Functional Safety from the Perspective of an AUTOSAR BSW Supplier
Overview

- Introduction and Requirements
- Safety Concept
- Experiences and Optimizations
- Conclusion
1. **Identify driving situations**
   - e.g. “driving on the highway”

2. **Identify potential hazards**
   - e.g. “sudden brake impulse on one wheel
     ➔ vehicle gets off the road”

3. **Assess risk and determine safety integrity level**
   - very high risk: ASIL-D
   - ... 
   - low risk: ASIL-A
   - no risk: QM

4. **Define safety goals and derive safety requirements**
   ... in order to minimize the risk to an acceptably low level
   ➔ safety requirements inherit the ASIL of the hazard
System Architecture

Technical Safety Concept

- Define technical system architecture
- Assign safety requirements to architecture elements
- Generate system extract for ECUs
- Goal: Implement functionality, meet safety requirements
Software for Safety ECUs

Mixed ASIL ECUs
Development of Mixed ASIL Software acc. ISO 26262

1. ASIL Lift-up
2. Qualification
3. Coexistence
1. ASIL Lift-up

Development according highest ASIL

- high development effort
- larger footprint
Development of Mixed ASIL Software acc. ISO 26262

1. ASIL Lift-up

2. Qualification

Qualification of existing software

- Evidence needed that software ...
  ... complies with its requirements and
  ... is suitable for intended use in a project
- Typically up to ASIL-B

- High project specific effort
- has to be repeated on any change of
  - QM module or
  - ASIL environment or
  - configuration
Components of different ASIL exist in one ECU

- Assign software safety requirements and ASIL to software components (based on Technical Safety Concept)
- Ensure *Freedom from Interference* between software components with different ASIL
  - evidence needed that safety requirements are not violated
Development of Mixed ASIL Software - Coexistence

QM  ASIL

Threat: Propagation of failures

a. across defined interfaces
b. across undefined interfaces

QM
Overview

- Introduction and Requirements
- Safety Concept
- Experiences and Optimizations
- Conclusion
Basic Software Modules to realize Freedom from Interference

- ASIL

Freedom from Interference:

- regarding Exchange of Information: SafeCom*
- regarding Timing & Execution: SafeWatchdog*
- regarding Memory: SafeContext / SilentBSW

*developed by/with TITech
FfI Communication: **SafeCom** (E2E Protection)

- Ensures FfI Communication (ASIL-D)
- Consists of E2E Protection Wrapper and E2E Lib acc. to AUTOSAR 4.x
- Developed by TTTech
- Certified by TÜV
## Basic Software Modules to realize Freedom from Interference

<table>
<thead>
<tr>
<th>Freedom from Interference</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>regarding Exchange of Information</td>
<td>SafeCom*</td>
</tr>
<tr>
<td>regarding Timing &amp; Execution</td>
<td>SafeWatchdog*</td>
</tr>
<tr>
<td>regarding Memory</td>
<td>SafeContext / SilentBSW</td>
</tr>
</tbody>
</table>

*developed by/with TTTech
Each SWC has its own state and program flow

Checkpoints are used to signal the internal state and program flow to the Watchdog Manager

Examples

- State initializing
- State ready, waiting for start signal
- State normal running
- Correct program flow
Safe-Watchdog Manager (S-WdgM) supervises all individual states.

SWCs that fail to reach their checkpoints in time and in correct order are detected.

Individual states are combined to a system health state.

Appropriate actions can be triggered.
### Basic Software Modules to realize Freedom from Interference

<table>
<thead>
<tr>
<th>Freedom from Interference</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exchange of Information</td>
<td>SafeCom*</td>
</tr>
<tr>
<td>Timing &amp; Execution</td>
<td>SafeWatchdog*</td>
</tr>
<tr>
<td>Memory</td>
<td>SafeContext / SilentBSW</td>
</tr>
</tbody>
</table>

*developed by/with **TTTech**

© 2013 All rights reserved. Any distribution or copying is subject to prior written approval by Vector.
FfI Memory: **SafeContext**

**SafeContext**

- MPU avoids invalid memory access, but tasks must run with correct MPU settings
- SafeContext safely provides the context for each safety related task:
  - register settings
  - stack pointer and program counter
  - MPU settings

