MSE | MASTER OF SCIENCE IN ENGINEERING

TSM_AdvEmbSof

Scheduling for Embedded Systems Part III

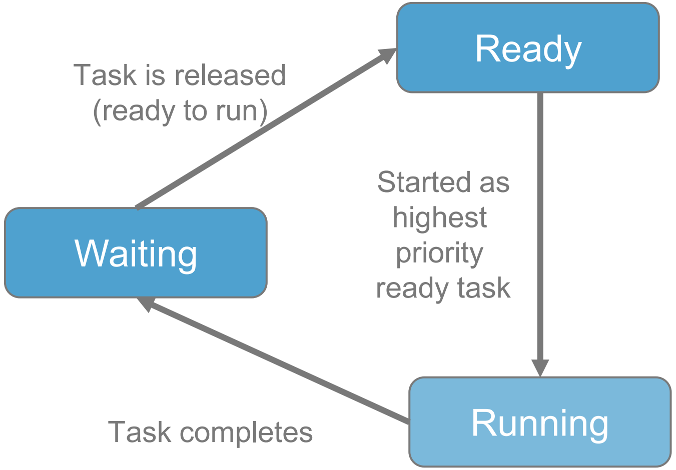
Serge Ayer | 10.11.2023 | Cours MSE

Dynamic scheduling

- Rather than applying a static order of tasks, allow task scheduling to be computed dynamically online:
  - Based on importance (priority) or any other criteria (e.g. task deadline, duration or creation time).
  - This also simplifies the creation of tasks with arbitrary rates.
- Scheduling based on task importance
  - Prioritization means that less important tasks don’t delay more important ones.
- How often does the scheduler decide what to run?
  - Coarse grain: after a task finishes. It is non preemptive or Run-To-Completion (RTC)
  - Fine grain: at any time. It is preemptive – one task can preempt another less important task.

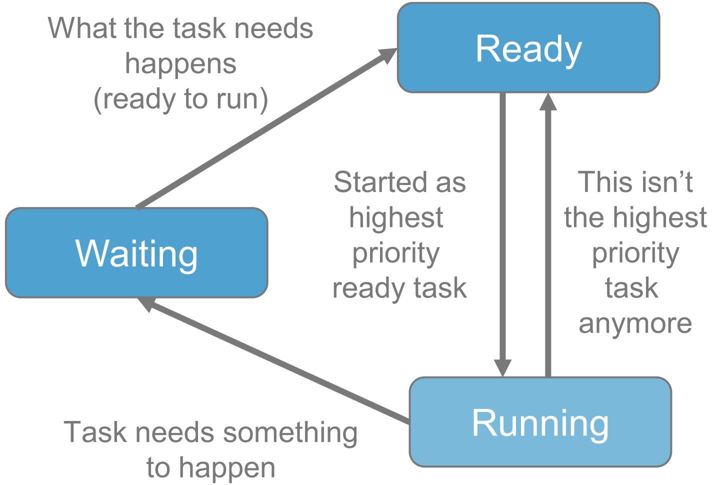
RTC: Task State and Scheduling Rules

- The Scheduler chooses among Ready tasks for execution based on priority
- Events can change a task state
- Scheduling rules:
  - If no task is ready, the scheduler sits in idle state.
  - If no task is running, the scheduler starts the highest priority ready task, if any.
  - Once started, a task runs until it completes (no preemption).
  - Completed tasks enter the waiting state until released again



Preemption: Task State and Scheduling Rules

- The Scheduler chooses among Ready tasks for execution based on priority
- Scheduling rules:
  - A task's activities may lead it to waiting (blocked)
  - A waiting task never gets the CPU. It must be signaled by an ISR or another task.
  - Only the scheduler moves tasks between ready and running



Preemptive Scheduling Algorithms (Periodic Tasks)

- Accepted constraints
  - No resource sharing
  - D = T, periods are fixed, worst-time execution times are fixed
- Rate Monotonic Scheduling
  - Tasks with higher request rates/shorter periods have higher priorities.
  - Fixed periods means fixed priorities.
  - Is optimal among fixed-priority algorithms.
  - Schedulability test: $\mathrm{U} = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq \mathrm{n}(2^{\frac{1}{n}} - 1)$
- Earliest Deadline First
  - Tasks with earlier absolute deadlines will be executed at higher priorities.
  - Priorities are dynamic since absolute deadlines of periodic tasks vary over time.
  - Schedulability test: $\mathrm{U} = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$

