

Gliederung

- Titel
- uCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Debugging mit ucLinux auf ARM-Prozessoren

- Thema der Diplomarbeit
- betreut durch Steffen Köhler

Gliederung

- Titel
- µClinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# µClinux

- µ für „micro“ und C für „Controller“
- sprich: „you-see-linux“
- Linuxportierung für Mikrocontroller ohne Memory Management Unit
- Open-Source-Projekt
- basiert auf Linux-Kernel

Gliederung

- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Geschichte

- erste Portierung auf Motorola MC68328 (1998/99)
- erstes Zielsystem: PalmPilot
- entstand aus Linux-2.0 Kernel
- heute: komplette Distribution, enthält Linux-2.6, Anwendungen, Bibliotheken und Tool-Chains

## Gliederung

- Titel
- $\mu$ CLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Vorteile

- komplette Funktionalität des aktuellen Linux-Kernels (Treiber, Dateisysteme, ROMFS, Konsole)
- geringere Imagegröße (< 300 KB)
- optimierte C und C++ Bibliotheken
- uneingeschränkter Systemzugriff für Anwendungen (auch Register)

Gliederung

- Titel
- $\mu$ CLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Unterschiede zu Linux

- betreffen Speicherverwaltung
- kein `fork()`, sondern `vfork()`  $\rightarrow$  Multitasking
- kein wachsender Stack  $\rightarrow$  `mmap()`
- kein Speicherschutz (protected memory)
- andere Bibliotheken

Gliederung

- Titel
- uCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Unterstützte Architekturen

- Motorola Dragonfire, Coldfire
- ARM7TDMI, (ARM9)
- ATARI
- Xilinx Microblaze
- T-Concept XI524 DSL
- T-Eumex 630 LAN/DSL
- µCsim

Gliederung

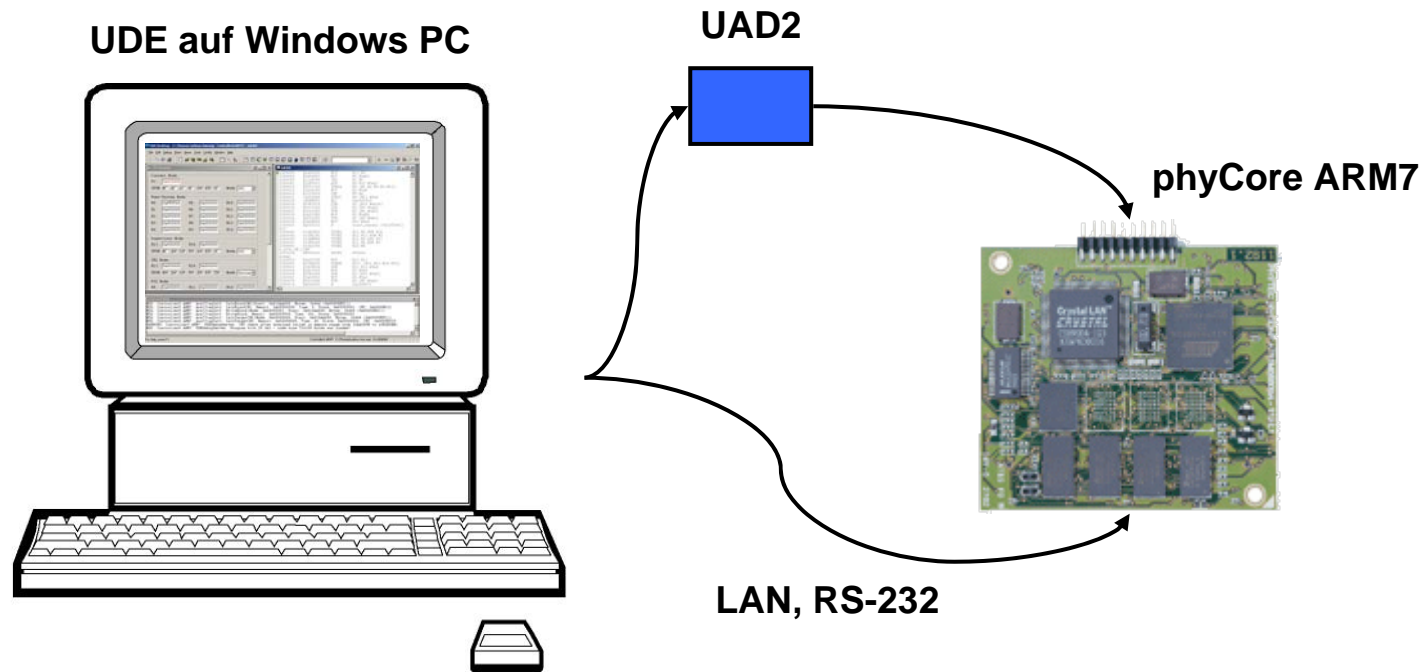
- Titel
- µClinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Ziel der Diplomarbeit

