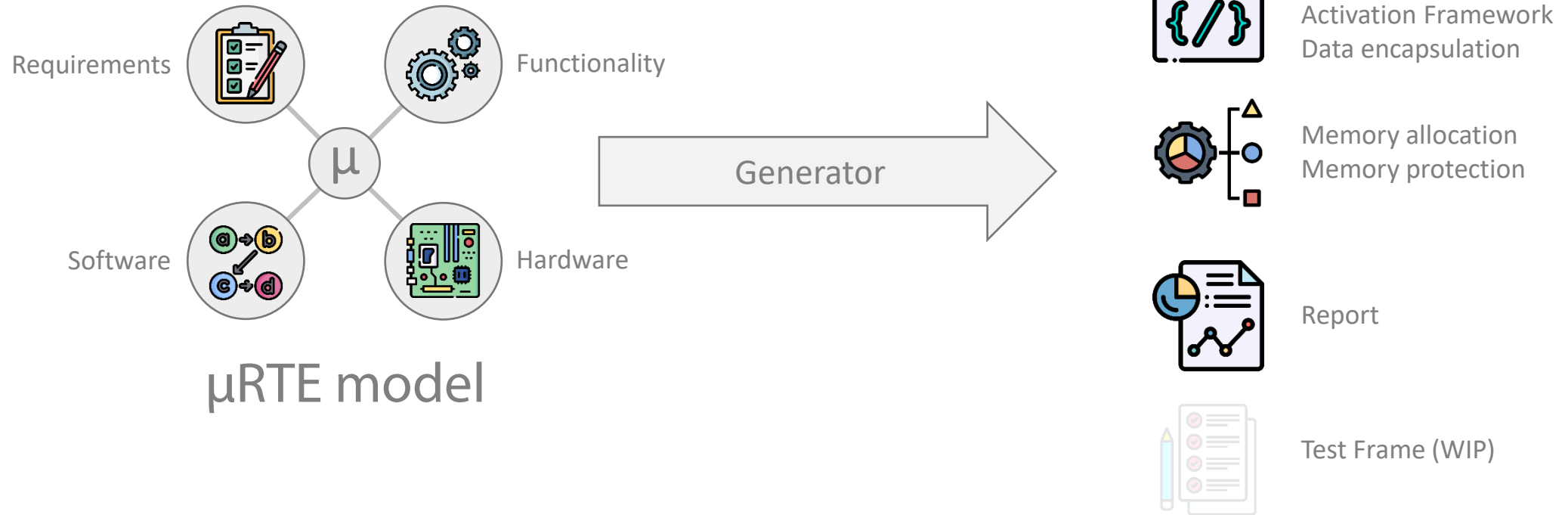


μRTE

A modeling framework for safe code on (Multi-Core) Controllers

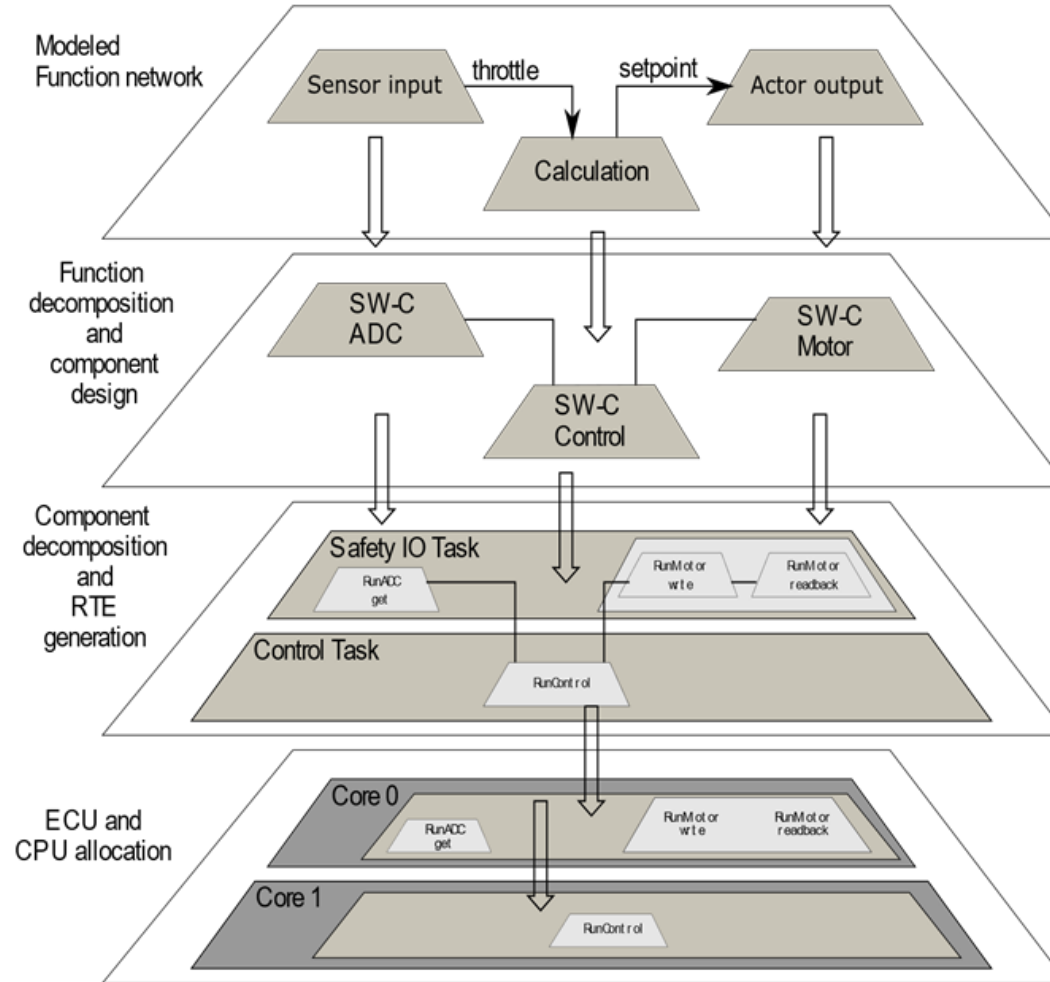
Modeling Overview

Scope



Software modeling

Abstract overview

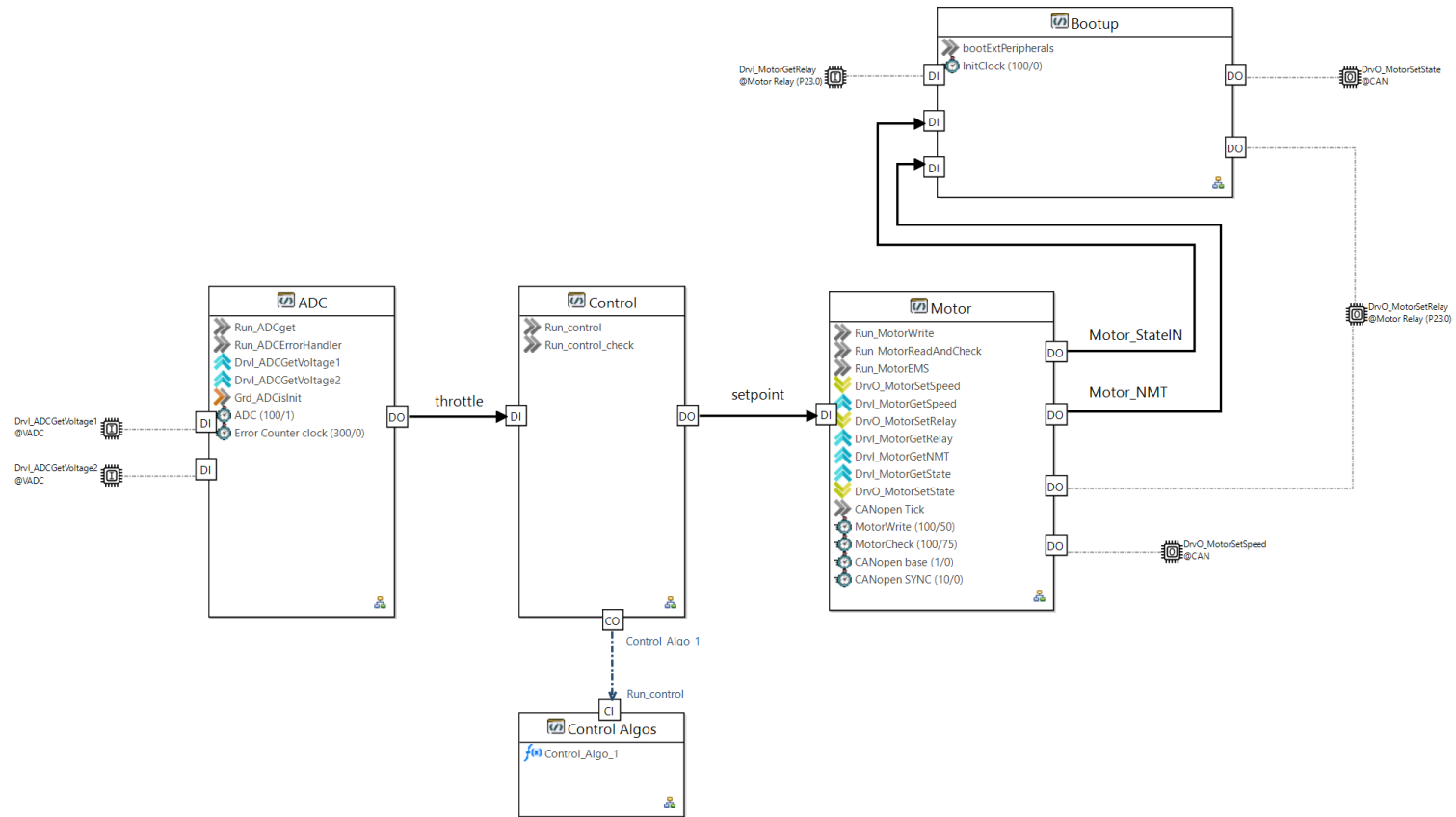


μRTE Modeling overview

Software modeling

