RTOS

"UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity."

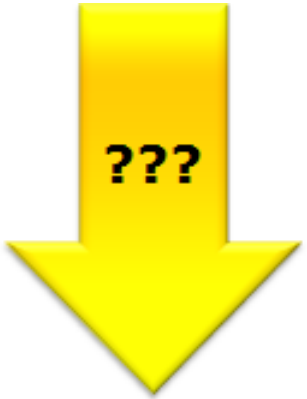
— Dennis Ritchie

Prof. Erich Styger

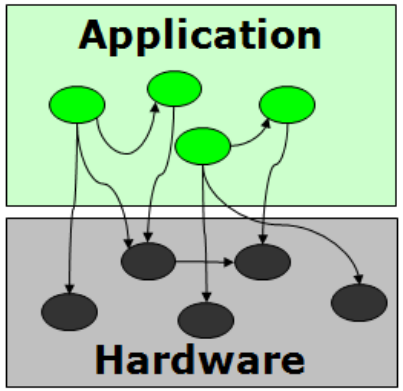
erich.styger@hslu.ch
Lucerne University of Applied Sciences and Arts

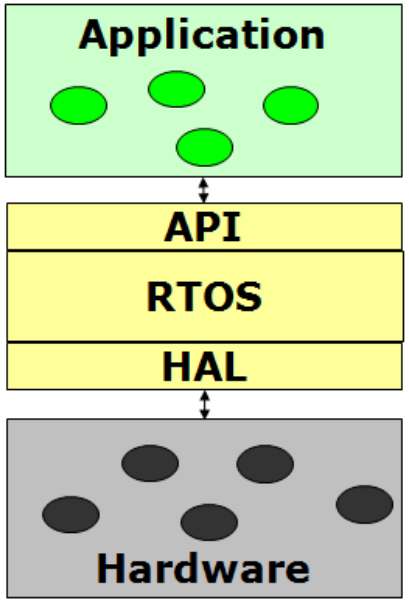
Learning Goals- Goals

- Understand reasons for RTOS
- Knows requirements for RTOS
- Differentiates Standard vs. Realtime OS



Introduction



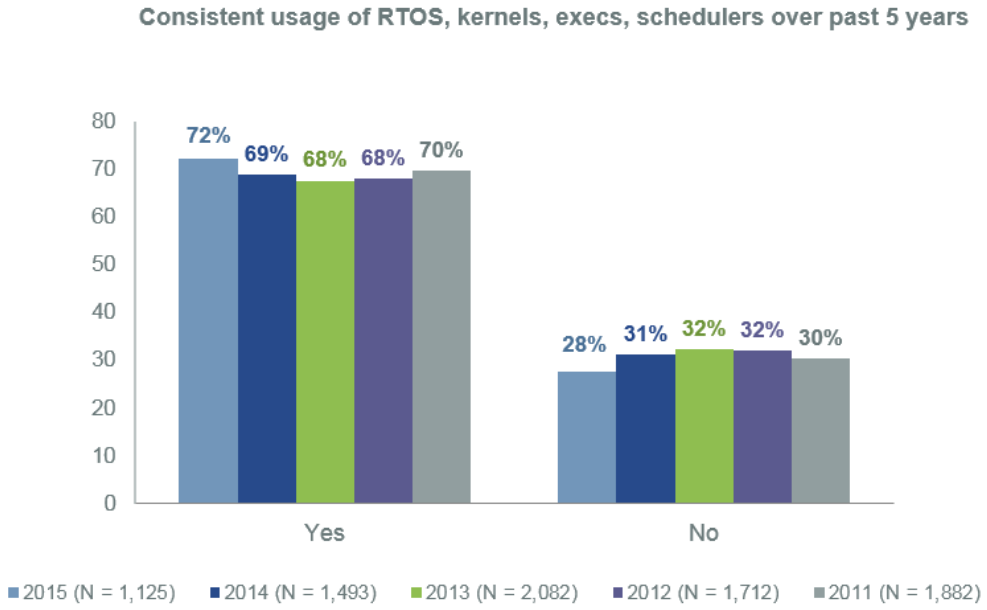


- Solves Synchronization problems
- Resource Pooling, provides services
- Scalability, maintainability
- Need a scheduler

Reasons for a RTOS

2015 UBM Electronics Embedded Markets Study

Does your current embedded project use an operating system, RTOS, kernel, software executive, or scheduler of any kind?



