µClinux on a ARM Cortex-M4!?

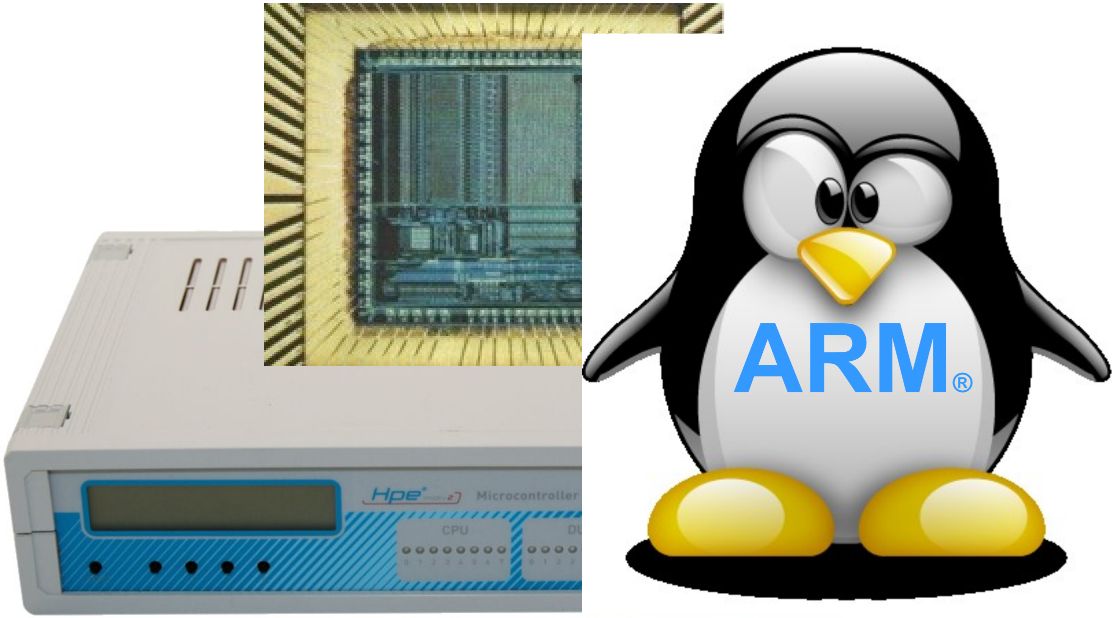
(**a** cost-benefit analysis)

DOULOS

DOULOS

University of Applied Sciences
Fachhochschule EMDEN·LEER

APPROVED TRAINING CENTRE ARM



Master Thesis at University of Applied Sciences (Hochschule Emden/Leer)
Supported by Doulos Ltd.

Student	Frank Mölendörp
Supervisor (Doulos)	Jens Stapelfeldt
Supervisor (University Emden/Leer)	Prof. Dr. –Ing. Gerd von Cölln

Content

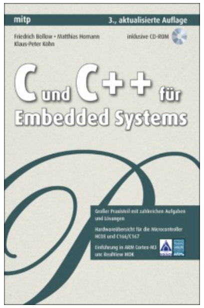
- What is MPS?
- What is uClinux?
- Purpose of this work
- What was archieved?
- Interesting facts
- Demonstration
- Cost-benefit-analysis
- Conclusion

Doulos and Standards

Doulos is a Training and Know-How provider in electronics for over 18 years and has been involved with standards form day 1

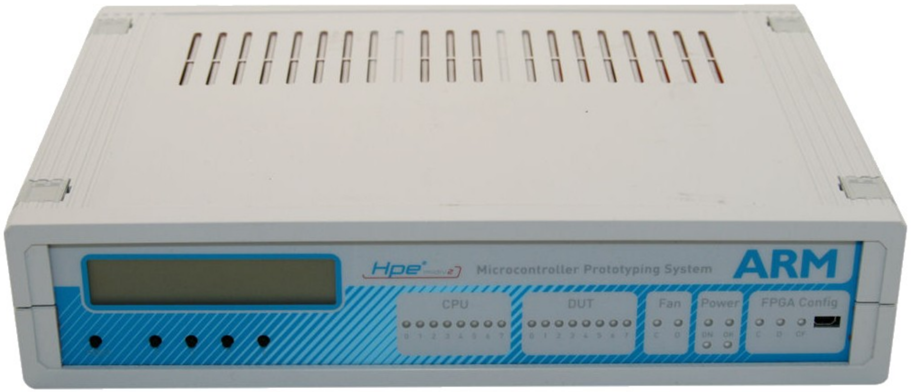
VHDL, SystemC, SystemVerilog, C++, OVM……, ARM, CMSIS

- Doulos is has delivered Training to over 800 companies and in 36 countries world-wide.
- Public classes are scheduled regularly in Europe and the USA
- Doulos has been an ATC (Approved Training Centre) for over 10 years
  - Free resources in our Know-How section on the web
  - See www.doulos.com/knowhow
    - Tutorial: Getting started with CMSIS
    - German book with nice Cortex-M3 Intro and Examples !



