Documentary of Unit 3_ Create General Project in the AUTOSAR

1. AUTOSAR Basics Overview

Software Components (SWCs):

- ☐ SWC (Software Component) is the basic unit of an AUTOSAR system. These components contain the logic and functionalities that are crucial to the system.
- ☐ Example SWCs:
- Application SWC: A component like the Counter that generates numbers.
- Sensor SWC: A component that reads sensor data and sends it to another SWC.
- Actuator SWC: A component that controls actuators based on data received from other components.
-

Software:

- dSpace SystemDesk 5.6 for the System Design of the AUTOSAR application
- dSpace VEOS

Ports:

- **Provided Ports (PPorts):** These ports are used by SWCs to send data to other components.
- **Required Ports (RPorts)**: These ports are used by SWCs to receive data from other components.

Interfaces:

- Interfaces define the data types exchanged between ports.
  - **Sender/Receiver Interface** is the most commonly used in this practical. This defines a producer-consumer relationship where one SWC (the producer) sends data to another SWC (the consumer).

Virtual Functional Bus (VFB):

- VFB facilitates the communication between components. It abstracts the actual communication methods, like CAN or Ethernet.
- It is implemented by the Runtime Environment (RTE), which handles communication without hardware-specific details.

…………………………………………………………………………………………………

2. Practical Tasks Breakdown

Practical Task 1: Creating an AUTOSAR Project

1. **Starting SystemDesk:**

   a. **SystemDesk** is used to design and configure the AUTOSAR system. You begin by opening SystemDesk via the shortcut on your computer.
   b. **SystemDesk will be used to define the system structure**, create software components, define their interactions, and configure the RTE.
2. **Creating a New Project:**
   a. Go to File -> New in SystemDesk.
   b. Name the project Tutorial and **sav**e it in the **"Unit3"** folder on the Desktop. This ensures you have a proper file structure.
3. **Preparing the Project Structure:**
   a. Create three packages (folders) in the Project Manager:
      i. Right click the "Tutorial"
      ii. New Package
      iii. (Three Sub folder)
         1. Interfaces: Stores all the communication interfaces.
         2. SWCs: Stores all software components.
         3. Overview: Contains architecture and composition diagrams that show the system structure.
4. **Importing Data Types:**
   a. Import AUTOSAR standard data types using the ".arxml" file. This file defines platform-independent types, like uint8, int16, etc., so that the software can run on different hardware platforms. (C:\Program File\dSPACE SystemDesk 5.6\Templates\AUTOSAR_Platform.arxml)
      i. AUTOSAR_PLATFORM
         1. BasicTypes
         2. CompuMethods
         3. DataConstrs
         4. ImplementationDataTypes

      ☑ ~~Import Autosar engineering objects~~

   ……………………………………………………………………………………

Practical Task 2: Defining a New Software Component (SWC)

1. **Create the Counter SWC Under Tutorial>Tutorial>SWCs:**
   - In **SWC**s, right-click
     - and select **"New"**
     - "**Application SW Component** " Type.
     - Name the component **"Counter"** because it generates numbers.(This SWC will provide an output port to send numbers to other components.)

2. **Creating PortsTutorial>Tutorial> SWCs >(Counter)**
   - Right-click on **"Counter"** and
   - select **"New"**
   - **"Provided Port Prototype"**.
   - Name it "**Number".**
   - (This port will send data out, so it's a Provided Port (PPort)).

3. **Creating Interfaces under Tutorial>Tutorial"> Interfaces :**
    - Right-click on the I**nterfaces** folder,
        - select **New**
        - **Sender/Receiver Interface.**
        - Name it **"IfSingleNumber"**.(This interface will hold the data that will be sent over the Number port).
        - **Inside the interface**, define a **Data Element** called "**Value"** with data type "**uint8"**.

4. **Assign Interface to Port:**
    - Drag the **"IfSingleNumber"** interface
        - and assign it to the "**Number"** port. (I can find it as a **Tutorial>Tutorial>**SWC > Counter > number). (This step ensures that the Counter SWC will send uint8 values through the Number port.)

5. **Add Internal Behavior Tutorial>Tutorial>SWCs>Counter:**
    - Right-click on the **"Counter"**SWC,
        - select New
        - -> S**WC Internal Behavior**,
        - and name it **IB_Counter**.
    - This internal behavior will contain the logic that will execute when the component runs.

