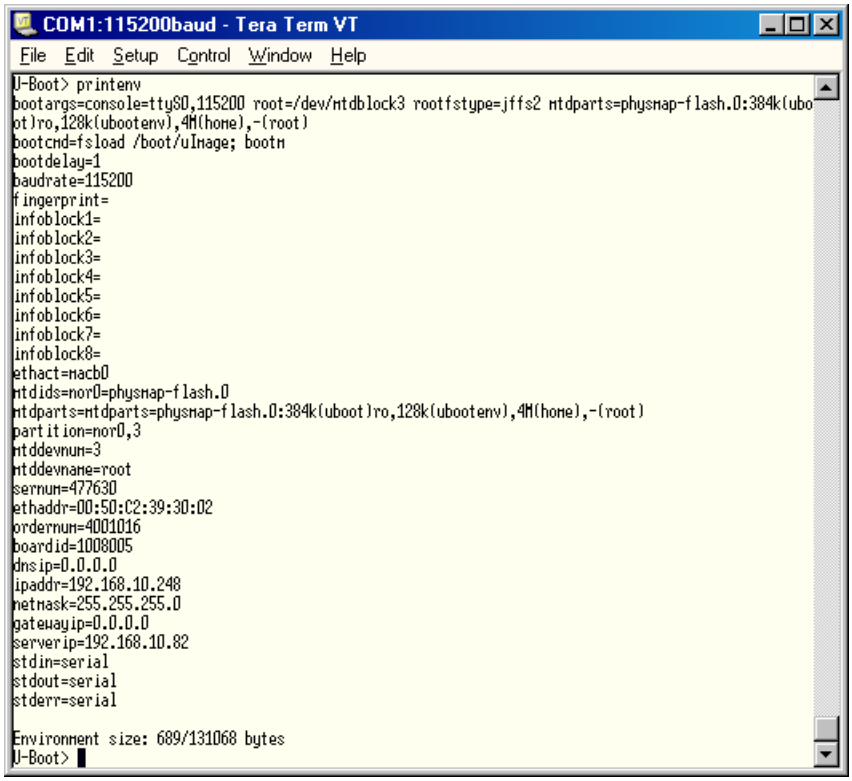


Bild 8: Anzeigen der aktuellen Modulkonfiguration im Bootloader "U-Boot"

Auf die vom Bootloader "U-Boot" verwalteten Konfigurationseinstellungen kann auch aus Linux zugegriffen werden, die Vorgehensweise dazu beschreibt Abschnitt 5.10.

5.4 Ethernet-Konfiguration des ECUcore-9G20

Die zentrale Ethernet-Konfiguration des ECUcore-9G20 erfolgt im Bootloader "U-Boot" und wird von hier für alle anderen Softwarekomponenten übernommen (Linux, Anwendersoftware, HTTP-Server usw.). Die Ethernet-Konfiguration erfolgt über die serielle Schnittstelle COM0. **Dazu ist wie im Abschnitt 5.2.1 beschrieben der "U-Boot"-Kommandoprompt zu aktivieren.** Tabelle 5 listet die zur Ethernet-Konfiguration des ECUcore-9G20 notwendigen "U-Boot"-Kommandos auf.

Tabelle 5: "U-Boot" Kommandos zur Konfiguration des ECUcore-9G20

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 des ECUcore-9G20	setenv ipaddr <xxx.xxx.xxx.xxx>	Dieses Kommando setzt die lokale IP-Adresse des ECUcore-9G20. Die IP-Adresse ist vom Netzwerkadministrator festzulegen. Um dem ECUcore-9G20 eine dynamische IP-Adresse über DHCP zuzuweisen, ist für die Adresse der Wert "0.0.0.0" anzugeben. **Siehe Hinweis zu DHCP unten im Text!**
Netzwerkmaske	setenv netmask <xxx.xxx.xxx.xxx>	Dieses Kommando setzt die Netzwerkmaske des ECUcore-9G20. Die Netzwerkmaske ist vom Netzwerkadministrator festzulegen.
Gateway-Adresse	setenv gatewayip <xxx.xxx.xxx.xxx>	Dieses Kommando definiert die IP-Adresse des vom ECUcore-9G20 zu benutzenden Gateways. Die Gateway-Adresse ist vom Netzwerkadministrator festzulegen. **Hinweis:** Befinden sich ECUcore-9G20 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.
IP-Adresse des Linux-Entwicklungs-systems	setenv serverip <xxx.xxx.xxx.xxx>	Dieses Kommando definiert die IP-Adresse des Linux-Entwicklungssystems. Sie wird beispielsweise zum Update des Linux-Images (siehe Abschnitt 5.15) sowie zum Einbinden des Linux-Entwicklungssystems per NFS in das lokale Filesystem des ECUcore-9G20 (siehe Abschnitt 7.5.1) benutzt. Die Vorgehenswiese zum Ermitteln der IP-Adresse des Linux-Entwicklungssystems beschreibt Abschnitt 6.5.
Speichern der Konfiguration	saveenv	Dieses Kommando speichert die aktuellen Einstellungen im Flash des ECUcore-9G20.

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 ECUcore-9G20 gespeichert. Die Änderungen werden mit dem nächsten Reset des ECUcore-9G20 übernommen.

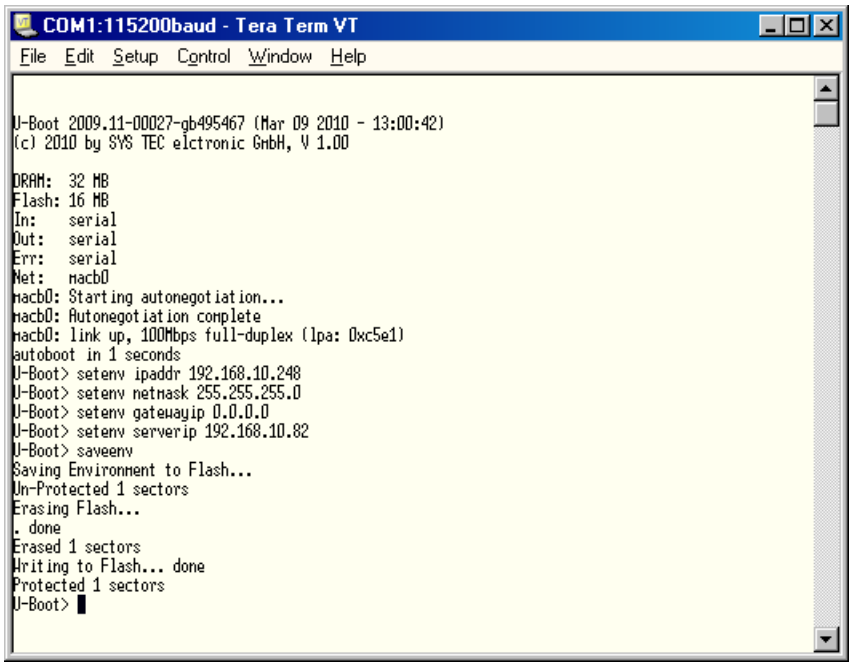


Bild 9: Speichern der Modulkonfiguration für das ECUcore-9G20

Hinweis zur Verwendung von DHCP:

Das Embedded Linux des ECUcore-9G20 ist in der Lage, über DHCP eine dynamische IP-Adresse anzufordern. Dazu ist die lokale IP-Adresse als "0.0.0.0" zu konfigurieren:

```
setenv ipaddr 0.0.0.0
```

Bei der Verwendung von DHCP sind folgende Punkte zu beachten:

- DHCP wird nur von Linux unterstützt, nicht aber vom Bootloader "U-Boot". Um beispielsweise ein Update des Linux-Images vorzunehmen (siehe Abschnitt 5.15) ist in jedem Fall die Zuweisung einer (temporären) statischen IP-Adresse an das Modul notwendig.

- Um eine Telnet- oder FTP-Verbindung zum ECUcore-9G20 aufzubauen, muss dessen IP-Adresse bekannt sein. Die momentane, über DHCP dynamisch zugewiesene Adresse kann mit Hilfe eines **Terminalprogramms** (z.B. HyperTerminal oder TeraTerm, siehe Abschnitt 5.1) über die serielle Schnittstelle **COM0** des ECUcore-9G20 ermittelt werden. Hierzu ist zunächst die im Abschnitt 5.5.1 beschriebenen Anmeldung an der Kommando-Shell des ECUcore-9G20 durchzuführen. Anschließend lässt sich die derzeitige IP-Adresse mit Hilfe des Kommandos *"ifconfig"* abfragen.

Nach Abschluss der Konfiguration sind gemäß Abschnitt 5.2.1 die Voraussetzungen für einen Linux-Autostart wieder herzustellen.

Nach Reset (z.B. Taster S405 am Developmentboard) startet das Modul mit den aktuellen Einstellungen.

Hinweis: Nach Abschluss der Konfiguration ist die serielle Verbindung zwischen PC und ECUcore-9G20 nicht mehr erforderlich.

5.5 Anmeldung am ECUcore-9G20

5.5.1 Anmeldung an der Kommando-Shell

Zur Administration sowie für die Softwareentwicklung ist die Eingabe von Linux-Kommandos in der Kommando-Shell des ECUcore-9G20 erforderlich. Dazu ist eine direkte Anmeldung am Modul zwingend notwendig. Dies kann auf zwei alternativen Wegen erfolgen:

- Mit Hilfe eines **Terminalprogramms** (z.B. HyperTerminal oder TeraTerm, siehe Abschnitt 5.1) über die serielle Schnittstelle **COM0** des ECUcore-9G20, analog zum Vorgehen bei der im Abschnitt 5.4 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 ECUcore-9G20 möglich.

Um sich über den in Windows standardmäßig enthaltenen Telnet-Client am ECUcore-9G20 anzumelden, ist der Befehl "*telnet*" unter Angabe der in Abschnitt 5.4 festgelegten IP-Adresse für das ECUcore-9G20 aufzurufen, z.B.

```
telnet  192.168.10.248
```

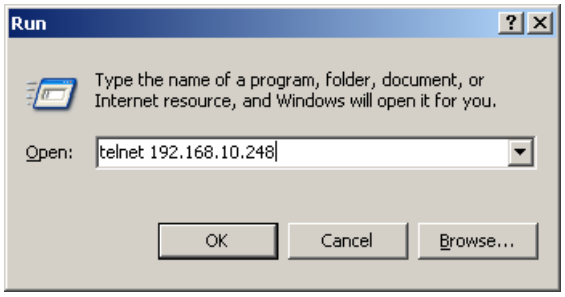


Bild 10: Aufruf des Telnet-Clients unter Windows

Innerhalb des Terminal-Fensters (bei Verbindung über COM0) bzw. des Telnet-Fensters (bei Verwendung von ETH0) ist die Anmeldung am ECUcore-9G20 möglich. Bei Auslieferung des Moduls ist zur Administration des ECUcore-9G20 folgendes Nutzerkonto vorkonfiguriert (siehe auch Abschnitt 5.6):

User: *PlcAdmin*
Passwort: *Plc123*

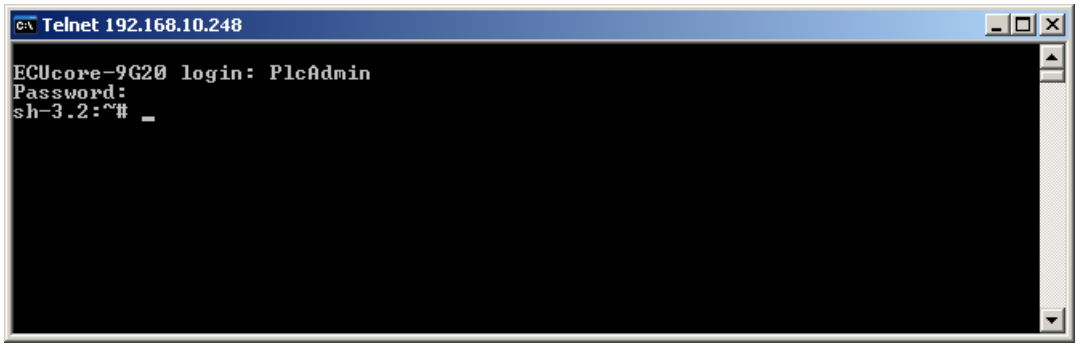


Bild 11: Anmeldung am ECUcore-9G20

Bild 11 verdeutlicht am Beispiel die Anmeldung am ECUcore-9G20 mit Hilfe des in Windows standardmäßig enthaltenen Telnet-Clients.

5.5.2 Anmeldung am FTP-Server

Das ECUcore-9G20 verfügt über einen FTP-Server (FTP Deamon), der den Austausch von Dateien mit einem beliebigen FTP-Client ermöglicht (z.B. Up- und Download von Dateien zu bzw. von einem PC). Aus Sicherheits- und Performance-Gründen ist dieser FTP-Server jedoch standardmäßig deaktiviert und muss bei Bedarf erst manuell gestartet werden. Hierzu ist zunächst die im Abschnitt 5.5.1 beschriebene Anmeldung an der Kommando-Shell des ECUcore-9G20 durchzuführen. Anschließend ist im Telnet- bzw. Terminal-Fenster folgendes Kommando einzugeben:

```
pureftp
```

Bild 12 verdeutlicht am Beispiel das Starten des FTP-Servers (*"pureftp"*).

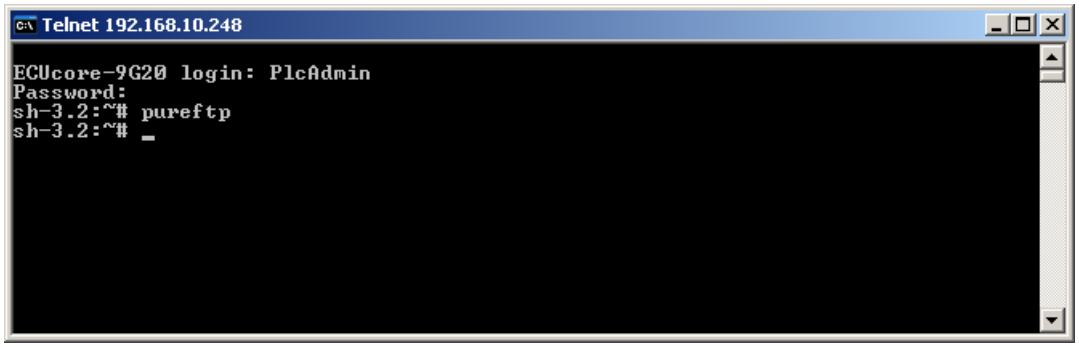


Bild 12: Starten des FTP-Servers

Hinweis: Durch Einfügen des Kommandos *"pureftp"* in das Startskript **"/home/etc/autostart"** lässt sich der Aufruf des FTP-Servers beim Booten des ECUcore-9G20 automatisieren (siehe dazu Abschnitt 5.2.2).

Als FTP-Client für einen Windows-PC eignet sich beispielsweise das als Open-Source verfügbare *"WinSCP"* (siehe Abschnitt 5.1), 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 13), in dem folgende Einstellungen vorzunehmen sind:

File protocol:	FTP
Host name:	die im Abschnitt 5.4 festgelegte IP-Adresse für das ECUcore-9G20
User name:	*PlcAdmin* (für vorkonfiguriertes Nutzerkonto, siehe Abschnitt 5.6)
Password:	*Plc123* (für vorkonfiguriertes Nutzerkonto, siehe Abschnitt 5.6)

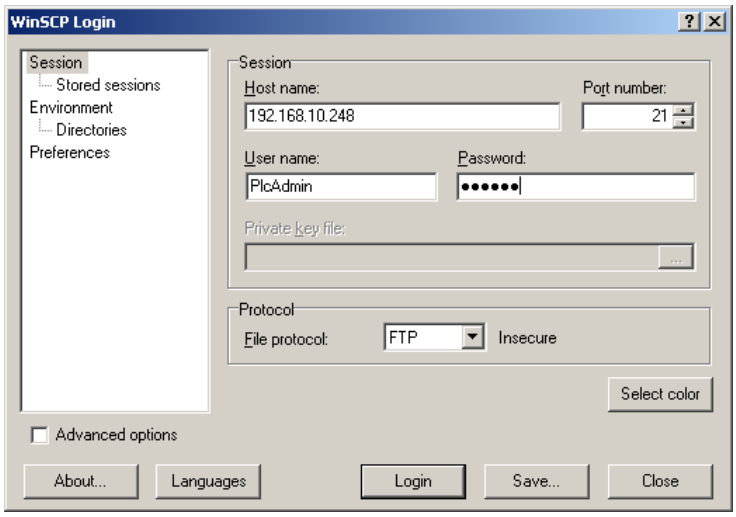


Bild 13: Login-Einstellungen für WinSCP

Nach dem Betätigen der Schaltfläche *"Login"* meldet sich der FTP-Client am ECUcore-9G20 an listet im rechten Fenster den aktuellen Inhalt des Verzeichnisses *"/home"* auf. Bild 14 zeigt den FTP-Client *"WinSCP"* nach der erfolgreichen Anmeldung am ECUcore-9G20.

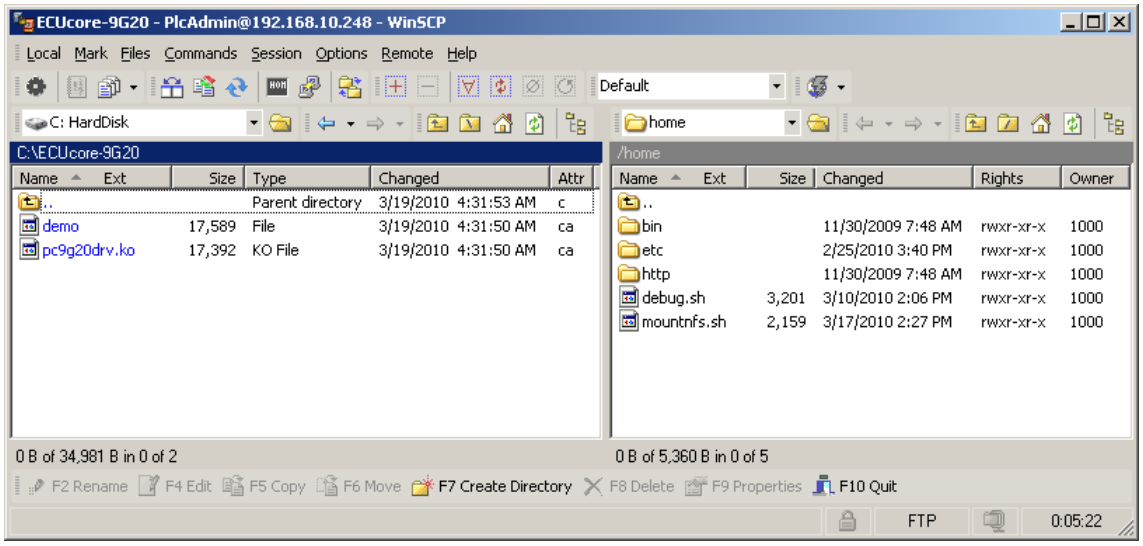


Bild 14: FTP-Client für Windows "WinSCP"

Nach einer erfolgreichen Anmeldung lassen sich innerhalb des FTP-Clients *"WinSCP"* mit Hilfe der Taste *"F4"* bzw. der Schaltfläche *"F4 Edit"* Konfigurationsdateien auf dem ECUcore-9G20 bearbeiten (dazu Transfermodus *"Text"* selektieren). Mit Hilfe der Taste *"F5"* bzw. der Schaltfläche *"F5 Copy"* können Dateien zwischen dem PC und dem ECUcore-9G20 transferieren werden, um beispielsweise Datensicherungen des ECUcore-9G20 durchzuführen oder um Software-Updates auf das Modul zu übertragen (dazu Transfermodus *"Binary"* selektieren).

5.6 Vordefinierte Nutzerkonten

Bei der Auslieferung des ECUcore-9G20 sind die in Tabelle 6 aufgelisteten Benutzerkonten vordefiniert. Diese erlauben eine Anmeldung an der Kommando-Shell (serielle RS232-Verbindung oder Telnet) und am FTP-Server des ECUcore-9G20.

Tabelle 6: Vordefinierte Benutzerkonten des ECUcore-9G20

Benutzername	Passwort	Bemerkung
PlcAdmin	Plc123	vordefiniertes Benutzerkonto zur Administration des ECUcore-9G20 (Konfiguration, Nutzerverwaltung, Softwareupdates usw.)
root	Sys123	Haupt-Benutzerkonto ("root") des ECUcore-9G20

5.7 Anlegen und Löschen von Nutzerkonten

Das Anlegen und Löschen von Nutzerkonten erfordert zunächst die Anmeldung am ECUcore-9G20 wie in Abschnitt 5.5.1 beschrieben.

Das **Anlegen** eines neuen Nutzerkontos erfolgt mit Hilfe des Linux-Kommandos *"adduser"*. Da es bei einem Embedded System wie dem ECUcore-9G20 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 ECUcore-9G20 ist das Linux-Kommandos *"adduser"* wie folgt anzuwenden:

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

Bild 15 verdeutlicht am Beispiel das Anlegen eines neuen Kontos auf dem ECUcore-9G20 für den Nutzer *"admin2"*.

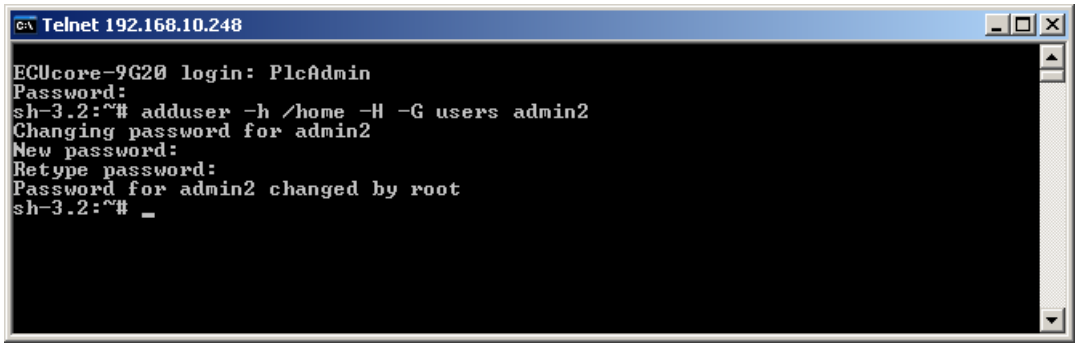


Bild 15: Anlegen eines neuen Nutzerkontos

Zum **Löschen** eines vorhandenen Nutzerkontos vom ECUcore-9G20 ist das Linux-Kommando *"deluser"* unter Angabe des Benutzernamens zu verwenden:

```
deluser <username>
```

5.8 Passwort eines Nutzerkontos ändern

Das Ändern von Passwörtern erfordert zunächst die Anmeldung am ECUcore-9G20 wie in Abschnitt 5.5.1 beschrieben.

Zum Ändern des Passwortes für ein auf dem ECUcore-9G20 vorhandenen Nutzerkonto ist das Linux-Kommando *"passwd"* unter Angabe des Benutzernamens zu verwenden:

```
passwd <username>
```

Bild 16 verdeutlicht am Beispiel das Ändern des Passwortes für den Nutzer *"PlcAdmin"*.

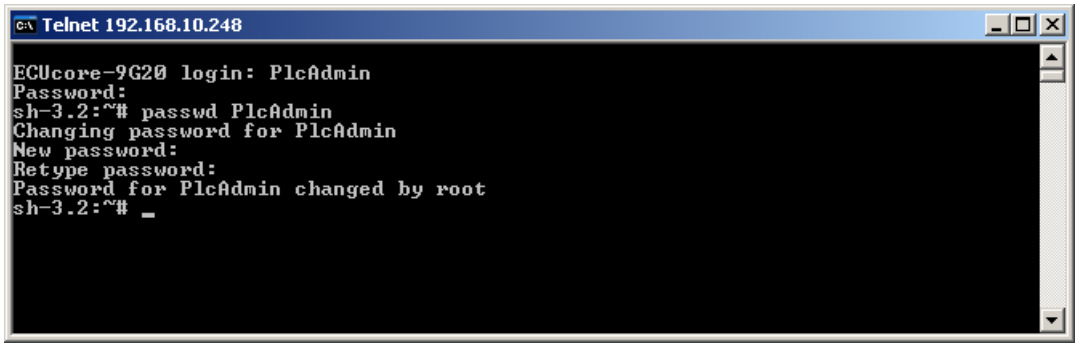


Bild 16: Passwort eines Nutzerkontos ändern

5.9 Setzen der Systemzeit

Das Setzen der Systemzeit erfordert zunächst die Anmeldung am ECUcore-9G20 wie in Abschnitt 5.5.1 beschrieben.

Das Setzen der Systemzeit für das ECUcore-9G20 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 ECUcore-9G20 übernommen.

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

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

Beispiel:

```
date   2010.02.25-11:34:55
          |  |  |  |  |  |
          |  |  |  |  |  +--- Sekunde
          |  |  |  |  +------ Minute
          |  |  |  +--------- Stunde
          |  |  +------------ Tag
          |  +--------------- Monat
          +------------------ Jahr
```

Um die aktuelle Systemzeit des ECUcore-9G20 wie im obigen Beispiel dargestellt auf den 2010/02/25 um 11:34:55 zu setzen, ist folgende Befehlssequenz notwendig:

```
date  2010.02.25-11:34:55
hwclock  -w
```

Die aktuelle Systemzeit wird durch Eingabe des Linux-Kommandos *"date"* (ohne Parameter) angezeigt, die aktuellen Werte der RTC lassen sich durch das Linux-Kommando *"hwclock -r"* abfragen. Mit *"hwclock -s"* werden die aktuellen Werte der RTC als Systemzeit für Linux übernommen (Synchronisation des Kernels auf die RTC). Bild 17 verdeutlicht das Setzen und Anzeigen der Systemzeit am Beispiel.

