Embedded Software for Functional Safety

MICROSAR Safe
Functional Safety

ISO 26262 has been renewed in 2018 addressing safety-related electronic automotive systems.

- To reduce liability risks state of the art development methods as described in the ISO 26262 should be applied for such systems.
Introduction

ISO 26262-compliant Development

- Item Definition
- Hazard Analysis and Risk Assessment
- Derivation of Safety Goals
- Functional Safety Concept
- Technical Safety Concept
- Basic SW Development
- Appl. SW Development
- Hardware Development
- Safety Validation
- Item Integration and Testing
Introduction

Achieving Safety Together!

Vehicle Project
- Hazard Analysis and Risk Assessment
- Derivation of Safety Goals
- Safety Concept
  - Software Development acc. ISO 26262
- Software Safety Requirements
- ECU SW Integration
- Safety Case

SEooC (Vector)
- Specific Use-case is Unknown!
- Assumptions on Technical Safety Requirements
  - Software Development acc. ISO 26262
- Software Safety Requirements
- Safety Manual
- Safety Case

validate!
consider!
reference!
Evolution of Safety Concepts

Introduction

Enhancing driver actions

Highly Automated Driving

Taking over driver decision

fail-safe

- Redundancy (e.g. redundant data)
- Partitioning
- High Performance Integrity

fail-operational

- Redundancy (redundant functions)
- ...

Evolution of Safety Concepts

Redundancy (e.g. redundant data)

Partitioning

High Performance Integrity
In many cases we see mixed-ASIL Systems

Adequate Safety Concepts:
- Partitioning
- High Performance Integrity

SW Safety Requirements

Software Elements

ECU Software

ECU 1

ECU 2

ECU 3

ECU 4

QM

QM

QM

QM

ASIL

ASIL

ASIL

QM

QM

QM

QM

ASIL

ASIL

ASIL

QM

QM

QM

QM

ASIL

ASIL

ASIL
Introduction

Safety Concepts and Contribution of MICROSAR Safe

**Partitioning**

- **ASIL**
  - SafeOS
  - SafeWDG
  - SafeE2E
  - SafeRTE

- **QM**
  - SWC

- **Hardware**
  - BSW
  - MCAL

**Rationale**

- **MSR Safe Components**
  - SafeOS
  - SafeWDG
  - SafeE2E
  - SafeRTE

- **ISO 26262** requires that software with different ASIL in the same element do not interfere with respect to memory, timing and communication aspects.

**High Performance Integrity**

- **ASIL**
  - SafeOS
  - SafeWDG
  - SafeE2E
  - SafeRTE
  - SafeBSW

- **QM**
  - SWC

- **Hardware**
  - MCAL

**Rationale**

- **MSR Safe Components**
  - SafeOS
  - SafeWDG
  - SafeE2E
  - SafeRTE
  - SafeBSW

- **ISO 26262** requires to implement software components of different ASIL, or safety-related and non-safety-related software components, in accordance with the highest ASIL.
Introduction

Safety Building Blocks of MICROSAR Safe

MICROSAR SafeOS
- Supports memory partitioning using an MPU
- Provides safe context switch for each safety related task:
  - register settings
  - stack pointer and program counter
  - MPU settings
- Available for single- and multi-core
- OS Applications can be restarted individually

MICROSAR SafeRTE
- Ensures safe communication within the ECU
- Supports safe communication across partition boundaries to exchange information between ASIL and QM applications

MICROSAR SafeWDG
- Detects timing and execution order faults
- Provides deadline, alive and logic monitoring
- Capable of using internal or external watchdogs as well as system basis chips (SBCs)

MICROSAR SafeE2E
- Ensures safe communication between ECUs
- Available as E2E Protection Wrapper and E2E Transformer
- All AUTOSAR profiles supported

MICROSAR SafeBSW
- Increased performance
  - complete BSW as ASIL software
  - reduced partition switches
- Additional safety requirements, e.g.:
  - Correct initialization using an ASIL EcuM
  - Safe write/read of non-volatile data
Introduction

