This tutorial presents an overview of the features in AUTOSAR that help provide support for engineering automotive safety-related systems.

1. Background on AUTOSAR

AUTOSAR is an automotive industry initiative that develops and maintains an open-standard software architecture [1]. A key aspect of this architecture is a design abstraction called the Virtual Function Bus (VFB), shown in Figure 1, that enables software systems to be designed without reference to target hardware and allows hardware allocation decisions to be deferred until later in the development process.

![VFB view](image)

**Figure 1:** AUTOSAR Virtual Function Bus

Software components (SWCs) are the main architectural unit for structuring applications in AUTOSAR systems. SWCs communicate with the other components through a well-defined interface (an AUTOSAR interface) that is typed and statically defined. AUTOSAR defines two communication paradigms:

1. sender-receiver (message passing) communication; and
2. client-server (function invocation) communication.

Internally, SWCs provide one or more threads of control called “runnable entities” that are responsible for responding for handling communication and providing the functionality of software component. The functionality of the VFB is provided at run-time by AUTOSAR’s Run-Time Environment (RTE). The RTE provides the encapsulation of
the communication mechanisms defined by the VFB and defines the ways in which runnable entities can interact.

AUTOSAR also defines a common computing platform called the basic software (BSW). The BSW provides the scheduling, communication, memory management and HW abstraction infrastructure to support applications.

Figure 2 shows the how the pieces of the AUTOSAR architecture fit together.

![Figure 2: AUTOSAR Architecture: Application Layer, RTE, Basic Software & Hardware](image)

2. Background on Safety Engineering

Safety-related systems are those that may result in physical harm if they fail to perform their intended function. Harm occurs when the system fails to protect against a hazard, or when the system itself fails, or both. The first step of any safety engineering process is hazard analysis: identifying which hazards are associated with the system.

Of course, not all hazards are equal - they differ in severity, likelihood of occurrence, duration of exposure and the likelihood of the hazard leading to an accident. This means that some hazards present a greater risk of harm than others. The second step in a safety engineering process is therefore risk assessment: determining the level of risk associated each hazard.
Given adequate knowledge of the hazards and risks in the system, the third step in the safety engineering process is to reduce overall risk to a tolerable level by minimizing or otherwise managing system failures that contribute to hazards. Two complimentary measures help to achieve this:

2. Product measures: engineering the product to maximise the efficacy of process measures (e.g. modular architecture, hardware abstraction); to prevent failures where possible (e.g. defensive programming, partitioning); and to ensure that the random failures can be handled safely (e.g. self-tests, checksums).

3. Scope of the Tutorial

At first glance there appears to be a dichotomy between safety engineering as a system driven activity and using an a priori developed architecture like AUTOSAR which has not been designed with the safety of a specific system in mind.

However, many systems have similar requirements on the general type of product measures that are required. AUTOSAR addresses these in two ways by:

1. Defining a clean, modular, architectural concept with mechanisms from which safety mechanisms can be built, supported by the RTE, which allows users to de-compose, structure and reason about the system being built.
2. Assuming a common set of failure modes (and assuming possible system architectural choices) and providing features in the basic software to address them.

This tutorial presents an overview of the features in AUTOSAR that can help in the development of safety-related systems and covers:

- Communication integrity
  - Both within an ECU (via the RTE) and between ECUs (via the communication stack)
  - Protecting against message corruption, both with packets of data and messages in packets
  - Deadlines, timeouts and re-transmission

- Memory integrity
  - Memory tests, checksums, encryption
• Error handling
  o Global error reporting and management
  o Diagnostics

• Defensive programming
  o Macro scale: Partitioning for error containment in both temporal and spatial domains
  o Micro scale: Numerical protection (range limits, saturation, preventing over-/under-flow and division by zero)

• Computation integrity
  o Self-tests
  o Liveness guarantees (watchdogs)
  o Check-pointing (program flow monitoring)
  o Redundancy

Understanding what features are provided by AUTOSAR can help to inform design decisions when designing a system-specific safety concept. Knowing what set of capabilities are already provided by AUTOSAR (and are therefore accepted by the automotive community) avoids “re-designing the wheel” and can also help in the construction of a “proven through use”-like safety arguments.

While this tutorial is about AUTOSAR and safety, there are clear parallels between engineering systems for safety and engineering systems for security (swapping “threat” for “hazard” and “harm” for “loss” in the Section 2 gives some indication how close). In both domains, the process and product measures that are undertaken and what must be demonstrated in the safety case or security conformance claim are remarkably similar [2]. There are issues of prevention, enforcement, detection and diagnosis in both domains. Aspects of AUTOSAR that have particular relevance to security (for example preventing unauthorized read access to data) are mentioned where required.

4. References