A versatile demonstrator for distributed real-time systems: Using a model-railway in education

ERCIM / DECOS Workshop

Christian-Albrechts Universität Kiel
Faculty of Engineering
Dept. of Computer Science and Applied Mathematics
Real-Time Systems and Embedded Systems Group

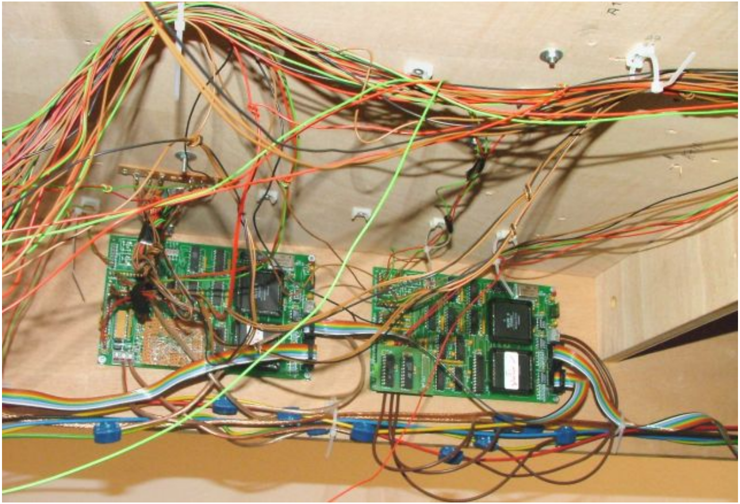
Stephan Höhrmann
Hauke Fuhrmann
Steffen Prochnow
Reinhard von Hanxleden

The Model Railway Installation

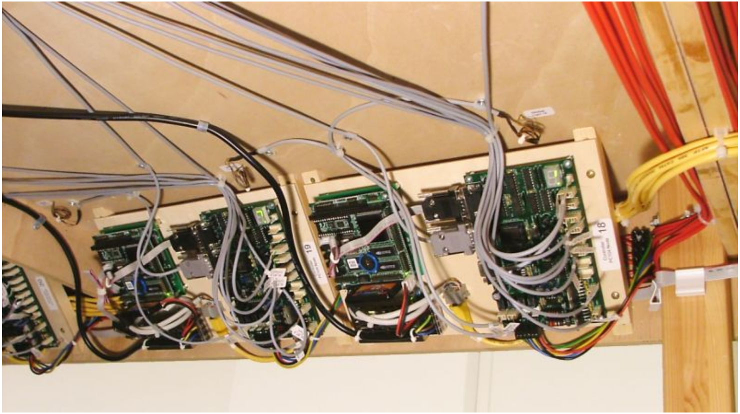
- Idea came up 1995 at research group of Prof. Kluge
- Based on a mountain pass in Canada: Kicking Horse Pass
- Originally to demonstrate on the management of resources (using Petri-Nets)
- Has been developed over three generations yet
- Scale H0, actually
  - 127 meters of railtrack
  - up to 8 trains concurrently
  - 28 switch points, 56 semaphores, 80 reed-contacts
  - 24 lanterns

