AUTOSAR and Functional Safety

Simon Fürst, BMW Group
Safetronic 2011
8 Nov. 2011, Sheraton Arabellapark Hotel, Munich
Basic aspects of AUTOSAR architecture and methodology

Safety mechanisms supported by AUTOSAR

Technical safety concepts supported by AUTOSAR

Relationship to ISO 26262 and Conclusion
AUTOSAR and Functional Safety
AUTOSAR Vision

AUTOSAR aims to standardize the software architecture of ECUs.
AUTOSAR paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.

Hardware and software will be widely independent of each other.
Development can be de-coupled by horizontal layers. This reduces development time and costs.
The reuse of software increases at OEM as well as at suppliers. This enhances quality and efficiency.
Ports implement the interface according to the communication paradigm (here client-server based).

Ports are the interaction points of software components.

The communication is channeled via the RTE.

The communication layer in the basic software is encapsulated and not visible at the application layer.
AUTOSAR and Functional Safety
Software Architecture – AUTOSAR Defined Interfaces

Automotive Open System Architecture (AUTOSAR):
- Standardized, openly disclosed interfaces
- HW independent SW layer
- Transferability of functions
- Redundancy activation

AUTOSAR RTE:
by specifying interfaces and their communication mechanisms, the applications are decoupled from the underlying HW and Basic SW by the RTE. This enables the realization of re-usable application software components.
The Basic Software Layers are further divided into functional groups. Each functional group consist of multiple basic software modules.
The AUTOSAR Meta Model is the backbone of the AUTOSAR architecture definition. It contains complete specification, how to model AUTOSAR systems.

M0: Realized System in the car (Implements a real system)

M1: Model of the system (Defines a real system)

M2: Model of the model (Meta Model) (Defines AUTOSAR Modeling Elements)

M3: Model of the Meta Model (Meta-Meta Model) (Defines UML Modeling Elements)
AUTOSAR and Functional Safety
Overview

Basic aspects of AUTOSAR architecture and methodology

Safety mechanisms supported by AUTOSAR

Technical safety concepts supported by AUTOSAR

Relationship to ISO 26262 and Conclusion
AUTOSAR and Functional Safety

Approach of AUTOSAR with regard to Functional Safety.

ISO WD 26262
Requirements from WPs & WGs
Requirements from Applications
Requirements from Safety Concepts

Consolidated Safety Requirements

Process Safety Requirements
AUTOSAR Safety Guidelines

Technical Safety Requirements
Interface Class 1
Interface Class 3

Methodology Safety Requirements
Tools
Generation

List of safety requirements allocated to methodology

List of safety requirements allocated to BSW & RTE

Update of existing documents of WPs
Requirements on tools and generation process

Consolidated Safety Requirements

Development Process
BSW & RTE
Tools

BSW & RTE Requirements
SRS
SWS

Tools and Generation Process
Tools
Generation

List of requirements on development processes
Requirements on how to develop AUTOSAR SW and Tools
Built-in self test mechanisms for detecting hardware faults (testing and monitoring)

Run-time mechanisms for detecting software faults during the execution of software
  Program flow monitoring

Run-time mechanisms for preventing fault interference
  Memory partitioning for SW-Cs
  Time partitioning for applications

Run-time mechanisms for protecting the communication
  End-to-end (E2E) communication protection for SW-Cs

Run-time mechanisms for error handling
AUTOSAR and Functional Safety
Safety mechanisms for detecting errors.

Memory:
- RAM Test
- Flash Test
- Support for ECC memory

Core:
- Core Test

Watch Dog
Logical and temporal program flow monitoring
AUTOSAR and Functional Safety
Run-time mechanisms for error handling

Detected errors in the basic software:
- Are reported through DEM to SW-Cs. SW-Cs then executes application-specific actions
- Are reported to FIM, which permits to disable some functions of SW-C

Detected hardware errors:
- Arithmetic exceptions (e.g. division by 0): handled by OS callouts (small error handling routines in the context of basic software). Typical reaction – ECU reset
- HW errors detected by HW testing: handled by callouts. Typical reaction – ECU reset
- Errors detected my MMU/MPU (memory and time partitioning). It will shut down or restart the faulty SW-C partition
Enables create protection boundaries around groups of SW-Cs
This is realized by user-mode/non-trusted memory partitions (for groups of SW-Cs)
This protects from interference:
(1) basic software and
(2) SW-Cs in other partitions
Basic software is not partitioned. It runs with in CPU supervisor mode with full access to memory, CPU and all other hardware resources
E2E protection detects faults in data caused by both hardware and in software.

