ABOUT AUTOSAR CLASIC & ADAPTIVE ON NXP AUTOMOTIVE DEVICES

MARIUS, ROTARU
SOFTWARE ARCHITECT – AUTOMOTIVE SOFTWARE

GEORGE, CIUSLEANU
SOFTWARE ARCHITECT – AUTOMOTIVE SOFTWARE

AMF-AUT-T2789 | AUGUST 2017
AGENDA

• PART 1 – AUTOSAR CLASSIC PLATFORM
  - Introduction to AUTOSAR Classic Platform
  - Evolution: new features added by R4.3.x
  - NXP AUTOSAR Classic SW Products

• PART 2 – AUTOSAR ADAPTIVE PLATFORM
  - Introduction to AUTOSAR Adaptive Platform
  - Running AUTOSAR Adaptive Demonstrator on NXP Platforms
PART 1: AUTOSAR CLASSIC PLATFORM
INTRODUCTION TO AUTOSAR CLASSIC PLATFORM
Embedded Software

Mars Curiosity Rover
5 MLoC

Android
11.8 MLoC

F-35 Joint Strike Fighter
23.5 MLoC

Mercedes S Class
~100 MLoC

Software Size (million Lines of Code)

Source:
http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code
http://www.informationisbeautiful.net/visualizations/million-lines-of-code/
https://www.linkedin.com/pulse/20140626152045-3625632-car-software-100m-lines-of-code-and-counting
AUTOSAR Vision

• AUTOSAR aims to improve the complexity management of integrated E/E architectures through increased reuse and exchangeability of software modules between OEMs and suppliers.

Source: Autosar_GuidedTour.pdf

(AUTomotive Open System Architecture)
AUTOSAR Partnership Structure

Source: AUTOSAR
AUTOSAR Classic – Architecture: Overview of Software Layers

Source: AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
AUTOSAR Classic - Architecture: Overview of Modules
AUTOSAR Classic – Architecture: Basic Approach

Virtual Integration
Independent of hardware
Virtual Functional Bus.

Introduction of Hardware Attributes
Holistic view of the entire system, both software and hardware.

ECU Configuration
Run-Time Environment
Separation of system into its ECU (plus common infrastructure).

Source: AUTOSAR

1. Software Component (SW-C) description
2. Integration of SW-C via Virtual Functional Bus (VFB)
3. ECU Description
4. System Constraints
5. Mapping of SW-C on specific ECU
6. Configuration of Basic Software Modules (BSW) and Run-Time Environment (RTE)
AUTOSAR Classic – Architecture: Standardized Interfaces
EVOLUTION OF AUTOSAR CLASSIC – R4.3.X

CRYPTO STACK
The architectural design of the CSM (Crypto Service Manager) and the CAL (Cryptographic Abstraction Library) consists of two layers:

- **Wrapper Layer** – acts as the Interface for BSWs, CDDs or as handle for the RTE
- **Implementation Layer** – provides either the algorithm of the service or accesses the API of a cryptographic library (hardware based)

The Implementation Layers are defined in the AUTOSAR SWSs:

- The **CPL (Cryptographic Primitive Library)** for the CAL.
- The **CRY (Cryptographic Library Module)** for the CSM

Usage of CSM is recommended if Security Hardware is used (support for Async Mode).

The CSM supports everything the CAL does and more.

CAL has been deprecated from ASR4.3.0 !!!
New Crypto Stack features:

- Designed **to provide services** with cryptography functionality, **based on software libraries and/or on hardware modules** (internal/external)

- **Flexible integration** with the other stacks (Comm, V2X, Diagnostic)

- Introduced **the base for key management concept**

- Different mode of operation "streaming" (start, update, finish) and “one-shot”

- **Optimized for HSM Hw** (introduced job concept to allow running in parallel different crypto primitives)

References:

AUTOSAR_SWS_CryptoServiceManager.pdf
AUTOSAR_SWS_CryptoInterface.pdf
AUTOSAR_SWS_CryptoDriver.pdf
AUTOSAR Classic - New Crypto Stack: Detailed View

Crypto Driver:
- CryptoDrv is a driver for a specific device providing independent Crypto Channels with supported CryptoPrimitives
  - SHE: single instance- no support of queueing
  - HSM: multiple virtual instances with different processing priorities and HW-queuing support;
- Could be available in a pure software implementation
- Offers generic interface for synchronous and asynchronous cryptographic primitives.
- supports key storage, key configuration, and key management for cryptographic services.

Crypto Hardware Abstraction (Interface)
- group of modules which abstracts from the location of cryptographic primitives (internal- or external hardware or software-based)

Crypto Service Manager (CSM)
- provide cryptographic primitives and key storage to the application in a uniform way. Abstract from hardware devices and properties.
AUTOSAR Classic - Crypto Stack: Channelization for HSM efficiency

- The path from a CSM queue via the CryIf to a Crypto Driver Object is called a channel.

- Crypto Driver Object represents an instance of an independent crypto “device”.

- Optionally a queue in Crypto Driver to optimize the HW usage.