First Generation

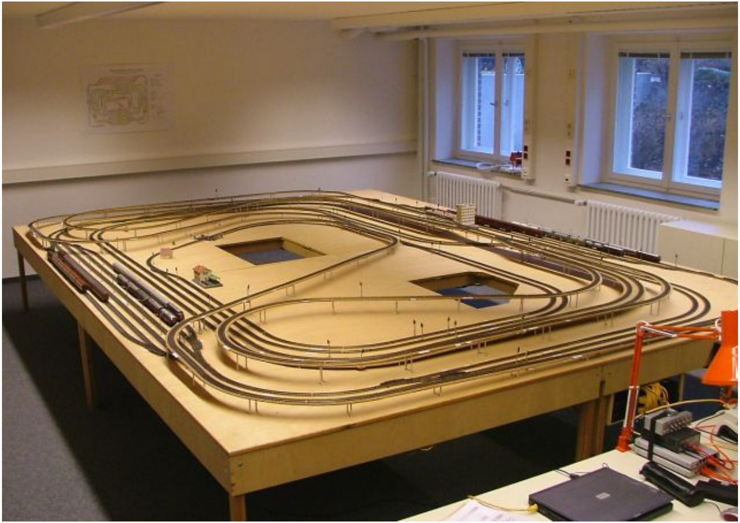
Second Generation



Third Generation



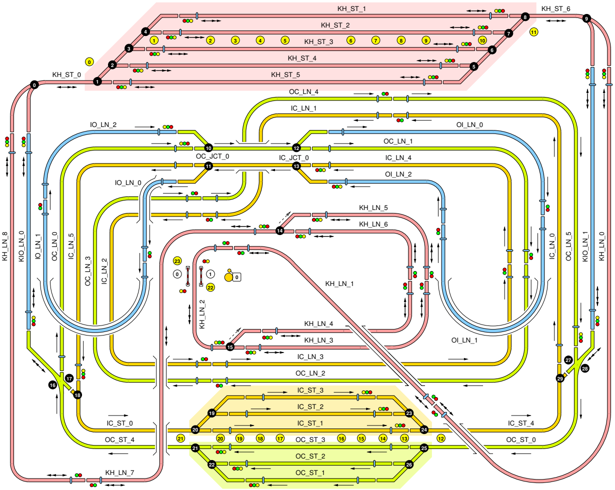
Third Generation



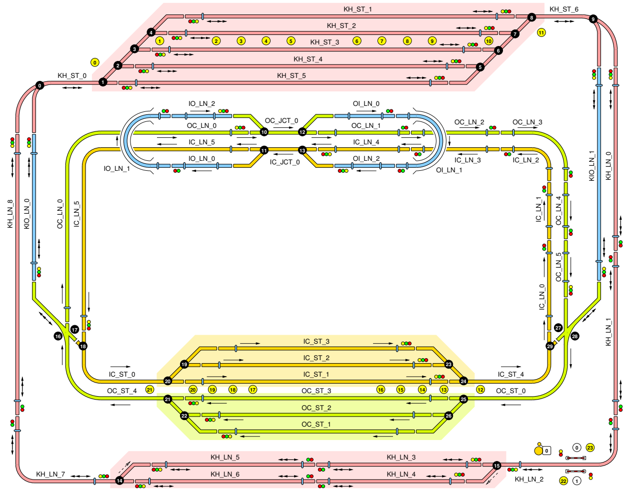
Third Generation

- Repair of damaged railway parts
- Complete replacement of controllers
  - Hardware Controller Circuits
  - Cabling
  - Network Bus Systems
  - Software
- Development of a new, modular and upgradeable controller system
  - CAN, TTP and Ethernet
- Multiple Programming Paradigms Possible
  - Standard C/C++ development with comfortable API
  - Model-Based System Design

Track Layout

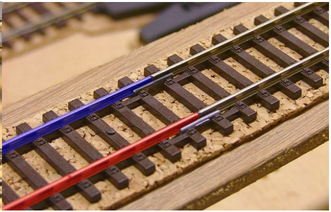


Simplified Track Scheme



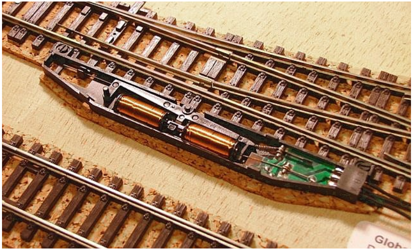
Periphery: Motor Power

- Engines are driven by 12V DC
- Polarity controls direction, PWM controls speed
- Railway tracks are seperated into distinct blocks

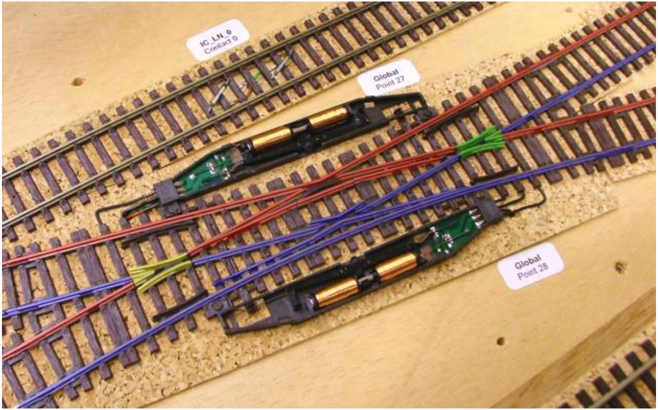


Periphery: Switch Points

- Switch points enable to switch between main and side tracks
- The guides are actuated electronically by solenoids
- Induces strong interfering fields!



Periphery: Crossing Switch Points



Periphery: Contacts

- Reed-Contacts embedded in the tracks detect magnets
- Small magnets attached to front and back of each train
- Task: Follow train positions, stopping at exact location
- Redundancy: Recognize direction of train

Periphery: Semaphores

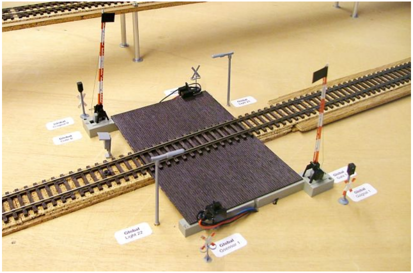
- Visualizing system states, realistic impression
- Main semaphores (red/green), block semaphores (+yellow)
- Controlled independently by software

Periphery: Lights

- Lanterns mark prominent locations (stations)
- Simply for neat looking
- Might be used for debugging (indicate some internal state)

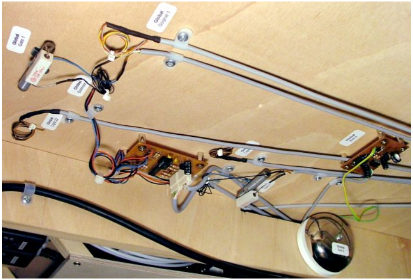
Periphery: Railroad Crossing

- Railroad crossing with gates, semaphores, bell and gate sensors
- Freely programmable

