§ 4 Real-time programming

© 2004 IAS, Universität Stuttgart

Chapter 4 - Learning targets

- to know what is meant by real-time programming
- to know the requirements of real-time programming
- to be able to differ between hard and soft real-time
- to understand what is meant by synchronous programming
- to understand the asynchronous programming
- to be able to explain what tasks are
- to know how time synchronization can be done
- to be able to use semaphores
- to know what scheduling methods are
- to be able to use the different scheduling methods
- to know what a schedulability test is

© 2004 IAS, Universität Stuttgart

§ 4 Real-time programming

© 2004 IAS, Universität Stuttgart

What is real-time programming?

Non-real-time computing

- Correctness of the result

Real-time computing

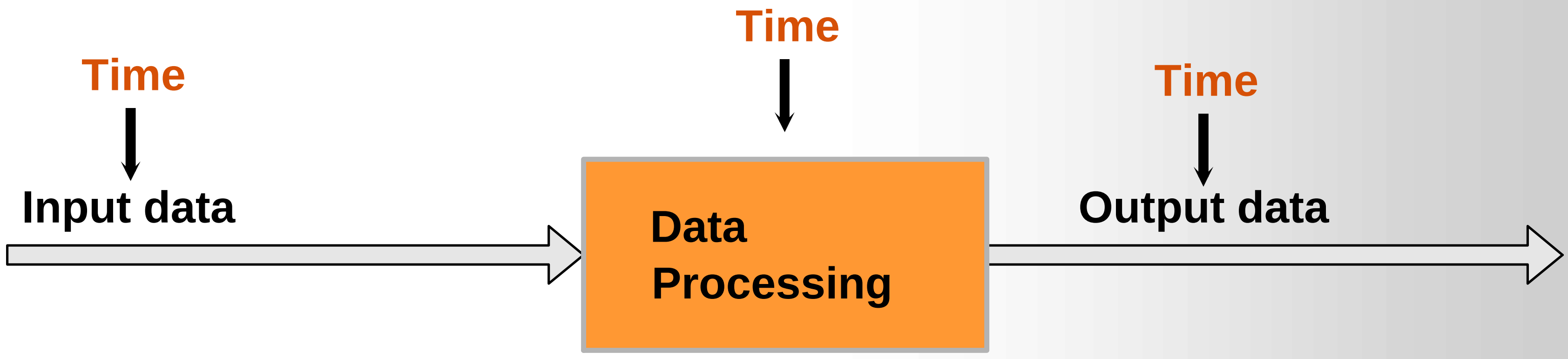
- Correctness of the result
- Timeliness of the result

Not too soon, not too late

© 2004 IAS, Universität Stuttgart

NON-REAL-TIME Computing :

REAL-TIME Computing :



© 2004 IAS, Universität Stuttgart

Real-Time Programming

Creation of programs in such a way that the time requirements on the compilation of input data, on the processing and on the delivery of output data are fulfilled.

time requirements

Time requirements are dependent on the processes in the technical system

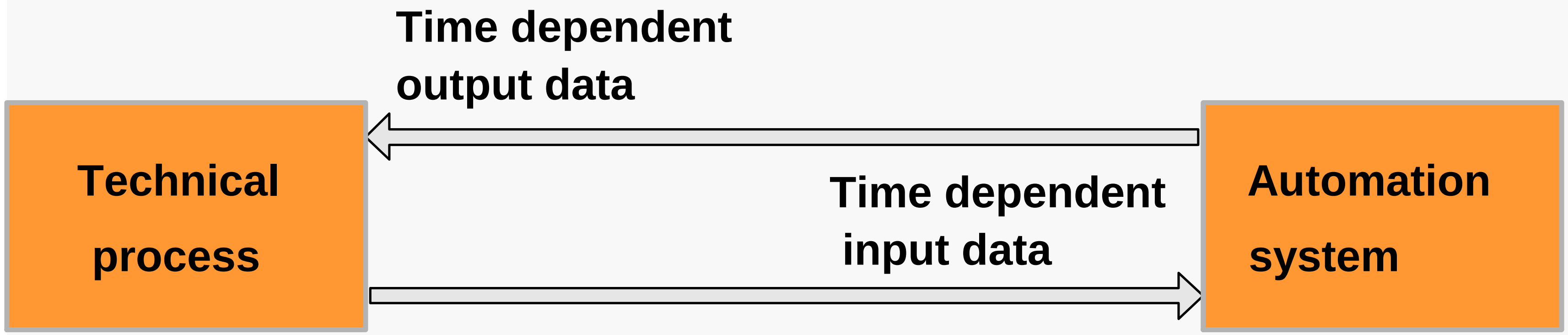
coordination with the technical process

Real-time : according to the real-time sequences, no time expansion, no time compression

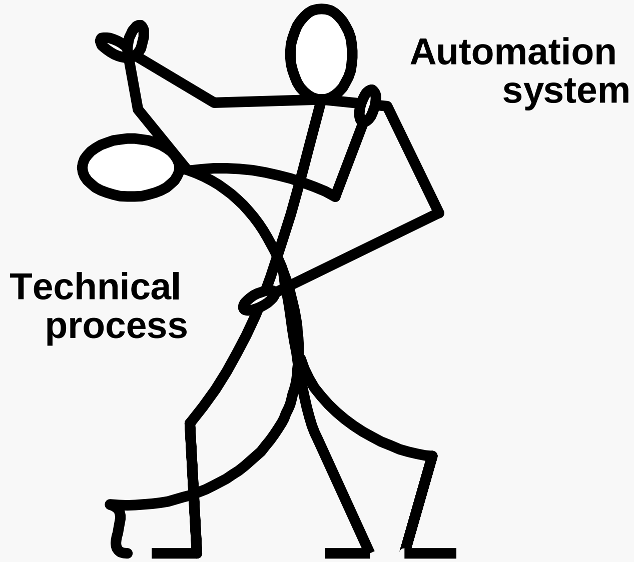
Different kinds of requirements on the time behavior of the data processing

- Requirements for timeliness
- Requirements for concurrency

Interaction technical process and automation system



Example:



malfunction due to too early/ too late arriving sensor/actuator values

Differences between information and real-time systems

Information systems	Real-time systems
data-driven system	event/time driven system
complex data structure	simple data structure
great amount of input data	small amount of input data
I/O-intensive	computation-intensive
machine-independent	machine-dependent

© 2004 IAS, Universität Stuttgart

Important terms (1)

Reactive systems

Real-time systems that react to the input signals of the technical process and deliver output signals to influence the technical process

Example: Automation systems

Embedded systems

Integration of the automation system in the technical process physically and logically.

Examples: electric razor, mobile telephone, washing machine, power drill

© 2004 IAS, Universität Stuttgart

Important terms (2)

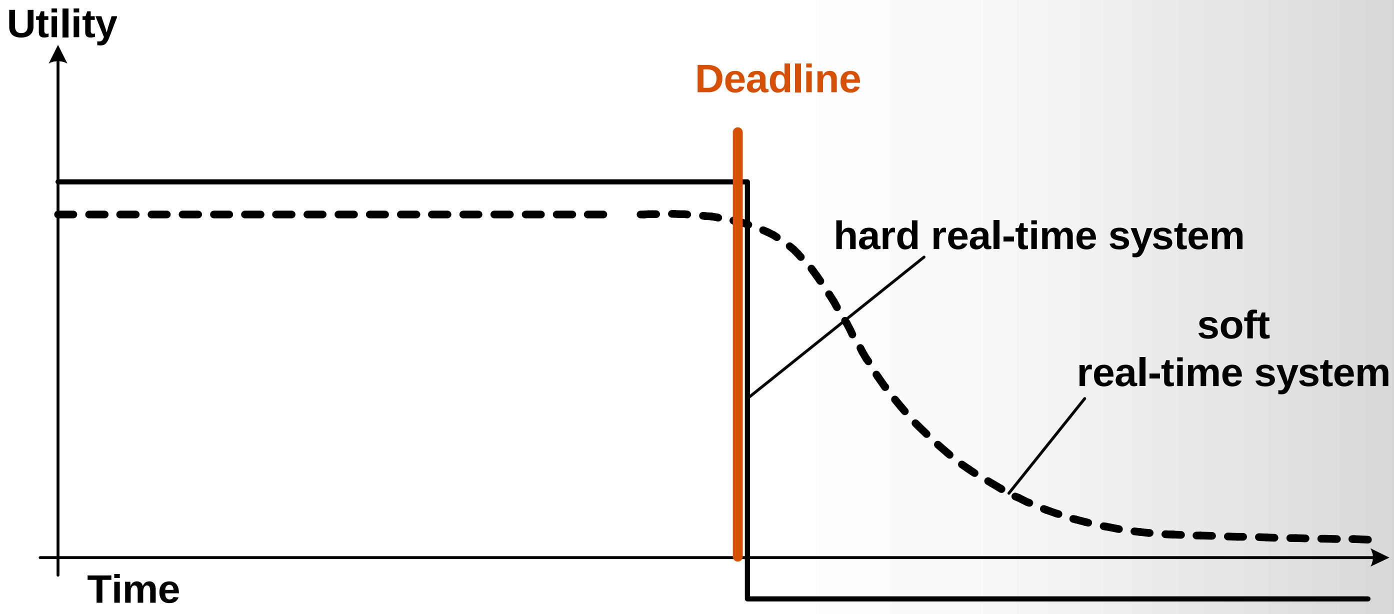
Hard real-time systems

Strict deadlines must not be missed in any case

Examples: **DDC- control in airplanes, motor control in automobiles**

Soft real-time system

A violation of deadlines can be tolerated



Requirements for timeliness

– timely compilation of input data

– timely data processing

– timely delivery of output data

Time requirements regarding timeliness

– **Absolute time requirements**

e.g..: 11:45 signal for departure

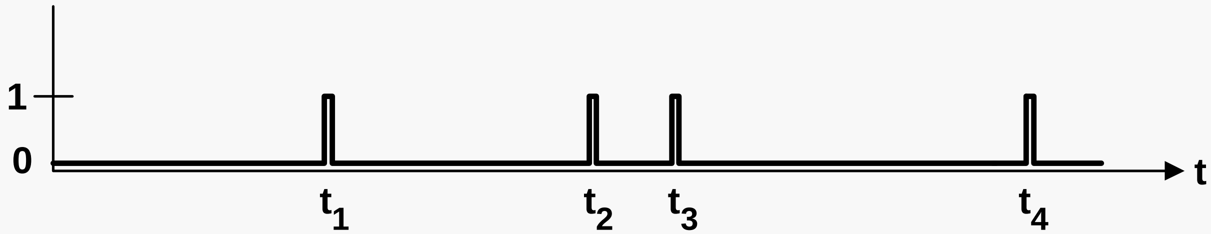
– **Relative time requirements**

e.g..: Turn off signal 10 seconds after a measured value exceeds its threshold

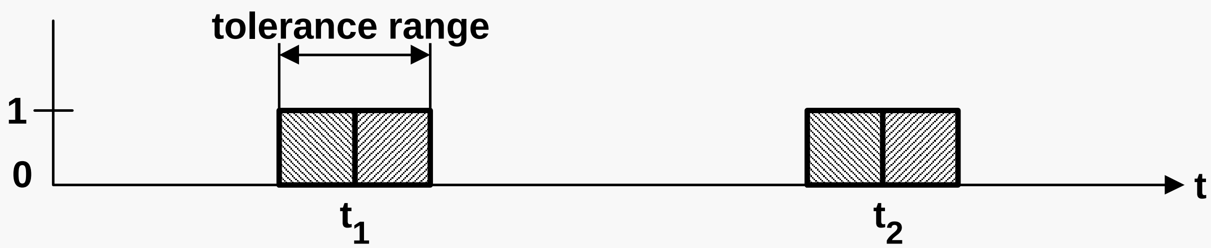
© 2004 IAS, Universität Stuttgart

Classification of time requirements (1)

Execution of a function at fixed times t_1, t_2, t_3 , t_4



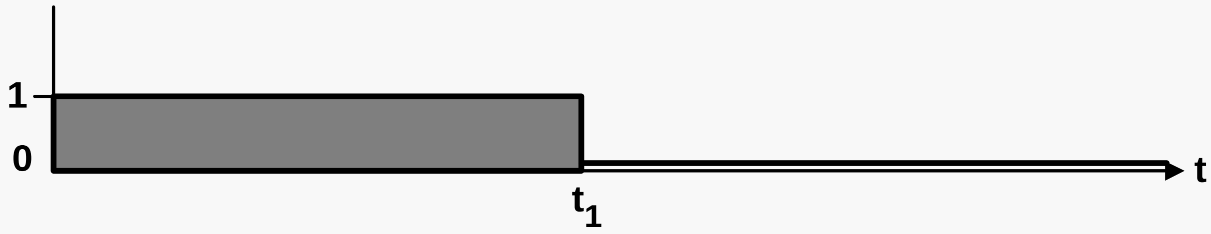
Execution of a function within a tolerance time range assigned to each time t_n