- NXP HSM/HSE Subsystems supports channelization concept.
AUTOSAR Classic- Crypto Services and Key Management Interface

Crypto Services:
- AEAD_DECRYPT
- AEAD_ENCRYPT
- DECRYPT
- ENCRYPT
- HASH
- MAC_GENERATE
- MAC_VERIFY
- RANDOM
- SIGNATURE_GENERATE
- SIGNATURE_VERIFY

Key Management:
- Key Setting Interface
- Key Extraction Interface
- Key Copying Interface
- Key Generation Interface
- Key Derivation Interface
- Key Exchange Interface
- Certificate Interface
EVOLUTION OF AUTOSAR CLASSIC – R4.3.X

OFF-BOARD COMMUNICATION V2X
AUTOSAR Classic - Off-Board Communication: V2X

- The **Off-board Communication Services** are a group of modules for Vehicle-to-X communication via an ad-hoc wireless network.
- **Facilities**: implement the functionality for reception and transmission of standardized V2X messages, build the interface for vehicle specific SW-Cs
- **Basic Transport Protocol** = Layer 4
- **Geo-Networking** = Layer 3 (Addressing based on geographic areas, etc…)
- **V2X Management**: manages cross-layer functionality (like dynamic congestion control, security, position and time)

New Specifications added:
- Specification of Vehicle-2-X Geo Networking
- Specification of Vehicle-2-X Basic Transport
- Specification of Vehicle-2-X Facilities
- Specification of Vehicle-2-X Management
- Specification of Wireless Ethernet Driver
- Specification of Wireless Ethernet Transceiver Driver
AUTOSAR Classic - Off-Board Communication: V2X Integration

Planned to be implemented by NXP
EVOLUTION OF AUTOSAR CLASSIC – R4.3.X

SAFETY
AUTOSAR Classic – Safety: Hardware Test Manager

- **Hardware Test Management on Startup and Shutdown (HTMSS)**
  - infrastructure for integrating/transforming the microcontroller manufacturer specific start up and shutdown tests (e.g. BIST) results/status within the AUTOSAR standard software platform

- **Features:**
  - collects the test results/status from the MSTP
  - configures MSTP tests,
  - start tests execution
  - provides the MSTP test status to EcuM module and application SWC to evaluate the test results for the system behavior.

 Planned to be implemented by NXP
AUTOSAR Classic – Safety: Hardware Test Manager

• Role of HTMSS module in different phases of the standard AUTOSAR software
**Motivation:** QM software needs to access area that is restricted in user mode. QM software should not run in supervisor mode.

**Solution:**
- New **Peripheral Access APIs** and **Interrupt Handling Access APIs** to enable the QM software to access restricted areas. The access rights are configured per OS application.
- Accessible from BSW and CDDs.
AUTOSAR Classic – Safety: Operating System-QM Access to Peripherals and Interrupts

- New AUTOSAR OS APIs for HW Peripheral and Interrupt Controller Access

```c
WritePeripheral32/16/8/AreaIdType Area, uint32* Address, uint32 WriteValue)
WritePeripheral32/AreaIdType Area, uint32* Address)
ModifyPeripheral32/AreaIdType Area, uint32* Address, uint32 Clearmask, uint32 Setmask)
```

EnableInterruptSource(ISRType ISRID, boolean ClearPending )
DisableInterruptSource(ISRType ISRID)
ClearInterruptSource(ISRType ISRID)

QM Application (BSW)

```
... WritePeripheral32/AreaId, Address, WriteValue)
...
```

AUTOSAR OS

```
WritePeripheral32/AreaIdType Area,
uint32* Address,
uint32 WriteValue)
{
    /* Validates the request */
    /* Changes to Supervisor Mode */
    /* Performs the request */
    /* Changes back to previous mode */
}
```

NXP AUTOSAR MCAL can run in both SUPERVISOR (Privileged) and USER(Non-Privileged) modes!
AUTOSAR Classic – Milestones

ASR R4.3.1
(08.12.2017)

SERVICE RELEASE

ASR R4.4.0
(29.10.2017)

- SECURITY:
  CONC_626_POLICYMANAGER
  CONC_636_SECURITYEXTENSIONS

- COMMUNICATION:
  CONC_631_LINSLAVESUPPORT
  CONC_632_ETHERNETWAKEONDATALINE
  CONC_634_BUSMIRRORING
  CONC_637_COMSTACKCLEANUP
  CONC_629_SOMEIPTLPROTOCOL
  CONC_635_SOMEIPTLV

- OS/RTE:
  CONC_628_ARTI
  CONC_630_RTE_IMPLEMENTATION_PLUG-INS

Source: AUTOSAR
AUTOSAR Classic – at NXP

NXP is involved in the following WPs:

- WP-A / WP-A-LIB
- WP-A2 (V2X)
- WP-A5
- WP-X-SEC

- AUTOSAR MCAL/OS
- Middleware Solutions and Enablement for NXP parts
NXP AUTOSAR SOFTWARE PRODUCTS
AUTOSAR Classic – NXP SW Solutions: Overview

From NXP: MCAL (source code), OS (source code) and Config Tool (executable) for MCAL and OS

From 3rd party AUTOSAR software companies: The rest of AUTOSAR basic software as needed and integration services

- 3rd party software companies for AUTOSAR are e.g.:
  - Arccore
  - Elektrobit
  - ETAS
  - KPI
  - Mentor Graphics
  - Vector
  and others.
AUTOSAR Classic - NXP MCAL Package