- lauffähiges µClinux auf Entwicklerboard installieren
- Verbindung mit UDE (Windows) herstellen (UAD2, RS232, LAN)
- Anwendungen unter µClinux ausführen und debuggen
- Kontrolle und Ausgabe über UDE

- Gliederung
- Titel
- $\mu$ CLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Ablauf





Gliederung

- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Bisheriger Stand

- Entwicklerboard phyCore AT91M55800A von Phytec
- angepasster µCLinux-Kernel auf Basis der Sourcen für Atmel AT91
- Download des Kernel als Image (.elf) auf Board über UDE und UAD/UAD2 funktioniert

Gliederung

- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Portierung und Anpassung

- Portierung unter Debian / Cygwin
- Installation einer aktuellen µCLinux-Distribution (15.12.2004)
- Installation einer Toolchain für ARM/.elf auf Basis von gcc
- Download aktueller Kernelsources von µCLinux/ARM2.6-Projekt

## Gliederung

- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Portierung und Anpassung

- Erstellen eigener Plattform auf Basis von Atmel's AT91
- Anpassen des Quellcodes
  - RS-232 Schnittstelle für Terminal
  - RAM- und Flashgrößen, Takt
  - eigener Interrupthandler
  - Treiber für Netzwerkschnittstelle

Gliederung

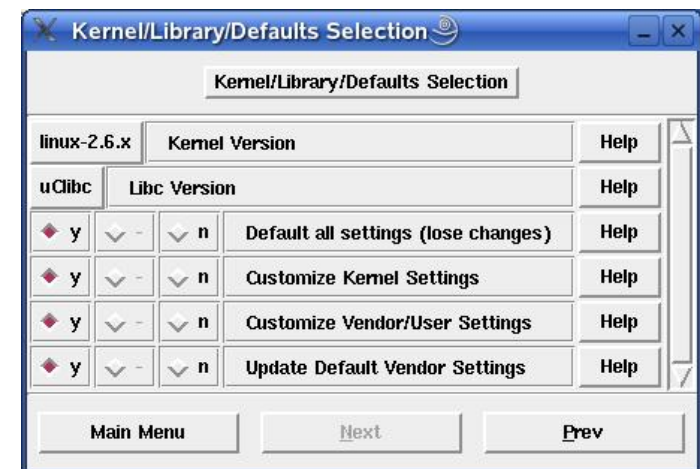
- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Portierung und Anpassung

- Quellen der „userland“-Umgebung anpassen
- Kernel konfigurieren und erstellen
- „userland“-Anwendungen erstellen
- Images in .elf-Format umwandeln und verknüpfen

