**Realtime Systems**

**1.You are able to name the two most requirements for RTS**

1.Functional requirements have to be met! (like for all systems)

2.For real-time systems:defined response times need to be met!i.o.w.: the reaction has to be in a predefined time interval,not too late and not too early.

Important: real time systems do not have to be fast, they have to respond in a predefined time-frame!real-time systems have to meet deadlines.

**2.You are able to name additional requirements for RTS**

Real-time systems' requirements:

•determinability(it comes to a well defined end!)

•predictability(when it comes to the end!)

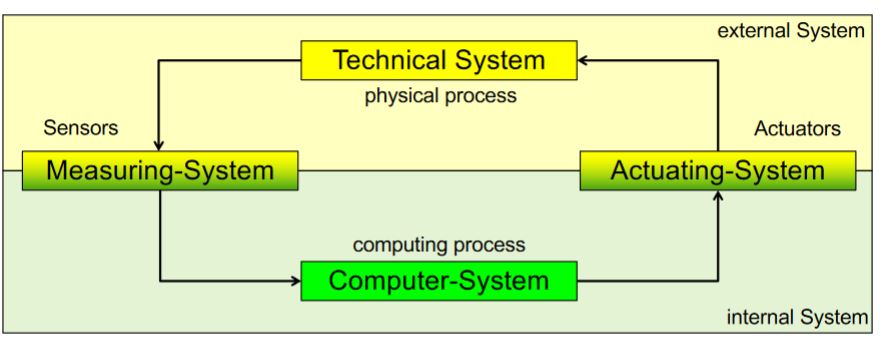
(Vorhersagbarkeit/Vorhersehbarkeit/Determiniertheit

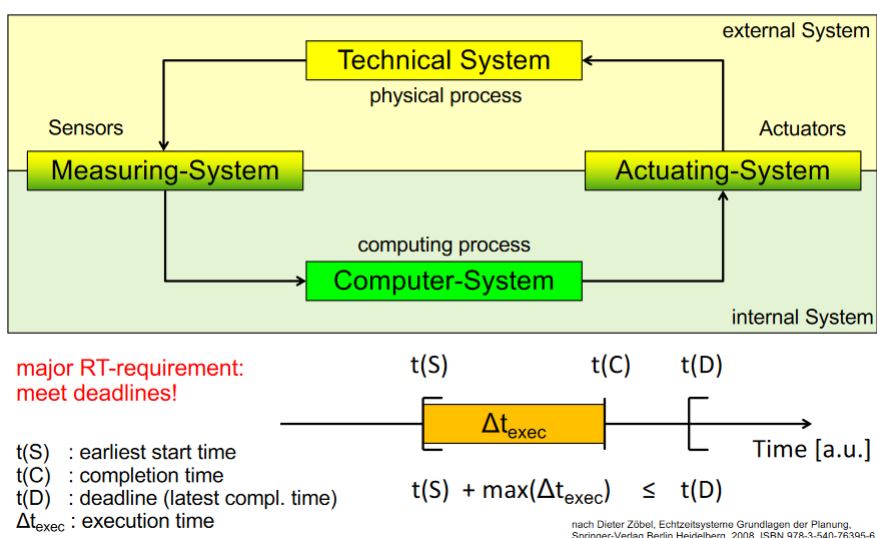
Real-Time Systems are in general safety-critical.This leads to a third requirement:

•reliability(Zuverlässigkeit)

**3.You able to illustrate these requirements with an example**

**4.You can describe the basic model for real time systems using an image explain how it works**

****

****

**5.You know the tree basic types of RTS and can describe their properties as well as name examples for fields of application**

* **Hard RTS**: Missing a deadline is a total system failure (e.g. airbag in car, use case in aviation).
* **Soft RTS**: The usefulness of a result degrades after its deadline, there

by degrading the system‘s quality of service. (e.g. warning systems, e.g. distance Warner).

* **Firm RTS**: Infrequent deadline misses are tolerable but may degrade the system‘s quality of service.. (E.g. ignition-point-optimizer for motor).