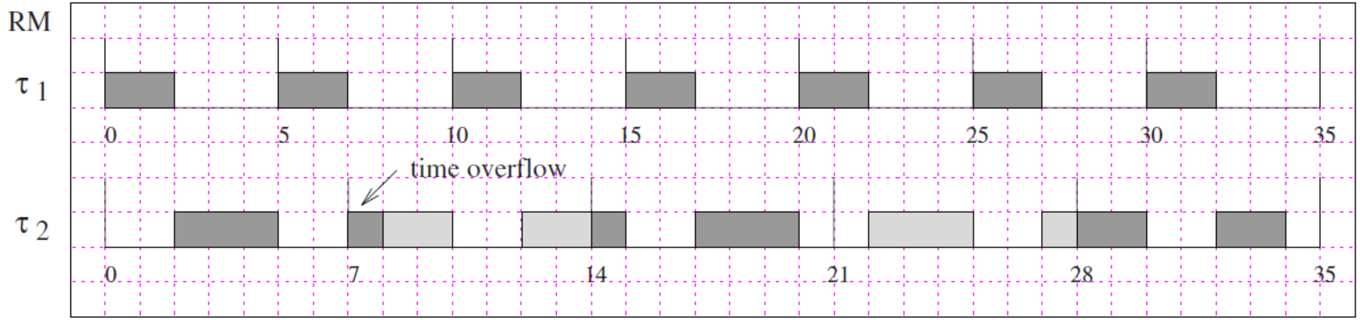
Schedulability: a simple example

- Given two tasks:

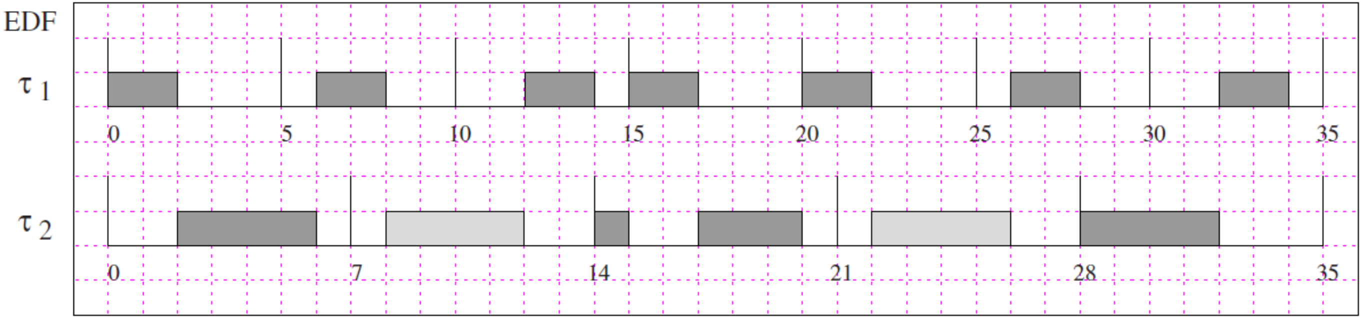
$$C_1 = 2 \;\; T_1 = 5 \;\; C_2 = 4 \;\; T_2 = 7$$

- $U = \frac{2}{5} + \frac{4}{7} = \frac{34}{35} \approx 0.97$
- $U > 2\left(\sqrt{2} - 1\right) \approx 0.83$
  - Schedulability using Rate Monotonic is not guaranteed
  - Schedulability using EDF is guaranteed

Schedulability: a simple example



(a)



(b)

