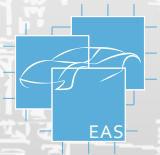
A Co-Designed RTOS and MCU Concept for Dynamically Composed Embedded Systems

Renata M. Gomes, Marcel Baunach, Maja Malenko, Leandro B. Ribeiro, Fabian Mauroner



Renata Martins Gomes
renata.gomes@tugraz.at

Institute of Technical Informatics
Embedded Automotive Systems Group
Graz University of Technology

27 June 2017

Agenda

1. Introduction
2. MCSmartOS
   - Dynamic Composition
   - Portability
3. The mosartMCU
   - OS-Aware MCU Architecture
   - Security
4. Evaluation Platform

Agenda

1. Introduction

2. MCSmartOS
   - Dynamic Composition
   - Portability

3. The mosartMCU
   - OS-Aware MCU Architecture
   - Security

4. Evaluation Platform

Introduction
The Future of Embedded Systems

Upcoming technologies

- Car2X communication
- Autonomous cars
- Home automation
- Medicine
- Industry 4.0
- Internet of Things (IoT)

Resulting challenges

- High diversity of
  - Hardware
  - Software
  - Services
- Huge network
- Life-critical applications
- Long-term support after deployment

Introduction
The Future of Embedded Systems

What will be needed

- Computational power
- Dependability
  Availability, reliability, safety, integrity, confidentiality, maintainability
- Dynamic composition
- Reconfigurable harware
- Portability

→ Co-designed hardware-software architecture

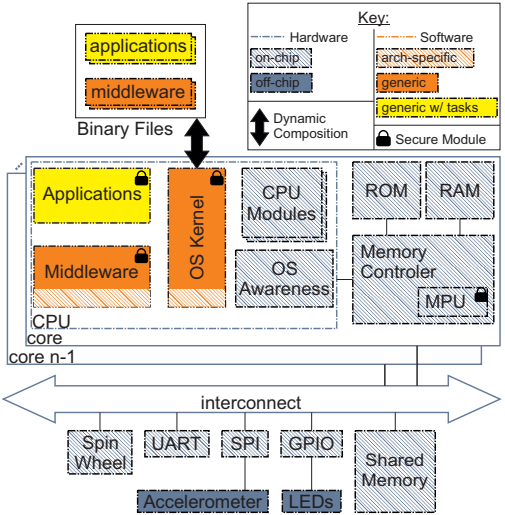
Introduction

*MCSmart*OS and *mosart*MCU

MCSmartOS

MosartMCU

Multi-core Operating-System-Aware Real-Time



Introduction
*MCSmart*OS and *mosart*MCU

Five dissertations:

- L. Ribeiro → Dynamic composition
- R. Gomes → Portability
- F. Mauroner → *mosart*MCU
- M. Malenko → Security
- P. Brungs → Hardware partial reconfiguration

Agenda

1. Introduction

2. MCSmartOS
   - Dynamic Composition
   - Portability

3. The mosartMCU
   - OS-Aware MCU Architecture
   - Security

4. Evaluation Platform

MCSmartOS
Overview

Multicore RTOS for the IoT and automotive domains

- Dependability
- Portability
- Hard and soft real-time
- Dynamic
  - Module changes
  - Reaction to events
  - Dynamic resource sharing
- Small ROM and RAM footprint

*MCSmart*OS
Current State

Kernel

- Priority-based scheduling
- Centralized interrupt handling
- Synchronization mechanisms
  - Events
  - Resources
- Time management
- Multi-target toolchain

Support for

- MSP430
- Aurix
- RISC-V (softcore)
- *mosart*MCU (softcore)

*MCSmart*OS
Dynamic Composition

Remote updates mandatory for bug/security hole fixes, changing requirements and laws, new applications, etc

- Huge number of devices
- Devices in unreachable places
- Variant diversity
- Keep system dependable

→ *Dynamic composition* of software!

