NTNU
Norwegian University of
Science and Technology

Lecture 1: Introduction to Embedded Systems

Asbjørn Djupdal
ARM Norway and IDI, NTNU
2013

Lecture overview

- Introduction
- Characteristics of embedded computers
- Embedded system design
- Design process examples

Lecture overview

- Introduction
- Characteristics of embedded computers
- Embedded system design
- Design process examples

Embedded systems

Any device that includes a programmable computer but is not itself a general-purpose computer

Examples

- Washing machine
- Quartz wristwatch
- PC keyboard
- Printer
- TV, radio
- GPS
- Digital camera

Embedded systems are everywhere!

Example: Automotive

- Over 100 embedded processors
  - Check seat belt
  - Dashboard devices
  - Engine control
  - Brake system
  - ...
- Many different systems with various processing and communication needs

Lecture overview

- Introduction
- Characteristics of embedded computers
- Embedded system design
- Design process examples

Characteristics of embedded systems

- Take advantage of application characteristics to optimize the design
- Very varied
  - From simple to sophisticated functionality
  - May have real-time requirements
  - Probably requires low manufacturing cost
  - Often has low power as requirement
  - Might have environment requirements
- Designed with tight deadlines by small teams
  - Acceptable is good enough

Functional complexity

- Functional complexity can be high for some devices
- Sophisticated algorithms
  - Laser printer
  - Cell phone
- Sophisticated user interfaces
  - Multitouch GPS

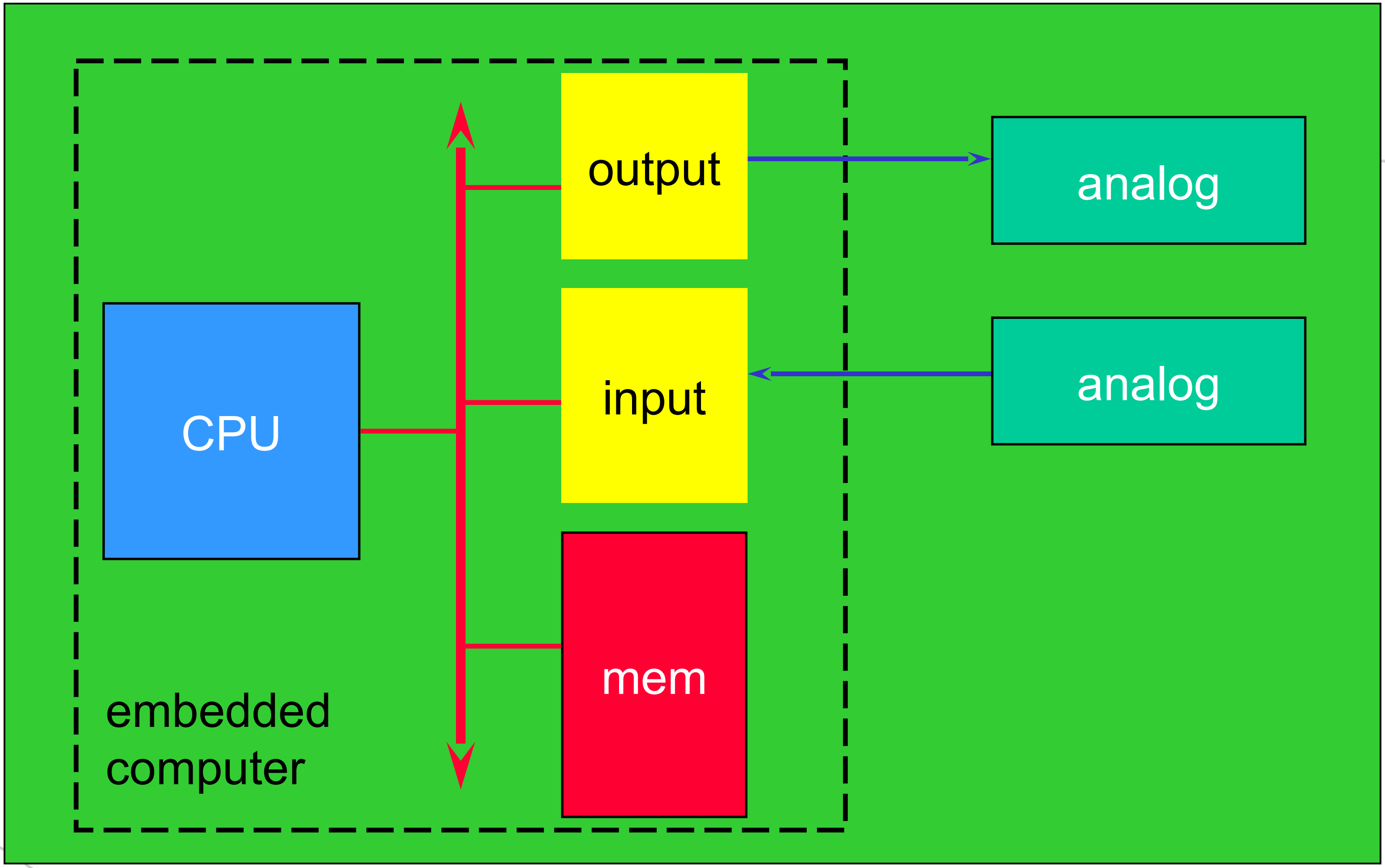
Real-time requirements

- Must finish operations by deadlines
  - Hard real-time: Missing deadlines causes failures
  - Soft real-time: Missing deadlines degrades performance
- Some systems are multi-rate
  - Different operations with different rates
    - Video playback: Audio has different requirements from video

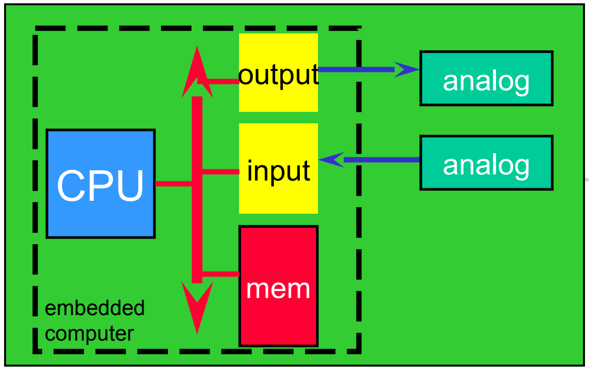
Non-functional requirements

- Mass-marked items
  - Low manufacturing cost is essential
  - Limits memory, processing power, ...
  - Must achieve requirements at minimum cost
- Reduce power consumption
  - Essential for battery powered application
  - Reduces operating cost even in wall powered devices
  - Reduces temperature -> increases life time
- Work in extreme environment
  - Satellites
  - Deep sea

Embedding computers



Processing units



- Processing units
  - CPU – general purpose computing
    - 8 bit, 16 bit, 32 bit
  - GPU – graphics ++
  - DSP – signal processing
- Typically integrated into a single chip, together with I/O-devices, memory, ...
  - Microcontroller (small, cheap)
  - System-On-Chip (powerful, expensive)