Each block represents a code file in which software functions are contained
Arrows represent data exchange
Chip-Icons represent hardware interfaces

- μRTE System Model
 - Software Model
 - Application
 - LED
 - Throttle Control
 - SWC ADC
 - SWC Control
 - SWC Motor
 - SWC Bootup
 - SWC Control Algos
 - SWC LED
 - Source-Folder Complex Device Drivers
 - Source-Folder Algorithms
 - RTE
 - RTOS
 - Linkage
 - DataTypes
 - Hardware Model
 - Logical Function Model
 - Requirement Model



Software modeling

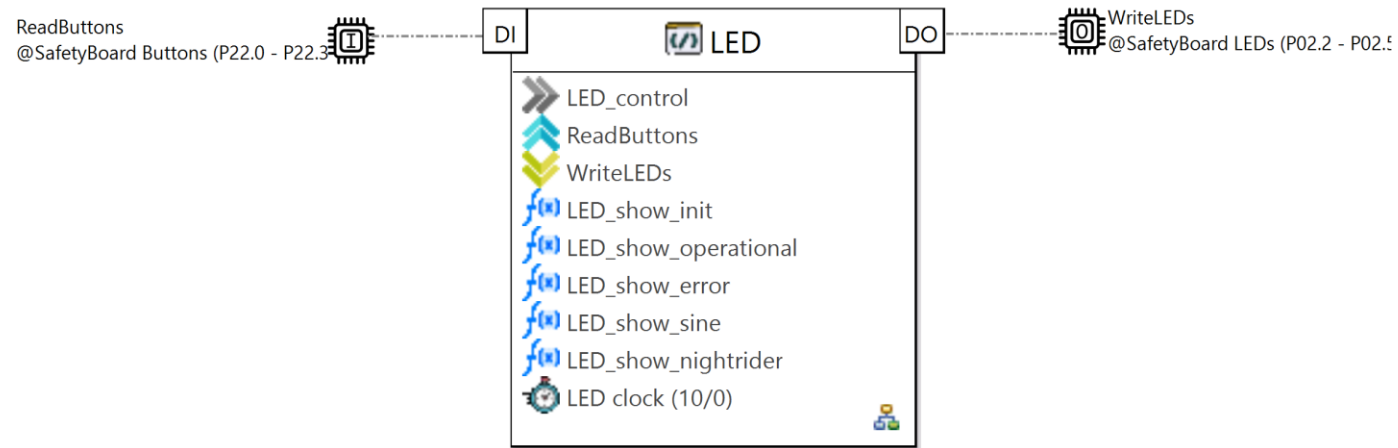
General concept:

Multiple diagrams within the same model possible with selectable data visualized.