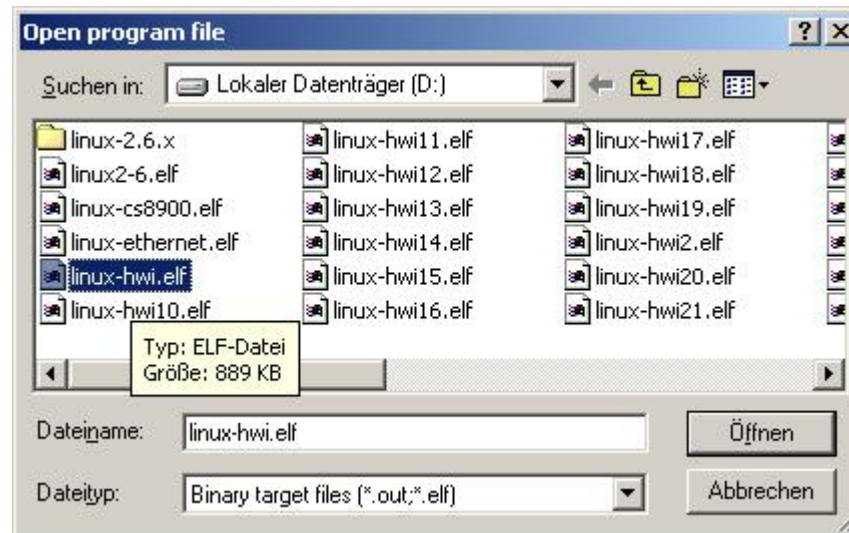
# Portierung und Anpassung

- Gliederung
- Titel
  - $\mu$ CLinux
  - Geschichte
  - Vorteile
  - Unterschiede
  - Architekturen
  - Ziele
  - Ablauf
  - Stand
  - Portierung
  - Test
  - Ausblick
  - Quellen



# Testen

- Download des erstellten Images auf das Versuchsboard per UDE



## Gliederung

- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

Gliederung

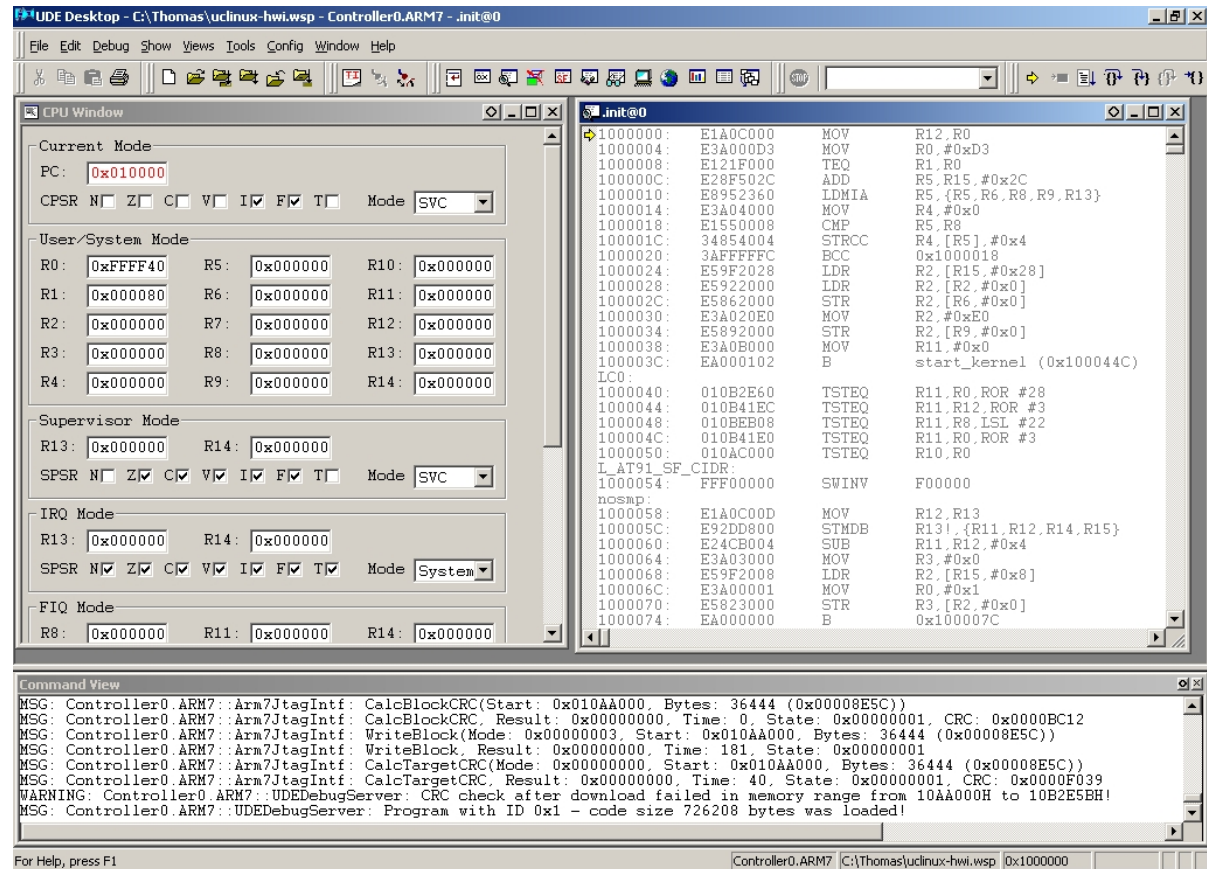
- Titel
- µClinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Testen

