Session 21

AUTOSAR Methodology Overview general

The AUTOSAR (Automotive Open System Architecture) methodology provides a standardized approach to developing automotive software, enabling modularity, reusability, and portability across different Electronic Control Units (ECUs) and vehicle platforms. It focuses on defining a common architecture and standardized interfaces for software components, facilitating collaboration between different automotive manufacturers and suppliers.

Here's a breakdown of the key aspects of the AUTOSAR methodology:

1. Core Principles:

- **Standardization:**

  AUTOSAR defines a common architecture and standardized interfaces for automotive software, promoting interoperability and reusability.

- **Modularity:**

  Software components are designed as independent modules, allowing for easier development, integration, and updates.

- **Abstraction:**

  AUTOSAR abstracts the underlying hardware and communication infrastructure, allowing developers to focus on functional aspects.

- **Reusability:**

  Standardized components and interfaces enable the reuse of software across different vehicle models and projects.

- **Portability:**
  AUTOSAR facilitates the portability of software components across different ECUs and platforms.

2. Methodology Overview:

- **System Configuration Input:**

  The process starts with defining the overall system architecture, including software components, hardware resources, and system constraints.

- **ECU Configuration:**

Software components are mapped to specific ECUs, taking into account resource availability and timing requirements.

- **Executable Generation:**

  The final step involves generating executable code for each ECU based on the configured software components and their mapping.

- **Software Component Development:**

  This involves the development of individual software components based on AUTOSAR specifications.

- **ECU Configuration:**
  This involves configuring each ECU with the necessary software components, communication stacks, and other functionalities.

3. Key Aspects of the Methodology:

- **Software Component Model:**

  AUTOSAR defines a standardized way to describe and model software components, including their interfaces and behavior.

- **Communication Management:**

  AUTOSAR provides a standardized communication stack for inter-ECU communication, enabling seamless data exchange.

- **Operating System (OS):**

  AUTOSAR specifies a real-time operating system (OS) for ECUs, ensuring predictable and timely execution of software components.

- **Basic Software (BSW):**
  AUTOSAR defines a set of basic software modules that provide common functionalities for ECUs, such as communication, memory management, and diagnostics.

4. Benefits of AUTOSAR:

- **Reduced Development Costs:**

  Standardization and reusability can significantly reduce development time and costs.

- **Improved Quality:**

  Standardized components and interfaces can lead to higher quality and reliability.

- **Faster Time to Market:**

  Faster development cycles can enable faster time to market for new vehicle models.

- **Increased Flexibility:**

  Modularity and portability allow for easier adaptation to changing requirements and new technologies.
- **Enhanced Collaboration:**
  AUTOSAR promotes collaboration between different automotive manufacturers and suppliers.

Sample Pseudocode

```
#include <stdint.h>


#define ENGINE_TEMP_THRESHOLD 95  // degrees Celsius


// AUTOSAR RTE-provided API stubs (normally generated)
extern uint16_t Rte_Read_EngineTempSensor_Value(void);  // S/R port read
extern void Rte_Call_WarningLamp_TurnOn(void);          // C/S port call
extern void Rte_Call_WarningLamp_TurnOff(void);         // C/S port call


void EngineTempMonitorRunnable(void) {
    uint16_t engineTemp = Rte_Read_EngineTempSensor_Value();


    if(engineTemp > ENGINE_TEMP_THRESHOLD) {
        Rte_Call_WarningLamp_TurnOn();
    } else {
        Rte_Call_WarningLamp_TurnOff();
    }
}
```