LINUTRONIX
LINUX FOR INDUSTRY

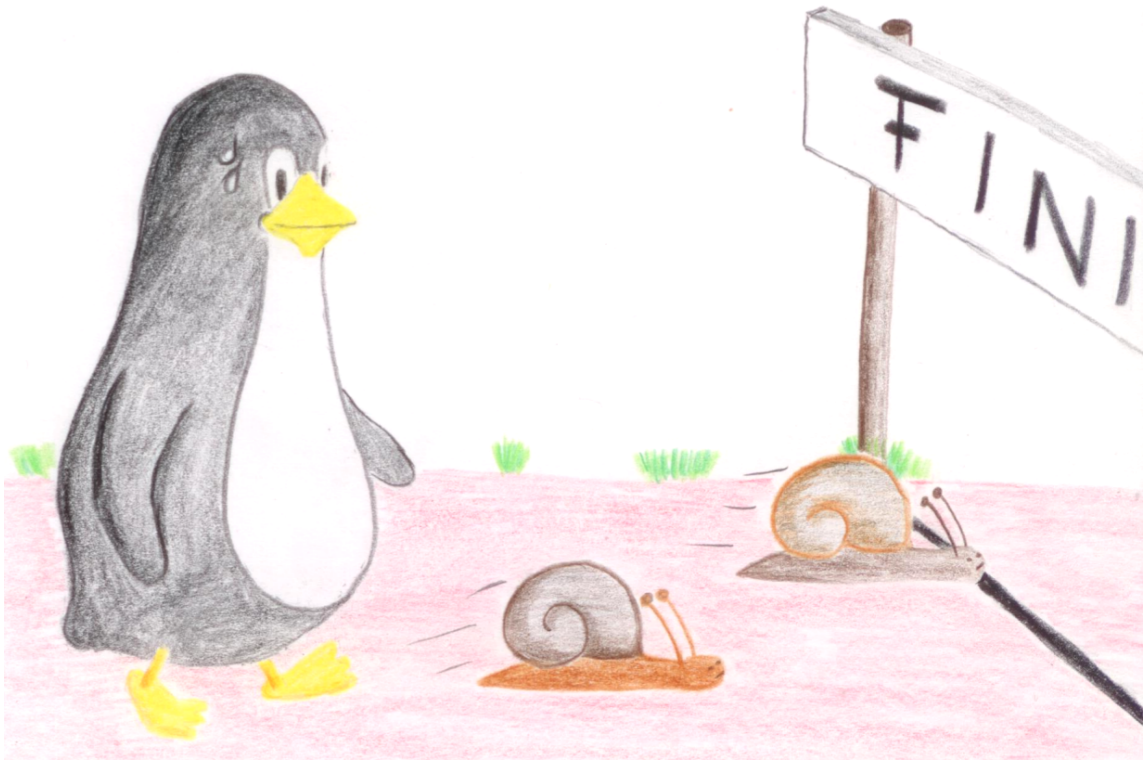
How to boot Linux in one second

...why userland is a waste of time ;)

Jan Altenberg

Linutronix GmbH

from zero...



to hero...

LINUTRONIX
LINUX FOR INDUSTRY

Overview

1. **Basics**
   - Motivation
   - Some technical basics
2. **Optimizations**
   - Bootloader
   - Kernel
   - Filesystem
   - Application
3. **Example**
   - Optimizing an ARMv5 based device
   - Optimizing the test system

LINUTRONIX
LINUX FOR INDUSTRY

LINUTRONIX
LINUX FOR INDUSTRY

Motivation

- "marketing"
- automotive applications
- energy saving

LINUTRONIX
LINUX FOR INDUSTRY

Motivation

- "marketing"
- automotive applications
- energy saving
  - **solution:** power-off instead of suspending

LINUTRONIX
LINUX FOR INDUSTRY

Motivation

- "marketing"
- automotive applications
- energy saving
  - **solution:** power-off instead of suspending
  - **BUT:** Users are not used to wait

LINUTRONIX
LINUX FOR INDUSTRY

LINUTRONIX
LINUX FOR INDUSTRY

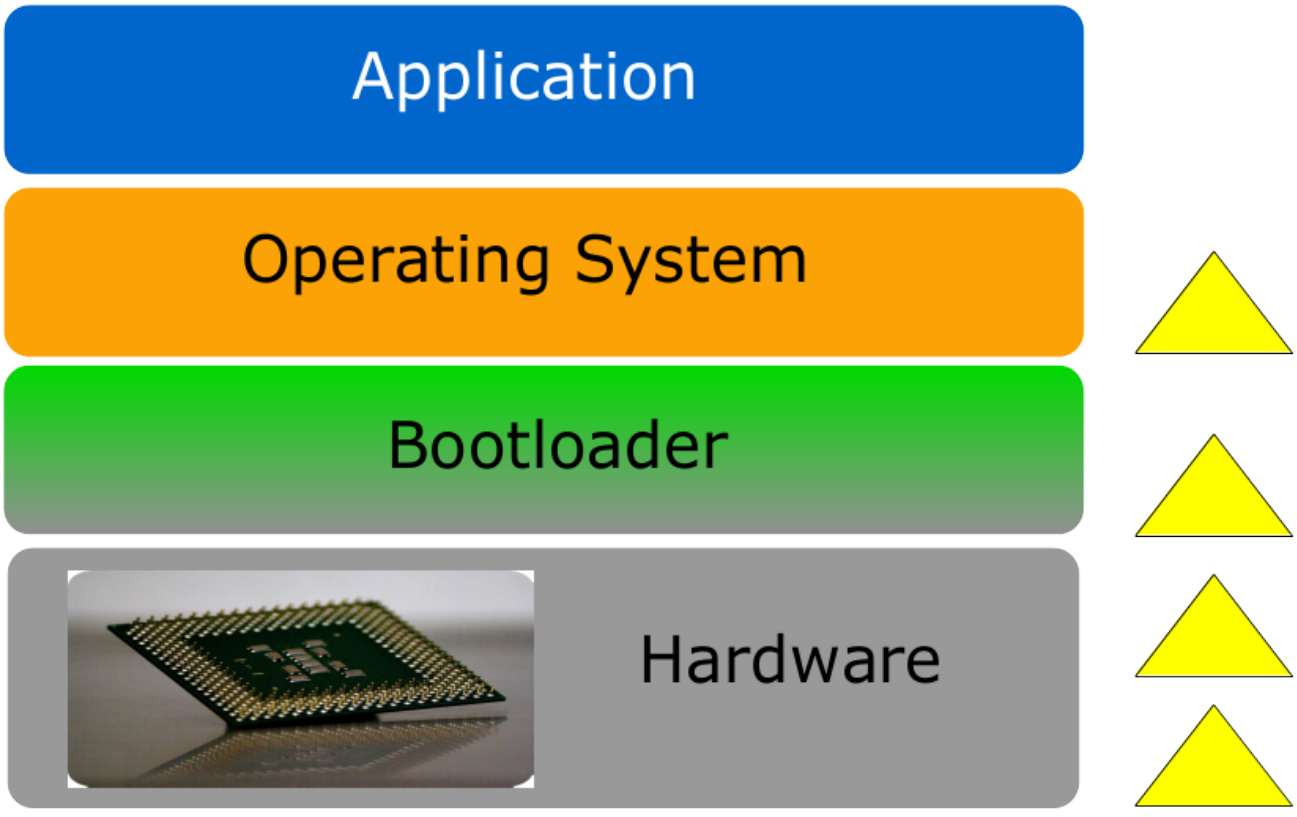
First step: Define your requirements!!!!