39

UBM

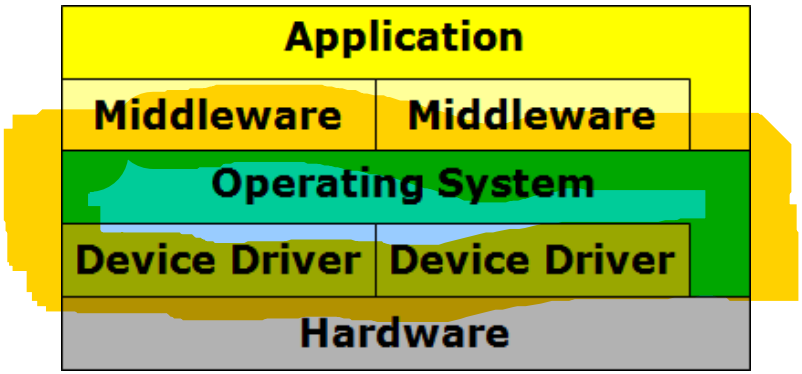
[1]

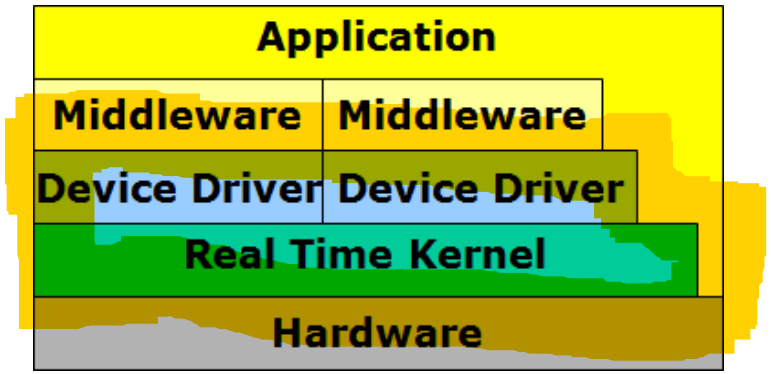
[1] Source: UBM Embedded Market Study 2014

Operating System Services

- Not “reinventing the wheel”
- Operating System provides Services
  - **Resource Pooling**: mutual access to hardware and resources, memory
  - **Scheduling**: quasi-concurrent execution of services
  - **Abstraction**: time base, memory protection (MMU)
  - **Middleware**: file system, communication stacks
- Different Level of Services
  - **Timer based Scheduler**: ’Trigger’ or ’Interrupt’ systems
  - **Mini-Kernels**: FreeRTOS, μC-OS, mbed OS, RTXC
  - **Mid-Range RTOS** with driver stacks: MQX, QNX, eCOS
  - Mobile and **Embedded Linux**: Android, iOS, Debian
  - **Host** Linux and OS: MS Windows, Mac OS X, Ubuntu