Custom logic vs. microprocessors

- Custom logic: FPGA, ASIC
  - Custom made logic, either in ASIC or FPGA
  - Hard to design (expensive)
  - Potential for creating more optimal solution
- Microprocessors
  - General, “easy” to write software
  - Simplifies the design
  - Simplifies the design of product families
    - Can use SW to differentiate product

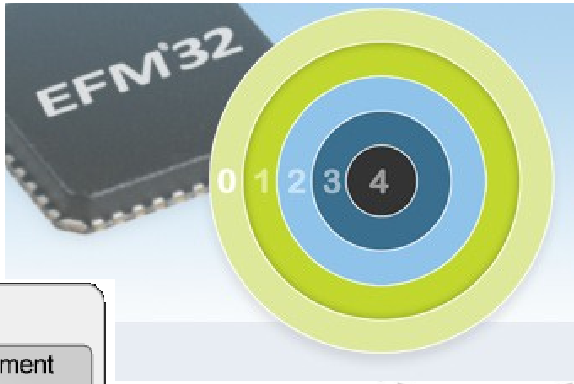
The performance paradox

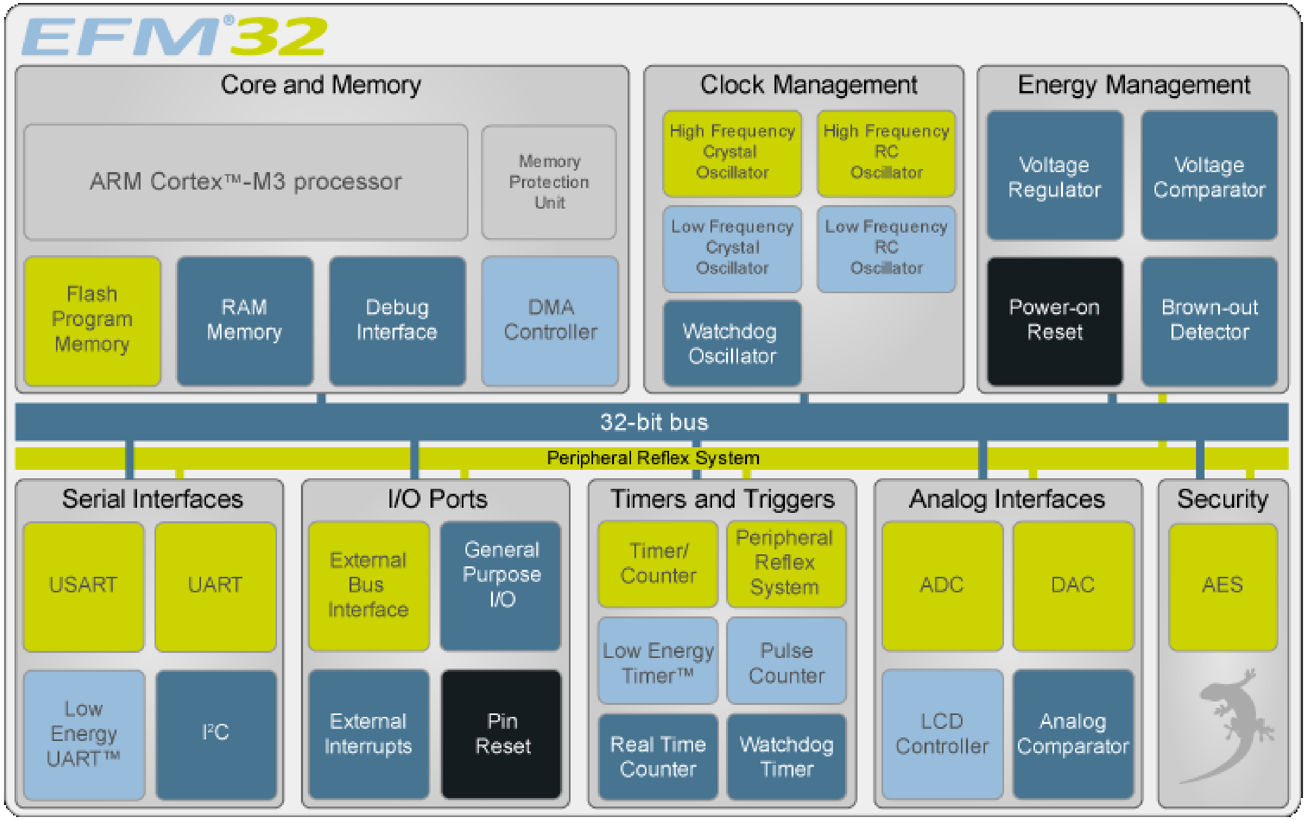
- Would think custom logic always wins:
  - Microprocessors use much more logic (silicon area) than custom logic (for small systems)
  - Microprocessors are (mostly) sequential, while custom logic often exploits parallelism much more
- Not true, microprocessor can often be as fast or faster:
  - Large design teams
    - Complex, deep pipelines
  - Aggressive VLSI technology
    - Too expensive for all but the highest volumes

Power and energy

- Custom logic might beat processors for low power devices
- But CPUs have SW and HW techniques to reduce power consumption
- Only extreme power requirements results in implementing custom logic

Energy micro EFM32





Lecture overview

- Introduction
- Characteristics of embedded computers
- Embedded system design
- Design process examples

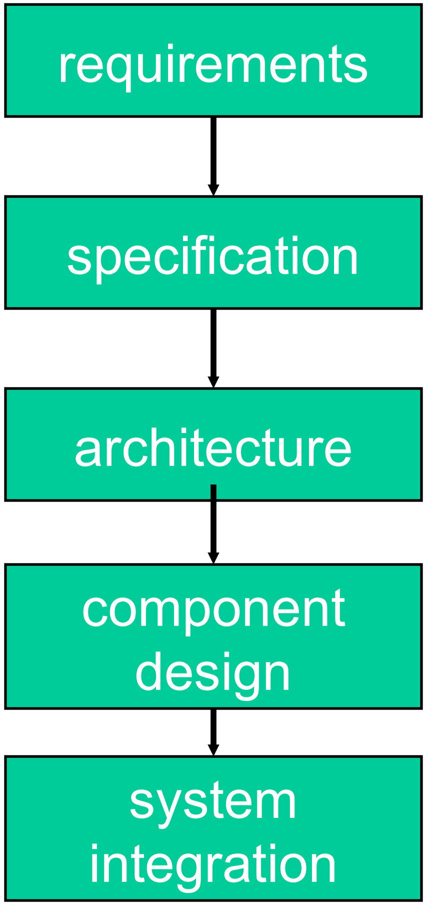
Challenges in embedded system design

- How to meet performance requirements
  - Improve HW: CPU, memory, busses, clock, ...
  - Improve SW: More clever algorithms, optimizations
- How to minimize power?
  - Turn off unnecessary logic? Change SW algorithms?
- How to meet deadlines?
  - Increase component cost or development cost?