**6.You can list some classification criteria of RTS and comment on them. You can name the requirements for RT-clocks.**

* Distribution: centralised or distributed RTS
* Interactive or autonomic system
* Hierarchical or flag system
* Time-driven or event-driven RTS

**Real-Time Clock:**

-high resolution (*hohe Auflösung*)

-high accuracy (*hohe Genauigkeit*)

-smooth time adjustment (*reibungslose Zeiteinstellung*)

**Time, Foundation**

**1.What are the qualities of time in computer science?**

**We assume that:**

•time exists

•in real world there exists one and only one coordinated Universal Time

**We define:**

•that time is growing all the time

•that every clock/computer has its own computer time

•the computer time is bijective related to or somehow synchronized with the Universal Time

**We assume:**

•time in real world is a continuum and

•time is discrete in a computer and the granularity depends on the rate of clock of the used computer

**2.You can tell the terms absolute point in time, relative point in time and duration apart and explain them**

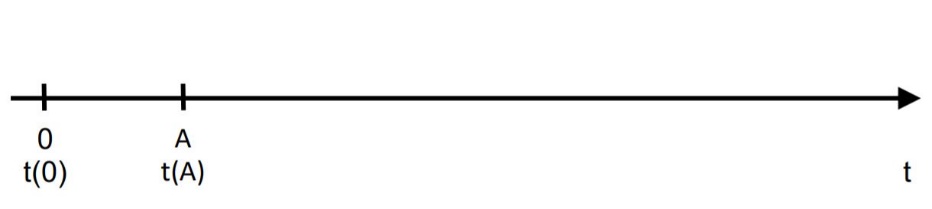
**3.You can classify sequence of events as single instance, periodic, aperiodic and sporadic**

When the time of the reappearing is constant, the task is called a **periodic task.**

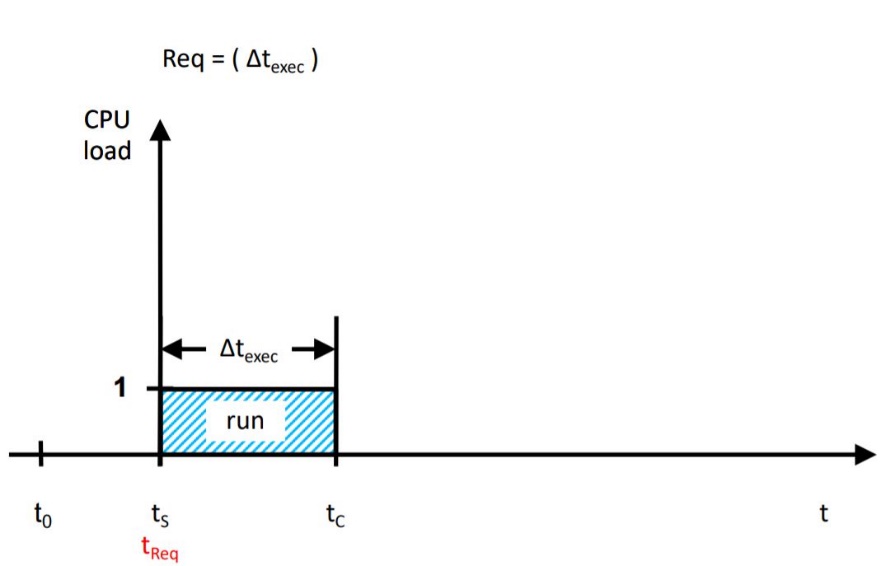
When the recurrence is random but there is a minimum time difference between two jobs of the same task, the task is called a **sporadic task.**

When the recurrence is random but there is no minimum time difference between two jobs of the same task, means they may even appear at the same time, the task is called an **aperiodic task.**