Driver Model





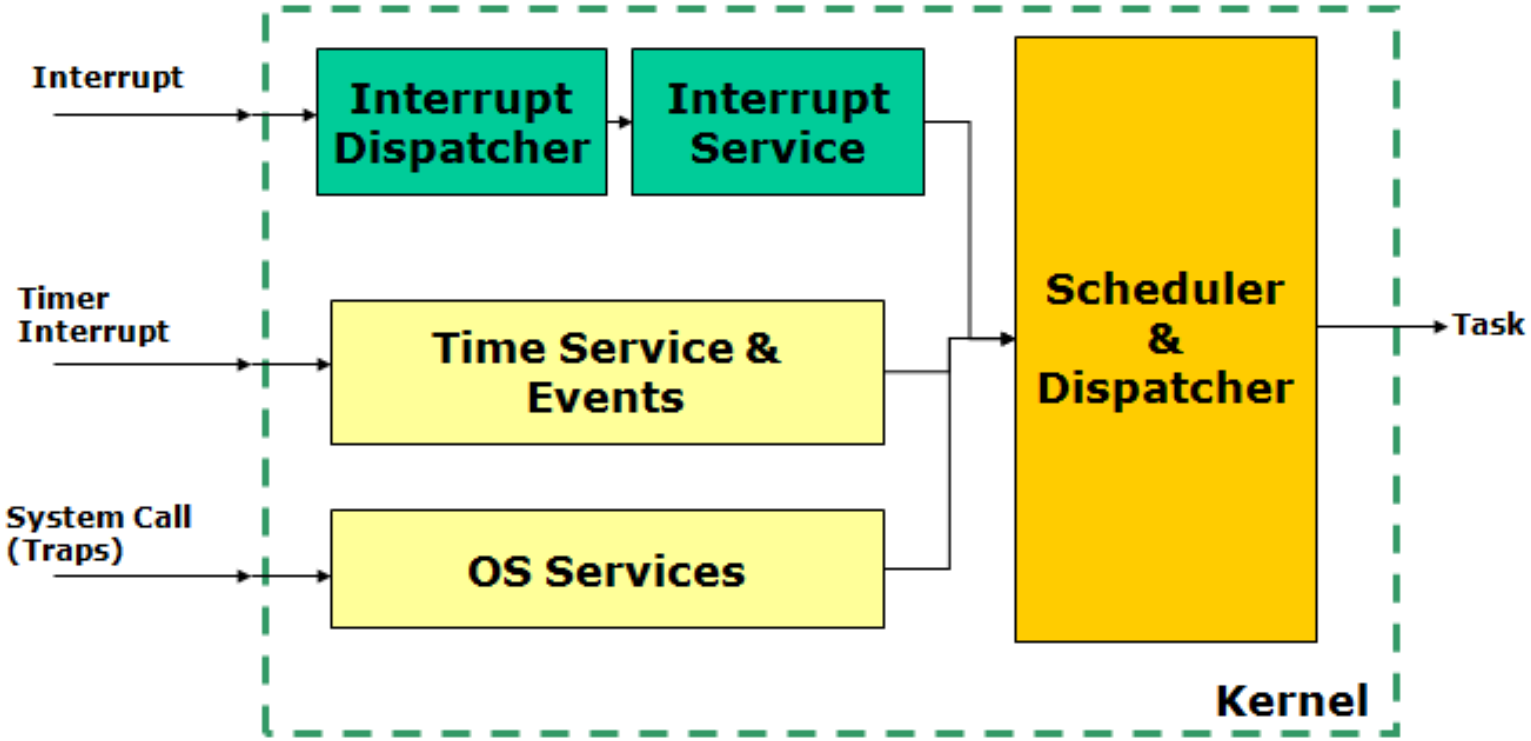
- Standard Operating System
  - Drivers part of OS
  - Application not allowed to take control of hardware
  - Usually better protection
- Real Time Operating System
  - Drivers as services outside OS
  - Application has direct access to hardware
  - Better performance, closer to the hardware

Realtime Operating System

- "A Real-Time Operating System (RTOS) is an Operating System (OS) intended for real-time Systems."[2]
- $\neq$ “As fast as possible”!
- Requirements
  1. The correct result
  2. At the correct time
  3. Independent of the current system load
  4. In a deterministic and foreseeable way
- General purpose OS will not be able to fulfill these requirements