- Production qualified software abstraction of complex hardware features
  - **MCAL** is maintained as a single SW codebase (per AUTOSAR version):
    - Easy porting from one platform to another.
    - Integrates new features and defect corrections implemented on other platforms.
  - Developed using SPICE Level 3 compliant process.
  - Developed as Safety Element out of Context (SEooC) integrable within ASIL-D product.
  - Compliant to versions 4.3 (under development) 4.2, 4.0, 3.2, 30 or 2.1 of the AUTOSAR standard
  - Integrates multiple software features requested by different customers as extensions to AUTOSAR standard
    - 17 standard drivers (MCU, PORT, DIO, ADC, GPT, PWM, ICU, WDG, FLS, EEP, FEE, SPI, LIN, CAN, OCU, FlexRay, Ethernet)
    - 6 complex drivers extending AutoSAR on various platforms:
      - MCL – centralized DMA and DMAMux configuration and functionality, common timer code – all platforms
      - I2C – Inter-Integrated Circuit driver – S32K
      - MCEM – MicroController Error Management covering FCCU, MEMU – safety extension
      - CRCU – CRC hardware acceleration – safety extension
      - CSEc- Crypto Service engine drivers for CSM integration

**efficient qualified SW layer abstracting microcontroller complexity**
<table>
<thead>
<tr>
<th>AUTOSAR Classic - NXP MCAL Products</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advanced Driver Assistance Systems</strong></td>
</tr>
<tr>
<td>MCAL 2.1</td>
</tr>
<tr>
<td>MPC564xB-C</td>
</tr>
<tr>
<td>MPC560xS</td>
</tr>
<tr>
<td>MPC5668G</td>
</tr>
<tr>
<td>Connectivity &amp; Security</td>
</tr>
<tr>
<td>Vehicle Dynamics and Safety</td>
</tr>
<tr>
<td>MCAL 2.1</td>
</tr>
<tr>
<td>MPC563xM</td>
</tr>
<tr>
<td>MPC564xF</td>
</tr>
<tr>
<td>MPC560xP</td>
</tr>
<tr>
<td>MPC564xL</td>
</tr>
<tr>
<td>MPC567xK</td>
</tr>
<tr>
<td>MPC574xP</td>
</tr>
<tr>
<td>General Purpose &amp; Integrated Products</td>
</tr>
<tr>
<td>MCAL 2.1</td>
</tr>
<tr>
<td>MPC551x</td>
</tr>
<tr>
<td>MPC560xB</td>
</tr>
<tr>
<td>S12</td>
</tr>
</tbody>
</table>

- **Discontinued**
- **Retired**
- **Maintenance**
- **Optimization**
- **In development**

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC564xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPC560xB-C</th>
<th>MPC564xB-C</th>
<th>MPC560xB</th>
<th>MPC560xB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
<td>ARM</td>
</tr>
</tbody>
</table>
AUTOSAR Classic - MCAL Release Packages

- **Software Package** - containing drivers as eclipse plugins for Elektrobit Tresos
  - For each driver:
    - Source code + configuration templates
    - Driver User Manual
    - Driver Integration Manual
  - For entire package:
    - MCAL sample application
    - MCAL Release Note

- **Quality Package** - delivered to customers for RTM releases (only on request for Beta)
  - For each driver:
    - Driver Test Specification
    - Driver Test Summary Report
    - Driver MISRA Summary Report
    - Driver Code Coverage Summary Report
    - Driver Traceability Matrix
    - Driver VSMD Report
    - Driver Code size, Stack size, RAM size Reports
    - Driver Static analysis Report (added only on customer request)
  - For entire MCAL package:
    - MCAL Test Summary Report
    - MCAL Quality Matrix
    - List of changes

- **Safety Package** - delivered to customers for IS RTM releases
  - For each driver:
    - Driver FMEA
  - For entire MCAL package:
    - MCAL Safety Manual
AUTOSAR Classic - NXP AUTOSAR OS

• OSEK based Preemptive real-time multitasking operating system
• Compliant with AUTOSAR specifications: v2.1, v3.x, v4.x
• Supports: s12(x), s12z, PowerPC, Arm Cortex-Mx
• Multicore support
• Available in Scalability Classes 1, 2, 3, 4 to fit the needs of different applications.

<table>
<thead>
<tr>
<th>Feature</th>
<th>SC1</th>
<th>SC2</th>
<th>SC3</th>
<th>SC4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSEK OS features</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Stack monitoring</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Timing protection</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Memory protection</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>OS-Apps</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Service Protection</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Trusted Functions</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
## AUTOSAR Classic - NXP AUTOSAR OS: Availability Matrix

<table>
<thead>
<tr>
<th>QM</th>
<th>ISO26262 ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR 4.0.3</td>
<td></td>
</tr>
<tr>
<td>MPC5643L</td>
<td>MPC574xB-C-G</td>
</tr>
<tr>
<td>MPC564xB-C</td>
<td>S32V23 (A53, M4)</td>
</tr>
<tr>
<td>MPC560xB-C-D</td>
<td>S32V23 (A53, M4)</td>
</tr>
<tr>
<td>MPC560xP</td>
<td>S32Rx7 (Z4, Z7)</td>
</tr>
<tr>
<td>MAC57D5x (M4)</td>
<td></td>
</tr>
<tr>
<td>S12Z MagniV</td>
<td>S32K14x</td>
</tr>
<tr>
<td>S32K11x</td>
<td>S32V23 (A53, M4)</td>
</tr>
</tbody>
</table>

| ASR 4.2     |                   |
| MPC564xB-C-G|                   |
| S32V23 (M4) |                   |

| ASR 3.0     |                   |
| MPC563xM    |                   |
| MPC5604E    |                   |

Available, commercially licensed
In development / planned
Not planned
Available from www.nxp.com
Customers Benefits:

- Reduced Time-to-Market
- Reduced Risk
- Reduced Development Cost
### AUTOSAR Classic: Wireless Charging and NFC

**15W MULTICOIL AUTOMOTIVE TRANSMITTER**

**Solution:** WCT-15WTXAUTOS  
**Device:** WCT1014S, NCF3340, SBC

**Key Features:**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compliant with Wireless Power Consortium (WPC) Extend Power specifications to support up to 15W power transfer with full AutoSAR Support</td>
<td></td>
</tr>
<tr>
<td>Coil configuration from 1 coil to N Coil</td>
<td></td>
</tr>
<tr>
<td>Integrated digital demodulation</td>
<td></td>
</tr>
<tr>
<td>Support free positioning multiple coils low power transmitter solutions using fixed frequency PWM control (Better EMC)</td>
<td></td>
</tr>
<tr>
<td>Support both Q-Factor and Power Loss FOD methods</td>
<td></td>
</tr>
<tr>
<td>Support key FOB and AM band avoidance</td>
<td></td>
</tr>
<tr>
<td>Low standby power consumption</td>
<td></td>
</tr>
<tr>
<td>NFC ready (NCF3340) and CAN/LIN interface</td>
<td></td>
</tr>
<tr>
<td>Support PMA SR1 specifications and Samsung Galaxy Quick Charging</td>
<td></td>
</tr>
<tr>
<td>AEC-Q100 grade 2 certification</td>
<td></td>
</tr>
<tr>
<td>Voltage/current/temperature protection</td>
<td></td>
</tr>
<tr>
<td>Software based solution with NXP embedded wireless charger software libraries to provide maximum design freedom and product differentiation</td>
<td></td>
</tr>
<tr>
<td>FreeMASTER GUI tool to enable customization and calibration</td>
<td></td>
</tr>
<tr>
<td>Compliant with the Qi v1.2.3 specification</td>
<td></td>
</tr>
</tbody>
</table>

**Availability & Certification:**

- Available for Q3 2017
- Qi Certified – device, software now, full solution Q4 2017
AUTOSAR Classic: WPC & NFC INTEGRATION INTO AUTOSAR
WCT-15WTXAUTOS – MWCT1014S, NCF3340

Note: This image is not complete with respect to all internal paths and data flows.

AUTOSAR NFC Stack Architecture
AUTOSAR Classic: NXP Ethernet SW

AVB

- Camera SW
  - Video Streaming
  - Camera Application
  - MPC5603 Reference KIT
- Audio SW
  - Audio Streaming Driver
  - Audio Interface
- Protocols
  - gPTP
  - AVDECC
  - SRP

ROUTERS

- Fast Path Router
  - FPR for MPC574XG
  - FPR for S32G A53 (QNX)
  - FPR for S32G A53 (Linux)
  - FPR for S32G M7
- DoIP Router
  - DoIP router for MPC574XG
  - DoIP router for S32G

SWITCHES

- MPC574xG
  - MPC574xG Switch Driver
- SJA11xx
  - gPTP Bridge stack for SJA1100
  - 802.1X for SJA110x
  - SRP for SJA110x

MOTIVATION FOR ETHERNET

- The cable weight and cost reducing
- Synchronous and low latency - extending of AVB protocols (TSN)
- High network bandwidth – need for fast routing
- Mixed data traffic and security - distributed (Central and Domain) gateways
AUTOSAR Classic: NXP Ethernet SW for External Components

**Ethernet Switch Driver (EthSwtDrv)**
- Provided in for latest AUTOSAR version 4.3.0
- Available for SJA1105\{P,Q,R,S\} switches
- Supporting time synchronization over Ethernet
- Includes high-end (non-standard) features like 802.1Qbv (Time Aware Shaper) support

**Ethernet Transceiver driver (EthTrcvDrv)**
- Provided in for latest AUTOSAR version 4.3.0
- Available for TJA1102(S) PHYS
- Including advanced features like wake up over data line
AVB/TSN Stacks as AUTOSAR CDD
PART 2.
AUTOSAR ADAPTIVE PLATFORM
AUTOSAR ADAPTIVE PLATFORM OVERVIEW
AUTOSAR Adaptive - Motivation
New Application Domains & UseCases

Highly automated driving

Car-2-X applications

Open access to vehicle

Stronger interaction

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive - Motivation

Technology Drivers

**Ethernet**
- High bandwidth
- Communication system is not limiting aspect any more
- Switched network
- Efficient point-to-point communication
- Efficient transfer of long messages

**Processors**
- Switch from microcontroller to processors with external memory (and maybe filesystems)
- Many core processors
- Parallel computing
- „Cheap“ availability of computing power

**Heterogeneous architectures**
- Special purpose processors
AUTOSAR Adaptive - Motivation
Incremental deployment

- A set of applications shall be deployable without generating the full target image

| Communication paths must be established at start-up or run-time |
| Applications must be dynamically initiated and scheduled |
| Security measures must control the access to vehicle network and ECUs |
| Methodology must support comprehensive design of systems consisting of classic and adaptive platform |