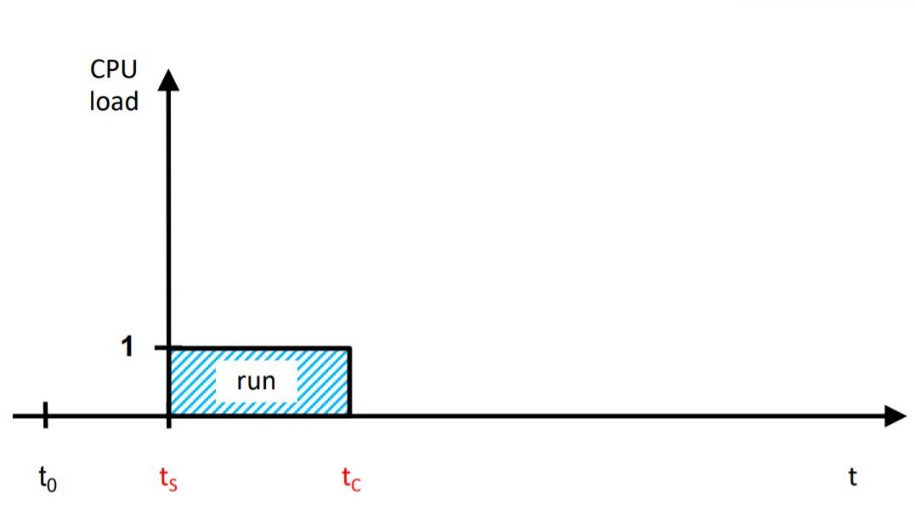
**4.You can explain the points in time: zero-/reference point, request time, start time, completion time and instant of deadline(absolute) using a time path diagram**

**Zero-/reference point**

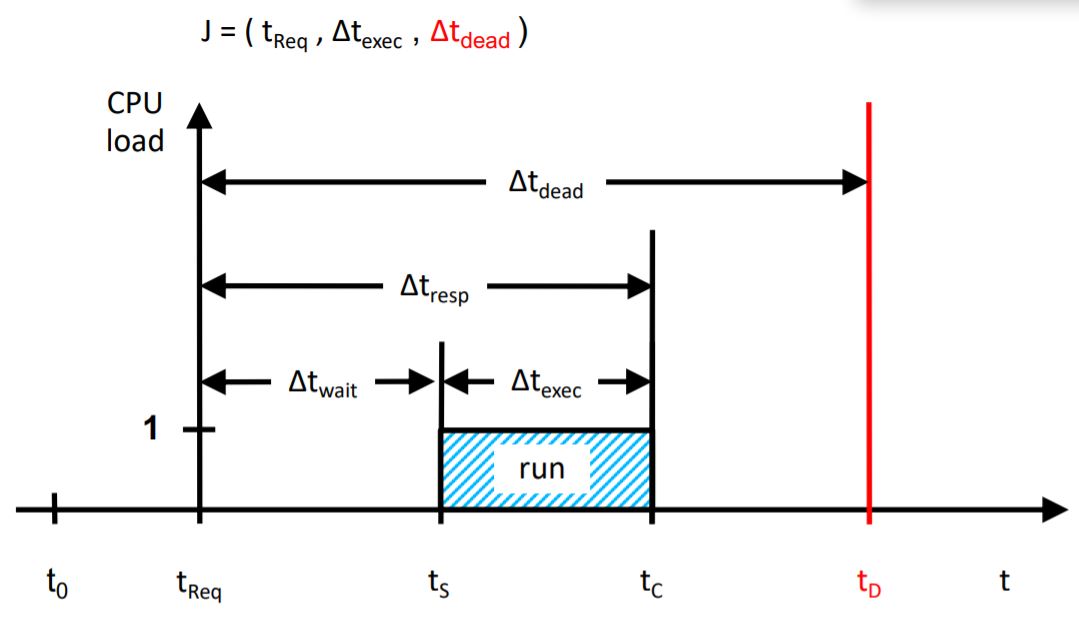
**Request time (Δtexec)**

****

**Start and completion time**

****

**Instant of deadline(absolute)**

****

**5.You can explain the duarations: waiting time, execution time, response time, period(and rate), relative deadline, worst/average/best case execution time, slack time, tardiness, feasable interval**