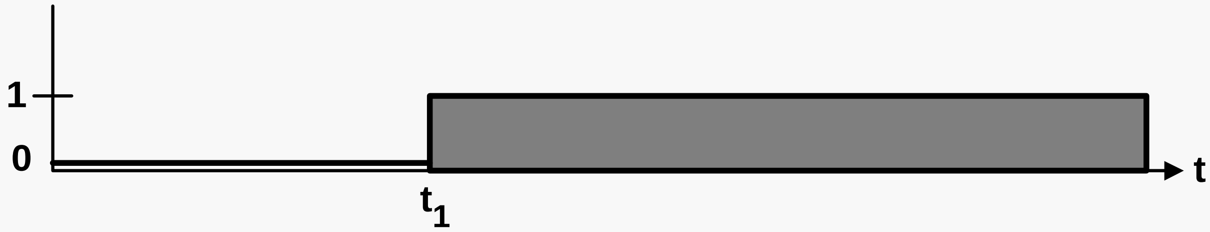
© 2004 IAS, Universität Stuttgart

Classification of time requirements (2)

Execution of a function within an interval up to a latest time t_1



Execution of a function within an interval starting from an earliest time t_1



© 2004 IAS, Universität Stuttgart

Typical examples of applications with time requirements

	Absolute time requirements	Relative time requirements
Execution of a function at a fixed time	Recording of test values	Analysis of substances in chemistry
Execution of a function within a tolerance time range	Recording of controlled variables	Measured value supervision on sliding boundaries
Execution of a function within an interval up to a latest time	Recording of data telegrams	Recording of cargo indicator labels
Execution of a function within an interval starting from an earliest time	Sequence control in batch processes	Recording of signals of a light barrier

© 2004 IAS, Universität Stuttgart

Requirements for concurrency

Processes in the “real world” take place simultaneously

è Real-time systems have to react to that simultaneously

è Several computation tasks have to be executed simultaneously

Examples:

- Reaction to simultaneous trips of several trains
- Processing of several simultaneously occurring measurements in heating systems
- Control motor and ABS system simultaneously

Realization of concurrency

– Each computation task is processed on a separate computer

real parallelism

– One computer for all data processing tasks

quasi parallel

Prerequisites:

Processes in environment are slow in comparison to the computation of the programs on the computer

© 2004 IAS, Universität Stuttgart

Requirements for determinism

Determination = predictability of the system behavior

Parallel and quasi- parallel sequences are generally not predictable; time shifts can lead to different execution sequences.

– the time behavior is not deterministic

– no guaranty of the safety of automation systems

© 2004 IAS, Universität Stuttgart

Deterministic real-time system

A real-time system is called deterministic if for each possible state and for each set of input information there is a defined set of output information and a defined following state

Prerequisites: a finite set of system states

Timely deterministic system

The response times for all output information are known

In a deterministic system it is guaranteed that the system can react at every time. In a timely deterministic system it is additionally guaranteed at which time the reaction will have taken place.

Dialog systems

- Input data from input mediums
  - Keyboard
  - Light pen
  - Mouse
- Waiting for reply, i.e. output of results on an output medium
  - Screen
  - Printer

Examples of dialog systems

- Seat reservation systems of airline companies
- Account management in banks
- Storekeeping systems

Timeliness in dialog systems **Permitted response time within the range of seconds**

© 2004 IAS, Universität Stuttgart

Automation systems

Time reaction depends on the processes in the technical system

Timeliness in automation systems

Permitted response time within the range of milliseconds /microseconds

Methods for real time programming are similar for

- Automation systems
- Dialog systems

© 2004 IAS, Universität Stuttgart

§ 4 Real-time programming

© 2004 IAS, Universität Stuttgart

Different methods

– **Synchronous programming:**
  Planning of the time sequence of the execution of programs

  planned economy

– **Asynchronous programming (parallel programming):**
  Organization of the time sequence during the execution of the programs.

  market economy

© 2004 IAS, Universität Stuttgart

e.g.: Dentist's office as real-time system (chalkboard writing)

Synchronous programming method

Synchronous programming:
Planning of the time behavior of subprograms that have to be executed cyclically, before their actual execution

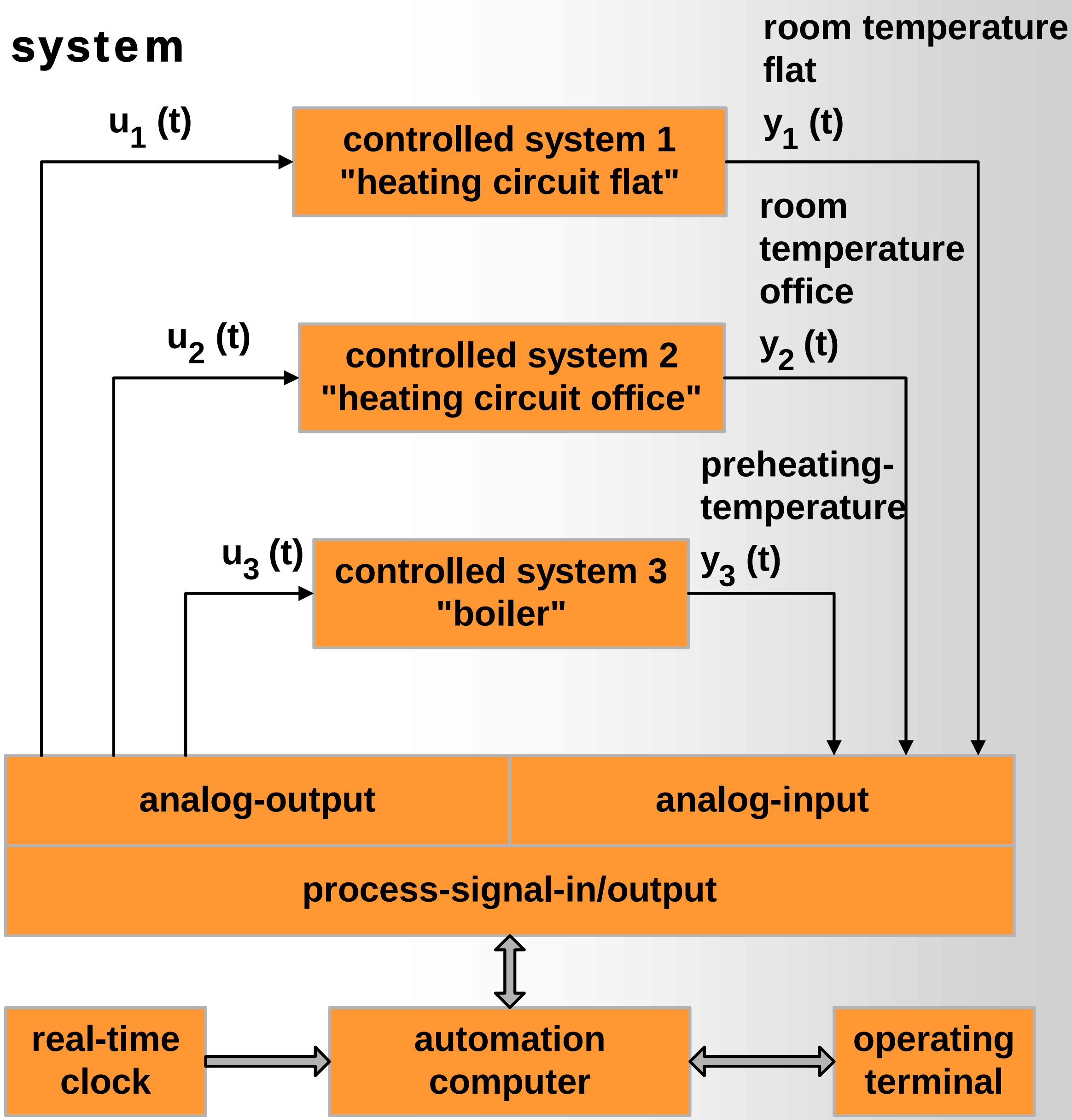
- Synchronization of the cyclical subprograms with a time grid
- Time grid generated by a real-time clock, interrupting signal for the call of subprograms

Time triggered

- Strictly predefined sequence of the execution of subprograms

© 2004 IAS, Universität Stuttgart

Example: heating system



Technical (control engineering) conception

Planning

– Design and calculation of the control algorithms and control parameters

– Determination of the sampling time for the control circuits

T	=	real-time clock - cycle time
T_i	=	sampling time for control circuit i
T_1	=	T
T_2	=	$2T$
T_3	=	$5T$

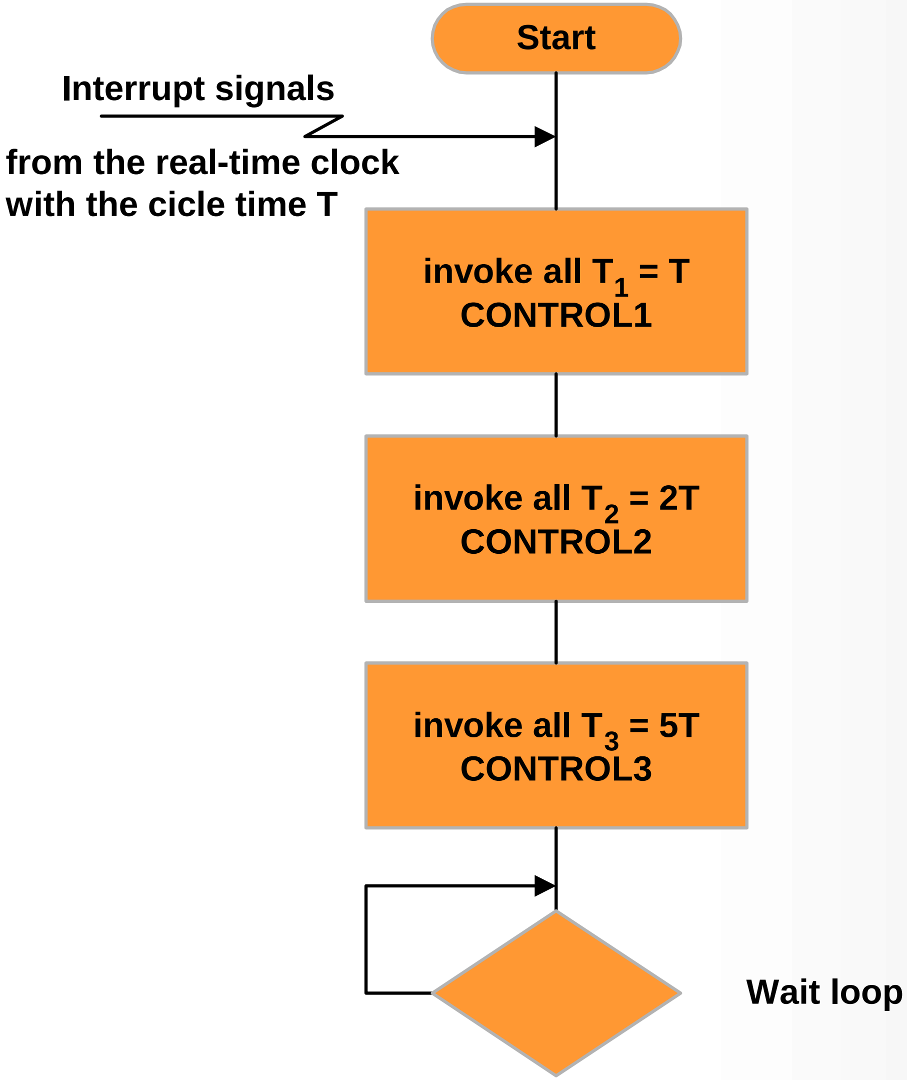
© 2004 IAS, Universität Stuttgart

Assignment of the identifiers and sampling times to the subprograms (control programs)