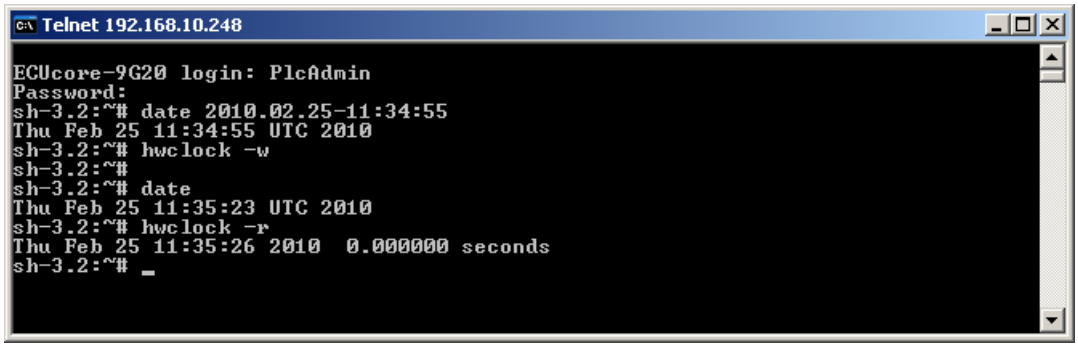


Bild 17: Setzen und Anzeigen der Systemzeit

Beim Starten des ECUcore-9G20 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.

5.10 Auslesen und Anzeigen von "U-Boot"-Konfigurationsdaten

Das Kommando *"fw_printenv"* ermöglicht unter Linux den Zugriff auf die Konfigurationsdaten des Bootloaders "U-Boot". Es wird beispielsweise genutzt, um im Startskript "/etc/rc.d/S05network" die im "U-Boot" für das ECUcore-9G20 festgelegte Ethernet-Konfiguration (siehe Abschnitt 5.4) zur Parametrierung des Interfaces "ETH0" unter Linux weiter zu verwenden. Gleiches gilt auch für die Übernahme der im "U-Boot" definierten Server-Adresse in den Shell-Skripten *"/home/mountnfs.sh"* und *"/home/debug.sh"*.

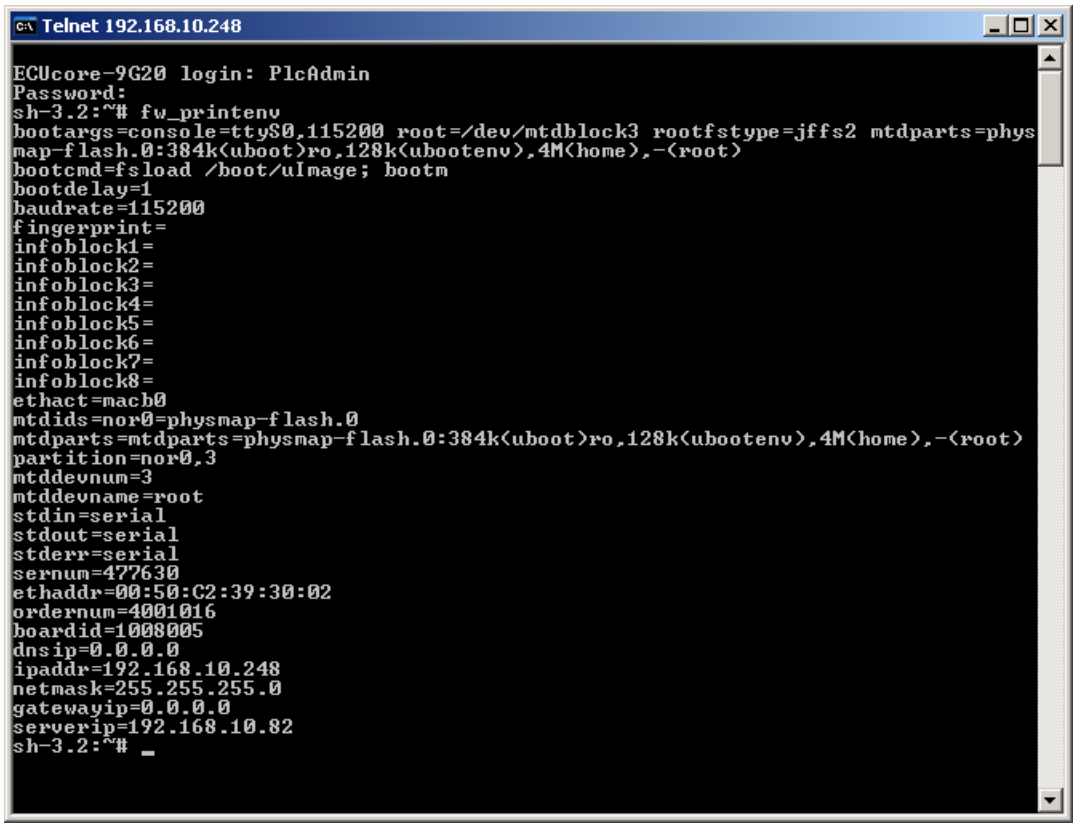


Bild 18: Anzeigen der "U-Boot"-Konfigurationsdaten unter Linux mittels "fw_printenv"

Der Aufruf von *"fw_printenv"* ohne Angabe von Parametern zeigt alle "U-Boot"-Konfigurationsdaten an (siehe Bild 18). Die gezielten Abfrage einzelner Konfigurationsinformationen ist durch den Aufruf von *"fw_printenv <paramname>"* möglich (z.B. "*fw_printenv ipaddr*"). Die oben angeführten Skripte verdeutlichen die Anwendung von *"fw_printenv"*, demonstrieren die Zuweisung der abgefragten Informationen an Umgebungsvariablen und zeigen die Auswertung der Informationen in einem Shell-Skript.

5.11 Anzeigen der installierten Linux-Version

Die auf dem ECUcore-9G20 installierte Linux-Version wird durch Aufruf des Kommandos *"version"* angezeigt (siehe Bild 19).

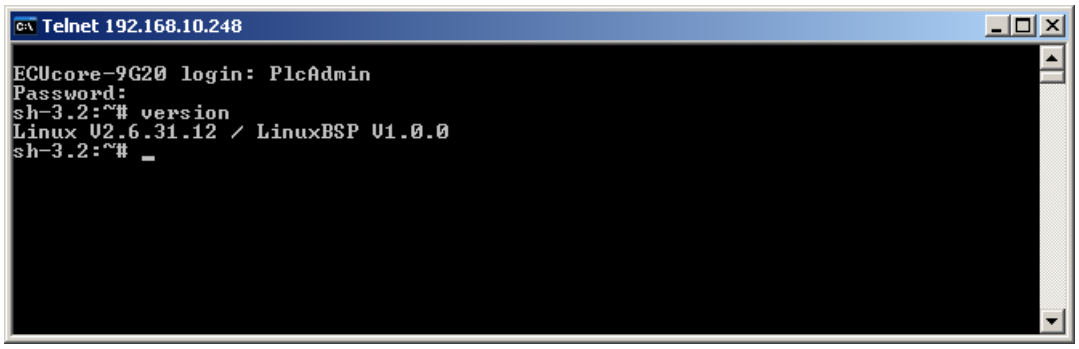


Bild 19: Anzeigen der installierten Linux-Version

5.12 Dateisystem des ECUcore-9G20

Das auf dem ECUcore-9G20 vorinstallierte Embedded Linux stellt den überwiegenden Teil des on-board Speichers in Form einer Flash-Disk zur Verfügung. Als Dateisystem wird dabei JFFS2 verwendet, ein im embedded Bereich weit verbreitetes System, das speziell für den Einsatz von Flash-Bausteinen als Datenträger entwickelt wurde. Der on-board Flash des ECUcore-9G20 ist in zwei unabhängige JFFS2-Partitionen aufgeteilt. Die erste Partition beinhaltet den Linux-Kernel und das Root-Filesystem. Die zweite Partition wird komplett als Verzeichnis *"/home"* eingebunden. Auf beide Partitionen besteht voller Schreib-/Lesezugriff.

Um ein problemloses Update des Linux-Images inkl. Root-Filesystem zu gewährleisten, sollten anwenderspezifische Daten wie beispielsweise Konfigurationsdateien und die auf das Modul geladenen Anwenderprogramme ausschließlich im Verzeichnis ***"/home"*** abgelegt werden, das physikalisch eine eigenständige, vom Root-Filesystem unabhängige Partition bildet. Alle Files, die in anderen Verzeichnissen abgelegt wurden, gehen bei einem Update verloren.

Für Tests während der Entwicklungsphase sollte möglichst eines der beiden RAM-Disk-Verzeichnisse *"/tmp"* oder *"/var/log"* benutzt werden, falls nicht ohnehin Teile des Linux-Entwicklungssystems via NFS eingebunden sind (siehe Abschnitt 7.5.1).

Tabelle 7: Dateisystemkonfiguration des ECUcore-9G20

Zweig	Nutzbare Größe	Beschreibung
/home	4096 kByte	Flash-Disk, zur persistenten Ablage vom Anwender modifizierbarer und updatefähiger Files (z.B. Anwendersoftware und Konfigurationsdateien), wird bei Update von Linux-Kernel und Root-Filesystem nicht überschrieben, Datenerhalt bei Spannungsausfall ist gegeben
/tmp	8192 kByte	RAM-Disk, geeignet als Zwischenpuffer für FTP-Downloads, aber kein Datenerhalt bei Spannungsausfall
/var/log	1024 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 anderer Systeme via NFS, siehe dazu auch Abschnitt 7.5.1

Hinweis: Die Größen für die Dateisystemzweige *"/tmp"* und *"/var/log"* können durch Anpassung in der Konfigurationsdatei *"/etc/fstab"* geändert werden.

Die konfigurierten sowie jeweils aktuell noch verfügbaren Größen der einzelnen Dateisystemzweige können mit Hilfe des Linux-Kommandos "df" ("DiskFree") ermittelt werden (siehe Bild 20).

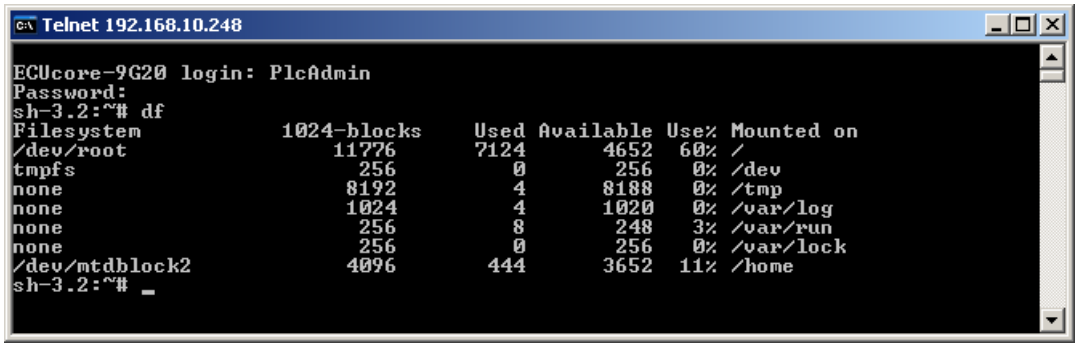


Bild 20: Anzeige von Informationen zum Dateisystem

Einzelheiten zur Anmeldung am System und zum Umgang mit der Linux-Kommando-Shell des ECUcore-9G20 behandelt Abschnitt 5.5.1.

5.13 Vorinstallierte Files im Verzeichnis "/home"

An das Verzeichnis *"/home"* ist eine Flash-Disk gebunden ("mounted"), die einen Teil des on-board Flash-Speichers des ECUcore-9G20 als Dateisystem zur Verfügung stellt. Dieser Pfad ist zur Laufzeit beschreibbar und dient zur persistenten Ablage modifizierbarer und updatefähiger Files wie Konfigurationsdateien oder Anwenderprogramme (siehe auch Abschnitt 5.12). In seinem Auslieferungszustand enthält das ECUcore-9G20 im Verzeichnis *"/home"* folgende Files:

/home	
+- etc	
+- autostart	Skript zum automatisierten Start von Anwendersoftware (siehe Abschnitt 5.2.2)
+- rc.usr	optionales anwenderspezifisches Konfigurationsskript
+- hotplug.sh +- diskadded.sh +- diskremoved.sh	Skripte zur Reaktion auf Anstecken/Abziehen von USB-Sticks (siehe Abschnitt 9.1)
+- bin	
+- pc9g20drv.ko	Kernelspace-Modul des I/O-Treibers (siehe Abschnitt 7.3.1)
+- iodrvdemo	Userspace-Programm zum Testen der Hardwareanschaltung (siehe Abschnitt 8.2)
+- candrv.ko	Kernelspace-Modul des CAN-Treibers (siehe Abschnitt 7.4.1)
+- http	Demo-Dateien für HTTP-Server (siehe Abschnitt 5.14)
+- mountnfs.sh	Skript zum vereinfachten Einbinden von NFS-Verzeichnissen in das Filesystem des ECUcore-9G20 (siehe Abschnitt 7.5.1)
+- debug.sh	Skript zum vereinfachten Starten des Demoprogramms unter Kontrolle des Debugservers (siehe Abschnitt 7.6.2.3)

Bei Bedarf kann der Auslieferungszustand aller Files im Verzeichnis *"/home"* durch Ausführen des Setup-Skripts *"setup-ecucore-9g20.sh"* wieder restauriert werden. Das Setup-Skript *"setup-ecucore-9g20.sh"* befindet sich im Verzeichnis *"SetupFlashdisk_ECUcore-9G20"* der DVD "SO-1105". Im Abschnitt 7.5 sind verschiedene alternative Möglichkeiten zur Übertragung des Files auf das ECUcore-9G20 beschrieben.

5.14 Nutzung des HTTP-Servers

Auf dem ECUcore-9G20 ist der HTTP-Server *"boa"* installiert. Dadurch ist der Zugriff auf das Modul mit einem beliebigen WEB-Browser möglich (z.B. Microsoft Internet Explorer, Mozilla Firefox usw.). Die Konfiguration des Servers erfolgt mit Hilfe der Datei *"boa.conf"*. Beim Starten des Servers ist das Verzeichnis, das diese Datei enthält, über den Kommandozeilenparameter *"-c"* anzugeben. Im Auslieferungszustand enthält das ECUcore-9G20 im Verzeichnis *"/home/http"* vorgefertigte Demo-Dateien, mit deren Hilfe sich die Anwendung des HTTP-Servers veranschaulichen lässt. Um diese Demo-Konfiguration zu aktivieren, ist der HTTP-Server *"boa"* manuell unter Angabe des Demo-

Verzeichnisses zu starten. Hierzu ist zunächst die im Abschnitt 5.5.1 beschriebenen Anmeldung an der Kommando-Shell des ECUcore-9G20 durchzuführen. Anschließend ist im Telnet- bzw. Terminal-Fenster folgendes Kommando einzugeben:

```
boa  -c  /home/http
```

Bild 21 verdeutlicht den Start des HTTP-Servers *"boa"* am Beispiel der im Auslieferungszustand das ECUcore-9G20 enthaltenen, vorgefertigten Demo-Konfiguration.

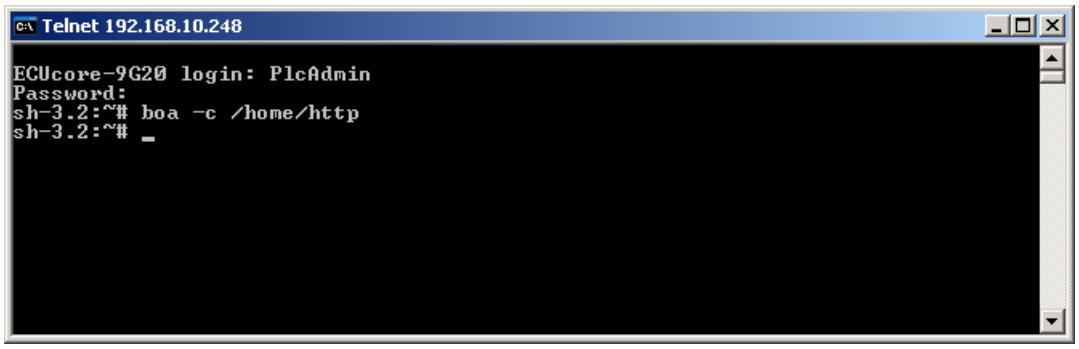


Bild 21: Starten des HTTP-Servers "boa"

Zum Abrufen der durch den HTTP-Server bereitgestellten Seiten ist in der Adressleiste des WEB-Browser der Präfix *"http://"* gefolgt von der im Abschnitt 5.4 festgelegte IP-Adresse für das ECUcore-9G20 anzugeben, z.B. *"http://192.168.10.248"*. Bild 22 verdeutlicht die Anzeige von HTML-Seiten des ECUcore-9G20 im WEB-Browser.

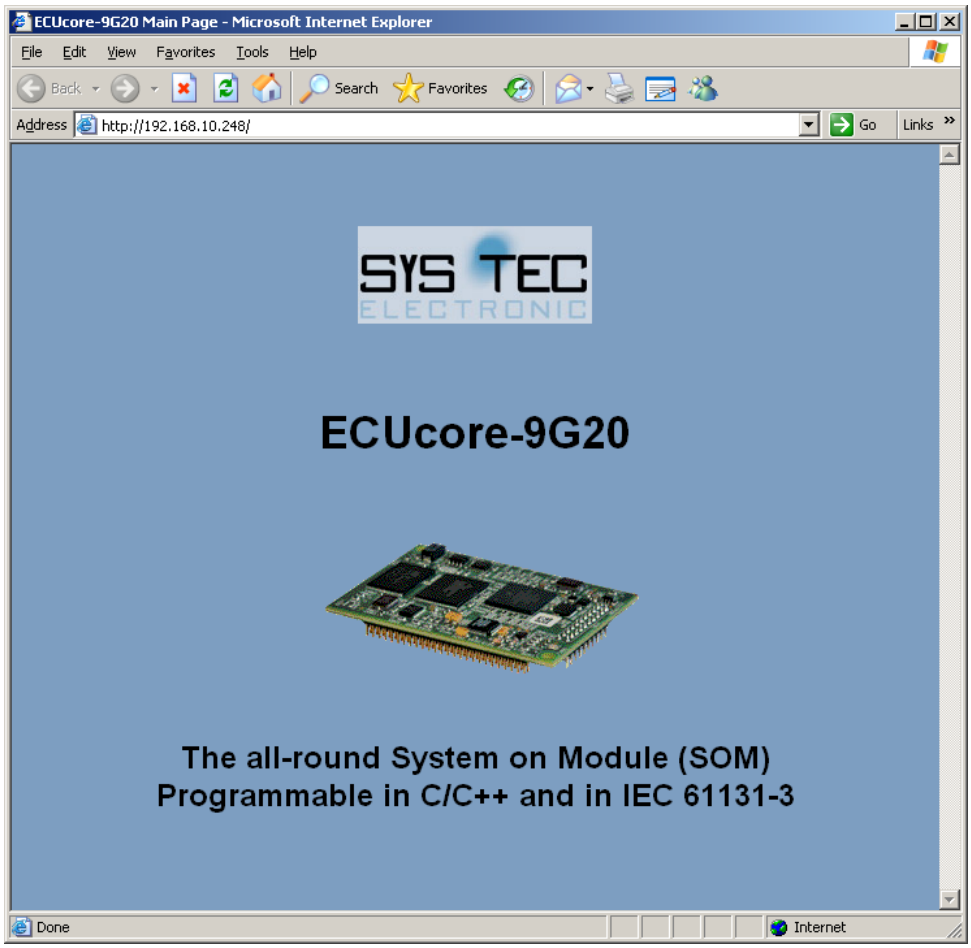


Bild 22: Anzeige von HTML-Seiten des ECUcore-9G20 im WEB-Browser

Hinweis: Durch Einfügen des Startaufrufes für den HTTP-Server (z.B. *"boa -c /home/http"*) in das Startskript **"/home/etc/autostart"** lässt sich der Aufruf des HTTP-Servers beim Booten des ECUcore-9G20 automatisieren (siehe dazu Abschnitt 5.2.2).

5.15 Update des Linux-Images

Der Update des Linux-Images erfolgt via TFTP (**T**rivial **FTP**) innerhalb des Linux-Bootloaders *"U-Boot"*. Auf dem Entwicklungs-PC ist hierfür ein entsprechender TFTP-Server notwendig. Standardmäßig ist ein entsprechend konfigurierter TFTP-Server bereits im VMware-Image des Linux-Entwicklungssystems enthalten. Um alternativ ein Update des Linux-Images von einem Windows-PC durchführen zu können, bietet sich beispielsweise die Freeware *"TFTPD32"* an (siehe Abschnitt 5.1). Das Windows-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:\ECUcore-9G20"*). In diesem Verzeichnis muss sich das Linux-Image für das ECUcore-9G20 befinden (*"root.sum.jffs2"*).

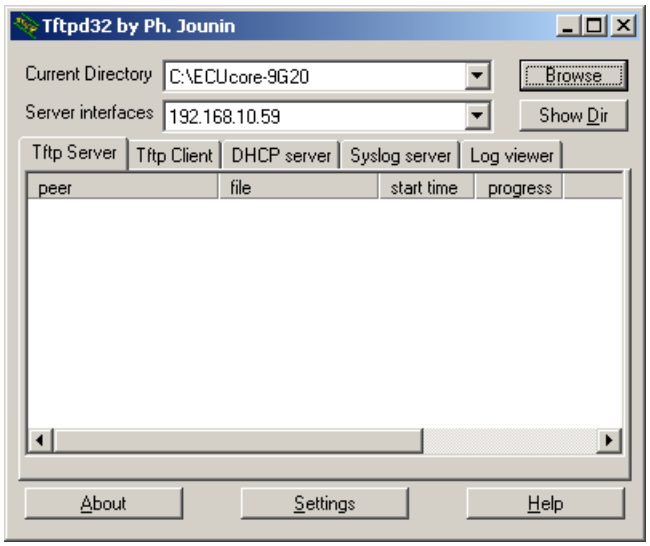


Bild 23: TFTP-Server für Windows "TFTPD32"

Voraussetzung für den TFTP-Download des Linux-Images **ist die abgeschlossenen Ethernet-Konfiguration** des ECUcore-9G20 **gemäß Abschnitt 5.4**. Für das Update des Linux-Images ist neben der Ethernet-Verbindung noch eine serielle Verbindung zum ECUcore-9G20 erforderlich. Hier gelten ebenfalls die im Abschnitt 5.2.1 beschriebenen Einstellungen für das Terminalprogramm (115200 Baud, 8 Datenbits, 1 Stopbit, keine Parität, keine Flusskontrolle).

Ein Update des Linux-Images auf dem ECUcore-9G20 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 5.2.1.

Nach Reset (z.B. Taster S405 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 8: Kommando-Sequenz zum Update des Linux-Images auf dem ECUcore-9G20

Kommando	Bedeutung
setenv serverip <host_ip_addr>	Setzen der IP-Adresse des TFTP-Servers. Bei Verwendung des TFTP-Servers im VMware-Image des Linux-Entwicklungssystems ist die zu verwendende IP-Adresse mit Hilfe des Linux-Kommandos *"ifconfig"* zu ermitteln (siehe Abschnitt 6.5), bei Verwendung des Windows-Programms *"TFTPD32"* wird diese auf dem PC im Feld "Server Interface" angezeigt
tftp root.sum.jffs2	Download des Linux-Images vom Entwicklungs-PC auf das ECUcore-9G20
erase nor0,3	Löschen des vom Linux-Image benötigten Flash-Bereiches
cp.b ${fileaddr} 0x10480000 ${filesize}	Speichern des Linux-Images im Flash des ECUcore-9G20

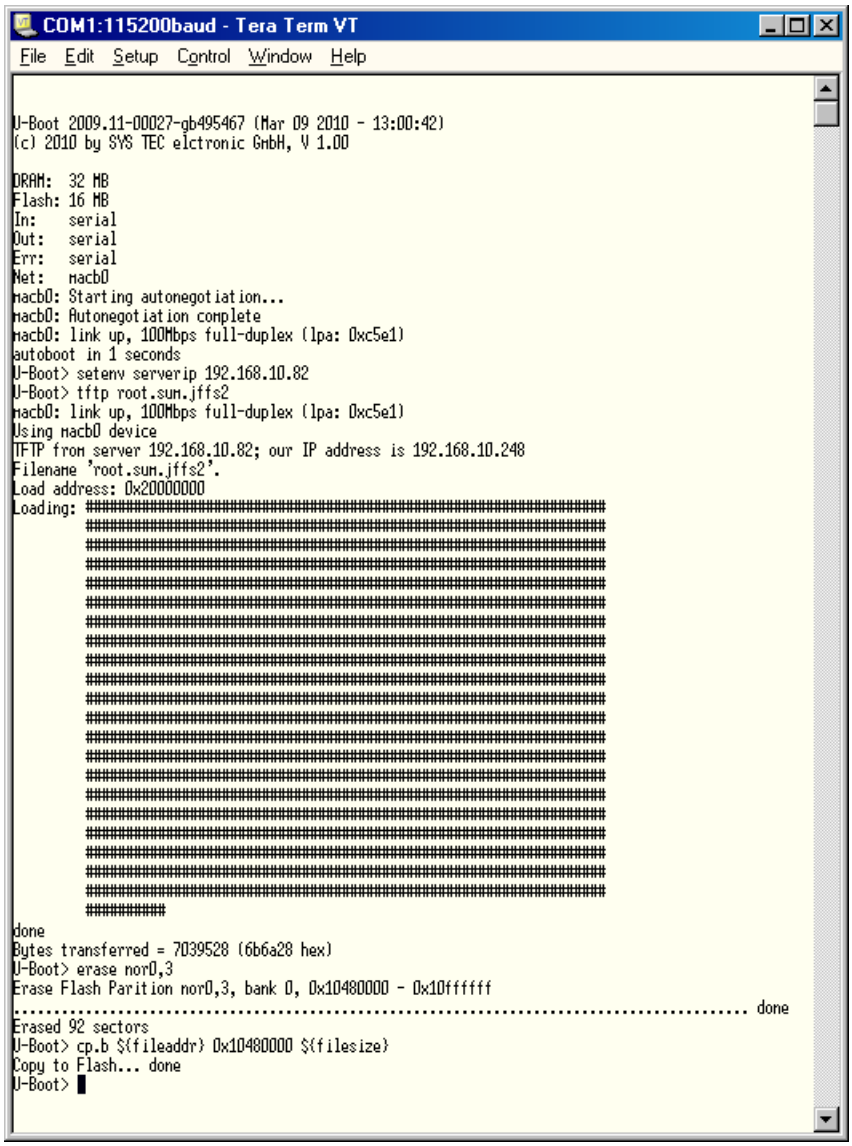


Bild 24: Download des Linux-Images auf das ECUcore-9G20

Nach Abschluss der Konfiguration sind gemäß Abschnitt 5.2.1 die Voraussetzungen für einen Linux-Autostart wieder herzustellen.

