Low-Power Linux

Wolfram Luithardt und Andéol Demierre

Hochschule für Technik und Architektur,
Fribourg, Schweiz

1.) Motivation

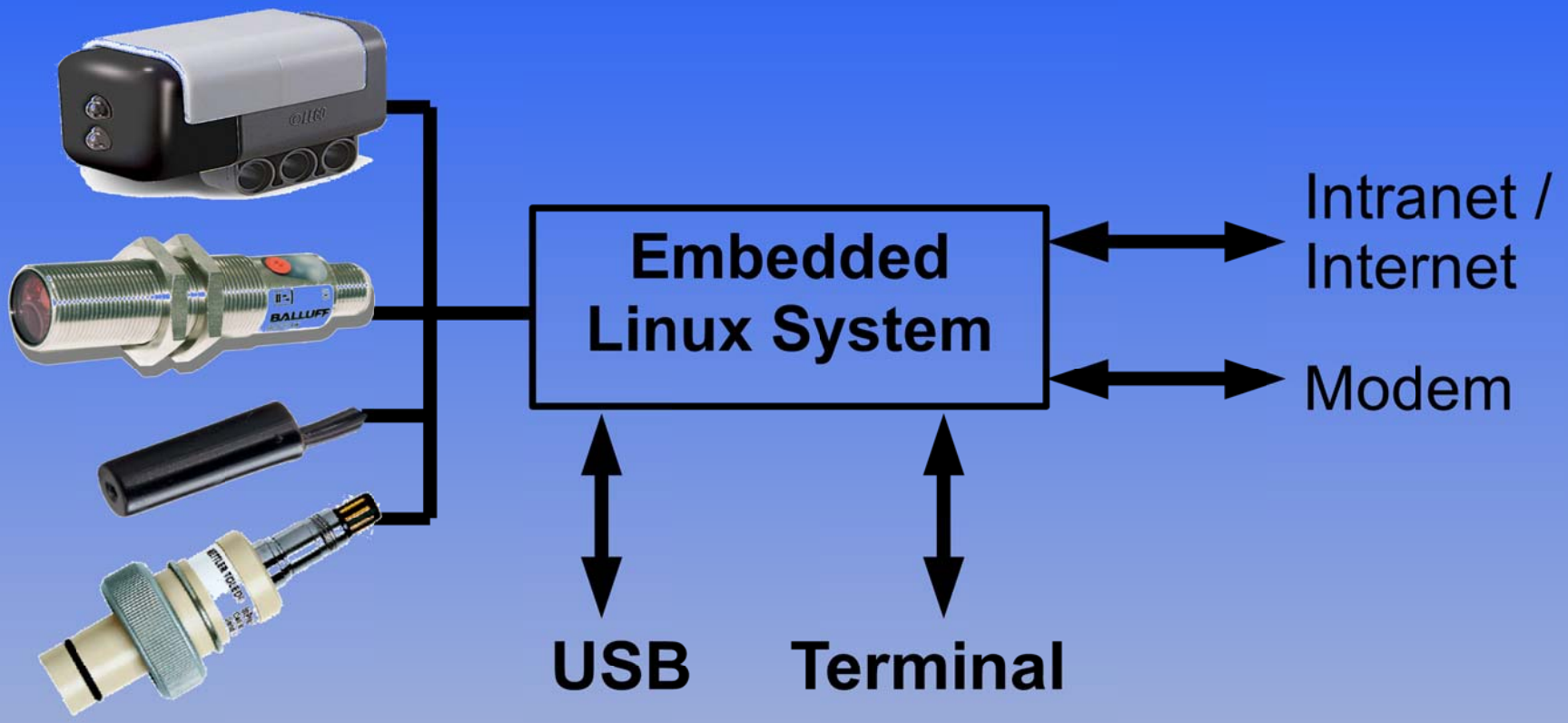
2.) Grundlagen

3.) Verschiedene Methoden zur Energiereduktion

4.) Zusammenfassung

Linux in embedded Systems

Ein industrielles Szenario



Ein Computer macht meistens nichts!

Grundlegende Gleichungen

$$P_{gesamt} = P_{statisch} + P_{dynamisch}$$

$$P_{statisch} = \Sigma\, U \cdot I_{Leak}$$

U = Corespannung

I_{Leak} = Leckstrom

$$P_{dynamisch} = \alpha C U^2 f$$

α: Aktivitätsfaktor

C: Parasitäre Kapazität (Transistoren und Leitungen)

U: Corespannung

f: Frequenz

Technologische Entwicklungen

zur Reduktion statischer Verluste:

$$P_{statisch} = \Sigma\ U \cdot I_{Leak}$$

- Peripheriegeräte (intern und extern) stromlos schalten

- Leckströme reduzieren: Neue Materialien innerhalb der Transistoren

Technologische Entwicklungen

zur Reduktion dynamischer Verluste:

$$P_{dynamisch} = \alpha C U^2 f$$

- ULV-Prozessoren

- Clock-Gating: Optimierung der Anzahl der Transistoren, die bei einer gewissen Operation getaktet werden müssen.