- What's the limit for the boot time?
- Which functionality should be available?
- Speed vs. flexibility

NOTE: FastBOOT is not a product, it's a concept!!

LINUTRONIX
LINUX FOR INDUSTRY

Boot process



LINUTRONIX
LINUX FOR INDUSTRY

Components of the boot process

- Hardware reset
- Bootloader
- Operating System (drivers, filesystem, ...))
- INIT process, application (userland)

LINUTRONIX
LINUX FOR INDUSTRY

Critical hardware components

- Power supply
- Reset logic
- Boot logic / boot order
- Boot media
- Peripherals which need to be accessed while booting

IMPORTANT: the hardware is a central part of a fastboot concept!!!

LINUTRONIX
LINUX FOR INDUSTRY

Bootloader

- Basic setup of the CPU
- Preparing and handing over ATAGS / devicetree
- Flushing the caches
- Switch off the MMU

LINUTRONIX
LINUX FOR INDUSTRY

The Linux Kernel

- A lot of functions for boot time optimization
- Very flexible
- Configurable compression type
- Can deffer or parallelize initializations
- 150ms - 250ms from starting the kernel to mounting the RFS

LINUTRONIX
LINUX FOR INDUSTRY

The application

- Usually the biggest target for optimizations
- Start scripts / INIT process
- Linking

LINUTRONIX
LINUX FOR INDUSTRY

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the bootloader (U-Boot) 1

Remove unused features:

```
/* include/configs/boardname.h */
[...]
#include <config_cmd_default.h>
#undef CONFIG_CMD_NET
[...]
```

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the bootloader (U-Boot) 2

Verifying the kernel image:

```
setenv verify n
```

Switch off the bootloader console:

```
setenv silent 1
```

Switch off the boot delay:

```
setenv bootdelay 0
```

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the bootloader: IPL / SPL

- Replacing the general purpose bootloader by an optimized IPL
- ...also useful for update concepts
- U-Boot offers a generic way: The U-Boot SPL (CONFIG_SPL_OS_BOOT)

LINUTRONIX
LINUX FOR INDUSTRY

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the kernel

- Configuration and build
- Compression method
- Boot parameters (kernel commandline)
- Driver init calls
- Rootfilesystem (RFS)

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the kernel: Configuration

```
General setup  --->
Kernel compression mode -->
```

- LZO usually a good choice for embedded system
- Copy vs. de-compress
- "Execute in Place (XIP)'

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the kernel: Kernel commandline

- Delay Loop Calibration: "lpj="; can save > 100ms on ARMv5 based systems
- Parameters for boot time analysis: "initcall_debug", "printk_time=1"

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the kernel: printk.time

```
...
[0.000000] VIC @f1140000: id 0x00041190,
           vendor 0x41
[0.000000] FPGA IRQ chip 0 "SIC" @ f1003000,
           21 irqs
[0.000000] Console: colour dummy device 80x30
[0.018847] Calibrating delay loop...
           626.68 BogoMIPS (lpj=3133440)
[0.316717] pid_max: default: 32768 minimum: 301
[0.317552] Mount-cache hash table entries: 512
...
```

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the kernel: Delay Loop

```
...
[0.018847] Calibrating delay loop...
           626.68 BogoMIPS (lpj=3133440)
[0.316717] pid_max: default: 32768 minimum: 301
...
```

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the kernel: initcall_debug

```
[0.452115] calling  exceptions_init+0x0/0x90 @ 1
[0.452172] initcall exceptions_init+0x0/0x90
               returned 0 after 0 usecs
[0.452203] calling  versatile_i2c_init+0x0/0x24 @ 1
[0.452321] initcall versatile_i2c_init+0x0/0x24
               returned 0 after 0 usecs
[0.452352] calling  pl011_init+0x0/0x54 @ 1
[0.452382] Serial: AMBA PL011 UART driver
[0.453647] dev:f1: ttyAMA0 at MMIO 0x101f1000
             (irq = 12) is a PL011 rev1
[0.481540] console [ttyAMA0] enabled
...
[0.484427] initcall pl011_init+0x0/0x54
               returned 0 after 29296 usecs
```