*MCSmart*OS analyzes changes before they happen to guarantee **interoperability**

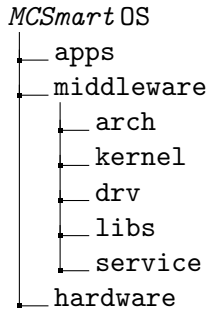
- Compatibility
- Consistency
- Schedulability

*MCSmart*OS
Portability

The software environment is built such that it is

- Portable
- Maintainable
- Developer friendly

Building an application is straightforward and the most specific software will automatically be selected by the make environment.



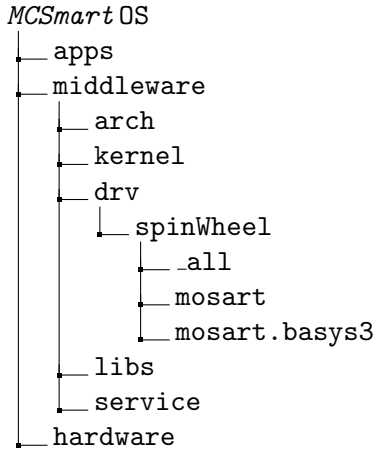
*MCSmart*OS
Portability

Application's `Makefile`:

```
BOARDS += mosart.basys3
BOARDS += mosart.nexus4
BOARDS += msp430f5529.LaunchPad

LIBS += drv.ledWheel
```

- Build for *mosart*MCU on Basys3 board
- Build for *mosart*MCU on Nexus4 board
- Build for MSP430f5529 on LaunchPad board



Agenda

1. Introduction
2. MCSmartOS
   - Dynamic Composition
   - Portability
3. The mosartMCU
   - OS-Aware MCU Architecture
   - Security
4. Evaluation Platform

The *mosart*MCU
Overview

Multicore MCU with OS awareness

- Softcore
- Based on RISC-V/*vscale*
- User-defined on-chip peripherals
  - USART
  - Timer
  - Profiling unit
  - ...
- Partially reconfigurable at runtime

The *mosart*MCU
OS-Aware MCU Architecture

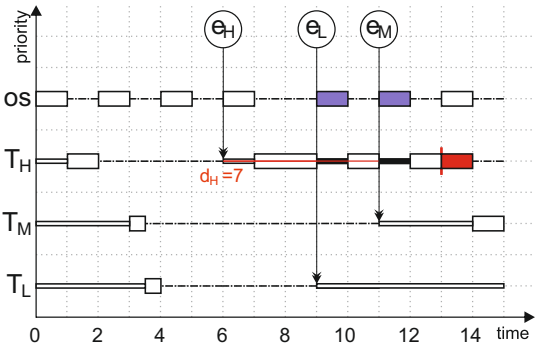
Goals

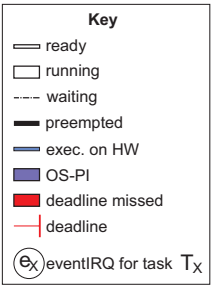
- Increase overall efficiency
- Support the RTOS
  - Task awareness
  - Event/IRQ handling
  - Resource awareness
- Direct access to OS data structures
  - OS housekeeping in hardware
  - Unbounded number of tasks and resources

The *mosart*MCU

OS-Aware MCU Architecture - Example

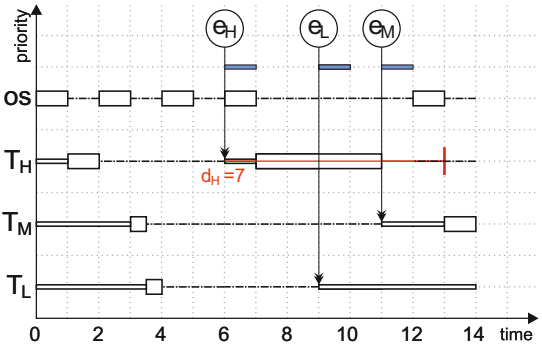
Problem: kernel priority inversion when handling interrupts and events

