Course

(Introduction to) Embedded Systems Summer 1999

Mo: 16-18

Peter Marwedel, Informatik XII
OH16, E21

phone 755 6111
fax. 755 6116
e-mail: marwedel@cs.uni-dortmund.de

Consulting: Mo, 10:30-12:00

1. Introduction

1.1 Scope

1.1.1 Definition of embedded systems

computer science/informatics = science of information processing.

Frequently constrained to using PCs and mainframes.

Def.: Embedded systems (ES): Systems reading, processing and controlling physical data using information processing technologies.

Embedded systems are **reactive systems**:

Def. [Berge]: *A reactive system is one that is in continual interaction with its environment and executes at a pace determined by that environment.*

Reaction depends on input and current state.

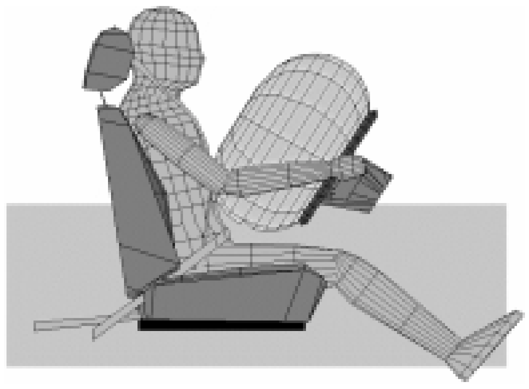
Timing behaviour must be considered in detail.

Def.: An ES which has to meet certain timing constraints is called a **real time system**.

Def.: If not meeting certain time contraints could result in a catastrophe, then the time constraints are called **hard real-time constraints**.

1.1.2 Examples of embedded systems

- **ES in transportation systems**
  - **cars**



Today, cars can only be sold if they contain a significant amount of electronics.

  - **air planes**

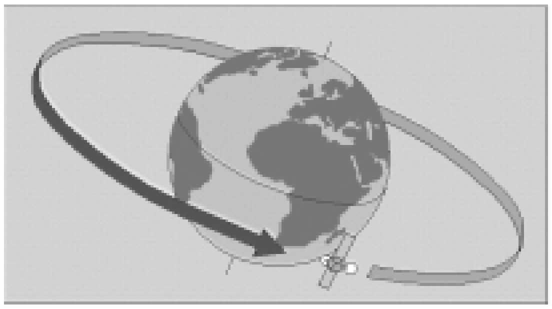
Used for pilot information systems, local distance control system, fly-by-wire.

Very high dependability requirements.

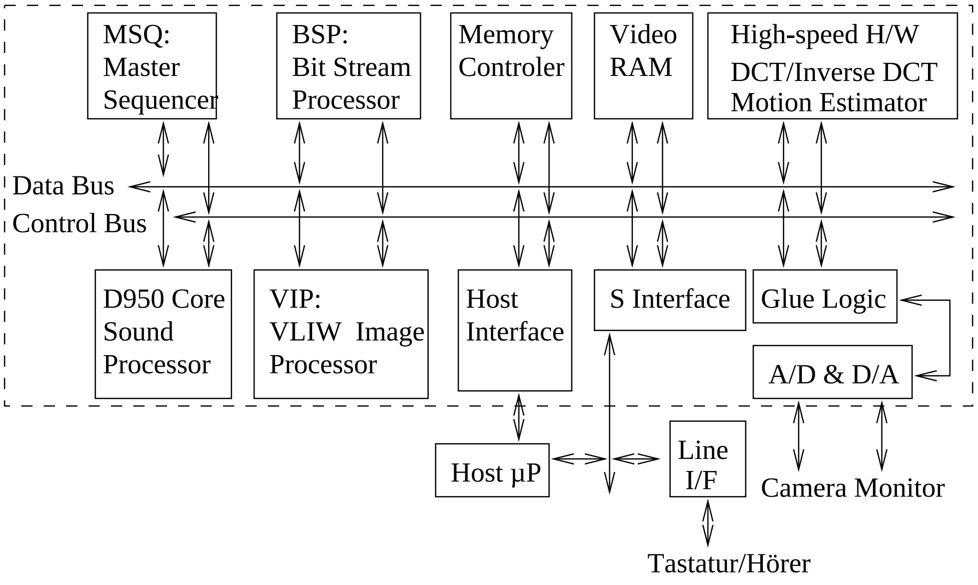
- **Trains**

Train safety systems, traveller information systems, efficient use of energy, driver information systems.