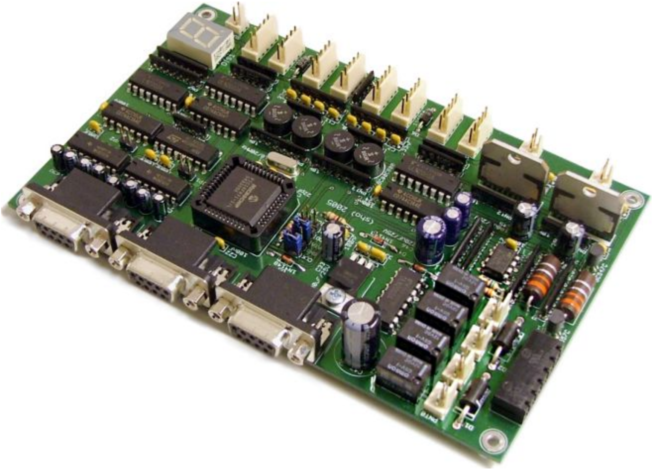


Periphery: Railroad Crossing

- Controlling by special interface circuits
- Generic interfaces allow connection of komplex sensors and actuator



Hardware Periphery Controller



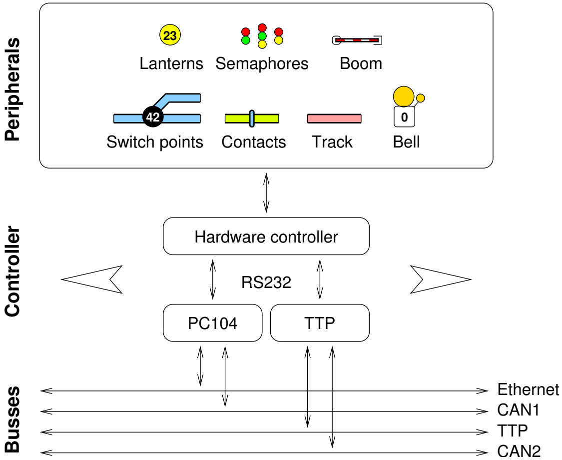
Hardware Periphery Controller

- Interface circuits controlls whole periphery of two blocks
- Connection to any bus systems possible
- Semaphores
  - 4 Outputs each for 3 LEDs (+ power supply)
- Contacts
  - 4 Inputs each for 2 Reed-Contacts (+ power supply)
  - complex filtering and evaluation, redundancy management
- Track Driver Units
  - 2 Outputs to the tracks, individual modes (off, fwd, rev, brake) and PWM, short-circuit safe
  - Integrated track occupied detector, measurement and controlling of engine speed

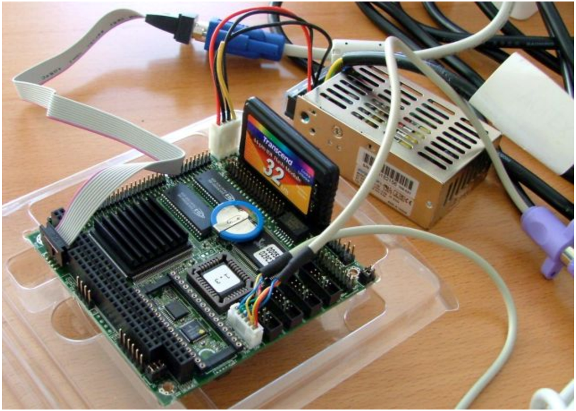
Hardware Periphery Controller

- Switch Point Drivers
  - 4 Outputs of switchable high power supply
  - Used for switch actuators, lights, gates, bell,...
  - Externel filters prevent influence of electrical interferences
- Serial Communication Interfaces
  - 4 Ports to connect up to 4 external computers
  - Controlling of circuit board by a serial protocol
  - Only one port active by a port choosing algorithm
  - 19200 Baud, 8N1, Fullduplex, Cycle-Time 10 ms max
- Error Protocol
  - EEPROM of Microcontroller saves faults of reed-contacts, short-circuits and other critical faults
- Integrated Display
  - 7-Segment-Display indicates state of the board

Controlling: Modular Concept



Controlling: PC104-Computer



Controlling: PC104-Computers

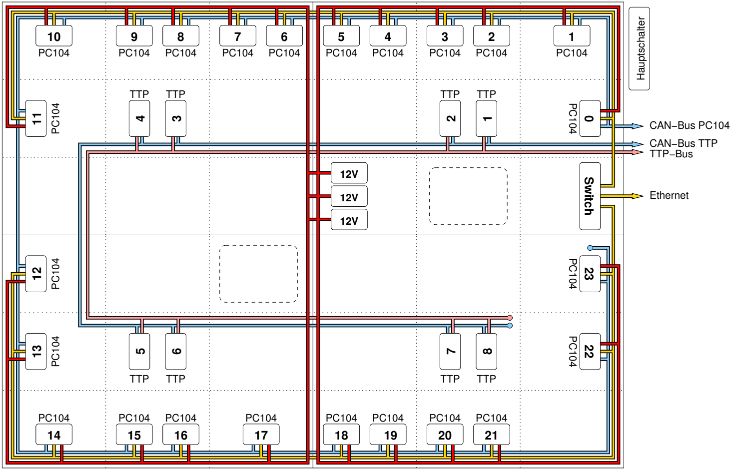
- CPU 80386, 40 MHz, 4 MB RAM, 32 MB Flash-Disc
- 4 serial interfaces, Ethernet, CAN-Controller
- Boots and operates completely via network (Kernel, Root, Home, Swap)
- Custom compiled Linux and cross compilation toolchain
- Daemon *railwayd* is a router between Ethernet/CAN and the serial interface