a subprogram for each control circuit

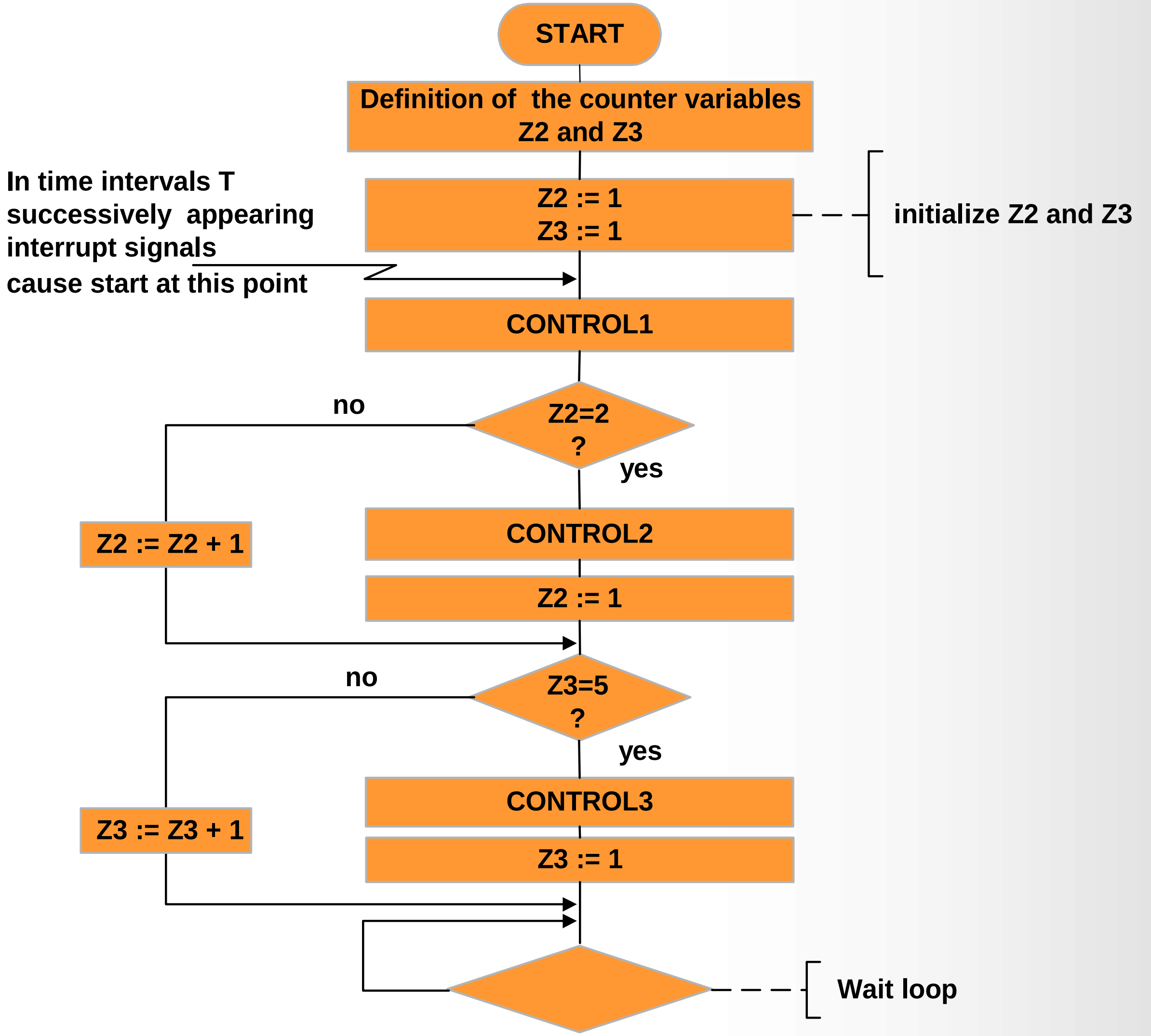
Subprogram	Identifier (name)	Sampling time (cycle time)
Temperature control for subsystem “heating circuit: flat“	CONTROL 1	$T_1 = T$
Temperature control for subsystem “heating circuit: office“	CONTROL 2	$T_2 = 2T$
Temperature control for subsystem “heating circuit: boiler“	CONTROL 3	$T_3 = 5T$

Preliminary design with synchronous programming

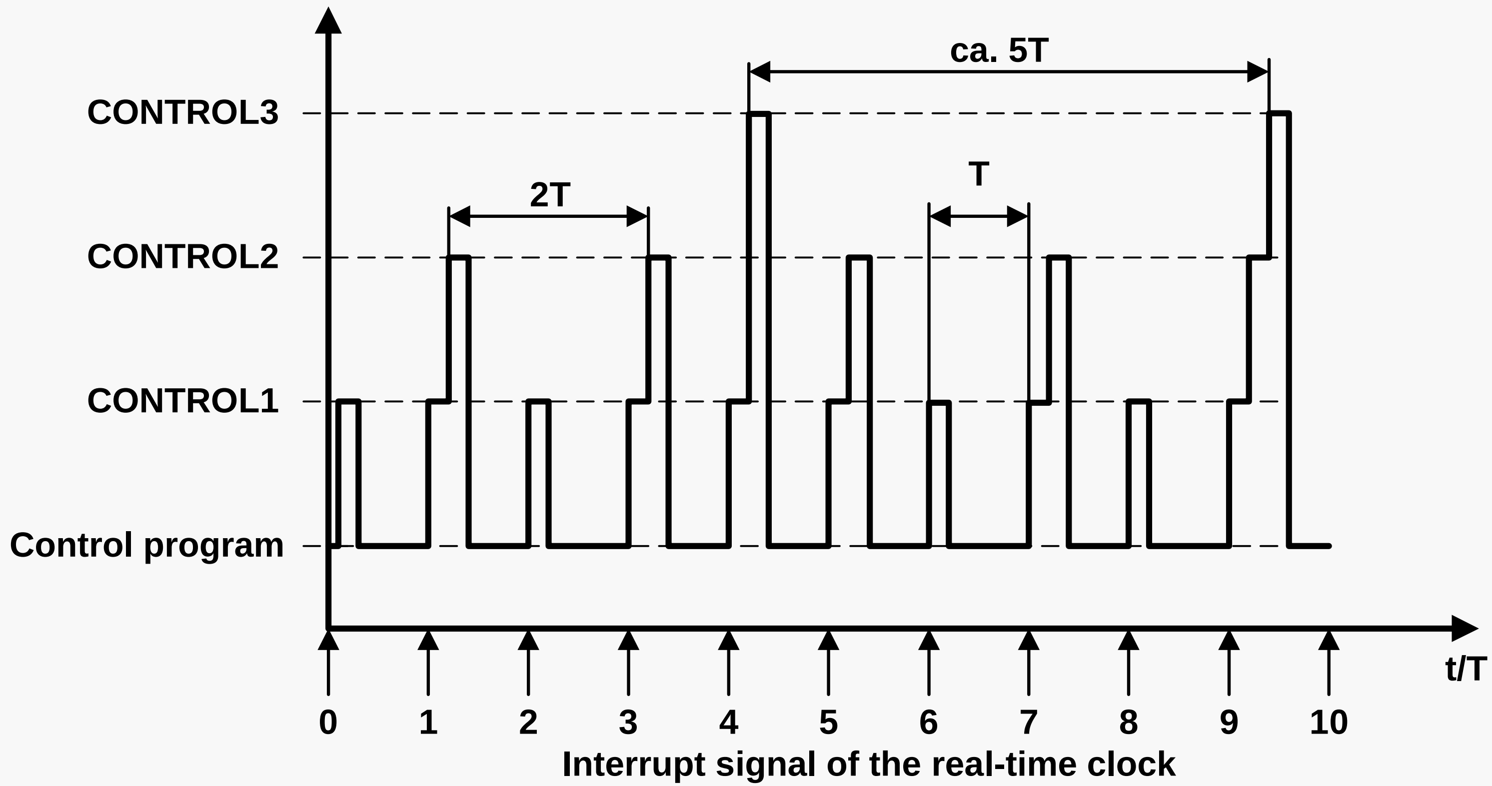


© 2004 IAS, Universität Stuttgart

Final design of the control program according to the synchronous programming method



Time sequence of the synchronous programming method



Assumption:

- Computation time for subprograms is identical
- Sum of computation times of the three subprograms is smaller than cycle time

© 2004 IAS, Universität Stuttgart

Characteristics of the synchronous programming method (1)

– Requirement for timeliness is approximately fulfilled

slight shifts

– Requirements for concurrency is fulfilled, if cycle time T is small compared to the time constants in the technical process

– Synchronous programming is good for real-time systems with cyclic program execution

predictable behavior

– Synchronous programming is not suitable for the reaction on timely non-predictable (asynchronous) events
  - Increase of computation time through constant polling
  - Delay of the reaction

© 2004 IAS, Universität Stuttgart

Characteristics of the synchronous programming method (2)

– Normally deterministic behavior

– No complex organization program

– A bit more expensive in its planning (development)

Disadvantages of synchronous programming :

Modification of the requirements specification causes modification of the program structure

e.g.: PLC-programming

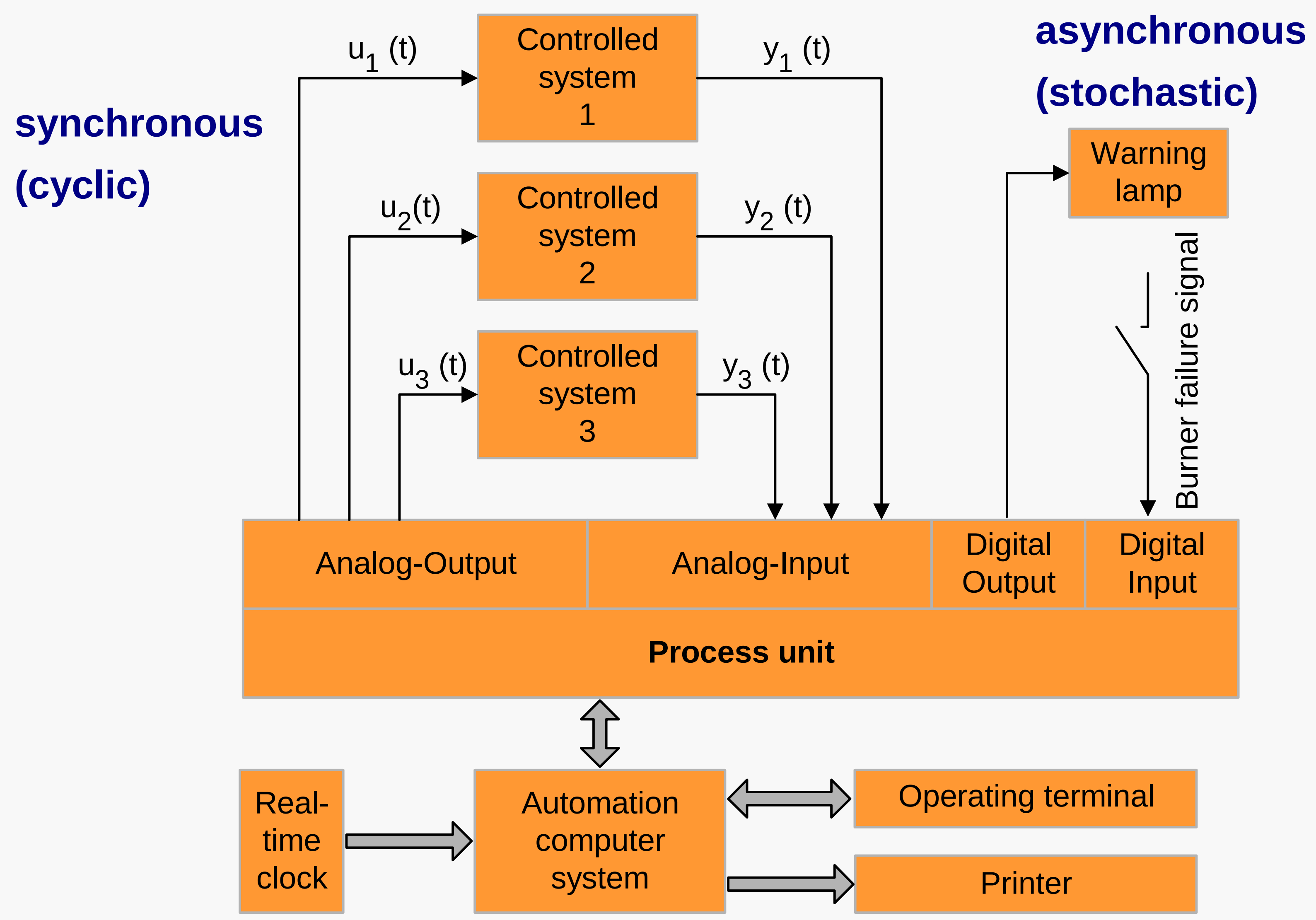
Asynchronous programming method (parallel programming)