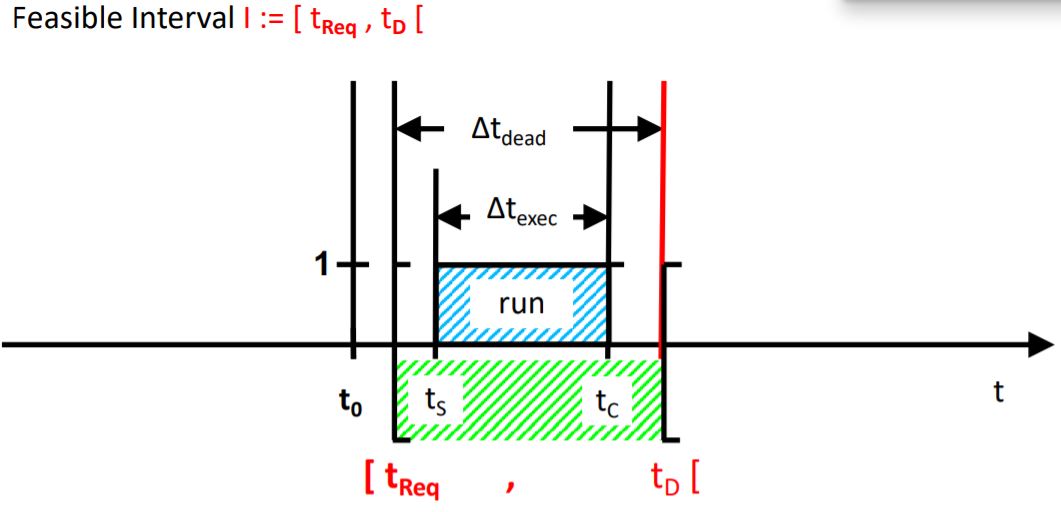
**Response time –** minimize the time for systems tasks

**Execution time** is the duration of a specific Job between Job request and Job completion.

* Minimal net execution time (best case)
* Maximum net execution time (worst case)
* Average net execution time

**Feasable interval**

The start time is then after the request time and the completion time must be before the absolute deadline or **tReq ≤ tS ≤ tC < tD**

****

**6.Why ist the WCET so important for real time systems?**

Relevant for deadlines is the maximum execution timeof a program on a certain hardware.

The WCET is the maximum limit of the variable execution time.

**7.How can you determine a value fort he WCET (of specific peace of software on a predefined system)**

There are two major approaches:

1.macroscopic approach (empirical evidence)observe and measure the execution times of you program --> derive the WCET

2.microscopic approach (reductionism)analyze the small pieces (CPU execution cycles) and put it together--> calculate the WCET

**8.How can you determine a value for the WCET for a program that contains branches?**

**Two different levels:**

1.macroscopic examination:What does a computer program do?

2.microscopic examination:What happens inside the microprocessor?

**Two methods:**

1.Dynamical WCET Analysis: (deductive, let's measure) Measurement of an fair number of execution runs on a certain hardware

2.Static WCET Analysis: (inductive, let's think)Calculating execution time based on the computer program.

**9.You can explain the functionality of world clocks, alarm clocks, stop watches and timers**

**World Clock/Weltzeit Uhr**

Gives us just the actual UTC in our local time zone.

This time is a point in time!

This point in time is an absolute time!

**Alarm Clock/Wecker**

You can set an absolute alarm time.

The alarm time is a point in time.

There is a time-based trigger:

The alarm clock compares the actual (point in) time with the alarm (point in) time.

When the alarm time is lower or equal the actual time the alarm clock triggers a sound.

**Stop Watch/Stoppuhr**

There is nothing to set.

The reference point in time is set to 00:00.00 s (also: 00' 00" 00‰ or 00' 00" 000‱)

There is a manual/electronic trigger to start the stop watch.

The stop watch is counting upwards with the SI-second or parts of it.

It displays the relative time difference based on the trigger time point.

There is a manual/electronic trigger to stop the stop watch.

After this stop trigger the stop watch stops the clock.

The display shows the time difference (duration) between:start event and stop event.

**Stop watch with lap times/Rendenzeiten**

A stop watch with lap times has the following additional functionality:

Between start time and stop time you can have additional events.

These events are lap times.

When a lap is terminated the event initiates an additional time stamp.