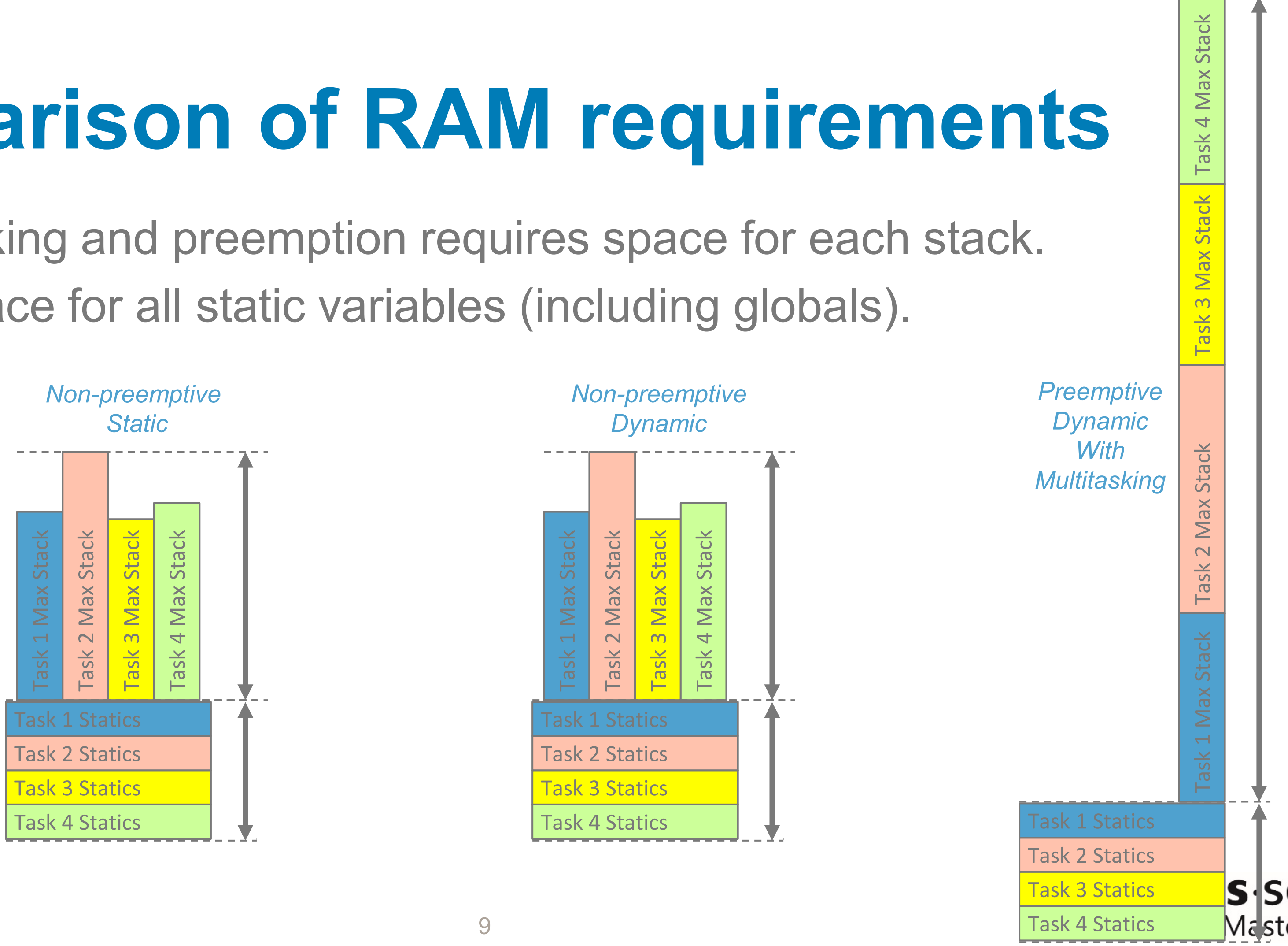
Taken from "Hard Real-Time Computing Systems, Giorgio C. Buttazzo

Preemption or not Preemption?

- Preemption offers better response times
  - Can do more processing.
  - Can lower processor speed, saving money and power.
- Preemption requires more complicated programming and more memory.
- Preemption introduces vulnerability to data race conditions.
- Most RTOS support preemption and allow task scheduling based on priorities.

Comparison of RAM requirements

- Multi-tasking and preemption requires space for each stack.
- Need space for all static variables (including globals).