Controlling: TTP Powernodes

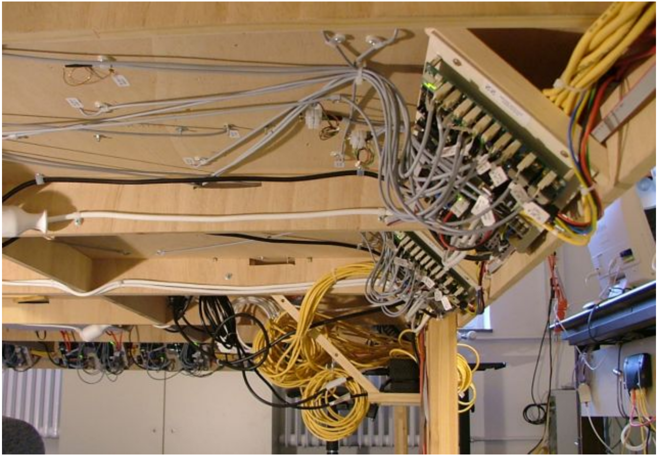
Controlling: TTP-Powernodes

- Motorola MPC555 with 40 MHz, PowerPC-Architecture with floating point unit
- 1 MB RAM, 4 MB Flash
- Time Triggered Architecture
- Programming either directly in C or model-based design
- Three serial interfaces by custom port-extender
- Employed in education with model-based system design already

Controlling: Crosslinking the Components



Controlling: Cabling



Controlling: 12 Volt Power-Supply

Controlling: TTP-Powernodes

Balance Sheet

- 50 weeks of work
- About 4000 working hours
- 15000 lines of code (Assembler and C)

Advanced Laboratory Course in University Education

- Time Frame: About 4 Months, 4 or 8 semester periods a week
- Students: 8 Students, forming teams of 2
- Background:
  - General knowledge about model-based system design
  - Detailed knowledge about statecharts semantics
  - No experience with Matlab/Simulink/Stateflow
  - Little experience with C programming
- Goal: Independent working in a bigger project
  - Application specification
  - Time planning (dividing work into 4 milestones)
  - Problem solving and team interaction
  - Documentation
  - Presentation (internal milestone and final public presentations)

Application

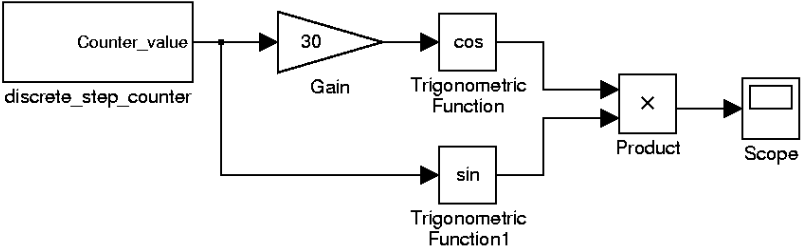
- Introduction to the tools
- Control the trains!
  - Definition of concrete task defined by students according to time frame themselves
  - Simulation prior to testing on real hardware was required

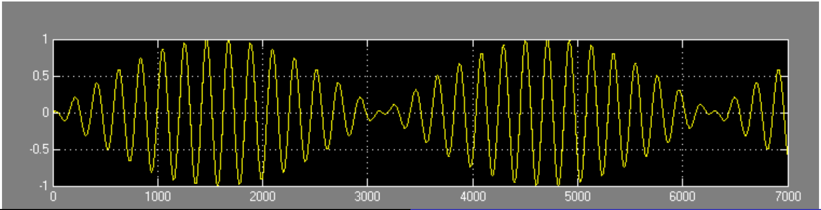
$\Rightarrow$

- One Group: Modular, clear simulation model of whole installation
- Other Groups: Controllers for the railway

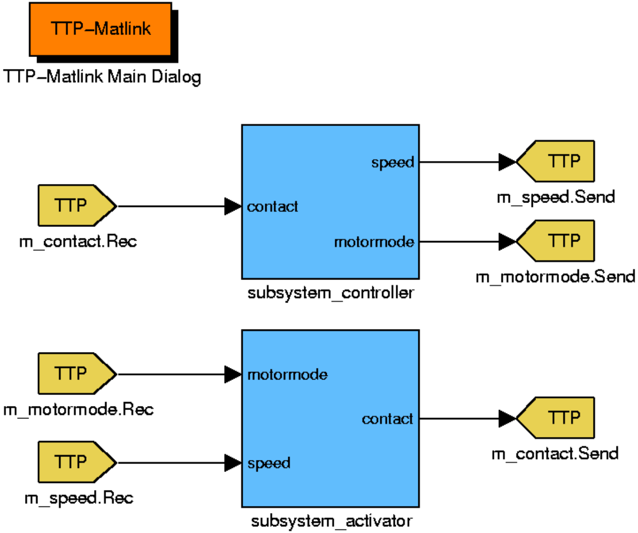
Model-Based System Design: Matlab/Simulink/Stateflow

Graphical representations model system behaviour and can be executed in simulation.