LINUTRONIX
LINUX FOR INDUSTRY

bootgraph.pl

1. **Boot your system with "initcall_debug loglevel=8**
2. **On the target:**

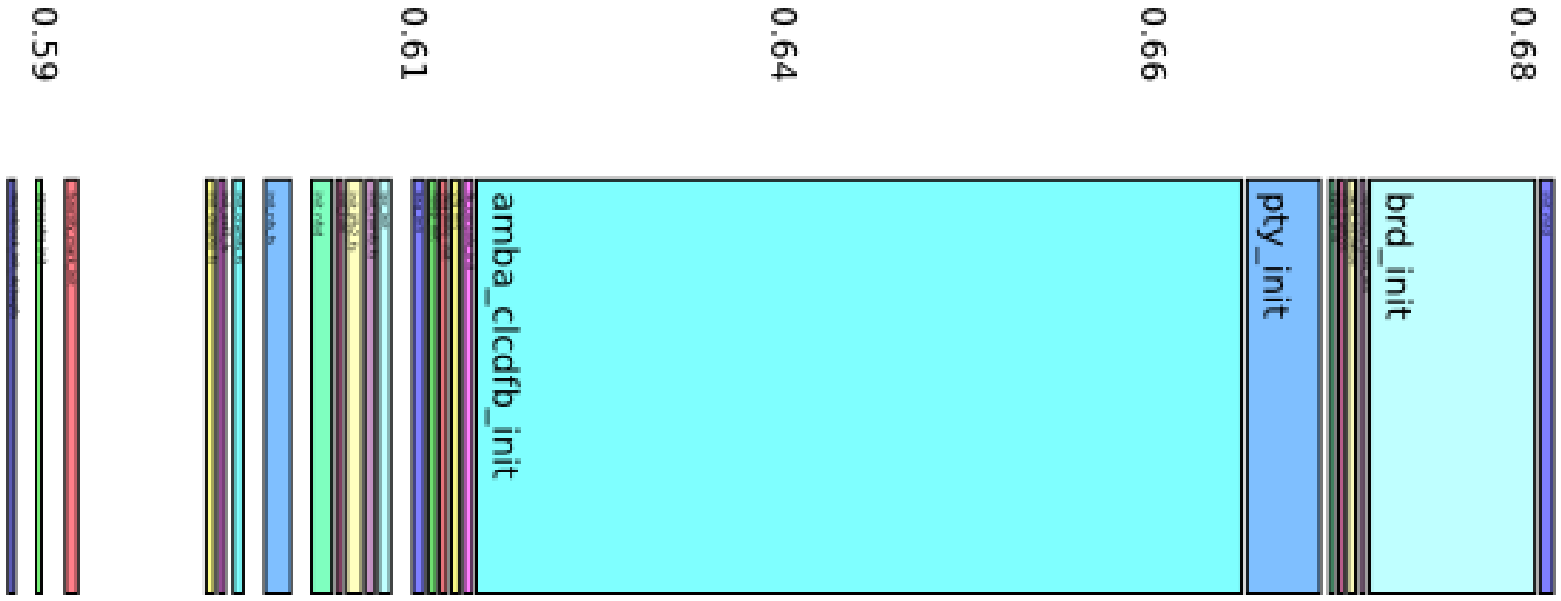
```
$ dmesg > bootlog.txt
```

3. **On the host:**

```
$ cd linux-XXX
$ cat /path_to_rfs/bootlog.txt | \
   perl scripts/bootgraph.pl > bootlog.svg
```

LINUTRONIX
LINUX FOR INDUSTRY

scripts/bootchart.pl



LINUTRONIX
LINUX FOR INDUSTRY

LINUTRONIX
LINUX FOR INDUSTRY

UbiFS

- The best choice for flash devices
- Power-Fail safe
- The underlying UBI layer can be optimized with (FastMAP)
- ...

LINUTRONIX
LINUX FOR INDUSTRY

InitRAMFS

```
dir /dev 755 0 0
nod /dev/console 644 0 0 c 5 1
nod /dev/loop0 644 0 0 b 7 0
dir /bin 755 1000 1000
slink /bin/sh busybox 777 0 0
file /bin/busybox initfs/busybox 755 0 0
[...]
dir /proc 755 0 0
dir /sys 755 0 0
dir /mnt 755 0 0
```

LINUTRONIX
LINUX FOR INDUSTRY

InitRAMFS: Switch root

The INIT process for the InitRAMFS can be configured with rdinit=. For example: rdinit=/etc/init.d/start.sh

LINUTRONIX
LINUX FOR INDUSTRY

InitRAMFS: Switch root

/etc/init.d/start.sh:

```
#!/bin/sh
mount -t proc proc /proc
mount -t sysfs sysfs /sys
mount -t devtmpfs devtmpfs /dev

# Mount RFS / do some critical stuff
mount /dev/mmcblk0p1 /media
fbsplash -s /media/splash.ppm -d /dev/fb0

mount -o move /proc /media/proc
mount -o move /sys /media/sys
mount -o move /dev /media/dev

# Switch to production system
exec switch_root /media /linuxrc
```

LINUTRONIX
LINUX FOR INDUSTRY

LINUTRONIX
LINUX FOR INDUSTRY

The INIT process

- SystemV
- SystemD

One letter makes a BIG difference ;-)

LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the application

- Analyse the INIT process with bootchartd or systemd-analyze
- Replace the INIT process with your own application (init=)
- Linking
- Pre-Linking and function reordering

LINUTRONIX
LINUX FOR INDUSTRY

Moving start script tasks into your application

```
ret = mount("sysfs", "/sys",
            "sysfs", 0, NULL);

if(ret < 0)
      perror("Can't mount sysfs\n");
```

LINUTRONIX
LINUX FOR INDUSTRY

Dynamic linking

1. ELF DT_RPATH section
2. LD_LIBRARY_PATH
3. ELF DT_RUNPATH section
4. Binary file /etc/ld.so.cache
5. Default paths /lib und /usr/lib

LINUTRONIX
LINUX FOR INDUSTRY

Dynamic linking: Debug and visualize

```
$ LD_DEBUG=libs ls
   3082: find library=librt.so.1 [0];
         searching
   3082: search cache=/etc/ld.so.cache
   3082: trying file=/lib/librt.so.1
```