- Prozessor starten
- Kernel-Debugging per UDE
- bei Fehler Registerinhalte und Befehlszähler notieren, Speicherinhalte überprüfen
- Terminal gibt Kernelmeldungen aus
- Fehler beseitigen und neuen Kernel erstellen

# Testen

- Gliederung
- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen



The screenshot displays the uCLinux IDE Desktop interface. The main window is titled "Controller0.ARM7 - .init@0". It features a menu bar (File, Edit, Debug, Show, Views, Tools, Config, Window, Help) and a toolbar. The interface is divided into several panes:

- CPU Window:** Shows the current mode (SVC), PC (0x010000), and various registers (R0-R14) in both User/System and Supervisor modes. The Supervisor mode SPSR is also visible.
- Disassembly Window:** Displays assembly code for the ".init@0" section, including instructions like MOV, TEQ, ADD, LDHIA, CMP, STRCC, BCC, LDR, STR, and B. It shows addresses from 1000000 to 1000074.
- Command View:** Shows the output of various system messages, including CRC calculations and warnings. The messages indicate successful operations and a warning about a memory range (10AA000H to 10B2E5BH) that failed to load.

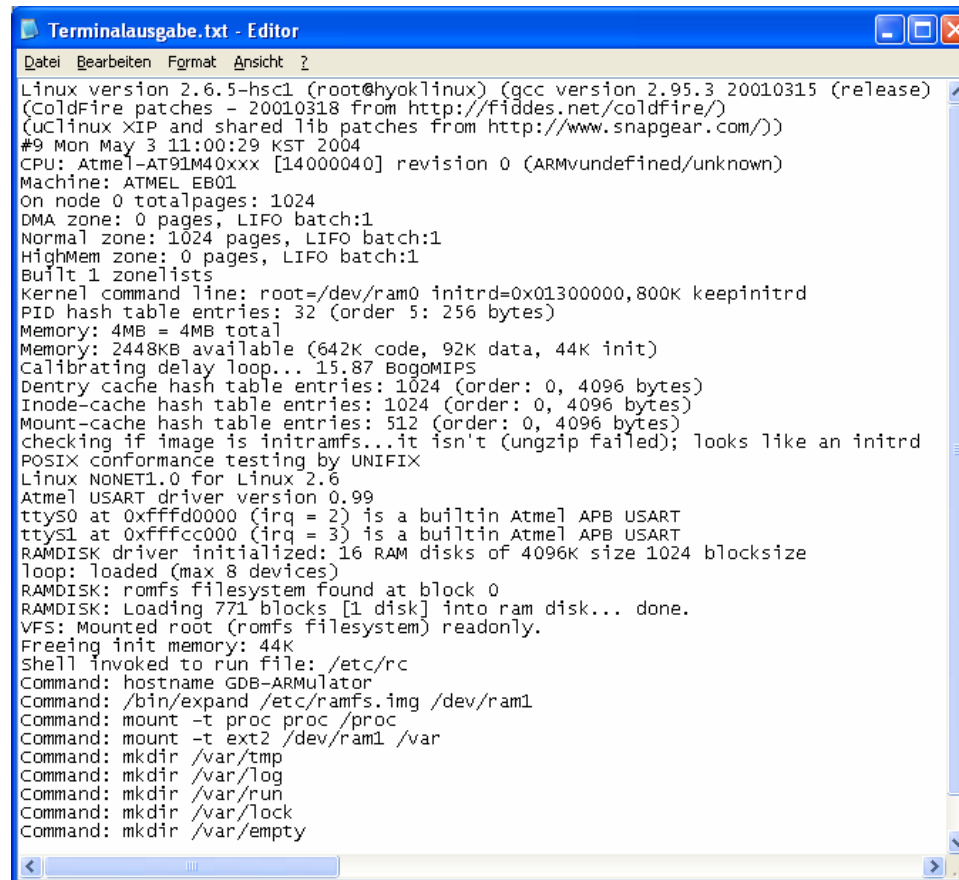
At the bottom, there is a status bar with the text "For Help, press F1" and "Controller0.ARM7 | C:\Thomas\uclinux-hwi.wsp | 0x1000000".



