Rapid Development for Embedded Systems
by Integration of Qualified Software Components

Ulrich Beier, Hitex Development Tools GmbH
February 2015
Challenges for Software Development of Embedded Systems

- Complexity of the Applications
- Availability of High Performance µC-Architectures
- Demand for a high degree of Reuse of Software
- Safety Requirements of the System
Complexity of the Applications

- High-Speed System Response-Time
- Inter-System Communication via different Channels
- Safety-Monitoring
Availability of High Performance µC-Architectures

- Multi-Core Architecture
- Complex Peripherals
- Safety Functions On-Chip
Demand for a high degree of Reuse of Software

- Application Software on High Software Appstraction Layer
- Intensive Usage of Real-time Operating Systems
- System Communication on Higher Protocol Level
Safety Requirements of the System

- Integration of **High Level Safety Functions**
- **Safety System Monitoring**
- Development according to a **Standard**
Criteria
- Know-How not available?
- Development Process according to Standards not established?
- Time to Market critical? ->
- Long Term Development Resources Allocation not possible?

Effects “Effort”
- Acquisition of Know-How for the whole team -> cost and time!
- High effort for the first project -> cost and time!
- Effort 4 times higher than for Standard project -> cost and time!

Effects “Feasibility”
- No access to Full FMEDA and μC internals -> certification!
- Proof of DC not completely possible -> certification!
Microcontroller Software Components (Middleware)

- Low-Level Drivers
- Safety Drivers
- Application Drivers
- File Systems
- DSP Libraries
- Communication Stacks
- RTOS

Software Components (Middleware)
Microcontroller Software Components – Safety Drivers

- Functional Safety Standards
- Self Test Libraries – *What is common and needed*
- Existing Safety Libraries
- Example Safety Library **Overview** -> SafeTlib
- Example Safety Library **Deliverables** -> fRSTL
Functional Safety Standards related to IEC61508
Self Test Libraries – What is common and needed

- Safety Concept with Safety Manual
- FMEDA of µC
- Complete set of needed Self Test Functions
- Documentation of Detection Coverage (DC)
- Complete Development Documentation

The only way to reach SIL 3 with acceptable effort and cost
### Self Test Functions – Existing Safety Libraries

<table>
<thead>
<tr>
<th>Library</th>
<th>Platform</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SafeTlib</td>
<td>AURIX TC2xx</td>
<td>Infineon</td>
</tr>
<tr>
<td>SafeTcore</td>
<td>XC2300, TriCore</td>
<td>Infineon</td>
</tr>
<tr>
<td>fRSTL</td>
<td>STM32, Cortex M</td>
<td>Yogitech</td>
</tr>
<tr>
<td>SafeTI</td>
<td>TMS570, RM48</td>
<td>Texas Instruments</td>
</tr>
<tr>
<td>ClassB</td>
<td>Cortex M</td>
<td>Energy Micro, Infineon, NXP…</td>
</tr>
</tbody>
</table>
Example Safety Library Overview -> SafeTlib

Core 0 Upper Layer

- Safe Watchdog Manager
- Test Manager

Safe Watchdog

- SafeWDG Interface
- Watchdog Driver
- SPI Driver
- IO Driver

SafeTlib

- Test Handler
- MicroTest Lib
- SMU Driver

Core x Upper Layer

- Test Manager

SafeTlib

- Test Handler
- MicroTest Lib

AURIX™ MCU
Full safety according to SIL2, SIL3

- fRSTL libraries for Cortex STM32

Deliverables:
- Source code (C and assembler)
- Compilation scripts for Keil, IAR and Raisonance’s GCC (support for other compilers may be available upon request)
- Datasheet
- User guide
- Safety documentation
Microcontroller Software Components – Low Level Drivers

- Basic Software - Layered Architecture
- Example of a Software-Layer-Structure for Automotive Applications
- Benefits to use Low-Level Driver for Automotive Application
Basic Software - Layered Architecture

Application Layer

AUTOSAR Runtime Environment (RTE)

System Services
- Onboard Device Abstraction
- Microcontroller Drivers

Memory Services
- Memory Hardware Abstraction
- Memory Drivers

Communication Services
- Comm. Hardware Abstraction
- Communication Drivers

I/O Hardware Abstraction
- I/O Drivers

Complex Drivers

BSW-Layers

Services
- ECU Abstraction and Complex Drivers
- Microcontroller Abstraction
Example of a Software-Layer-Structure for Automotive Applications

**Services Layer**
Highest layer. Provide basic services for application and basic software modules.
- Operating system services
- Vehicle network communication management services
- Memory services (NVRAM management)
- Diagnostics Services
- ECU state management

**ECU Abstraction Layer**
Interfaces the drivers of the µC Abstraction Layer. It also contains drivers for external devices.
**Task:**
Provides independence of higher software layers to the ECU hardware layout

**Microcontroller Abstraction Layer (MCAL)**
Lowest software layer of the Basic Software. It contains drivers, which are software modules with direct access to the µC internal peripherals and memory mapped µC external devices.
**Task:**
Make higher software layers independent of µC
Benefits to use Low-Level Driver for Automotive Application

**MC-ISAR**

- **Released for production**
  enables direct use in serial production

- **Qualified release**
  compliant to CMM L3 reduce development cost

- **Efficient implementation**
  for lowest resource consumption

- **Evaluation version (FoC)**
  Provided by Hitex

- **Documented product release**
  allows easy to use

- **Application Independent**
  from Chassis over Body to Powertrain

Copyright © Infineon Technologies 2012. All rights reserved.
Benefits to use Low-Level Driver for Automotive Application

Cost Reduction

- **Released** for mass production
- Extreme reduction of development effort and focus on application code and complex drivers
- Drivers available as standardized components
- **Reuse** standardized drivers in multiple applications

Time to Market

- Start with the application development shortly after the availability of Hardware
- Useable with or without AUTOSAR environment
- The complete AUTOSAR suite is available via Hitex
Functional Safety and RTOS

- RTOS has to be safe itself

RTOS examples with certification:
  - SAFERTOS
  - PXROS-HR
SAFE RTOS has been developed in accordance with the processes mandated by IEC 61508 functional safety standard for Safety Integrity Level (SIL) 3 applications and IEC 62304 Software Safety Class C Medical Device.

SAFE RTOS is periodically assessed and certified by TÜV SÜD.

The processes used to develop SAFE RTOS have also been certified as being compliant with ISO 9001
SAFERTOS is delivered as 2 main components:

- SAFERTOS ‘C’ source code
- Design Assurance Pack (DAP)

Why DAP? 

- It provides complete transparency about the SAFERTOS development
- It enables easy integration with customers process
- It ensures we can support any challenges to certification
What is PXROS-HR?
- Powerful
- Certified

Real-time Operating System (RTOS)

Main Features
- Real-time micro-kernel OS (*no interrupt locks*)
- MPU protected Multi-Core inter-task communication (*no shared memory needed, no spin-locks*)
- SIL-3 certification support
- Static and dynamic resource management (fail-operational)
Make or Buy?

It’s your decision!

It’s your project!

It’s your success!

Thank you for your attention