- Dynamische Anpassung der Frequenz an die Bedürfnisse

Sparmöglichkeiten

- Dynamische Reduktion der Taktfrequenz

- Prozessor schlafen legen

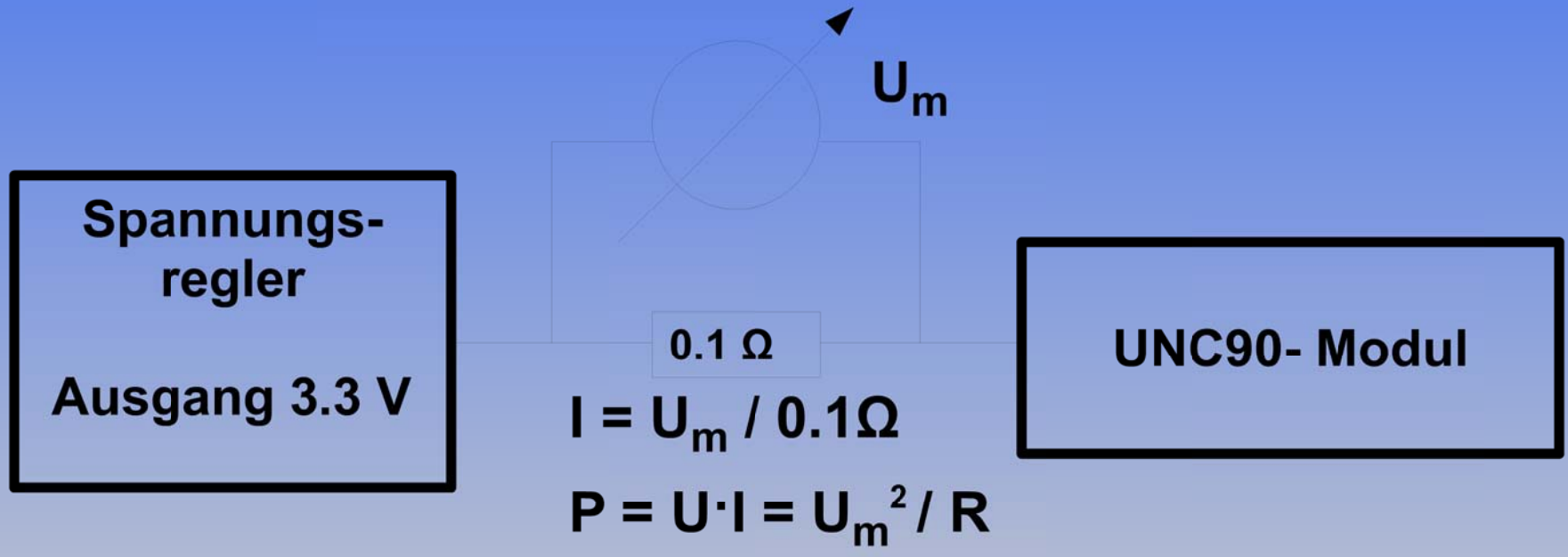
- Abschaltung von nicht benötigter Peripherie

- Reduktion der Versorgungs-spannung

Plattform UNC90

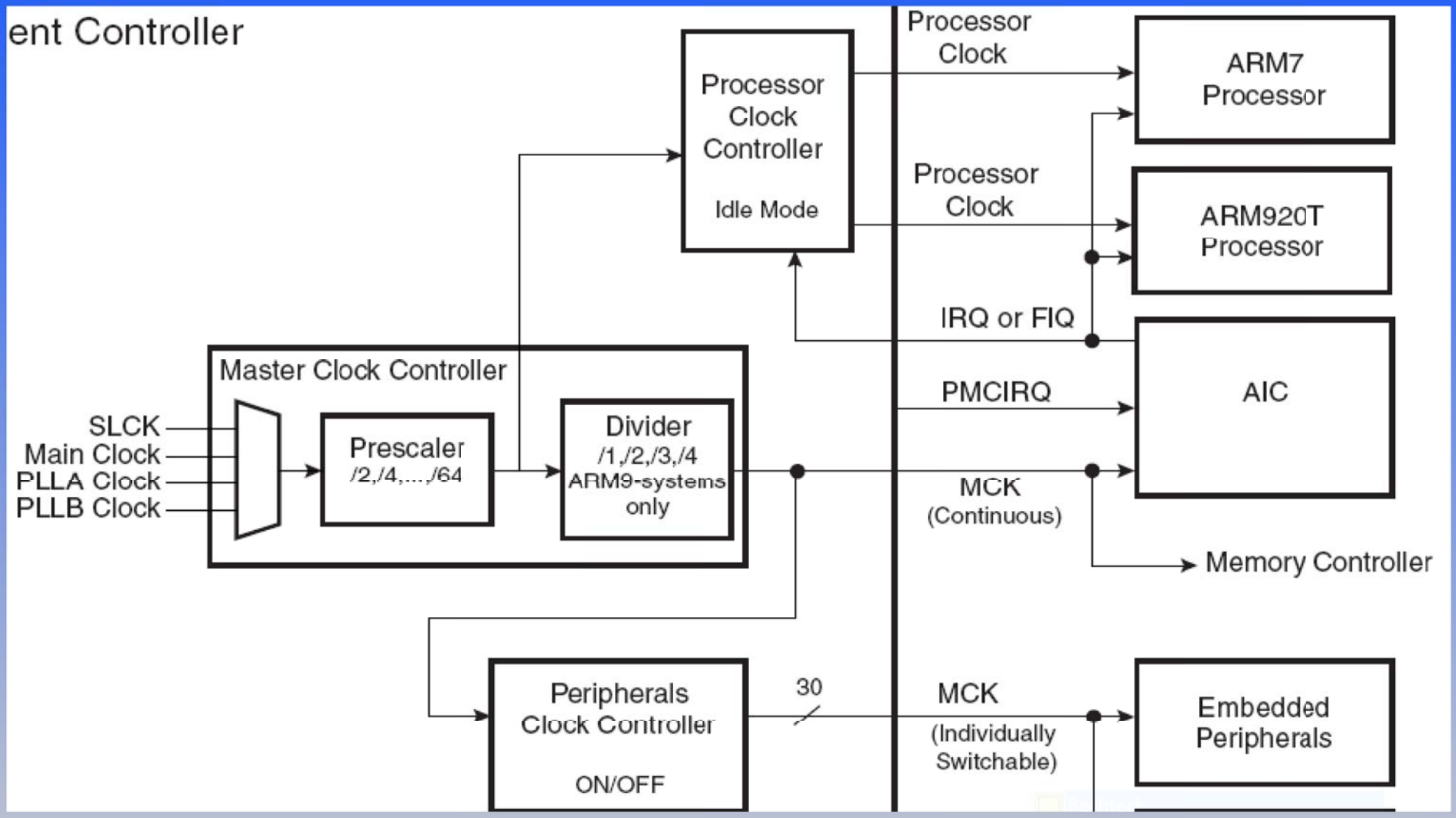
http://www.digi.com/pdf/prd_em_unc90.pdf