HW/SW codesign

- You are designing a complete system
- Where to put functionality, HW or SW?
- The physics of software
  - Computers are physical machines
  - Embedded SW designers need to understand this
    - Energy consumption
    - Performance

Design process



Top-down or bottom-up

- Top down:
  - Start from high level, abstract description
  - Work towards more details
- Bottom-up:
  - Work from small components to big system
- A real design team uses both

Architecture design

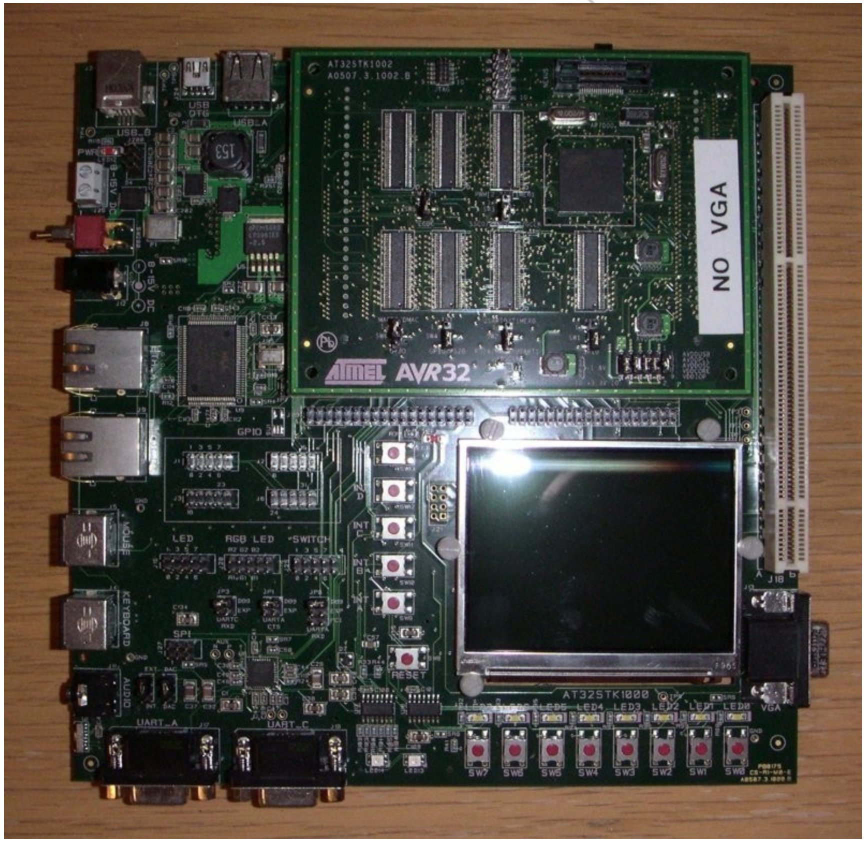
- How to choose which components are needed, how to divide tasks between HW and SW?
- Ideal
  - Draw toplevel, functional diagram
  - Use spec to decide which HW modules are needed for the various functions
    - Good enough performance
    - Low enough power consumption
    - Cheap
  - Design communication busses to support spec

How it is typically done

- Find development board which can do what you need
- HW design group uses this as reference, removes unneeded components and creates HW design
- SW design group uses the board for SW development while waiting for HW prototype

Development boards

- Very common and useful tool for developing embedded systems



Challenges with developing and testing

- Emulators, development boards, prototype boards
  - Reduces observability and controllability
- Complex test situation, many interactive components, timing is critical for behaviour

Lecture overview

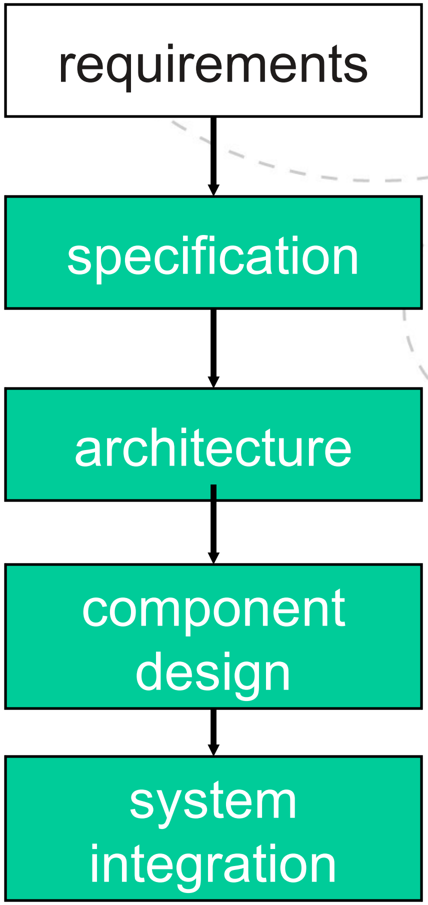
- Introduction
- Characteristics of embedded computers
- Embedded system design
- Design process examples

Design example: GPS

- A GPS device for car use
- Moving map
- Location from GPS
- Map from local storage

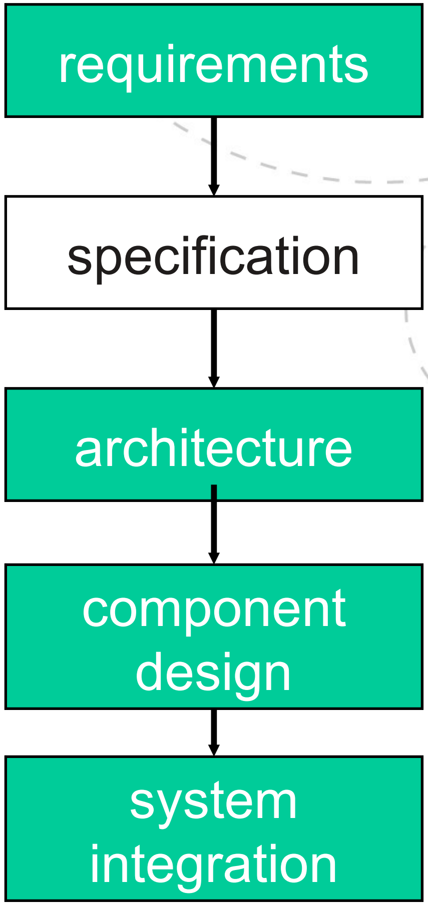
GPS: Requirements

- Show major roads and landmarks
- Output: 600x400 screen
- Input: Three buttons, popup menu
- Performance: Smooth map scrolling, 1s power up time, find position within 15s
- Cost: $500 street price
- Size: Fit dashboard
- Power consumption: Comparable to CD-player
- Function in -10ºC, storage -30ºC



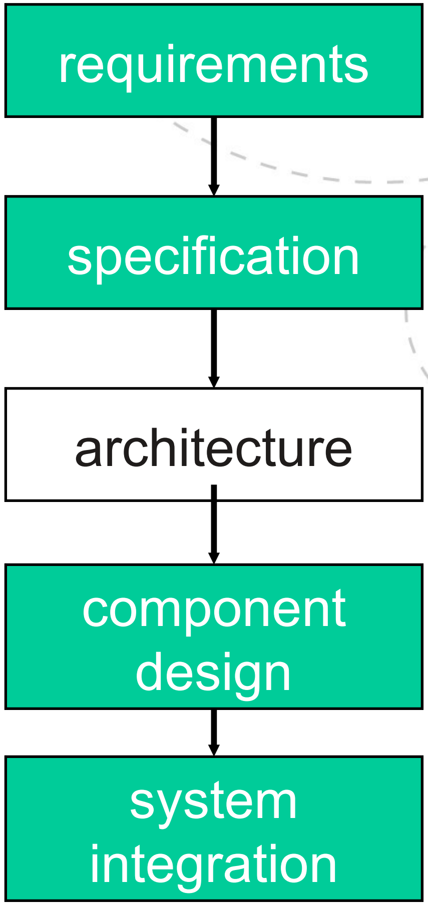
GPS: Specification

- More detailed than the requirements
- Unambigous
- Understandable
- All top level functionality and characteristics specified
  - Map data
  - GPS data
  - User interface
  - ...