- MICROSAF SafeOS
- MICROSAF SafeWDG
- MICROSAF SafeE2E
- MICROSAF SafeRTE
- MICROSAF SafeBSW

Process and Services

Summary
MICROSAR SafeOS

- Completely developed according to ASIL D
- Provides features to argue freedom from interference for application and BSW regarding
  - **Memory**: Supports memory separation using the MPU
  - **Timing**: Detection of time-budget violations
- Provides safe context switching for each safety related task:
  - register settings
  - stack pointer and program counter
  - MPU settings
- Available for single- and multi-core
- AUTOSAR 4.3 Safety Features:
  - Safe access to peripheral registers even from user mode
  - Interrupt source control API

Features **additional to AUTOSAR OS SC3/SC4**:
- Non-trusted function calls
- Optimized S/R communication across different contexts
### overview safeos

<table>
<thead>
<tr>
<th>Which faults are possible?</th>
<th>How do we address them?</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Memory violations</td>
<td>▶ Memory Partitioning</td>
</tr>
<tr>
<td>▶ Wild Pointers</td>
<td>▶ OS Applications can be defined that are protected by the MPU</td>
</tr>
<tr>
<td>▶ Stack overflow</td>
<td>▶ Protected peripheral access</td>
</tr>
<tr>
<td>▶ Writing outside of intended memory location of variables</td>
<td>▶ Stack Protection</td>
</tr>
<tr>
<td>▶ Timing violations</td>
<td>▶ Stack is separately protected by the MPU</td>
</tr>
<tr>
<td>▶ Endless loops</td>
<td>▶ Indicator values detect stack overflows also for systems without MPU</td>
</tr>
<tr>
<td>▶ Too frequent interrupts</td>
<td>▶ Timing protection</td>
</tr>
<tr>
<td>▶ Longer calculation times due to unexpected input</td>
<td>▶ Time budgets are monitored</td>
</tr>
<tr>
<td></td>
<td>▶ Termination of applications</td>
</tr>
</tbody>
</table>
The memory of individual OS Applications can be protected against writing by other OS Applications.

This requires dedicated hardware support (MMU/MPU).

Reading between OS Applications is typically not restricted.

OS is re-programming the MPU if the number of available regions is not sufficient.

**Partitioning**
Timing Protection

- Execution budgets are assigned to tasks and monitored
- If the budget is exceeded a protection hook is called
- Similarly, inter-arrival-times and resource locking times are monitored

Timing Protection vs. Watchdog
- Timing protection does not detect deadline violation!
- Detects causes of deadline violations earlier than watchdog
- Timing protection is limited to tasks / ISR 2 (ISRs of type 1 bypass the timing protection)
Access to Peripheral Registers

- Access to restricted registers might only be possible in supervisor mode
- For QM software supervisor mode is typically restricted
- The OS provides a register access API that is used by dedicated BSW modules to be able to run in user mode
  - Must be confirmed by the user
  - API can also be used by CDDs
Register Access API

- Example

```c
osWritePeripheral8(uint16 area, uint32 address, uint8 value)
{
    // Validate access request
    // Change to supervisor mode
    // Perform access
    // Change to previous mode
}
```

- API provided functions for read, modify and write access
- The functions are defined for 8 Bit, 16 Bit and 32 Bit registers
Agenda

Introduction
MICROSAR SafeOS

► MICROSAR SafeWDG
MICROSAR SafeE2E
MICROSAR SafeRTE
MICROSAR SafeBSW

Process and Services
Summary
A watchdog can detect timing violations in the Application and BSW

- Also provides Program Flow Monitoring up to ASIL D
- Is supervised by independent HW-Watchdog
- On detected violations in a supervised entity the ECU can be reset
- Acknowledgements of a checkpoint are performed without context switch
### Which faults are possible?