Organization program (real-time operating system) controls the timely execution of subprograms at run-time

- Execution of subprograms, when time requirements are fulfilled
- Simultaneous execution is sequentialized according to a certain strategy
  - Assignment of priority numbers
  - The priority is the higher, the lower the priority number is

© 2004 IAS, Universität Stuttgart

Example: heating system



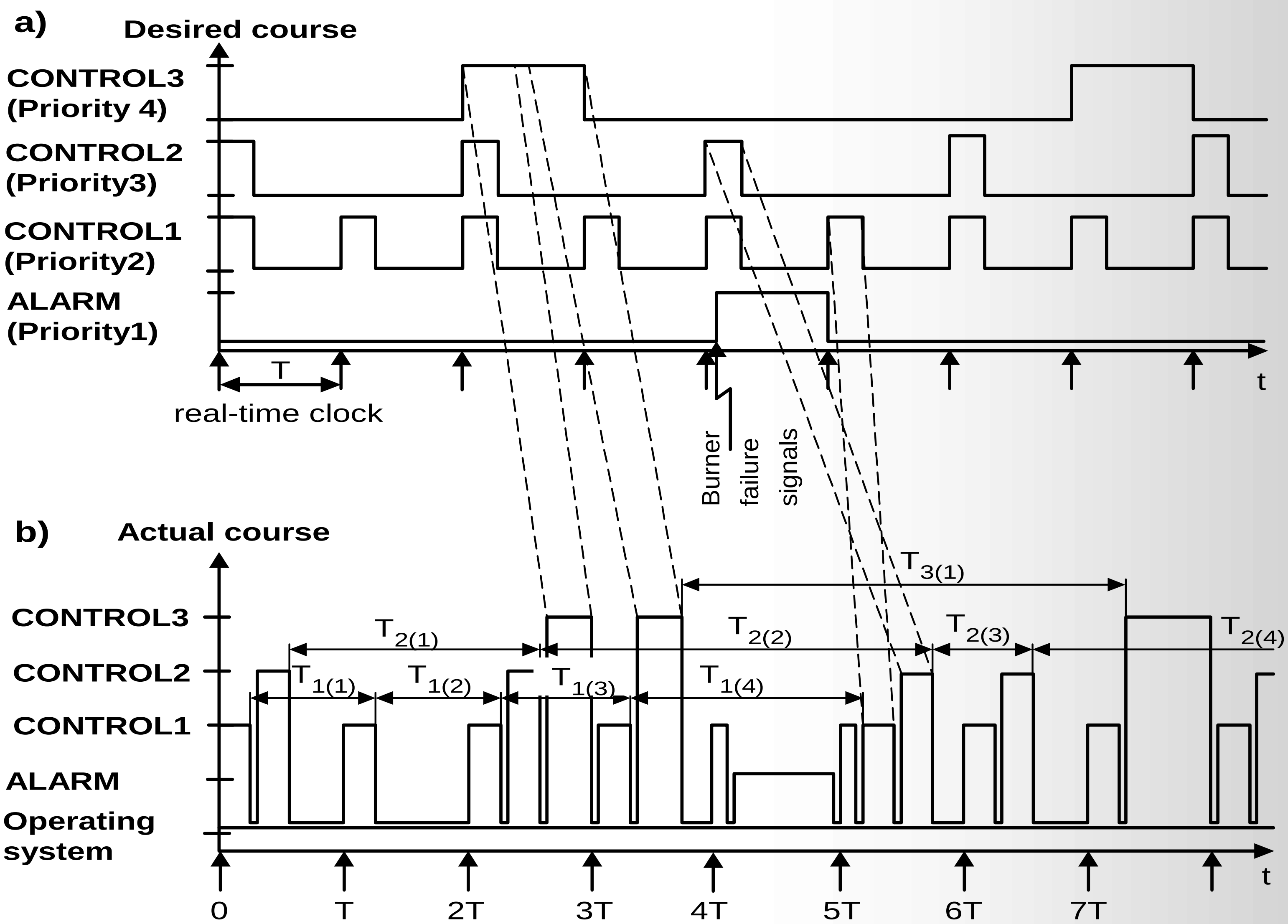
© 2004 IAS, Universität Stuttgart

Assignment of indicators, sampling times and priority

Subprogram	Name	Sampling time	Priority number	Priority
Reaction on burner failure with alarm message	ALARM	–	1	highest
Temperature control unit for heating circuit 1	CONTROL 1	$T_1 = T$	2	second highest
Temperature control unit for heating circuit 2	CONTROL 2	$T_2 = 2T$	3	third highest
Temperature control unit for heating circuit 1	CONTROL 3	$T_3 = 5T$	4	lowest

© 2004 IAS, Universität Stuttgart

Time sequence of the four subprograms



Characteristics of the asynchronous programming method

– Requirements for timeliness only approximately fulfilled

Bad for low priority subprograms

– Time requirements are fulfilled the better the higher the priority of the according subprogram

– Actual time sequence can shift away from the desired time sequence

Subprograms can pass each other mutually

– The succession of the subprograms is not deterministic, but occurs dynamically

– At the program development it cannot be indicated in advance which of the subprograms will run at which point of time

- **simple development**
- **complexity in the administration program**
- **program sequence is not transparent**

Event-driven architectures — Asynchronous programming

- all activities as sequences of events
  - activation of tasks
  - transmission of messages
- support through real-time operating systems
- non-deterministic behavior
- flexible regarding modifications

Time-driven architectures — Synchronous programming

- periodical execution of all tasks and communication actions
- sampling of external variable at determined times
- low flexibility in case of modifications
- easy to analyze

PLC-systems are time-driven real-time systems

§ 4 Real-time programming

© 2004 IAS, Universität Stuttgart

Distinction

- Program (sequence of commands)
- Execution of the program (a single execution of command sequence)
- Task

Invocation of subprograms

- Execution of the invoking program is interrupted
- Execution of the subprogram
- Continuation of the invoking program

Invocation of a task

- Simultaneous execution of the invoking program and of the invoked task

Task

a procedure of the execution of a sequential program controlled by a real time operating system

Introduction of the term computation task

- Task starts with entry in a list of the real-time operating system and ends with the deletion from that list
- Task does not exist only during the execution of the commands, but also during planned and forced waiting times

© 2004 IAS, Universität Stuttgart

Differences between task and thread

Task	Thread
Owner of resources	Cannot own resources besides the processor’s, accesses all resources of the task, to which it belongs
Own adress space	Adress space of the task, to which it belongs **Common address space**
Contains one or several threads	Element of a task
Communication beyond the task boundaries, preferred via messages	Communication between the threads, preferred via shared data

© 2004 IAS, Universität Stuttgart

4 basic states

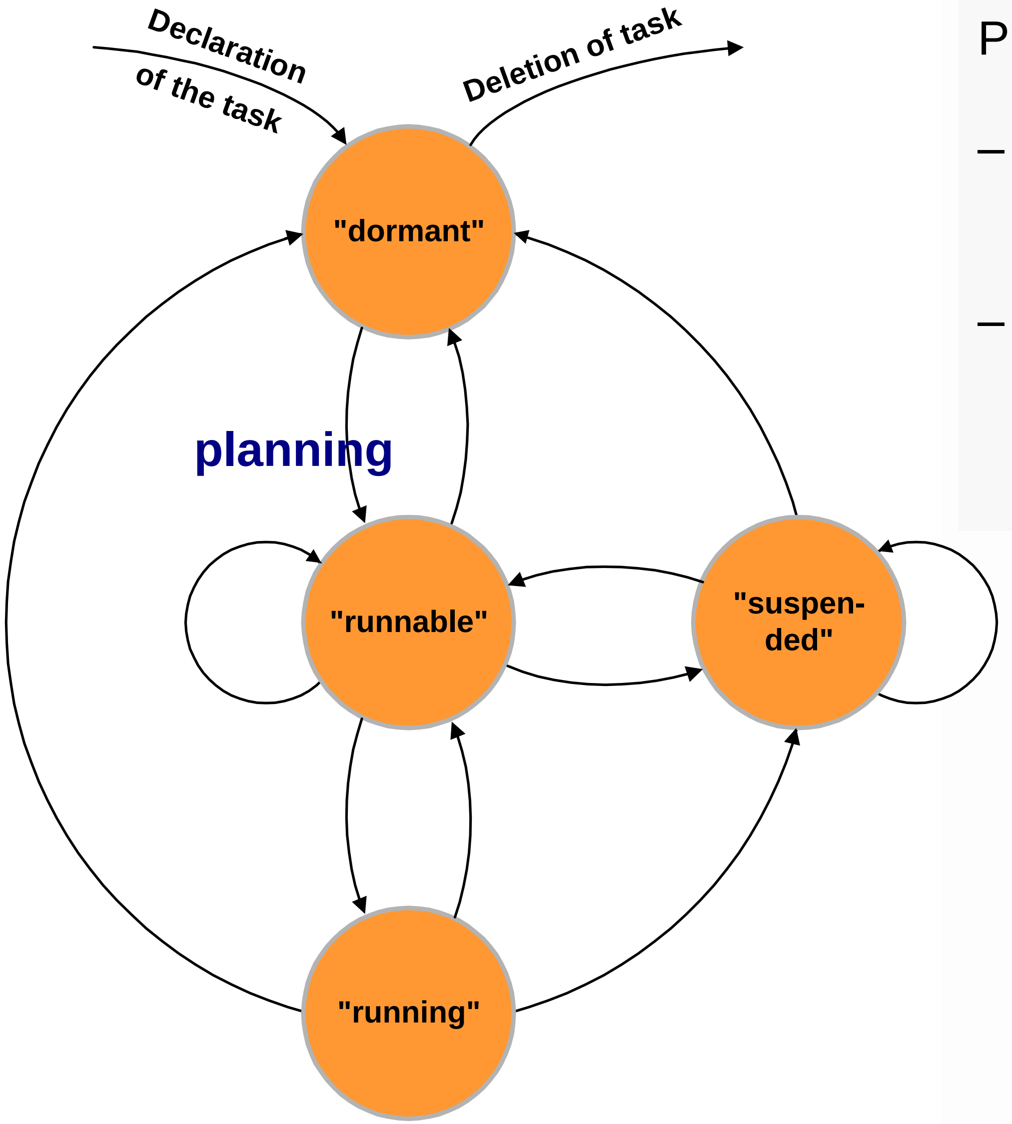
- State “running”
  - the subprogram is processed
- State “runnable” or “ready”
  - all time conditions for the process are fulfilled
  - what is missing is the start from the operating system

i.e. the execution

- State “suspended”
  - task is waiting for the occurrence of an event
  - as soon as event occurs transition from state “suspended” to “ready”
- State “dormant”
  - task is not ready because time conditions or other conditions are not fulfilled

