AUTOSAR Software Overview

AMF-AUT-T0022

Rebeca Delgado
Field Applications Engineer
Freescale in Automotive

• Freescale Leadership in Driving Standards
  • First semiconductor supplier to join the AUTOSAR partnership
  • Active member of JASPAR
  • Member of GENIVI
  • Co-founded Open Alliance on Ethernet for Automotive

• Broadest Automotive MCU Product Portfolio
  • Auto-qualified products (8/16/32-bit MCUs & MPUs) span body electronics, powertrain, safety and chassis and driver information systems.

• Customer Relationships
  • Freescale has solid, long-standing customer relationships with nearly every automotive manufacturer and Tier 1 supplier in the world

• Long-term Global Presence
  • Freescale has what it takes to meet the stringent requirements of the global automotive market
Freescale provides software products where in-depth hardware knowledge is crucial – including AUTOSAR MCAL and OS, Core Self Test, and application-specific libraries to address unique hardware features.
# 32-bit Automotive Software Product Overview by Segment

<table>
<thead>
<tr>
<th>Segment</th>
<th>Body</th>
<th>Chassis/Safety</th>
<th>Powertrain</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOSAR 3.x</td>
<td>560xB/C/D, 564xB/C, 5668G</td>
<td>560xP, 560xE, 564xL, 567xK</td>
<td>563xM, 564xA, 567xF</td>
</tr>
<tr>
<td>AUTOSAR 4.0</td>
<td>MCAL</td>
<td>MCAL</td>
<td>MCAL</td>
</tr>
<tr>
<td></td>
<td>OS</td>
<td>OS</td>
<td>OS</td>
</tr>
<tr>
<td>Non-AUTOSAR</td>
<td></td>
<td>Eth Streaming SW</td>
<td></td>
</tr>
<tr>
<td>(Prod Code)</td>
<td></td>
<td>Camera Appl SW</td>
<td></td>
</tr>
<tr>
<td>Non-AUTOSAR</td>
<td></td>
<td>Flash / EE Drv</td>
<td>Flash / EE Drv</td>
</tr>
<tr>
<td>(Demo Code)</td>
<td></td>
<td></td>
<td>eTPU Lib</td>
</tr>
<tr>
<td>Safety</td>
<td></td>
<td>Instruction based Core Self Test (e200 cores)</td>
<td>Core Self Test 60%</td>
</tr>
</tbody>
</table>

Not all shown products are available for all MCUs
AUTOSAR – Global Automotive Software Standard

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

- Hardware and software is widely independent of each other.
- Development can be de-coupled by horizontal layers. Reduces development time and costs.
- Reuse of software enabled at OEM and at suppliers. Enhances quality and efficiency.
BERBS OF THE AUTOSAR DEVELOPMENT PARTNERSHIP
s: December 2011)
AUTOSAR Architecture
AUTOSAR – Global Automotive Software Standard

• **Benefits for car manufacturer**
  - Establish development distribution among suppliers
  - Compete on innovative functions with increased design flexibility
  - **Simplify** software and system integration
  - **Reduce cost** of overall software development

• **Benefits for supplier**
  - Reduce version proliferation
  - **Reuse** software modules across car manufacturers
  - **Increase efficiency** of application development

**Volume of ECUs with AUTOSAR**

• Members represent about 80% of worldwide car production.
• In 2016 approx 25% of ECUs will be based on AUTOSAR.

Source: AUTOSAR

Source: AUTOSAR Development Partnership
Reusability of BSW Modules and SW Components

Scenario A
The supplier provides the ECU to a different OEM.

Scenario B
Integration of features, delivered from different sources.

Scenario C
The hardware changes.