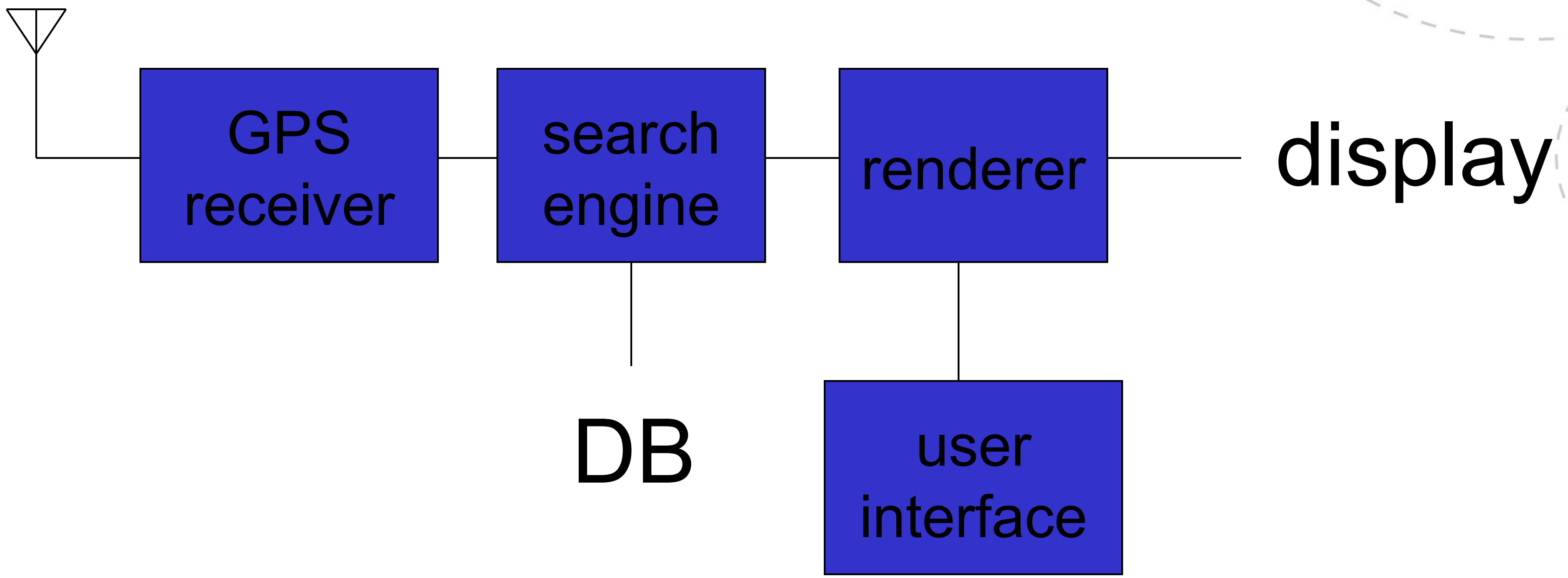


GPS: Architecture design

- Identify major components
  - HW and SW
  - Use off-the-shelf when possible

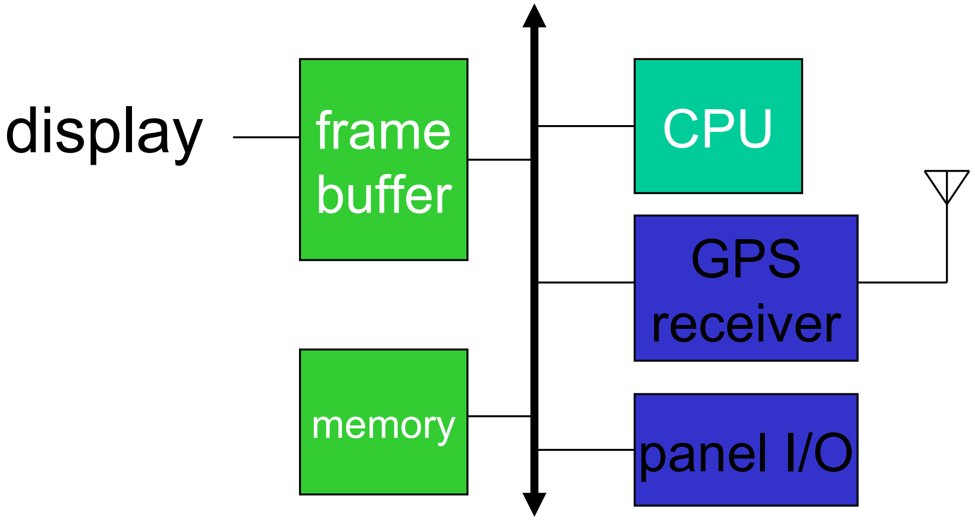


GPS: Architecture design

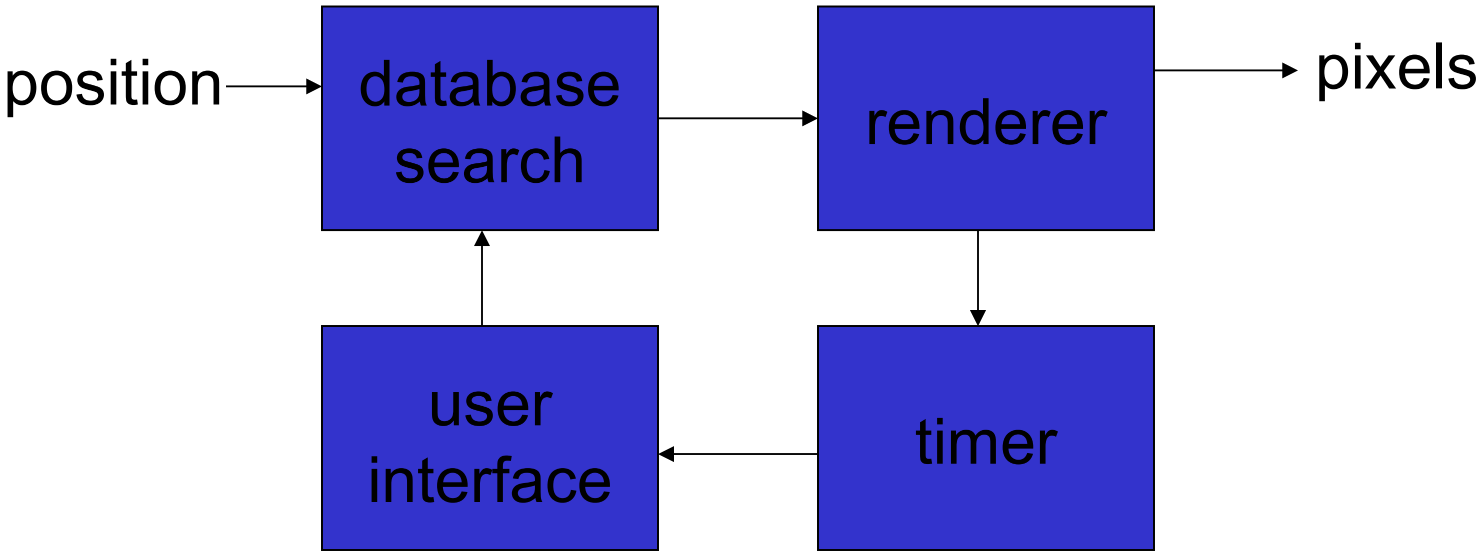


GPS: Architecture design

- Important high level choices
  - HW: CPU architecture, necessary I/O-devices, necessary HW accelerators (DSP, GPU, FPGA, ASIC, ...)
  - SW: Operating system, programming language
- Find a commercial development board that mostly fits
  - STK1000
  - RS232 GPS module
- Use devboard as reference for more detailed architecture

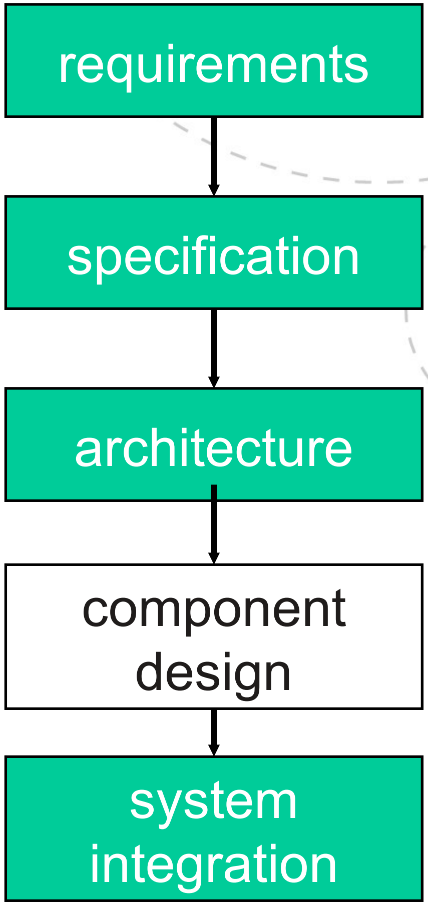