© 2004 IAS, Universität Stuttgart

State diagram of a task

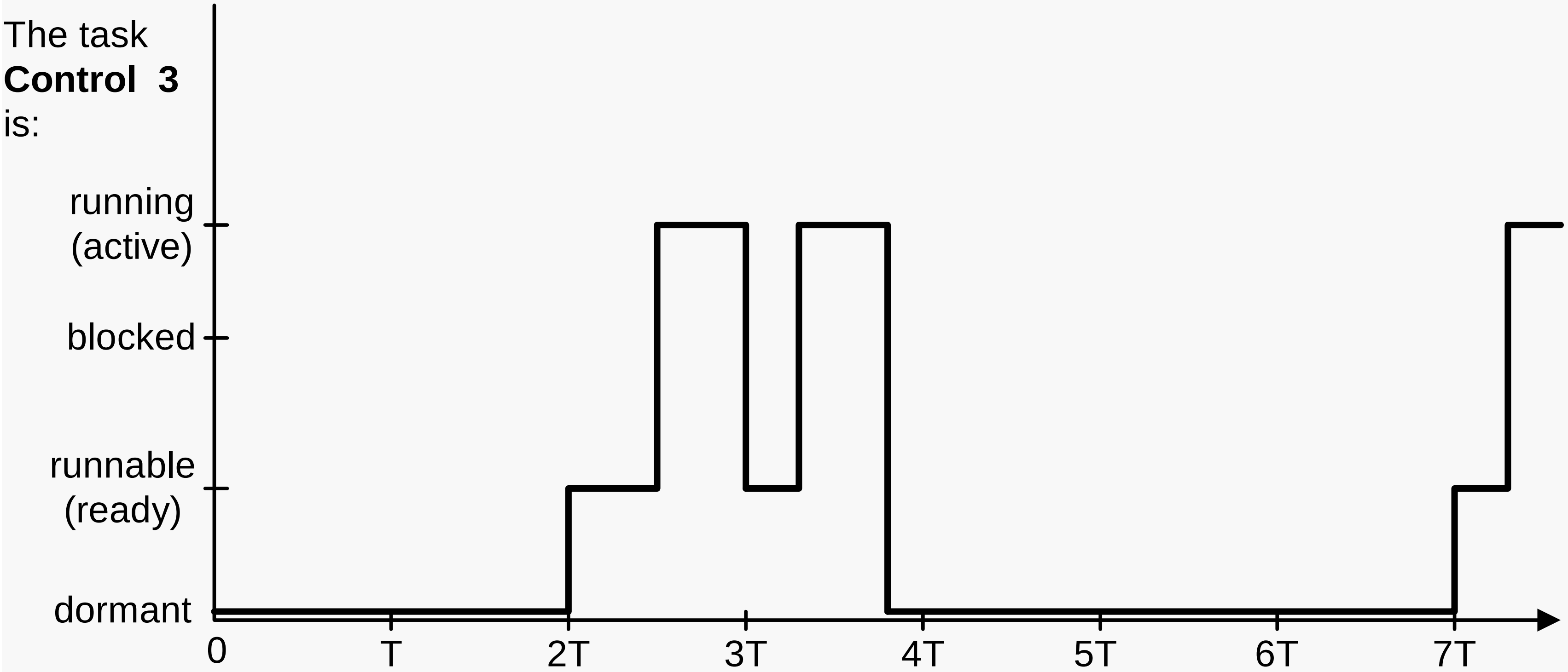


Planning of tasks

– Invocation of a task cyclically or at certain times

– “Planning” is the transition from the state “dormant” into the state “runnable”

Course of the task “Control 3” in the asynchronous programming method



© 2004 IAS, Universität Stuttgart

Assignment of priorities for tasks

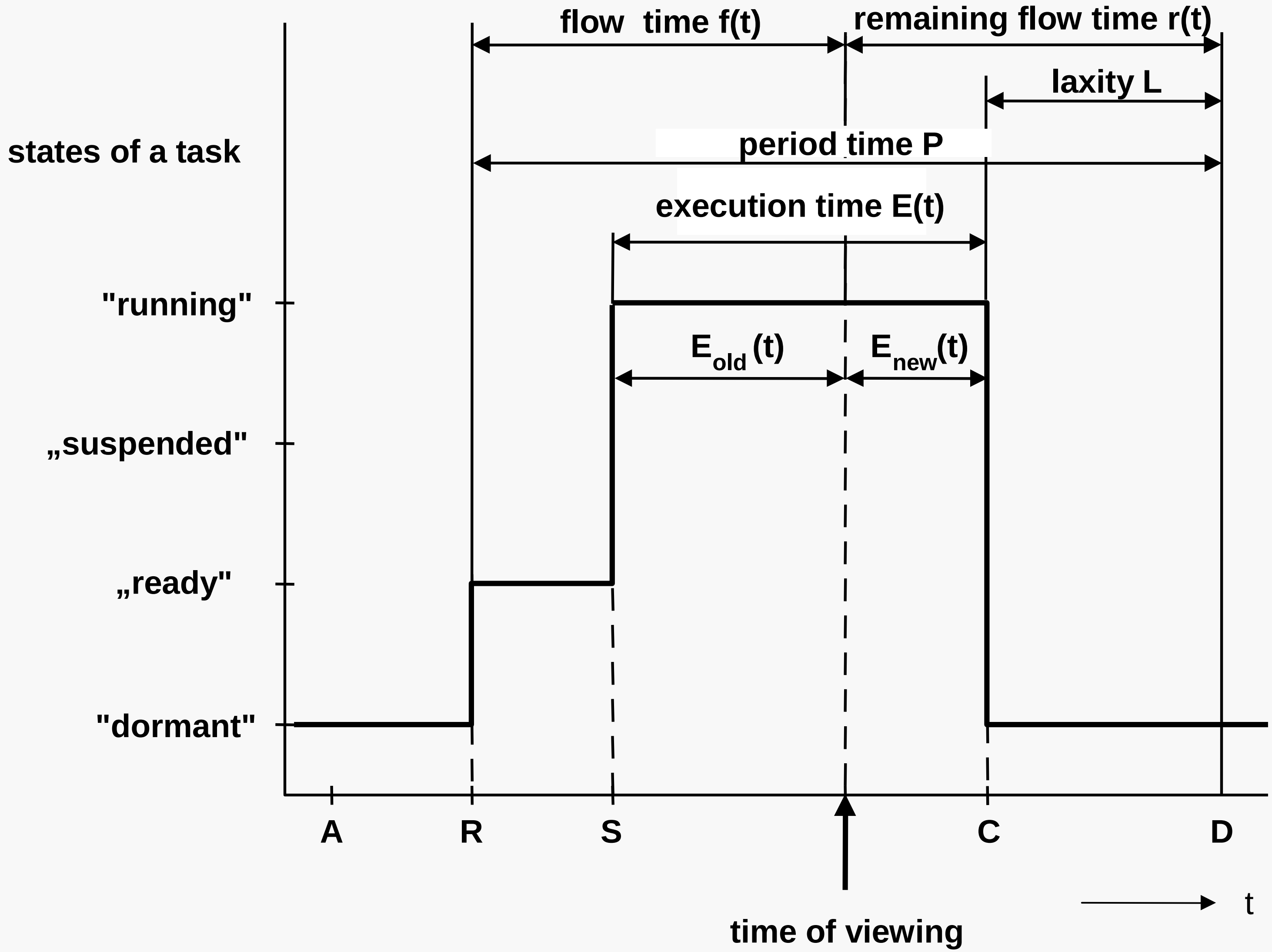
- static priority assignment
- dynamic priority assignment (use of deadlines)

Time parameters of a task :

A: Arrival time
R: Request time
S: Start time
C: Completion time
D: Deadline
E: Execution time
P: Period time
L: Laxity
r: Remaining flow time
f: Flow time

© 2004 IAS, Universität Stuttgart

Appearance of the time parameters of a task



© 2004 IAS, Universität Stuttgart

Corrrelation between the time parameters of a task

A <= R <= S <= C - E <= D - E

Execution time (computing time)

$E(t) = E_{old}(t) + E_{new}(t)$

$E_{old}(t)$:	present execution time at time of viewing
$E_{new}(t)$:	remaining execution time

Laxity

for the execution of a task

L = D - S - E

© 2004 IAS, Universität Stuttgart

§ 4 Real-time programming

© 2004 IAS, Universität Stuttgart

Classification of the actions of a task

- Two actions within a task are called parallel if they can run simultaneously
- Two actions are called sequential if they are arranged in a certain sequential order
- Two actions of two different tasks are called concurrent if they can run simultaneously (outer parallelism)
- Two actions of a task are called simultaneous, if they can be run at the same time

Actions = Threads

© 2004 IAS, Universität Stuttgart

Requirement on the execution of tasks

Synchronicity between tasks and the technical process

Dependencies between tasks

– Logical dependency because of the technical process

e.g.: The desired values for the control have to be defined at least once before they can be used

– Dependencies through the use of shared resources

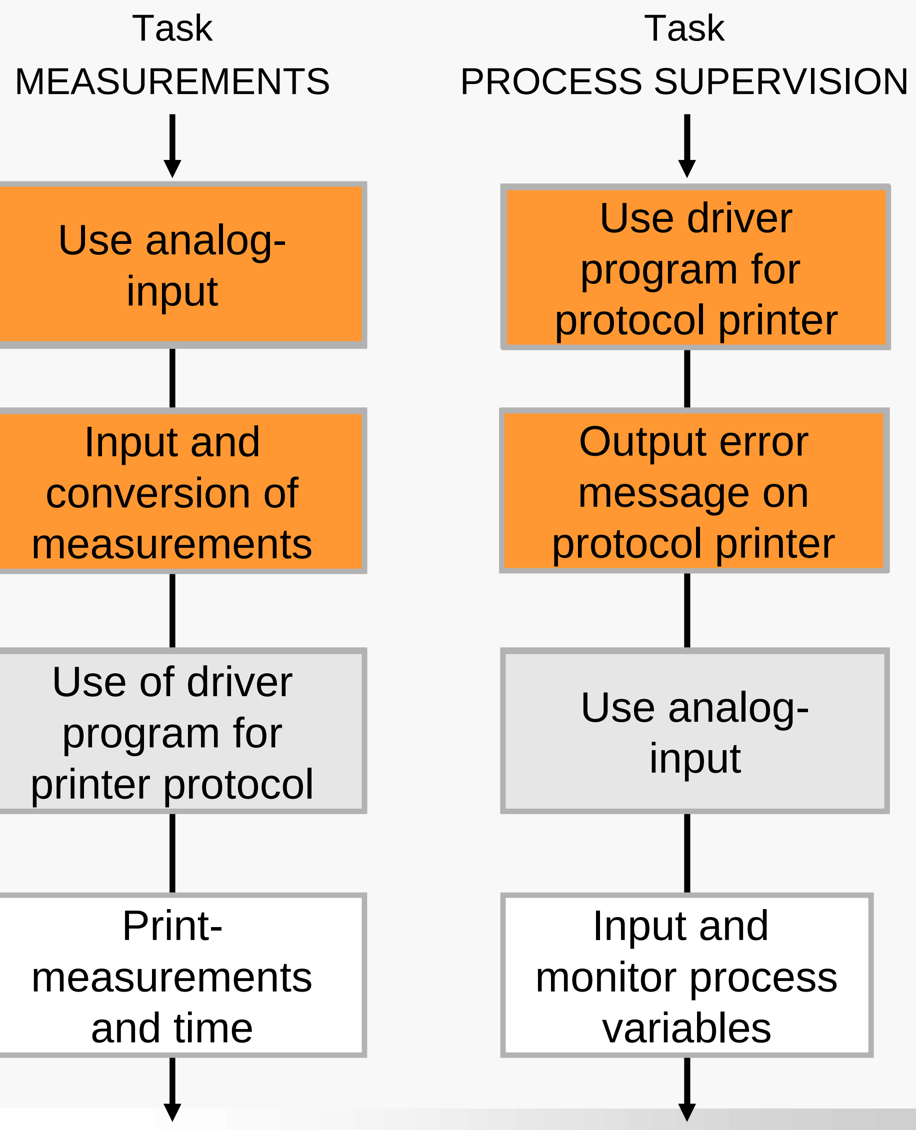
© 2004 IAS, Universität Stuttgart

Example

the dependencies of tasks due to shared resources

Shared resources:
– protocol printer
– analog-input

possibility of deadlock!



