AUTOSAR Basic Software for Complex Control Units

Dirk Diekhoff
Elektrobit Automotive
Erlangen, Germany

Abstract—Dirk Diekhoff, Elektrobit Automotive

"The development of complex control units requires mature and reliable basic software as well as integration support particularly in early phases of the project. In this presentation Elektrobit Automotive will focus on new AUTOSAR basic software features such as multi core and functional safety. We will show how integration and validation will be enhanced by diagnostic logging and tracing functionalities."

I. INTRODUCTION

For years, AUTOSAR (AUTomotive Open System Architecture) has been a de facto standard for software architecture in vehicle control systems.

In the past few years, requirements for safety, environmental protection, and comfort have grown significantly in automotive development. Manufacturers in the automobile industry are faced with increased requirements for safety, environmental protection, and comfort with complex vehicle electronics.

Up to 90 electronic control units (ECUs) are in use in a high-end car. An entirely new challenge for OEMs and suppliers is the high networking degree between functions and devices. The growing number of functions and electrical consumers means greater requirements for the on-board communications and power network in all classes of vehicle.

In addition to more complex overall systems, the larger number of control units in the vehicle also leads to increased material requirements: to connect the individual ECUs, a total of several kilometers of copper wire must be used, making the cable harness larger and heavier.

While the automobile industry is already using AUTOSAR release 3.1 in development and production, release 4.0 of the standard has progressed even further in important points in order to meet the new requirements.

II. GOALS OF AUTOSAR

The global AUTOSAR partnership has defined a standard guaranteeing the efficient development of ECU software for the automotive industry.

AUTOSAR supports the modular development of ECU-software. The standardized basic software forms the foundation for the development of application software by different manufacturers. Functions in the ECU network are easier to integrate and replace, making vehicle manufacturers more flexible in the use of ECU software. They can use individual functions on different platforms, model different variants and equipment sets in the software architecture, and replace specific new features as needed. Software and hardware can be replaced and updated throughout the vehicle's lifetime.

III. CURRENT VERSION (RELEASE 3.1)

In the AUTOSAR architecture, application software, runtime environment (RTE), basic software, and hardware are clearly separated from one another.

A. AUTOSAR layered architecture

AUTOSAR specifies a layered architecture of basic software (BSW) modules and application programmer interfaces (APIs):

1) Application layer
This layer contains the functional applications of an ECU network. The AUTOSAR standard refers to them as AUTOSAR Software Components.

2) Microcontroller abstraction layer (MCAL)
Functions and peripherals of the microcontrollers are abstracted here. This makes the higher layers independent of the specific microcontroller.
3) **Electronic control unit (ECU) abstraction layer**
   In this layer, the design of the ECU is abstracted.

4) **Service layer**
   This layer provides basic services. These include the operating system, network services, storage management, and bus communications services.

5) **Run-time environment (RTE)**
   The run-time environment (RTE) connects the basic software with the application software.

**B. AUTOSAR Basic-Software**

The release 3.1 of the standard consists of 47 defined BSW modules, which includes an OSEK-based operating system and five clusters:

- Mode Management
- Diagnostic
- Memory
- Communication (including CAN, FlexRay and LIN)
- Firmware

**C. Modularization**

Modularization and the standardized interfaces make software development more flexible. The development of individual applications no longer depends on the basic software used. This makes replacement of individual components throughout the product life cycle of the vehicle easier and more efficient.

**IV. NEW FEATURES IN AUTOSAR 4.0**

The basic concepts of the AUTOSAR standard have been stable since Release 2.1, and in release 3.1 had already reached a high degree of maturity.

Release 4.0 adapts the standard to the new, more stringent requirements and provides an important contribution to the efficient use of resources in ECU development.

Some of the new features will be highlighted below. Specifically the topics of functional safety, multi-core architectures, and functionality for diagnostic logging and tracing will be presented in the following.

1) **Multi-Core Architectures**

One possible solution for reducing the material costs in automotive ECU networks is to use domain controllers. A single domain controller can handle the functions for an entire area, for example the chassis or the engine.

Domain controllers are the central nodes of the domains, and are connected via a central gateway or a shared FlexRay backbone. Each domain controller has a large number of functions and uses sub-bus systems (e.g. CAN or LIN) to connect additional ECUs for less frequently used functions, as well as intelligent actuators and sensors. Clever use of ECUs can save material and weight during vehicle manufacture. This leads to more cost-efficient production and more energy-efficient vehicles.

But how can architectures be efficiently implemented with domain controllers? Here, too, AUTOSAR 4.0 and its extensions to multi-core functionality provide important solution approaches.

If domain controllers handle the tasks of multiple individual ECUs, they need more powerful microcontrollers. Higher clock frequencies and larger packing density of the components in the assemblies, however, lead to higher power consumption and greater waste heat.

Multi-core solutions promise to cover the need for computing power. The use of multiple cores in one ECU is more cost-effective and more efficient. The architectures can be scaled more easily and achieve better performance per watt, as well as better data throughput.

Multi-core architectures do not only lead to a more efficient distribution of control functions, they also offer new possibilities in the area of functional safety.

2) **Functional safety**

Functional safety is becoming one of the most important topics in automotive development. New functions in the area of driver assistance and driving dynamics control, as well as functions for passive safety, are usually implemented using electronic control units. To ensure safety, these functions must satisfy strict requirements. In vehicle technology, safety-relevant functions are already subject to the standard by International Electrotechnical Commission, the IEC 61508. The new ISO 26262 Automotive Safety Standard ("Road vehicles – Functional safety") to be released in 2011 will present an industry-specific variation of that standard.