LINUTRONIX
LINUX FOR INDUSTRY

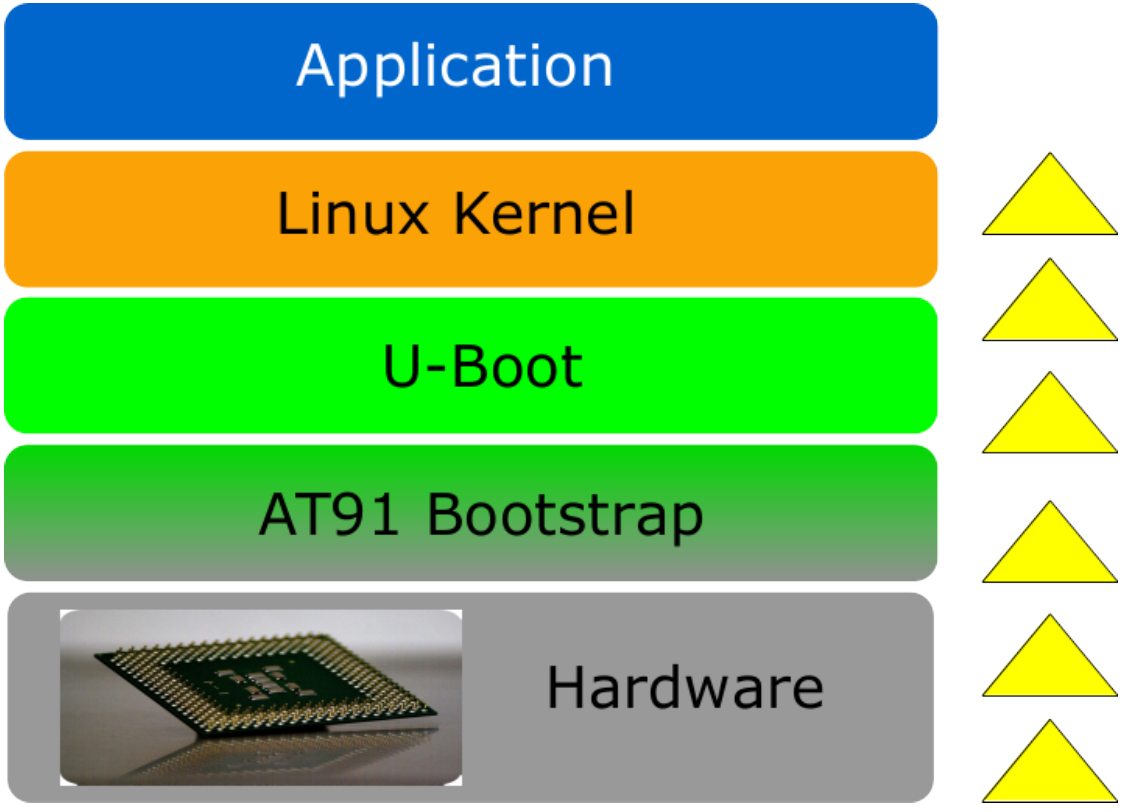
LINUTRONIX
LINUX FOR INDUSTRY

Test system

- ARM9 CPU (Atmel AT91 series)
- Starting point: Busybox based image (Angstrom Distribution)
- Boot media: NAND-Flash
- Test application: Toggling a GPIO via SysFS

LINUTRONIX
LINUX FOR INDUSTRY

Boot strategy of the AT91 controller family



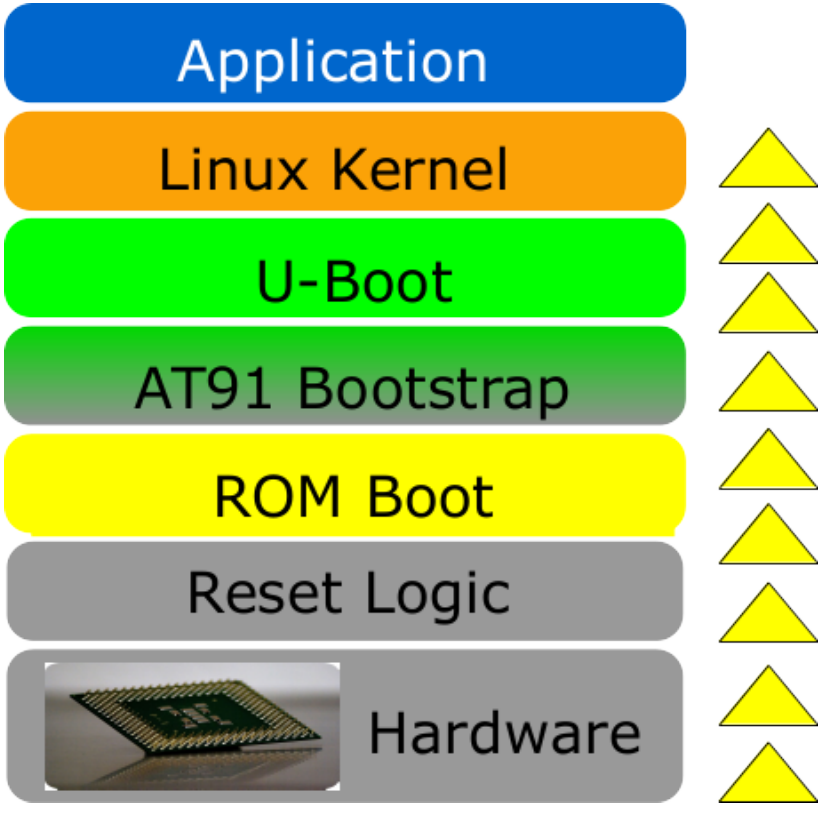
LINUTRONIX
LINUX FOR INDUSTRY

Bootmodes of the AT91 controller family

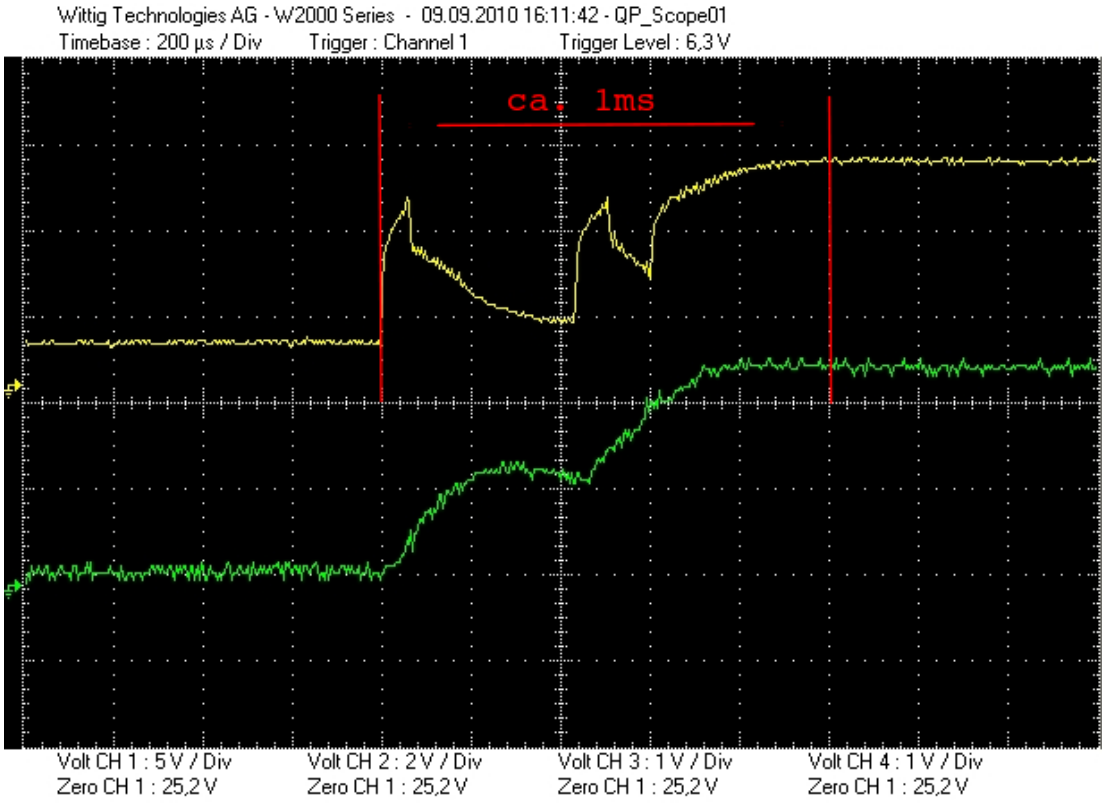
- RomBOOT: internal boot logic
- External bus interface (CS0, e.g. NOR flash)