Problems of dependency

Deadlock:

Two or more tasks block themselves mutually

Permanent deadlock (livelock, starvation)

A conspiracy of tasks block a task

è Time coordination of the tasks

= synchronization of tasks

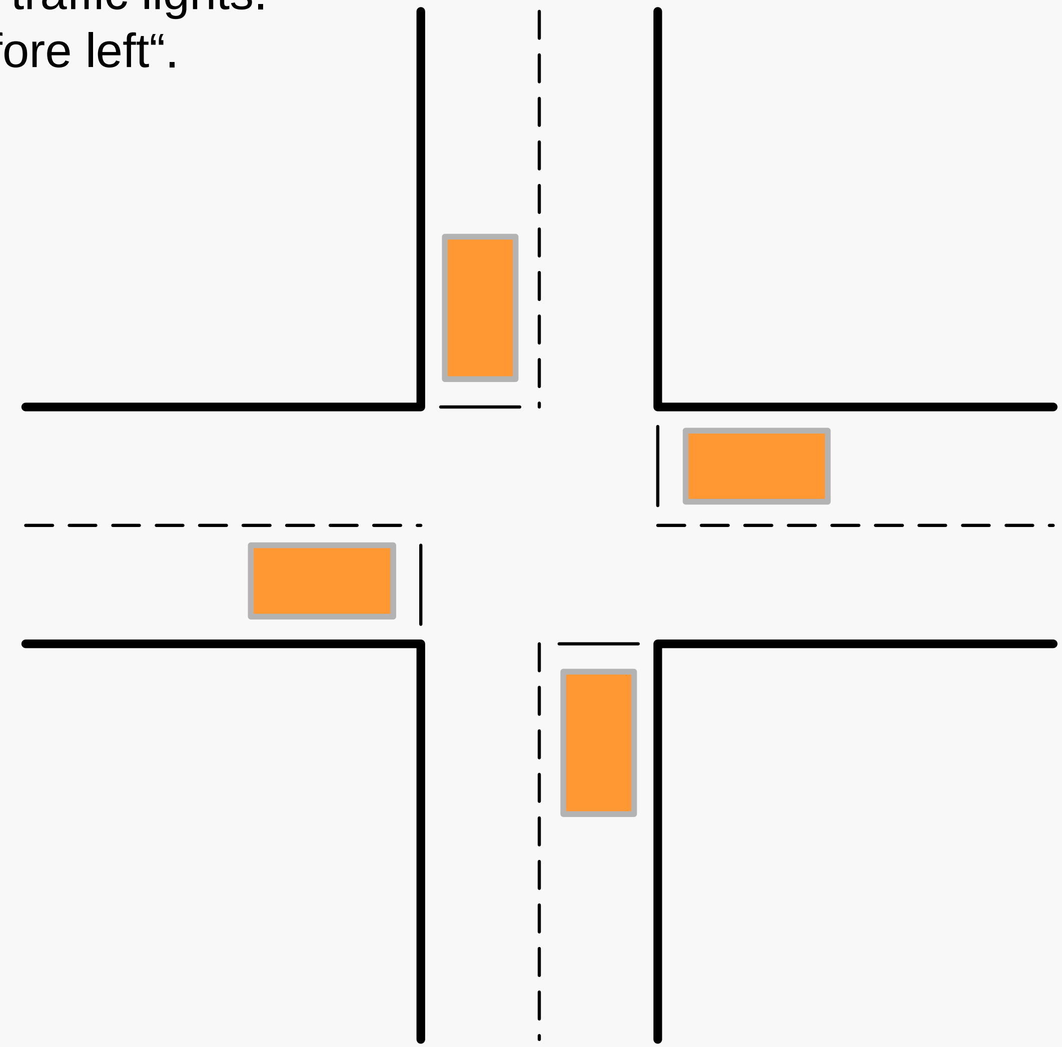
= limitation of the free parallel execution

The synchronization of tasks is equivalent to the synchronization of their actions.

© 2004 IAS, Universität Stuttgart

Example for a deadlock

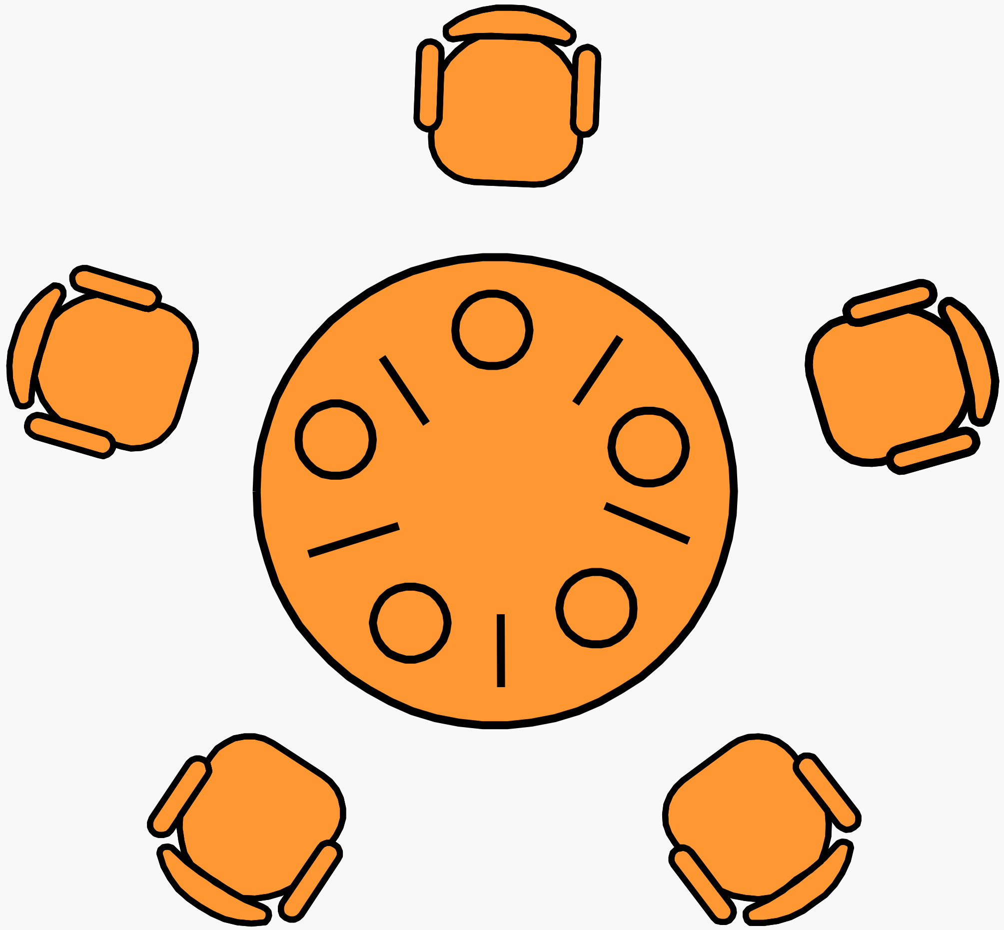
Road crossing without traffic lights.
Traffic rule is “right before left“.



Example for a livelock: “ The Dining Philosophers”

- every philosopher needs 2 chopsticks to eat

- no chopstick can be reserved



© 2004 IAS, Universität Stuttgart

Main forms of synchronization

Logical synchronization

(task oriented or process oriented synchronization)

- Adaptation of the sequence of actions of a task to the sequence of operations in the technical process.
- Synchronization means
  - Fulfillment of requirements regarding the sequence of actions
  - Consideration of given times resp. intervals
  - Reaction to the interrupt message from the technical process

Resource-oriented synchronization

- Fulfillment of requirements regarding the use of shared resources

© 2004 IAS, Universität Stuttgart

Synchronization methods

- Semaphore
- Critical regions
- Rendezvous concept

Basic idea of all synchronization methods

- Task has to wait until a certain signal or event occurs
- Use of waiting conditions at critical places

© 2004 IAS, Universität Stuttgart

Semaphore concept

Synchronization of tasks through signals (Dijkstra)

Semaphore variable: positive, integer value

semaphore operations: V(S) und P(S)

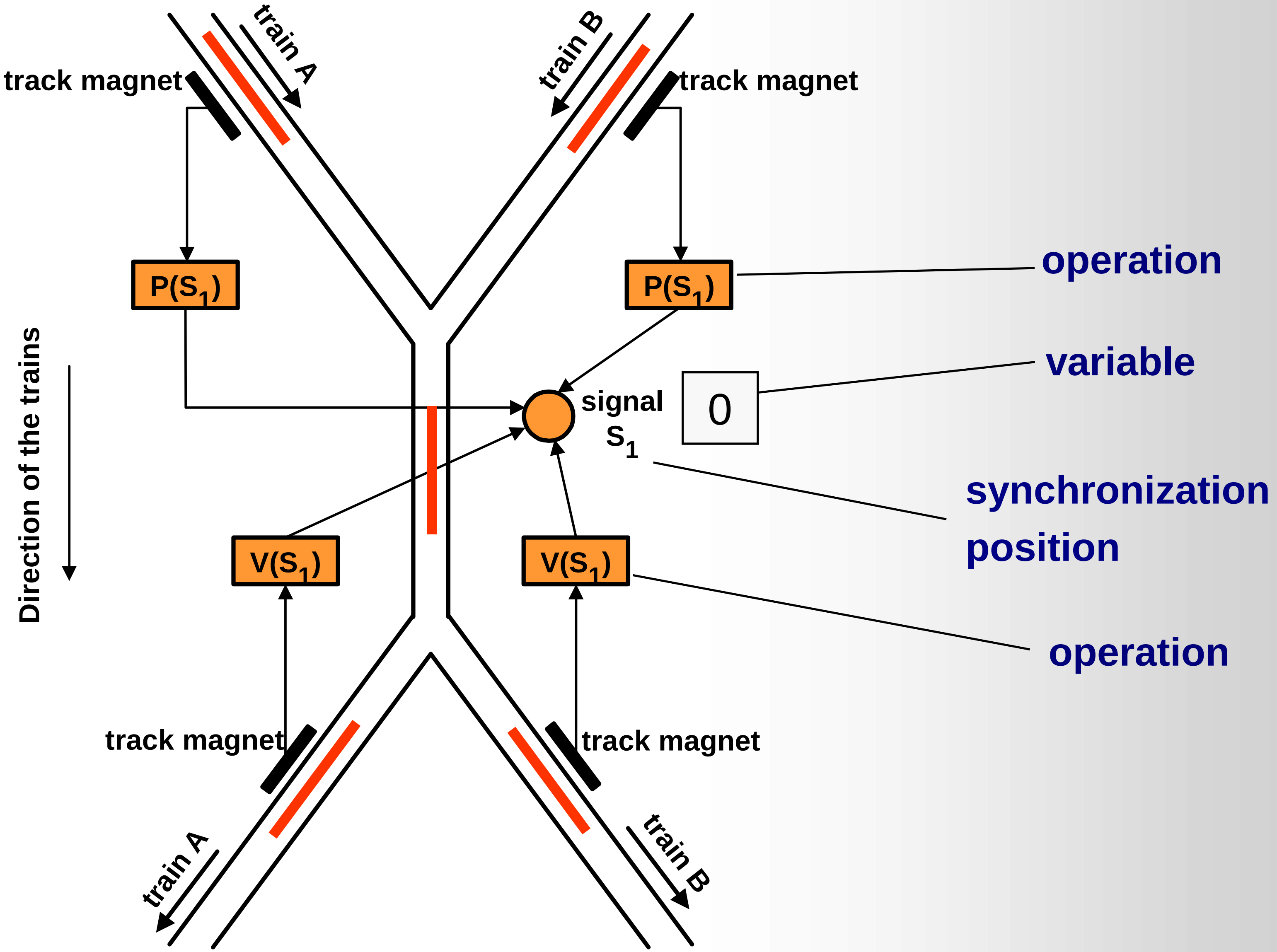
$V(S_i)$: Operation V (S_i) increases the value of the semaphore variable S_i by 1

P(Si): Operation P(S_i) determines the value of S_i

- if value of $S_i > 0$ decrease S_i by 1
- if $S_i = 0$ it has to be waited, until $S_i > 0$