Each graphical element can be selected and edited directly.

Rich toolset for diagram editing based on the SIRIUS framework.

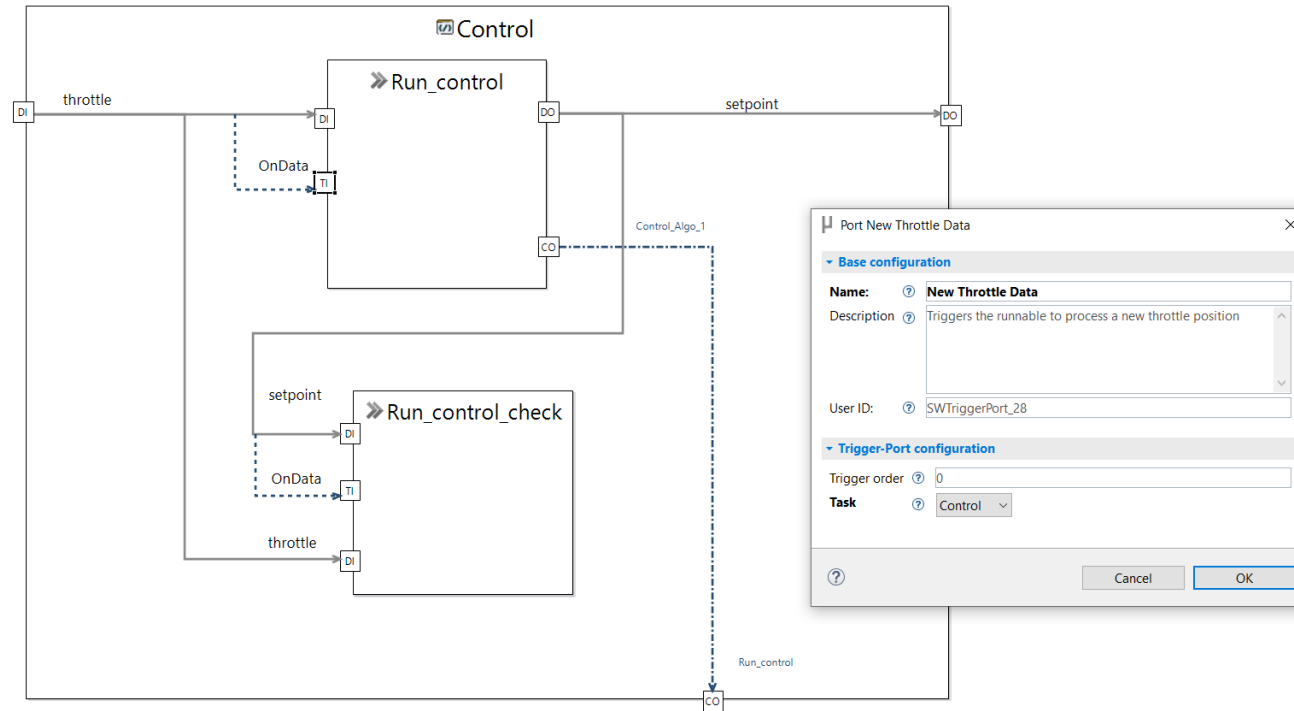
- μRTE System Model
 - 01 Software Model
 - Application
 - LED
 - Throttle Control
 - SWC ADC
 - SWC Control
 - SWC Motor
 - SWC Bootup
 - SWC Control Algos
 - SWC LED
 - Source-Folder Complex Device Drivers
 - Source-Folder Algorithms
 - RTE
 - RTOS
 - Linkage
 - DataTypes
 - Hardware Model
 - Logical Function Model
 - Requirement Model