LINUTRONIX
LINUX FOR INDUSTRY

AT91 RomBOOT

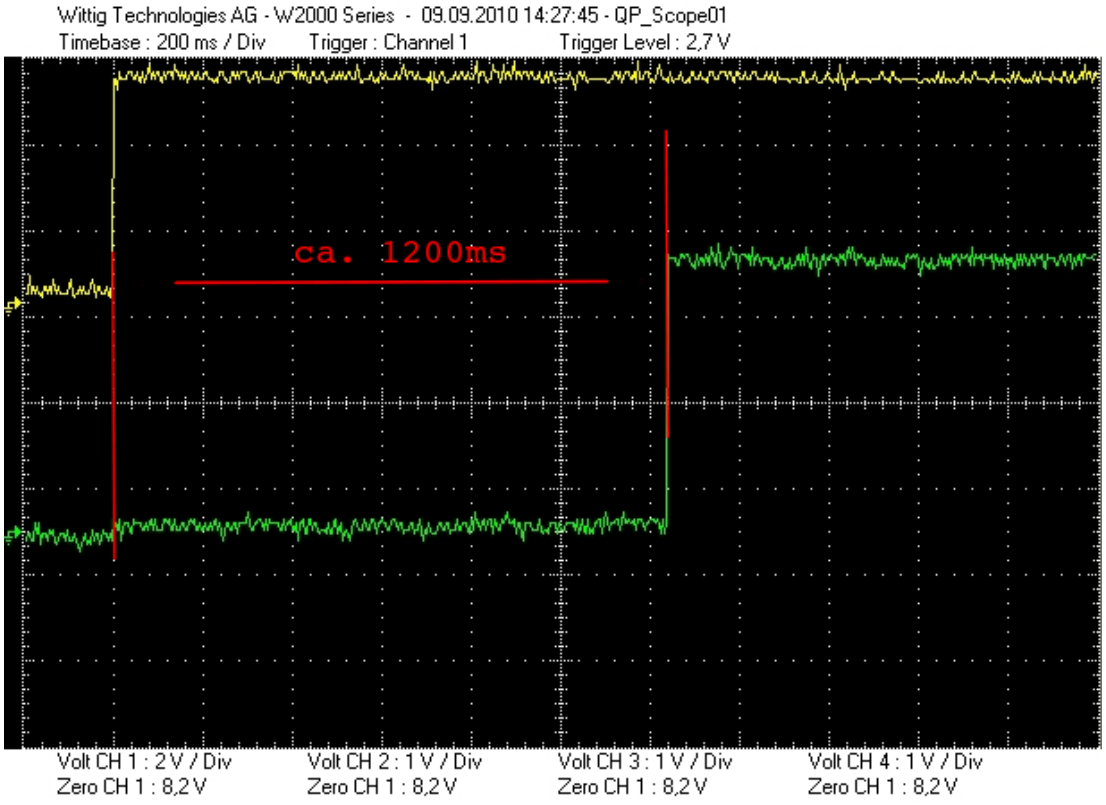


Power supply



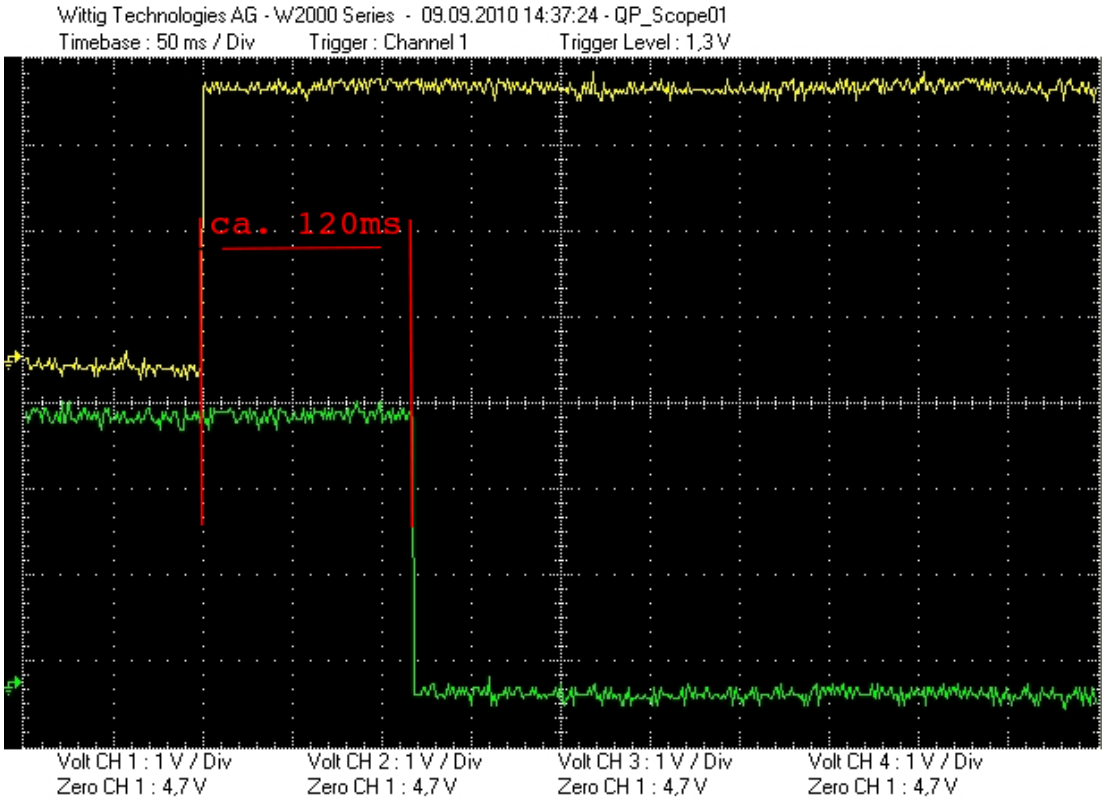
LINUTRONIX
LINUX FOR INDUSTRY

Reset logic



LINUTRONIX
LINUX FOR INDUSTRY

RomBOOT



LINUTRONIX
LINUX FOR INDUSTRY

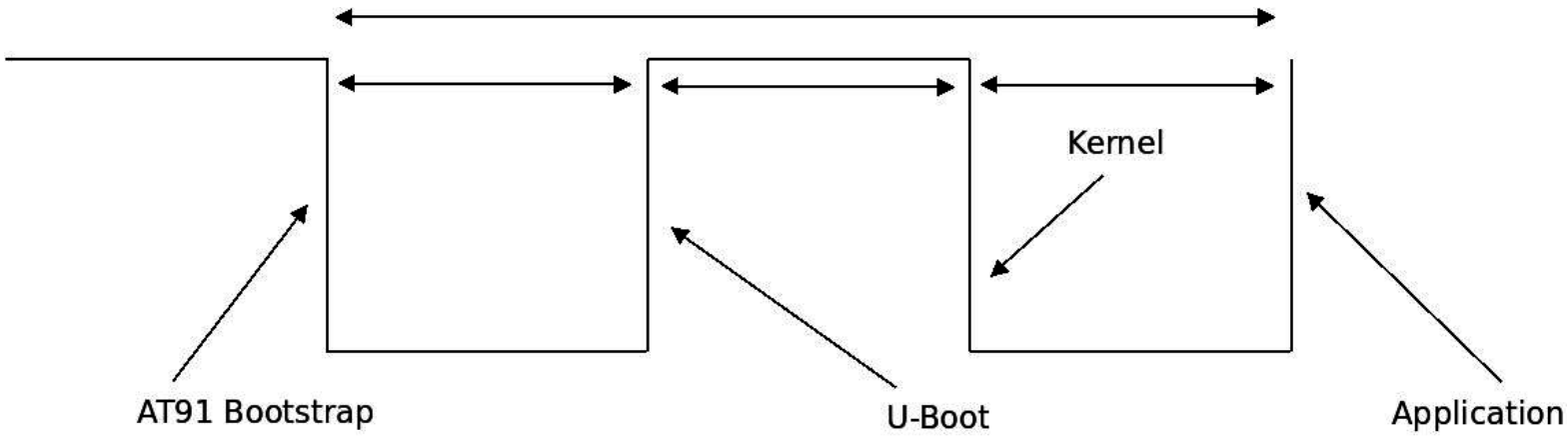
Possible hardware optimizations

- Using the internal oscillator for deriving the slowclock saves > 1s!!
- booting from CS0 will save 100 - 150ms

LINUTRONIX
LINUX FOR INDUSTRY

LINUTRONIX
LINUX FOR INDUSTRY

Boot time measurements with a GPIO



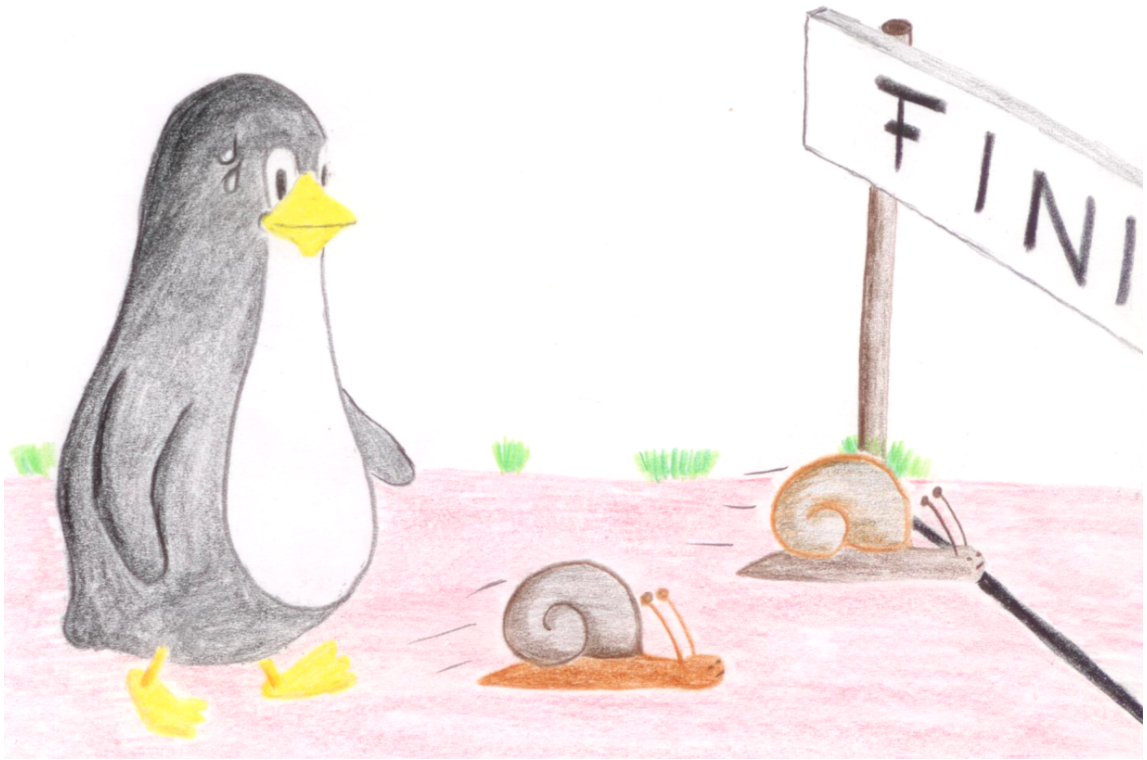
LINUTRONIX
LINUX FOR INDUSTRY

Measuring points

- Bootstrap - U-Boot
- U-Boot - Early-Boot-Code of the kernel (incl. relocation and decompression)
- Kernel - application (incl. mounting the RFS)

LINUTRONIX
LINUX FOR INDUSTRY

Initial boot time



LINUTRONIX
LINUX FOR INDUSTRY

Initial boot time

measuring point	time
bootstrap - u-boot	---
u-boot - kernel	6,5s
kernel - application	4,5s
total	**11s**

LINUTRONIX
LINUX FOR INDUSTRY

Simple optimizations

LINUTRONIX
LINUX FOR INDUSTRY

U-Boot w/o networking support

measuring point	time
bootstrap - u-boot	---
u-boot - kernel	4,25s
kernel - application	4,5s
total	**8,75s**