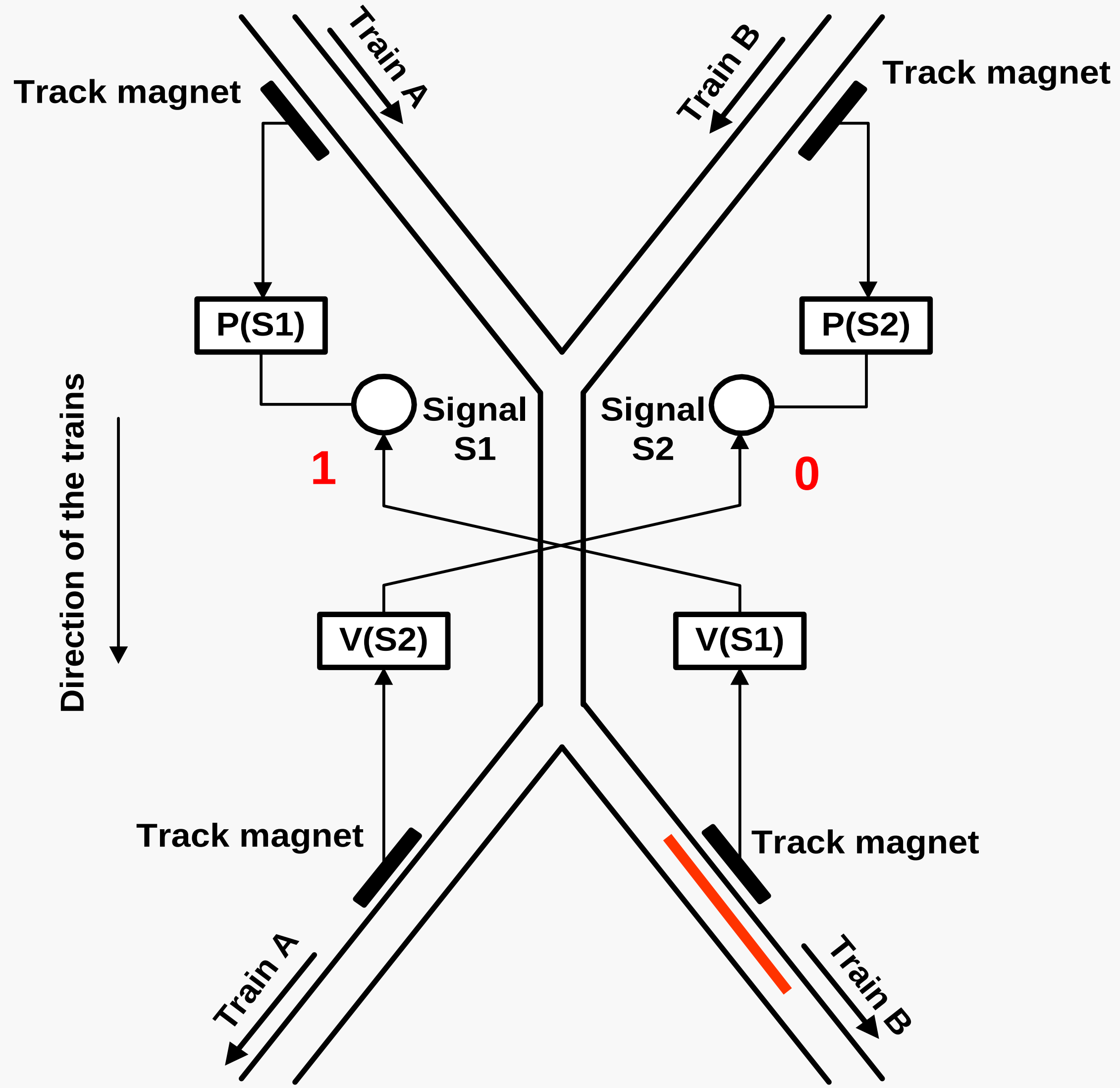
indivisible !

Example 1: Railroad traffic with single-track route



© 2004 IAS, Universität Stuttgart

Example 2: Railroad traffic with single-track route

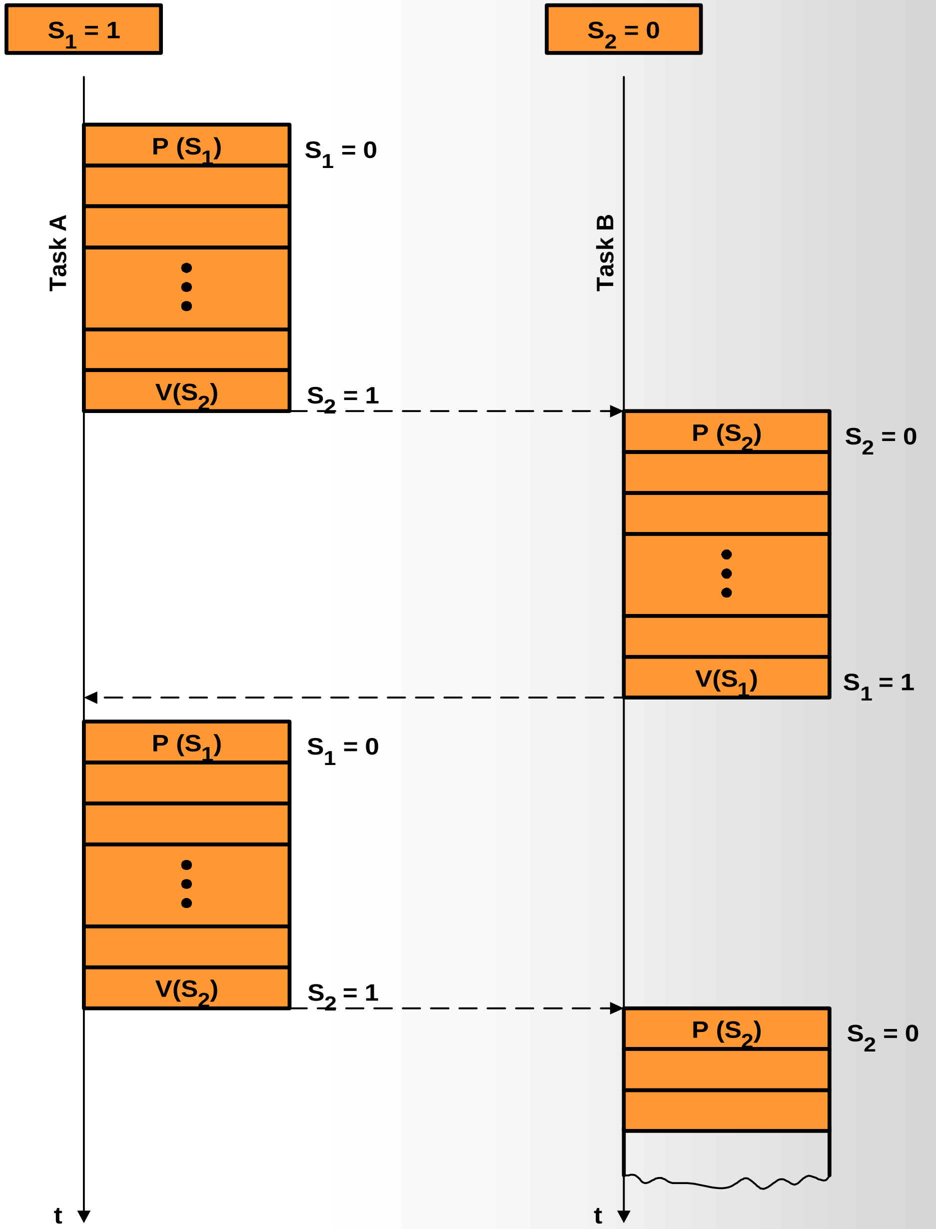


with ordered sequence

Example:

operations on two tasks, that always run alternately

logical synchronization



§ 4 Real-time programming

© 2004 IAS, Universität Stuttgart

Definition

Synchronization =
Fulfillment of time related and logical conditions in the parallel run of tasks

Communication =
Exchange of data between parallel running tasks

Interrelation: Synchronization - Communication

Synchronization :	Communication without information
Communication :	Synchronization for information exchange

Possibilities of data communication

- Shared memory (commonly used memory)
  - common variable
  - common complex data structure
- Sending of messages
  - transmission of messages from a task to another (message passing)
  - especially in distributed systems

Different kinds of communication

- Synchronous communication
  - Sending and receiving processes communicate at a certain predefined position in the program flow

waiting through blocking

- Asynchronous communication
  - Data are buffered

no waiting times

© 2004 IAS, Universität Stuttgart

§ 4 Real-time programming

© 2004 IAS, Universität Stuttgart

Scheduling problem

- Tasks need resources (processor, in/output device etc.)
- Number of resources is limited
- Tasks compete for resources
- The allocation of resources has to be managed

Example: Railroad tracks in station area

Scheduling:

Allocation of the processor to runnable tasks according to a predefined algorithm (Scheduling-method)

Problem:

1. Is there an executable schedule for a set of tasks?
2. Is there an algorithm that can find an executable schedule?

Example: Lecture- room- time- allocation

© 2004 IAS, Universität Stuttgart

Classification dependent on the time of the planning

– Static scheduling **Inflexible in case of modifications**

- Planning of the execution sequence of the tasks is done before the actual execution (dispatching table)
- Consideration of information on taskset, deadlines, execution times, sequential relations, resources
- Dispatcher carries out the allocation according to dispatching table
- Runtime - overhead minimal
- Deterministic behavior

– **Dynamic scheduling** **Flexible in case of modifications**

- Organization of the execution sequence during the execution of the tasks
- Considerable runtime - overhead

Classification according to the kind of execution

- **Preemptive scheduling**
  - running task can be interrupted
  - higher priority tasks replace lower priority tasks

cooperative scheduling

- **Non-preemptive scheduling**
  - running task cannot be interrupted
  - processor-deallocation by the task itself

© 2004 IAS, Universität Stuttgart

Scheduling methods

- FIFO scheduling (first-in-first-out)
- Scheduling with fixed (invariable) priorities
- Round-Robin-Method (Time slice method)
- Earliest-deadline-first method
- Rate monotonic scheduling
- Method of minimal laxity (least laxity)

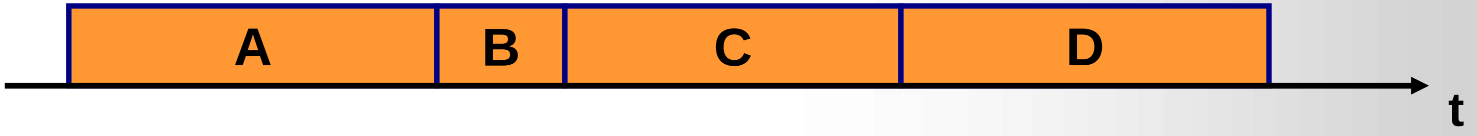
© 2004 IAS, Universität Stuttgart

FIFO scheduling

- Non-preemptive scheduling
- The processor is allocated to the task with the longest delayed planning
- Simple implementation

unsuitable for hard real time systems

The tasks are executed in the order in which they become runnable



© 2004 IAS, Universität Stuttgart

Round-Robin-Method (time slice method)

- Each task has a determined time slot in which the processor is allocated to it
- Sequence is determined statically
- Execution of a task “step by step”
- Used in dialog systems (multi-tasking systems)

unsuitable for hard real-time systems

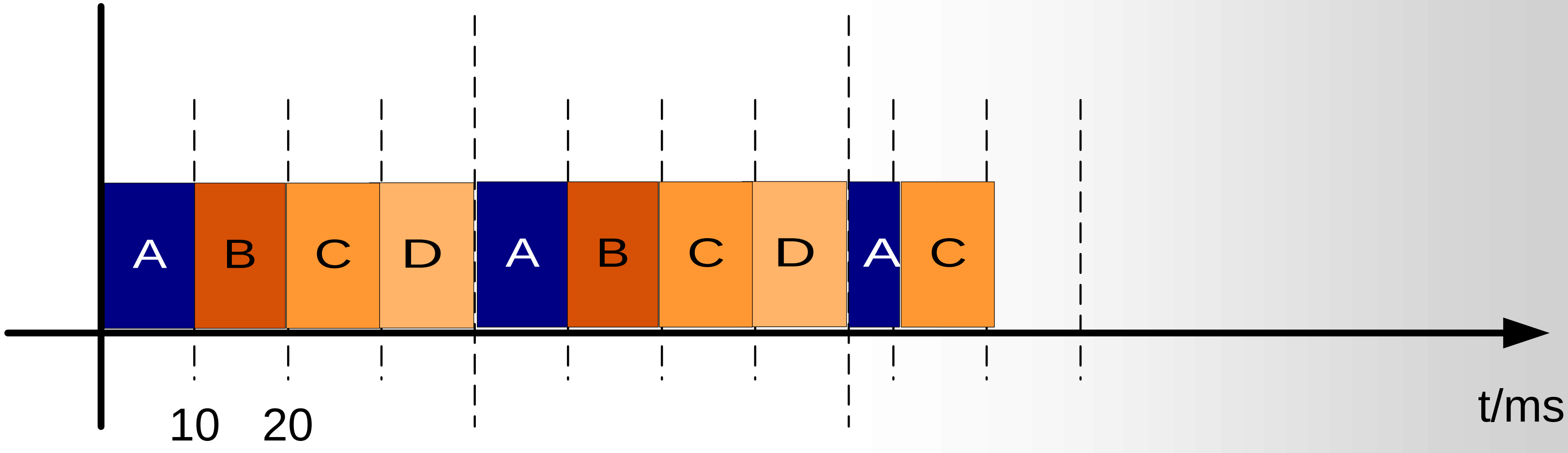
© 2004 IAS, Universität Stuttgart

Example: Round-Robin method

Each time slice has 10ms and the tasks were arranged in the following order: A-B-C-D.