6. Add Runnables **Tutorial>Tutorial>SWCs>Counter**:
    - Right-click on IB_Counter,
        - select New
        - -> Runnable Entity,
        - and name it "**GenerateNextNumber"**.
        - (This function will execute every 500ms to generate a new number and send it via the Number port.)
            - Data Access
            - Select element
                - Port element
                - Number
                - Value
                - Okay
                -
7. Create an RTE Event **Tutorial>Tutorial>SWCs>Counter>IB_counter**::
        - For the GenerateNextNumber runnable, add a Timing Event with a period of 0.5 seconds (500ms) so the runnable runs at that interval.
8. Add an Implementation Placeholder **Tutorial>Tutorial>SWCs>Counter>**:
    - Right-click on IB_Counter
        - and select New
        - -> SWC Implementation, naming it IMPL_Counter.
        - This placeholder will later hold the actua**l C code.**

…………………………………………………………………………

Practical Task 3:3. Completing the Application Design

3.3.1 Add a Second SWC (DataLogger) **Tutorial>Tutorial>SWCs>**:

- .Create a second SWC called **DataLogger**.
- This SWC will receive the numbers from Counter and log them.
- It will have a Required Port called **DisplayValu**e to receive data.

3.3.2 Composing the ApplicationTutorial>Tutorial>Overview>:

- Right-click on **Overview**,
    - select **New**
    - **Composition SW Component** Type,
    - and name it "**MainComposition"**
        - Inside MainComposition,
        - create a "**new "**
        - **Empty Composition Diagram**.
        - Name "**MainCompositionDiagram"**
        - (double click)
        - Drag Counter and DataLogger SWCs to the diagram and connect their ports. Connect Number from Counter to DisplayValue in DataLogger.

3.3.3 Providing Template Source Code Files:

- Open Visual Studio Code
- Create Folder name"Runnables"
- and in Runnables > create empty C files
- named Counter.c
- and **DataLogger.c.** These will hold the code for the runnables you will implement later.

Code Example for All SWCs

Let's provide the final integrated code for each SWC.

Counter.c

```
#include <Rte_Counter.h>
uint8 count - 0;
void Counter_GenerateNextNumber() {
    count++;
    Rte_Write_GenerateNextNumber_Number_value(count);
}
```

DataLogger.c

```
#include <Rte_DataLogger.h>
#include <Sab.h>
void datalogger_Log() {
 uint8 value = Rte_IRead_Log_DisplayValue_Value();
 Sab_|SubmitInfo("bcd: %d",Value);
}
```

……………………………………………………………

Adder.c

```
#include <Rte_Adder.h>

void Adder_Compute() {
  /*uint8 Value =
Rte_IRead_[RunnableName]_[PortName]_[DataElementName]();*/

    uint8 a = Rte_IRead_Compute_InputA_Value();
    uint8 b = Rte_IRead_Compute_InputB_Value();
    uint8 sum = a + b;
/*Rte_IWrite_[RunnableName]_[PortName]_[DataElementName]([Value to write]);*/

    Rte_Write_Compute_OutputValue_Value(sum);
}
```

……………………………………………………………………………………

Practical Task 4: Generating the RTE

4.4.1 Create a Virtual ECU:

- In SystemDesk, go to the **Home** tab
- and click Create **Classic V ECU**.
- This generates the **Runtime Environment** (RTE) and **Basic Software** (BSW) for your ECU.

4.4.2. Validate and Build:

- Build the **virtual ECU** to ensure everything is set up correctly.
- After building, the RTE is generated, and you are ready to implement the logic of each SWC.

Practical Task 5: Implementing SWC Functionality

- Implement C Functions:
- Now that the RTE is generated, you can implement the logic for the **Counter and DataLogger SWCs.**
- In Counter.c, implement the GenerateNextNumber function to increment a counter and send the result to Number.
- In DataLogger.c, implement the Log function to read the value from DisplayValue and log it.

Practical Task 6: Running and Testing the AUTOSAR Application in a Simulator

- Start dSpace VEOS:
- Open dSpace VEOS to simulate your AUTOSAR application.
- You'll use VEOS to test the functionality of the system in a virtual environment.
- Run the Simulation:
- mport the EcuInstance.vecu file generated by SystemDesk.
- Build the simulation, and then start it. You should see the output logged by DataLogger.

Practical Task 7: Implementing a Processing SWC (Adder)(Adder.c)

1. Create the Adder SWC:
    - Create an additional SWC named Adder. This component will process the numbers before sending them to DataLogger.
    - Add two Required Ports (InputA, InputB) and one Provided Port (Sum).

2. Add RTE Events for Adder:
    - For each input port, create a Data Received Event to trigger the Adder_Compute function when the data changes.

3. Composing and Updating the Application:
    - Add the Adder SWC between the Counter and DataLogger in the MainComposition.
    - Leave InputB unconnected and test using VEOS to manually set the input values.