Fully AUTOSAR compliant ECU

Source: AUTOSAR
Freescale AUTOSAR Products

- Freescale offers cost effective **production-ready** MCAL and OS
- What the customer gets:
  - From Freescale (shaded blue below): MCAL (source code), OS (source code) and supporting Configuration Tool (executable).
  - From Partners (Elektrobit, Vector, KPIT, etc.) – The rest of AUTOSAR basic software as needed. Partner does integration (Freescale IP + Partner IP + Customer IP)
AUTOSAR MCAL Product

- MCAL drivers for each MCU peripheral, compliant to AUTOSAR 2.1 / 3.x / 4.0
- AUTOSAR 2.1/3.0 MCAL: excl. RamTst module
- AUTOSAR 4.0 MCAL: excl. RamTst, CoreTst, FlashTst modules
- All components configurable in any AUTOSAR-compliant configuration tool
- Configuration Tool EB tresos Studio™ and plug-ins are part of the product
AUTOSAR Operating System

- Configurable in AUTOSAR configuration tool
- Available in Scalability Classes 1, 2, 3, 4 to fit the needs of different applications
  - SC1 – deterministic RTOS baseline (tasks, events, counters, alarms, messages)
  - SC2 – timing based task determinism (low-latency, precise timing for periodic tasks)
  - SC3 – protected memory (MMU/MPU) for tasks avoids memory collisions for safety systems
  - SC4 – timing and memory protected tasks, utilizes the full capabilities of the silicon for secure and protected RTOS designed specifically for the automobile.
- Availability of SC2,3,4 depends on MCU family / presence of MPU
AUTOSAR Documents

• Released AUTOSAR documents can be found at [www.autosar.org](http://www.autosar.org)
• 2 documents exist for each BSW module:
  - SRS: Software requirement specification
  - SWS: Software Specification
• The **SRS** describes requirements, that must be fulfilled by a Basic Software Module (BSW).
  - Chapters of SRS documents
    - Chapter 1 defines the area of application of the BSW
    - Chapter 2 defines the structure of the document
    - Chapter 3 defines the acronyms used in that document
    - Chapter 4 is the main chapter.
    - Contains a brief description of the BSW functional overview and continues with the requirements.
AUTOSAR Documents

- The **SWS** (Software Specification) contains the most detailed information for each Basic Software Module.
- Each SWS document is structured as follows:
  - Chapter 1 – introduction and brief overview of functional behavior of the BSW
  - Chapter 2 – used acronyms
  - Chapter 3 – referenced documents
  - Chapter 4 – restrictions and applicability for the automotive domain
  - Chapter 5 – relation to other BSW modules and the file structure of the BSW
  - Chapter 6 – requirement matrix containing links to requirements from the related SRS document
  - Chapter 7 & 8 – contain the description of the:
    - functional behavior of the BSW
    - applications programming interface (API)
  - Chapter 9 – Message sequence charts are used to describe the sequential behavior of a SWC in relation to other SWCs
  - Chapter 10 – possibilities of configuration are defined
  - Chapter 11 – Release changes are documented
BSW Configuration Classes

• Pre-compile configuration
  - Configuration parameters can not be changed after compilation
  - Example: Mapping of microcontroller pins to signals

• Link-time configuration
  - Configuration is determined by linker scripts
  - Configuration parameters can not be changed after link process
  - Purpose: provides capability to deliver object code to the integrator

• Post-build configuration
  - Post-build time loadable
    ▪ Configuration parameters can be changed after build process without complete re-flash of ECU
  - Post-build time selectable
    ▪ Configuration parameter set is selected from multiple configuration sets during boot time
    ▪ All possible configuration sets need to be included at compile time
  - Configuration parameters are stored at a known memory location
  - Post-build configuration class BSW modules might also contain pre-compile or link-time parameters (not all parameters have to be post-build)
  - Purpose: use one software package in different vehicles
AUTOSAR and ISO26262
ISO 26262 : Automotive Norm on Functional Safety

- ISO 26262 is a Functional Safety standard applicable to automotive systems. This norm is an adaptation of the Functional Safety standard IEC 61508.
- ISO 26262 is applied to ensure that electronic systems in automotive applications are completely safe. Thus it covers functional safety aspects of the entire development process, including requirements specification, design, implementation, integration, verification, validation, and configuration.