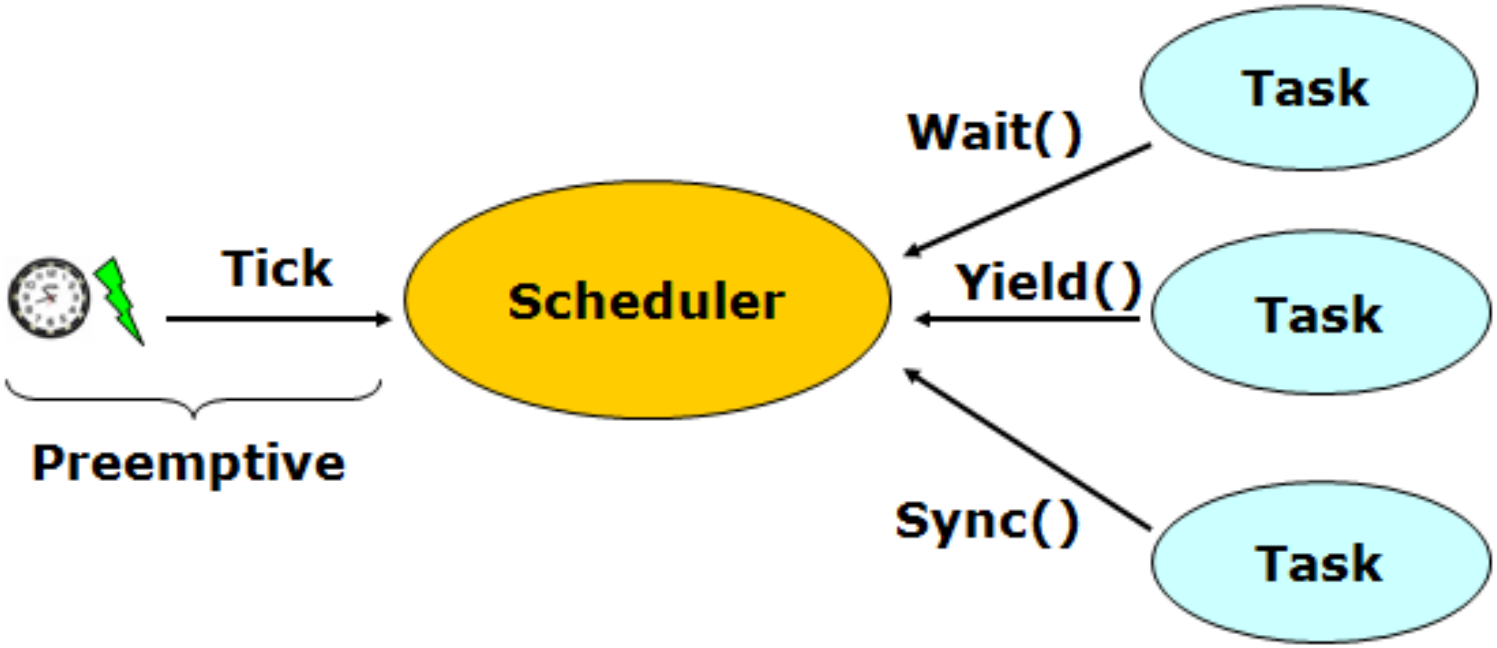
[2] Source: Wikipedia

RTOS Kernel Architecture



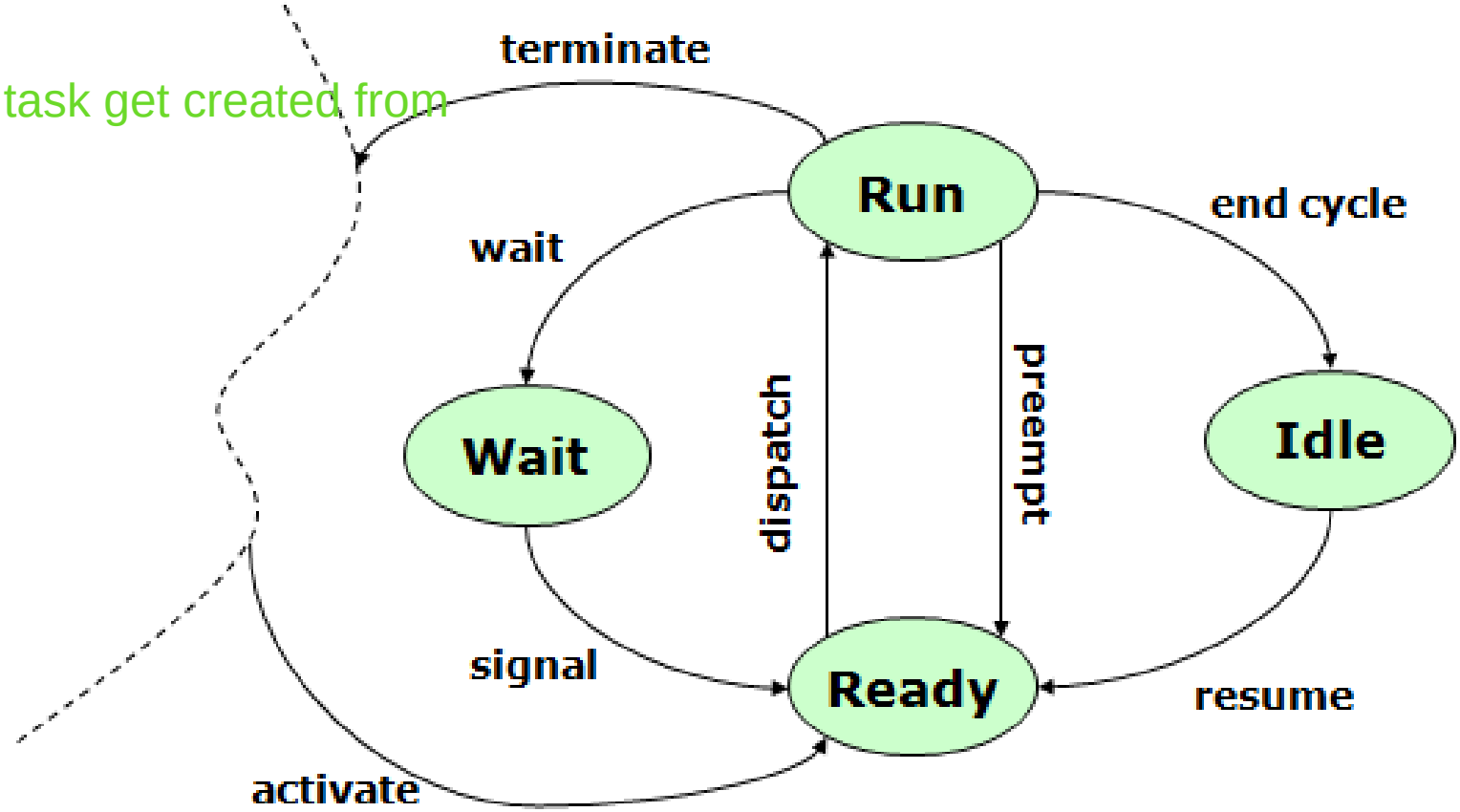
- Catching interrupts (few or all)
- Needs time base: Timer/Tick interrupt
- System calls and traps: trigger interrupt
- Scheduling: System call or interrupts

Scheduler



- Preemption with Tick interrupt
- Tick passes control to Scheduler
- Scheduler can schedule other task
- Opportunity for scheduler: Tick, Wait(), Yield() and Sync()

Operating System Process States



- Life-Cycle with different task states
- Only one task in running state

Embedded Linux vs. RTOS

Realtime OS

- Hard Realtime
- CPU $\geq$20 MHz (100 MHz), Cortex-M, FPU
- RAM $\geq$1 KByte (16-64 KByte SRAM onchip)
- FLASH $\geq$16 KByte (64-256 kByte)
- Boot time $\leq$10 ms (1 ms)
- No File System, simple UI
- Controller, UART, CAN, SPI, USB, ...
- 1μ-10mA, Battery

Standard OS

- Soft Realtime
- CPU $\geq$500 MHz (1 GHz) Cortex-A8, MMU
- RAM $\geq$32 MByte (512 MB DDR3)
- FLASH $\geq$256 kByte (4 GB)
- Boot time $\geq$100 ms (60s)
- File System, GUI
- Gateway, HTTP, TCP/IP, HDMI, ...
- 200-500mA, Wall plug

Summary

- Helps solving synchronization problem
- Makes application scalable
- Not every application needs RTOS
- The correct result at the correct time, independent of the system load, in a deterministic and foreseeable way
- Differences in driver model: access to hardware
- Requirements for standard OS and Realtime OS are different