- Prozessor: AT91RM9200 (ARM 920)
- Maximale Frequenz: 180 MHz
- 32 MB Ram, 16 MB Flash
- Linux Kernel 2.6.12

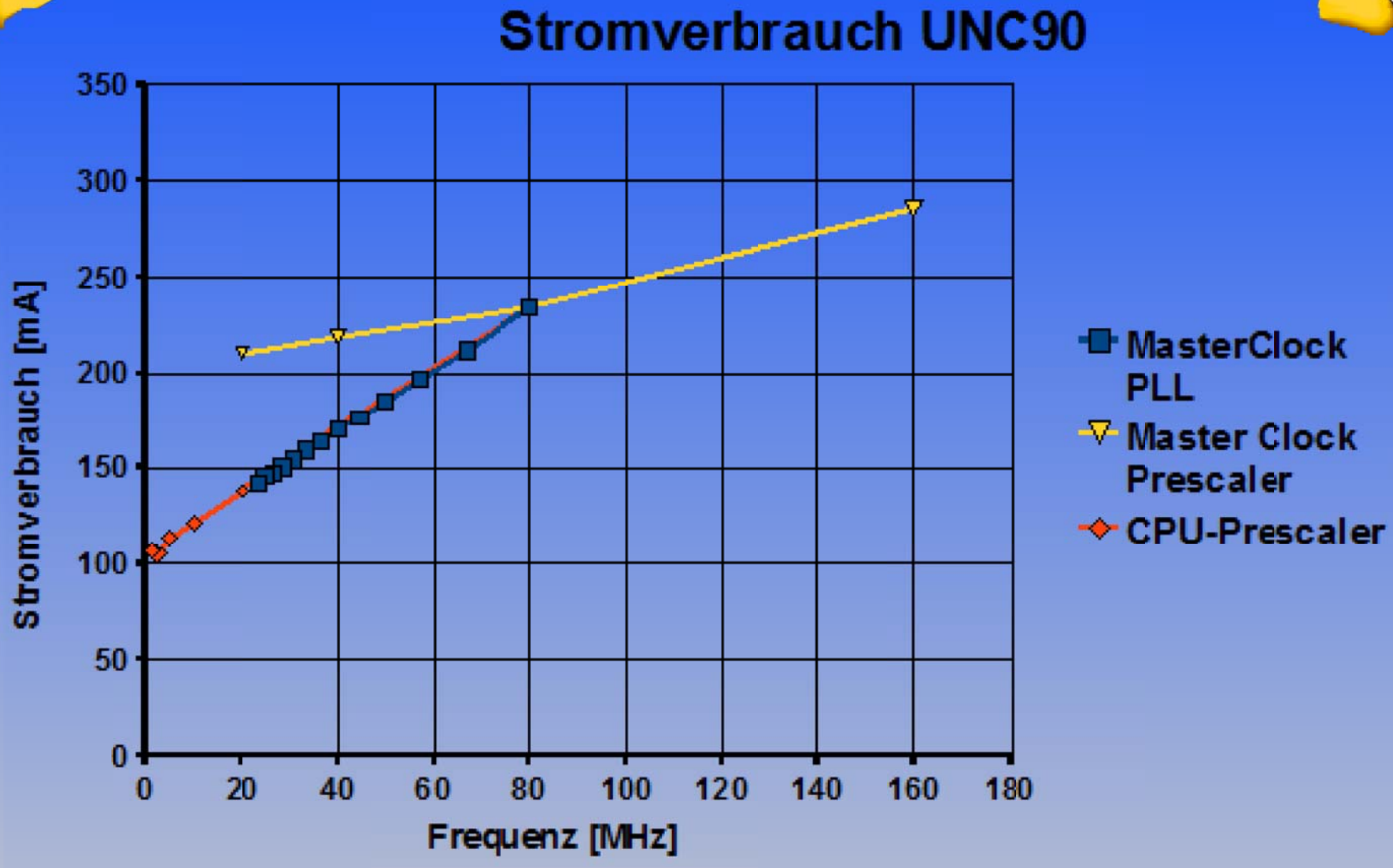


Frequencyscaling

Power Management Controller PMC



Strom (Frequenz)



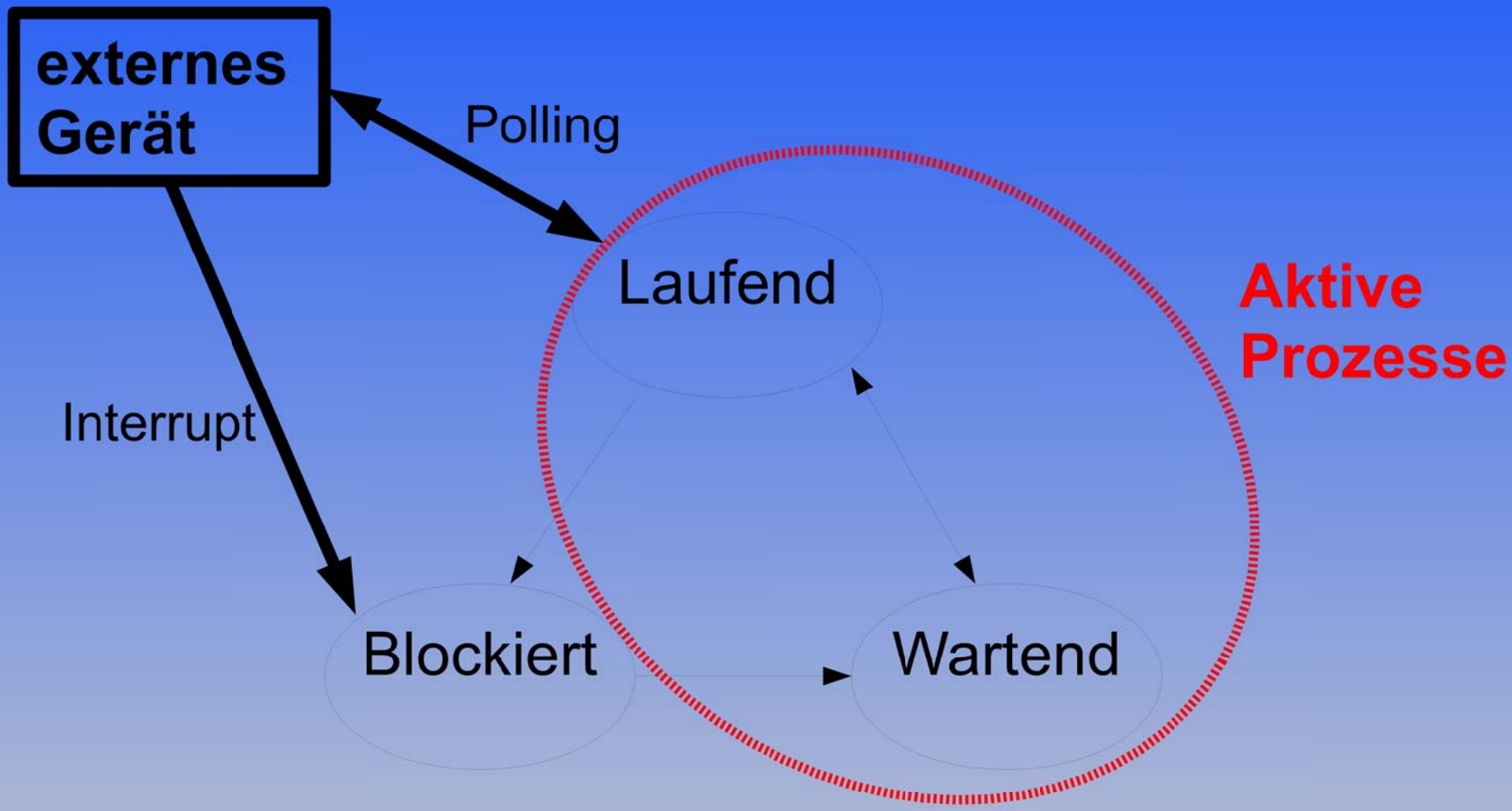
Manuelles Frequencyscaling

```
saveCurrent()
{
    PMC_ MCKR |= PRES_DIV2;      // Master Clock register auf /2 stellen
    US_BRGR = US_BRGR /2;        // USART Baudrate Divider Register
}


normalCurrent()
{
    PMC_ MCKR &= ~PRES_DIV2;     // Master Clock register auf /1 stellen
    US_BRGR = US_BRGR *2;        // USART Baudrate Divider Register
}
```

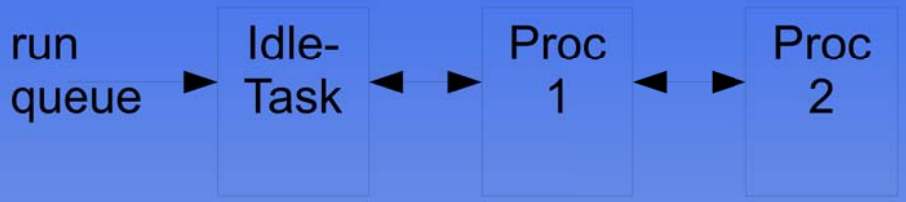
Wie kann dieses automatisiert werden ?