- Execution of code without request
- Code not executed although requested
- Execution of code started too early or too late
- The execution time of an code is longer or shorter than expected
- The program flow of an code differs from the expected behavior

### How do we address them?

- **Deadline Monitoring**
  - Applicable for aperiodic entities
  - Time between two checkpoints is compared to min/max values
- **Alive Monitoring**
  - Applicable for periodic entities
  - Number of checkpoints in interval is monitored
- **Logic Monitoring**
  - Detect wrong execution order
  - Validate checkpoint activation sequence against preconfigured execution graphs

- It is assumed for all MICROSAR Safe components, that timing faults are handled using a watchdog.
Periodic Supervised Entities have constraints on the number of times they are executed within a given time span.

The Watchdog Manager checks periodically if the Checkpoints of a Supervised Entity have been reached within the given limits:

- not too frequently
- not too rarely.
Aperiodic Supervised Entities have individual constraints on the timing between two Checkpoints.

The Watchdog Manager checks
- the timing of transitions between two Checkpoints of a Supervised Entity
- if the time between two steps are within the configured minimum and maximum
Logic Monitoring

- The execution order of the code is monitored
- The Watchdog-Manager traces the checkpoints using a checkpoint graph, determining the validity of an execution sequence.
- Checkpoints are declared as start and end checkpoints

- Reaction on monitoring results
  - For each violation
  - After n violations
Global Watchdog State

- Watchdog Manager combines all Supervised Entity states to a system state
- Depending on the system state the Watchdog Manager triggers the Watchdog Driver
System Basis Chips in the AUTOSAR Stack

WDGM

WDGIF

WDG Driver (SBC)

SBC Driver

SPI Driver

CAN Transceiver Driver (SBC)

LIN Transceiver Driver (SBC)

Transceiver drivers can be put in different partition than the watchdog stack.

SBC Driver is specific for external hardware unit

SPI Driver is specific for microcontroller
MICROSAR SafeWDG

Configurations

Watchdog

- Internal Watchdog
  - WDGM
  - WDGIF
  - WDGDRV

- External Watchdog
  - WDGM
  - WDGIF
  - EXT_WDGDRV

- System Basis Chip
  - WDGM
  - WDGIF
  - SBC
  - SPI

1 Digital I/O (DIO) and Serial Peripheral Interface (SPI) drivers can be developed by Vector or used from another source (e.g. MCAL) if available with the required ASIL.
Agenda

Introduction
MICROSAR SafeOS
MICROSAR SafeWDG

► MICROSA R SafeE2E
MICROSAR SafeRTE
MICROSAR SafeBSW

Process and Services
Summary
Communication from one SWC to another SWC on a different ECU over an unsafe channel

- This channel comprises:
  - RTE
  - Com-Stack
  - Bus Controller
  - Wiring

- E2E is able to **detect** if a fault occurred during the transmission

- The application needs to use this information to react on the faults
### Which faults are possible?

- Failure of communication peer (even in lower software layers)
- Message masquerading
- Message corruption
- Unintended message repetition
- Insertion of messages
- Re-sequencing
- Message loss
- Message delay

### How do we address them?

- **CRC over data, data ID and sequence counter**
  - Allows to detect corruption and masquerading of the signal
- **Sequence counter**
  - Allows to detect faults in the order of messages
  - Allows to detect repeated/inserted messages
- **Timer** on receiver side
  - React on lost messages
  - React on delayed messages

See also: ISO 26262 Part 6 D.2.4 Exchange of information
MICROSAR SafeE2E

E2E Protection Wrapper

```
Rte_Write_<A>_<B>

E2E LIB

CRC LIB

data  cnt.  crc

E2EPW_Write( data )

Rte_Read_<A>_<B>

E2E LIB

CRC LIB

data  status

E2EPW_Read( data  cnt.  crc )

Rte_Write_<A>_<B>

BZW

RTE

BSW

Bus
```
### E2E Profile Overview