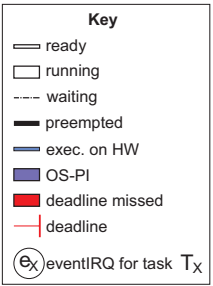




The *mosart*MCU
OS-Aware MCU Architecture - Example

EventIRQ: avoid kernel priority inversion when handling interrupts and events





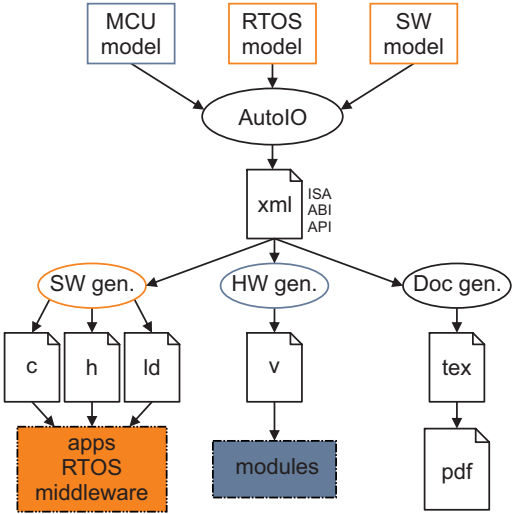
Accepted paper DSD/2017 - EventIRQ: An Event based and Priority aware IRQ handling for Multi-tasking Environments

The *mosart*MCU
Interface - Auto IO

Tool to generate the interface between MCU and OS

- Model hardware and its peripherals
- Model RTOS and applications
- Generate hardware and software modules, and documentation

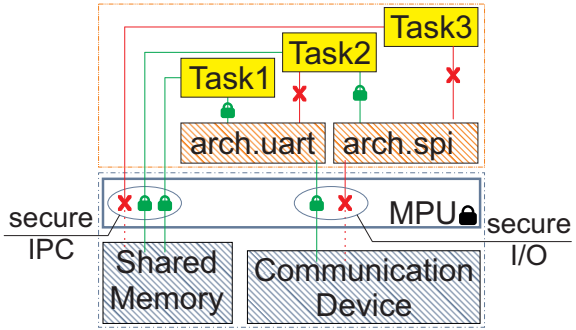
→ Firmware and bitstream



The *mosart*MCU
Security

Kernel manages a tailored MPU. Tasks can only access their own code and data, and addresses they share with other tasks.

- Isolate tasks
- Secure inter-process communication (IPC)
- Secure access to I/O devices



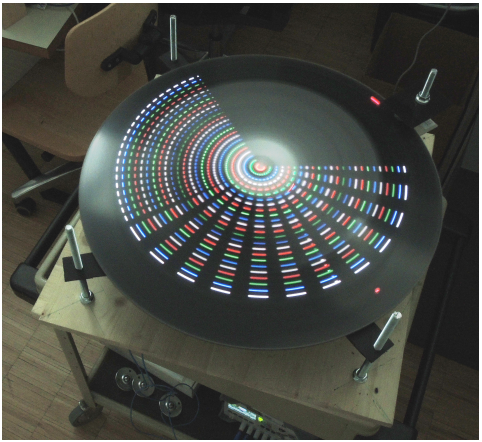
Agenda

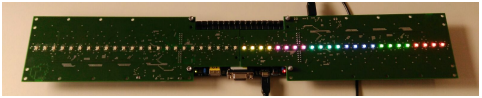
1. Introduction
2. MCSmartOS
   - Dynamic Composition
   - Portability
3. The mosartMCU
   - OS-Aware MCU Architecture
   - Security
4. Evaluation Platform

Evaluation Platform
Circular LED Display

Test platform for our concepts

- *MCSmart*OS + *mosart*MCU (On Artix-7 FPGA)
- External interferences
- Dynamic composition
- Security attacks
- Real-time guarantees





Thank you!