- **ES in telecommunications**

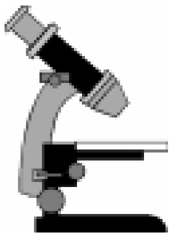


Block diagramm of a one chip video phone:



- **ES in medical instrumentation**

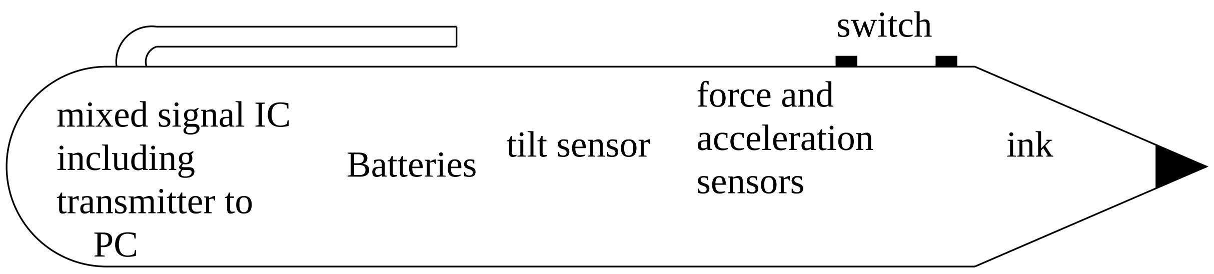
  (medical analysis and control systems)



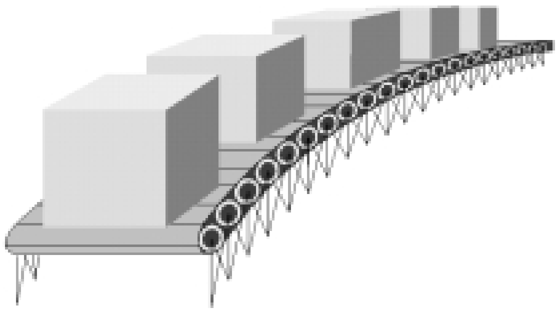
- **ES in military applications**

- **payment systems using ES**

  Smartpen:


switch
mixed signal IC including transmitter to PC
Batteries
tilt sensor
force and acceleration sensors
ink

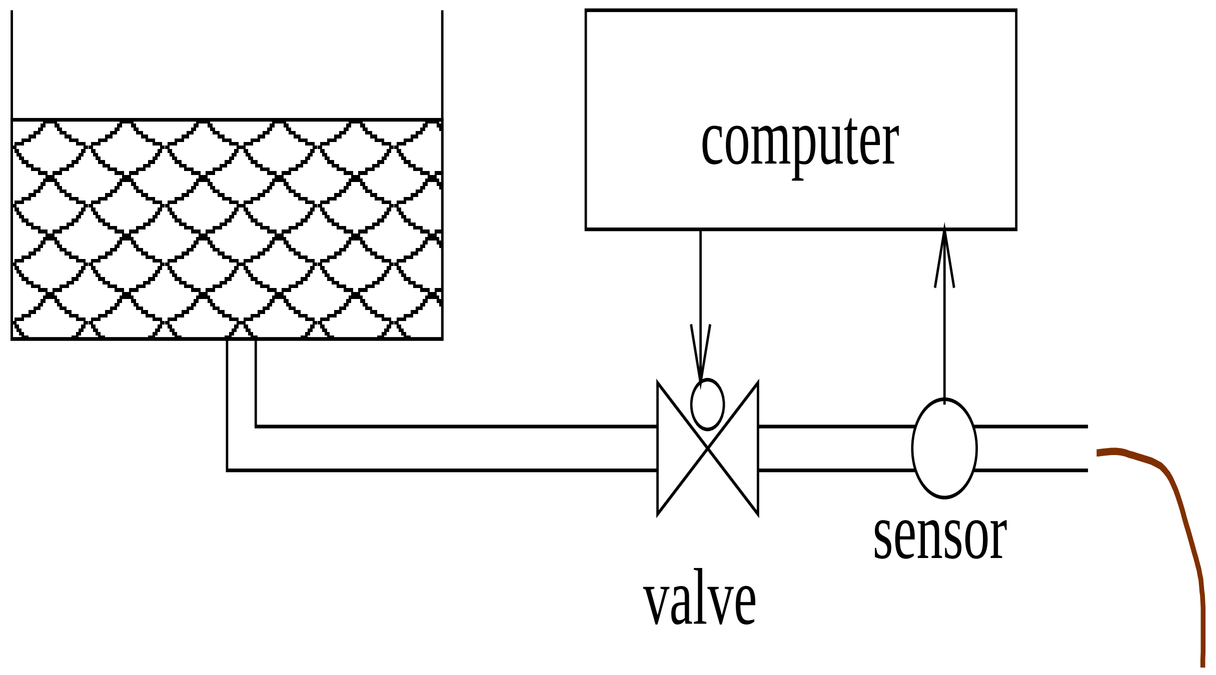
- **ES in fabrication**



Example (from Kopetz):

Given: Container with liquid

Flow to be controled using a valve:



Flow to remain constant. Sensor measuring flow.