<table>
<thead>
<tr>
<th>Profile</th>
<th>CRC</th>
<th>CRC Length</th>
<th>Counter</th>
<th>Data ID</th>
<th>Explicit&lt;sup&gt;1&lt;/sup&gt;</th>
<th>Dynamic Data ID&lt;sup&gt;2&lt;/sup&gt;</th>
<th>Msg. Length&lt;sup&gt;3&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A&lt;sup&gt;4&lt;/sup&gt;</td>
<td>0x1D</td>
<td>8 Bit</td>
<td>4 Bit</td>
<td>16 Bit</td>
<td>0</td>
<td>No</td>
<td>32 Byte</td>
</tr>
<tr>
<td>1 B&lt;sup&gt;4&lt;/sup&gt;</td>
<td>0x1D</td>
<td>8 Bit</td>
<td>4 Bit</td>
<td>16 Bit</td>
<td>0</td>
<td>No</td>
<td>32 Byte</td>
</tr>
<tr>
<td>1 C</td>
<td>0x1D</td>
<td>8 Bit</td>
<td>4 Bit</td>
<td>16 Bit</td>
<td>4</td>
<td>No</td>
<td>32 Byte</td>
</tr>
<tr>
<td>2</td>
<td>0x2F</td>
<td>8 Bit</td>
<td>4 Bit</td>
<td>8 Bit</td>
<td>0</td>
<td>Yes</td>
<td>32 Byte</td>
</tr>
<tr>
<td>4</td>
<td>0x1F4ACFB13</td>
<td>32 Bit</td>
<td>16 Bit</td>
<td>32 Bit</td>
<td>32</td>
<td>No</td>
<td>4096 Byte</td>
</tr>
<tr>
<td>5</td>
<td>0x1021</td>
<td>16 Bit</td>
<td>8 Bit</td>
<td>16 Bit</td>
<td>0</td>
<td>No</td>
<td>4096 Byte</td>
</tr>
<tr>
<td>6</td>
<td>0x1021</td>
<td>16 Bit</td>
<td>8 Bit</td>
<td>16 Bit</td>
<td>0</td>
<td>No</td>
<td>4096 Byte</td>
</tr>
<tr>
<td>7</td>
<td>0x42F0E1EBA9EA3693</td>
<td>64 Bit</td>
<td>32 Bit</td>
<td>32 Bit</td>
<td>32</td>
<td>No</td>
<td>4 MByte</td>
</tr>
</tbody>
</table>

<sup>1</sup> How many bits of the data ID are explicitly transmitted

<sup>2</sup> Different data IDs for different counter values exist

<sup>3</sup> Maximum possible message size

<sup>4</sup> Difference between 1A and 1B is the inclusion of different parts of the data ID in the CRC
**Configurations**

**E2E Protection**

**Protection Wrapper Solution**
- E2EPW
- E2ELIB
- CRCLIB

**Transformer Solution**
- E2EXf
- COMXF and/or SOMEIPXF
- E2ELIB
- CRCLIB

Protection Wrapper only supports Profiles 1, 2 and JLR
Agenda

Introduction
MICROSAR SafeOS
MICROSAR SafeWDG
MICROSAR SafeE2E

► MICROSA SafeRTE
MICROSAR SafeBSW
Process and Services
Summary
The RTE can be used to communicate between application software components.

The RTE can provide communication between different memory partitions and connects QM and ASIL software.

The RTE is usually required as ASIL if it is used within the same partition as ASIL application Software Components.

Using the RTE in safety relevant ECUs:
- The RTE is completely generated using DaVinci Configurator PRO and a corresponding generator.
- Thus, ISO 26262 Part 8 Clause 11 (Confidence in the use of software tools) applies.
Derivation of Tool Confidence Level (TCL)

Tool classification and qualification are usually performed by the user of the tool.
## RTE vs. SafeRTE

### RTE

- **Classification of RTE:**
  - TI2: “a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed”
  - TD2: “there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected”

### SafeRTE

- **Classification of RTE:**
  - TI2: “a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed”
  - TD1: “high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected”

---

### Classification of DaVinci Configurator PRO as TCL2 necessary

**Manual qualification of DaVinci Configurator PRO or the generated software by the user is necessary!**

---

### Vector provides argumentation to classify DaVinci Configurator PRO as TCL1.

**No qualification for TCL 1 tools needed!**
Arguments for TD1 in Detail

ARG1: RTE Generator is developed with additional effort.
ARG1a: Output of RTE Generator is tested according to ISO 26262:6-9.