LINUTRONIX
LINUX FOR INDUSTRY

U-Boot verify=n

measuring point	time
bootstrap - u-boot	---
u-boot - kernel	3,89s
kernel - application	4,5s
total	**8,39s**

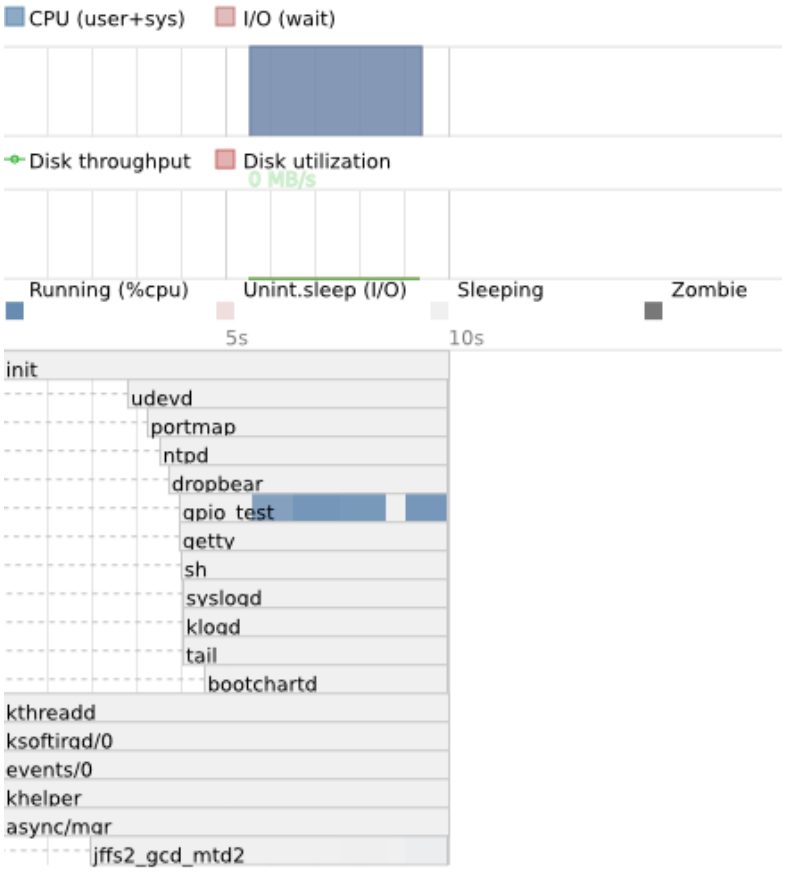
LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the kernel config

measuring point	time
bootstrap - u-boot	---
u-boot - kernel	3,77s
kernel - application	4,33s
total	**8,1s**

LINUTRONIX
LINUX FOR INDUSTRY

Analyzing the INIT process: Bootchartd



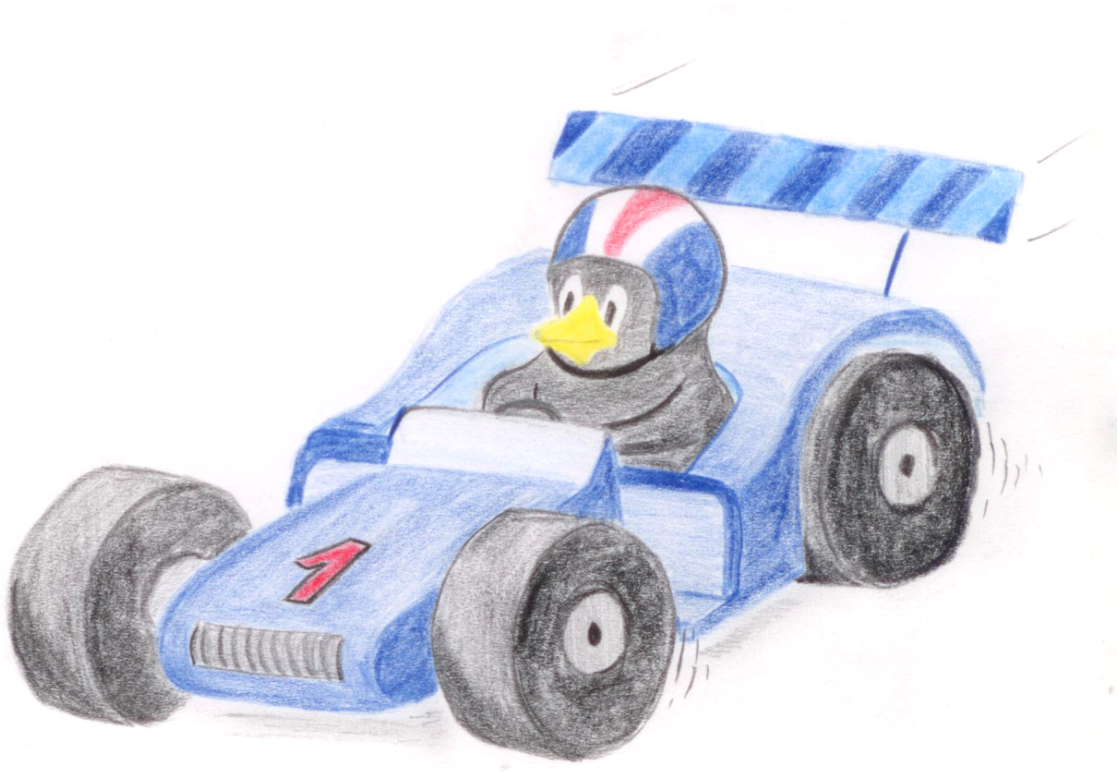
LINUTRONIX
LINUX FOR INDUSTRY

Optimizing the start scripts

measuring point	time
bootstrap - u-boot	---
u-boot - kernel	3,77s
kernel - application	3,61
total	**7,38s**