Software modeling

Detailed view describes runnable execution

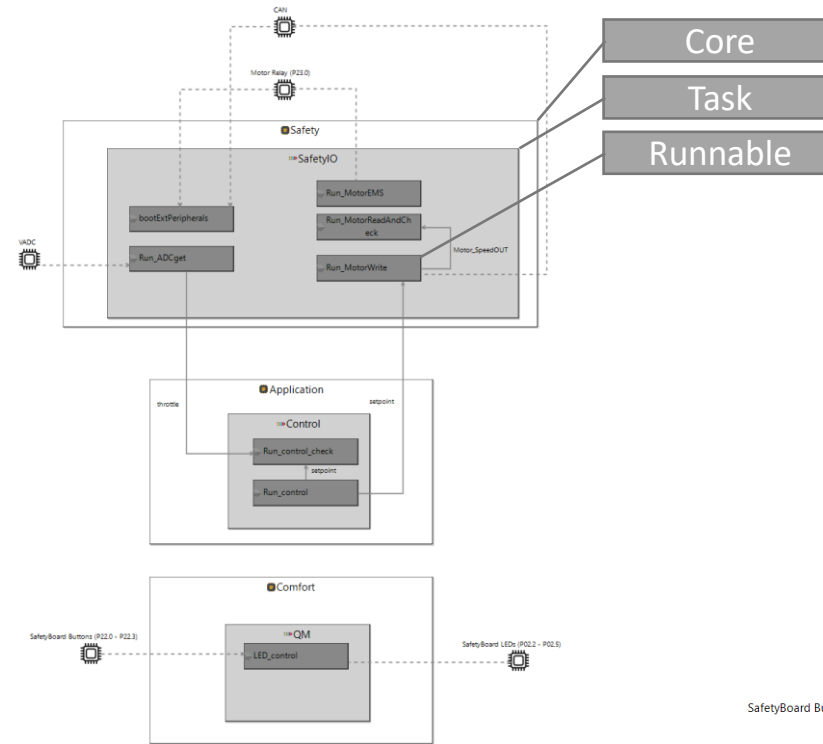
- μRTE System Model
 - 01 Software Model
 - Application
 - LED
 - Throttle Control
 - SWC ADC
 - SWC Control
 - Control
 - Runnable Run_control
 - Runnable Run_control_check
 - SWC Motor
 - SWC Bootup
 - SWC Control Algos
 - SWC LED
 - Source-Folder Complex Device Drivers
 - Source-Folder Algorithms
 - RTE
 - RTOS
 - Linkage
 - DataTypes
 - Hardware Model
 - Logical Function Model
 - Requirement Model



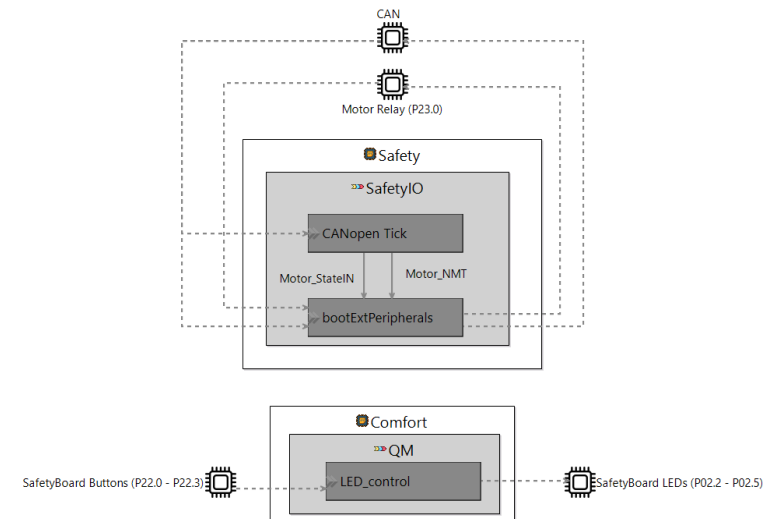
Software modeling

Core/Task centric views for the data-flow
Also for specific system-states
Shows which software is active and which data is accessed

- μRTE System Model
 - Software Model
 - Application
 - RTE
 - ActivationEngine
 - SignalLayer
 - SystemStates
 - taskCentric_detailed
 - taskCentric_grouped
 - PlantUML StateMachine
 - SystemState operational
 - SystemState error_detected
 - SystemState init
 - detailed task view for state init**
 - StateHandler
 - RTOS
 - Linkage
 - DataTypes
 - Hardware Model
 - Logical Function Model
 - Requirement Model