This time stamp is a relative pointin time.

If you calculate the time difference between two absolute subsequent lap time-stamps.

You get the time difference(duration) of the preceding lap.This duration is called lap time.

**Concurrency and scheduling**

**1.You can explain the advantages and drawbacks of concurrency**

**2.You can list at least four potential problems of concurrency (deadlock, livelock, race condition, priority inversion) and elaborate on them**

Deadlocks: One wait for another

Livelocks: One triggers the other (distributed enless loops)

Race conditions: The output is dependent on the sequence or timing of other uncontrollable events

**3.You can explain the terms online and offline scheduling**

|  |  |
| --- | --- |
| **Online-Scheduling** | **Offline-Scheduling** |
| Inflexible (changes) | Flexible (changes) |
| Always maximum use | Use depends on scheduling algorithm |
| Low costs at execution | Higher cost at execution |
|  | Normally priority based |

**4.You can explain static and dynamic priorities**

|  |  |
| --- | --- |
| **Static Priority** | **Dynamic Priority** |
| EDD (Earliest Due Date First) | EDF (Earliest Deadline First) |
| RMS (Rate Monotonic Scheduling) | LRT (Latest Release Time) |
| DMS (Deadline Monotonic Scheduling) |  |

**5.You can perform schedulability tests (see calculations) and interpret the results**

**Schedulabilitytest**

There are two major types of test:

• necessary tests (notwendig)

• sufficient tests (hinreichend)

**Necessary means:** • if one of the appropriate necessity test fails then there is no feasible schedule! **Sufficient means:** • if you find at least one sufficient necessity test, than the task package is feasible schedulable.

**Load test**

Load U = 1 means the processor never idles Load U > 1 there is no feasible schedule Load U < 1 means: this test does not exclude that a feasible schedule may exist

**6.Determination of task priorities for RMS**

**7.You can name at least four approaches to solve concurrency issues (mutex, semaphore, PIP, PCP)**

Race conditions can be avoided by allowing access to a critical section for one task only at a time. This is called mutual exclusion (gegenseitiger Ausschluss).

The mathematical instrument to solve this issue is called a semaphore.

A Semaphore with N=1 is called Binary Semaphore or a Mutex.

It tells you only free or not free and the size of the queue respectively.

**8.You can explain the priority inversion problem**

Priority Inversion occurs when a high priority task - requiring a resource - is blocked due to the lock of this resource by a low priority task.

Then the high priority task is blocked until the low priority task is completed and has released the resource.

This can take a long time, especially when mid-prioritized tasks delay the completion of the low prioritized one.

**9.You can explain how the priority inversion problem can be solved using the priority inheritance protocol (PIP)**

Avoids blocking by Priority Inheritance however issues like chaining („cascaded delays“) or deadlocks.

**10.You can explain how the priority inversion problem can be solved using the priority ceiling protocol (PCP)**

Enhancement of the PIP without the issues.

**Petri Net**

**1.You can name the elements (places, tokens, transitions, edges) of Petri-nets and explain their interactions**

1. **Places (Stellen)**

* Places represent states (Zustände).

1. **Transitions(Transitionen)**

* Transitions represent actions or event (Ereignisse)

1. **Edges (Kanten)**

* Arcs (flow relations) connect places and transitions where:
  + An arc is directed either from place to transition or  
    from transition to place
  + An arc never connects a transition with a transition  
    an arc never connects a place with a place

1. **Tokens (Marken)**

* Marks can move from place to place via a transition along the related arcs, but only when the predefined switch condition is fulfilled.
* Marks are also called tokens:
* A switch of tokens are an atomic action (uninterruptible)
* A switch takes no time (instantaneous)  
  (instantan = sofort und unmittelbar)

**2.You can explain capacities of places**

The capacity of a place is the maximum amount of tokens that a place can cover.

Default capacity is ∞

**3.You can explain multiplicities of edges**

A Petri net graph is a Petri net structure as a bipartite directed multigraph. Usually, in the graphical representation, parallel arcs connecting a place (transition) to a transition (place) are represented by a single directed arc labeled with its multiplicity, or weight k. A circle contains a dot represents a place contains a token.