Refined by

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive - Motivation
Another Platform for Different Applications

Real time requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>High, in the range of micro-sec</th>
<th>Mid, in the range of milli-sec</th>
<th>Low, in the range of sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety Criticality</td>
<td>High, up to ASIL-D</td>
<td>High, at least ASIL-B</td>
<td>Low, QM</td>
</tr>
<tr>
<td>Computing power</td>
<td>Low, ~ 1000 DMIPS</td>
<td>High, &gt; 20.000 DMIPs</td>
<td>High, ~ 10.000 DMIPs</td>
</tr>
</tbody>
</table>

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive - Characteristics

Application framework
- Support for run-time configuration
- Service-oriented communication
- Partial update

Formats for design data
- Planning of dynamic behavior (e.g. constraints for scheduling and communication).
- Consider automotive specific cooperation scenarios
- Support integration with existing systems (Classic Platform)

Reference architecture
- Reuse existing (non-automotive) standards
- Ease software development
- Support automotive use-cases and protocols
AUTOSAR Adaptive - Comprehensive System Design for Integration of Classic-, Adaptive- and Non-AUTOSAR ECUs

- Expected total number of ECUs: 120
- 5 - 10 domain controllers will run with Adaptive platform

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive - Sample vehicle architecture

authority, OEM, map, … via backend

environment, driver state and vehicle state perception

environment and state model

maneuver- and trajectory planning

trajectory

C2C / C2I

camera, radar, lidar, …
inertial sensors, odometry, GPS, …
safety function

trajectory control

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive – Architectural Overview

Functional Clusters

AUTOSAR Runtime for Adaptive Application

API or Service Interface of a Functional Cluster:
- Programming language specific API for a Functional Cluster as specified in SWS
- The first programming language supported by the Adaptive Platform will be C++

Behavioral specification of Functional Cluster

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview

APIs and services exposed to applications by functional clusters

Services provided via Communication ARA API

Language specific APIs as part of ARA
AUTOSAR Adaptive – Architectural Overview

Functional Clusters

- Ordered and mode-aware startup and tear-down of applications
- Coordination of mode-changes
- Collection of diagnostic event data
- Data exchange with the diagnostic backend
- Provision of standardized diagnostic protocols
- Construction and supervision of service based communication
- Local and remote

Adaptive AUTOSAR Services

- Execution Management
- Software Configuration Management
- Security Management
- Diagnostics

- Persistency
- Hardware Acceleration
- Communications

Adaptive AUTOSAR Foundation

- Operating system
- Platform Health Management
- Logging and Tracing

- Bootloader
- API
- API
- API

Collection and distribution:
- Security related events
- Measurement data

 Provision of isolated runtime environments for applications
 Standardized access to HW

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive – Architectural Overview

Hardware Abstraction

- Adaptive Applications use-cases add several constraints to the HW platform:
  - High Computational Power
  - Huge amount of memory
  - High Speed interfaces
  - Heterogenous Architecture (HW Accelerators)

- Adaptive platform provides an efficient abstraction of the hardware to the application
  - Operating System
  - Persistency Management
  - Hardware Acceleration Management (e.g. GPU)
AUTOSAR Adaptive - Architectural Overview

HW Abstraction: Interfaces

Technical architecture:
- Shows standardized and non-standardized interfaces of the Adaptive platform
- Gives informative example for SW architecture of a platform implementation

Public Interface
Part of the adaptive AUTOSAR API and specified in the SWS.

Protected Interface
Interaction between functional clusters. Not normative, intended to make specification more readable and to support integration of SW into demonstrator.

Private Interface
Interaction between elements within a functional cluster. Not used in specifications.

Network

normative interface
informative interface
private interface

ARA API

Communications
C++14 Language Binding
Dispatching and Discovery
SOME/IP Transport
IPC Transport
TCP/IP
IPC
Ethernet Driver
AUTOSAR Adaptive – Architecture Overview

Hardware Abstraction: Operating System Interface

- POSIX = Portable Operating System Interface
  - Standardized interface between Application and Operating system (e.g. pthread.h semaphore.h)

- There are **four** POSIX Profiles
  - PSE 51: Minimal (~290 APIs)
  - PSE 52: Controller (~230 APIs)
  - PSE 53: Dedicated (~750 APIs)
  - PSE 54: Multi-purpose (~900 APIs)
AUTOSAR Adaptive – Architecture Overview

Adaptive Applications

- are single/multi-thread processes (no longer runnable based as for CP)
- the configuration is done dynamically at run time (no longer static at compile time)
  - manifest files (ARXML format) specifies the configuration parameters.

- Interface with Adaptive AUTOSAR Foundation and Services via ARA API.

- POSIX PSE51 profile to access OS services

- Communication between Application is done via AP Middleware - Communication Manager (PSE51 does not provide IPC interface).

- C++14 as default programming language.

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
Service-oriented Communication

Flexible deployment of applications requires flexible communication allowing maximum flexibility and scalability in processing distribution

- SW components executed on the adaptive platform will use service-oriented communication
- Communication paths can be established at design- and at run-time
- The AUTOSAR Adaptive platform will therefore provide middleware functionality: Communication Management

The **Service Oriented Communication** is a pattern in which application components provide services to other components via the SOME/IP communication protocol

**SOME/IP**: Scalable service-Oriented MiddlewarE over IP (SOME/IP)
AUTOSAR Adaptive - Architectural Overview
Transparent Communication with Dynamic Topology

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive - Architectural Overview
ARA::COM - Service-oriented Communication

C++ Language Binding is based on Proxy / Skeleton Paradigm

Provided Interface
Events
Methods
Fields

SOME/IP-Serialization
E2E Protection

SOME/IP Service Discovery

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
Two code artifacts are generated from AUTOSAR ARXML service description.