Figure 1: Functional Safety Standards Details

<table>
<thead>
<tr>
<th>Standards Defined</th>
<th>Level Comparison</th>
<th>Failure Measures</th>
<th>New Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC 61508</td>
<td>No direct correlation for SIL and ASIL levels</td>
<td>IEC 61508</td>
<td>- Information is more structured in ISO 26262</td>
</tr>
<tr>
<td>Integrity levels</td>
<td>SIL 1, SIL 2,</td>
<td>SIL</td>
<td>- Concept of safety culture exists in ISO 26262</td>
</tr>
<tr>
<td></td>
<td>SIL 3, SIL 4</td>
<td>Random HWFR target</td>
<td>- Terminology is well defined in ISO 26262 (safety plan, safety case, work products, confirmation measure, etc.)</td>
</tr>
<tr>
<td>Publication date</td>
<td>More than 10</td>
<td>4</td>
<td>- Roles and responsibilities are better defined in ISO 26262. (PM, safety manager)</td>
</tr>
<tr>
<td></td>
<td>years ago</td>
<td>≥10⁻⁸ to &lt;10⁻⁸</td>
<td></td>
</tr>
<tr>
<td>ISO 26262</td>
<td>Automotive industry standard, adaptation of IEC 61508 for electronic systems in road vehicles</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Integrity levels</td>
<td>ASIL, ASILB,</td>
<td>≥10⁻⁴ to &lt;10⁻³</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASILC, ASILD</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Publication date</td>
<td>Target end</td>
<td>≥10⁻¹ to &lt;10⁻⁰</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2011</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Simplifies the process of system compliance, with solutions designed to address the requirements of automotive and industrial functional safety standards.

Reduces the time and complexity required to develop safety systems that comply with ISO 26262 and IEC 61508 standards.

Supports the most stringent Safety Integrity Levels (SILs), enabling designers to build with confidence.

Zero defect methodology from design to manufacturing to help ensure our products meet the stringent demands of safety applications.
Automotive Software Development Process Evolution

Enabling ISO 26262 compliance

Products in development

2008
Customer Assessed
Automotive SPICE
Level 1

2009
Customer Assessed
Automotive SPICE
Level 2

2010
Customer Assessed
Automotive SPICE
Level 3

2011
Automotive SPICE
Level 3 + ISO26262
(Prototypes)

2012
Automotive SPICE
Level 3 + ISO 26262
### SPICE Capability Level 3 in all Process Areas (HIS-Scope) – since 2010

<table>
<thead>
<tr>
<th>ID</th>
<th>Process Name</th>
<th>PA 1.1</th>
<th>PA 2.1</th>
<th>PA 2.2</th>
<th>PA 3.1</th>
<th>PA 3.2</th>
<th>Cap. Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAN.3</td>
<td>Project Management</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>L</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>ENG.4</td>
<td>Analysis</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>ENG.5</td>
<td>Software Design</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>ENG.6</td>
<td>Software Construction</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>ENG.7</td>
<td>Software Integration Test</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>L</td>
<td>L</td>
<td>3</td>
</tr>
<tr>
<td>SUP.1</td>
<td>Quality Assurance</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>SUP.8</td>
<td>Configuration Management</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>SUP.9</td>
<td>Management</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>SUP.10</td>
<td>Change Request Management</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>

F: Fully achieved  
L: Largely achieved
Automotive Software for ISO 26262

- Support efficient achievement of system-level safety goals up to ASIL-D
  - Safety with minimized performance degradation
  - Safety simplified for integrators
  - Cross-platform consistent architecture
- Support achievement of hardware architectural metrics up to ASIL-D
Summary

- Freescale’s mission is to be the **benchmark provider for silicon and software** that enables our customers to build scalable platforms for automotive body, powertrain, safety and chassis, and driver information systems.

- Since several years Freescale successfully delivers production ready AUTOSAR MCAL and OS software. Freescale now expands its software roadmap to support our SafeAssure program, as well as Automotive Ethernet, Motor Control, and Radio solutions.