Opening and closing of valve takes 10 secs (special case of an **actor**).

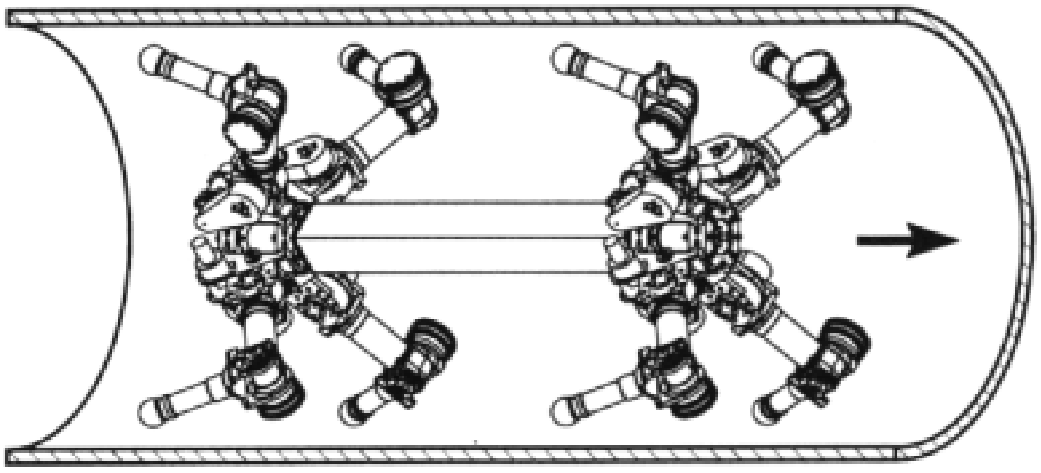
Sensor: limited resolution, e.g. 1%.

Possible set-up: sending sensor values every 100 ms. Delay between changing the valve and sensed changes is important.

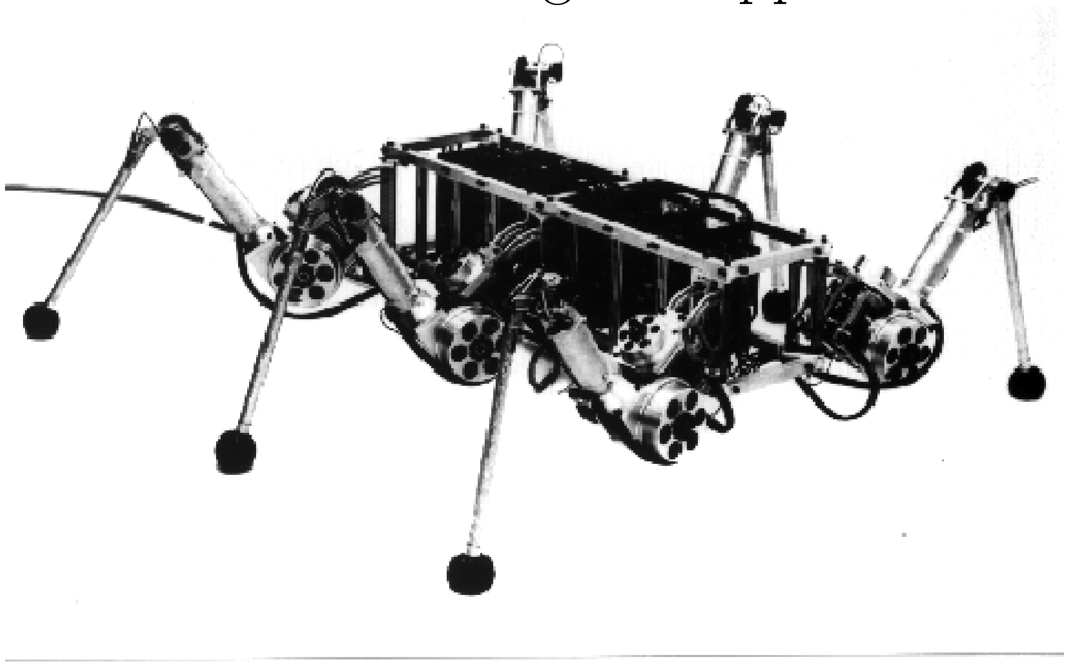
Situation much more complex if many sensors and actors are present.