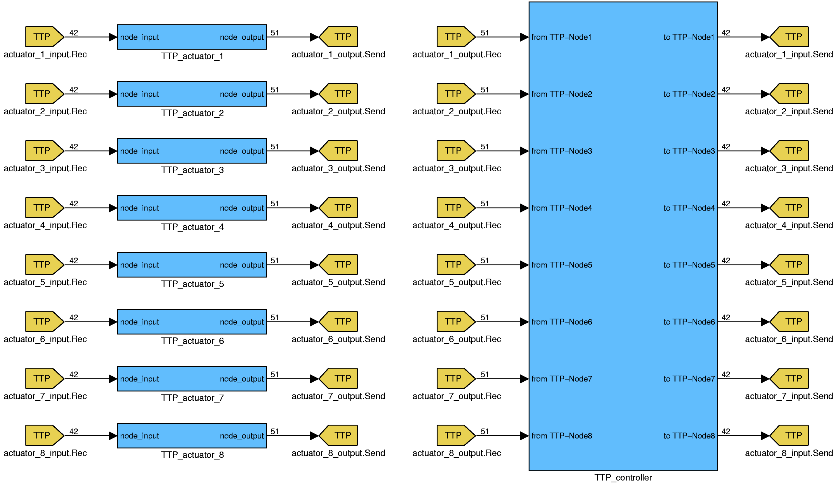




Matlink (TTTech) to specify communication in Simulink



Communication Model for the Railway Application



Lessons Learned

What did we learn from the project?
Concerning...

- ... the application
- ... the students
- ... the tools

Motivation

- Motivation of students very high.
- Presence of physical demonstrator increased motivation.
- Spent much more time (ca. double) than scheduled.

Time Planning

- Time plannings were too optimistic.
- All groups massively underestimated efforts for the individual tasks.
- Hardly able to finish tasks in time.
- Time efforts more than doubled.
- Some goals of milestones postponed or completely removed.

Preparation

- Pressure of time lead to more unstructured process.
- Quick-and-dirty prefered to “time consuming” preparations.
- Prefered trial-and-error over tutorial usage and manual studies.
- Prefered debugging on real hardware to usage of simulation.

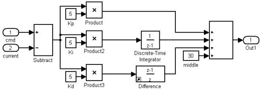
Application

- Closed loop controllers
- Simulation of railway

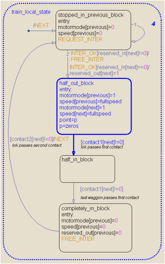
- Reactive Controllers
- Train management (Safety, Deadlock avoidance, Fairness)

- Graph algorithms
- Find shortest path

Data Flow



Statecharts



?

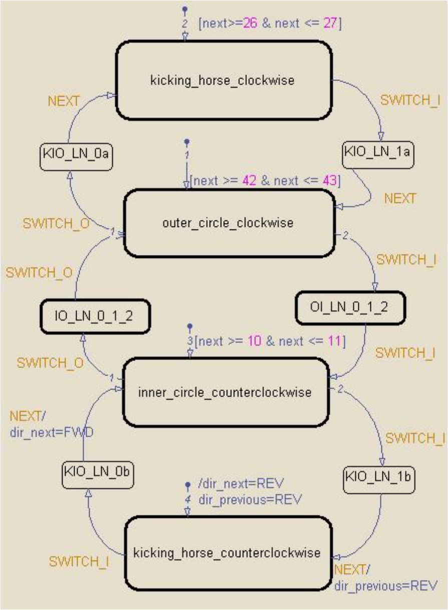
Simulation

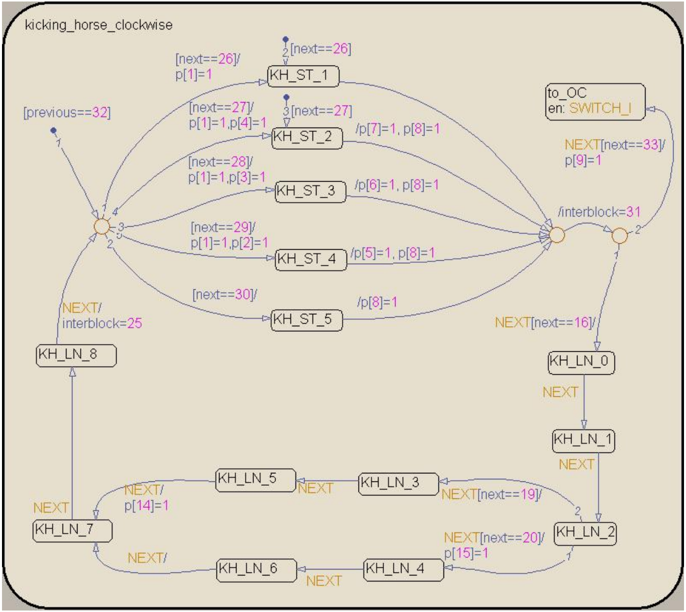
- Building Simulation Model very effort prone
- Development of controller without simulation hopeless
- Debugging of controller with simulation is comfortable
- Porting of controller from simulation to real hardware worked seamlessly
- Simulation for this scale of application mandatory