**Silicon + Software + Services + Support**
AUTOSAR Configuration Methodology / Tool
Basic Software Configuration Process

Vehicle- or Network-level Design Tool

AUTOSAR BSW Configuration Tool

ECU Parameter Definitions (XML)

ECU Configuration Description (XML)

RTE Generator

OS Generator

Communication Services Generator

MCAL Generators

.h

.c

.h

.c

.h

.c

.h

.c
Static configuration

- Static configuration allows to change code behaviour dependent on configuration parameters
- Functionality can be designed to be statically defined instead of definition during runtime, e.g. Creating tasks in configuration instead of during runtime with a function
- lower memory footprint, faster execution, scalable/tailored to the application needs
EB tresos Studio

- EB tresos Studio is an easy-to-use tool for ECU standard software configuration, validation and code generation
- Full support for the AUTOSAR standard
- Full support for the Freescale AUTOSAR software and the EB tresos AutoCore

Source: Elektrobit
Freescale AUTOSAR Integration Partners receive Freescale MCAL and OS releases for pre-integration into their proprietary AUTOSAR BSW products.
AUTOSAR BSW Configuration Tool
Example: tresos® ECU

• Graphical representation of ECU configuration description (ECD)
• Import/export of ECD
• Easy configuration of AUTOSAR BSW using pre-compile methodology
XDM is a proprietary format (EB) providing enhanced usability features during configuration with EB tresos Studio.

Source: Elektrobit
EPD is the standard AUTOSAR format. This allows the Freescale Autosar software to be used with any other Autosar GCE tool.
Parameter Description Files – Beyond MCAL

Legend
- AUTOSAR Files
- Elektrobit Files
- Generated Files

Source: Elektrobit
Errors & Warnings

User corrects the problem

Link to error or warning

Interactive problem resolution

Source: Elektrobit
Parameter Definition

... and its corresponding entry in the description file (*.EPD)

Source: Elektrobit
Software Release Framework
AUTOSAR Software Release Framework - Overview

• Early Access Release – EAR (Lead customers only)
  - Includes a subset of MCAL Drivers with limited testing coverage
  - No Quality Documentation

• BETA Release (all customers)
  - Includes all MCAL Drivers (Feature Complete) fully verified and documented
  - Includes Integration Testing
  - Complete Quality Package

• Release to Market Candidate – RTMC (all customers)
  - Beta criteria +
  - 100% Decision Coverage for one configuration
  - No open S1 and S2 defects

• Customer Compiler Tests (no new release)
  - Specific for the provided customer compiler version and settings
  - Production Approval and starting point for frozen branch support - if required by customer
<table>
<thead>
<tr>
<th></th>
<th>EAR</th>
<th>Beta</th>
<th>RTM-C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content</strong></td>
<td>Product scope</td>
<td>subset of MCAL drivers</td>
<td>full set of MCAL drivers</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>Technical</td>
<td>subset of user documentation</td>
<td>full user documentation</td>
</tr>
<tr>
<td></td>
<td>Quality docs</td>
<td>no</td>
<td>Complete quality package</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td>Used HW</td>
<td>first samples</td>
<td>unqualified samples</td>
</tr>
<tr>
<td></td>
<td>Test coverage</td>
<td>limited test coverage</td>
<td>all drivers fully verified, 100% of tests</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successfully executed</td>
</tr>
<tr>
<td></td>
<td>Decision coverage</td>
<td>no</td>
<td>90% DC for one configuration</td>
</tr>
<tr>
<td></td>
<td>Extended testing</td>
<td>no</td>
<td>includes integration testing</td>
</tr>
<tr>
<td><strong>Feature coverage</strong></td>
<td></td>
<td>partial feature coverage</td>
<td>100% coverage of defined features</td>
</tr>
<tr>
<td><strong>Release criteria</strong></td>
<td>Error Status</td>
<td>No open S1 defects</td>
<td>No open S1 &amp; S2 defects</td>
</tr>
</tbody>
</table>