Multi-core processor ECUs support different safety architectures because they can operate redundantly as well as in a purely parallel mode with independent operation of the processor cores. This allows them to guarantee an optimum balance between safety and performance.

In addition to hardware safety, the ECU-software has also an impact on functional safety. The AUTOSAR consortium addresses the topic of ISO 26262 in the release 4.0 with a series of new features that allow safety-relevant applications and non-safety-relevant applications to operate on the same controller.

Support for multi-core architectures has also been added. For example, software components can be partitioned at the application software level and can also be executed on different cores within an ECU. The new inter-core synchronization feature ensures that the execution of partitions runs at the same speed.

To increase the reliability of signal transmission between software components, the new AUTOSAR release provides end-to-end protection (e2e protection libraries). For example, signals can be assigned a serial number or a checksum for a cyclical redundancy check (CRC).

3) **Memory protection**

Extended memory protection permits the partitioning of the memory and assignment of access privileges to memory and
peripherals. In addition to the operating system, memory protection now also applies to the run-time environment (RTE).

Not all of this newly introduced functionality must necessarily be used. They can much rather be seen as parts in a toolbox, from which the user can select items to match the particular safety requirements.

V. CONFIGURING THE BASIC SOFTWARE EFFICIENTLY

The foundation of the basic software is defined in the specification. However, that isn't enough for efficient configuration of the software. AUTOSAR consists of nearly 50 modules of vastly different levels of complexity. That means that a large number of parameters must be configured. There are about 90,000 to 100,000 configuration parameters, with up to five selection possibilities for each parameter. To cope with the complexity of configuration, there must ways and means of supporting this process.

EB, with their EB tresos Studio, provides an open, Eclipse-based graphical development environment in which customer-specific extensions can also be integrated. Each basic software component of an ECU is configured, validated, and generated in the same environment.

A. Tool support

A good tool for the configuration of AUTOSAR modules must not only enable the user to configure the individual modules, but also provide some guidance in selecting suitable parameters. Since there are clear dependencies between the module configurations, the developer can be guided through the configuration process without him having to know the whole AUTOSAR specification by heart. Taking the parameters already configured into consideration, a tool can make a preliminary selection and only offer the configurations that are valid in combination with the AUTOSAR specification. At any time during development, it is therefore ensured that the resulting basic software is error-free and optimized.

If only error-free configurations are permitted right from the start of development, this represents an enormous gain in efficiency. Erroneous configurations need not be laboriously cleaned up, and communication between individual modules can work correctly because they all comply with the same specification.

An optimized setup of the basic software is not only about avoiding errors, but also about keeping the code size of the basic software as small as possible. Not all the parts specified in AUTOSAR are necessary or practical in every use case. A tool can help in using only the code the software really needs, thus consuming less memory and resources.

VI. DIAGNOSTICS AND TROUBLESHOOTING

In practice, it is always the case that an error-free module configuration is not enough. Conflicts often arise during the integration and commissioning of the basic software modules and application software, as well.

The efficient creation of AUTOSAR-compliant ECU software also includes the quick detection and correction of such conflicts. The earlier an error is found, the less effort it takes to correct it. Here, too, software tools are available to help with real-time monitoring and diagnostic functionality for basic software modules.

As of AUTOSAR 4.0, a debugging module new to the basic software stack allows signals and raw data to be recorded in the controller's memory and be transmitted over standard bus systems like CAN, FlexRay, or Ethernet.

Hardware interfaces like EB's own EB 61x0 are then used to read this data on a host PC. Using software tools, the signals and data can be displayed graphically, analyzed, and interpreted in order to localize errors.

1) Development error trace

The Development Error Tracer (Det) is a system service located in the service layer. In the debugging module, one can configure which data should be collected and transmitted to the host PC. The EB tresos Inspector can then show not only the error number, but also an error description.

2) Run-time environment (RTE) trace

In the application layer, the communications between applications can be checked for errors by monitoring the transmitted and received signals passing through the RTE. The signals to be monitored must first be configured in the debugging module. For this purpose, EB provides a configuration wizard integrated into EB tresos Studio.

These signals are displayed on the host PC with their signal values (EB-specific) and a time stamp.

3) Operating system (OS) trace

Monitoring in the OS module includes task switches, OS API and hook calls, and interrupts (EB-specific). In the software tool, in addition to a task list, statistical data can be shown on task execution times, periods, and deviations.

4) Future analysis topics

Other interesting topics for analysis have also been identified:

- Start-up times (EB-specific)
- Signal propagation delay (EB-specific)
- Online status of the basic software modules (EB-specific)
However, analysis is just the first step; the error must also be corrected. That is quickest when the result of the analysis can be annotated directly back into the configuration environment for error correction. The adapted software can then be tested directly on the ECU again.

VII. CONCLUSION

AUTOSAR's victory lap can't be stopped. In the future, the automotive industry and suppliers will increasingly rely on this standard to remain flexible in the development and use of ECU software and to be able to concentrate their resources on unique selling points.

While the already existing AUTOSAR releases are going to remain under maintenance, the development partnership is already working to advance the standard. Particular focus is given to the selective extensions, greater maturity, support for new hardware mechanisms and the further development of the AUTOSAR system. Thanks to the efforts of the development partners, the standard is guaranteed to stay on top of technological developments and to adapt to changing requirements.