Automotive & Embedded Info

Never Forget Basics Whether its Life or Anything Else ... Basics are Cores. While seeing a Tree how can we forget Seed...

Why AutoSAR ? What it is?

Why AUTOSAR?

There are several major issues that need to be addressed:

1. Managing the growing complexity of automotive E/E systems.
2. Flexibility for product modification, upgrade and update.
3. Scalability of solutions within and across product lines.
4. Quality and reliability of E/E system

The standard for automotive software now under development will be a first step towards tackling these problems. The concept for the standard is a layered software architecture with standard APIs. Establishing a software standard will be a big step forward, but on its own it is not enough. Therefore standardization will apply not only to the software, but also to the whole development process from functional description to software testing. In future, design engineers developing a new E/E architecture will adhere to the AUTOSAR design process. The process uses a top down approach:

available hardware and system constraints).
2. Allocation of software components to each ECU.
3. Configuration of the software on each ECU.
4. Conformance testing

The goal is to have a standardized tool chain that will guide and assist the design engineer through the complete process. Introducing the AUTOSAR process would be a breakthrough in automotive E/E design. It would be a radical step and its repercussions would change the industry forever.

Here I would like to explain in a very simple manner:

LET US SEE SOME QUESTION ANSWER:

1. Who requires ECU? OEM for their CAR.
2. Who develop the ECU? Tier 1 Company
3. Who provides requirement to Tier 1 Company ? OEM.
4. Will ECU communicate with other ECU's of the CAR? Yes
5. What types of software's an ECU can use? Communication Stack like LIN, CAN, FLEXRAY, ETHERNET etc. Memory Stack, I/O Stack, OS, MCAL drivers, ECU Manager many more modules and complex drivers depend on application need.
6. Who will provide these stacks, tools and drivers? Tier 1 will alone make it or they will ask from some other vendor? Yes.

What If I say here for all the above development there will be some standard and whole automotive industry will follow these standards. The work we can divide like this:

1. Based on the system specification/standard OEM will create their requirement and will follow standard architecture for ECU

2. Based on the specification of application and CDD tier 1 will develop ECU.
3. Based on the specification of BSW stack(Communication Stack like LIN, CAN, FLEXRAY, ETHERNET etc. Memory Stack, I/O Stack, OS etc.) registered vendors will develop BSW stack.
4. Based on the specification of microcontroller peripherals silicon registered vendors will develop MCAL drivers
5. Based on the specification of RTE, registered vendors will develop tool to generate RTE.
6. And Many more things.

Conclusion: Above all points are indicating towards an module wise abstracted and robust software which can be reused by any ECU. That is the reason why AUOTSAR has been introduced automotive software development life cycle so that one team has a responsibility to write specification for the development of ECU and other teams of different expertise can follow these specification and will develop their software as per standards.

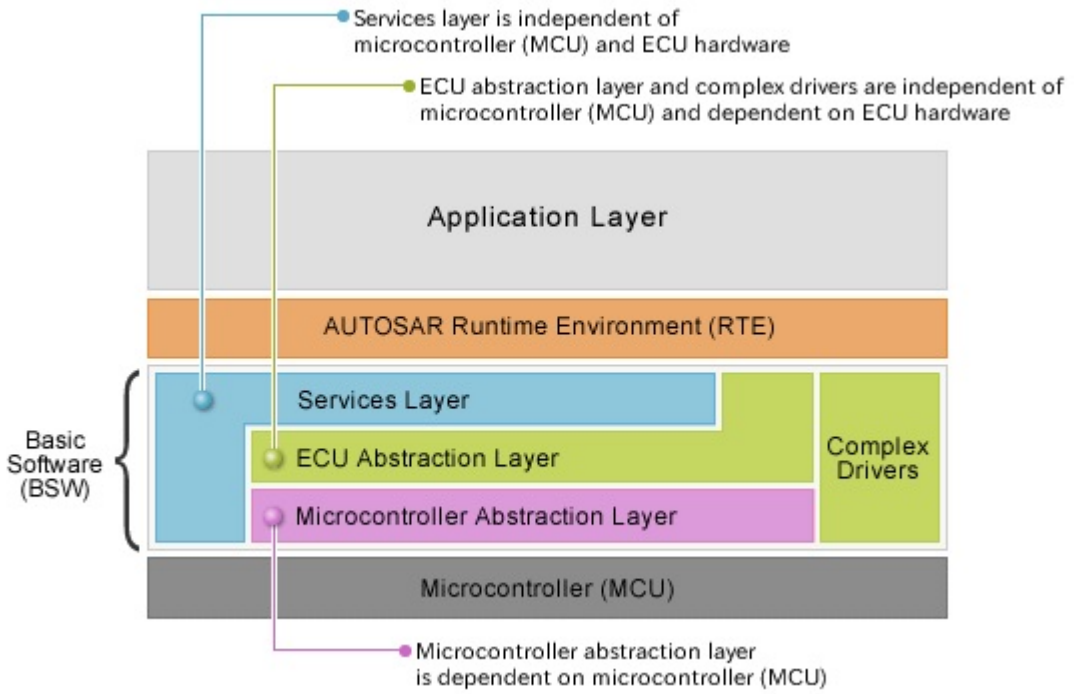
Benefits: Reusability, Choice to get software from different vendors, reduction in complexity, Quality assurance etc.