1) During Beta and RTMC release work, tests are added:
   - To cover additional features / functionality requested after BETA release
   - As test suite is continously extended and improved over time
Microcontroller Abstraction Layer
package name schema defined by Elektrobit

Module_TS_TxDyMzIaRb

X = Target (11 – Freescale S12X)
Y = Derivate (3 – S12XEP100)
Z = Module Major Version
A = Module Minor Version
B = Reserved
Freescale Implementation Topics
3 Layer Architecture

- 3 Layer Architecture
  - AUTOSAR API
  - LLD: low level, IP independent
  - IP dependent

Gpt_StartTimer(…)
/* error reporting and detection */
MCAL
Port and Dio Modules
- Initialization of all pins and ports of the Mcu
- Reinitialization with alternate configurations at runtime possible
- Reconfiguration of pins at runtime

**Port**
- Port Pin Function Assignment (GPIO, Adc, SPI, PWM, ...)
  - PadSelection implicitly via HW assignment
- PortPin is the only structural element

**Dio**
- No initialization (done by Port)
- Provides APIs to read and write GPIO ports/pins
- Requires an initialized Port module
  - pins/ports need to be initialized via Port module
  - no formal connection between Port and Dio Ecu Configurations
- API synchronous and unbuffered
- Consistent read and write services (interruptible read-modify-write not allowed)

**Structural Elements:**
- Channel (single pin)
- ChannelGroup (adjacent pins in the same port)
- Port (aggregates Channels and ChannelGroups)

<table>
<thead>
<tr>
<th>Driver:</th>
<th>Name for a Port Pin:</th>
<th>Name for Subset of Adjacent pins on one port</th>
<th>Name for a whole port</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIO Driver</td>
<td>Channel</td>
<td>Channel Group</td>
<td>Port</td>
</tr>
<tr>
<td>PORT Driver:</td>
<td>Port pin</td>
<td>--</td>
<td>Port</td>
</tr>
</tbody>
</table>
Port/Dio Module
Functional Overview

Port

- Initialization of all pins and ports of the Mcu
- Reinitialization with alternate configurations at runtime possible
- Reconfiguration of pins at runtime
- Port Pin Function Assignment (GPIO, Adc, SPI, PWM, ...)
  - PadSelection implicitly via HW assignment
- PortPin is the only structural element

Dio

- No initialization (done be Port)
- Provides APIs to read and write GPIO ports/pins
- Requires an initialized Port module
  - pins/ports need to be initialized via Port module
  - no formal connection between Port and Dio Ecu Configurations
- API synchronous and unbuffered
- Consistent read and write services (interruptible read-modify-write not allowed)
- Structural Elements:
  - Channel (single pin)
  - ChannelGroup (adjacent pins in the same port)
  - Port (aggregates Channels and ChannelGroups)
Port/Dio Modules
Freescale Implementation

Port Access:
- Port_Init(...)
- Port_SetPinDirection(...)
- Port_RefreshPinDirection(...)
- Port_SetPinMode(...)

Dio Write Accesses:
- Dio_WriteChannel
- Dio_WritePort
- Dio_WriteChannelGroup

Dio Read Accesses:
- Dio_ReadChannel
- Dio_ReadPort
- Dio_ReadChannelGroup
Port Module
Pin Configuration

typedef struct {
    uint8 SIUPin; /* Pin Defined on Part SIULx */
    uint16 PCR;  /* Pad Control Register */
    sint8 PDO;   /* Pad Data Output */
    boolean DC; /* Direction changeable */
} Port_LLD_PinConfigType;

Pad Configuration Register PCR

SMC – Safe Mode Control
APC – Analog Pad Control
PA – Output Pad Assignment
OBE – Output Buffer Enable
IBE – Input Buffer Enable
ODE – Open Drain Enable
SRC – Slew Rate Control
WPE – Weak Pull Up/Down Enable
WPS – Weak Pull Up/Down Select
Port Module
API Functions

void Port_Init(
    const Port_ConfigType *ConfigPtr
)

- Initializes all Pins and Padselection Registers
- Reconfiguration at runtime possible
- Should be called also after reset