Unix- Prozessmodel



Die runqueue

runqueue-Struktur im Kernel



geblockte Prozesse:

Ein Feld dieser Struktur: **nr_running** enthält die Anzahl der aktiven Prozesse (inklusive des Idle-Tasks)

Ein Patch in sched.c

Die Funktion schedule() wird bei jedem Kontextwechsel (preemptiv oder gewünscht) aufgerufen:

```
if(unlikely(rq -> nr_running == 1)){
   if((saveMode == 0)&&(saveCounter > SAVEWAIT)){
      saveMode_on();
   } else {
      saveCounter++;
   }
} else {
      if(saveMode != 0){
         saveMode_off();
      }
      saveCounter = 0;
}
```

Die Routinen

```
saveMode_on() {
      local_irq_disable();
      waitForTXReady();
      saveCurrent();
      local_irq_enable();
}

saveMode_off() {
      local_irq_disable();
      waitForTXReady();
      normalCurrent();
      local_irq_enable();
}
```

```
waitForTXReady(){
  do {
        /* wait until TX_Ready
           Bit in USART Channel
           Status Register set
        */
  }
  while((US_ CSR & TXRDY)==0)
}
```

Ergebnisse

	ohne Patch	SAVEWAIT = 1	SAVEWAIT = 2	SAVEWAIT = 3	SAVEWAIT = 5
Active	254 mA	254 mA	254 mA	254 mA	254 mA
Idle	254 mA	188 mA	188 mA	188 mA	188 mA
Time	1581	1670	1620	1600	1583
Stabilität	5	15	12	8	7

Die Zeit wird mit einem Programm gemessen, das eine einfache mathematische Funktion durchführt. Dauer des Programms ca. 16 Sekunden

Warum nicht die Frequenz noch weiter senken?

Sleep-Mode

in linux/include/asm-arm/arch-at91rm9200/system.h

```
static inline void arch_cpu_idle(){

    AT91_SYS -> PMC_SCDR = AT91C_PMC_PCK;
}
```

Ergebnisse: sleepmode: 254 mA → 151 mA

mit Frequenzscaling: 188 mA → 128 mA

Dies war ein alter Kernel, wie sieht es bei neueren Systemen aus ?

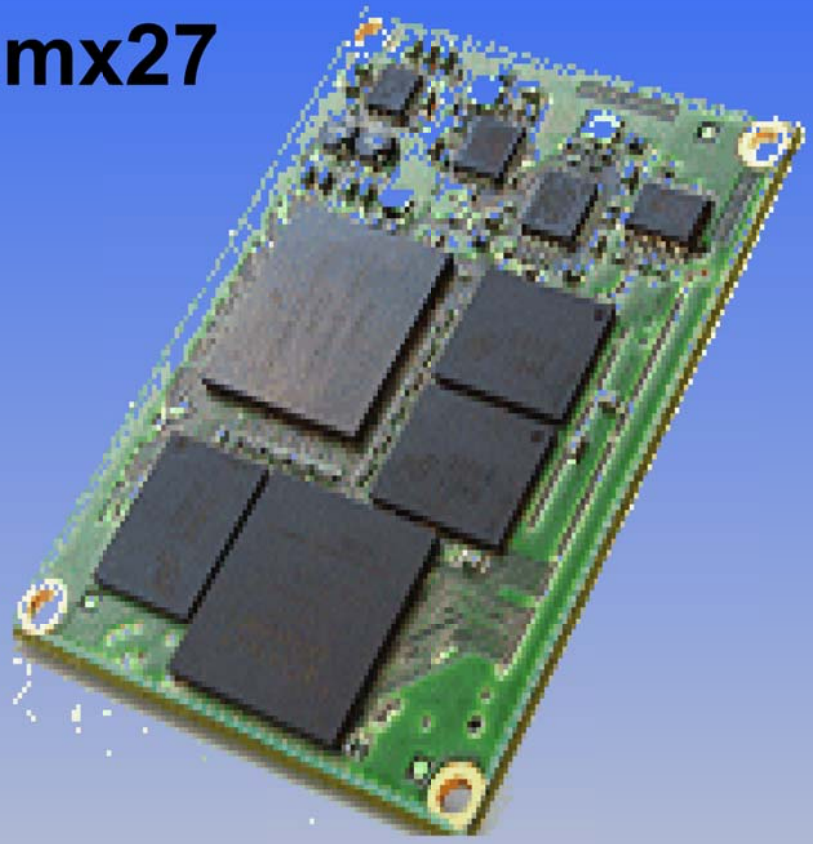
APF-27

www.armadeus.org

- Prozessor Freescale i.mx27
- Takt bis 400MHz
- 64 MB RAM
- 16 MB Flash
- Ethernet, USB etc...
- FPGA

....

