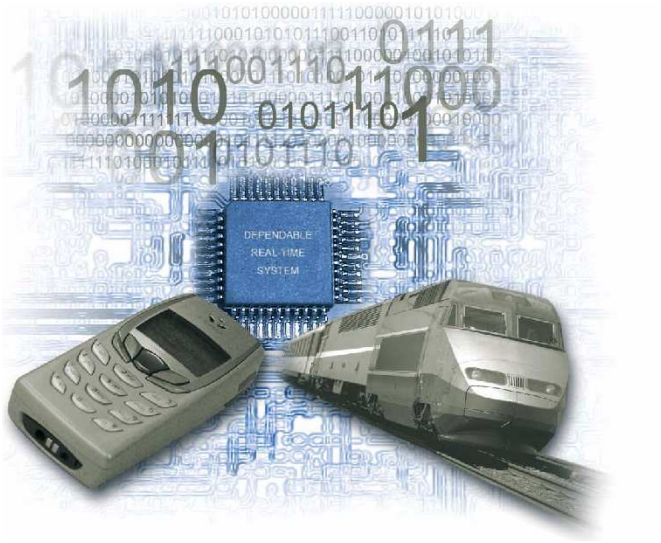
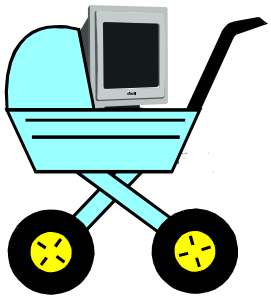
Embedded Systems



Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

What is an embedded system?



These are not the embedded systems we will talk about!

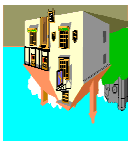
Embedded Systems

Embedded systems (ES) = **information processing systems embedded into a larger product**

Examples:







Main reason for buying is **not** information processing

Characteristics of Embedded Systems (1)

- Must be **dependable**,
  - **Reliability *R(t)* =** probability of system working correctly provided that is was working at *t*=0
  - **Maintainability *M(d)* =** probability of system working correctly *d* time units after error occurred.
  - **Availability**: probability of system working at time *t*
  - **Safety**: no harm to be caused
  - **Security**: confidential and authentic communication
  - Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong.
  - Making the system dependable must not be an after-thought, it must be considered from the very beginning

Characteristics of Embedded Systems (2)

- Must be **efficient**
  - Energy efficient
  - Code-size efficient (especially for systems on a chip)
  - Run-time efficient
  - Weight efficient
  - Cost efficient
- **Dedicated** towards a certain **application**
  Knowledge about behavior at design time can be used to minimize resources and to maximize robustness
- **Dedicated user interface**
  (no mouse, keyboard and screen)

Characteristics of Embedded Systems (3)

- Many ES must meet **real-time constraints**
  - A real-time system must react to stimuli from the controlled object (or the operator) within the time interval **dictated** by the environment.
  - For real-time systems, right answers arriving too late are wrong.
  - **„A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe“** [Kopetz, 1997].
  - All other time-constraints are called **soft**.
  - A guaranteed system response has to be explained without statistical arguments

Characteristics of Embedded Systems (4)

- Frequently **connected to physical environment** through sensors and actuators,
- **Hybrid systems** (analog + digital parts).
- Typically, ES are **reactive systems**:
  „**A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment“** [Bergé, 1995]
  Behavior depends on input **and current state**.
  ☞ automata model appropriate,
  model of computable functions inappropriate.

Characteristics of Embedded Systems (5)

- ES are **underrepresented in teaching** and public discussions:
  „*Embedded chips aren‘t hyped in TV and magazine ads ...*
  [Mary Ryan, EEDesign, 1995]

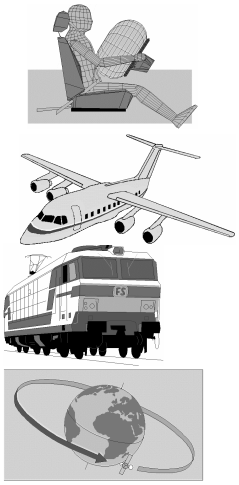
Not every ES has all of the above characteristics.