Experience at Modeling

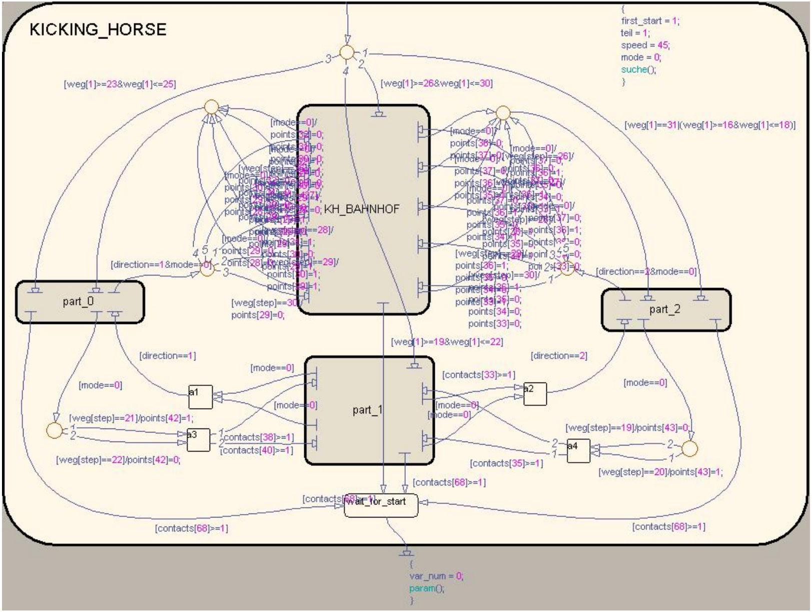
- Models are easy to read but hard to write!
- Good modeling differs from good programming in textual languages
- Common design patterns are hardly applicable
- Quick-and-dirty prefered to "beautiful" models
- Abusing of mechanisms makes reading of models harder
- examples...

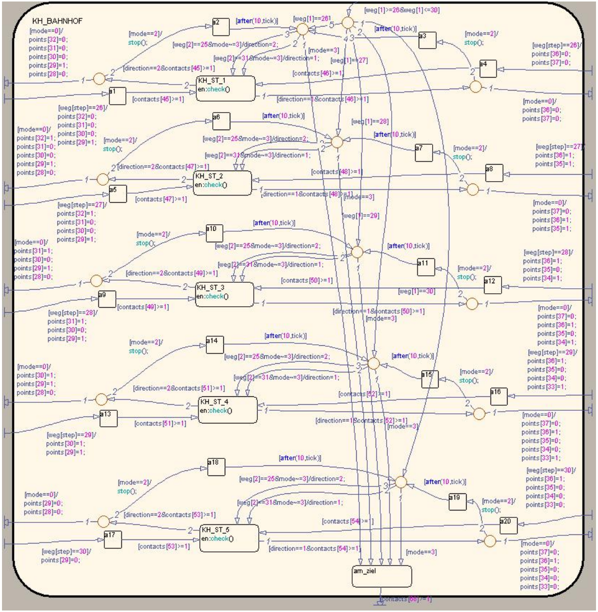
Well structured Statechart

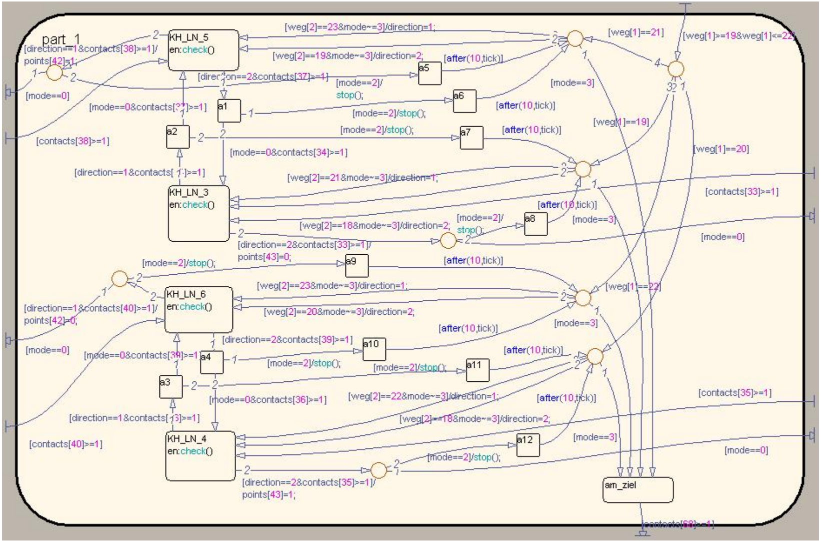



kicking_horse_clockwise
[next==26]
[next==26]/
p[1]=1
KH_ST_1
to_OC
en: SWITCH_I
[previous==32]
[next==27]/
p[1]=1,p[4]=1
[next==27]
KH_ST_2
/p[7]=1, p[8]=1
NEXT[next==33]/
p[9]=1
[next==28]/
p[1]=1,p[3]=1
KH_ST_3
/p[6]=1, p[8]=1
/interblock=31
[next==29]/
p[1]=1,p[2]=1
KH_ST_4
/p[5]=1, p[8]=1
[next==30]/
/p[8]=1
NEXT/
interblock=25
KH_ST_5
NEXT[next==16]/
KH_LN_8
KH_LN_0
NEXT
KH_LN_1
NEXT
KH_LN_5
NEXT
KH_LN_3
NEXT/
p[14]=1
NEXT[next==19]/
NEXT
KH_LN_7
KH_LN_2
NEXT[next==20]/
p[15]=1
NEXT/
KH_LN_6
KH_LN_4
NEXT