Nach Reset (z.B. Taster S405 am Developmentboard) startet das ECUcore-9G20 mit dem aktuellen Linux-Image.

Hinweis: Nach Abschluss der Konfiguration ist die serielle Verbindung zwischen PC und ECUcore-9G20 nicht mehr erforderlich.

5.16 Update des Bootloaders "U-Boot"

Ein Update des Bootloaders "U-Boot" ist zwar prinzipiell möglich, birgt aber die Gefahr, dass nach einem missglückten Update oder nach dem Einspielen einer fehlerhaften Software das ECUcore-9G20 nicht mehr startet. Aus diesem Grund sind auch die Flashsektoren, in denen der "U-Boot" abgelegt ist, gegen versehentliches Löschen geschützt.

Ein Update des Bootloaders "U-Boot" sollte daher nur in dringend begründeten Fällen vorgenommen werden. Bitte setzen Sie sich bei Bedarf dazu mit unserem Support in Verbindung:

support@systec-electronic.com

6 VMware-Image des Linux-Entwicklungssystems

6.1 Übersicht

Das ECUcore-9G20 wird mit einem vorinstallierten Embedded Linux ausgeliefert. Anwendungen, die auf diesem Modul ausgeführt werden sollen, sind dementsprechend als Linux-Programme zu entwickeln. Das Kit beinhaltet ein komplett eingerichtetes Linux-Entwicklungssystem in Form eines VMware-Images und ermöglicht so einen leichten Einstieg in die Softwareentwicklung für das ECUcore-9G20. Das VMware-Image kann dabei unverändert unter verschiedenen Host-Systemen benutzt werden. Geeignete Nachschlagewerke zur Linux-Programmierung sind in Tabelle 1 im Abschnitt 2 aufgeführt.

Das im VMware-Image des Linux-Entwicklungssystems beinhaltet folgende Softwarekomponenten:

- GNU-Crosscompiler Toolchain für ARM9-Prozessoren
- Linux-Sourcecode für das ECUcore-9G20 (LinuxBSP)
- Eclipse (grafische IDE zur Vereinfachung der Softwareentwicklung)
- Samba-Server (ermöglicht Zugriff "von außen" über Windows-Netzwerkumgebung)
- FTP-Server (ermöglicht Benutzung einer Linux-Konsole "von außen", in Form eines Telnet-Client unter Windows sowie die Dateiübertragung zwischen ECUcore-9G20 und Entwicklungs-PC)
- TFTP-Server (ermöglicht Download des Linux-Images für das ECUcore-9G20 vom Entwicklungs-PC)
- NFS-Server (ermöglicht Einbindung des Entwicklungs-PC in das lokale Dateisystem des ECUcore-9G20)

6.2 Installation des Linux-VMware-Images

Das Development Kit ECUcore-9G20 beinhaltet auf der DVD "SO-1105" sowohl das VMware-Image des Linux-Entwicklungssystems für das ECUcore-9G20 als auch den "VMware Player" für Windows. Der "VMware Player" ist eine kostenlose Software zur Desktop-Virtualisierung, mit deren Hilfe das VMware-Image des Linux-Entwicklungssystems auf einem Windows- oder Linux-PC ausgeführt werden kann. Bei Bedarf kann die jeweils aktuelle Version des "VMware Players" bzw. Player für andere Hostsysteme auch direkt von der Homepage des Herstellers unter http://www.vmware.com herunter geladen werden. Zur Installation des VMware Players ist das entsprechende Setup-Programm auszuführen.

Hinweis: Bei der Installation des "VMware Players" sollte unbedingt die Standardeinstellung "Bridged Mode" für das Ethernet-Interface beibehalten werden, da andernfalls unter Umständen keine Kommunikation zum ECUcore-9G20 möglich sein kann.

Das VMware-Image ist auf der DVD als selbstentpackendes Archiv ***"SO-1105.exe"*** enthalten. Beim Starten von *"SO-1105.exe"* werden sämtliche zum VMware-Image gehörenden Dateien auf die lokale Festplatte entpackt. Das dekomprimierte Image beansprucht ca. 10 GByte.

6.3 Starten des Linux VMware-Images

Auf dem Host-PC ist zunächst der "VMware Player" zu starten. Im Programmfenster des Players ist dann mit Hilfe des "Open"-Symbols das File *"Xubuntu-ECUcore9G20.vmx"* zu öffnen (siehe Bild 25).

Bild 25: Programmfenster des VMware Players mit "Open"-Symbol

VMware speichert beim Ausführen eines Images den "Fingerabdruck" des Hostrechners in Form einer UUID im File **.vmx*. Beim Starten des Linux VMware-Images auf einem anderen PC erscheint der in Bild 26 dargestellte Dialog. Wenn dasselbe Linux-Image nicht zeitgleich auf einem weiteren Rechner im selben Netzwerk ausgeführt wird, dann sollte die Option *"I moved it"* gewählt werden. Dabei bleibt die bisher verwendete MAC-Adresse der virtuellen Netzwerkkarte im Gastsystem gültig. Bei der Voreinstellung *"I copied it"* erzeugt VMware eine neue MAC-Adresse für das Gastsystem, was dazu führt, dass dem Entwicklungssystem eine neue IP-Adresse zugewiesen wird. Das Linux-Image ist so konfiguriert, dass es sich via DHCP-Client dynamisch eine IP-Adresse anfordert.

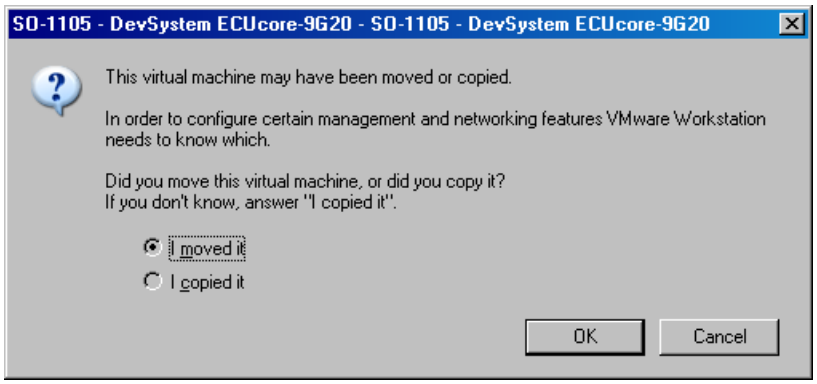


Bild 26: VMware Auswahldialog zum Neuerstellen oder Beibehalten der MAC-Adresse

Bild 27 zeigt den Desktop des Linux-Entwicklungssystems nach dem Starten des Linux-Images im "VMware Player".

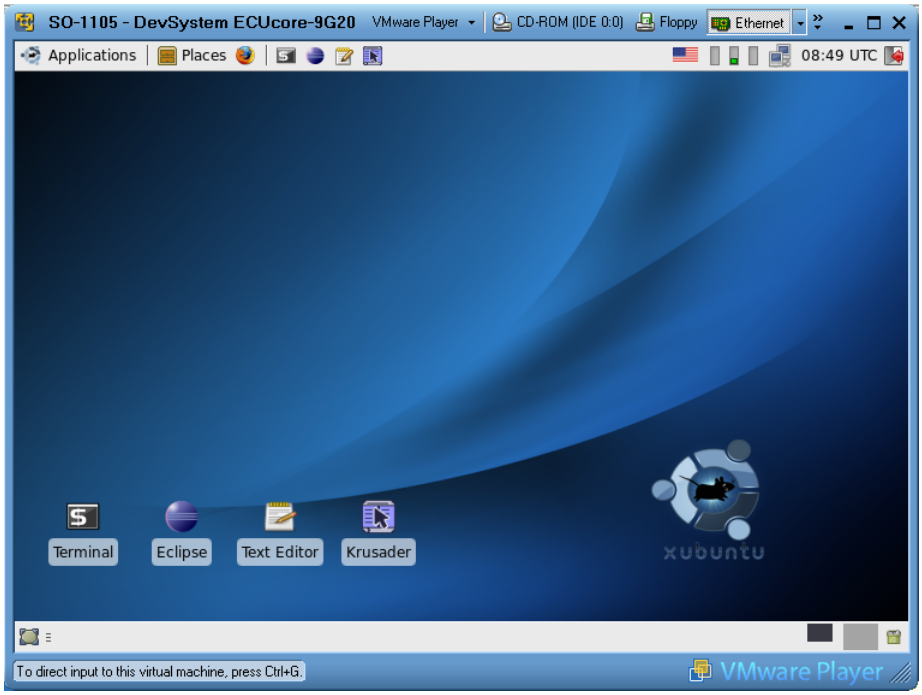


Bild 27: Desktop des Linux-Entwicklungssystems

6.4 Benutzerkonten zur Anmeldung am Linux-Entwicklungssystem

Tabelle 9 listet die zur Anmeldung am Linux-Entwicklungssystem vordefinierten Benutzerkonten auf.

Tabelle 9: Vordefinierte Benutzerkonten des Linux-Entwicklungssystems

Anmeldung	Benutzerdaten	Bemerkung
lokale Konsole / Terminal (normale Nutzerrechte)	User: vmware Passwort: vmware	vordefiniertes Benutzerkonto innerhalb des Linux-Entwicklungssystems
Administratorrechte	Kommando: sudo Passwort: vmware	das verwendete Linux-Entwicklungssystem unterstützt keine explizite Anmeldung als "root", bei Bedarf ist dem jeweils mit Administrator-rechten auszuführenden Befehl das Linux-Kommando *"sudo"* voranzustellen, z.B. *"sudo cat /etc/shadow"*
Windows-Netzwerkumgebung	Gruppe: Workgroup Computer: Vm-xubuntu User: vmware Passwort: vmware	vordefiniertes Benutzerkonto für Zugriff auf das Linux-Entwicklungssystem über die Windows-Netzwerkumgebung (Samba)
Telnet-Zugang	User: vmware Passwort: vmware	vordefiniertes Benutzerkonto zur Anmeldung an das Linux-Entwicklungssystem über einen Telnet-Client (z.B. Telnet-Client unter Windows)

6.5 IP-Adresse des Linux-Entwicklungssystems ermitteln

Um die dem Linux-Entwicklungssystem per DHCP zugewiesene IP-Adresse zu ermitteln, ist zunächst innerhalb von Linux ein Konsolenfenster zu starten (Symbol "Terminal"). Nach Eingabe des Befehls *"ifconfig"* wird dann unter anderem die IP-Adresse des Linux-Images angezeigt (im Screenshot Bild 28 farblich markiert).

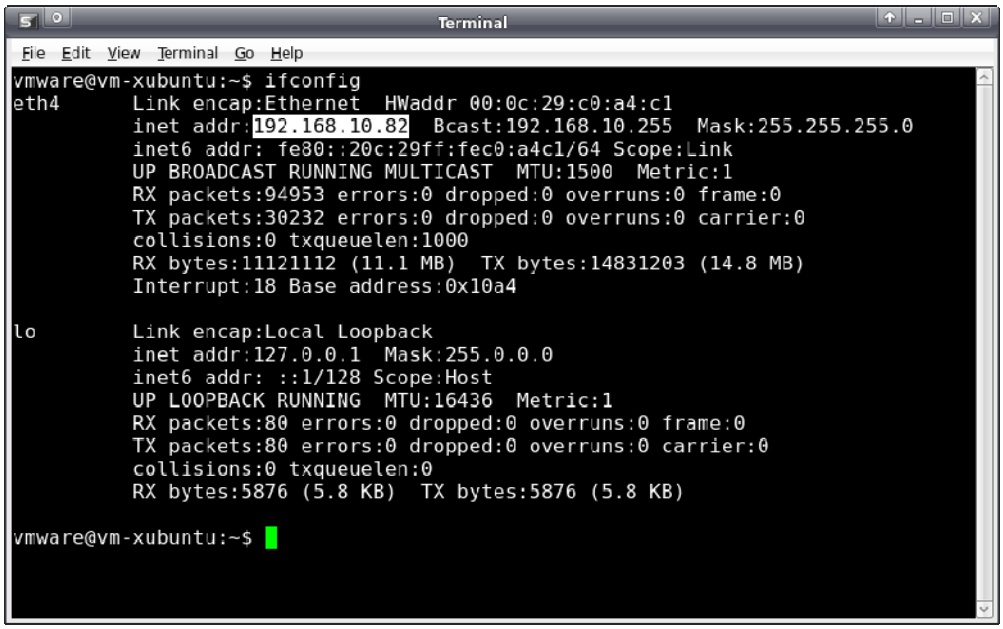


Bild 28: IP-Adresse des Linux-Entwicklungssystems ermitteln

6.6 Zugriff auf das Linux-Entwicklungssystem von einem Windows-PC

6.6.1 Zugriff über die Windows-Netzwerkumgebung

Über den im VMware-Image installierten Samba-Server ist der Zugriff auf Dateien im Linux-Entwicklungssystem über die Windows-Netzwerkumgebung möglich. Das erlaubt ein komfortables Erstellen und Bearbeiten von Quelltexten mit Hilfe eines beliebigen Windows-Editors, wie beispielsweise "Visual Studio". Das Dateisystem des Linux-Entwicklungssystem in der Windows-Netzwerkumgebung ist im Zweig **"Workgroup"** unter dem Rechnernamen **"Vm-xubuntu"** zugänglich (siehe auch Tabelle 9 im Abschnitt 6.4).

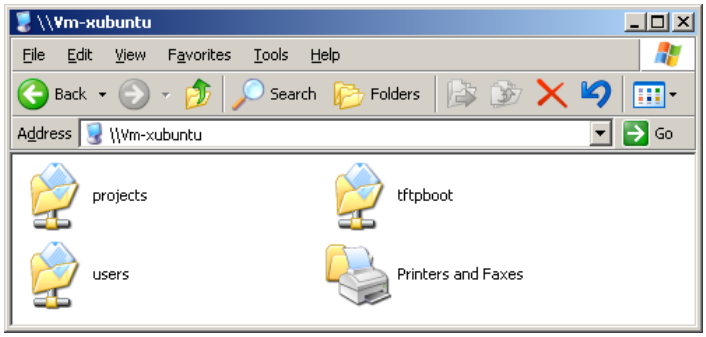


Bild 29: Linux-Entwicklungssystem in der Windows-Netzwerkumgebung

Nach einem Doppelklick auf das Symbol *"users"* (siehe Bild 29) ist die Anmeldung als **Nutzer "vmware"** mit dem **Passwort "vmware"** möglich (siehe Bild 30). Abschließend besteht Zugriff auf den Pfad "\\Vm-xubuntu\users\".

Bild 30: Anmeldung an "Vm-xubuntu"

Alternativ (z.B. bei Problemen wegen langer Timeouts beim Durchsuchen der Windows-Netzwerkumgebung oder generell beim Auffinden des virtuellen Rechners) kann das VMware-Image des Linux-Entwicklungssystem auch direkt über den Windows-Befehl *"net use"* an einen Laufwerksbuchstaben gebunden werden. Dies kann entweder über den symbolischen Namen oder auch direkt über die IP-Adresse des Linux-Entwicklungssystems erfolgen. Für letzteren Fall ist zunächst wie ist im Abschnitt 6.5 beschrieben die IP-Adresse des Linux-Entwicklungssystems zu ermitteln ist. Der Befehl *"net use"* hat folgende Syntax:

```
net use  <local_device>  <\\computername\sharename>  /user:<username>  [options]
```

Um beispielsweise das VMware-Image mit dem Linux-Entwicklungssystem an den lokalen Laufwerksbuchstaben *"X:"* zu binden, ist der Befehl *"net use"* wie folgt anzuwenden:

```
net use x: \\Vm-xubuntu\users  /user:vmware  /persistent:NO
```

Alternativ kann statt des symbolischen Namens auch direkt die IP-Adresse des Linux-Entwicklungssystems angegeben werden, z.B.:

```
net use x: \\192.168.10.82\users  /user:vmware  /persistent:NO
```

6.6.2 Zugriff über Telnet-Client

Über den im VMware-Image installierten Telnet-Server ist der Zugriff auf eine Konsole des Linux-Entwicklungssystem auch über einen entsprechenden Telnet-Client unter Windows möglich. Das erlaubt den Aufruf Kommandozeilen-Programmen wie beispielsweise *"make"* zum Übersetzen von Anwenderprojekten über eine Windows-Oberfläche.

Der Zugriff mittels Telnet-Client erfolgt direkt über die IP-Adresse des Linux-Entwicklungssystems. Abschnitt 6.5 beschriebt, wie die IP-Adresse des Linux-Entwicklungssystems zu ermitteln ist. Um sich über den in Windows standardmäßig enthaltenen Telnet-Client am Linux-Entwicklungssystem anzumelden, ist der Befehl "*telnet*" unter Angabe der IP-Adresse aufzurufen. Das Vorgehen hierzu ist analog zur Anmeldung an der Kommando-Shell des ECUcore-9G20 (siehe dazu Bild 10 im Abschnitt 5.5.1), z.B.

```
telnet  192.168.10.82
```

Innerhalb des Telnet-Fensters ist die Anmeldung als **Nutzer "vmware"** mit dem **Passwort "vmware"** möglich (siehe auch Tabelle 9 im Abschnitt 6.4):

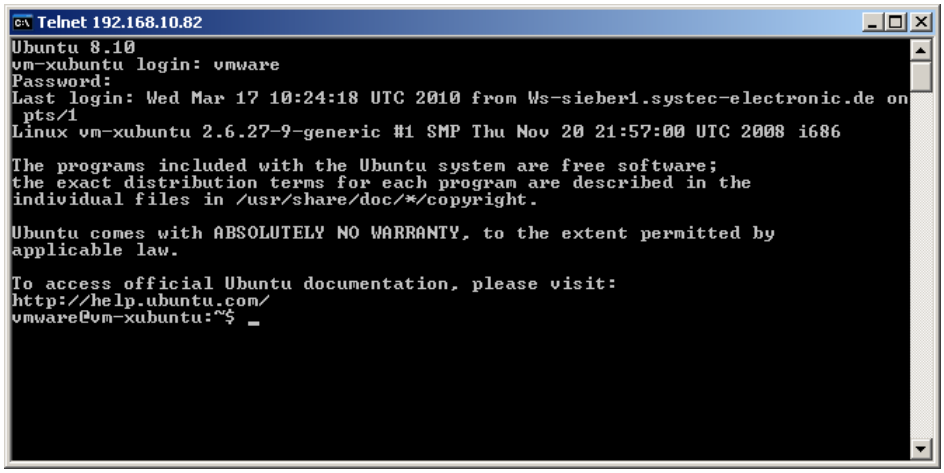


Bild 31: Zugriff über Telnet-Client auf das Linux-Entwicklungssystem

Bild 31 verdeutlicht am Beispiel die Anmeldung am Linux-Entwicklungssystem mit Hilfe des in Windows standardmäßig enthaltenen Telnet-Clients.

6.7 Persönliche Konfiguration und Aktualisierung des Linux-VMware-Images

6.7.1 Anpassung von Tastaturlayout und Zeitzone

Standardmäßig ist das Linux-VMware-Image auf US-Tastaturlayout und UTC-Zeitzone eingestellt. Über das Ländersymbol in der Taskleiste ist der bequeme Wechsel zu einem alternativen, bereits vorinstallierten Tastaturlayout möglich (siehe Bild 32).

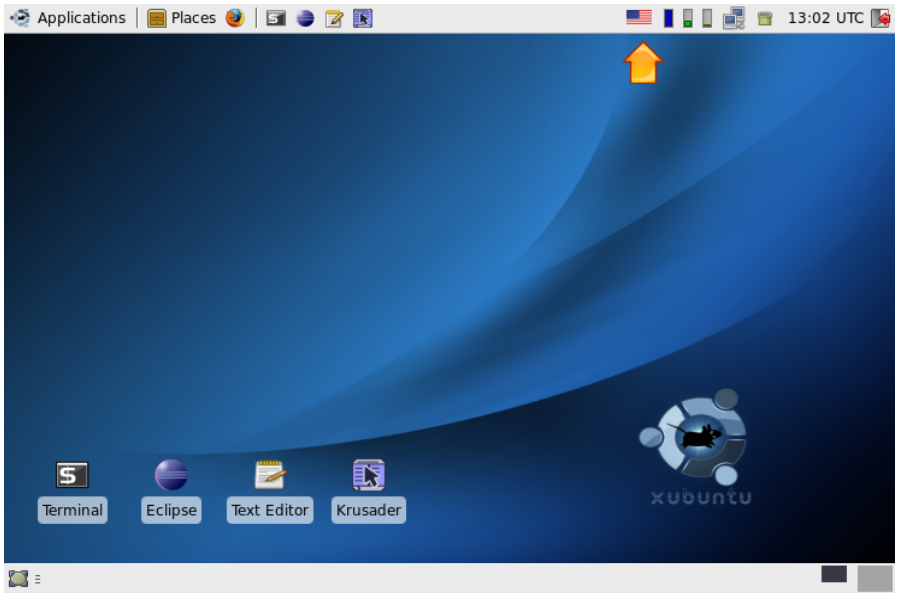


Bild 32: Ländersymbol für Wechsel des Tastaturlayouts

Um ein alternatives **Tastaturlayout permanent auszuwählen**, ist das Ländersymbol in der Taskleiste (siehe Bild 32) mit der **rechten Maustaste** anzuklicken und aus dem Popup-Menü der Eintrag *"Properties"* aufzurufen. Dabei öffnet sich der Dialog *"Configure Keyboard Layout Switcher"*, hier kann das gewünschte Layout als *"Default Layout"* festgelegt werden (siehe Bild 33).

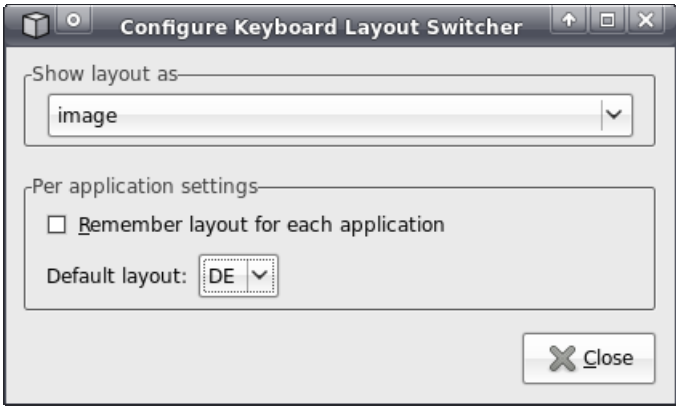


Bild 33: Tastaturlayout permanent auszuwählen

Das **Hinzufügen weiterer Tastaturlayouts** erfolgt mit Hilfe des *"Xfce Settings Manager"*, der direkt über das Startmenü aufrufbar ist: *"Applications -> Settings -> Settings Manager"* (siehe Bild 34).

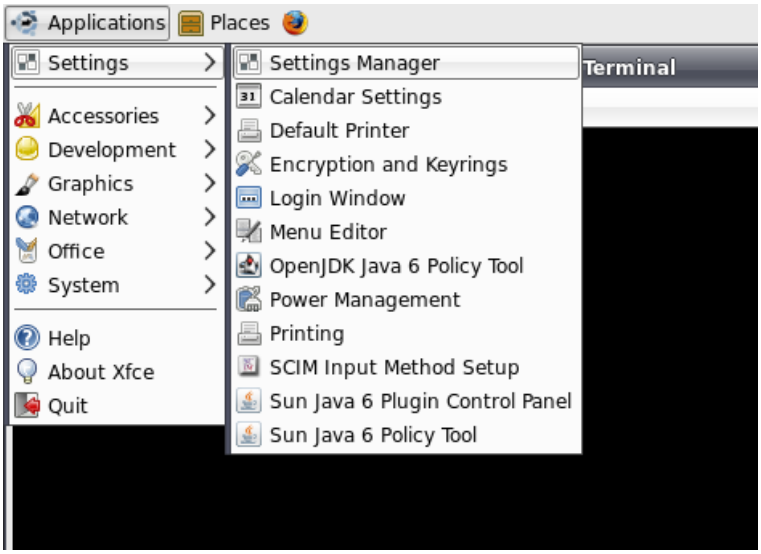


Bild 34: Aufruf des "Xfce Settings Manager" über das Startmenü

Innerhalb des *"Xfce Settings Manager"* können über die Option *"Keyboard"* und deren Unterpunkt *"Layouts"* weitere Tastaturlayouts hinzugefügt bzw. auch wieder entfernt werden (siehe Bild 35).