- **Service Proxy:**
  *facade:* an instance of a generated class, which provides methods for all functionalities the service provides.

- **Service Skeleton:**
  instance of a generated class which allows to connect the service implementation to the Communication Management transport layer

Note: **ara::com** only defines the API signatures and its behaviour visible to the application developer. The Implementation of those APIs and the underlying middleware transport layer is the responsibility of the AUTOSAR AP vendor.

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive - Foundation Example

- AUTOSAR communication between two Applications can be realized by dynamically configured software platform.

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
Methodology for Adaptive AUTOSAR must
- Support development of mixed vehicle systems built from Classic and Adaptive platforms
- Cover on-board vehicle network only
- Support detailed definition of service interfaces
- Support the definition of configuration parameters necessary for deploying and executing SW
- Not support the integration of off-board systems
- Not be responsible for the compatibility of interfaces

Figure 2.1: Methodology Overview: Overall Structure

“Lifetime deployment of SW”
“Develop mixed vehicle systems” (not for R17-03)
Adaptive AUTOSAR Systems will be developed in subsequent cycles of design and deployment.

Challenges

- A set of Applications shall be deployable without generating the full target image
- Target packages must be specified and prepared at design time based on partial knowledge of deployment

Source:
AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive - Methodology & Manifests
Overview on how adaptive methodology might be implemented.
AUTOSAR Adaptive: Integration of Different Platforms

Source: AUTOSAR_AdaptivePlatformFor_EXP_TechnicalOverview
AUTOSAR Adaptive- Integration of Different Platforms: Machine View

One **Machine** is generally occupied exclusively by one **Software Platform Instance**.
AUTOSAR Adaptive – Deliverables and Demonstrator

• Deliverables:
  
  • AUTOSAR will provide the implementations (“ASR SW”) additionally to specifications
  • The ASR SW is based on existing Standard solutions which are supplemented by specific solutions developed by the AUTOSAR partners
  • The ASR SW can be used as basis for production (AUTOSAR does not grant any warranty nor liability)

• Reference Platform:
  
  • The MinnowBoard Max is a small embedded board with a 64-bit ATOM processor that can easily be used to start developing adaptive AUTOSAR applications on.
  
  • The home page of the project can be found at http://wiki.minnowboard.org
  
  • The Renesas Stout is an ADAS prototype in the R-CAR series provided by Renesas.
  
  • QEMU - machine emulator and virtualizer.
AUTOSAR ADAPTIVE

SUMMARY
## AUTOSAR Adaptive – Architectural Overview
### Summary: Classic Platform vs Adaptive Platform

<table>
<thead>
<tr>
<th>Classic Platform</th>
<th>Adaptive Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same address space for all applications (MPU support for safety)</td>
<td>Each application has its own (virtual) address space (MMU support)</td>
</tr>
<tr>
<td>Optimized for signal-based communication (CAN, FlexRay)</td>
<td>Service-oriented communication</td>
</tr>
<tr>
<td>Based on OSEK</td>
<td>Based on POSIX (PSE51)</td>
</tr>
<tr>
<td>Execution of code directly from ROM</td>
<td>Application is loaded from persistent memory into RAM</td>
</tr>
<tr>
<td>Fixed task configuration</td>
<td>Support of multiple (dynamic) scheduling strategies</td>
</tr>
<tr>
<td>Specification</td>
<td>Specification as binding Standard Code as Demonstrator</td>
</tr>
</tbody>
</table>

Source: AUTOSAR
AUTOSAR Adaptive – Milestones

ASR RELEASE 17.03
- EXECUTION MANAGEMENT
- COMMUNICATION/MIDDLEWARE
- DLT
- DIAGNOSTICS
- OPERATING SYSTEM
- PERSISTENCY
- PLATFORM DEMONSTRATOR

ASR RELEASE 17.10
- SAFE STORAGE OF PERSISTENT DATA
- PERSISTENT DATA ENCRYPTION
- COMMON SYSTEM MODELING APPROACH FOR SERVICE BASED COMMUNICATION CP/AP
- USER AUTHENTICATION & CERTIFICATES
- KEY MANAGEMENT
- SECURE COMMUNICATION
- INTEGRATION OF CAN/CAN-FD
- ADAPTIVE APPLICATIONS ACCESSING SIGNALS VIA SIGNAL-TO-SERVICE-MAPPING
- EXTENDED C++ CODING GUIDELINES
- SUPPORT OF MANIFEST FEATURES

ASR RELEASE 18.03
- PLATFORM HEALTH
- FR, LIN SUPPORT
- CRYPTO HW SUPPORT
- ...

ASR RELEASE 18.10
- CONTAINER SUPPORT
- VEHICLE API
- ...

