Quo Vadis Embedded OS?
ECUs gain more and more functionality due to increasing computing power of automotive microcontrollers and new vehicle features.
Operating Systems Characteristics

Introduction

General Purpose OS
- Soft Real-Time
  - Bad Jitter behaviour
  - Bad interrupt latency
- Large Footprint
- Dynamic executable handling
- POSIX compliance
- Broad support for Peripherals
- Slow startup time
- Large Communities
  - Libraries
  - Apps
  - Programming languages

RTOS for embedded systems
- Hard Real Time
  - Good interrupt latency
  - Good jitter behavior
- Small/Medium Footprint
- Dynamic executable handling
- POSIX compliance
- Support for peripherals is vendor specific
- Fast startup time
- Typically microkernel approach
- Some Microkernels compliant to ISO26262

RTOS for deep embedded systems
- Hard Real Time
  - Very good interrupt latency
  - Very good Jitter behavior
- Very small footprint
- Statically configured
- AUTOSAR / OSEK
- Support for peripherals is vendor specific
- Very Fast startup time
- Monolithic
- Compliance to
  - Automotive SPICE
  - ISO26262
Introduction

Drivers for increasing functionality of ECUs

**Infotainment**
- 2D/3D accel. support
- Video Codecs, Streaming support, multi-media library, etc...

**Connectivity**
- Car-2-X (LTE, WiFi, GPS, etc.)
- Multimedia (USB, SD-Card, NFC, etc.)

**Hardware**
- Increasing Performance
  - Multicore
  - GPUs,...

**Image Processing / Sensor Fusion**
- ADAS: Camera/Radar/LIDAR
- Image processing and machine learning

**Vision: Dynamic Software Platform**
- “App-Store” for automotive applications
- Currently not in the direct focus
Use Cases

Introduction

High Level functionality + ASR Classic connectivity

High Level functionality + ASR Classic connectivity

AUTOSAR Adaptive

Multiprocessor

Consolidation
Agenda

Introduction

- How to integrate high level functionality?
  And what about deep embedded systems?

Summary
AUTOSAR Adaptive

- Targets
  - Highly automated driving
  - Car2-X
  - Connectivity
  - Vehicle in the cloud

- POSIX OS as basic runtime environment
- Planned Dynamics

- The whole system needs to be
  1. Fail-Safe
  2. Fail-Operational
  3. Secure
How to integrate high level functionality?

**POSIX only systems**

- High Level functionality already implemented on POSIX OS
  - Image processing algorithms
  - 2D/3D visualization
  - Machine learning

- AUTOSAR BSW elements can be executed on top of POSIX OS to fill this gap.
  - MSR Guest OS
  - Other BSW components
POSIX only systems

- Tight Timing constrainings may not be met in this setup
- HW Fault metrics are typically an issue
  - HW too complex
- Higher ASIL levels may not be reachable
- Can also be realized by
  - Multiprocessor setup
  - Hypervisor setup
How to integrate high level functionality?

Using a hypervisor to add high level functionality in a safety environment

- Suitable for mixed criticality
- Current HW architectures typically offer a separate safety core
- High Level functionality can still run on the high performance cores
- Safety relevant functionality is isolated on separate safety core
  - ASIL decomposition
- Another alternative is redundancy
Agenda

Introduction
How to integrate high level functionality?

- And what about deep embedded systems?

Summary
And what about deep embedded systems?

- There is still the need for deep embedded systems
  - Cost as a driver

- Automotive MCUs have more and more cores
  - Next generation has up to 6 cores

- Restriction of parallelizability
  - Communication
  - Shared Resources
  - Synchronization

- Parallelizability is the key
Consolidation the AUTOSAR way

- AUTOSAR OS supports multicore functionality
- SWCs can be distributed among cores
- According to Amdahl’s Law parallelizability is the key to success
- If parallelizability is not possible multicore systems can become slower than single core systems
- Alternative use case for multicore systems?
And what about deep embedded systems?

**System partitioning**

- Multicore systems can also be used for consolidation
- Mixed criticality possible
  - SW/HW supports up to ASIL-D
- 1:n mapping of Guests to Cores
  - Otherwise hierarchical scheduling becomes a topic
- Elimination of shared resource usage where possible
- Virtualization of shared resources where necessary
And what about deep embedded systems?

**System partitioning**

- Small hypervisor footprint
- No or low virtualization overhead
- Each partition can be configured separately → Easy integration
- Hypervisor may provide inter partition communication interface

---

![Diagram of System Partitioning](image-url)
Introduction
How to integrate high level functionality?
And what about deep embedded systems?

Summary
There are different approaches to integrate high level functionality in embedded systems

- AUTOSAR Adaptive is on its way

Until that

- POSIX Only systems with hosted ASR components
- Hypervisor setups
- Redundancy

Automotive MCUs gain more and more cores (many core systems)

- Lean hypervisor setups that focus in separation only
For more information about Vector and our products please visit

www.vector.com

Author:
Dr. Timo Kerstan
Vector Germany