# Testen

## Gliederung

- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen



```
Terminalausgabe.txt - Editor
Datei Bearbeiten Format Ansicht ?
Linux version 2.6.5-hsc1 (root@hyoklinux) (gcc version 2.95.3 20010315 (release)
(ColdFire patches - 20010318 from http://fiddes.net/coldfire/)
(uclinux XIP and shared lib patches from http://www.snapgear.com/))
#9 Mon May 3 11:00:29 KST 2004
CPU: Atmel-AT91M40xxx [14000040] revision 0 (ARMvundefined/unknown)
Machine: ATMEL EB01
On node 0 totalpages: 1024
DMA zone: 0 pages, LIFO batch:1
Normal zone: 1024 pages, LIFO batch:1
HighMem zone: 0 pages, LIFO batch:1
Built 1 zonelists
kernel command line: root=/dev/ram0 initrd=0x01300000,800k keepinitrd
PID hash table entries: 32 (order 5: 256 bytes)
Memory: 4MB = 4MB total
Memory: 2448KB available (642K code, 92K data, 44K init)
Calibrating delay loop... 15.87 BogoMIPS
Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
checking if image is initramfs...it isn't (ungzip failed); looks like an initrd
POSIX conformance testing by UNIFIX
Linux NONET1.0 for Linux 2.6
Atmel USART driver version 0.99
ttyS0 at 0xffffd0000 (irq = 2) is a builtin Atmel APB USART
ttyS1 at 0xffffc0000 (irq = 3) is a builtin Atmel APB USART
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
RAMDISK: romfs filesystem found at block 0
RAMDISK: Loading 771 blocks [1 disk] into ram disk... done.
VFS: Mounted root (romfs filesystem) readonly.
Freeing init memory: 44K
Shell invoked to run file: /etc/rc
Command: hostname GDB-ARMulator
Command: /bin/expand /etc/ramfs.img /dev/ram1
Command: mount -t proc proc /proc
Command: mount -t ext2 /dev/ram1 /var
Command: mkdir /var/tmp
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/lock
Command: mkdir /var/empty
```

Gliederung

- Titel
- µClinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Ausblick

- Stabilität und Performance des µClinux verbessern
- aktuellen GNU Compiler und Debugger für µClinux erstellen und in „userland“-Image einbinden
- Netzwerktreiber verbessern
- Netzwerkkommunikation erweitern

Gliederung

- Titel
- µCLinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Ausblick

- Terminal in UDE-Oberfläche integrieren
- Download einer µCLinux-Anwendung per Netzwerk
- Debuggermeldungen in UDE darstellen bzw. verwerten

Gliederung

- Titel
- µClinux
- Geschichte
- Vorteile
- Unterschiede
- Architekturen
- Ziele
- Ablauf
- Stand
- Portierung
- Test
- Ausblick
- Quellen

# Quellen

- <http://www.uclinux.org>
- <http://www.ucdot.org>
- µClinux/ARM2.6-Projekt betreut von Hyok S. Choi  
<http://opensrc.sec.samsung.com>
- <http://www.phytec.com>