Source: AUTOSAR
AUTOSAR Adaptive – at NXP

- **SPECIFICATIONS OF ADAPTIVE AUTOSAR**
  - NXP is involved in the following WP/FT
    - WP- A
    - WP-A5
    - WP-X-SEC
    - FT-PD (Passive member)
    - FT-SEC (Passive member)

- **DEVELOPMENT OF OS BSPs (incl. HW ACC) FOR NXP PLATFORMS – Adaptive AUTOSAR Compliant**
  - LINUX BSP
  - Collaboration with major OS vendors (QNX, Integrity, )

- **ADAPTIVE PLATFORM DEMONSTRATOR**
  - Collaboration with NXP integration partners to make the AP Demonstrator running on NXP platforms
AUTOSAR ADAPTIVE

PORTING & RUNNING THE ADAPTIVE DEMONSTRATOR ON NXP BLUEBOX(S32V) PLATFORM
Automotive + Ethernet I/O

- Separated I/O for ASIL-B/ASIL-D domains
- Automotive Ethernet partitioned through switches
- Desktop convenience connectors for convenience Ethernets
- 10G Ethernet for logging purposes
- SFP+ cage for fiber based system links
QorIQ Layerscape: LS2084A

- Deliver more performance per space, weight, area, & power than any other range of competing embedded processors.

**General Purpose Processing**
- 8x ARM® Cortex®-A72 CPUs
  - 64b, up to 2GHz
  - 80,000 DMIPS
  - SIMD w/64 GFLOPS
  - 1 MB L2 cache per cluster w/ECC
  - Hardware Data Prefetch
- 1 MB L3 platform cache w/ECC
- 2x64b DDR4 w/ECC up to 2.1 GT/s

**Accelerated IO Processing**
- 20 Gbps SEC — crypto acceleration
- 10 Gbps Pattern Match/RegEx
- 20 Gbps Data Compression Engine

**Network IO**
- Wire Rate IO Processor:
  - 8x1/10 GbE + 8x1 G
  - XAUI/XFI/KR and SGMII
  - Support for 1588v2, MACSEC

**High Speed Serial Interfaces**
- Up to 4 PCIe controllers, Gen 3
- One controller w/x8 and SR-IOV
**S32V234: Safety Controller, 1st Gen Vision and AD ECU**

### General Purpose Processing
- Two 2x ARM A53 Safe Clusters
  - 64 Bit, 1.0 GHz
  - 2 x 256 kB L2 cache per cluster
  - Neon SIMD
  - ~10000 DMIPS
- 2 x 32b DDR3/LPDDR2 at 533MHz

### High Speed Serial Interfaces
- 1 PCIe controllers
- 1 Dual Channel FlexRay
- 1 Zipwire
- 2 x MIPI CSI2 - 4 lanes 6Gb/s

### Low Speed Serial Interfaces
- 2 CAN –FD
- 4 SPI
- 2 LinFLEX
- 4x Timer
- FlexRay

### Accelerated Processing
- Image Signal Processing
- 2 x APEX2 – Image cognition Processing Open CL
- H.264 Codec and MJPEG decoder
- 3D GPU GC3000 (4 Shader)

### Security
- 1 CSE3 – Flashless

#### Functional SAFETY
- Classic ASIL B capable SoC
- LBIST, MBIST
- Voltage Monitoring, Temperature Monitoring
- Full memory ECC, E2E ECC
- SW Core Self Tests
- SW independent Fault monitoring and reporting
- Safe DMA, CRC processing
- Safe MCAL

## Coherency Fabric
- System RAM
- Memory Controller
- Memory Controller
- FCCU
- CSE3
- PCIe
- 48

## Power Management
- SDHC
- lineflex
- I2C
- GPIO, JTAG
- ADC

## Functional Interfaces
- 2x CSI2 4x
- 2x 16 bit Parallel I/F
- PCIe
- FlexRay
- Zipwire
- ECC
- APEX

## Vision
- Surround
- Fusion

### System RAM
- 4MB
- 32-bit DDR3/LPDDR2
- Memory Controller
- Memory Controller

### Memory Controller
- 256kB Banked L2
- 32KB L1-D
- 32KB L1-I
- 32KB L1-D
- 32KB L1-I
- 32KB L1-D
- 32KB L1-I
AUTOSAR Adaptive – The Platform Demonstrator Environment

Adaptive Demonstrator SDE includes:

- Platform images for hardware boards (MinnowBoard, R-CAR)
- Emulation environment
- Compiler and build tools
- Sample applications
- System Test applications
- Manifest editors, validators and generators
- Utility tools
AUTOSAR Adaptive – Get the Adaptive Platform Demonstrator

- TOOLs/FWKs used: **YOCTO, GIT and Google “repo”**
- The source code for the APD can be obtained from the AUTOSAR Gitlab repo: [https://code.autosar.org/groups/tf-apd](https://code.autosar.org/groups/tf-apd)
- The content of the ADP is split in 4 repositories/projects

**ara-api**: Repository for the ARA-API
- contains all ara related sources and tests

**ara-services**: Repository for ARA-SERVICES
- contains all ara related sources and tests

**yocto-layers**: Repository for yocto recipes.
- contains: all used yocto recipes

**sample-applications**: Sample applications for ARA architecture.
- contains: Sample applications showing the usage of the different APIs
AUTOSAR Adaptive - Get the Adaptive AUTOSAR Repo