General Flow

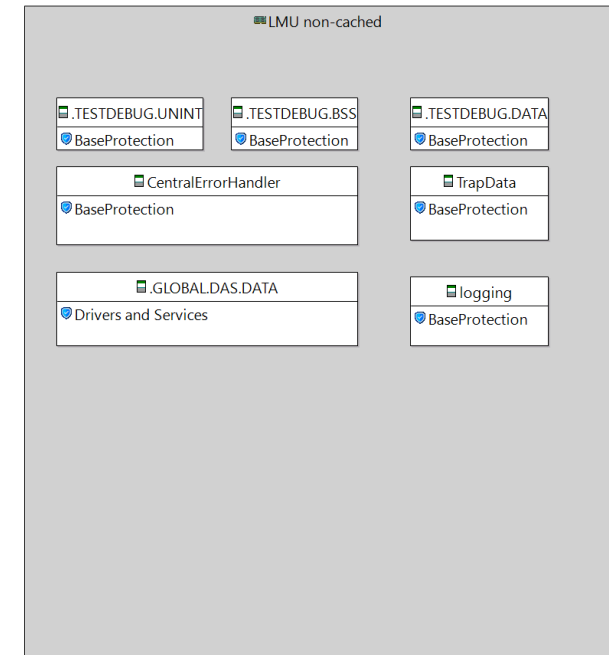
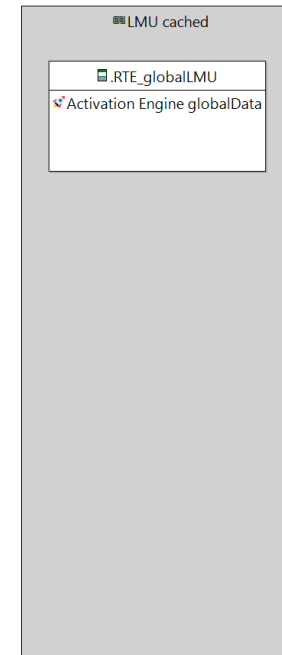
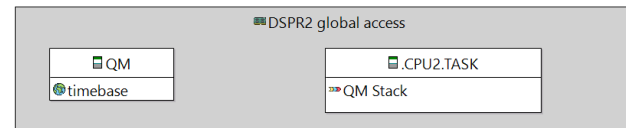
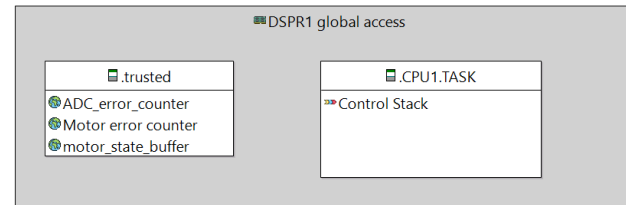
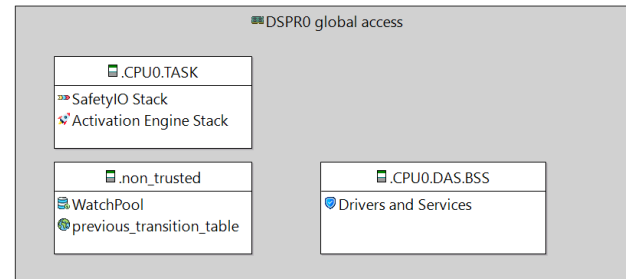


Init-State

Software modeling

Linkage overview:
memory modules with outputsections and references to the outputsections

