DaVinci –
Development of
Distributed Automotive Software

Matthias Wernicke, Vector Informatik GmbH

Mathworks IAC,
June 16th 2004, Stuttgart
Agenda

- Motivation
- DaVinci methodology
- DaVinci and Simulink
- Conclusion
Motivation: How to design and reuse distributed functions?

Function Library
- Seat Adjustment A
- Seat Adjustment B
- Lighting
- Seat Heating
- Air Conditioning

Hardware Topology

Software Configuration

Distributed System

Code Integration

Vehicle A

Vehicle B
Motivation: How to automate the integration of ECU code?

- Integration of applications with complex ECU software architectures
  - Real-time OS
  - Bus drivers, interaction layers
  - Network management standard modules
  - Diagnostics standard modules
  - I/O-Drivers
  - other standard modules (calibration, power management, ...)

- Integration of legacy code

- Support of future ECU target architectures (AUTOSAR)
Agenda

- Motivation
- DaVinci methodology
- DaVinci and Simulink
- Conclusion
Vehicle functions are modeled by SW components

Formal requirements specification
  - Interfaces: Input- and Output-signals
  - Point-to-point latencies between the inputs and outputs

Optional: Attachment of additional specification documents
Hierarchical decomposition possible (sub-structure)
DaVinci methodology: ECU Framework

- Specify the signal interfaces of the ECUs
  - Sensors
  - Actuators
- I/O Firmware Implementation
  - Provide the I/O drivers for signal access
  - Attach additional firmware files (e.g. EEPROM drivers)
- Result: ECU framework capable of running SW components
DaVinci methodology: Mapping

- Create a distributed system
  - Step 1: Component Mapping
    - Assign each component to one of the ECUs
    - Result: Bus signals and ECU internal signals
  - Step 2: Communication Mapping
    - Specify bus messages
    - Assign the bus signals to these messages

- Process flexibility: Bus communication may be
  - defined by DaVinci (dbc file is generated)
  - externally defined and imported in DaVinci

- Model may be used as specification for ECU suppliers
Behavioral design: Implementation of a SW component

- Implementation alternatives e.g.
  - Finite-State-Machine
  - C-Code
  - Truth table
- Heterogeneous systems possible
- Consistent versioning of specification and implementation

DaVinci Tool Suite

**DaVinci methodology:** Behavioral Design

```c
void LightingFunction(void) {
    ...
    ...
}
```
DaVinci methodology: ECU Code Integration

- Configuration of target architecture
  - Assign the SW components to OS tasks
  - Specify scheduling parameters (e.g. cycle times)

- Automatic generation process
  - Generate DaVinci "glue code" for signal access
  - Integrate behavior code
    - Generated from the Behavior Modeling Tool
    - Hand-coded SW components
  - Generate the OS, CAN-Driver etc.
  - Generate make-files, compile and link
DaVinci methodology: ECU Target Architecture

- Flexible target package concept
  - various CPUs
  - specific communication stacks of the OEMs
- PC experimental target available

Diagram:
- OSEK OS
- Standard applications
- Standard components
- Interaction Layer
- CAN, LIN, ... Driver
- Communication Hardware (CAN, LIN, ...)
- I/O Hardware
- Component
- DaVinci-Application
- Generated by production code generators, or hand-coded
- Option ally generated by DaVinci
- Automatically configured
- I/O Firmware

© 2004. Vector Informatik GmbH. All rights reserved. Any distribution or copying is subject to prior written approval by Vector.

V1.0 2004-06-14
DaVinci methodology: Experimental Target

- **Goal:** simulating, testing, monitoring, debugging
  - of functionality and/or network communication
  - in early design phases (e.g. real target not yet available)

- **Solution:** CANoe as simulation backplane
  - High quality of simulation: Original target architecture on PC
  - Operating system emulated by CANoe (osCAN library)
  - CANoe executes the fully implemented code
  - Interactive tests via panels
  - Automated tests via scripts
Agenda

- Motivation
- DaVinci methodology
- DaVinci and Simulink
- Conclusion
Simulink model framework is derived from a DaVinci SW component
Special Simulink Library block for DaVinci Signals
  - Reading/writing a signal value
  - Getting signal state information
  - Control of network management

SW component behavior is designed in a Simulink subsystem
No restrictions for the design (Stateflow, Standard blockset, ...)
DaVinci SW Component

Equivalent Simulink model

Automatically generated model framework

DaVinci block

Behavior model (FSM, ...)

DaVinci and Simulink: Integration (models)
RTW-GRT is used for prototyping code generation
DaVinci Blocks are realized by S-Functions
Scaling is managed through floating point signal access macros
Each DaVinci component gets its own Simulink runtime driver instance
Multiple Simulink models within a component shares one Simulink runtime driver
- Generation of Simulink model framework for a DaVinci SW component
- Import of .mdl files (generate DaVinci SW components)
- Merge SW component models into an overall .mdl file
Agenda

❑ Motivation
❑ DaVinci methodology
❑ DaVinci and Simulink
❑ Conclusion
Conclusion

- Features of DaVinci
  - Structured software design for reusable applications
  - Integration environment for applications
  - Integration environment for diagnostics
  - Test support with CANoe
  - Support of configuration management
  - Seamless Simulink integration

- Benefits for Simulink users
  - Automated generation of efficient ECU target architectures
  - Simulation including network behavior and OS behavior
  - Easy migration/integration of legacy code

More infos? Please visit the Vector booth in the exhibition!