Kernel: 2.6.29.4



On demand Governor

make linux26-menuconfig

```
.config - Linux Kernel v2.6.29.4 Configuration
                        CPU Power Management
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

      [*] CPU Frequency scaling
      [ ]   Enable CPUfreq debugging (NEW)
      <*>   CPU frequency translation statistics (NEW)
      [*]     CPU frequency translation statistics details
            Default CPUFreq governor (performance)  --->
      -*-   'performance' governor
      <*>   'powersave' governor
      <*>   'userspace' governor for userspace frequency scaling
      <*>   'ondemand' cpufreq policy governor
      <*>   'conservative' cpufreq governor
      [*] CPU idle PM support
      < > i.MX27 frequency driver

                <Select>    < Exit >    < Help >
```

Messungen unterschiedlicher Frequenzscalingmethoden

Frequenzscaling Methode	Strom Idle [mA]	Strom aktiv [mA]
Perfomance Governor	512 (400 MHz)	723 (400 MHz)
Powersave Governor	494 (133 MHz)	494 (133 MHz)
Ondemand Governor	**494** (133 MHz)	723 (400 MHz)
Scheduler Patch	371 (200 MHz)	723 (400 MHz)
Scheduler Patch	**348** (133 MHz)	723 (400 MHz)

Die aktuelle Frequenz kann über das sysfs bestimmt werden:

/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq

Interessantes Paper zum Thema:

http://ftp.kernel.org/pub/linux/kernel/people/lenb/acpi/doc/OLS2006-ondemand-paper.pdf

Sleep Mode

Im File:/linux-2.6.29.4/arch/arm/mm/proc-arm926.S

```
ENTRY(cpu_arm926_do_idle)
    mov     r0, #0
    mrc     p15, 0, r1, c1, c0, 0       @ Read control register
    mcr     p15, 0, r0, c7, c10, 4      @ Drain write buffer
    bic     r2, r1, #1 << 12
    mrs     r3, cpsr                    @ Disable FIQs while Icache
    orr     ip, r3, #PSR_F_BIT          @ is disabled
    msr     cpsr_c, ip
    mcr     p15, 0, r2, c1, c0, 0       @ Disable I cache
    mcr     p15, 0, r0, c7, c0, 4       @ Wait for interrupt
    mcr     p15, 0, r1, c1, c0, 0       @ Restore ICache enable
    msr     cpsr_c, r3                  @ Restore FIQ state
    mov     pc, lr
```

Ausschaltbar über Kernelboot-Parameter: **no-hlt** oder über Systemfunktion: **disable_hlt();**

Strom: **512 mA → 725 mA**

Ticks und Tickless Kernel

Linux Kernel
< 2.6.21

Timer-Ticks (alle 1, 4 oder 10 ms)