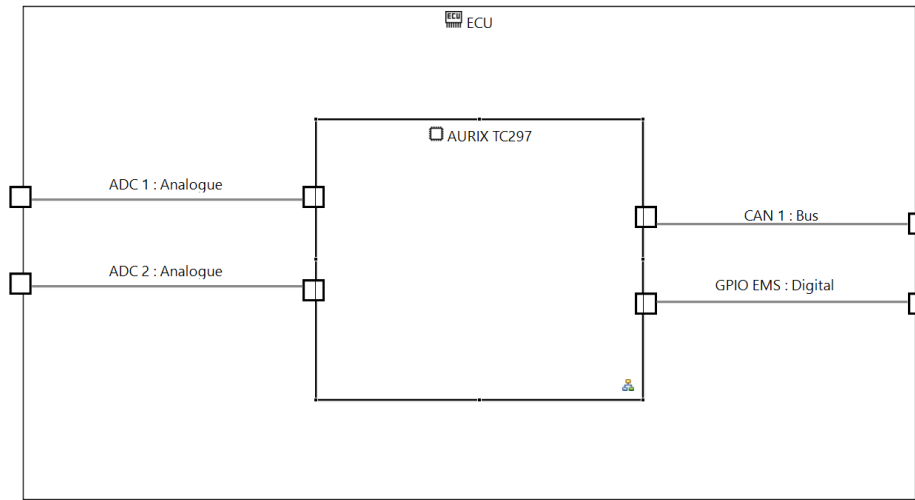
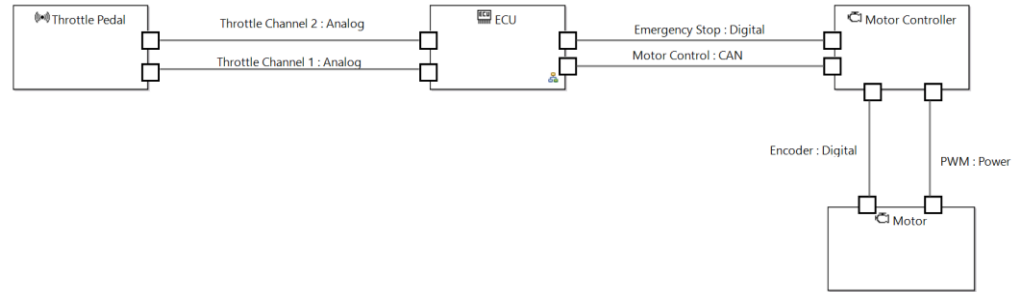
- μRTE System Model
 - 01 Software Model
 - Application
 - RTE
 - RTOS
 - Linkage
 - OutputSections
 - new OutputSections Overview with Memory
 - OutputSection .trusted
 - OutputSection .non_trusted
 - OutputSection .RTE_globalLMU
 - OutputSection CentralErrorHandler
 - OutputSection .CPU0.TASK
 - OutputSection .CPU1.TASK
 - OutputSection .CPU2.TASK
 - OutputSection .TESTDEBUG.UNINT
 - OutputSection .TESTDEBUG.BSS
 - OutputSection .TESTDEBUG.DATA
 - OutputSection .CPU0.DAS.BSS
 - OutputSection TrapData
 - OutputSection logging
 - OutputSection .GLOBAL.DAS.DATA
 - OutputSection QM
 - ProtectionSets
 - DataTypes
 - Hardware Model
 - Logical Function Model
 - Requirement Model



Hardware modeling

Modeling of functional and non-functional components
Hierarchical graphical representation

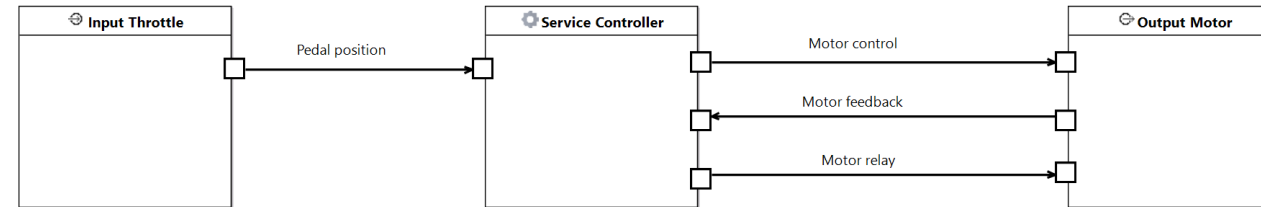
- μRTE System Model
 - 01 Software Model
 - Hardware Model
 - Hardware
 - Sensor Throttle Pedal
 - ECU ECU
 - HWC ECU
 - MCU AURIX TC297
 - Actor Motor Controller
 - Actor Motor
 - Logical Function Model
 - Requirement Model