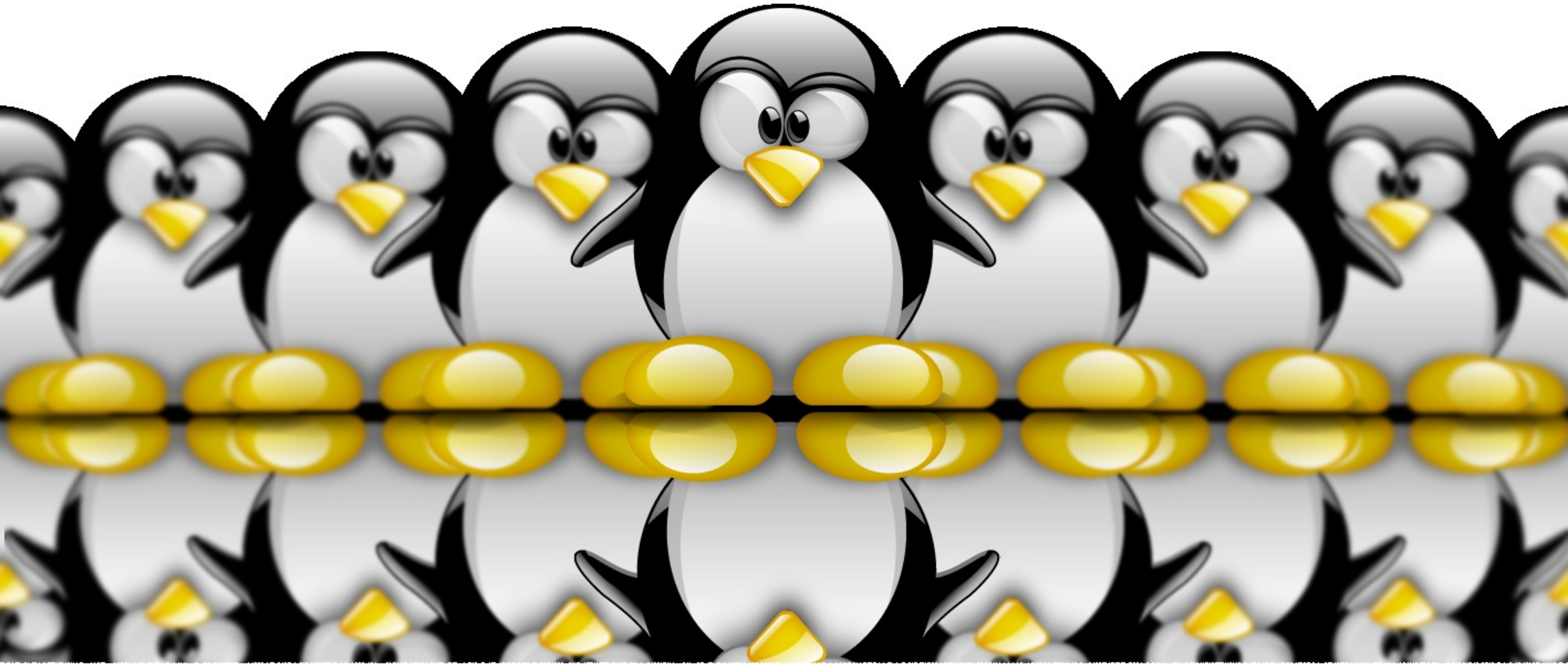
What is the MPS?

- **M**icrocontroller **P**rototyping **S**ystem
- FPGA based rapid prototyping system, which emulates ARM Cortex-M microcontrollers



What is the MPS?

- offers a variety of interfaces, and I/O devices
  - Ethernet, USB (host/device), UART, DVI, Flexray, CLCD, etc.
  - push buttons, LED
- includes 8 MB RAM and 64 MB Flash



Linux is on the rise!

What is uClinux?

- Linux distribution for microcontroller
- Slightly modified standard-Kernel
- Does not require MMU
  - No virtual memory
- Smaller memory footprint than standard kernel
- Includes special features to support embedded systems
  - e.g. XIP (eXecute In Place)

Purpose of this work

- Academic approach to determine whether uClinux on a Cortex-M4 makes any sense
- To use uClinux at the microcontroller level
- Integrate the features of Cortex-M4 (FPU, SIMD) into uClinux
  - starting with Cortex-M3 variant of uClinux

The purpose of this work

- Using vanilla distribution
  - requires bootloader
  - ARM-provided standalone-kernel not used here
    - needs RV Debugger scripts to load image
    - more of a proof-of-concept solution
- Demonstrate Example application

What has been achieved?