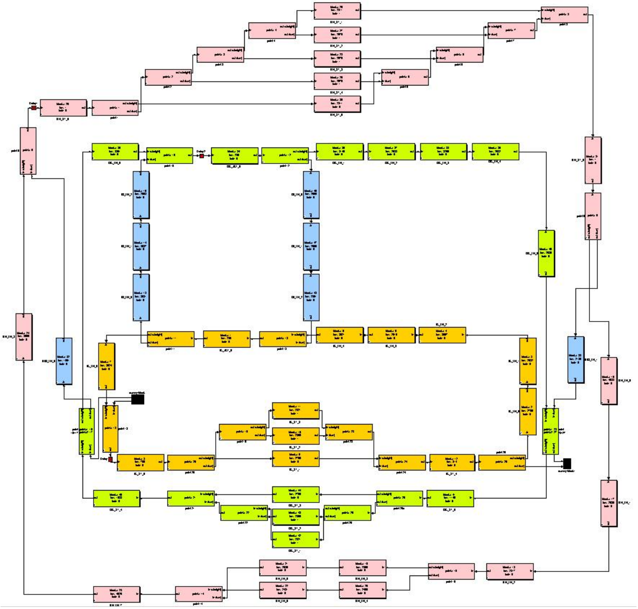
A tangled mess of wiring



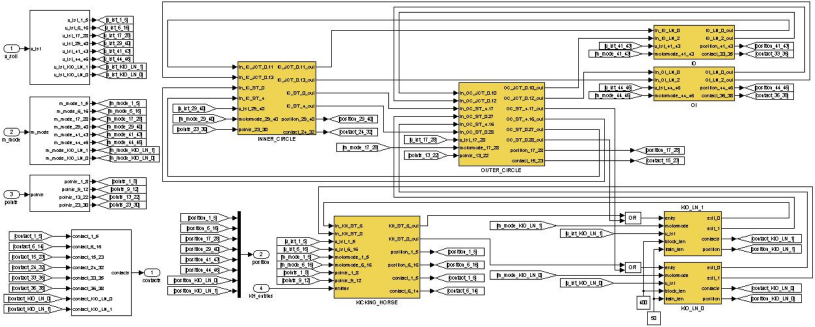

KH_BAHNHOF
KH_ST_1
en:check()
KH_ST_2
en:check()
KH_ST_3
en:check()
KH_ST_4
en:check()
KH_ST_5
en:check()
am_ziel
[after(10,tick)]
[mode==2]/
stop();
[mode==3]


part_1
KH_LN_5
en:check()
KH_LN_3
en:check()
KH_LN_6
en:check()
KH_LN_4
en:check()
am_ziel
a1
a2
a3
a4
a5
a6
a7
a8
a9
a10
a11
a12
[direction==1&contacts[38]>=1]/
points[42]=1;
[mode==0]
[contacts[38]>=1]
[mode==0&contacts[37]>=1]
[direction==2&contacts[37]>=1]
[direction==1&contacts[ ]>=1]
[weg[2]==23&mode~=3]/direction=1;
[weg[2]==19&mode~=3]/direction=2;
[after(10,tick)]
[mode==2]/
stop();
[mode==2]/stop();
[mode==0&contacts[34]>=1]
[weg[1]==21]
[weg[1]>=19&weg[1]<=22]
[mode==3]
[weg[1]==19]
[weg[1]==20]
[weg[2]==21&mode~=3]/direction=1;
[weg[2]==18&mode~=3]/direction=2;
[contacts[33]>=1]
[mode==0]
[direction==2&contacts[33]>=1]/
points[43]=0;
[weg[2]==23&mode~=3]/direction=1;
[weg[2]==20&mode~=3]/direction=2;
[weg[1]==22]
[direction==1&contacts[40]>=1]/
points[42]=0;
[mode==0&contacts[39]>=1]
[direction==2&contacts[39]>=1]
[mode==0&contacts[36]>=1]
[direction==1&contacts[ ]>=1]
[contacts[40]>=1]
[weg[2]==22&mode~=3]/direction=1;
[weg[2]==18&mode~=3]/direction=2;
[contacts[35]>=1]
[direction==2&contacts[35]>=1]/
points[43]=1;
[contacts[ ]>=1]