Execution time of the tasks:

Task A:	25ms
Task B:	20ms
Task C:	30ms
Task D:	20ms



Scheduling with fixed priorities

- Priorities are assigned statically
- The processor is allocated to the task with the highest priority
- Preemptive and non-preemptive strategy is possible
- Simple implementation

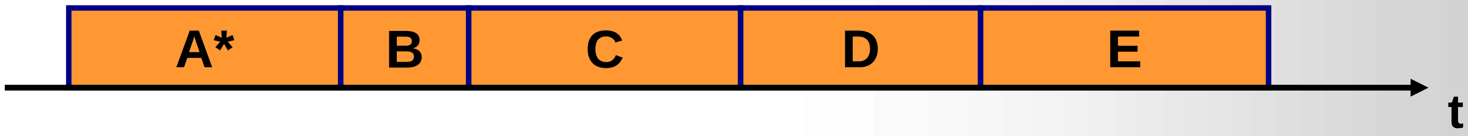
for hard real-time systems only suitable under certain conditions

© 2004 IAS, Universität Stuttgart

Example: fixed priorities

Task	Priority	
A	1	Highest Priority
B	1	
C	2	
D	3	
E	3	

Depending on the strategy one of the two tasks A or B is running as long as it has the highest priority



*** FIFO scheduling: A was the first runnable task**

Rate monotonic scheduling

– Special case of scheduling with fixed priorities of cyclic tasks

– The shorter the period, the higher the priority

– Task with the shortest period has the higher priority

– Preemptive strategy

Method frequently used in real applications

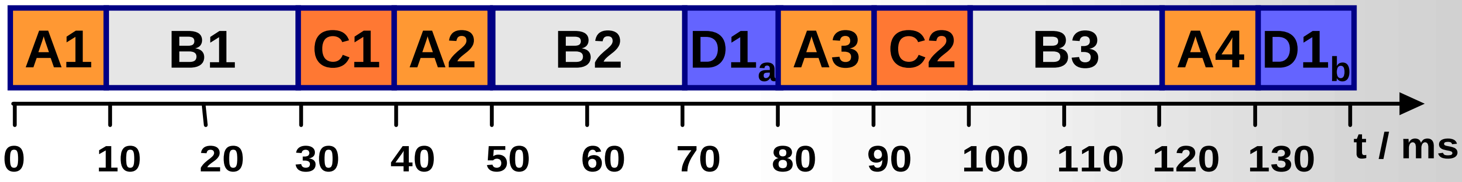
© 2004 IAS, Universität Stuttgart

Example: Rate-Monotonic scheduling

Task	Execution time	Period	Priority
A	10 ms	40 ms	1
B	20 ms	50 ms	2
C	10 ms	80 ms	3
D	20 ms	100 ms	4

First scheduled call of all tasks at t = 0 ms.

Afterwards the tasks have to be repeated cyclically.



Task D was interrupted

Earliest-Deadline-First-method (Minimal-remaining-flow-time-method)

- The processor is allocated to the task with the shortest remaining flow time
- Preemptive method
- High computation effort for the scheduling
- Compliance with requirements on time is specially supported

© 2004 IAS, Universität Stuttgart

Example

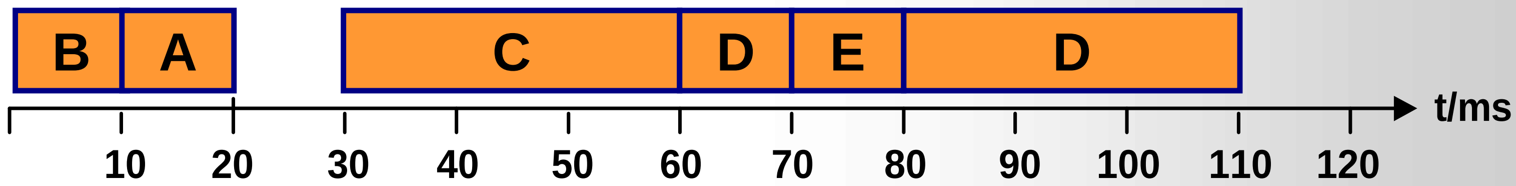
Task	Time	T_{min}	T_{max}
A	10 ms	0 ms	40 ms
B	10 ms	0 ms	30 ms
C	30 ms	30 ms	100 ms
D	40 ms	50 ms	200 ms
E	10 ms	70 ms	90 ms

t_{min}: earliest time

t_{max}: latest time

= deadline

Execution sequence:



Reason: **Deadline of B is earlier as the one of A**

Deadline of E is earlier as the one of D

ð Preemption

Least laxity method

- Processor is allocated to the task with the smallest laxity
- Consideration of deadlines and of the execution time
- Very expensive method

best suitable for hard real-time systems

© 2004 IAS, Universität Stuttgart

Schedulability test

A schedulability test is a mechanism to proof whenever a set of tasks can be scheduled in such a manner that the deadlines are not missed.

Being a mathematical proof, it is important to differentiate between:

- **necessary condition**
- **necessary and sufficient condition**

© 2004 IAS, Universität Stuttgart

Theorem of Liu and Layland (Part I)

Any given set of tasks (Deadline = Period) can be called by the preemptive Earliest-Deadline-First method when:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Where:

C_i : Execution time of task i
T_i : Period of task i
n: Number of tasks

Concerning the EDF method this test is *necessary* and *sufficient*.

For any scheduling method this is only a *necessary* condition.

Theorem of Liu und Layland (Part II)

Any given set of tasks (Deadline = Period) can be called by the

Rate-Monotonic method when:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq U_n^* = n\left(2^{\frac{1}{n}} - 1\right), \quad n = 1, 2, \ldots$$

$$U_1^* = 1{,}0$$

$$U_2^* = 0{,}828$$

$$U_3^* = 0{,}779$$

$$U_\infty^* = \ln 2 = 0{,}693$$

Where:

C_i : Execution time of task i
T_i : Period of task i
U_n^*: Utilisation of processor with n tasks
n: Number of tasks

and:

$$U_i = \frac{C_i}{T_i}$$

Sufficient test

Question referring to Chapter 4.1

Which kind of real-time systems are the following ones?

Answer

	hard real-time system	soft real-time system
paper printing device	X	
car electrical window control	X	
TV electron beam control		X
telephone switching control		X
CNC milling head control	X	
aircraft turbine engine control	X	

If a deadline is missed in a hard real-time system, then this is equal to a failure of the automation system. Often damage to property or to persons can occur in such systems.

© 2004 IAS, Universität Stuttgart

Question referring to Chapter 4.2

Which of the following statements regarding to real-time programming do you agree?

Answer

- f ¨ In real-time programming only the timeline of an event is in center.
- f ¨ Real-time means „as fast as possible“.
- f ¨ In soft real-time systems timelines do not have to be fulfilled.
- ü The asynchronous programming method is more flexible regarding to outer events than a syhchronous programming method.
- ü A synchronous programming requires cyclic programm flow.
- f ¨ The synchronous programming method does not fulfill the requirement for concurrency.

Question referring to Chapter 4.4

a) Two tasks access the same temperature sensor in an automation system.

b) A control algorithm is divided into three tasks: „input of actual value“, „calculation of control variable“ and „output of control variable“. These tasks always have to be executed in this sequence.

Which type of synchronization has to be used respectively ?

How many semaphore variables are necessary in each case ?

Answer

a) This is a resource-oriented synchronization. One semaphore variable is required for each resource. (In this example: one)

b) The second example is a logical synchronization. One semaphore variable is required for each point of synchronization at a transition from one task to the next one. (Three in this example)

© 2004 IAS, Universität Stuttgart

Question referring to Chapter 4.6

In the so called „shortest-job-first"-scheduling method, the task with the shortest execution time is selected for processing at run-time. However, running tasks cannot be interrupted by other tasks.

Which type of scheduling method is this ?

Answer

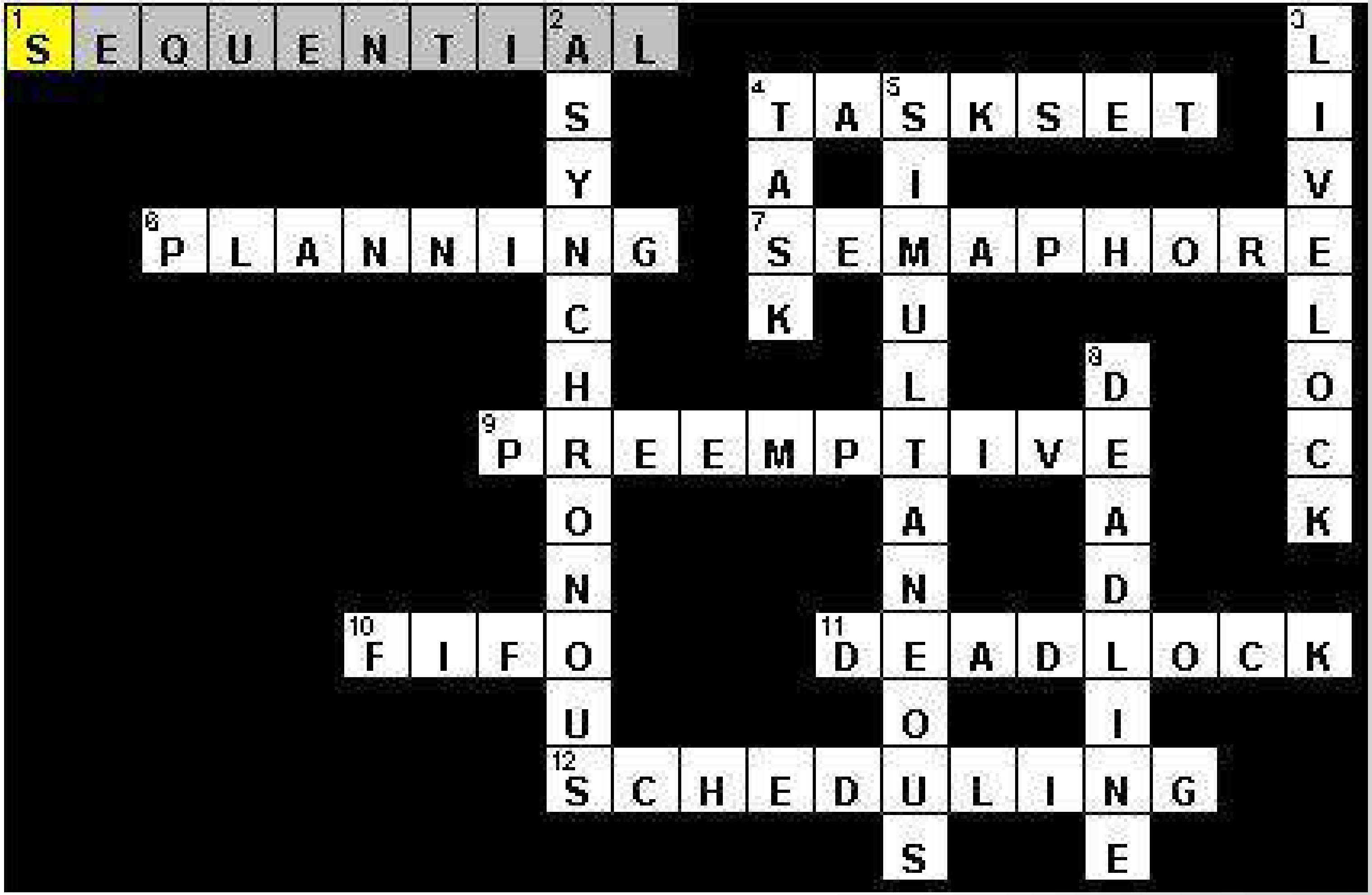
The planning of the task sequence is done only at run-time of the system, during the program execution. Furthermore, the tasks cannot be interrupted.

Consequently, we can classify this method as a dynamic, non-preemptive scheduling method.

Remark:

The "shortest-job-first" method determines a schedule, that minimizes the average flow-time.

Crosswords to Chapter 4



Crosswords to Chapter 4

Across

1	Term for activities that are ordered in a specific order. (10)
4	Set of all tasks controlled by a scheduler. (4,3)
6	Transition from the state "dormant" into the state "runnable" (8)
7	Synchronization construct (9)
9	Interruptible processor allocation. (10)
10	Execution of activities according to their arrival. (4)
11	A stalemate that occurs when two (or more) tasks are each waiting resources held by each other. (8)
12	Allocation of processor resources to tasks ready to execute. (10)

Down

2	Organization of timely execution of subprograms at runtime. (12)
3	Situation in which a task can not realize its duty, even if it continues to run. (8)
4	Procedure of the execution of a sequential program, controlled by a real time operating system. (4)
5	Term for two activities of a task that could be executed at the same time. (12)
8	Latest execution time point of a task. (8)

© 2004 IAS, Universität Stuttgart