- Executable Kernel with Initramfs
- Use of the uClinux distribution and the C-library uClibc
- Minimal example application was added to the distribution

uClinux vs. others

uClinux

- Easily extendable
- Driver support
- Console integrated
- File system integrated
- High memory requirements
  - compared to RTOS
  - requires external memory
- Big community
- Use of standard kernel

Realtime Operating Systems

- Easily configurable
- No direct driver support
- No console
- Comes without file system
- Low to medium memory-footprint
- Medium community
- Independent OS

Cost-benefit-analysis

Cost

- Porting kernel (400 to 800 man-hours)
- Porting distribution (300 to 500 man-hours)
- Implement own software

Benefits

- Usage of file system, network, USB
- Usage of varity tools
- Use own functionality

Cost-benefit-analysis

- uClinux offers plenty communication options
- RAM and Flash usages require external memory
  - additional peripherals needed
- High development effort needed if architecture is not already supported

Interesting facts

- Busybox Applet mechanism

  first argument is the name of the command
- only UART 2 has RTS-control flow
  - MPS limitation
- uClinux uses bFLT instead of ELF
- Bootloader required

System Requirements

- Minimum of 8 MB RAM and 4 MB Flash
  - External memory required (in 2010)
  - Increases total system cost
  - kernel reserves 2.5 MB for itself
  - mounting FS requires minimum 0.5 MB
- System frequency of 50+ MHz

Memory Options (normal)

- Kernel and applications will be loaded from Flash into RAM
  - Requires larger RAM
  - Copying image consumes time during system start-up
- Might increase execution performance
  - In case of Flash wait states and no cache

Memory Options (XIP)

- eXecute In Place
  - Kernel will be executed from Flash
  - Applications still executed from RAM
- Reduces RAM usage by up to 4 MB
- Accelerates the initialization process
- Might *reduce* execution performance
  - In case of Flash wait states and no cache