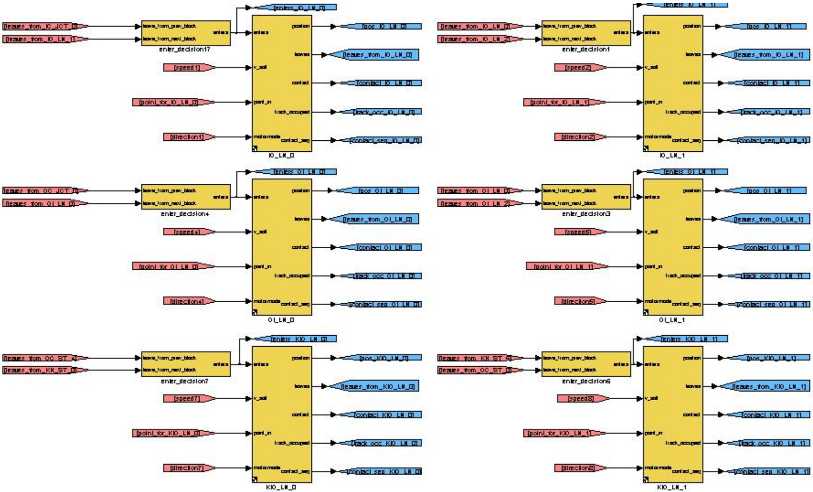
Very well readable simulation model



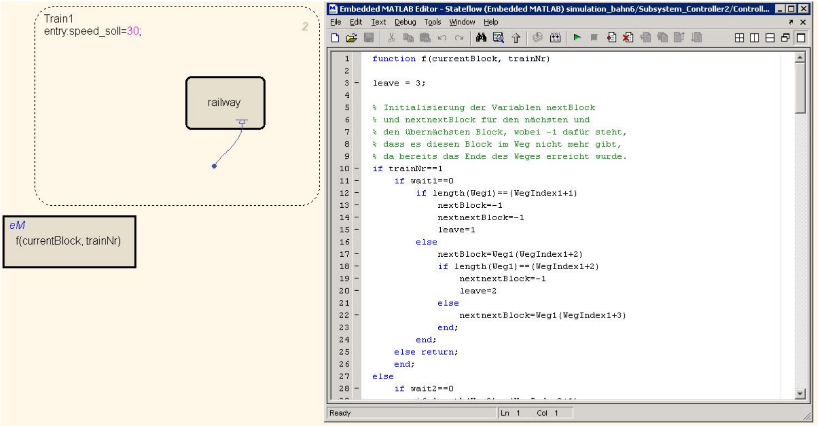
Long connections, many inputs and outputs



Abusing the *goto* and *from* mechanism



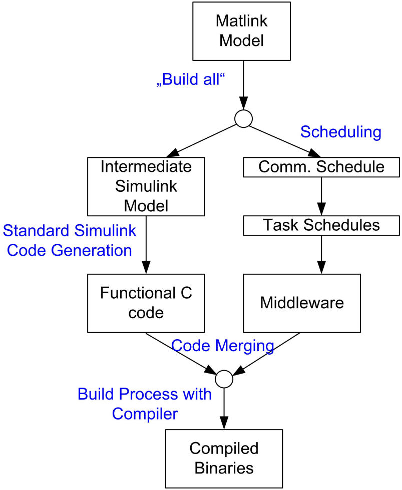
Abusing the *embedded Matlab function* mechanism



Interaction with tools

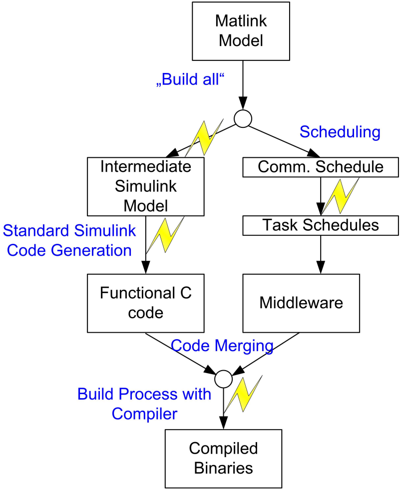
- Toolchains can comprise many hierarchy levels
- *e. g.* modeling, transforming, scheduling, code generation, building, deployment
- Most time consuming problems: Origin of error messages are not clear to the developer $\longrightarrow$ effort prone debugging necessary
- Lower level error messages cannot be mapped to the higher levels

Overview: The build process of Matlink



- Matlink model is “terrain” of the developer
- Building process generates functional code and middleware
- Middleware requires scheduling with two external tools
- Simulink Code Generator is used with an “invisible” intermediate Simulink model

Problems at error propagation



- Simulink error messages address Intermediate Simulink Model
- Error at task schedules but communication schedule is too tight
- Scheduler messages hardly relate to Matlink objects
- C "parse error near ;" due to wrong option in Matlink model

Perfect Process

- Developer can stay at the topmost hierarchy level
- Requirement: Good communication between levels
- Consistent error propagation from lower levels to each next higher level
- Requirement: Consistent mapping of objects in different toolchain levels

Summary

- Versatile Demonstrator
  - Multiple Bus Systems
  - Multiple Programming Paradigms
- Scale of Model-Railway yields to challenging application
- Model-Based System Design employed
  - Graphically presents Functionality
  - Simulation possible $\Rightarrow$ Enables Debugging
  - Easy to read but hard to write
    - Unclear error messages from tools
    - Uncommon Design Patterns
    - Difficult to right "beautiful" models

Thank you!