Mbed OS/RTX Task Scheduling

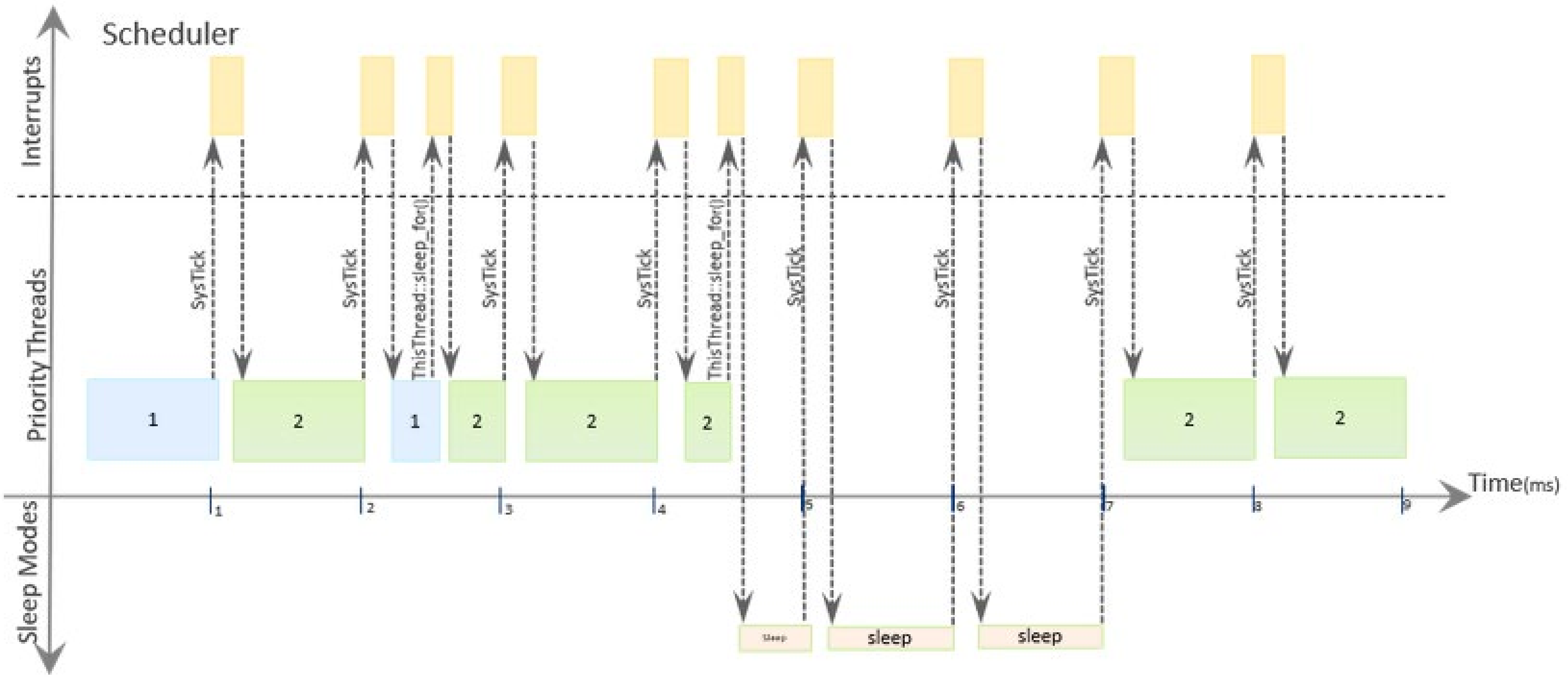
- Mbed OS RTOS is based on RTX5
  - RTX5 is the ARM CMSIS-RTOS implementation
  - RTX5 implements a low-latency preemptive scheduler
- Scheduling is tightly linked with the concept of threads
- Cortex-M processors support two modes of operation, Thread and Handler modes
  - Entering Thread Mode: on Reset or as a result of an exception return (privileged or unprivileged code)
  - Entering Handler mode: as a result of an exception (only privileged code).