Figure is self explanatory about OEM, Tier 1, Silicon Vendor, BSW Vendor, Tool Vendor etc.

What is AUTOSAR?

AUTOSAR Software Architecture:

The basic concept underlying this architecture is that abstraction of the hardware will be done in layers.



MICROCONTROLLER ABSTRACTION LAYER:

This layer abstracts the microcontroller. The goal is to have a hardware-independent API but a hardware-dependant implementation.

ECU ABSTRACTION LAYER:

This layer is to abstract the other components on the ECU printed circuit board.

SERVICES LAYER:

This layer is almost hardware independent (the operating system needs a timer) and its task is to handle the different types of background services needed. Examples are network services, NVRAM handling, operating system.

RUNTIME ENVIRONMENT:

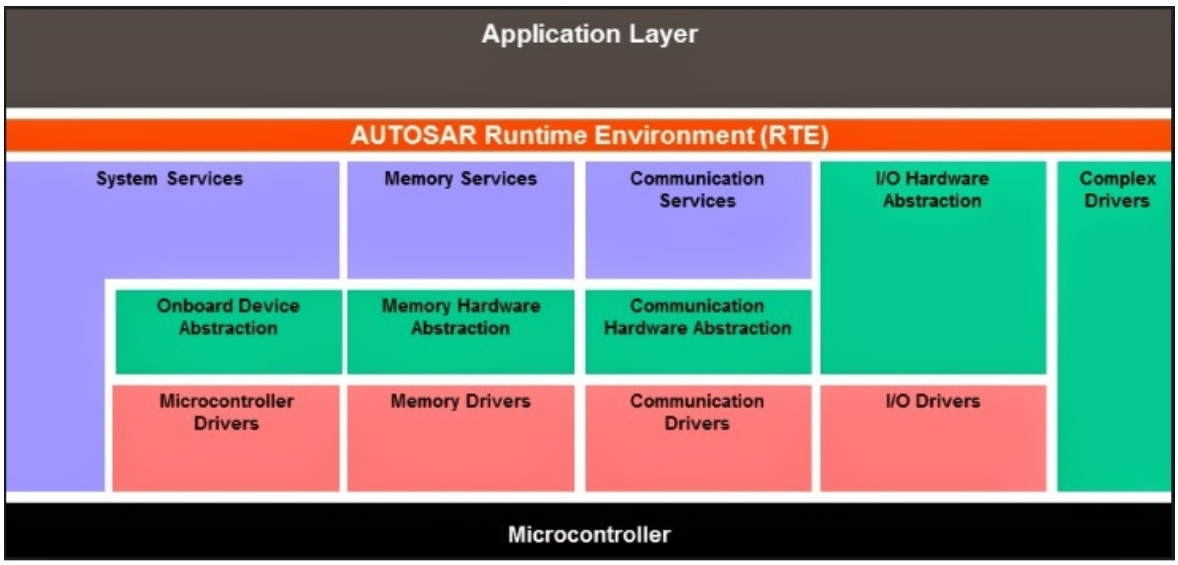
This layer abstracts the application from the basic software. All software components

running above the RTE are hardware independent components. Within the partnership, the APIs and the features of these modules, except for the complex drivers, are specified. How to realize this API functionality is up to each implementation.