- Erhöhung Jiffies-Variable
- Kontextwechsel?
- SW-Timer ?
- Process Accounting

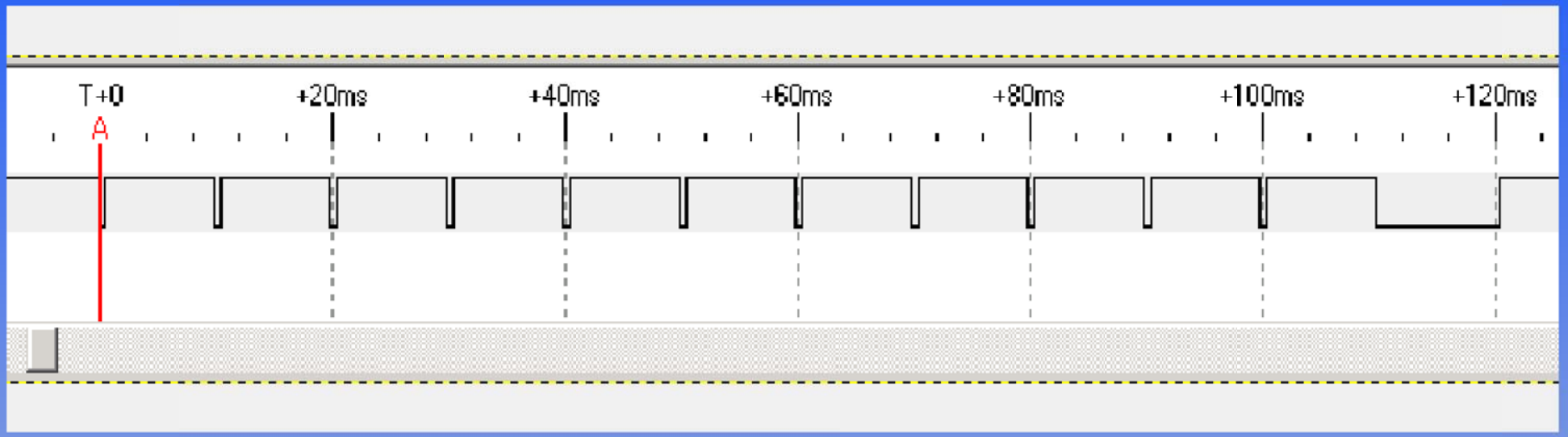
Linux Kernel
≥ 2.6.21

Tickless: Kernel wacht nur dann auf, wenn etwas zu tun ist

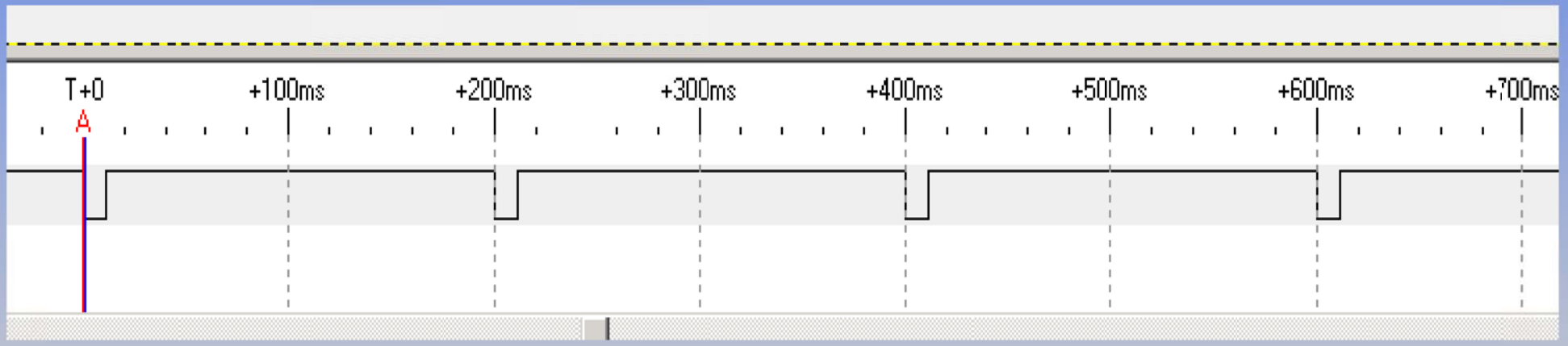
- Hardware-Timer wird entsprechend programmiert

Was bringt Tickless ?