ARG2: Output of RTE Generator is analyzed by RTE Analyzer.

ARG3: Output of RTE Generator is integrated and verified according to ISO 26262:6-10/11 by customer based on configuration feedback provided by RTE Analyzer.
ISO 26262-compliant Software Development with SafeRTE

Verification is performed on several levels to ensure that ECU performs as specified.

- SW Safety Requirements
- SW Architectural Design
- System Description/ECU Extract
- DaVinci Configurator PRO/RTE Generator
- SW Unit Design and Implementation
- SW unit testing
- Generated RTE code
- RTE Analysis
- Verification of software safety requirements
- Software integration and testing
- Requires complete target SW, configuration and target ECU
- Requires complete target SW, configuration and (target) ECU
- Software unit testing for RTE already performed by Vector
Fault Detection and Prevention

Potential faults in generated RTE

- Development process of RTE Generator
  *Vector*
- Unit test of generated RTE
  *Vector*
- Self-test and validation before generation
  *Tool*
- Usage of non-complex subset of RTE features
  *User of RTE*
- RTE Analyzer
  *Tool*
- Integration testing based on output of RTE Analyzer
  *User of RTE*

Generated RTE is considered sufficiently fault-free for ASIL D here
## Features of the RTE Analyzer

### General
- Compilation check for RTE code
- Detection of recursive call sequences
- Analysis report generation

### Concurrent Access
- Detection of RTE variables that are accessed from concurrent execution contexts without protection
- Detection of concurrent calls to non-reentrant APIs within the RTE
- Detection of variables that are accessed from multiple cores and that are not mapped to non-cacheable memory sections

### Out-of-bounds Access
- Detection of out-of-bounds write accesses within RTE APIs
- Detection of non type-safe interfaces to the BSW and SWCs where a call with a wrong parameter might cause out of bounds writes by the RTE or a called runnable/BSW API

### RTE API Usage
- Detection of interrupt lock API sequence mismatches within RTE APIs
- Detection of unreachable RTE APIs and runnables
- Detection of RTE APIs for which a call from a wrong context might cause data consistency problems
Agenda

Introduction
MICROSAR SafeOS
MICROSAR SafeWDG
MICROSAR SafeE2E
MICROSAR SafeRTE
▶ MICROSAR SafeBSW
Process and Services
Summary
MICROSAR SafeBSW

- BSW modules developed according to ASIL D
  - Certified by exida

- SafeBSW can be run in the **same OS application** as your ASIL SW components.
  - There is **no need for context switches** for calls to SafeBSW.

- SafeBSW can **implement (parts of) safety mechanisms**.
Choosing the Right Approach

- Calls to BSW are necessary for e.g. external communication, notifications, ...

- If the majority of application software has the same ASIL, performance can be boosted by having an ASIL BSW that allows to coexist in the same partition.
Improving Performance

- **Speedup of ASIL SWC Comm.**
  - BSW can run in same partition as ASIL SWCs
  - No partition switch necessary if ASIL SWCs communicate with BSW
  - Reduced overhead for scheduling of ASIL tasks
  - Direct access to protected registers possible from ASIL drivers

- **Speed-up of QM SWC Comm.**
  - QM SWC can use trusted-function calls to call BSW functions
  - There is a mode-switch, but no context switch ➔ The code is executed on the stack of the caller
  - Resulting time to cross partition boundary is reduced
  - Trusted Function stubs can be automatically generated for BSW services
Agenda

- Introduction
- MICROSAF SafeOS
- MICROSAF SafeWGD
- MICROSAF SafeE2E
- MICROSAF SafeRTE
- MICROSAF SafeBSW