LINUTRONIX
LINUX FOR INDUSTRY

Booting an InitRAMFS



LINUTRONIX
LINUX FOR INDUSTRY

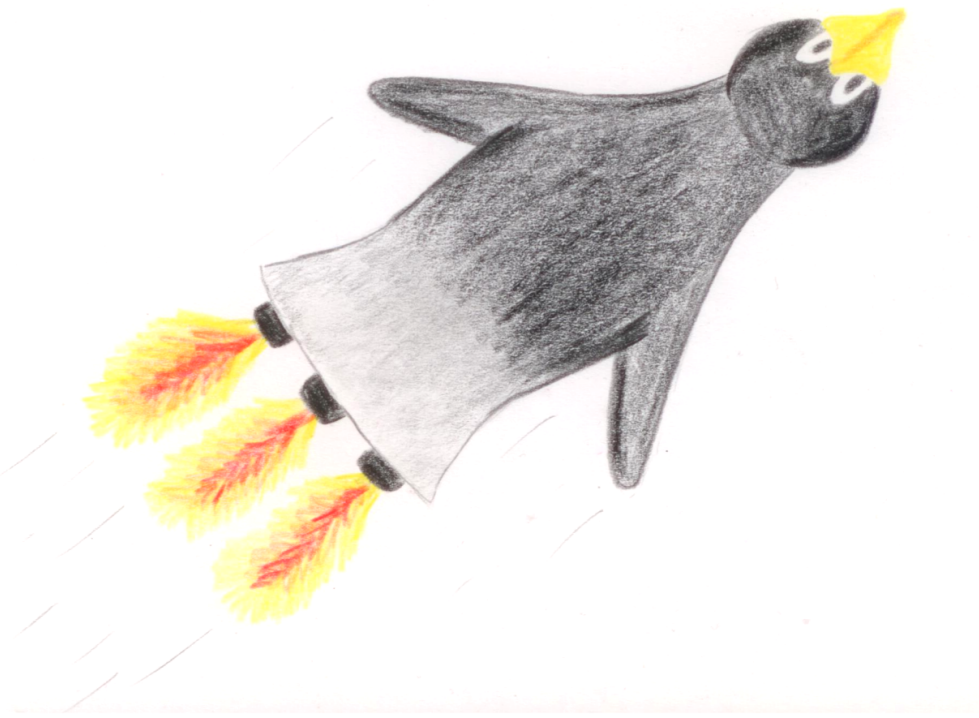
LZO compressed InitRAMFS

The test application is used as an INIT process (rdinit=)

measuring point	time
bootstrap - u-boot	---
u-boot - kernel	3,79s
kernel - application	0,372s
total	**4,162s**

LINUTRONIX
LINUX FOR INDUSTRY

Modified AT91 Bootstrap



LINUTRONIX
LINUX FOR INDUSTRY

Modified AT91 Bootstrap

AT91 Bootstrap starts Linux (without U-Boot)

measuring point	time
bootstrap - kernel	676ms
kernel - application	584ms
total	**1,260s**

LINUTRONIX
LINUX FOR INDUSTRY

lpj=

measuring point	time
bootstrap - kernel	676ms
kernel - application	384ms
total	**1,060s**

Optimizing the test system

LINUTRONIX
LINUX FOR INDUSTRY

< 1s !!

LINUTRONIX
LINUX FOR INDUSTRY

No (serial) console output (quiet)

measuring point	time
bootstrap - kernel	524ms
kernel - application	212ms
total	**736ms**

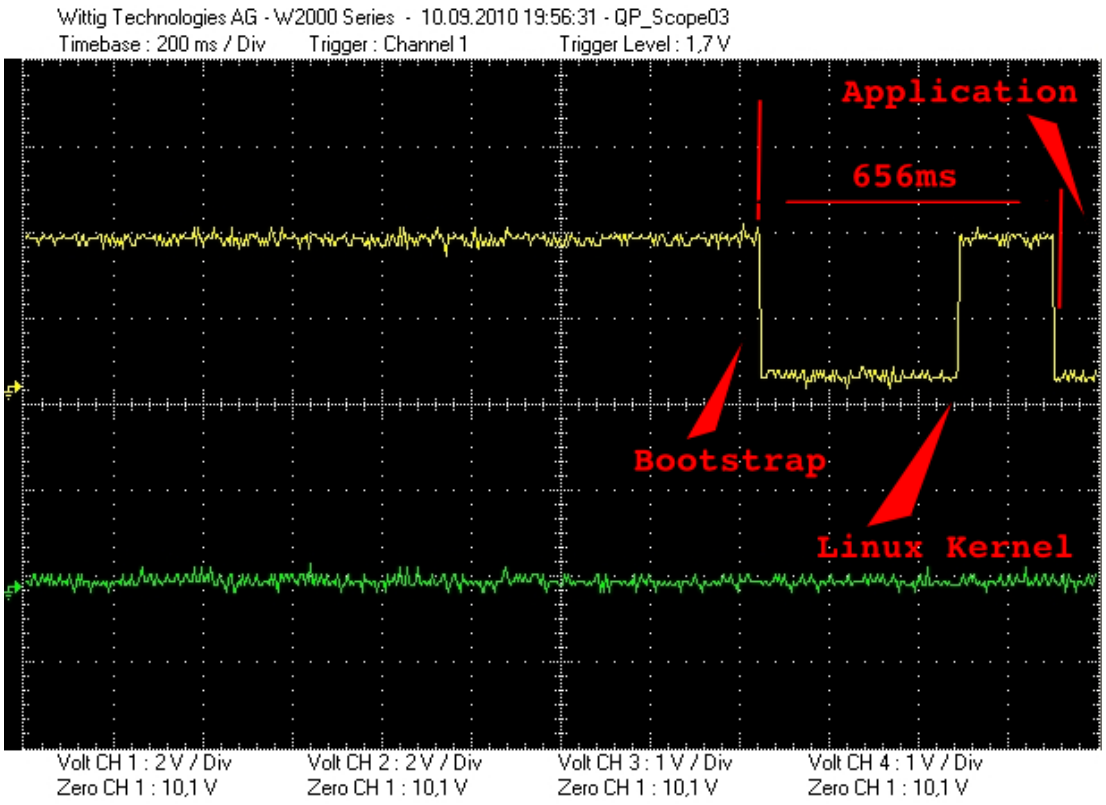
LINUTRONIX
LINUX FOR INDUSTRY

LZO compressed kernel image

measuring point	time
bootstrap - kernel	444ms
kernel - application	212ms
total	**656ms**

LINUTRONIX
LINUX FOR INDUSTRY

Final boot behaviour



LINUTRONIX
LINUX FOR INDUSTRY

Conclusion

- Linux can combine the advantages of a modern OS with hard boot time requirements
- Saving boot time with simple optimizations
- The hardware is an IMPORTANT part of a FastBOOT concept
- The boot concept is architecture independent!

LINUTRONIX
LINUX FOR INDUSTRY

Questions?

I'll also be around at the technical showcase! :)