Typical sources of interferences, causing errors detected by E2E protection:

SW-related sources:
- S1. Error in mostly generated RTE.
- S2. Error in partially generated and partially hand-coded COM
- S3. Error in network stack
- S4. Error in generated IOC or OS

HW-related sources:
- H1. Microcontroller error during core/partition switch
- H2. Failure of HW network
- H3. Microcontroller failure during context switch (partition) or on the communication between cores
AUTOSAR and Functional Safety

End-to-End communication protection (2/4)

Application is almost un-impacted by the introduction of end-to-end protection wrapper

End-to-End protection wrapper protects/checks the communication on behalf of application, i.e. SW-Cs

End-to-End Protection wrapper encapsulates the data protection and also invokes RTE
Protection of data exchanged over communication channels like FlexRay and CAN

Failure modes addressed as defined by ISO DIS 26262 for communication (repetition, deletion, insertion, incorrect sequence, corruption, timing faults, addressing faults, inconsistency, masquerading)

Three different protection mechanisms for data are used

- CRC, counter, Data ID, timeout detection
- Data ID included in to calculated CRC, but not sent

\[
\text{CRC := CRC8 over (1) Data Id, (2) all serialized signal (including empty areas, excluding CRC byte itself)}
\]
Fully AUTOSAR compliant design with major impact on ASIL inheritance

Example: overall flow at sender
AUTOSAR and Functional Safety
Overview

Basic aspects of AUTOSAR architecture and methodology

Safety mechanisms supported by AUTOSAR

Technical safety concepts supported by AUTOSAR

Relationship to ISO 26262 and Conclusion
AUTOSAR and Functional Safety
Technical safety concepts supported by AUTOSAR

Implementation of typical safety concepts in the automotive domain
  Intelligent HW watchdog (ASIC) / 3-level safety concept
  Monitored channel (2 µCs, the second is a simple µC monitoring the first µC)
  Dual channel (2 AUTOSAR µCs)

Application redundancy (on the same or different µCs)
Basic Software redundancy inside one ECU
Assuming integrity of HW/ECU and AUTOSAR basic software implementation, software redundancy with ASIL decomposition can be used within the same ECU.
Distribution of SW channels across ECUs is also possible.
Redundancy inside AUTOSAR e.g. double input/output data paths through
- Redundant IO hardware abstraction and IO drivers
- Redundant and diverse (e.g. ADC + DIO, internal ADC + external ADC)

Redundancy through integration of complex drivers running on the same µC offering a redundant data path.
Basic aspects of AUTOSAR architecture and methodology

Safety mechanisms supported by AUTOSAR

Technical safety concepts supported by AUTOSAR

Relationship to ISO 26262 and Conclusion
Essential concepts of ISO 26262 have been developed in sync with AUTOSAR:

- Software configuration: Part 6, Chapter 7 and Annex C
- Freedom of interference by partitioning: Part 6, Chapter 7 and Annex D
- Safety Element out of Context (SEooC): Part 10, Chapter 9
- Qualification of software tools: Part 8, Chapter 10
Due to rules on ASIL inheritance defined in ISO 26262 the AUTOSAR basic software and RTE inherits safety relevance.

Either implement complete AUTOSAR basic software according to max. ASIL of application software or demonstrate freedom of inference in basic software by appropriate mechanisms.

Implementers have to tailor ISO 26262 according to their activities in the safety-lifecycle.

For all implemented safety mechanisms a safety manual is needed containing

The fault model according to which the safety mechanism was developed

The constraints that must be fulfilled when applying a safety mechanism.
AUTOSAR and Functional Safety

Conclusion

AUTOSAR systematically derived safety mechanisms supported in release 4.0

AUTOSAR provides support for dedicated safety mechanisms with generic fault models

AUTOSAR supports typical technical safety concepts

During system and software design the safety manual is considered to appropriately use the safety mechanisms of an AUTOSAR implementation.

**AUTOSAR provides essential support for building of safety related systems**