Examples of industrial robots:

Pipe climber:



Mechanical model of a grasshopper:



- **ES in buildings**

Example: telecom company Helsinki:

- sensors indicate people entering room,

- light and air conditioning will be switched on,

- connected to light sensors and controls,

- unused rooms will not be heated or cooled as much as used rooms,

- CO_2-sensor affects air conditioning system.

- Noise created by ventilation reduced to minimum.

- Display informs about status of each room.

- Energy consumption of each room is recorded.

- Communication using LAN network.

$\rightarrow$ **huge variety of embedded systems.**

1.1.2 Size of the embedded systems market

79% of all high end microprocessors are used in embedded systems.

ES very important market for Europe.

Importance typically underestimated.

Mary Ryan:

Embedded chips aren't hyped in TV and magazine ads ... but embedded chips form the backbone of the electronics driven world in which we live. ... they are part of almost everything that runs on electricity.

Volume: 31 Billion US $.

(*general purpose computing*: 46,5 Billion US $)

Annual growth rate for ES 18% ,

(general purpose computing: 10%)

Trends

- increasing flexibility (software)
- increasing size of the programs
- increasing development complexity.

1.1.3 Characteristics of ES

- Using **sensors** for reading values
  actors for controlling environment
  man/machine interface (MMI).
- Frequently come with hard timing constraints
- ES are reactive systems; automata model useful:
  (input $\times$ state $\rightarrow$ output, new state)
- Behaviour known at design time.
- Very high requirements for:
  - **reliability**
    Reliability $R(t)$ is the probability of a system working correctly at time t provided that it was working at time $t = 0$.
  - **maintainability**
    Maintainability is the probability $M(d)$, of a system working d time units after an error occured.
  - **availability**
    Availability is the probability of having a working system at time t.
  - **safety**

  - **security**
  - very complex, difficult to teach.
  - ES not well-represented in discussions.

    Embedded chips aren't hyped in TV and magazine ads ... but embedded chips form the backbone of the electronics driven world in which we live. ... they are part of almost everything that runs on electricity [Ryan95].

- Frequently, ES come without keyboard, screen and mouse.

"ES = information processing without screen and keyboard."

But: there are differences between different types of ES.

Not every ES has all of the above characteristics.

If major number of characteristics is present → ES.

Common properties of embedded systems → it makes sense to talk about ES in general and not to discuss each application area independently.

Things to remember from an introduction to ES (according to Kopetz Kopetz97):

- *A real-time computer system must react to stimuli from the controlled object (or the operator) within the time interval* **dictated** *by its environment. If a catastrophe could result in case a firm deadline is missed, the deadline is called* **hard***.*
- The probability for a perfectly designed system to fail is equal to the probability that assumptions about the work load and possible erros turn out to be wrong.
- A guaranteed system response has to be explained without using statistical arguments.
- An embedded real-time system is part of a well-specified larger system, an intelligent product.
- Knowledge about the behaviour at design time cane be used to minimize required resources and maximize robustness.
- The embedded system market is expected to grow significantly.

1.2 Motivation and structure of this course

Why this course?

- ES very important
- ES not discussed in other courses.
- ES important for Technical University.
- Broad scope.

Courses on ES very common at computer science departments.

Course includes:

- Specification of embedded systems
- Target architectures
- Hardware/software codesign
- Compilers for embedded systems
- Real-time operating systems
- Designing embedded systems
- Validation of embedded systems

1.3 Relation to other courses

Basic knowledge of computer architecture required.

Overlaps with courses 'Rechnergestützter Entwurf' (ECAD) and 'Prozessrechnertechnik'.

Could be complemented by 'software technology'.

Provides knowledge for working on robot control.

For regular students of this University: this course is a 'Spezialvorlesung' for all computer science students.