GPS: Software development



GPS: Software development

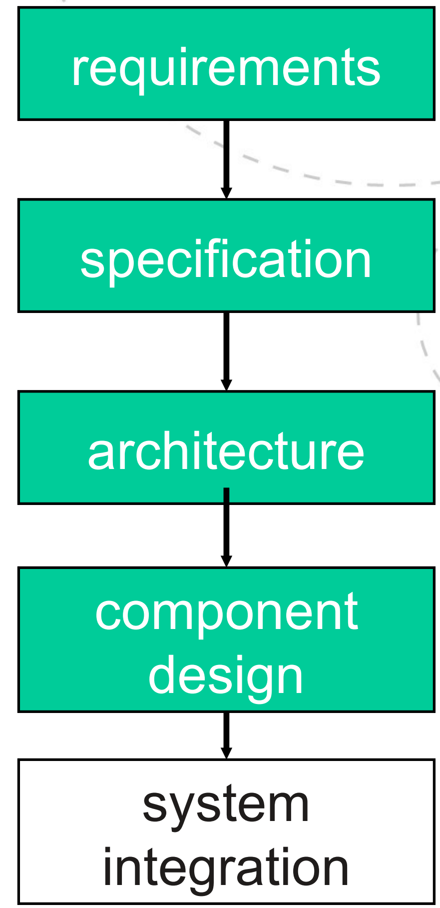
- Start developing for chosen devboard
- Use existing code when possible
  - OS
  - Map DB and SW
- Usually, most of the system can be finished before the HW prototype is done
- If HW prototype is delayed: Implement the rest of the system without HW
  - Simulate missing HW functionality



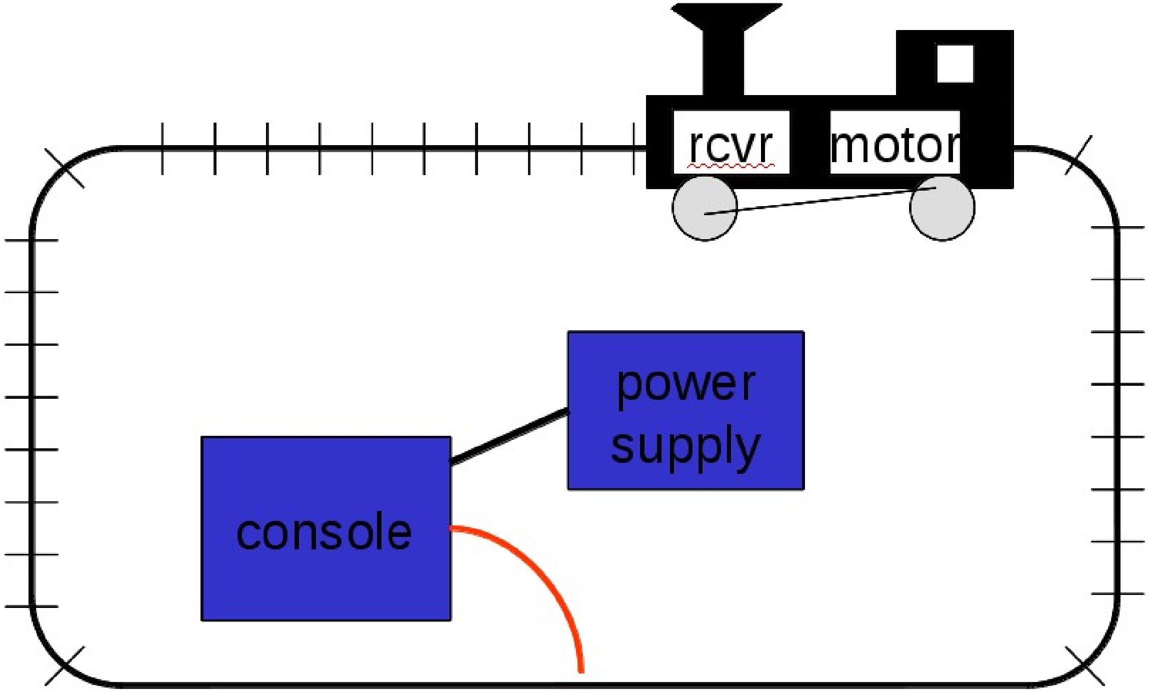
GPS: System integration

- Move SW over to HW prototype
  - Many bugs appear at this stage
- Test, debug, maybe redesign

Test as much functionality as early as possible (using devboard, emulators, test code, dummy modules, ...)