- Process and Services

- Summary
Process and Services

Example timeline for Safety Case

1. Delivery
   Pre-Production
   [Production SIP]
2. Delivery
   Pre-Production
   [Maintenance]
   (optional)

n-th. Delivery
   Production
   [Safety-ready]
   [Maintenance]

(n+1)-th. Delivery
   Production
   [Safe]
   [Safety Case]

Integration of n-th. Delivery

Order

Request Production
   (Safety-ready)
   Delivery

Standard Lead Time

Provision of BSW
   configuration to Vector
   (for ASIL C/D)

26 weeks
   Lead Time for Production
   (Safety-ready)

12 weeks
   Project-specific Safety Case activities*

If Safety Case is required, it has to be ordered at latest with the request for Production (Safety-ready)

*) For ASIL A/B project-specific Safety Case activities
   are started with the Production (Safety-ready) delivery
The following activities are performed to create a Safety Case by Vector:

- **Validation of component tests**
  - Based on your configuration we examine if Vector’s component tests match your configuration. In doubt Vector performs additional tests on your configuration.
  - Necessary for ASIL C/D only

- **Confirmation report for completeness by quality department**
  - Our quality department is considered sufficiently independent (I3).

- **Creation of safety case report**
  - The safety case report summarizes the relevant information, i.e.:
    - Set of MICROsAR Safe components
    - Configuration
    - Microcontroller (and external hardware) derivative
    - Compiler and compiler options
  - The safety case report is your confirmation from Vector that the basic software has the requested ASIL.
Safety Trainings and Workshops

1. MICROSAU Safe Training
   - 1-day basic ISO 26262 knowledge
   - Conception and application of MICROSAU Safe

2. Integrated Safety Training
   - 2-days hands-on-training with Vector Tools (PREEvision, MICROSAU Safe)

3. ISO 26262 Workshop (Vector Consulting Services)
   - Implementation of a safety process
   - Consideration of the specific situation
   - Option: Usage of MICROSAU Safe within the safety project

4. MICROSAU Safe Workshop
   - Analysis of the safety concept
   - Usage of MICROSAU Safe and the Safety Manual
   - Discussion of technical details regarding MICROSAU Safe
   - Review of work products
Agenda

Introduction
MICROSAR SafeOS
MICROSAR SafeWDG
MICROSAR SafeE2E
MICROSAR SafeRTE
MICROSAR SafeBSW
Process and Services

Summary
Summary

Overview and Availability

- **Partitioning:**
  - SafeOS available
  - SafeE2E available
  - SafeWatchdog available

- **RTE:**
  - SafeRTE available

- **BSW**
  - SafeBSW available for most modules (e.g. ETH will follow)

- **ASIL (Tier1/OEM):**
  - SafeOS
  - SafeE2E
  - SafeWatchdog

- **QM (Tier1/OEM):**
  - SafeRTE

- **MICROSAR Safe:**
  - SafeBSW

- **MICROSAR QM:**
  - SafeOS

- **ASIL (3rd party):**
  - SafeE2E

1,2: RTE and BSW are either ASIL or QM

Partitioning: SafeOS, SafeE2E, SafeWatchdog available
RTE: SafeRTE available
BSW: SafeBSW available for most modules (e.g. ETH will follow)
**Components acc. ASIL D available with R22.7 (2019/04)**

### Summary

#### 3rd Party Software
- Vector Standard Software
- Vector Standard Software developed acc. to ASIL D
- Vector Standard Software developed acc. to ASIL B

#### Application

<table>
<thead>
<tr>
<th>OS</th>
<th>SYS</th>
<th>DIAG</th>
<th>MEM</th>
<th>COM</th>
<th>Rte</th>
</tr>
</thead>
<tbody>
<tr>
<td>vC HYP</td>
<td>BswM</td>
<td>Dcm</td>
<td>Ea</td>
<td>E2eXF</td>
<td>Mirror</td>
</tr>
<tr>
<td>OS</td>
<td>Det</td>
<td>Dem</td>
<td>Ee</td>
<td>IpduM</td>
<td>PduR</td>
</tr>
<tr>
<td></td>
<td>EcuM</td>
<td>MemIF</td>
<td>Fee</td>
<td>SecOC</td>
<td>SomeIpT</td>
</tr>
<tr>
<td></td>
<td>StbM</td>
<td>FIM</td>
<td>LdCom</td>
<td>PduR</td>
<td>SomeIpF</td>
</tr>
<tr>
<td></td>
<td>Tm</td>
<td>J1939Dcm</td>
<td>Nm</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WdgM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Rte