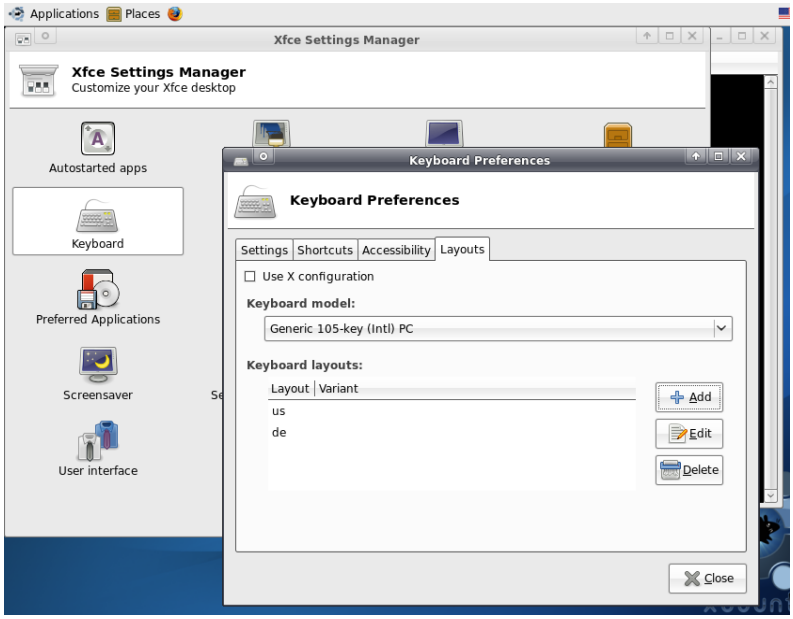


Bild 35: Hinzufügen von Tastaturlayouts im "Xfce Settings Manager“

Die **Einstellung der Zeitzone** erfolgt über ein Control-Pannel der Systemkonfiguration, das ebenfalls direkt über das Startmenü aufrufbar ist: *"Applications -> System -> Time and Date"* (siehe Bild 36). Da das Ändern der Zeitzone eine administrative Tätigkeit ist, muss der Dialog zunächst durch Betätigen der Schaltfläche *"Unlock"* freigegeben werden (ebenfalls siehe Bild 36). Dazu wird analog zum Konsolen-Kommando *"sudo“* das Administratorpasswort abgefragt. Standardmäßig ist hierfür als **Passwort "vmware"** einzugeben (siehe auch Tabelle 9 im Abschnitt 6.4). Anschließend kann über den Punkt *"Time zone“* die Hauptstadt und damit die jeweilige Zeitzone ausgewählt werden.

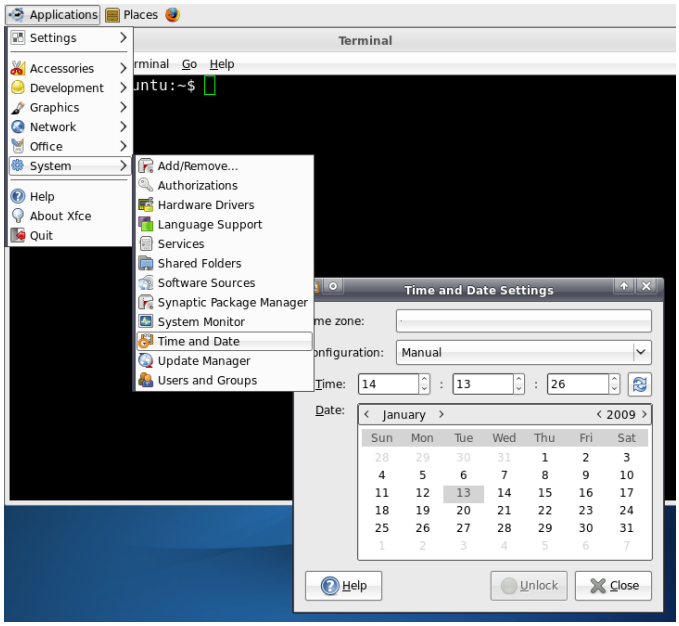


Bild 36: Anpassen der Zeitzone

6.7.2 Anpassen der Desktopgröße

Das Linux-VMware-Image ist in der Lage, die Desktopgröße automatisch an die Fenstergröße des VMware-Players anzupassen. Da sich der VMware-Player wie eine normale Windows-Applikation

verhält, muss also nur der Fensterrahmen des VMware-Players mit der Maus auf die gewünschte Größe gezogen werden, um die Desktopgröße des Linux-VMware-Images zu ändern. Die maximal nutzbare Fenstergröße ist in der Konfigurationsdatei ***"SO-1105\Xubuntu-ECUcore9G20.vmx"*** des VMware-Players definiert und kann hier bei Bedarf verändert werden:

```
####### display ######################
...
svga.maxWidth = "1024"
svga.maxHeight = "768"
...
```

6.7.3 Festlegen einer statischen IP-Adresse für das Linux-VMware-Image

Im Linux-VMware-Image ist standardmäßig die dynamische Konfiguration der IP-Adresse über DHCP aktiviert. Damit kann das Linux-VMware-Image in den meisten Netzwerkumgebungen ad-hoc benutzt werden, ohne dass zuvor eine manuelle Einstellung von Parametern notwendig ist. In Netzwerken, in denen kein DHCP-Server vorhanden ist, muss dagegen vom Anwender eine statische IP-Adresse für das Linux-VMware-Image festgelegt werden. Andernfalls ist keine Ethernet-basierte Kommunikation mit dem ECUcore-9G20 möglich.

Um eine statische IP-Adresse für das Linux-VMware-Image festzulegen, ist das Symbol des *"Network Manager"* in der Taskleiste (siehe Bild 37) mit der **rechten Maustaste** anzuklicken und aus dem Popup-Menü der Eintrag *"Edit Connections"* aufzurufen. Dabei öffnet sich der Dialog *"Network Connections"* (siehe Bild 38).

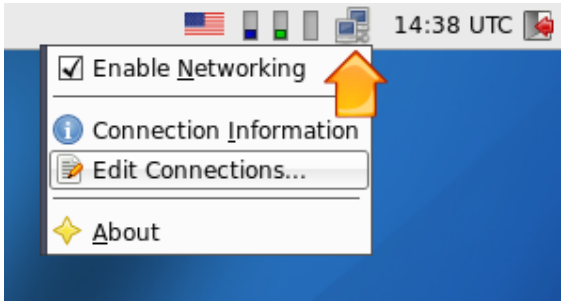


Bild 37: "Network Manager" zur Konfiguration der Ethernet-Schnittstelle

Im Dialog *"Network Connections"* (siehe Bild 38) kann im Tabsheet *"Wired"* mit Hilfe der Schaltfläche *"Add"* eine neue Netzwerkverbindung angelegt werden. Dabei öffnet sich der Dialog *"Edit Wired Connection"* (siehe Bild 39).

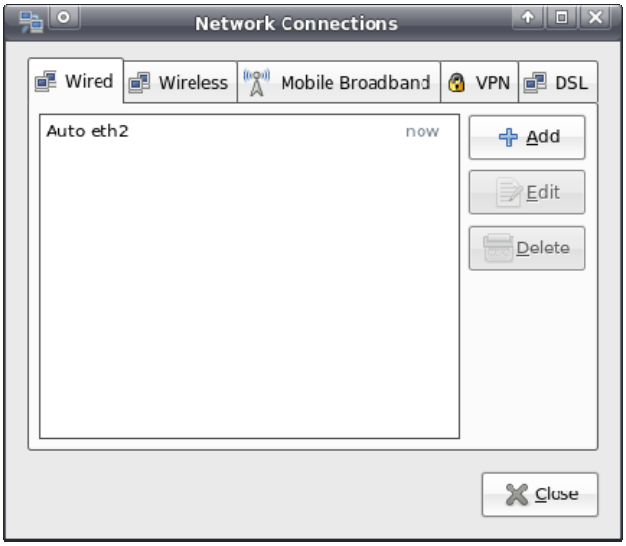


Bild 38: Hinzufügen einer Netzwerkverbindung

Im Dialog *"Edit Wired Connection"* (siehe Bild 39) ist zunächst im Auswahlfeld *"Method"* die Voreinstellung *"Automatic (DHCP)"* auf die alternative Option *"Manual"* zu ändern. Anschließend können im Tabsheet "IPv4 Settings" die Einstellungen für IP-Adresse, Netzwerkmaske, Gateway, DNS-Server usw. vorgenommen werden.

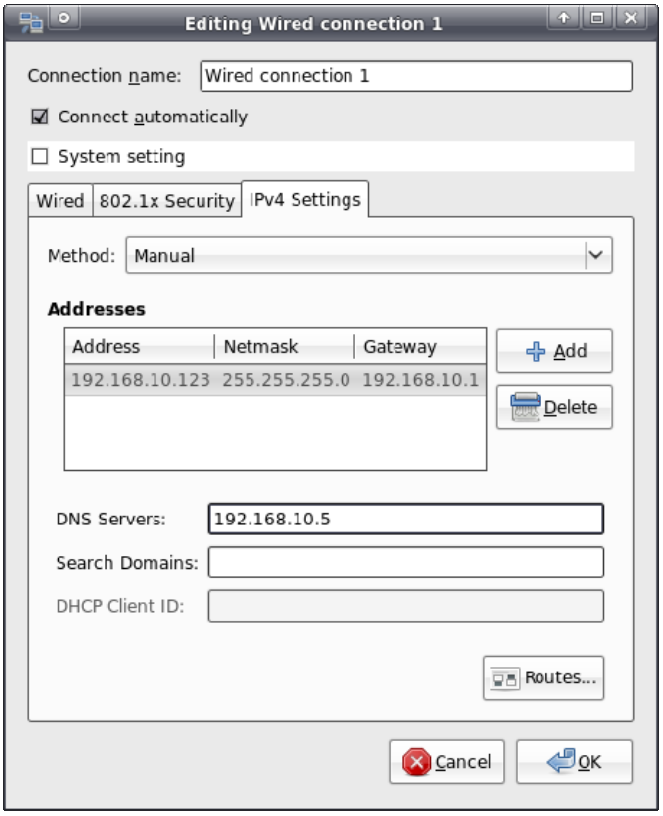


Bild 39: Konfiguration der Netzwerkverbindung

Nach Abschluss der Konfiguration und Schließen aller Dialoge ist wiederum das Symbol des *"Network Manager"* in der Taskleiste anzuklicken, jedoch diesmal mit der **linken Maustaste** (siehe Bild 40). Hier ist die neu angelegte Netzwerkverbindung mit der statischen IP-Adresse entsprechend als aktive Verbindung auszuwählen.

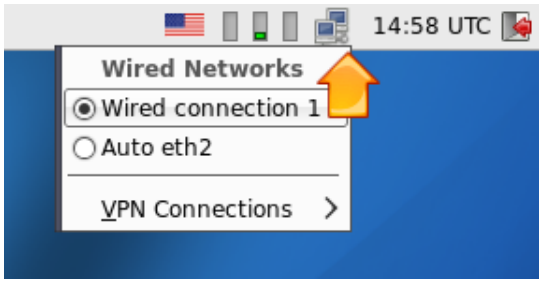


Bild 40: Wechsel der Netzwerkkonfiguration im "Network Manager"

6.7.4 Systemaktualisierung des Linux-VMware-Images

Im Linux-VMware-Image ist die automatische Aktualisierung der Paketlisten **deaktiviert**. Damit ist sichergestellt, dass ein definierter Systemstand bestehen bleibt. Es erfolgt jedoch auch keine Information an den Benutzer, falls aktualisierte Pakete verfügbar sind, die eventuelle Sicherheitslücken schließen.

Es ist dennoch möglich, die automatische Aktualisierung der Paketlisten wieder zu aktivieren bzw. das Aktualisieren manuell zu veranlassen. Mögliche Programme dafür sind *"Synaptic Package Manager"* (aus dem Menü System) und das Konsolenprogramm *"aptitude"*. Nach der Aktualisierung der Paketlisten ist es mit diesen Programmen auch möglich, neue Paketversionen zu installieren.

Es wird jedoch ausdrücklich darauf hingewiesen, dass nicht gewährleistet werden kann, dass nach einem Update das Linux-Entwicklungssystem für das ECUcore-9G20 weiterhin vollständig funktionsfähig bleibt. Es wird daher dingend empfohlen, vor einem Update unbedingt eine Backup-Kopie des Linux-Entwicklungssystems anzufertigen.

6.7.5 Ändern des Computer-Namens in der Windows-Netzwerkumgebung

Das Linux-VMware-Image benutzt standardmäßig den Computer-Namen *"Vm-xubuntu"* in der Windows-Netzwerkumgebung (siehe auch Tabelle 9 im Abschnitt 6.4). Da der Zugriff auf einen Computer im Windows-Netzwerk über dessen Namen gesteuert wird, sind beim parallelen Einsatz des Linux-VMware-Images auf mehreren Rechnern die Computer-Namen eindeutig anzupassen. Damit wird eine Mehrfachverwendung desselben Namens verhindert, was andernfalls zu Kollisionen und Zugriffsfehlern führen würde.

Der Computer-Namen wird über die Datei *"/etc/hostname"* definiert. Um diesen zu modifizieren, ist der Befehl ***"sudo gedit /etc/hostname"*** einzugeben. Der geänderte Name wird dann nach einem Neustart übernommen. Eine sofortige Änderung des Namens bewirkt das Kommando *"hostname"*. Dazu wird der neu zu setzende Computer-Name als Parameter angegeben, z.B. ***"sudo hostname vm-xubuntu-2"***. Die mit diesem Befehl vorgenommene Änderung wirkt jedoch im Gegensatz zur Modifikation der Datei *"/etc/hostname"* nur temporär bis zum nächsten Neustart.

6.7.6 Schrumpfen des VMware-Images

Um das VMware-Image auf die notwendige Minimalgröße zu schrumpfen, ist die VMware-Toolbox durch Eingabe des Befehls ***"sudo vmware-toolbox"*** aufzurufen. Über das Tabsheet *"Shrink"* kann das VMware-Image auf seine notwendige Minimalgröße geschrumpft werden.

7 Softwareentwicklung für das ECUcore-9G20

7.1 Softwarestruktur für das ECUcore-9G20

Im VMware-Image des Linux-Entwicklungssystems sind sämtliche Komponenten, die zur Entwicklung von Software für das ECUcore-9G20 benötigt werden, im Pfad *"/projects"* abgelegt (respektive *"\\Vm-xubuntu\users\projects"* in der Windows-Netzwerkumgebung). Bild 41 veranschaulicht die Verzeichnisstruktur.

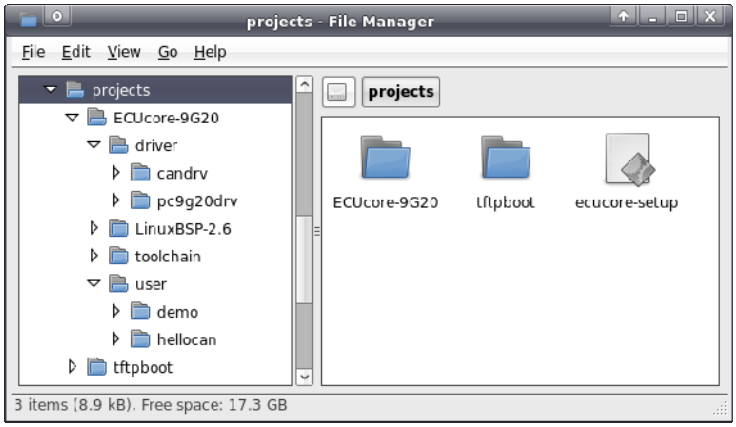


Bild 41: Struktur des Verzeichnisses "/projects" im Linux-Entwicklungssystem

```
/projects
   |
   +- ECUcore-9G20
   |     |
   |     +- driver                  Treiber-Bibliotheken für das ECUcore-9G20
   |     |     |
   |     |     +- pc9g20drv         I/O-Treiber zur Einbindung in eigene Anwenderprojekte
   |     |     |                    (incl. Sourcecode und Testapplikation)
   |     |     |
   |     |     +- candrv            CAN-Treiber zur Einbindung in eigene Anwenderprojekte
   |     |
   |     +- LinuxBSP-2.6            Sourcecode des Linux-Images für das ECUcore-9G20
   |     |     |                    (Linux-Kernel und User-Programme)
   |     |     |
   |     |     +- u-boot            Sourcecode des Bootloaders "U-Boot"
   |     |     |
   |     |     +- linux-2.6         Sourcecode des Linux-Kernels für das ECUcore-9G20
   |     |     |
   |     |     +- ptxdist           User-Programme und Buildsystem für das ECUcore-9G20
   |     |     |
   |     |     ...                  sonstige Komponenten des Linux-Images
   |     |
   |     +- toolchain               GCC-Toolchain für das ECUcore-9G20
   |     |
   |     +- user                    Pfad zur Ablage eigener Anwenderprojekte
   |           |
   |           +- demo              Referenz- und I/O-Demoprojekt für das ECUcore-9G20
   |           |
   |           +- hellocan          Referenz- und CAN-Demoprojekt für das ECUcore-9G20
   |
   +- tftpboot                     NFS-Verzeichnis zur Einbindung durch das ECUcore-9G20
   |
   +- ecucore-setup                Shell-Skript zum Setzen von Umgebungsparametern
```

Im Pfad ***"/projects/ECUcore-9G20/LinuxBSP-2.6"*** befinden sich die gesamten Linux-Sourcen für das ECUcore-9G20.

Das Verzeichnis ***"/projects/ECUcore-9G20/driver/pc9g20drv"*** beinhaltet sowohl den Sourcecode des I/O-Treibers für das ECUcore-9G20 (incl. Testapplikation), als auch **Headerfiles und fertig generierte Bibliotheken des I/O-Treibers** zur Einbindung in eigene Anwenderprojekte (siehe Abschnitt 7.3.1).

Das Verzeichnis ***"/projects/ECUcore-9G20/driver/candrv"*** beinhaltet **Headerfiles und fertig generierte Bibliotheken des CAN-Treibers** zur Einbindung in eigene Anwenderprojekte (siehe Abschnitt 7.4.1).

Im Verzeichnis ***"/projects/ECUcore-9G20/user/demo"*** ist ein Demoprogramm enthalten, das zum einen den Zugriff auf die Ein- und Ausgänge des ECUcore-9G20 beschreibt (Details siehe Abschnitt 7.3.1) und zum anderen als Template für eigene Projekte verwendet werden sollte. Alle weiteren Beschreibungen zur Softwareentwicklung beziehen sich im Folgenden auf diese Demoprojekt (speziell im Abschnitt 7.6).

Im Verzeichnis ***"/projects/ECUcore-9G20/user/hellocan"*** ist ein Demoprogramm enthalten, das zum einen den Zugriff auf das CAN-Interface des ECUcore-9G20 beschreibt und zum anderen als Template für eigene Projekte verwendet werden sollte.

Der Pfad ***"/projects/tftpboot"*** ist das Wurzelverzeichnis für die Einbindung des Linux-Entwicklungssystems in das lokale Filesystem des ECUcore-9G20 per NFS (siehe Abschnitt 7.5.1).

Das Shell-Skript ***"/projects/ecucore-setup"*** dient zum Setzen der erforderlichen Umgebungsparameter, die zur Ausführung des Build-Systems benötigt werden (siehe Abschnitt 7.2). Dieses Shell-Skript wird automatisch beim Öffnen einer Konsole ("Terminal") über das File *".bashrc"* ausgeführt, ebenso beim Starten der grafischen IDE "Eclipse" über das entsprechende Desktop-Symbol.

7.2 Makefile und Umgebungsvariablen zum Erstellen von Projekten

Die Erstellung von Anwenderprogrammen für das ECUcore-9G20 erfordert die Verwendung der GNU-Crosscompiler Toolchain für ARM9-Prozessoren. Diese ist im VMware-Image des Linux-Entwicklungssystems bereits vollständig installiert und konfiguriert. Von zentraler Bedeutung sind dabei die im Shell-Skript *"/projects/ecucore-setup"* definierten Umgebungsvariablen:

```
ARM9G20_BASE_PATH=/projects/ECUcore-9G20
ARM9G20_LINUX_BSP_PATH=$ARM9G20_BASE_PATH/LinuxBSP-2.6
ARM9G20_LINUX_KDIR_PATH=$ARM9G20_LINUX_BSP_PATH/linux-2.6
ARM9G20_CC_PATH=$ARM9G20_BASE_PATH/toolchain/arm-2009q3/bin
ARM9G20_CC_PREFIX=$ARM9G20_CC_PATH/arm-none-linux-gnueabi-
ARM9G20_CFLAGS=
ARM9G20_GDB_PATH=$ARM9G20_CC_PATH

export ARM9G20_BASE_PATH
export ARM9G20_LINUX_BSP_PATH
export ARM9G20_LINUX_KDIR_PATH
export ARM9G20_CC_PATH
export ARM9G20_CC_PREFIX
export ARM9G20_CFLAGS
export ARM9G20_GDB_PATH


# add toolchain to PATH
export PATH=$ARM9G20_CC_PATH:$PATH

# add ptxdist to PATH
export PATH=$ARM9G20_LINUX_BSP_PATH/ptxdist/bin:$PATH
```

Als Ausgangspunkt für die Entwicklung eigener Anwenderprogramme sollte das im Pfad *"/projects/ECUcore-9G20/demo"* enthaltene Template verwendet werden (bzw. *"\\Vm-xubuntu\projects\ECUcore-9G20\demo\"* in der Windows-Netzwerkumgebung). Von besonderer Bedeutung sind hierbei die im Makefile (*demo/source/Makefile*) bereits definierten Variablen sowie die Verweise auf die GNU-Crosscompiler Toolchain für ARM9-Prozessoren, basierend auf den Definitionen in *"/projects/ecucore-setup"*:

```
CDEFS         = $(ARM9G20_CFLAGS)
CFLAGS        += -O0 -g -Wall -march=armv5te -mcpu=arm926ej-s -mtune=arm9tdmi \
                 msoft-float $(INCLUDES) $(CDEFS)
LDFLAGS       += -march=armv5te -mcpu=arm926ej-s -mtune=arm9tdmi -msoft-float

CROSS         = $(ARM9G20_CC_PREFIX)
CC            = $(CROSS)gcc
STRIP         = $(CROSS)strip
AR            = $(CROSS)ar
```

Innerhalb des Makefiles erfolgt der Aufruf der einzelnen Tools über entsprechende Makros wie beispielsweise *"$(CC)"*:

```
$(CC)  $(CFLAGS)  -o $(EXEC)  demo.c
```

Ebenfalls im Makefile bereits vorbereitet ist das Umkopieren des erstellten Executeables in das Verzeichnis *"/tftpboot"* bzw. eines der darin enthaltenen Unterverzeichnisse. Hierdurch kann das ausführbare Programm später ohne weitere Zwischenschritte direkt auf dem ECUcore-9G20 gestartet werden (siehe dazu auch Abschnitt 7.5.1).

7.3 I/O-Treiber für das ECUcore-9G20

7.3.1 Einbindung des I/O-Treibers in eigene Anwenderprojekte

Das Verzeichnis ***"/projects/ECUcore-9G20/driver/pc9g20drv"*** des Linux-Entwicklungssystems beinhaltet sowohl den Sourcecode des I/O-Treibers für das ECUcore-9G20 (incl. Testapplikation), als auch Headerfiles und bereits fertig generierte Bibliotheken des Treibers zur Einbindung in eigene Anwenderprojekte.

Bild 42 verdeutlicht die Struktur des I/O-Treibers für das ECUcore9G20. Der Treiber untergliedert sich in ein Kernelspace-Modul (*pc9G20drv.ko*), das die Zugriffe auf die Hardware realisiert (FPGA und Portpins), sowie in eine Userspace-Bibliothek (*pc9g20drv.a* als statische Bibliothek bzw. *pc9g20drv.so* als dynamisch ladbare Bibliothek). Beide Komponenten, sowohl Kernelspace-Modul als auch Userspace-Bibliothek (statisch oder dynamisch) werden zur Ausführung von I/O-Zugriffen benötigt.