Def.:Information processing systems having most of the above characteristics are called embedded systems.

Course on embedded systems makes sense because of the number of common characteristics.

Application areas (1)

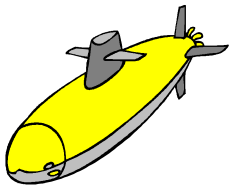
- Automotive electronics
- Aircraft electronics
- Trains
- Telecommunication



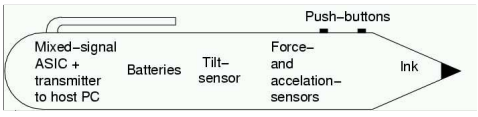
Application areas (2)

- Medical systems

- Military applications



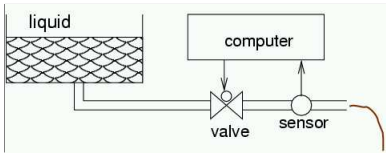
- Authentication



Application areas (3)

- Consumer electronics

- Fabrication equipment



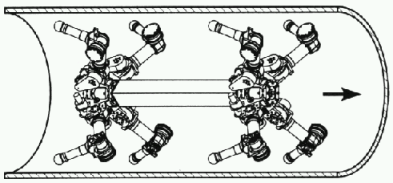
- Smart buildings

Application areas (4)

- Robotics

Robot „Johnnie“ (Courtesy and ©: H.Ulbrich, F. Pfeiffer, TU München)

„Pipe-climber“



Motivation for this course (1)

- Growing economical importance of embedded systems
  - .. *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* [Mary Ryan, EEDesign, 1995]
  - 79% of all high-end processors are used in embedded systems
  - **THE** growing area, according to all forecasts,
  - Majority of open positions for CS graduates now in the automotive business.

Motivation for this course (2)

- Foundation for the „post PC area“
- ES hardly discussed in other CS courses
- ES important for Technical University
- ES important for Europe
- Broad scope: sets context for more specialized courses

Importance of Embedded Software and Embedded Processors

"... the New York Times has estimated that the average American comes into contact with about 60 micro-processors every day...." [Camposano, 1996]

Latest top-level BMWs contain over 100 micro-processors [Personal communication]

Most of the functionality of embedded systems will be implemented in software!

Views on embedded software (1)

.. the next ten years may be as revolutionary for embedded system design as the last 10 years have been for IC design [Wayne Wolf, 1996]

... it is now common knowledge that more than 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development [A. Sangiovanni-Vincentelli, 1999]

Views on embedded software (2)

... dramatic growth of the number of embedded applications and the size and the complexity of the software used in these applications.

For many products in the area of consumer electronics the amount of code is **doubling every two years.**

[Fritz Vaandrager in: Rozenberg, Vaandrager (eds.): Lectures on Embedded Systems, LNCS, Vol. 1494, 1998]

Views on embedded software (3)

It is estimated that each year embedded software is written five times as much as 'regular' software

The vast majority of CPU-chips produced world-wide today are used in the embedded market ... ; only a small portion of CPU's is applied in PC's

... the number of software-constructors of Embedded Systems will rise from 2 million in 1994 to 10 million in 2010;
... the number of constructors employed by software-producers 'merely' rises from 0.6 million to 1.1 million.

[Department of Trade and Industry/ IDC Benelux BV: Embedded software research in the Netherlands. Analysis and results, 1997 (according to: www.scintilla.utwente.nl/shintabi/engels/thema_text.html)]

What's the problem?

If embedded systems will be implemented mostly in software, then why don't we just use what software engineers have come up with?

Some open problems

- How can we capture the required behaviour of complex systems ?
- How do we validate specifications?
- How do we translate specifications efficiently into implementation?
- Do software engineers ever consider electrical power?
- How can we check that we meet real-time constraints?
- Which programming language provides real-time features ?
- How do we validate embedded real-time software? (large volumes of data, testing may be safety-critical)