› is realized as part of AUTOSAR-OS SC3/4
› certified by TÜV
Overview

- Introduction and Requirements
- Safety Concept
- Experiences and Optimizations
- Conclusion
Freedom from Interference - Memory

**SafeContext**
- uses Memory Protection Unit (MPU)
- can be used up to ASIL-D
- Not optimal for intensive interaction between software elements of different MPU regions (runtime overhead)

**Goal:**
- Basic Software may coexist with ASIL Software
- but: avoid development of Basic Software acc. ISO 26262

**Solution:** **SilentBSW**
Silent Basic Software

**Approach:**

- Develop Basic Software as Safety Element out of Context (SEooC) with the assumed safety requirement
  - "Do not corrupt external memory"

1. **Perform general Safety Analysis**
   - FTA and FMEA
   - Approx. 50 critical patterns were identified that could violate the assumed safety requirement:
     - using generated pointers or indices
     - pointer and index arithmetic
     - unintended function calls
     - ...
   - Measures were derived for each pattern

2. **Apply measures in project context**

FTA – Fault Tree Analysis
FMEA – Failure Mode and Effect Analysis
Measures to realize Silent Basic Software

Critical Patterns → Tool based Analysis

Measures
- Test Cases
- Code Review
- Runtime Checks
- Checking Rules

Tool based Code Check → TCL2 ASIL-D

static BSW code
- .c
- .h

developed acc. SPICE process

Verdict

TCL2 ASIL-D → Tool based Analysis

Vector

Customer

generated BSW code
- .c
- .h

based on user configuration
FfI Memory - Summary

- **SafeContext**
  - avoids memory corruption by separation
  - advantage - realizes all separations needed (also within application software)

- **SilentBSW**
  - avoids memory corruption caused by Basic Software
  - by elimination of unsafe code patterns
  - advantages: - no MPU needed
              - less complex system
              - minimal runtime overhead
Basic Software Modules for Functional Safety

ASIL

2. Safe Communication

Freedom from Interference

regarding Exchange of Information: SafeCom*

regarding Timing & Execution: SafeWatchdog*

regarding Memory: SafeContext / SilentBSW

QM

*developed by/with TTTech
Safe Communication: **SafeRTE**

- SafeRTE realizes correct communication within an ASIL partition
- SafeRTE can replace E2E protection for Intra ECU communication in combination with MPU and safe hardware
Safe communication between different OS applications
- Sender-Receiver without context switch
- Client/Server-Call with context switch

RTE’s implementation supports memory partitioning

Implementation according to AUTOSAR
SafeRTE - Implementation

- **Safety optimized implementation**
  - Complexity of generated code decreased
  - Dependency to QM BSW modules removed
  - Split of generated code in separate files per OS application
  - reduces complexity and review effort

- **Provision of Safety Guide**
  - Design rules for usage in ASIL applications
  - List of recommended functions for ASIL applications
  - Review rules for generated code
    - Correct communication between SWCs
    - Freedom from Interference regarding memory
SafeRTE – Improvements under Development

- Decrease project specific effort:
  - Checks of generated code by qualified verification tools
    - RTE Verifier

- Increase number of recommended functionalities for safety applications:
  - For safe communication between SWCs
    - e.g. implicit Sender/Receiver communication
  - Safe mode management
  - ...

© 2013 All rights reserved. Any distribution or copying is subject to prior written approval by Vector.
Overview

- Introduction and Requirements
- Safety Concept
- Experiences and Optimizations
- Conclusion
MICROSAR Safe - Summary

► “Certified”
  ► Our solution is accepted by functional safety engineers of many OEMs.
    > This minimizes effort for design of the functional safety concept
  ► Design and development is certified by TÜV
    > This minimizes effort for your qualification and certification

► Standardized
  ► Our solution uses functionalities already standardized by AUTOSAR.
    > This minimizes effort for introduction and for re-use in later projects.

► Available
  ► Our solution is available for many hardware platforms

→ **Vector and TTTech reduces time-to-market and development costs**
Thank you for your attention.

For detailed information about Vector and our portfolio please go to: www.vector.com

Günther Heling
Joachim Kalmbach
Steffen Keul
Jochen Rein

Vector Informatik GmbH
Ingersheimer Str. 24
D-70499 Stuttgart