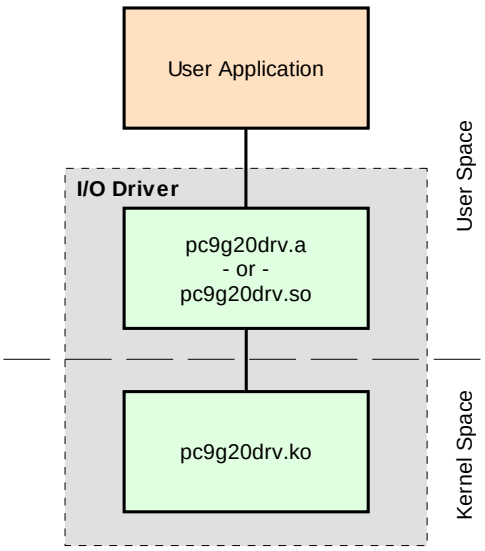


Bild 42: Struktur des I/O-Treibers für das ECUcore-9G20

Aus dem Projektverzeichnis des I/O-Treibers sind folgende Dateien zur Einbindung in eigene Anwenderprojekte relevant:

pc9g20drv.h: */projects/ECUcore-9G20/driver/pc9g20drv/include/pc9g20drv.h*
Headerfile zur Beschreibung des Treiber-Interfaces
Dieses Headerfile ist mittels *#include* in den Sourcecode (**.c*) des Anwenderprojektes einzubinden.

pc9g20drv.a: */projects/ECUcore-9g20/driver/pc9g20drv/lib/pc9g20drv.a*
Userspace-Bibliothek des I/O-Treibers zum statischen Linken an die Anwenderapplikation (vergleichbar mit den statischen Standardbibliotheken der C-Entwicklungsumgebung)
Beim statischen Linken wird die Userspace-Bibliothek fest an die Anwenderapplikation gebunden und ist dadurch untrennbar mit dieser verbunden. Diese Variante hat den Vorteil, dass beim Starten der Applikation auf dem ECUcore-9G20 keine expliziten Librarypfade angegeben werden müssen.

pc9g20drv.so: */projects/ECUcore-9G20/driver/pc9g20drv/lib/pc9g20drv.so*
Userspace-Bibliothek des I/O-Treibers zum dynamischen Laden durch die Anwenderapplikation zur Laufzeit (vergleichbar mit der Benutzung einer DLL unter Windows)
Diese Version der Userspace-Bibliothek wird erst zur Laufzeit in den Adressraum der Anwenderapplikation geladen. Dies hat den Vorteil, dass Treiberbibliothek und Anwenderapplikation unabhängig voneinander austauschbar sind und dass verschiedene Anwenderapplikationen dieselbe Version der Treiberbibliothek gemeinsam benutzen können. Auf dem ECUcore-9G20 ist hierfür die Umgebungsvariable *"LD_LIBRARY_PATH="* entsprechend zu setzen, z.B.:

```
export  LD_LIBRARY_PATH=.
```

pc9g20drv.ko: */projects/ECUcore-9G20/driver/pc9g20drv/lib/pc9g20drv.ko*
Kernelspace-Modul des I/O-Treibers für Zugriffe auf Hardware (PLD und Portpins)
Dieses Modul ist vor dem Starten der Anwenderapplikation mit Hilfe des Linux-Kommandos *"insmod"* zu laden:

```
insmod  pc9g20drv.ko
```

Im Beispiel des im Abschnitt 7.6 vorgestellten Demoprogramms wird die Userspace-Bibliothek des I/O-Treibers fest an die Anwenderapplikation gebunden. Dazu ist die statische Bibliothek *pc9g20drv.a*

beim Aufruf des GCC in die Liste der zu linkenden Objekte aufzunehmen. Im Makefile des Demoprojektes wird dazu die Variable *"LIBS="* entsprechend gesetzt, die dann wiederum an den Linker (via GCC-Aufruf) übergeben wird:

```
LIBS = ../lib/pc9g20drv.a

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

Die vom I/O-Treiber zur Verfügung gestellten Funktionen sind im Headerfile *"pc9g20drv.h"* aufgelistet, ihre Anwendung dokumentiert das Demoprogramm unter *"/projects/ECUcore-9G20/user/demo"* (siehe auch Abschnitt 7.6).

Im Auslieferungszustand des ECUcore-9G20 ist bereits ein fertig generiertes und sofort per *"insmod"* ladbares Kernelmodul des I/O-Treibers im Verzeichnis *"/home/bin"* abgelegt (siehe Abschnitt 5.13):

```
insmod  /home/bin/pc9g20drv.ko
```

Ein Beispiel für die praktische Anwendung des I/O-Treibers auf dem ECUcore-9G20 zeigt das Demoprojekt unter *"/projects/ECUcore-9G20/user/demo"* (siehe Abschnitte 7.3.2 und 7.6).

7.3.2 I/O-Treiber Demoprojekt

Das Demoprojekt für den I/O-Treiber ist im Verzeichnis ***"/projects/ECUcore-9G20/user/demo"*** des Linux-Entwicklungssystems abgelegt. Es veranschaulicht den Zugriff auf die Ein- und Ausgänge des Moduls. Dazu bedient sich das Demoprogramm der Hilfe des unter ***"/projects/ECUcore-9G20/driver/pc9g20drv"*** abgelegten I/O-Treibers. Das Demoprojekt wird im Abschnitt 7.6 als Referenzprojekt zur Softwareentwicklung für das ECUcore-9G20 sehr detailliert beschrieben.

Vor dem Start des Demoprogramms ist zunächst der I/O-Treiber mit dem Befehl *"insmod"* explizit zu laden, anschließend kann das Demoprogramm selbst aufgerufen werden:

```
insmod  pc9g20drv.ko
./demo
```

Die Ausführung des Demoprojektes auf dem ECUcore-9G20 verdeutlicht Bild 48 im Abschnitt 7.6.1. Das Demoprojekt kann entweder durch Setzen des Run/Stop-Schalters in die Position "MRes" oder durch Drücken von "Ctrl+C" beendet werden.

7.4 CAN-Treiber für das ECUcore-9G20

7.4.1 Einbindung des CAN-Treibers in eigene Anwenderprojekte

Das Verzeichnis ***"/projects/ECUcore-9G20/driver/candrv"*** des Linux-Entwicklungssystems beinhaltet Headerfiles und bereits fertig generierte Bibliotheken des CAN-Treibers für das ECUcore-9G20 zur Einbindung in eigene Anwenderprojekte.

Die Struktur des CAN-Treibers ist identisch zu der im Abschnitt 7.3.1 beschriebenen Struktur des I/O-Treiber. So untergliedert sich der CAN-Treiber ebenfalls in ein Kernelspace-Modul (*candrv.ko*), das die Zugriffe auf die CAN-Controller realisiert, sowie in eine Userspace-Bibliothek (*candrv.a*). Beide Komponenten, sowohl Kernelspace-Modul als auch Userspace-Bibliothek werden zum Zugriff auf den CAN-Bus benötigt.

Aus dem Projektverzeichnis des CAN-Treibers sind folgende Dateien zur Einbindung in eigene Anwenderprojekte relevant:

cdrvinc.h: */projects/ECUcore-9G20/driver/candrv/include/cdrvinc.h*
Headerfile zur Beschreibung des Treiber-Interfaces
Dieses Headerfile ist mittels *#include* in den Sourcecode (**.c*) des Anwenderprojektes einzubinden. Dieses Headerfile inkludiert dann alle weiteren Headerfiles im Verzeichnis *"include"* in der richtigen Reihenfolge.

candrv.a: */projects/ECUcore-9G20/driver/candrv/lib/candrv.a*
Userspace-Bibliothek des I/O-Treibers zum statischen Linken an die Anwenderapplikation.

candrv.ko: */projects/ECUcore-9G20/driver/candrv/lib/candrv.ko*
Kernelspace-Modul des CAN-Treibers zum Zugriff auf die CAN-Controller
Dieses Modul ist vor dem Starten der Anwenderapplikation mit Hilfe des Linux-Kommandos *"insmod"* zu laden:

```
insmod  candrv.ko
```

Das Interface sowie die Nutzung des CAN-Treibers in eigenen Applikationen dokumentiert das CAN Treiber Software Manual (Manual-Nr.: L-1023). Ein Beispiel für die praktische Anwendung des CAN-Treibers auf dem ECUcore-9G20 zeigt das Demoprojekt unter *"/projects/ECUcore-9G20/user/hellocan"* (siehe Abschnitt 7.4.2).

Im Auslieferungszustand des ECUcore-9G20 ist bereits ein fertig generiertes und sofort per *"insmod"* ladbares Kernelmodul des CAN-Treibers im Verzeichnis *"/home/bin"* abgelegt (siehe Abschnitt 5.13):

```
insmod  /home/bin/candrv.ko
```

7.4.2 CAN-Treiber Demoprojekt

Das Demoprojekt für den CAN-Treiber ist im Verzeichnis ***"/projects/ECUcore-9G20/user/hellocan"*** des Linux-Entwicklungssystems abgelegt. Es veranschaulicht den Zugriff auf die CAN-Schnittstellen des Moduls. Dazu bedient sich das Demoprogramm der Hilfe des unter ***"/projects/ECUcore-9G20/driver/candrv"*** abgelegten CAN-Treibers.

Zur Demonstration des Nachrichtenaustausches über den CAN-Bus wird eine entsprechende Gegenstelle benötigt. Hierfür bieten sich idealerweise das CAN-Analysetool *"CAN-REport"* in Verbindung mit einem USB-CANmodul an. Mit Hilfe des *"CAN-REport"* können beliebige CAN-Nachrichten gesendet und empfangen werden (siehe Bild 44).

Hinweis: USB-CANmodul und CAN-REport sind nicht im Lieferumfang des Development Kit ECUcore-9G20 enthalten. Beide Produkte zusammen sind unter der Bestellnummer SO-1054-U als Bundle erhältlich.

Das Demoprogramm *"hellocan"* benutzt **125kBit/s** als Bitrate. Es legt nach seinem Start zunächst je 10 Kanäle für den Empfang von CAN-Nachrichten sowie 10 Kanäle zum Versenden an:

Empfangsbereich: CAN-Nachrichten im Identifier-Bereich 0x100 - 0x109
Sendebereich: CAN-Nachrichten im Identifier-Bereich 0x200 - 0x209

Nach seiner erfolgreichen Initialisierung sendet das Demoprogramm *"hellocan"* einmalig eine CAN-Nachricht mit dem Identifier 0x400 und dem Dateninhalt *"68 61 6C 6C 6f 63 61 6E"* (ASCII: *"hellocan"*) auf den Bus. Anschließend geht es in seine Hauptschleife über, in der es auf dem Empfang von CAN-Nachrichten im spezifizierten Identifier-Bereich 0x100 - 0x109 wartet. Beim Empfang einer entsprechenden Nachricht wird diese mit einem um 0x100 erhöhten Identifier (entspricht Identifier-Bereich 0x200 - 0x209) als Echo wieder zurück gesendet. Bild 43 zeigt die Ausführung des

Demoprojektes *"hellocan"* auf dem ECUcore-9G20. Das Demoprojekt kann durch Drücken von "Ctrl+C" beendet werden.

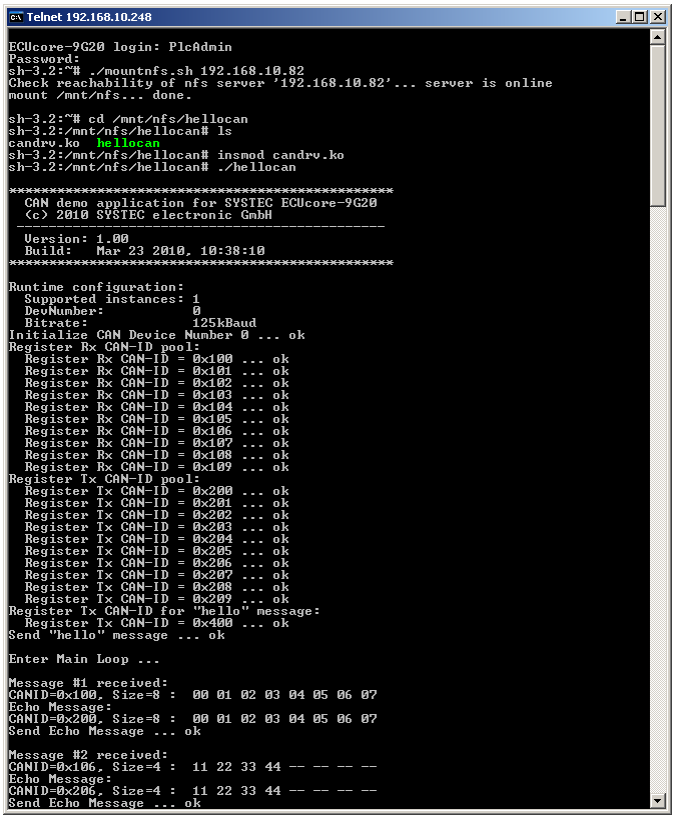


Bild 43: Ausführung des Demoprojektes "hellocan" auf dem ECUcore-9G20

Bild 44 zeigt den Datenaustausch mit dem Demoprogramm *"hellocan"* im CAN-Bus Analysetool *"CAN-REport"*.

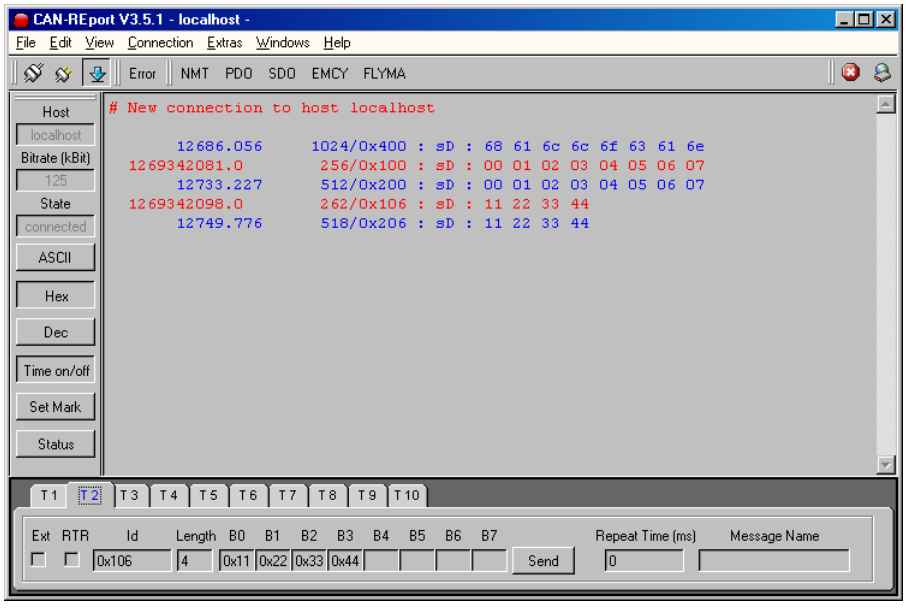


Bild 44: CAN-Analysetool "CAN-REport"

7.5 Übertragen von Programmen auf das ECUcore-9G20

Voraussetzung für das Übertragen - und damit auch Starten - von Programmen auf dem ECUcore-9G20 ist zunächst die Konfiguration des Moduls wie im Abschnitt 5.4 beschrieben. Anschließend ist die Anmeldung an der Kommando-Shell des ECUcore-9G20 entsprechend Abschnitt 5.5.1 notwendig.

Die Übertragung von Programmen auf das ECUcore-9G20, bzw. ganz allgemein der Dateiaustausch zwischen Entwicklungssystem (VMware Linux-Image) und dem ECUcore-9G20, können auf zwei verschiedenen Wegen erfolgen, die beide mit Vor- und Nachteilen behaftet sind:

NFS: Das "Network File System" (NFS) stellt die einfachste Form dar, ein im Linux-Image übersetztes Anwenderprogramm direkt auf dem ECUcore-9G20 zu starten. Dazu wird ein Verzeichnis aus dem VMware-Image des Linux-Entwicklungssystems in das lokale Dateisystem des ECUcore-9G20 eingebunden ("gemountet"). Zum Starten des Programms muss dann auf dem ECUcore-9G20 nur noch der entsprechende Befehlsname eingegeben werden. Die notwendige Dateiübertragung vom Entwicklungssystem auf das ECUcore-9G20 erfolgt implizit durch NFS, ohne dass vom Anwender hierzu weitere Kommandos notwendig sind. Es ist also sichergestellt, dass auf dem ECUcore-9G20 stets die aktuell auf dem Entwicklungssystem vorliegende Programmversion ausgeführt wird und nicht eine veraltete. Damit empfiehlt sich NFS insbesondere während der Softwareentwicklung. Nachteilig an NFS ist zum einen die Tatsache, dass hierüber nur Verbindungen zu anderen Linux-Maschinen möglich sind, nicht aber z.B. zu Windows-PC. Zum anderen lässt NFS nur eine sehr rudimentäre Nutzerverwaltung und damit Zugangskontrolle zu, was beim späteren realen Einsatz der Geräte im Feld oftmals unerwünscht ist. Die Verwendung von NFS beschreibt Abschnitt 7.5.1.

FTP: Das "File Transfer Protokoll" ist als standardisiertes und in der Praxis etabliertes Protokoll plattformunabhängig. Sowohl FTP-Server als auch -Clients sind für die verschiedensten Betriebssysteme verfügbar, darunter Linux und Windows. Mit Hilfe von FTP ist es also im Gegensatz zu NFS auch möglich, Dateien von einem Windows-PC auf das ECUcore-9G20 zu übertragen (z.B. Programmupdate durch Service-Techniker mit Windows-Laptop). Darüber hinaus erlaubt FTP eine detaillierte Zugangskontrolle durch die Authentifizierung mit Username und Passwort. Nachteilig an FTP ist die für jede Dateiübertragung erforderliche Kommandoeingabe, was gerade während der Entwicklungsphase oft als lästig empfunden wird bzw. vergessen werden kann. Auf dem ECUcore-9G20 wird dann unter Umständen unbeabsichtigt und unbemerkt eine veraltete Programmversion ausgeführt. Die Verwendung von FTP beschreibt Abschnitt 7.5.2.

7.5.1 Verwendung von NFS

Die einfachste Form, ein im Linux-Image übersetztes Anwenderprogramm auf dem ECUcore-9G20 zu starten, besteht darin, ein Verzeichnis aus dem VMware-Image des Linux-Entwicklungssystems direkt in das lokale Dateisystem des ECUcore-9G20 einzubinden ("zu mounten"). Dazu exportiert das VMware-Image des Linux-Entwicklungssystems das Verzeichnis *"tftpboot"*, inklusive aller darin enthaltenen Unterverzeichnisse. Um diesen Verzeichnisbaum des Entwicklungssystems in das lokale Dateisystem des ECUcore-9G20 einzubinden, sind folgende Schritte notwendig:

1. Bestimmung der IP-Adresse des Linux-Entwicklungssystems

Die Vorgehensweise zur Bestimmung der IP-Adresse des Linux-Images ist im Abschnitt 6.5 beschrieben.

2. Mounten des Linux-Entwicklungssystems auf dem ECUcore-9G20

Um das Verzeichnis *"/tftpboot"* des Linux-Images im lokalen Dateisystem des ECUcore-9G20 einzubinden, ist der Befehl *"mount"* wie folgt zu verwenden:

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

Um beispielsweise das ECUcore-9G20 an das Linux-Entwicklungssystem mit der im Abschnitt 6.5 ermittelten IP-Adresse anzubinden, ist auf dem ECUcore-9G20 folgender Befehl einzugeben:

```
mount  -t nfs  -o nolock  192.168.10.82:/tftpboot  /mnt/nfs
```

Um die Nutzung des Mount-Befehls etwas zu vereinfachen existiert im Home-Verzeichnis des ECUcore-9G20 (in diesem befindet man sich unmittelbar nach der Anmeldung am Kommandoprompt) das Skript *"mountnfs.sh"*. Diesem Skript ist als Parameter lediglich die IP-Adresse des Linux-Images zu übergeben. Für das oben angeführte Beispiel ergibt sich damit folgender Aufruf:

```
./mountnfs.sh  192.168.10.82
```

Nach der Ausführung des Mount-Befehls ist der gesamte Inhalt des Verzeichnisses *"/tftpboot"* des Linux-Images (incl. evtl. dort enthaltener Unterverzeichnisse) im lokalen Verzeichnis *"/mnt/nfs"* des ECUcore-9G20 zugänglich.

Bild 45 fasst die notwendigen Schritte zum Mounten des Linux-Images in das lokale Dateisystem des ECUcore-9G20 zusammen.

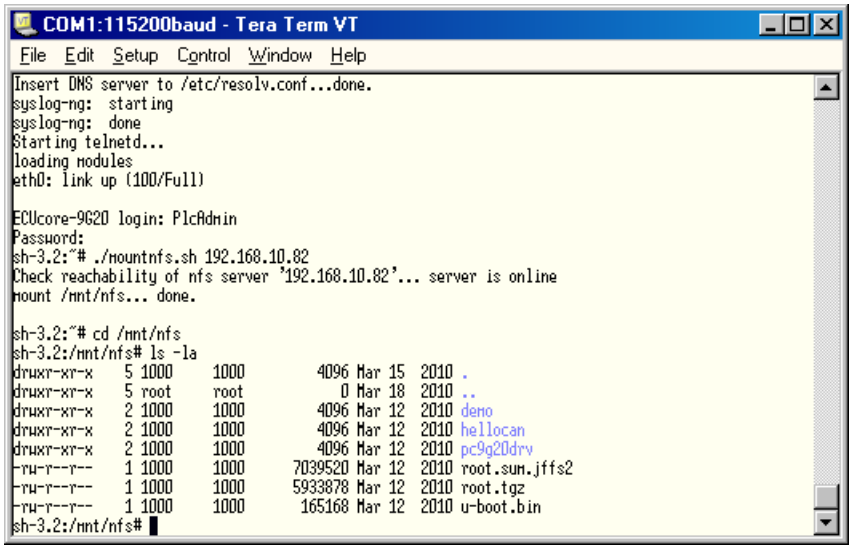


Bild 45: Mounten des Linux-Images in das lokale Dateisystem des ECUcore-9G20

7.5.2 Verwendung von FTP

Alternativ zur Einbindung des Linux-Images in das lokale Dateisystem des ECUcore-9G20 via NFS können Daten zwischen dem Entwicklungs-PC und dem ECUcore-9G20 über FTP in beiden Richtungen transferiert werden. Dabei kann das ECUcore-9G20 sowohl als Server als auch als Client fungieren.

Bei der Übertragung von ausführbaren Programmen über eine FTP-Verbindung ist zu beachten, dass hierbei grundsätzlich das "x"-Flag in den Dateiattributen ("eXecuteable") gelöscht wird. Daher ist nach einem FTP-Transfer das "x"-Flag mit Hilfe des Kommandos *"chmod"* wieder zu setzen (siehe auch Punkt "Aufruf von Programmen" im Abschnitt 10, "Tipps & Tricks im Umgang mit Linux"):

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

7.5.2.1 ECUcore-9G20 als FTP-Client

Beim Einsatz des ECUcore-9G20 als FTP-Client dient das Linux-Entwicklungssystem als Server. Diese Variante hat den Vorteil, dass beim späteren realen Einsatz des Gerätes im Feld kein Sicherheitsrisiko besteht, da auf dem Modul kein Serverdienst aktiviert sein muss und somit eine explizite Nutzerverwaltung nicht notwendig wird. Für den Einsatz des ECUcore-9G20 als FTP-Client ist zunächst wieder die dem Linux-Entwicklungssystem per DHCP zugewiesene IP-Adresse zu ermitteln. Das Vorgehen dazu ist im Abschnitt 6.5 beschrieben.

FTP-Download

Der Download von Dateien aus dem Linux-Image auf das ECUcore-9G20 erfolgt mit Hilfe des Befehls *"ftpget"*. Die beiden Parameter *"-u"* für Username und *"-p"* für Passwort dienen zur Authentifizierung am Hostsystem. Der Befehl *"ftpget"* besitzt folgende Syntax:

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

Um beispielsweise das im Abschnitt 7.2 übersetzte und in das Verzeichnis *"/tftpboot/demo"* kopierte Programm *"demo"* per FTP in das lokale Verzeichnis *"/tmp"* des ECUcore-9G20 zu übertragen, ist folgender Befehl notwendig:

```
ftpget  -u vmware  -p vmware  192.168.10.82  /tmp/demo  /tftpboot/demo/demo
```

FTP-Upload

Der Upload von Dateien des ECUcore-9G20 in das Linux-Image erfolgt mit Hilfe des Befehls *"ftpput"*. Die beiden Parameter *"-u"* für Username und *"-p"* für Passwort dienen zur Authentifizierung am Hostsystem. Der Befehl *"ftpput"* besitzt folgende Syntax:

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

Um beispielsweise das Startskript *"autostart"* vom ECUcore-9G20 per FTP in das Verzeichnis *"/tftpboot"* des Hostsystems zu übertragen, ist folgender Befehl notwendig:

```
ftpput -u vmware -p vmware  192.168.10.82  /tftpboot/autostart  /home/etc/autostart
```

Bild 46 verdeutlicht die Anwendung der Befehle *"ftpget"* und *"ftpput"* zum Download und Upload von Dateien über FTP.

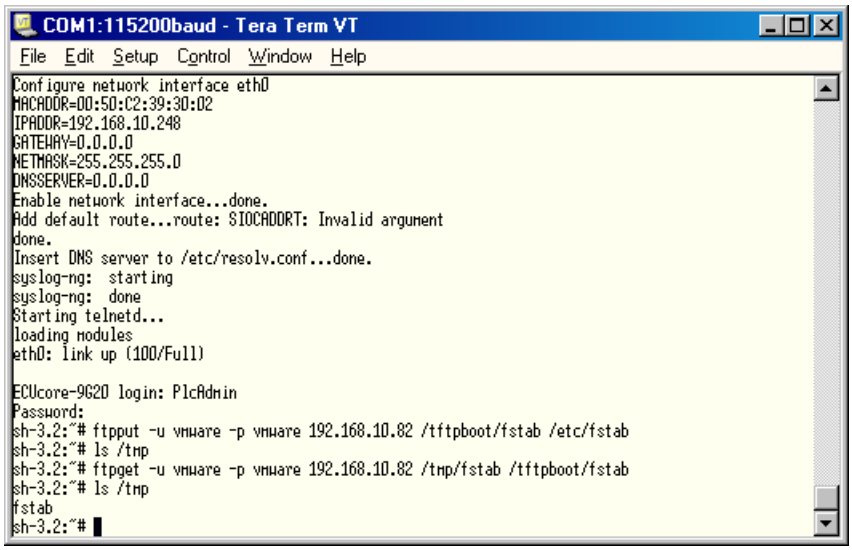


Bild 46: Download und Upload über FTP

7.5.2.2 ECUcore-9G20 als FTP-Server

Das ECUcore-9G20 verfügt über einen FTP-Server (FTP Deamon), der den Austausch von Dateien mit einem beliebigen FTP-Client ermöglicht (z.B. Up- und Download von Dateien zu bzw. von einem PC). Aus Sicherheits- und Performance-Gründen ist dieser FTP-Server jedoch standardmäßig deaktiviert und muss bei Bedarf zunächst manuell gestartet werden. Der Vorteil beim Einsatz des FTP-Servers auf dem ECUcore-9G20 besteht darin, dass für die Client-Seite komfortable grafische Programme existieren, die einen einfachen Datenaustausch ohne tiefgreifende Kenntnis von Linux-Kommandos ermöglichen. Ein Servicetechniker ist so beispielsweise beim späteren realen Einsatz des Gerätes im Feld in der Lage, mit einem grafischen FTP-Client auf seinem Windows-Laptop ein Firmwareupdate auf das ECUcore-9G20 zu übertragen oder Logfiles von diesem Modul auszulesen.

Die Aktivierung des FTP-Servers auf dem ECUcore-9G20 sowie die Anmeldung am Modul durch einen FTP-Client beschreibt Abschnitt 5.5.2. Geeignete FTP-Client Programme sind im Abschnitt 5.1 aufgeführt.

7.6 Übersetzen und Ausführen des Demoprojektes "demo"

7.6.1 Verwendung von "make"

Das VMware-Image des Linux-Entwicklungssystems enthält im Verzeichnis ***"/projects/ECUcore-9G20/user/demo"*** ein Demoprogramm, das den Zugriff auf die Ein- und Ausgänge des Moduls veranschaulicht. Das Demoprogramm bedient sich für die I/O-Zugriffe der Hilfe des unter ***"/projects/ECUcore-9G20/driver/pc9g20drv"*** abgelegten I/O-Treibers. Das hier beschriebene Demoprojekt, zumindest aber dessen Makefile, sollte als Template für eigene Projekte verwendet werden.

Während der Sourcecode über die Windows-Netzwerkumgebung mit jedem beliebigen Windows-Editor erstellt bzw. bearbeitet werden kann, ist das Übersetzen des Projektes ausschließlich innerhalb des Linux-Entwicklungssystems selbst möglich. Dazu kann entweder ein Konsolenfenster im Linux-Image verwendet werden (dort auch als "Terminal" bezeichnet), oder es ist der Zugriff über einen Telnet-Client "von außen" notwendig, in dem dann alle notwendigen Schritte analog erfolgen. Im Folgenden wird daher auch keine Unterscheidung zwischen Konsolenfenster innerhalb des Linux-Images und Telnet-Zugriff "von außen" getroffen.

Innerhalb des Konsolenfensters ist zunächst mit dem Befehl *"cd"* in das entsprechende Verzeichnis des Projektes zu wechseln, in dem sich das Makefile befindet. Zum Übersetzen des Demoprojektes ist dazu in das Verzeichnis *"/projects/ECUcore-9G20/user/demo/source"* zu wechseln:

```
cd  /projects/ECUcore-9G20/user/demo/source
```

Anschließend ist hier der Befehl *"make"* zu starten. Bild 47 zeigt die Anwendung der Befehle *"cd"* und *"make"* am Beispiel des im VMware-Image enthaltenen Demoprogramms.

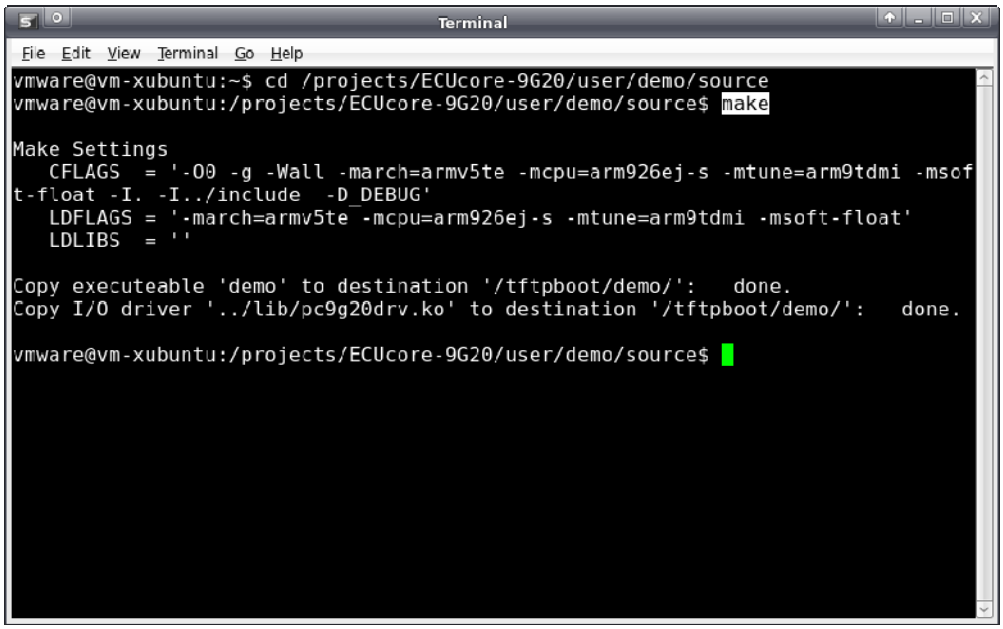


Bild 47: Übersetzen des Demoprojektes im Linux-Image

Bei der Abarbeitung des Makefiles wurden nach dem erfolgreichen Abschluss des Build-Prozesses sowohl das erstellte Demoprogramm selbst (File *"demo"*) als auch der zu seiner Ausführung benötigte I/O-Treiber (File *"pc9g20drv.ko"*) in das Verzeichnis *"/tftpboot/demo"* kopiert (siehe Screenshot in Bild 47). Mit dem Übersetzen und Kopieren sind alle notwendigen Schritte im Linux-Entwicklungssystem abgeschlossen.

Alle weiteren Schritte erfolgen jetzt ausschließlich auf dem ECUcore-9G20 selbst. Dazu ist zunächst die Anmeldung an der Kommando-Shell des Moduls notwendig (siehe Abschnitt 5.5.1). Innerhalb des Terminalprogramms bzw. Telnet-Clients sind anschließend folgende Schritte auszuführen:

1. Einbinden des Verzeichnisses *"/projects/tftpboot"* vom Linux-Entwicklungssystem in das lokale Filesystem des ECUcore-9G20 per NFS (siehe Abschnitt 7.5.1):

   ```
   mount  -t nfs  -o nolock  192.168.10.82:/tftpboot  /mnt/nfs
   ```

2. Wechseln in das NFS-Verzeichnis im lokalen Dateisystem mit Hilfe des Befehls *"cd"*:

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

   Das Verzeichnis *"/mnt/nfs"* auf dem ECUcore-9G20 ist identisch mit dem Verzeichnis *"/tftpboot"* des Linux-Entwicklungssystems im VMware-Image. Dementsprechend sind auch die vom Makefile auf dem Linux-Entwicklungssystem in das Verzeichnis *"/tftpboot/demo"* kopierten Files für das ECUcore-9G20 in seinem Dateisystem im Verzeichnis *"/mnt/nfs/demo"* zugänglich. Ein expliziter Download der ausführbaren Binärdateien auf das ECUcore-9G20 ist somit nicht notwendig.

3. Starten des Demoprogramms auf dem ECUcore-9G20.

   Da das Demoprogramm auf die Ein- und Ausgänge des ECUcore-9G20 zugreift, benötigt es für seine Ausführung den I/O-Treiber *"pc9g20drv"*. Dementsprechend ist zunächst der I/O-Treiber mit dem Befehl *"insmod"* explizit zu laden, anschließend kann das Demoprogramm gestartet werden:

```
insmod  pc9g20drv.ko
./demo
```

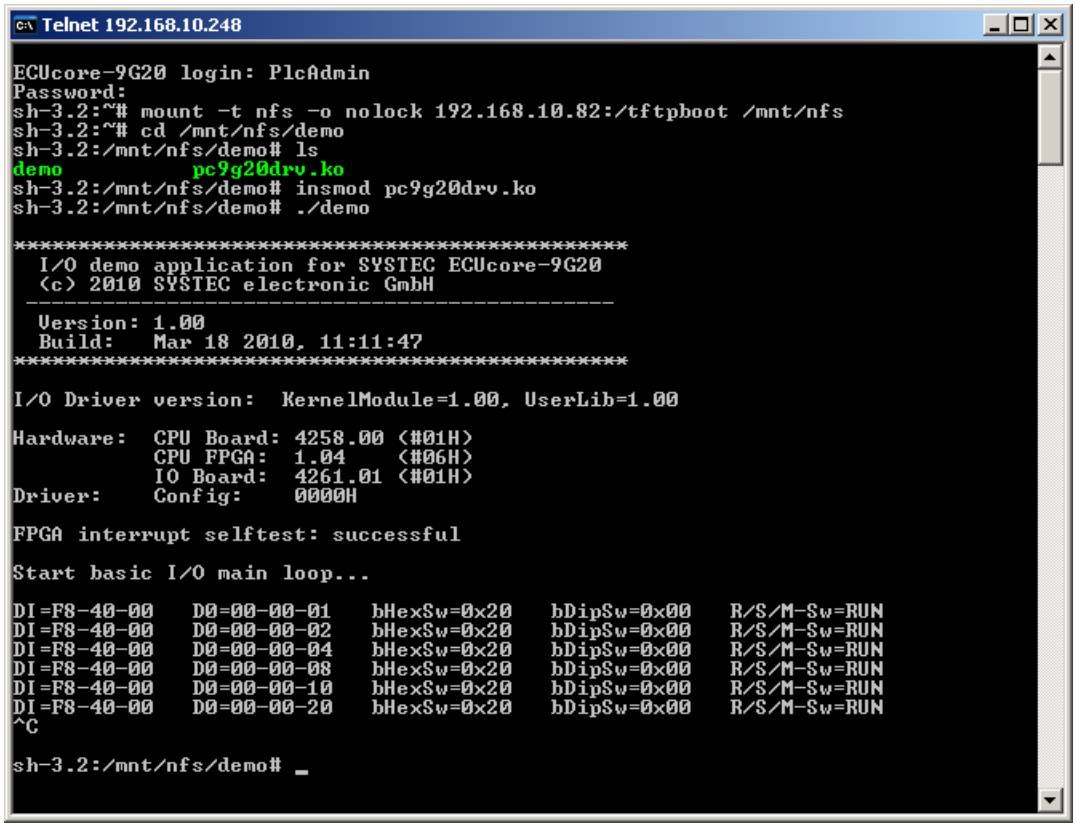


Bild 48: Ausführung des Demoprojektes "demo" auf dem ECUcore-9G20

Bild 48 verdeutlicht die Ausführung des Demoprojektes auf dem ECUcore-9G20. Das Demoprojekt kann entweder durch setzen des Run/Stop-Schalters in die Position "MRes" oder durch Drücken von "Ctrl+C" beendet werden.

7.6.2 Verwendung der grafischen IDE "Eclipse"

Im VMware-Image des Linux-Entwicklungssystems ist "Eclipse" als grafische IDE installiert. Diese erlaubt das Ausführen sämtlicher Arbeitsschritte im Softwareentwicklungsprozess wie Editieren, Übersetzen und Debuggen von Anwenderprogrammen innerhalb einer komfortablen Entwicklungsumgebung, vergleichbar beispielsweise mit "Visual Studio". Die folgenden Abschnitte beschreiben den Umgang mit der IDE "Eclipse" am Beispiel des im VMware-Image enthaltenen und im Abschnitt 7.6.1 bereits beschriebenen Demoprojektes im Pfad ***"/projects/ECUcore-9G20/user/demo"***.

7.6.2.1 Öffnen und Bearbeiten des Demoprojektes

Die grafische IDE "Eclipse" wird durch Anklicken des entsprechenden Desktop-Symbols gestartet. Dabei erscheint zunächst der Dialog "Workspace Launcher", wie im Bild 49 dargestellt. Im Feld "Workspace" ist der Pfad für das Workspace-Verzeichnis innerhalb des Projektbaumes einzutragen. Für das im Linux-Entwicklungssystem enthaltene Demo-Projekt ist dies *"/projects/ECUcore-9G20/user/demo/workspace"*.

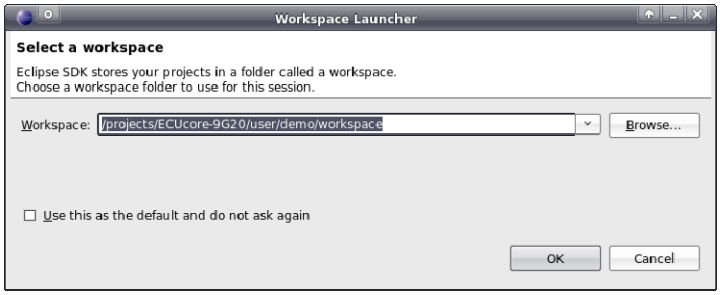


Bild 49: Eclipse-Dialog "Workspace Launcher"

Nach dem Betätigen der Schaltfläche "OK" startet die grafische Oberfläche der IDE und lädt den angegebenen Workspace. Bild 50 zeigt die Gesamtansicht von "Eclipse" nach dem Start.

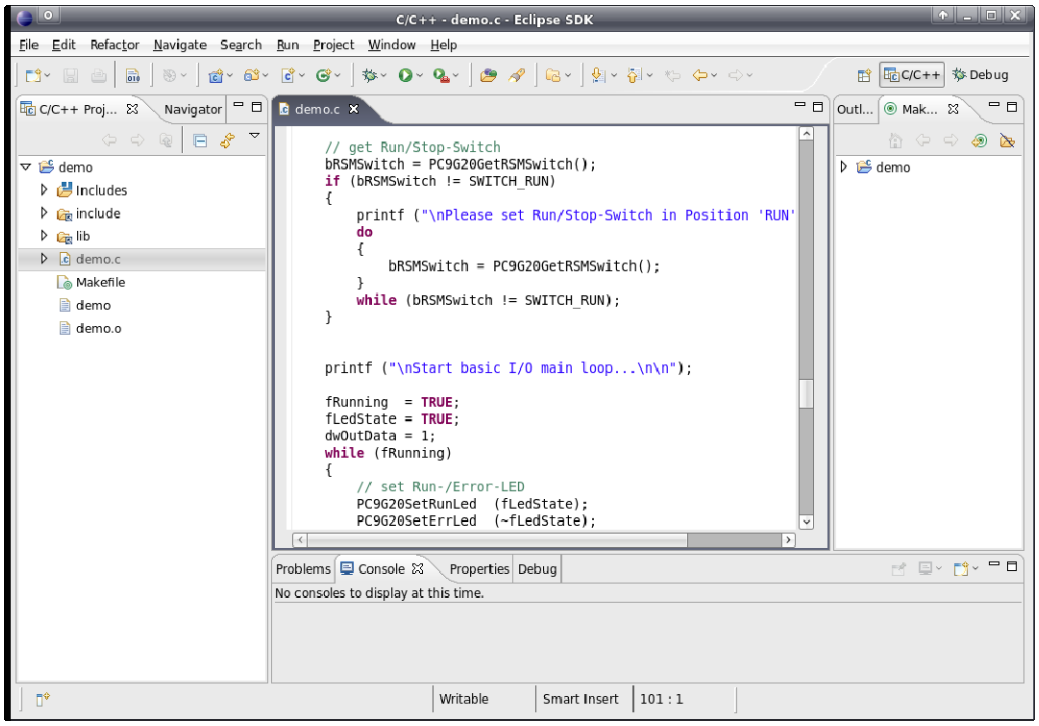


Bild 50: Ansicht der grafischen IDE "Eclipse"

Im Editorfenster kann der Sourcecode des Demoprogramms bearbeitet werden. Durch Doppelklick auf einen Eintrag im Projektbaum (links) lassen sich alle im Projekt enthaltenen Dateien öffnen und bearbeiten.

7.6.2.2 Übersetzen des Demoprojektes

Zum Übersetzen des Demoprojektes ist zunächst im Fenster *"Make Targets"* der Eintrag *"Demo"* durch Anklicken des vorangestellten Dreieckes aufzuklappen. Anschließend ist der Übersetzungslauf mit einem Doppelkick auf den Eintrag *"Build Project"* aufzurufen (siehe Bild 51). Dadurch führt Eclipse das Makefile des Demoprojektes im Verzeichnis *"source"* aus (*"/projects/ECUcore-9G20/user/demo/-source/Makefile"*). Das ist genau dasselbe Makefile, das bereits im Abschnitt 7.6.1 manuell in einem Terminalfenster aufgerufen wurde. Dementsprechend sind auch die im IDE-Fenster "Console" angezeigten Meldungen identisch zu denen im Bild 47. Ebenso werden auch hier wieder sowohl das erstellte Demoprogramm selbst (File *"demo"*) als auch der zu seiner Ausführung benötigte I/O-Treiber (File *"pc9g20drv.ko"*) in das Verzeichnis *"/tftpboot/demo"* kopiert. Folglich kann auch hier nach dem erfolgreichen Abschluss des Build-Prozesses das Demoprogramm wie im Abschnitt 7.6.1 beschrieben auf dem ECUcore-9G20 gestartet werden.

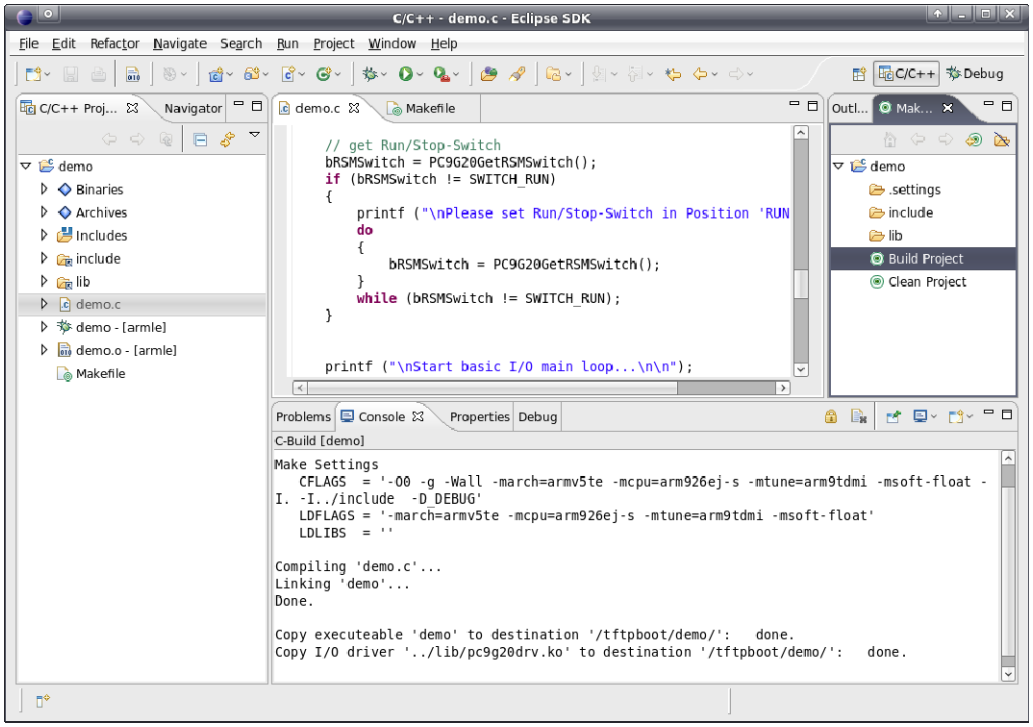


Bild 51: Übersetzen des Demoprojektes in Eclipse

Wenn beim Übersetzen Fehler oder Warnungen auftreten (z.B. infolge von Modifikationen des Demoprojekts durch den Anwender), dann werden diese im IDE-Fenster *"Problems"* strukturiert dargestellt. Durch einen Doppelklick auf einen Eintrag öffnet die IDE das betreffende Sourcefile und markiert die entsprechende Zeile im Editor.

Durch einen Doppelklick auf den Eintrag *"Clean Project"* (siehe Bild 51) werden alle generierten Files wieder gelöscht. Durch Anklicken der Einträge *"Demo Project"* bzw. *"Clean Project"* mit der rechten Maustaste können die entsprechenden Targets bearbeitet werden (z.B. Ändern des zugehörigen Namens).

7.6.2.3 Debuggen des Demoprojektes in der IDE

Einer der wesentlichen Vorteile bei der Verwendung von Eclipse ist die Möglichkeit, das übersetzte Programm innerhalb der IDE direkt auf Hochsprachenebene debuggen zu können. Dazu zählen z.B. die zeilenweise Programmabarbeitung auf C-Ebene, das Setzen von Breakpoints direkt im Quellcode sowie die Beobachtung von Variablen innerhalb der IDE. Während die Eclipse-IDE auf dem PC mit der Linux-Entwicklungsumgebung läuft, wird das zu debuggende Programm unter der Kontrolle eines Debugservers (gdbserver) direkt auf dem ECUcore-9G20 selbst ausgeführt. Bei dieser als "Remote-Debugging" bezeichneten Vorgehensweise wird sowohl Zugriff auf die Linux-Entwicklungsumgebung (Host-PC) als auch auf das ECUcore-9G20 (Target) benötigt.

Im Folgenden werden die zum Debuggen einer Anwenderapplikation notwendigen Schritte exemplarisch am Beispiel des im VMware-Image des Linux-Entwicklungssystems enthaltenen Demoprojektes beschrieben:

1. Starten des Debugservers mit dem Anwenderprogramm auf dem ECUcore-9G20

Zum Debuggen soll das Demoprogramm auf dem ECUcore-9G20 direkt aus dem NFS-Verzeichnis des Linux-Entwicklungssystems gestartet werden. Dazu ist zunächst die Anmeldung an der Kommando-Shell des Moduls notwendig (siehe Abschnitt 5.5.1). Innerhalb des Terminalprogramms bzw. Telnet-Clients sind anschließend folgende Schritte auszuführen:

1.1 Einbinden des Verzeichnisses *"/projects/tftpboot"* vom Linux-Entwicklungssystem in das lokale Filesystem des ECUcore-9G20 per NFS (siehe Abschnitt 7.5.1):

```
mount  -t nfs  -o nolock  192.168.10.82:/tftpboot  /mnt/nfs
```

1.2 Wechseln in das NFS-Verzeichnis im lokalen Dateisystem mit Hilfe des Befehls *"cd"*:

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

Das Verzeichnis *"/mnt/nfs"* auf dem ECUcore-9G20 ist identisch mit dem Verzeichnis *"/tftpboot"* des Linux-Entwicklungssystems im VMware-Image. Dementsprechend sind auch die vom Makefile auf dem Linux-Entwicklungssystem in das Verzeichnis *"/tftpboot/demo"* kopierten Files für das ECUcore-9G20 in seinem Dateisystem im Verzeichnis *"/mnt/nfs/demo"* zugänglich. Ein expliziter Download der ausführbaren Binärdateien auf das ECUcore-9G20 ist somit nicht notwendig.

1.3 Da das Demoprogramm auf die Ein- und Ausgänge des ECUcore-9G20 zugreift, benötigt es für seine Ausführung den I/O-Treiber *"pc9g20drv"*. Dementsprechend ist zunächst der I/O-Treiber mit dem Befehl *"insmod"* explizit zu laden:

```
insmod  pc9g20drv.ko
```

1.4 Starten des Demoprogramms auf dem ECUcore-9G20 unter Kontrolle des Debugservers, der dazu notwendige Befehl *"gdbserver"* besitzt folgendes, allgemeines Aufrufformat:

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

Für das oben angeführte Beispiel ergibt sich damit folgender Aufruf:

```
gdbserver  192.168.10.82:10000  ./demo
```

Dabei ist *"192.168.10.82"* die IP-Adresse des Linux-Entwicklungssystems (zur Ermittlung der IP-Adresse siehe Abschnitt 6.5). Die nach dem Doppelpunkt folgende Portnummer (hier: *"10000"*) ist frei wählbar, sie muss jedoch mit der nachfolgend im Punkt 2.3 innerhalb von Eclipse für die Target-Verbindung angegebenen Portnummer korrespondieren (auch siehe Bild 55).

Bild 52 verdeutlicht die auszuführenden Schritte auf dem ECUcore-9G20.

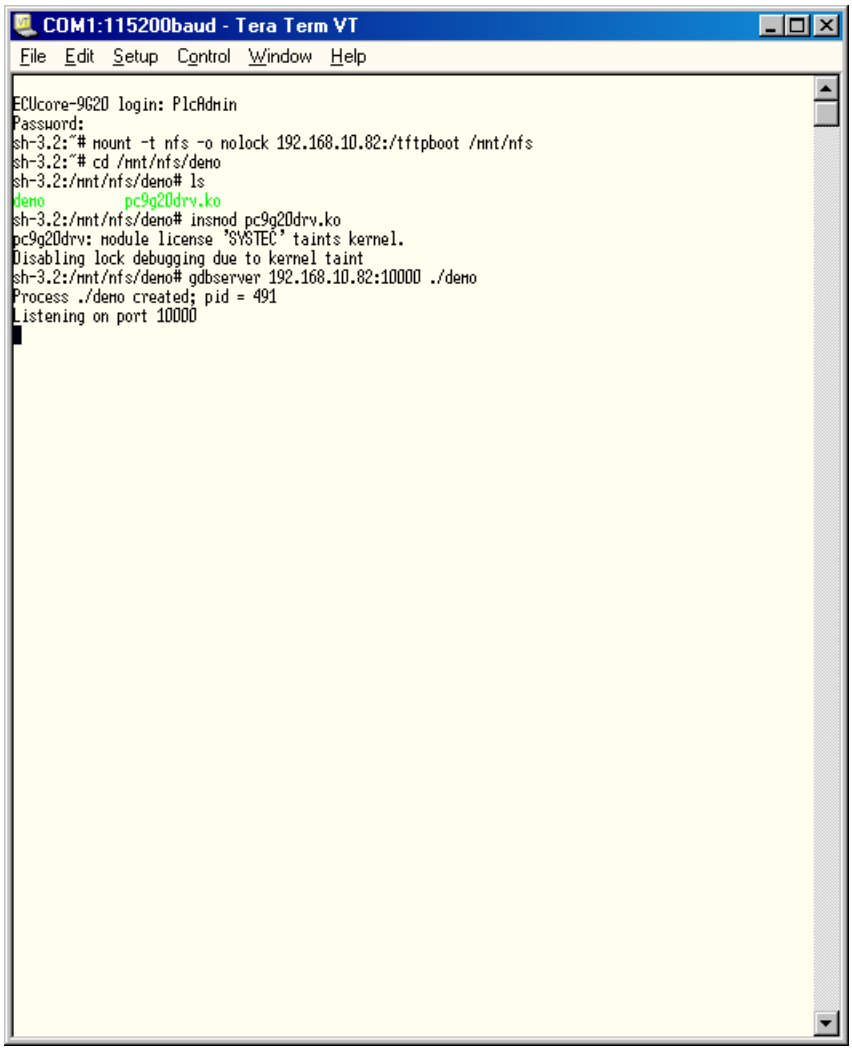


Bild 52: Starten des Debugservers auf dem ECUcore-9G20

Zur **Vereinfachung** enthält das ECUcore-9G20 im Auslieferungszustand im Verzeichnis *"/home"* das Shell-Skript ***"debug.sh"***, das alle hier im Punkt 1 beschriebenen Kommandos zusammenfasst. Damit muss innerhalb des Terminalprogramms bzw. Telnet-Clients nur noch das Shell-Skript gestartet werden, so dass die manuelle Eingabe aller hier beschriebenen Kommandos nicht mehr nötig ist:

```
cd
./debug.sh
```

Das Kommando *"cd"* ohne Parameter wechselt in das Home-Verzeichnis (*"/home"*), wo sich das Shell-Skript *"debug.sh"* befindet.

2. Konfigurieren des Debuggers in der IDE (nur einmalig beim ersten Aufruf notwendig)

Die Konfiguration des Debuggers in der IDE ist nur einmalig beim ersten Aufruf notwendig. Die Konfiguration erfolgt innerhalb von "Eclipse" über den Menüpunkt ***"Run -> Debug..."***. Dadurch wird der im Bild 53 dargestellte Konfigurationsdialog geöffnet. Hier sind folgende Schritte auszuführen:

2.1 Festlegen des im Debugger auszuführenden Programms (siehe Bild 53)

Dazu ist im Tabsheet *"Main"* im Feld *"C/C++ Application"* der Name des zu debuggenden Programms anzugeben (für das angegebene Beispiel: *"demo"*).

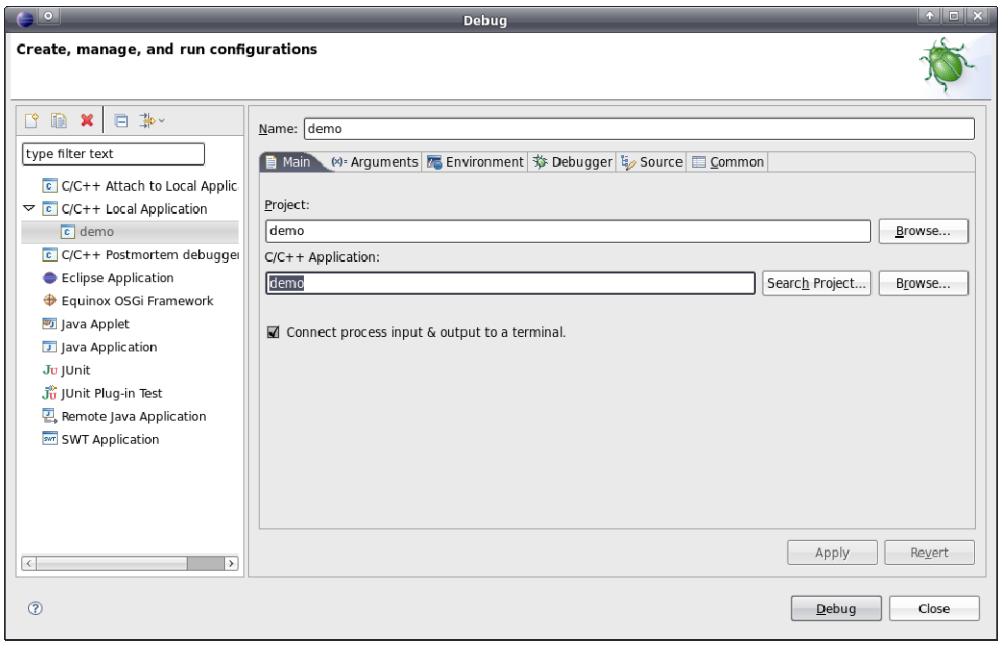


Bild 53: Festlegen der zu debuggenden Applikation

2.2 Auswahl des zu benutzenden GDB Debuggers (siehe Bild 54)

Auf der Host-Seite muss der in der GNU-Crosscompiler Toolchain für ARM9-Prozessoren enthaltene GDB Debugger verwendet werden. Um diesen auszuwählen, ist zunächst das Tabsheet *"Debugger"* zu aktivieren. In der Sektion *"Debugger Options"* ist im Unter-Tabsheet *"Main"* im Feld *"GDB debugger"* (siehe Bild 54) der entsprechende GDB Debugger aus der Crosscompiler Toolchain anzugeben (für das angegebene Beispiel: *"arm-none-linux-gnueabi-gdb"*).

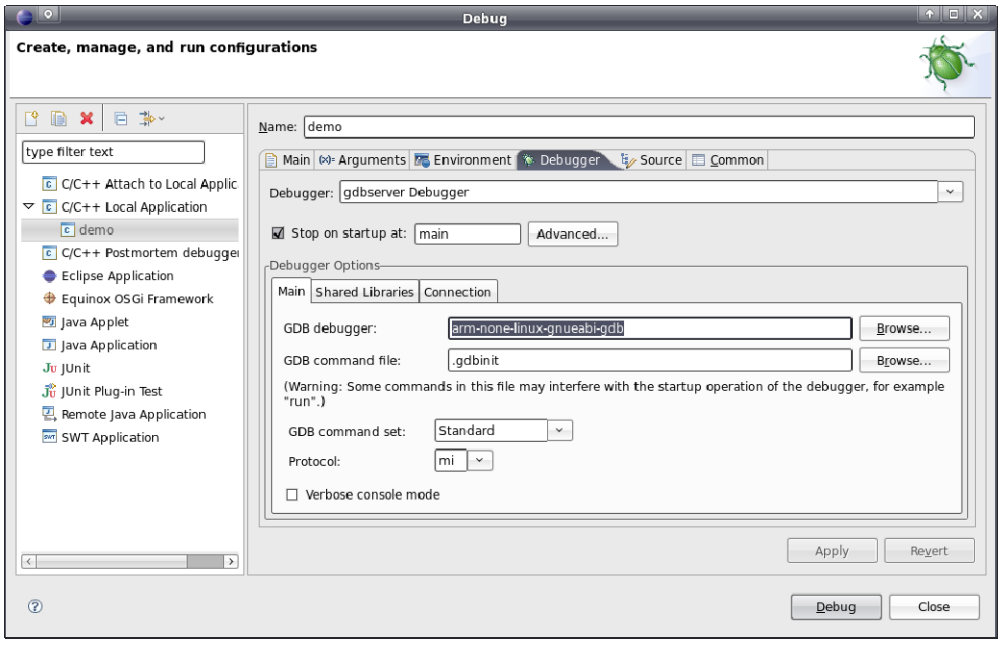


Bild 54: Auswahl des GDB Debuggers

2.3 Konfiguration der Verbindung zum Target (siehe Bild 55)

Die Konfiguration der Verbindung zum Target erfolgt ebenfalls im Tabsheet *"Debugger"*. Dazu sind in der Sektion *"Debugger Options"* im Unter-Tabsheet *"Connection"* die entsprechenden Informationen einzutragen (siehe Bild 55). Im Feld *"Host name or IP address"* ist die im Abschnitt 5.4 festgelegte IP-Adresse für das ECUcore-9G20 einzutragen (für das angegebene Beispiel: *"192.168.10.248"*). Im Feld *"Port number"* ist die beim Aufruf des Befehls *"gdbserver"* im Punkt 1.4 definierte Portnummer anzugeben (für das angegebene Beispiel: *"10000"*).

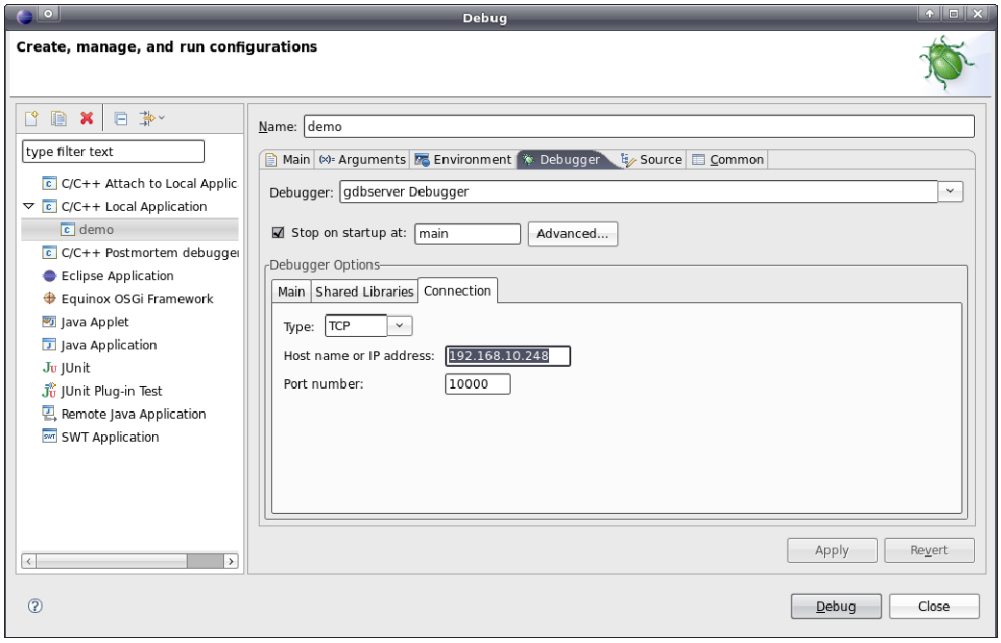


Bild 55: Konfiguration der Verbindung zum Target

Damit ist die Konfiguration des Debuggers abgeschlossen. Durch Betätigen der Schaltfläche *"Debug"* wird der Debugger mit den aktuellen Einstellungen gestartet.

3. Ausführen des Debuggers in der IDE

Nach Abschluss der im Punkt 2 beschriebenen Konfiguration kann der Debugger innerhalb von "Eclipse" über den Menüpunkt ***"Run -> Debug Last Launched"*** aufgerufen werden. Dabei wechselt die IDE von der "C/C++ Perspektive" in die "Debug Perspektive" (siehe Bild 56). Tabelle 10 listet die wichtigsten Debugger-Kommandos auf.

Tabelle 10: Übersicht wichtiger Debugger-Kommandos

Kommando		Funktion
F5		Step Into
F6		Step Over
F7		Step Return
F8		Run (ggf. bis zum nächsten Breakpoint)
Ctrl+Shift+B	(Doppelklick)	Toggle Line Breakpoint
		Terminate

Bild 56 verdeutlicht das Debuggen des Demoprogramms innerhalb der IDE "Eclipse". Beim Positionieren des Mauszeigers auf eine Variable wird deren aktueller Wert angezeigt.

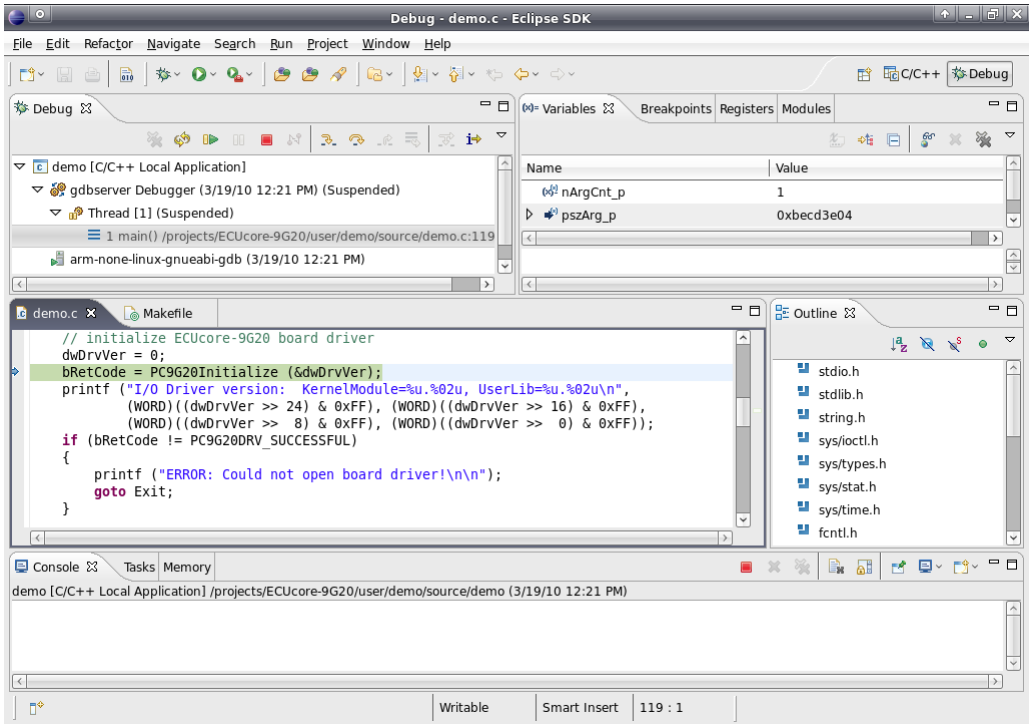


Bild 56: Debuggen des Demoprojektes in Eclipse

Die während des Debuggens auf dem ECUcore-9G20 generierten Terminalausgaben zeigt Bild 57.

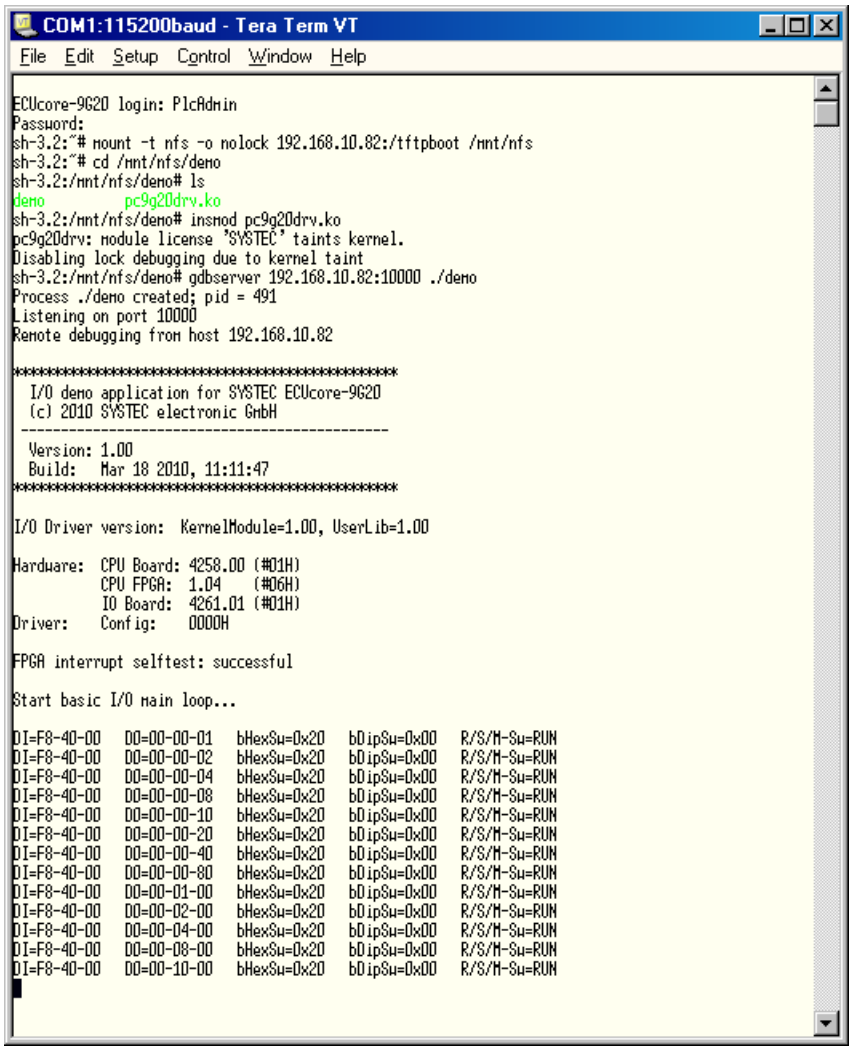


Bild 57: Terminalausgaben des ECUcore-9G20 während des Debuggens

7.7 Konfigurieren und Übersetzen von Linux-Image und U-Boot

Die gesamten Linux-Sourcen für das ECUcore-9G20 befinden sich innerhalb des VMware-Images für das Linux-Entwicklungssystem im Pfad ***"/projects/ECUcore-9G20/LinuxBSP-2.6"***. Um die Konfiguration des Linux-Kernels zu modifizieren, ist innerhalb eines Konsolenfensters zunächst mit dem Befehl *"cd"* in das Verzeichnis ***"/projects/ECUcore-9G20/LinuxBSP-2.6/ptxdist/ECUcore-9G20"*** zu wechseln und anschließend dort der Befehl *"ptxdist kernelconfig"* aufzurufen:

```
cd  /projects/ECUcore-9G20/LinuxBSP-2.6/ptxdist/ECUcore-9G20
ptxdist  kernelconfig
```

Bild 58 zeigt die typische Oberfläche zur Konfiguration des Linux-Kernels.

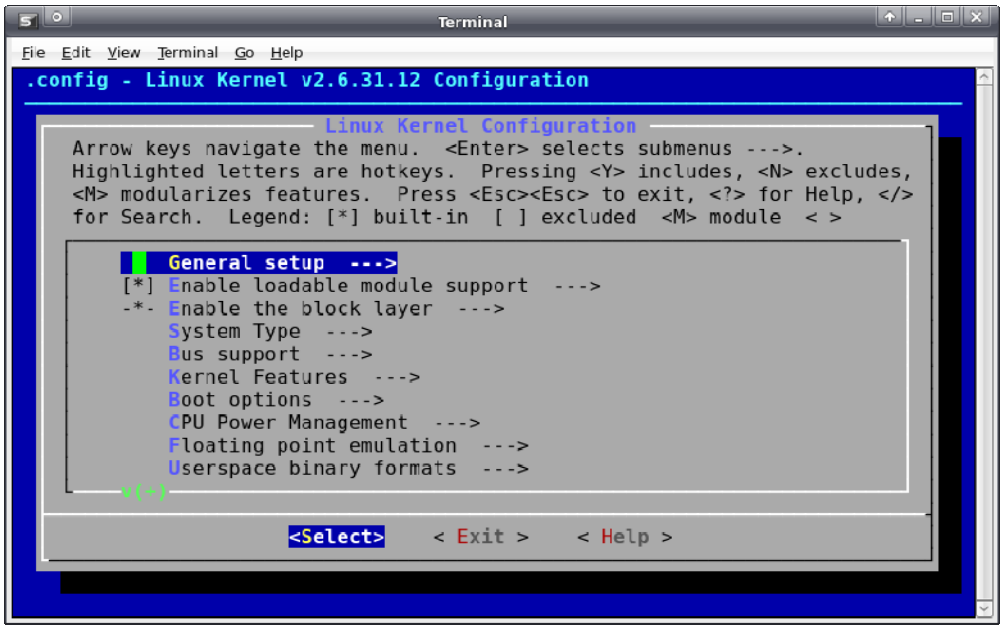


Bild 58: Benutzeroberfläche zur Konfiguration des Linux-Kernels

Die Konfiguration der User-Applikationen incl. BusyBox erfolgt im selben Verzeichnis wie die Konfiguration des Linux-Kernels. Hierzu ist der Befehl *"ptxdist menuconfig"* aufzurufen:

```
cd  /projects/ECUcore-9G20/LinuxBSP-2.6/ptxdist/ECUcore-9G20
ptxdist  menuconfig
```

Bild 59 zeigt die typische Oberfläche zur Konfiguration von User-Applikationen incl. BusyBox.

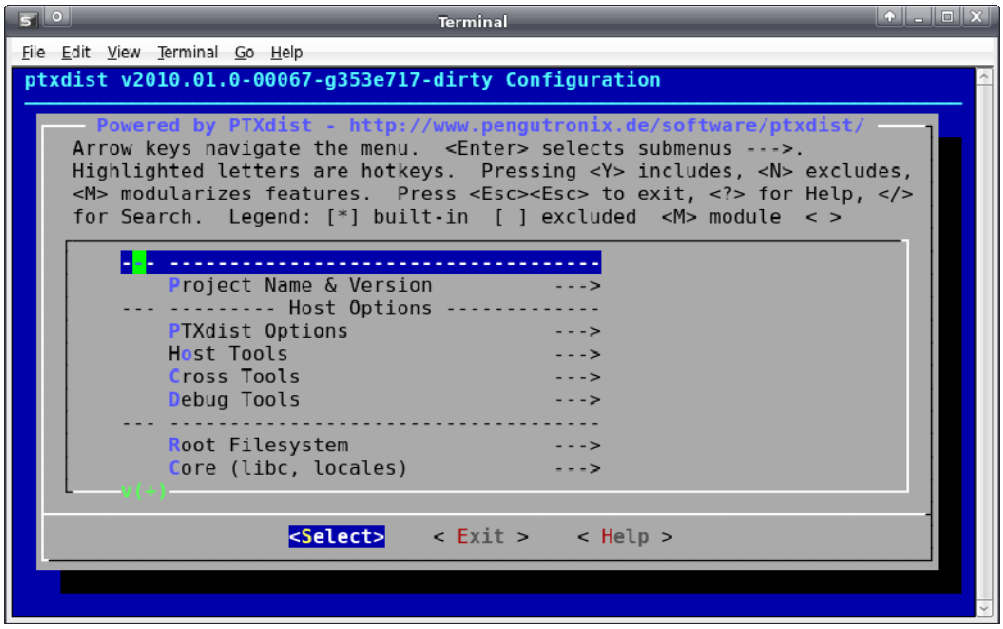


Bild 59: Benutzeroberfläche zur Konfiguration von User-Applikationen incl. BusyBox

Nach Abschluss der Konfiguration von Linux-Kernel, User-Applikationen sowie BusyBox ist in das übergeordnete Verzeichnis ***"/projects/ECUcore-9G20/LinuxBSP-2.6"*** zu wechseln. Durch den Aufruf von *"make"* werden sämtliche Softwarekomponenten übersetzt und zu einem neuen Image zusammengefasst:

```
cd  /projects/ECUcore-9G20/LinuxBSP-2.6
make
```

Um den Bootloader "U-Boot" neu zu erstellen, ist in das Verzeichnis ***"/projects/ECUcore-9G20/LinuxBSP-2.6/u-boot"*** zu wechseln und dort das Kommandos *"make"* aufzurufen:

```
cd  /projects/ECUcore-9G20/LinuxBSP-2.6/u-boot
make
```

Der Aufruf des Shell-Skripts *"copy_image"* kopiert Linux-Image und "U-Boot" in das Verzeichnis *"/tftpboot"*, von wo aus beide Softwarekomponenten auf das ECUcore-9G20 geladen werden können (siehe Abschnitt 5.15):