- Download the layer “yocto-layers” along with “ara-api” and “sample-applications” folders from the Adaptive Autosar OS repository.
  - curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
  - mkdir ~/ara-project
  - cd ~/ara-project
  - repo init -u git@code.autosar.org:tf-apd/ara-project.git
    (for this to work you must have a ssh key for code.autosar.org)
  - repo sync

Note: write permission in “ara-api” and “sample-applications” are needed (in the process of building an image the sources from “ara-api” and “sample-applications” folders need to be compiled and some files will be generated in those folders)
AUTOSAR Adaptive – Linux specific details of BlueBox

DN SDK 2.0 + meta-bluebox = BlueBox BSP

Yocto Rootfs: DN Yocto 2.0 + meta-bluebox

Linux kernel 4.1.35

u-boot 2016.09

LS2

Linux kernel 4.1.26

u-boot 2016.01

Bluebox

S32V
AUTOSAR Adaptive – Download NXP Bluebox (S32V234) environment

- Download and install the public NXP SDK 2.0 to generate default images for any BlueBox boards (e.g. S32V234).

  NXP SDK 2.0: http://sun.ap.freescale.net/bbsp/QorIQ-SDK-V2.0/QorIQ-SDK-V2.0-SOURCE-20160527-yocto.iso

- Adaptive AUTOSAR Target Machine is **s32v234pcie**; get the meta-bluebox layer:

  ```
  git clone ssh://git@sw-stash.freescale.net/fae/meta-bluebox.git
  ```

- Copy the meta-bluebox layer into the “sources” of the SDK 2.0.

  ```
  cp -r <path to meta-bluebox layer> <path to “sources” folder in the SDK 2.0>
  ```

- Run the “setup-meta-bluebox” script from the meta-bluebox layer with the path to where you installed the SDK 2.0

  ```
  ./setup-meta-bluebox <path to where you installed the NXP SDK 2.0>
  ```
AUTOSAR Adaptive – NXP Bluebox(S32V234) BSP Changes

- The BlueBox BSP is built with different version of toolchain&libs which needs to be aligned with APD toolchain versions
- Enter the “meta-bluebox” layer: “cd ./conf/machine” and add the following lines to the s32v234pcie.conf file

  - PREFERRED_PREFERRED_VERSION_gcc-cross-$(TARGET_ARCH) ?= "5.2%"
  - PREFERRED_VERSION_gcc cross-$(TARGET_ARCH) ?= "5.2%"
  - _VERSION_gcc-canadian-$(TARGET_ARCH) ?= "5.2%"
  - PREFERRED_VERSION_gcc-cross-initial-$(TARGET_ARCH) ?= "5.2%"
  - PREFERRED_VERSION_gcc-source-linaro-4.9 ?= "5.2%"
  - PREFERRED_VERSION_gcc-source-4.9.2 ?= "5.2%"
  - PREFERRED_VERSION_gcc-runtime ?= "5.2%"
  - PREFERRED_VERSION_gcc-sanitizers ?= "5.2%"
  - PREFERRED_VERSION_libgcc ?= "5.2%"
  - PREFERRED_VERSION_libgcc-initial ?= "5.2%"
  - PREFERRED_VERSION_glibc ?= "2.22%"
  - PREFERRED_VERSION_glibc-initial ?= "2.22%"
  - PREFERRED_VERSION_glibc-locale ?= "2.22%"
  - PREFERRED_VERSION_glibc-mtrace ?= "2.22%"
  - PREFERRED_VERSION_glibc-scripts ?= "2.22%"
create a build directory and set the environment for BlueBox

source ./fsl-setup-env -D bluebox -e "meta-bluebox" -m s32v234pcie

Note
- The list of buildable images specific to Adaptive AUTOSAR OS are:
  - core-image-apd-debug
  - core-image-apd-devel
  - core-image-apd-devel-fusion
  - core-image-apd-devel-radar
  - core-image-apd-minimal
  - core-image-apd-minimal-fusion
  - core-image-apd-minimal-radar
  - core-image-apd-minimal-radar-fusion
AUTOSAR Adaptive - Building an image

- copy the yocto-layers, “ara-api” and “sample-applications” folders from “code.autosar.org” repository to “sources” folder of SDK

- add the “meta-ara” layer in the `bblayers.conf` file located in “sources” directory
  
  ```bash
  bitbake-layers add-layer ../sources/yocto-layers/meta-ara
  ```

- build an Adaptive Autosar Image for s32v234pcie with the following line (in your build directory):
  
  ```bash
  bitbake <image_name>
  ```

  the “.sdcard” image will be generated at the following location: ./tmp/deploy/images/s32v234pcie

- copy the image on a sdcard with the “dd” linux command.
  
  ```bash
  dd if=<path to the .sdcard image> of=/dev/<partition of the sdcard inserted> bs=1M && sync
  ```

- insert the sdcard into the s32v234pcie board and the AUTOSAR Adaptive demo will start automatically after boot

  *Note*: the .sdcard format creates an image with all necessary partitions and loads the bootloader, kernel and rootfs to this image..
AUTOSAR Adaptive: example of ADP running on s32v234pcie

- Core-image-apd-devel-fusion run on s32v234pcie:

  ➢ Implements functions like:
    - ✔ Parking Brake Event Received
    - ✔ Service Availability Callback -> Callback executed whenever a change for radar service offers happen
    - ✔ Update Rate Received -> Callback Received when the Field Update Rate is changed.
SECURE CONNECTIONS
FOR A SMATER WORLD