<table>
<thead>
<tr>
<th>External</th>
<th>E2ePw</th>
<th>vLinXcp</th>
<th>CAN</th>
<th>LIN</th>
<th>FR</th>
<th>ETH</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td></td>
<td></td>
<td>Can</td>
<td>Lin</td>
<td>Fr</td>
<td>ETH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ien</td>
<td></td>
<td>Tp</td>
<td>Dcp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Complex Driver

<table>
<thead>
<tr>
<th>Complex Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2ePw</td>
</tr>
<tr>
<td>vLinXcp</td>
</tr>
<tr>
<td>CAN</td>
</tr>
<tr>
<td>LIN</td>
</tr>
<tr>
<td>FR</td>
</tr>
<tr>
<td>ETH</td>
</tr>
</tbody>
</table>

#### Os

- vC HYP
  - BswM
  - ComM
  - Det
  - EcuM
  - StbM
  - Tm
  - WdgM

#### Crypto

- Crypto (Sw)
- Crypto (Hw)
- vHsm

#### Mcal

- Adc
  - Crypto (Hw)
  - EthSwt
  - Gpt
  - Mcu
  - RamTst
- Can
  - Dio
  - Fls
  - FIs
c
- CorTst
  - Eep
  - FIsTst
  - Icu
  - Port
  - Wdg
- Crypt(vHsm)
  - Eth
  - Fr
  - Ic
  - Lin
  - Pwm
  - WEth

#### External

- CanTrc
  - LinTrc
  - EthTrc
  - WEthTrc
- Ext
  - V2xBtp
  - V2xFac
  - V2xGn
  - V2xM
  - V2X

#### Libs

- Crc
  - E2e

#### Io

- vDioHwAb

#### Libs

- vCanCcCdm
  - vCanCcGbt
  - vDns
  - vExi
  - vHttp
  - vJsn
  - vScc
  - vXmlEngine
  - vXmlSecurity

#### Avb

- vAvTp
  - vPtp
  - vSr
  - vSrp

#### Hsm

- vHsm
  - Crypto (Hw)
  - vHsm

#### Ext

1. Includes ExtAdc, EepExt, FIsExt, EthSwtDrvExt, EthDrvExt, vMem and WdgExt
2. Functionality represented in EthTSyn and StbM
3. Different variants available
### Safety Building Blocks of MICROSAR Safe

**MICROSAR SafeOS**
- Supports memory partitioning using an MPU
- Provides safe context switch for each safety related task:
  - register settings
  - stack pointer and program counter
  - MPU settings
- Available for single- and multi-core
- OS Applications can be restated individually

**MICROSAR SafeWDG**
- Detects timing and execution order faults
- Provides deadline, alive and logic monitoring
- Capable of using internal or external watchdogs as well as system basis chips (SBCs)

**MICROSAR SafeE2E**
- Ensures safe communication between ECUs
- Available as E2E Protection Wrapper and E2E Transformer
- All AUTOSAR profiles supported

**MICROSAR SafeRTE**
- Ensures safe communication within the ECU
- Supports safe communication across partition boundaries to exchange information between ASIL and QM applications

**MICROSAR SafeBSW**
- Increased performance
  - complete BSW as ASIL software
  - Reduced partition switches
- Additional safety requirements, e.g.:
  - Correct initialization using an ASIL EcuM
  - Safe write/read of non-volatile data
For more information about Vector and our products please visit

www.vector.com

Author:
Dr. Günther Heling, Jonas Wolf
Vector Germany