CONFIGURATION:

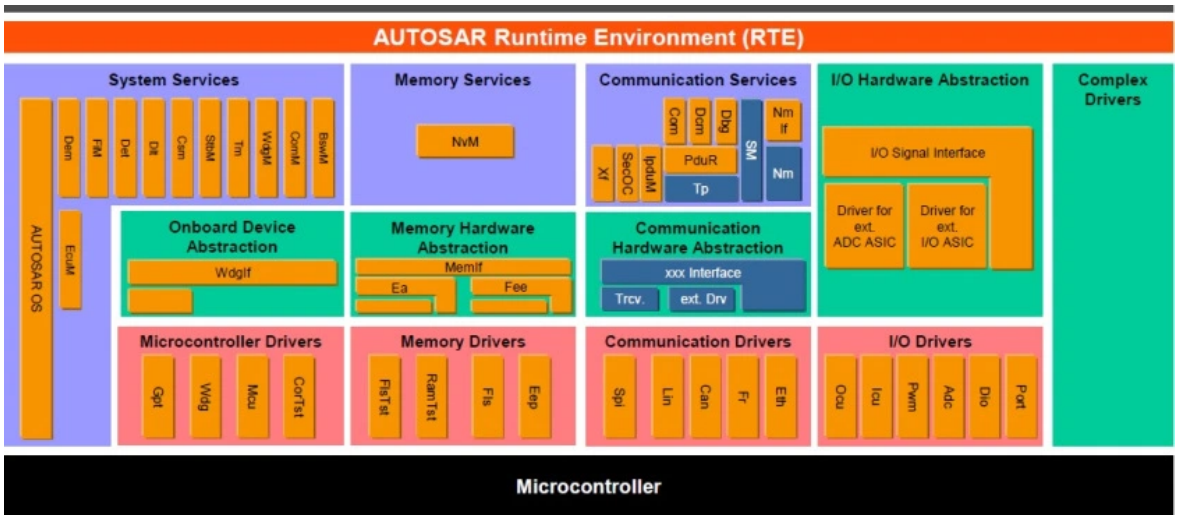
Let's assume that we have a standard API. The question is how do we adapt this API to fit
different microcontroller platforms and system needs? For example: one SPI channel for one application may need to run at 1 MHz with no chip select, but another SPI channel for a different application needs to run at 250 kHz with chip select. The AUTOSAR approach is to have a very flexible configuration concept. This means that the basic software should be configurable, so that it can be adapted to fit both the ECU hardware and the application needs. Inside AUTOSAR, this configuration concept will be

standardized.
The tools used for the configuration will, however, not be standardized. The tool feature set will be up to each tool vendor, but the input format to all. tools will be same. Different tools will have different graphical user interfaces and feature sets. Some tools may be able not only to configure the software but also to generate the source code for the software implementation.
To make possible seamless usage of different tools, it is essential that configuration is handled in a uniform way, i.e., based on a common file format. For this purpose, AUTOSAR will standardize a configuration template based on the XML file format.
This configuration template will describe the parameters that should be configured and the dependencies that exist between them.
In order to keep the software as portable as possible, the AUTOSAR partnership will specify as many of the configuration parameters as possible.



Here I would like to explain in a very simple manner:

Detailed Diagram AUTOSAR Layered Architecture:



Above diagram in details shows AUTOSAR layered architecture. We can see how many modules comes under this architecture.

What we mean by Layered Architecture here? After software compilation we will output file like hex etc. and in output file there will no existence of layered. In normal term we can say layered architecture is the way to understand the flow of communication within ECU whether it is data or memory or I/O etc.

SUMMARY:

1. Flow of signal/data form Application to Hardware.
2. Flow of communication Stack from top to bottom and vice versa.
3. Memory stack view.
4. I/O stack view.
5. Visualization of module service like OS, ECUM, BSWM, etc.
6. CDD interaction with other modules.

As a AUTOSAR introduction above things are enough. To get more details please go through the below link:

http://www.autosar.org/fileadmin/files/standards/classic/4-2/software-

architecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

Share this:

Twitter Facebook

Like

One blogger likes this.