Functional modeling

Abstract representation of the system functionality

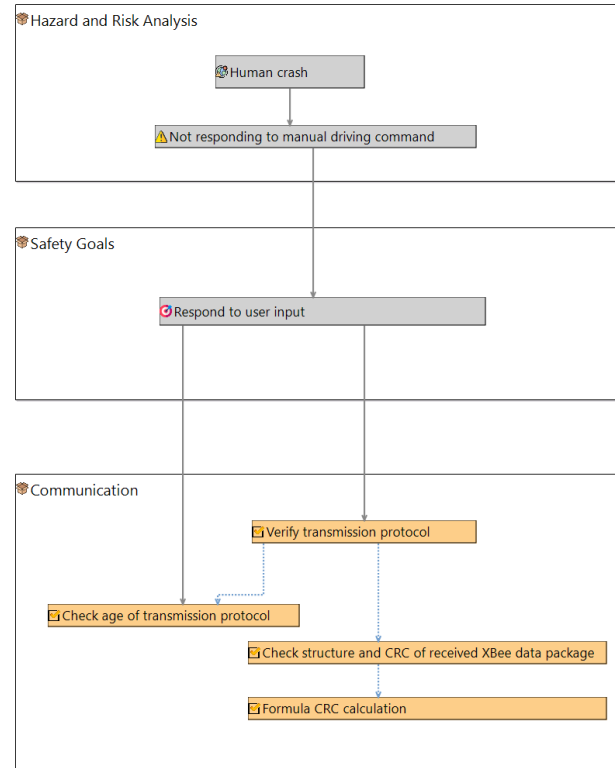
- μRTE System Model
 - 01 Software Model
 - Hardware Model
 - Logical Function Model
 - LED
 - Throttle Control
 - Logical Function Block Throttle
 - Logical Function Block Controller
 - Logical Function Block Motor
 - Logical Function Block Buttons
 - Logical Function Block LEDs
 - Technical Function Throttle control
 - Technical Function LEDs
 - Requirement Model



Requirements modeling

Derivation of requirements
Relationship (refinement/conflict) among requirements
Mapping to implementation units

- μRTE System Model
 - Software Model
 - Hardware Model
 - Logical Function Model
 - Requirement Model
 - Overview
 - Package Top Level Requirements
 - Package Hazard and Risk Analysis
 - Hazardous Event Not responding to manual driving command
 - Hazardous Event Not detecting obstacle
 - Hazardous Event Wrong actuator action
 - Hazardous Event Wrong targetspeed / controlspeed calculation
 - Hazard Scenario Human crash
 - Package Safety Goals
 - Safety Goal Respond to user input
 - Safety Goal Stop in front of obstacle
 - Safety Goal Supervise engine
 - Safety Goal Plausibility of targetspeed values
 - Package Communication
 - Safety Requirement Verify transmission protocol
 - refined by Check structure and CRC of received XBee data package
 - refined by Check age of transmission protocol
 - Safety Requirement Check age of transmission protocol
 - Requirement Dispatch received protocol



Modeling in general

Besides graphical editing any element in the model can be edited with a lightweight GUI
 The model is saved as XML for good manageability

The screenshot shows a software modeling tool interface. On the left, a tree view displays a 'Resource Set' containing a 'platform/resource/SafetyBoard_AURIX/SafetyBoard_AURIX.uRTE' folder. Under this folder, there is a 'μRTE System Model' which includes a 'Software Model' containing an 'Application' and an 'RTE'. The 'RTE' contains an 'ActivationEngine' with several 'CyclicEvent' elements, including 'CyclicEvent ADC', 'CyclicEvent MotorWrite', 'CyclicEvent MotorCheck', 'CyclicEvent InitClock', 'CyclicEvent Error Counter clock', 'CyclicEvent CANopen base', 'CyclicEvent CANopen SYNC', and 'CyclicEvent LED clock'. Other elements in the tree include 'SignalLayer', 'SystemStates', 'RTOS', 'Linkage', 'DataTypes', 'Hardware Model', 'Logical Function Model', and 'Requirement Model'. On the right, a 'Properties' table is displayed for the selected 'CyclicEvent ADC' element.

Property	Value
Description	first timer event in the 100ms cycle, used to collect new ADC data. Offset is to give the error counter mechanism time to operate
Name	ADC
User-ID	CyclicEvent_73
SIL achieved	QM
SIL justification	
SIL manual	derived/undefined
SIL manual reason	
Trigger	RunnableTrigger RunTPort_ADCget
Cycle time	100
Offset	1