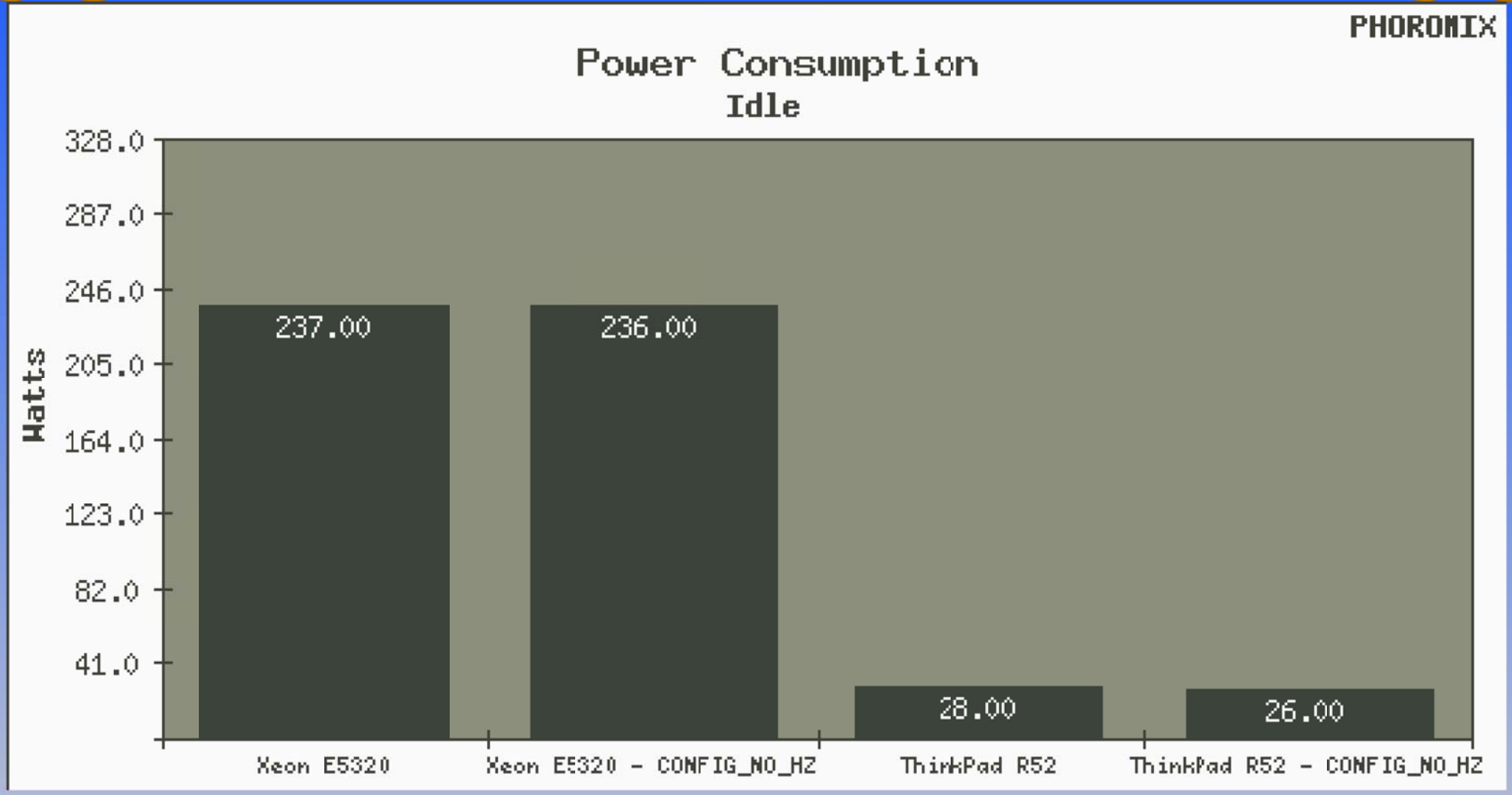
ohne Tickless: I = 535 mA



Tickless: I = 515 mA



Was bringt Tickless ?



http://www.phoronix.com/scan.php?page=article&item=651&num=1

Ausschalten von Peripherie

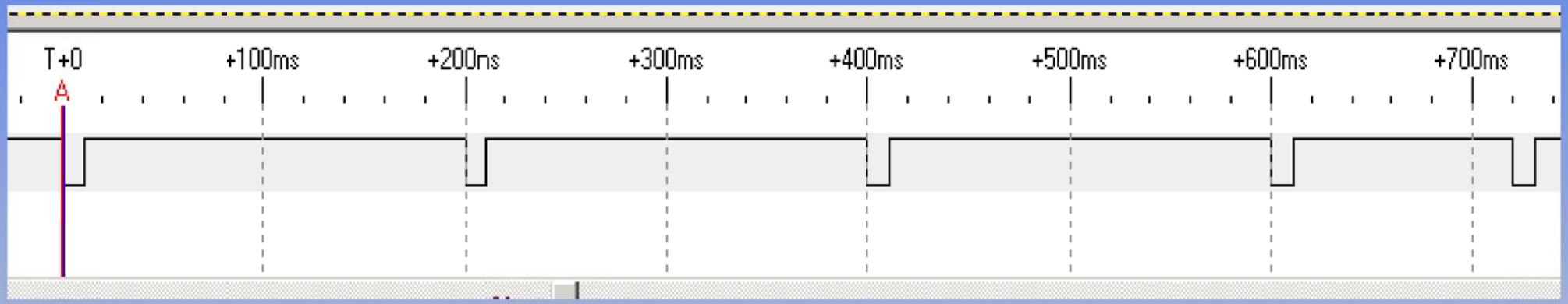
Bisher wurden die Messungen ohne USB-Support durchgeführt !

```
.config - Linux Kernel v2.6.29.4 Configuration
                              Device Drivers
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are
hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
<Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded
<M> module  < > module capable
        ^(-)
        -*- GPIO Support  --->
        < > Dallas's 1-wire support  --->
        < > Power supply class support  --->
        <M> Hardware Monitoring support  --->
        < > Generic Thermal sysfs driver  --->
        [*] Watchdog Timer Support  --->
            Sonics Silicon Backplane  --->
            Multifunction device drivers  --->
            Multimedia devices  --->
            Graphics support  --->
        <M> Sound card support  --->
        [*] HID Devices  --->
        [*] USB support  --->
            Armadeus specific drivers  --->
        <*> MMC/SD/SDIO card support  --->
        < > Sony MemoryStick card support (EXPERIMENTAL)  --->
        [ ] Accessibility support  --->
        v(+)
                    <Select>    < Exit >    < Help >
```

Eingeschaltetes USB

512 mA ⟶ 725 mA (+ ~30%)

Wird dies durch SW oder HW verursacht ?



Konfiguration des USB

- Über 100 Register !

- Peripheral Clock Control Register des Clockmoduls:

  Durch Ausschalten des Bits USB AHB Clock Enable, (Advanced High-performance Bus) wird der Strom reduziert (Clock-Gating ???)

- Der AHB Clock wird aber nur zum Erkennen eines neuen USB-Gerätes benötigt. Dies' ermöglicht, durch ein einfaches Programm, welches periodisch den USB AHB Clock einschaltet, den Stromverbrauch massgeblich zu reduzieren....

Bleibende Fragen

- Lassen sich die gemachten Änderungen generalisieren ?
- Wie lässt sich die Anzahl von Polling-Aktivitäten und Interrupts weiter verringern ?
- Warum benötigt USB so viel Energie und wie kann diese reduziert werden ?
- Wer hilft mit ?

Literatur

Daniel P. Bovet, Marco Cesati: **Understanding the Linux Kernel**, O'Reilly, ISBN: 978-0-596-00565-8

http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt

http://www.carlthompson.net/software/cpuspeed/

www.lesswatts.org

Manuals der Prozessoren

Vielen Dank für die Aufmerksamkeit !

Weitere Infos:

wolfram.luithardt@hefr.ch