**4.You can create a reachability graph for a given Petri-net**

The reachability graph of a Petri net is the part of the transition system reachable from the initial state in graph-like notation.

The reachability graph can be calculated as follows:

1. Let X be the set containing just the initial state and let Y be the empty set.

2. Take an element x of X and add this to Y. Calculate all states reachable for x by firing some enabled transition. Each successor state that is not in Y is added to X.

3. If X is empty stop, otherwise goto 2.

**Folie 33, 34**

**5.You can identify deadlocks**

* **According to Edward Coffman the conditions are:**

1. Mutual exclusion

2. Hold and wait

3. No preemption

4. Circular wait

**6.You can determine the properties of a given Petri net (boundedness, safe/unsafe and dead/alive**

**liveness mean for Petri Nets?**

* The Petri Net is a deadlock-free system.
* The concept of liveness is closely related to the deadlock situation, which has been situated extensively in the context of computer operating systems. A Petri net modeling a deadlock-free system must be live.

**safeness mean for Petri Nets?**

* A Petri Net is safe if there are no overflows. Meaning there is no possibility that a token is blocked from transitioning because the next place is full.

**reachability mean for Petri Nets?**

* An important issue in designing event-driven systems is whether a system can reach a specific state, or exhibit a particular functional behavior.

**Embedded System/Sensors/Actors**

**1.What does the technical term “embedded systems” describe?**

An Embedded System is a computer that is a part of a overall system (*der Teil eines Gesamtsystems ist*) and controls physical processes.

**2.Example of embedded systems**

Computer is embedded in a technical system

You don’t see a computer in an embedded system, for example pocket calculator, or a washing machine.

Repair costs are often higher than a new system, for example when your pocket calculator is defect you throw it away and buy a new one.

**3.Name examples of sensors**

A microphone converts an alternating air pressure into electrical alternating voltage. An analog/digital-converter (ADC) digitalizes the alternating voltage to digital Data.

**4.Name examples of actors**

**5.Differnce between active-low/active-high and its application when controlling actors like LEDs**

**6.Explain the terms ASIC and SoC**

SoC-System on the chip.

ASIC-

**7.Components of Microcontrollers**

**3 components:**

* processes core,
* access -input/output,
* to processing external data

**8.Resolution of analog-digital-converters**

Analog to Digital Conversion: Digital computers can not process analog signals. Therefore, analog signals need to be converted to digital form.

Analog signals can be converted to digital form using a circuitry whose block diagram.

**9.Quantization error**

**10.Sampling theorem**

Sampling frequency: needs to be at least twice the maximum of the recorded signal (Nyquist Theorem) therefore we need a low pass filter(Skript Sensing, Folie 5)

**11.Antialiasing**

**Real time operating systems**

**1.You can explain the terms(computer-)program, process and thread/LWP**

Process: -Process can be divided into multiple threads - Each process has its own memory space Threat: -Threads cannot be sub divided - It is easy to create a thread

**2.You can explain the functionality of a scheduler using a state diagram/how does a scheduler operate within an operating system**

**3.What are the different golas of standard and real time schedulers**

**4.You know the difference between preemptive and cooperative multitasking**

**5.You can describe the functionality of interrupts**

An interrupt is an event that stops a currently running program A, starts another program B and returns to the program A that is continuing exactly where it was interrupted.

**6.You can list the different categories of interrupts(hw/sw, (non-/)maskable) etc.)**

**Hardware interrupts** are used by devices to communicate that they require attention from the operating system. Hardware interrupts are asynchronous.

**A software interrupt** is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed.

**Non-maskable interrupt (NMI):** a hardware interrupt that lacks an associated bit-mask, so that it can never be ignored. NMIs are used for the highest priority tasks such as timers, especially watchdog timers.

**7.You can explain the operating sequence during a context switch**

**8.What does a MMU do?**

**9.What is „minor page fault“ and why must they be prevented in hard RTS?**

If the page is loaded in memory at the time the fault is generated, but is not marked in the memory management unit as being loaded in memory, then it is called a minor or soft page fault.