Session 17 : RTE layer in AUTOSAR

The RTE Layer in AUTOSAR

The **Runtime Environment (RTE)** is a generated layer of the AUTOSAR Basic Software (BSW). It acts as the intermediary between the application software components (SWCs) and the underlying Basic Software (BSW) and even between different SWCs.

Key Roles and Characteristics of the RTE:

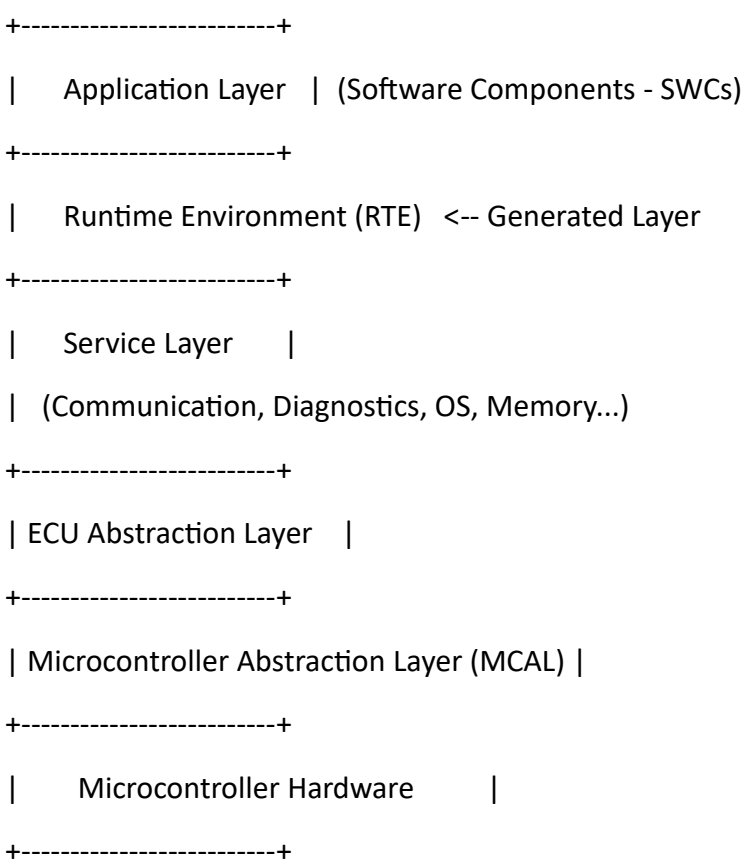
1. **Abstraction of Communication:**
    - **Inter-SWC Communication:** The RTE handles all communication between different Software Components, whether they reside on the same ECU (intra-ECU communication) or on different ECUs connected via the vehicle network (inter-ECU communication).
    - **Port-Based Communication:** SWCs don't directly "know" or "call" each other. Instead, they define **Ports** (e.g., Sender Ports, Receiver Ports, Client Ports, Server Ports). The RTE implements the connections between these ports. This decouples SWCs, making them highly reusable.
    - **Communication Methods:** The RTE supports various communication mechanisms:
        - **Sender-Receiver (S/R) Communication:** For sending data (e.g., sensor values). This is typically asynchronous.
        - **Client-Server (C/S) Communication:** For requesting services and getting responses (e.g., diagnostic services, memory access). This is typically synchronous.
        - **Mode Management:** For coordinating different operational modes of the system or individual SWCs.
        - **Parameter Access:** For reading or writing global parameters.
2. **Hardware Independence for Application Software:**
    - The most significant role of the RTE is to ensure that the application Software Components are **hardware-independent**.

- SWCs interact *only* with the RTE through well-defined AUTOSAR interfaces (APIs). They don't touch microcontroller registers, low-level drivers, or communication bus protocols (CAN, LIN, Ethernet).
- This abstraction allows SWCs to be reused across different ECU hardware platforms and even across different vehicle projects without modification, as long as the RTE interface remains consistent.

3. **Scheduling of Runnables:**
    - Application logic within SWCs is implemented in **Runnables**. A Runnable is a piece of C code that performs a specific function.
    - The RTE is responsible for orchestrating the execution of these Runnables. It maps Runnables to tasks in the AUTOSAR Operating System (OS).
    - The RTE ensures that Runnables are executed at the correct times (e.g., periodically, on data reception, on a specific event).
4. **Integration of Basic Software (BSW) Services:**
    - The RTE provides the interface between the SWCs and the complex BSW modules (e.g., Communication Stack, Diagnostic Manager, Memory Manager, Operating System).
    - When an SWC needs to send a CAN message, it calls an RTE API (e.g., Rte_Write_DataElement). The RTE then translates this into calls to the relevant ComStack modules.
    - When an SWC needs to report a diagnostic event, it calls an RTE API, and the RTE interfaces with the Diagnostic Manager.
5. **Role in the Development Workflow:**
    - The RTE is **not hand-coded**. It is **automatically generated** by AUTOSAR configuration tools (e.g., Vector DaVinci Configurator, Elektrobit EB tresos Studio, dSPACE SystemDesk).
    - This generation process is based on XML description files (ARXML) that define the system's software components, their interfaces (ports), and the connections between them.

- The generated RTE code typically includes header files (e.g., Rte_ComponentName.h) and source files that contain the actual implementation of the communication mechanisms and Runnable scheduling.

RTE in the AUTOSAR Layered Architecture:



Benefits of the RTE:

- **Decoupling**
- **Reusability**
- **Modularity**
- **Scalability**
- **Testability**
- **Automated Code Generation**

Example of an RTE API Call from an SWC:

Let's say a "Wheel Speed Sensor" SWC sends wheel speed data, and an "ABS Control" SWC receives it.