1.4 Literature

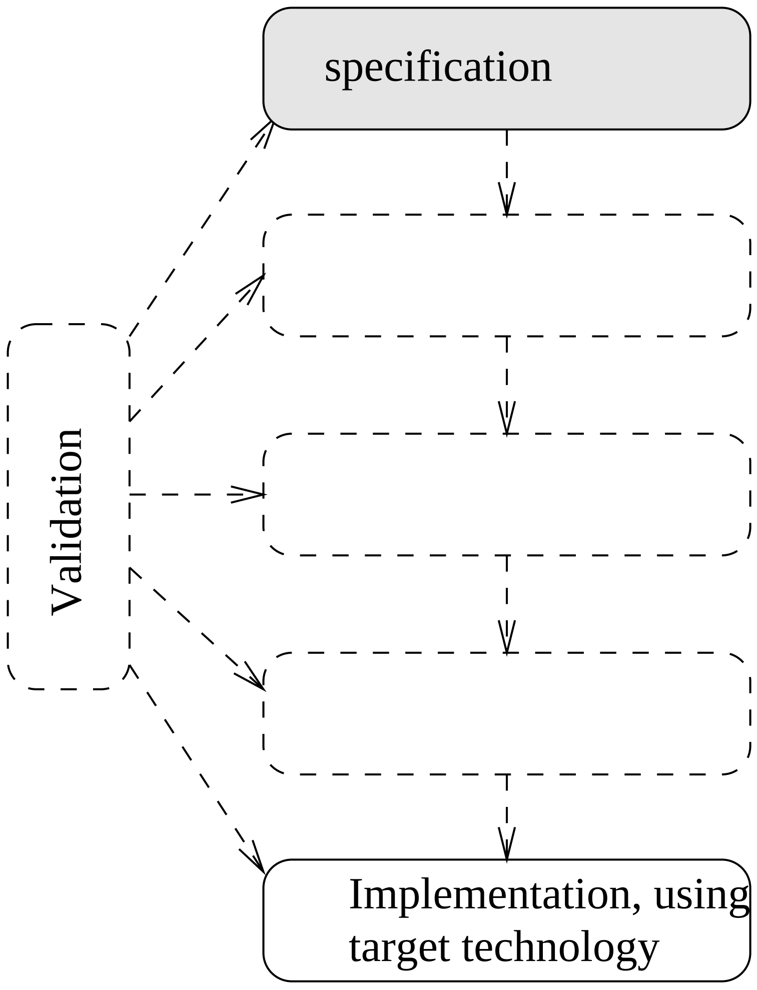
- C.M. Krishnan, K. G. Shin: Real-Time Systems, McGraw-Hill, Computer Science Series, 1997
- H. Kopetz97: Real-Time Systems –Design Principles for Distributed Embedded Applications–, Kluwer, 1997
- A. Burns, A. Wellings: Real-Time Systems and Their Programming Languages, Addison-Wesley, 1990
- Slides of R. Gupta (www.ics.uci.edu)

Slides: http://ls12-www.cs.uni-dortmund.de

2 Specification of embedded systems

2.1 Requirements

Context:



Specification using natural language?

Completeness, free of contradictions, how to implement specification?

Requirements for specifying reactive systems (according to Bergé, 1995):

1. **State-oriented behaviour**:

   The output generated by reactive systems depends on the input and the **current state**.

   The same applies to the next state.

   Automata (finite state machine) are a good model of reactive systems.

   However: classical automata not sufficient.

2. **Specification of timing behaviour**

   Timing constraints can exist for some state transitions.

3. **Concurrent behaviour** + means for specifying synchronisation and communication:

   Distributed systems, processing different tasks, have to be described with using the cross product of their parts.

   Classical FSM not sufficient.

4. **Exception oriented behaviour:**

   Systems have to react to exceptions (errors, crashes etc) regardless of their current state.

   Specifications are readable only if exceptions don't have to be specified at each and every state.

5. **Environment dependent behaviour:**

   Simple means of specifying global effects, for example of current temperature.

6. **Non functional properties**

   Example: fault tolerance, reliability, power consumption, weight, size, useful temperature range, user friendlyness, extendability, expected live time, recyclability.

Additional requirements according to Gajski:

7. **hierarchy (structural + behavioural)**:

   At any point in time, humans can comprehend only systems with a small number of objects, actual systems are much more complex $\rightarrow$ hierarchy is required to comprehend systems

   behavioural hierarchy, e.g. procedures, class libraries

   structural hierarchy, e.g. hardware components

8. **programming langauage elements:**

   For example, arithmetic operations, loops, and function calls should be available.

9. **termination**

   It should be clear, at which time all computations have been completed.

Additional requirements:

10. **object orientation**
11. **executable specifications**
12. **machine independence**

None of the available languages meets all requirements.