```
cd  /projects/ECUcore-9G20/LinuxBSP-2.6
./copy_image
```

8 Anpassen und Testen der Hardwareanschaltung

8.1 Driver Development Kit (DDK) für das ECUcore-9G20

Das Driver Development Kit (DDK) für das ECUcore-9G20 wird als zusätzliches Softwarepaket mit der Artikelnummer SO-1106 vertrieben. Es ist nicht im Lieferumfang des Development Kit ECUcore-9G20 enthalten. Details zum DDK beschreibt das "Software Manual Driver Development Kit für ECUcore-9G20" (Manual-Nr.: L-1257).

Das Driver Development Kit für das ECUcore-9G20 ermöglicht dem Anwender die eigenständige Anpassung der I/O-Ebene an ein selbst entwickeltes Baseboard. Damit ist der Anwender in der Lage, den I/O-Treiber komplett an die eigenen Bedürfnisse anzupassen.

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

- Peripherie (in der Regel GPIO) des AT91SAM9G20
- on-board FPGA (Lattice ECP2-6)
- 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 60 vermittelt einen Überblick über die DDK-Struktur und die enthaltenen Komponenten. Im DDK enthalten sind die Sourcen der FPGA-Software (VHDL), des Linux Kernel-Treibers (*pc9g20drv.ko*) und der Linux User-Bibliothek (*pc9g20drv.so*). Ebenfalls Bestandteil des DDK ist ein PLD/FPGA Programming Tool (*pldtool* + *plddrv.ko*), das ein Softwareupdate des FPGA unter Linux ohne zusätzliche JTAG-Hardware ermöglicht.

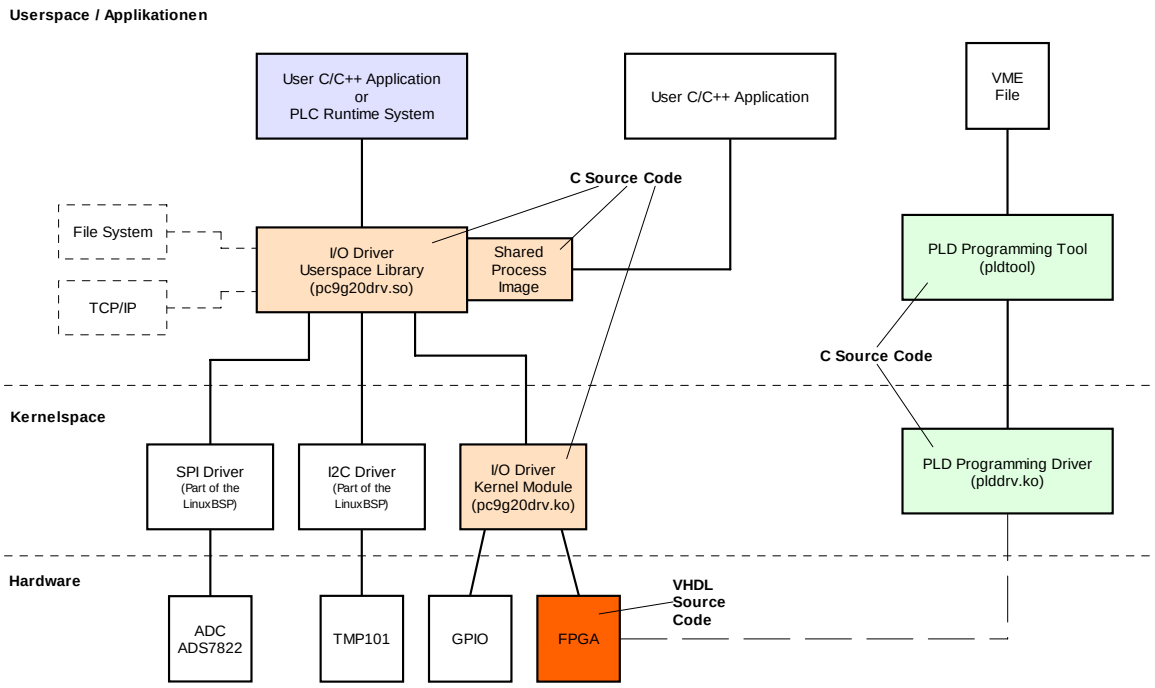


Bild 60: Übersicht zum Driver Development Kit für das ECUcore-9G20

Lieferumfang / Komponenten des DDK:

Das DDK beinhaltet folgende Komponenten:

1. VHDL-Projekt für FPGA, beinhaltet alle notwendigen Files um FPGA-Software neu zu erstellen (VHLS-Sourcefiles, Pin-Zuordnung, Timingeinstellungen, Projektfile usw.)

2. Sourcecode für Linux Kernel-Treiber (*pc9g20drv.ko*, siehe Bild 60), beinhaltet alle notwendigen Files um Kernel-Treiber neu zu erstellen (C- und H-Files, Makefile usw.)

3. Sourcecode für Linux User-Bibliothek (*pc9g20drv.so*, siehe Bild 60), beinhaltet alle notwendigen Files um User-Bibliothek (incl. der Implementierung des Shared Process Image) neu zu erstellen (C- und H-Files, Makefile usw.)

4. PLD/FPGA Programming Tool (*pldtool* + *plddrv.ko*), ermöglicht das Softwareupdate des FPGA unter Linux ohne zusätzliche JTAG-Hardware

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

6. Dokumentation

Das Driver Development Kit setzt das Softwarepaket **SO-1105** ("VMware-Image des Linux-Entwicklungssystems") voraus, das sowohl die Quellen des verwendeten LinuxBSP als auch die erforderliche GNU-Crosscompiler Toolchain für ARM9-Prozessoren enthält.

8.2 Testen der Hardwareanschaltung

Das ECUcore-9G20 ist primär als Zulieferteil für den Einbau in industriellen Steuerungen bestimmt. Dazu wird das ECUcore-9G20 typischerweise auf einer anwenderspezifischen Basisplatine betrieben. Um hier eine einfache Kontrolle der korrekten I/O-Anschaltung zu ermöglichen, steht das Testprogramm *"iodrvdemo"* zur Verfügung. Dieses Testprogramm setzt direkt auf dem I/O-Treiber auf und ermöglicht so den unmittelbaren Peripheriezugriff. Das Testprogramm ist bereits als direkt ausführbares Binary *"/home/bin/iodrvdemo"* auf dem ECUcore-9G20 vorinstalliert. Zudem sind die Sourcen des Programms als Referenz-Applikation im I/O-Treiber Projekt enthalten (siehe Abschnitt 7.3.

Das Testprogramm *"iodrvdemo"* ist wie folgt zu starten:

```
cd  /home/bin
insmod  pc9g20drv.ko
./iodrvdemo
```

Bild 61 veranschaulicht das Testen der Hardwareanschaltung mit *"iodrvdemo"*.

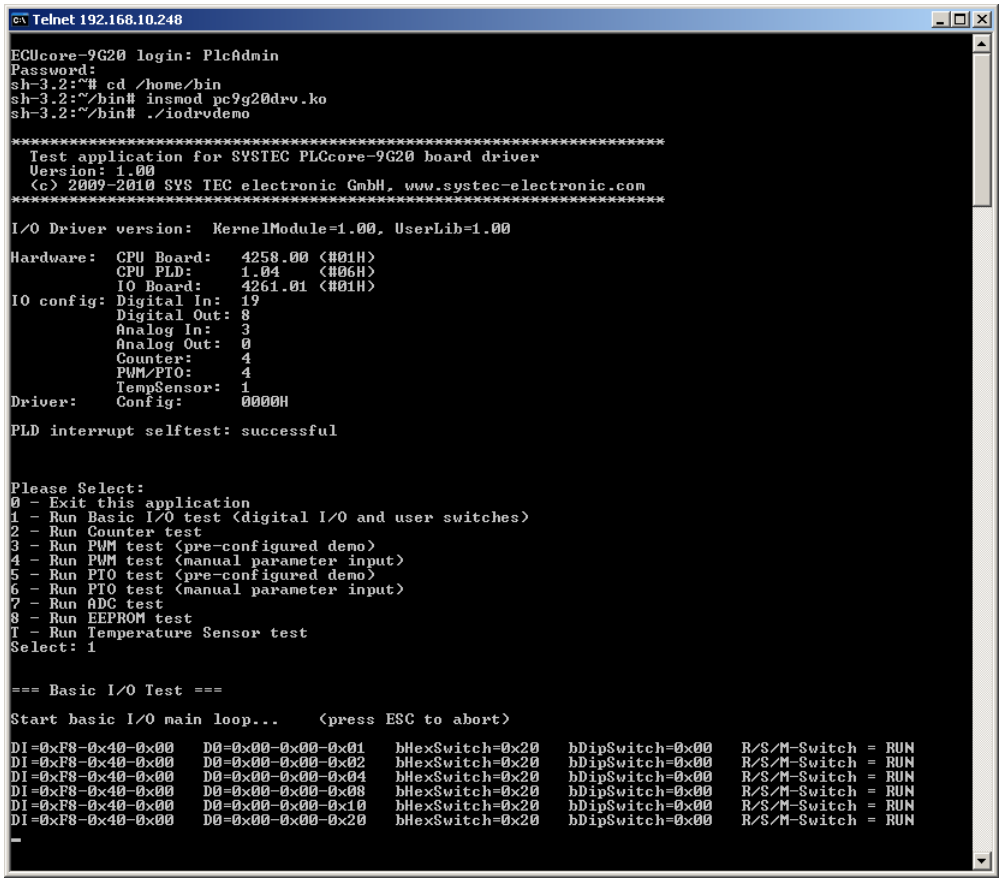


Bild 61: Testen der Hardwareanschaltung mit "iodrvdemo"

9 Nutzung von USB- und SD-Schnittstelle

9.1 Nutzung der USB-Schnittstelle

Das auf dem ECUcore-9G20 eingesetzte Embedded Linux unterstützt die Nutzung von USB-Geräten per "Hot Plug&Play". Das ECUcore-9G20 fungiert dabei als USB-Host und ist in Lage, USB-Devices wie z.B. USB-Memory-Sticks anzusprechen. Alle dafür notwendigen Treiber sind bereits im Linux-Image des ECUcore-9G20 enthalten. Die Reaktionen auf das Anstecken bzw. Entfernen eines USB-Memory-Sticks können vom Anwender flexibel angepasst werden.

Das im Linux-Image enthaltene *"mdev"* übernimmt bereits auf System-Ebene die Signalisierung und Verarbeitung von Ereignissen wie z.B. Anstecken oder Entfernen von USB-Memory-Sticks. Durch einen entsprechenden Eintrag in der Konfigurationsdatei *"/etc/mdev.conf"* werden alle relevanten Ereignisse an das anwenderspezifische Shell-Skript *"/home/etc/hotplug.sh"* weiter gemeldet. Die im Lieferumfang des ECUcore-9G20 enthaltene Standardimplementierung von *"hotplg.sh"* implementiert folgendes Verhalten:

Beim Anstecken eines Sticks bindet das Shell-Skript *"/home/etc/hotplug.sh"* das neu erkannte Gerät automatisch in das Unterverzeichnis *"/mnt/usb"* ein (Linux-Kommando *"mount"*). Anschließend wird das anwenderspezifische Skript *"/home/etc/diskadded.sh"* ausgeführt. Beim Abziehen des Sticks wird die Verbindung zum Unterverzeichnis *"/mnt/usb"* wieder aufgehoben (Linux-Kommando *"umount"*) und anschließend das anwenderspezifische Skript *"/home/etc/diskremoved.sh"* abgearbeitet. Beiden Skripten wird bei ihrem Aufruf das betreffende Verzeichnis für das zugehörige USB-Gerät als Parameter *"$1"* übergeben. Damit können anwenderdefinierte Aktionen als Reaktion auf diese Ereignisse automatisiert werden. Bild 62 veranschaulicht die beschriebenen Abläufe.

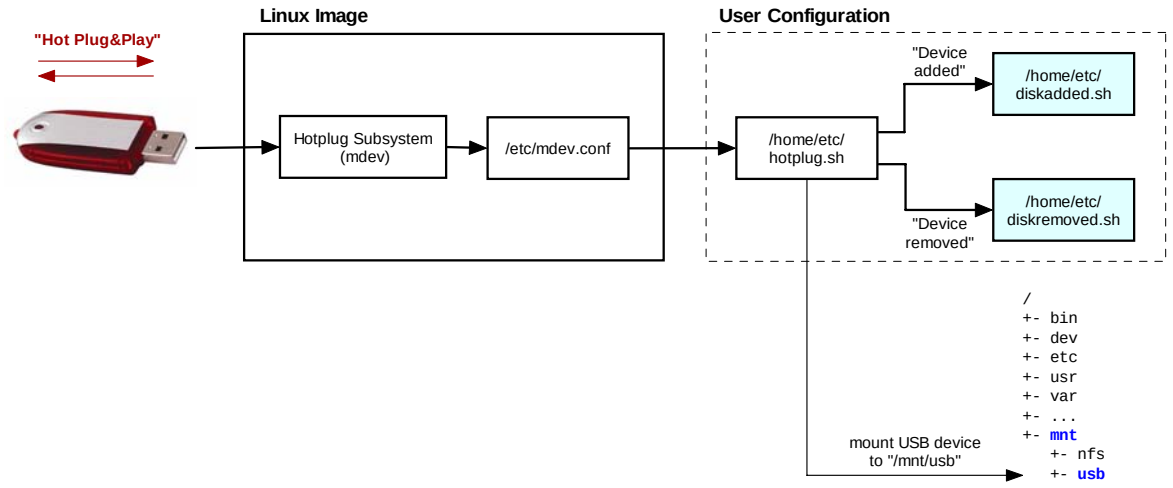


Bild 62: Interne Abläufe beim Anstecken bzw. Entfernen eines USB-Memory-Sticks

Das nachfolgende Beispiel zeigt eine einfache Variante für das Shell-Skript *"diskadded.sh"*, das beim Anstecken eines USB-Memory-Sticks die auf ihm enthaltenen Dateien automatisch auflistet:

```
#!/bin/sh

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

Das anwenderspezifische Shell-Skript *"hotplug.sh"* sowie die von ihm gerufenen Skripte *"diskadded.sh"* und *"diskremoved.sh"* werden alle im Kontext eines Systemprozesses ausgeführt, dem keine Konsole zugeordnet ist. Folglich erscheinen die von den Skripten generierten Ausgaben nicht im Terminalfenster. Um diese Ausgaben dennoch anzeigen zu können, leitet das Skript *"hotplug.sh"* sämtliche Meldungen in die Datei *"/var/log/hotplug.log"* um. Diese Datei kann dann beispielsweise mit Hilfe des Linux-Kommandos *"cat"* im Terminalfenster ausgegeben werden:

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

Bild 63 verdeutlicht die Ausgabe der von den Skripten beim Anstecken und Entfernen von USB-Memory-Sticks generierten Meldungen.

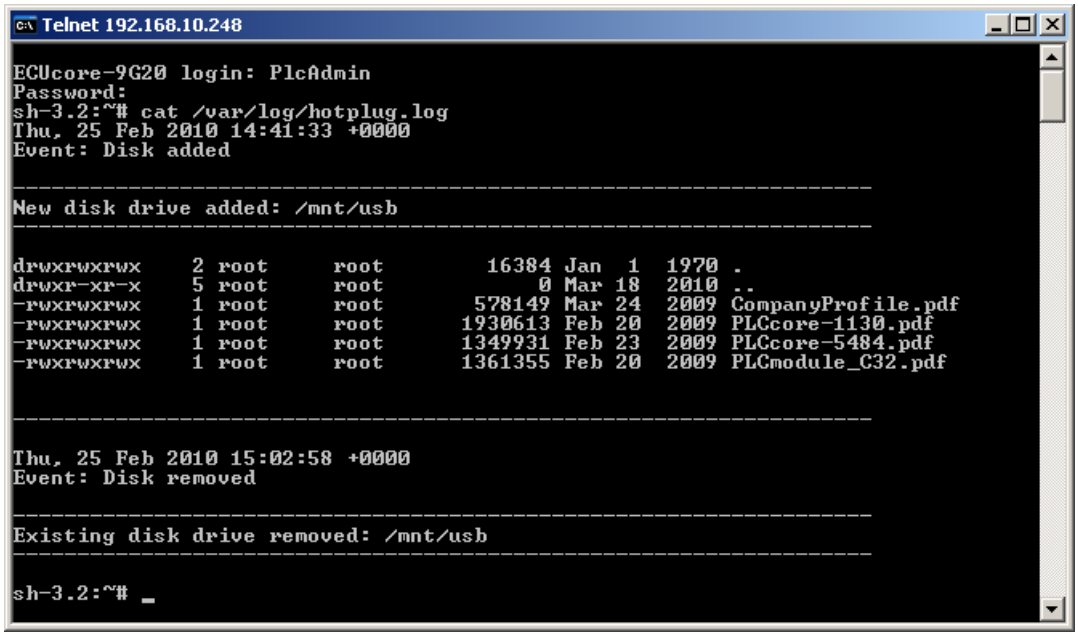


Bild 63: Ausgabe der in die Datei "/var/log/hotplug.log" umgeleiteten Meldungen

Die bei Auslieferung des Moduls auf dem ECUcore-9G20 vorinstallierte Version des Shell-Skriptes *"diskadded.sh"* prüft, ob sich im Wurzelverzeichnis des angesteckten USB-Memory-Sticks eine Datei *"autostart"* befindet. Ist diese vorhanden, wird diese aufgerufen und ausgeführt. Dies ermöglicht individuelle Aktionen wie beispielsweise das automatisierte Aufspielen von Softwareupdates auf das ECUcore-9G20.

Hinweis zur Nutzung von USB-Memory-Sticks:

Schreibzugriffe auf USB-Sticks erfolgen asynchron. Die Rückkehr der *"write"*-Funktion bedeutet somit nicht, dass die Daten bereits vollständig auf den Stick geschrieben wurden. Vielmehr werden die Daten vom ECUcore-9G20 zunächst im RAM zwischengepuffert und anschließend im Hintergrund auf den Stick übertragen. In C/C++ Programmen ist der Schreibvorgang erst nach einem *"close"*-Aufruf vollständig abgeschlossen. Für Shell-Skripte steht das Kommando *"sync"* zur Verfügung.

9.2 Nutzung der SD-Schnittstelle

Das auf dem ECUcore-9G20 eingesetzte Embedded Linux unterstützt die Nutzung von SD-Cards. Alle dafür notwendigen Treiber sind bereits im Linux-Image des ECUcore-9G20 enthalten. Um die SD-Card nutzen zu können, sind folgende Treiber zu laden:

```
modprobe  at91_mci
modprobe  mmc_block
```

Nach dem Laden aller Treiber kann die SD-Card mit Hilfe des Linux-Kommandos *"mount"* in das lokale Dateisystem des ECUcore-9G20 eingebunden werden:

```
mount  -t vfat  /dev/mmcblk0p1  /mnt/sd
```

Nach Ausführung des *"mount"*-Befehls ist die SD-Card im Pfad *"/mnt/sd"* eingebunden. Das Auflisten der auf der SD-Card enthaltenen Dateien ist mit folgenden Kommandos möglich:

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

Bild 64 verdeutlicht das Laden der Treiber, das Einbinden der SD-Card in das lokale Dateisystem sowie den Zugriff auf die SD-Card am ECUcore-9G20.

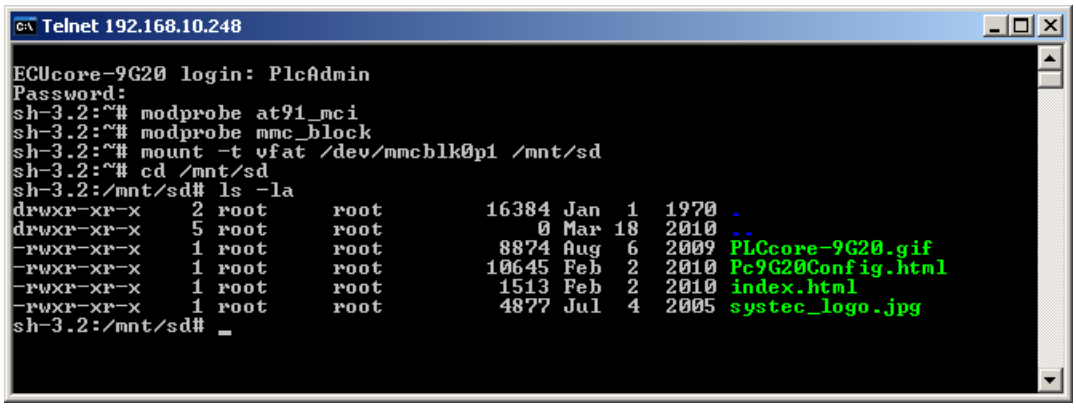


Bild 64: Zugriff auf die SD-Card am ECUcore-9G20

Hinweis zur Nutzung der SD-Card:

Schreibzugriffe auf die SD-Card erfolgen asynchron. Die Rückkehr der *"write"*-Funktion bedeutet somit nicht, dass die Daten bereits vollständig auf die SD-Card geschrieben wurden. Vielmehr werden die Daten vom ECUcore-9G20 zunächst im RAM zwischengepuffert und anschließend im Hintergrund auf die SD-Card übertragen. In C/C++ Programmen ist der Schreibvorgang erst nach einem *"close"*-Aufruf vollständig abgeschlossen. Für Shell-Skripte steht das Kommando *"sync"* zur Verfügung.

10 Tipps & Tricks im Umgang mit Linux

Dieser Abschnitt skizziert in Kurzform einige Besonderheiten, die beim Umgang mit Linux zu beachten sind. Hier können jedoch nur einige grundlegende Hinweise gegeben werden, für tiefgreifendere Informationen ist ein geeignetes Linux-Nachschlagewerk notwendig.

- **Aufruf von Programmen (Suchpfad)**

  Im Gegensatz zu DOS/Windows durchsucht Linux beim Aufruf eines Kommandos nur die in der Umgebungsvariablen *"PATH"* definierten Pfade, nicht aber das aktuelle Verzeichnis. Um beispielsweise das Programm *"demo"* aufzurufen, das sich im aktuellen Verzeichnis befindet, muss beim Aufruf explizit der Verweis auf das aktuelle Verzeichnis in Form von *"./"* mit angegeben werden. Das Programm *"demo"* ist daher als *"./demo"* aufzurufen.

  Die Standardkommandos wie z.B. *"ls"* können ohne Pfadangabe aufgerufen werden, da sie in einem der durch die Umgebungsvariable *"PATH"* definierten Pfade liegen.

- **Aufruf von Programmen (Ausführungsrechte)**

  Um ein Programm unter Linux ausführen zu können, muss die zugehörige Datei explizit als ausführbar gekennzeichnet sein. Dafür verantwortlich ist das "x"-Flag in den Dateiattributen ("eXecuteable"), die beim Aufruf von *"ls -la"* angezeigt werden, z.B.:

  ```
  ls -la ./mountnfs.sh
  -rwXr-Xr-X    1 1000     users        2236 Jan 21  2009 ./mountnfs.sh
  ```

  Ist dieses Flag nicht gesetzt, ist die Datei als nicht ausführbar gekennzeichnet und das betreffende Kommando kann nicht aufgerufen werden. Dies ist z.B. generell nach dem FTP-Download einer Datei der Fall. Das "x"-Flag kann jedoch mit Hilfe des Kommandos *"chmod"* wieder gesetzt werden:

  ```
  chmod +X ./mountnfs.sh
  ```

  Das Kommando *"chmod"* ermöglicht generell die Beeinflussung aller Dateiattribute (sowohl setzen als auch löschen). Details hierzu liefert der Aufruf *"chmod --help"*.

- **Automatische Vervollständigung von Eingaben durch TAB-Taste**

  Ein äußerst praktisches Feature unter Linux ist die automatische Vervollständigung von Datei- bzw. Kommandonamen mittels TAB-Taste. Damit müssen nur so viele Zeichen des Datei- oder Kommandonamens eingegeben werden, bis es nur noch einen Namen gibt, auf den diese Zeichen zutreffen. Durch Drücken der TAB-Taste wird der Rest der Eingabe dann automatisch vervollständigt.

  Beispiel: Um im Verzeichnis *"/home"* des ECUcore-9G20 das Shell-Skript *"./mountnfs.sh"* aufzurufen, ist die Eingabe von *"./m"* ausreichend. Beim Drücken der TAB-Taste vervollständigt die Shell dann automatisch die eingegebenen Zeichen zum Kommando *"./mountnfs.sh"*.

- **Anzeigen von Umgebungsvariablen**

  Die Anzeige von Umgebungsvariablen ist mit Hilfe des Kommandos *"echo"* möglich. So erfolgt beispielsweise die Anzeige des aktuell konfigurierten Suchpfades mit Hilfe von *"echo $PATH"*.

- **Setzen von Umgebungsvariablen**

  Beim Setzen einer Umgebungsvariablen ist zu beachten, dass diese nur innerhalb der aktuellen Shell-Instanz bzw. den von der aktuellen Shell-Instanz aufgerufenen Subshells sichtbar ist. Wird eine Umgebungsvariable beispielsweise in einem Shell-Skript definiert, ist sie nach dessen Ausführung nicht mehr gültig. Der Grund hierfür ist, dass zur Ausführung eines Skripts eine neue Shell-Instanz aufgerufen wird, die das Skript abarbeitet. Innerhalb dieser Shell-Instanz (und damit innerhalb des darin ausgeführten Skripts) kann auf die betreffende Umgebungsvariable zugegriffen werden. Nach der Abarbeitung des Skripts wird die eigens hierfür gestartete Shell-Instanz jedoch wieder beendet und damit auch die darin definierte Umgebungsvariable verworfen.

  Eine Möglichkeit, Umgebungsvariablen (z.B. *"TZ="*) dauerhaft zu setzen, besteht darin, diese im Skript *"/etc/profile.environment"* zu definieren. Dieses Skript wird einmalig beim Start der ersten Shell-Instanz während des Bootvorganges ausgeführt, so dass die darin definierten Variablen während der gesamten Laufzeit erhalten bleiben und an alle später gestarteten Subshells "vererbt" werden.

- **Hilfe zu einem Programm**

  Unter Linux kann eine kurze Hilfe zum Programm mit Beschreibung der unterstützten Parameter in der Regel durch Eingabe des Programmnamens gefolgt von *"--help"* aufgerufen werden (z.B. *"mount --help"*).

- **Fehlersuche in Shell-Skripten**

  Zur Vereinfachung der Fehlersuche bei der Ausführung von Shell-Skripten kann das zu untersuchende Skript mit dem Befehl "sh -x <script_file>" aufgerufen werden. Die Option *"-x"* weist die Shell an, jede ausgeführte Zeile auf der Konsole auszugeben. Dadurch kann z.B. leicht nachvollzogen werden, welche Zweige einer bedingten Ausführung (*"if"*, *"case"*, *"while"* usw.) tatsächlich ausgeführt wurden.

- **Setzen der Zeitzone auf dem ECUcore-9G20**

  Das Setzen der Zeitzone auf dem ECUcore-9G20 erfolgt durch die Definition der Umgebungsvariablen *"TZ"* im Startskript *"/etc/profile.environment"*. Die entsprechende Definition für Deutschland (UTC +1:00) mit Angabe von Beginn und Ende der Sommerzeit ist dort bereits als Kommentar enthalten. Für andere Zeitzonen ist die Definition entsprechend anzupassen.

Anhang A: GNU GENERAL PUBLIC LICENSE

Das auf dem ECUcore-9G20 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.

Zusätzliche SYSTEC-Systemsoftware sowie vom Anwender entwickelte 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 ECUcore-9G20
Dokumentnummer:	L-1253d_1, 1. Auflage März 2010

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
August-Bebel-Str. 29
D - 07973 Greiz
GERMANY
Fax : +49 (0) 36 61 / 6279-99
Email: info@systec-electronic.com