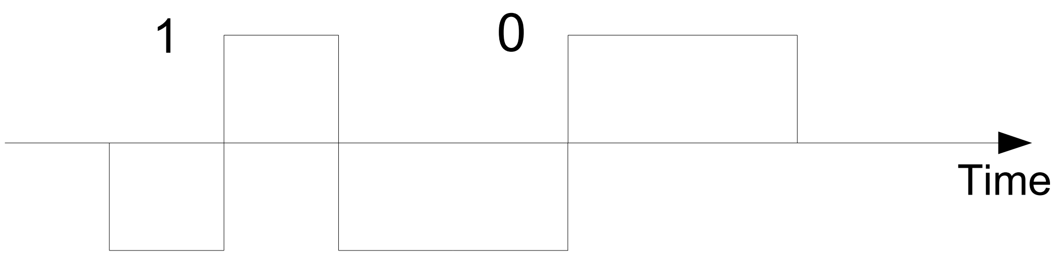


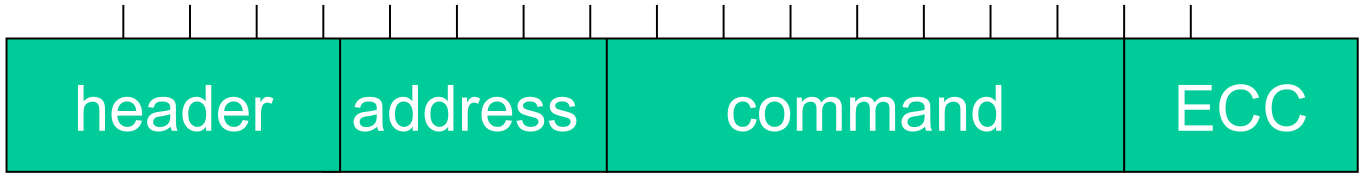
UML example: Model train controller



Digital Command Control (DCC)

- Standard for model train control
- Created by National Model Railroad Association