Mbed OS/RTX Task Scheduling

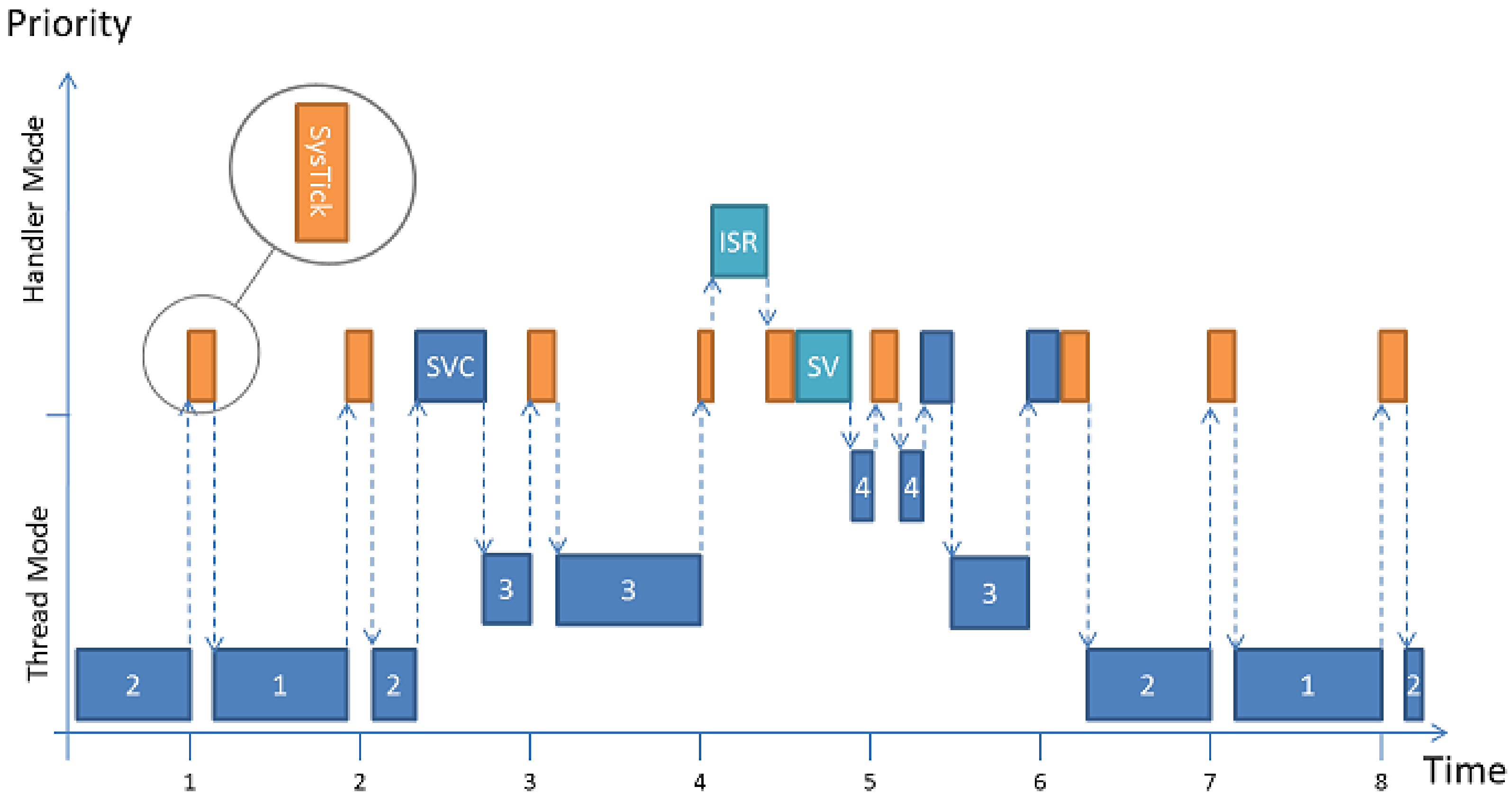
- Handlers used for low-latency scheduling
  - SysTick_Handler (time-based scheduling)
  - SVC_Handler (RTOS call, lock-based scheduling)
  - PendSV_Handler (interrupt-based scheduling)
- Priorities are configured such that no preemption happens between handlers
  - No need for critical section for protecting the scheduler
  - ISR can still preempt handlers, without latency
- Combination of priority and round-robin based scheduling
  - Round Robin for tasks of same priorities
  - Priority based for other tasks

Mbed OS Task Scheduling

Mbed OS uses the SysTick timer in periods of 1ms to process threads' scheduling.



Mbed OS/RTX Scheduling



RTX/Mbed OS Scheduling Options

- Pre-emptive scheduling
  - Each task will run until it is pre-empted (based on priority) or has reached a blocking OS call.
- Round-Robin scheduling
  - Each task with the same priority will run for a fixed period, or time slice, or until it has reached a blocking OS call.
  - Quantum is determined at compilation time (in the RTX_Config.h file: OS_TICK_FREQ/OS_ROBIN_TIMEOUT).
  - If quantum expires, the thread state will be changed to READY.
- The default scheduling option for RTX is Round-Robin and Preemptive
  - Round-Robin can be c.onfigured/disabled

Scheduling Algorithms with Mbed OS

- Considering only application threads
- Co-operative multi-tasking / RTC
  - Round-Robin is disabled.
  - Each task gets the same fixed priority.
  - Each task will run until it reached a blocking OS call, uses the ThisThread::yield() call or finishes (RTC).
- Rate Monotonic Scheduling
  - Round-Robin is disabled.
  - Each periodic task gets a fixed priority based on its period.
- Earliest Deadline First
  - Would require to recompute the priority of each thread each time a task gets ready
  - Is not feasible without modifying the scheduler itself

Task/Context Switching

- The Thread Control Block (TCB/osRtxThread_t) makes context switching a bit easier
  - Scheduler will start or stop a process accordingly
  - Stores necessary information in the TCB to stop
    - Hardware registers
    - Program Counter
    - Memory states, stack and heap
    - State
  - Similarly, loads necessary information from the PCB
- Notice that context switching does consume time!
  - Could be up to several thousand CPU cycles (for Cortex-M RTX around 200-300 cycles.
  - Hardware support is also needed
- But multitasking is feasible, although only one active process at any given time