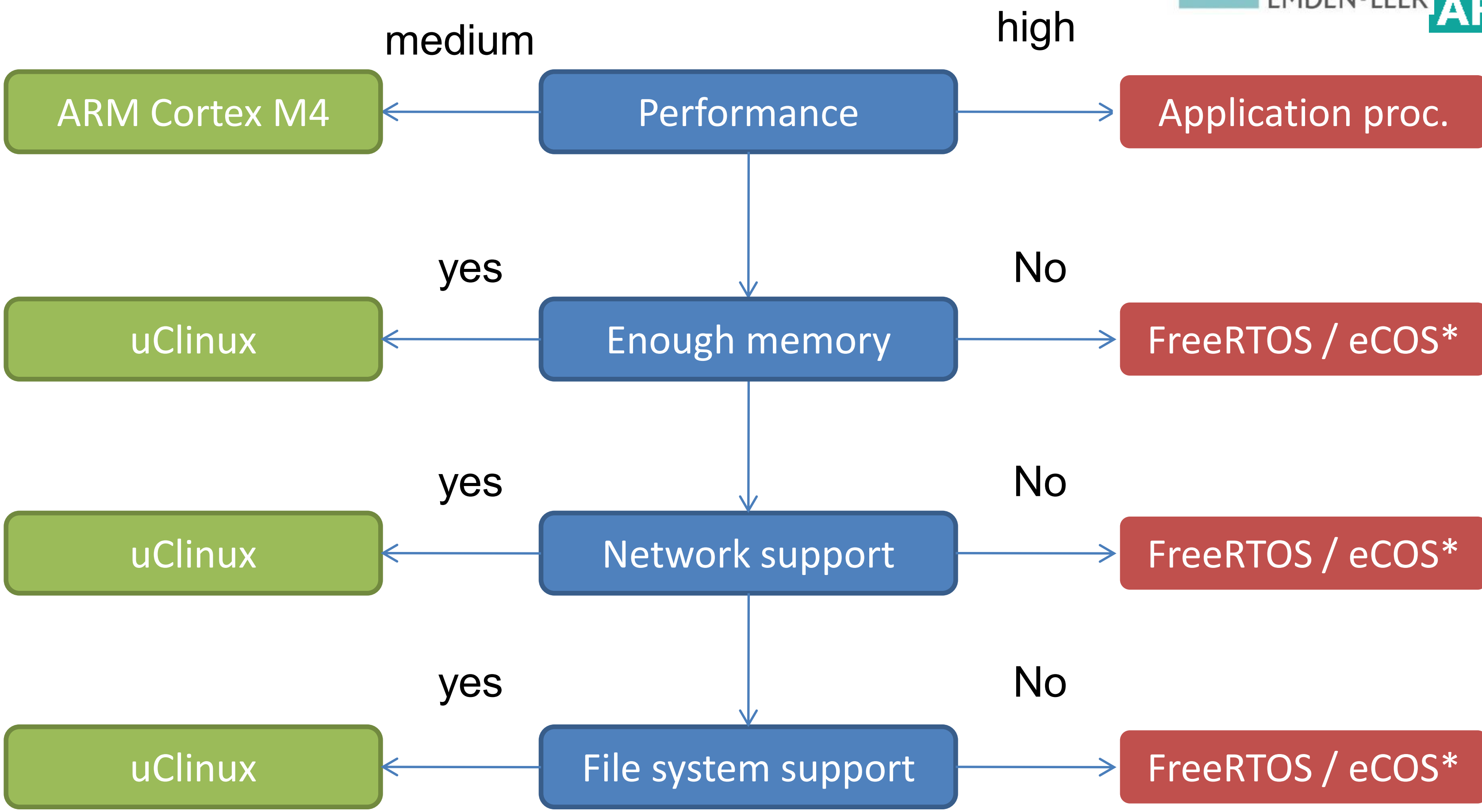
Normal configuration vs. XIP

Normal configuration	Address	XIP
Interrupt-vector	0x00000000	Interrupt-vector
...		...
Linux kernel start parameter	0x10000000	Linux kernel start parameter
Linux-Image RAM (Initramfs-Image)	0x10000200	Initramfs
	RAM 4 MB Bank 1	Free RAM
Initramfs		
...	0x10400000	...
Bootloader	0x18000000	Bootloader
Linux-Image Flash	0x18800000	Linux-Image Flash
	Flash 64 MB	
...	0x1B000000	...
Free RAM	0x20000000 RAM 4 MB Bank 2	Free RAM
...	0x20400000	...

When to consider uClinux?

- Low-cost products with multitasking requirements
- Low power systems that need file system and network support
- Systems that must be extensible

When to consider uClinux?



* Beispiele für andere (Echtzeit-)Betriebssysteme

Effort

- The hardest work is to understand the kernel build architecture

- Overall, with a time of 1000 man-hours expected.

Demonstration

- PuTTY is used as console
- Two serial connections are needed
  - The first interacts with the Bootloader
  - The second communicates with the Linux-console
- Compilation is made in a Linux-environment
  - The kernel build system uses symbolic links which don't exist in NTFS

Further work

- Full ROMFS integration
  - Flash devices currently not visible through UDEV
- Activation and testing of system interfaces
  - Ethernet, USB, MMC, Sound
- Bootloader extensions
  - Enabling additional transfer channels (JTAG, TCP/IP)

Conclusion

- Linux can be useful even on small microcontroller devices
- limited, manageable development overhead
- small runtime overhead
- requires external memory interface for Flash/RAM
  - additional costs justified if applications take advantage of uClinux features

Training & Support options

APPROVED TRAINING CENTRE ARM

University of Applied Sciences Fachhochschule EMDEN•LEER

- **ARM Training options include:**
  - ARM Cortex-M Embedded Software Workshop (4 days)
  - ARM Cortex-R4 Embedded Software Workshop (4 days)
  - RapidGain™ Advanced Debug for ARM Cortex-M
    - One day events with partners !

RAPIDGAIN Wissen kompakt

- **Embedded Software classes:**
  - Embedded C for real time applications
  - Embedded Linux
  - Fundamentals of RTOS
- More information about Doulos ARM trainings at www.doulos.com/arm

References

- Technical Documentation
  - ARM Cortex-M4 Technical Reference Manual (DDI 0439C)
  - ARM v7-M Architecture Reference Manual (DDI 0403D)
- Textbooks
  - The Definitive Guide to the ARM Cortex-M3
    - ISBN: 978-0-7506-8534-4
  - C und C++ für Embedded Systems
    - First German ARM Cortex-M Introduction by Doulos ARM Experts
    - ISBN: 978-3-8266-5949-2

ARM Cortex-M(3) embedded software

	Day 1	Day 2	Day 3	Day 4
9am	The ARM Architecture	Thumb-2 Instruction Sets	Embedded SW Development	Cortex-M3 MPU *
	Cortex-M3 Introduction & Processor core	Migrating Legacy ARM/Thumb code to Cortex-M3	Compiler Hints and Tips	CoreSight Debug Architecture Overview
Lunch				
	RealView Overview	Cortex-M3 Interrupts	Cortex-M3 Memory Types	Invasive & Non-invasive Debug
		Cortex-M3 Exception Handling		
5pm	**RealView Introductory Workbook**	**Cortex-M3 Embedded SW-Workbook**	**Embedded MCU SW Workbook**	**Embedded MCU SW Workbook**

Concluding slide

CONTACT

Frank Mölendörp

Junior Consultant

+49 511 277-1340

Frank.Moelendoerp@doulos.com

Doulos Central European Office
Garbsener Landstrasse 10
30149 Hannover
Deutschland