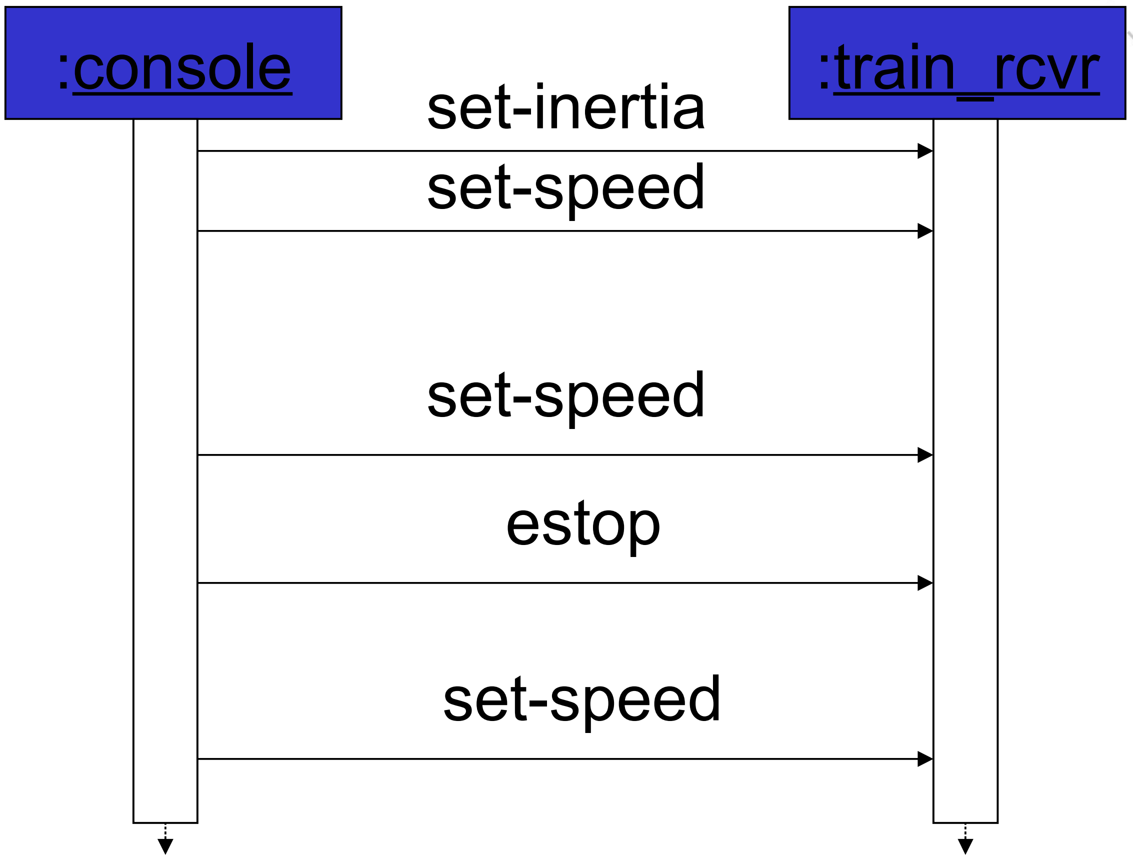




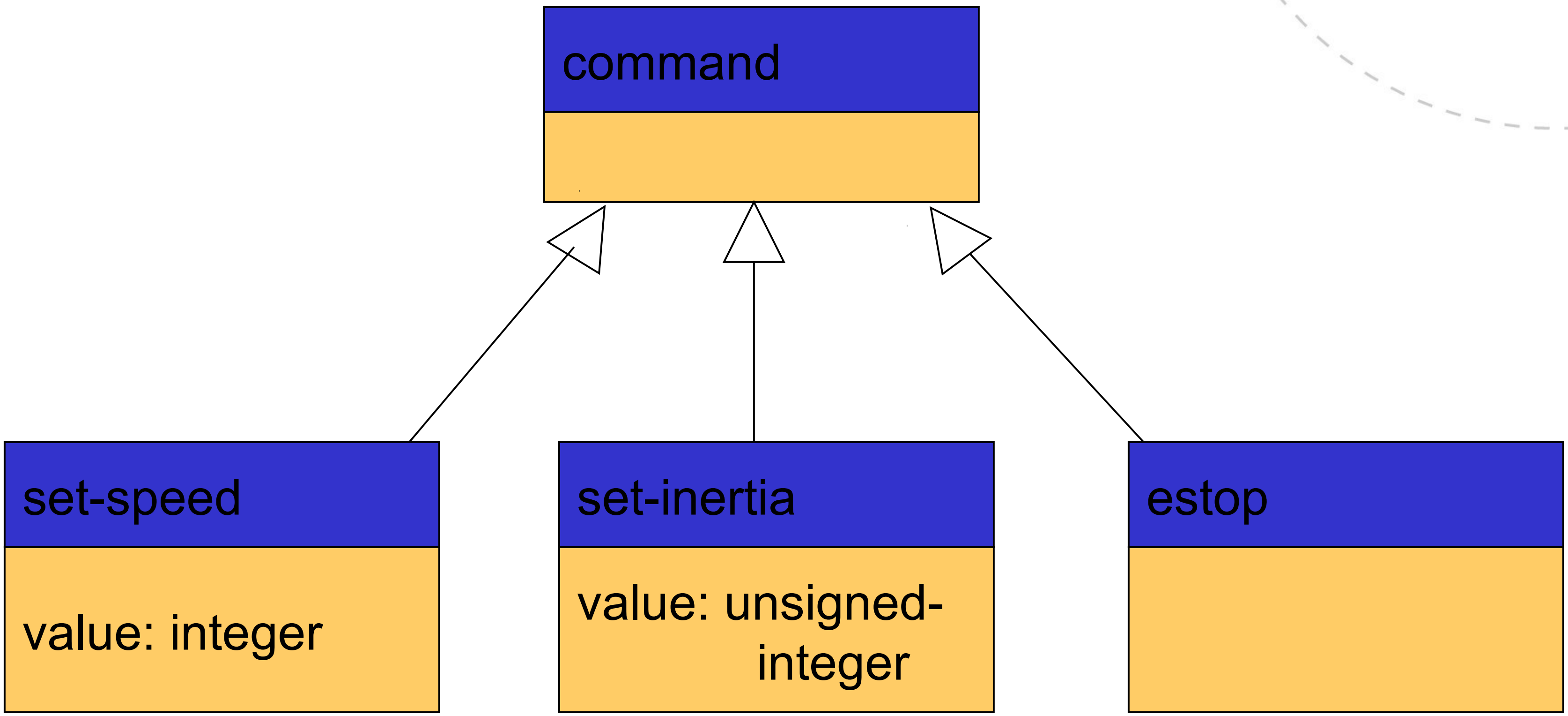
Requirements

- Console:
  - Control 8 trains on 1 track
  - Throttle with 63 levels
  - Inertia control (8 levels)
  - Emergency stop button
- Communication
  - Error detection scheme on messages

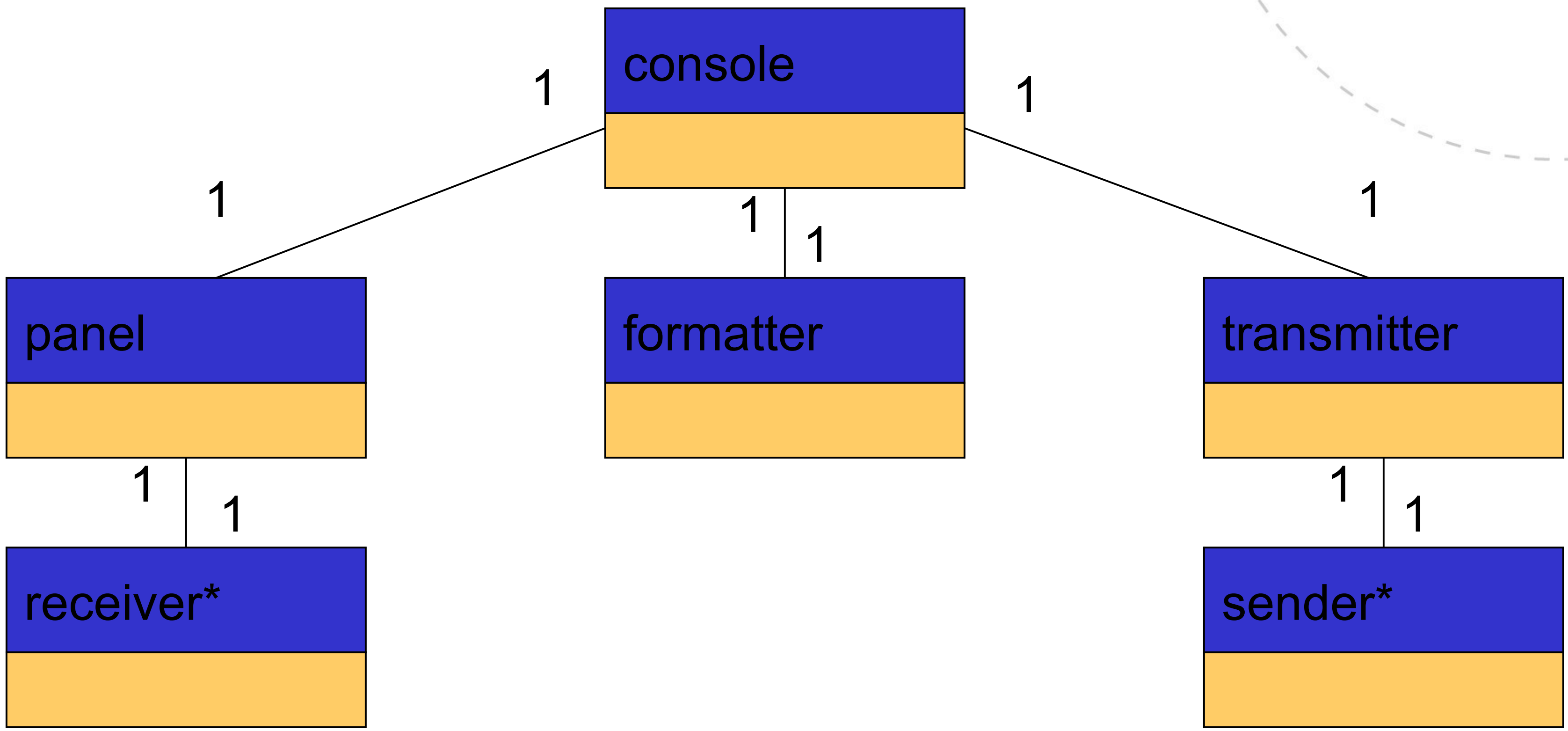
Typical sequence



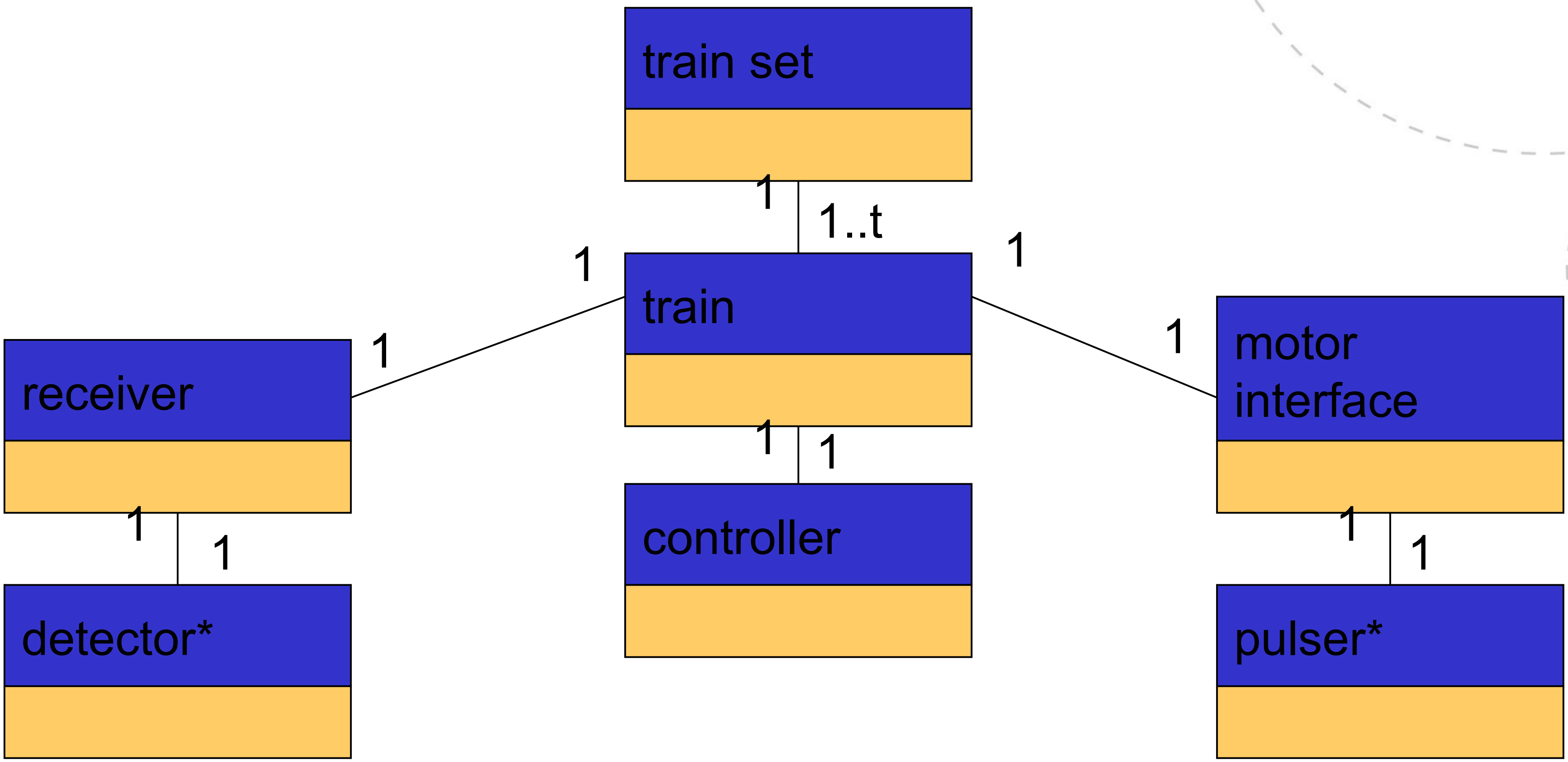
Message classes



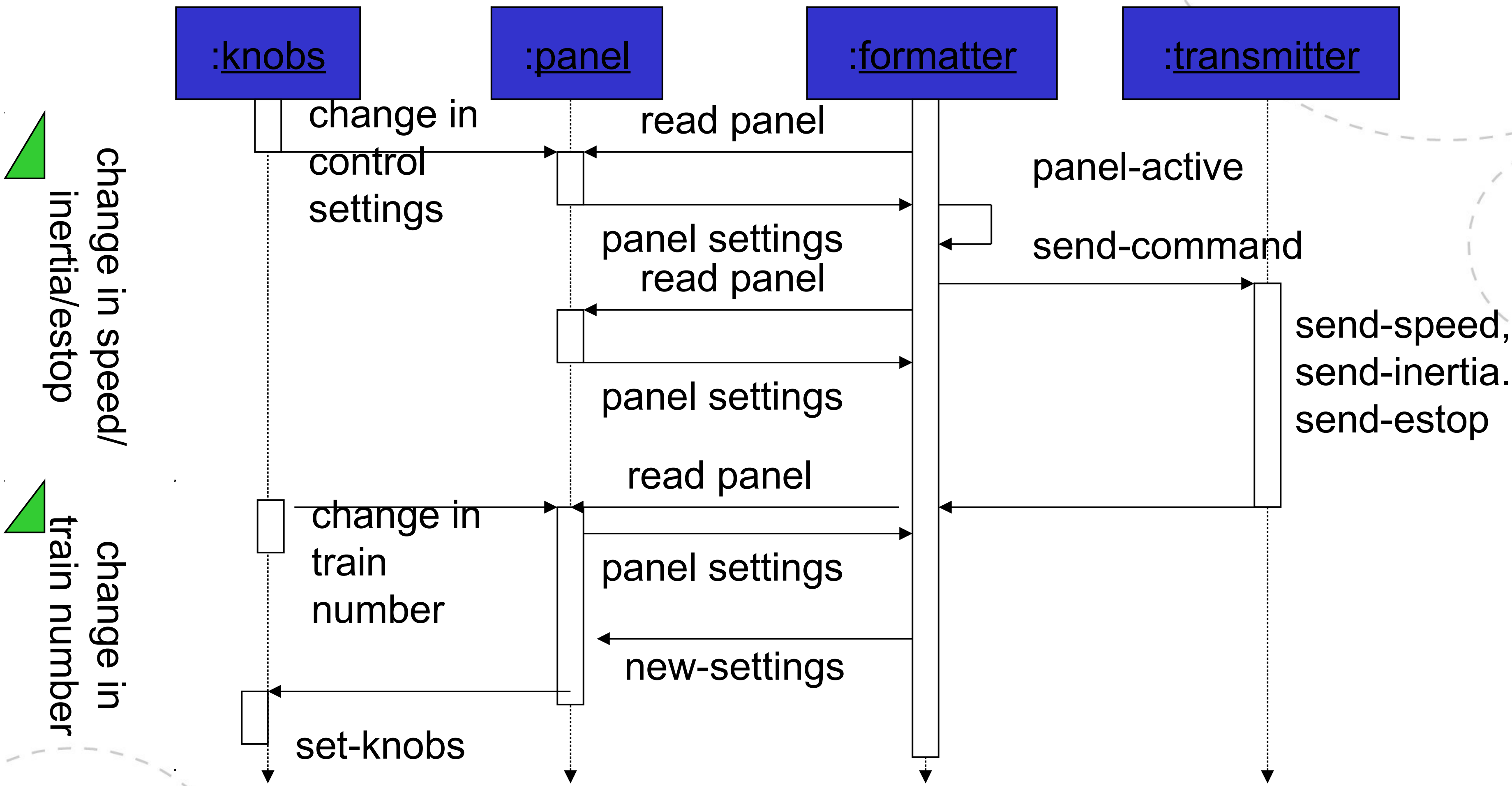
Console system



Train system



Control input sequence diagram



Next lecture

- Processor instruction sets
  - Assembly programming