User-friendly Configuration of AUTOSAR ECUs with Specialized Software Tools

The simple CAN ECU is a thing of the past. Now, a typical ECU utilizes many functions of the AUTOSAR basic software to perform its complex tasks. However, the more functions there are, the more difficult and extensive the configuration process is too. Without tool support, developers would be lost.

Three factors affect the complexity of ECU configuration:

- **More standardization**
  A large share of ECU software has now been standardized as basic software by means of AUTOSAR. However, the AUTOSAR principle “configuration instead of implementation” forces developers to create a consistent overall configuration from the start. There are no provisions for making corrections directly in the code.

- **More functions**
  New microcontrollers with multiple cores and memory protection features and new networks such as Ethernet have increased the scope of the basic software, and therefore configuration needs.

- **New cooperation models**
  The AUTOSAR methodology encourages new roles in the development process (Figure 1). In addition to the supplier of the AUTOSAR basic software (TIER2-BSW), there are also suppliers of the application software components (TIER2-SWC) and hardware-related Microcontroller Abstraction Layer modules (TIER2-MCAL). This means that more coordination effort is required initially in the configuration process.

How do Software Tools Help here?

The AUTOSAR standard continues to develop in a highly dynamic manner. There are changes with each new release: the functionality of existing BSW modules is changed and new modules are defined. All of these modules come with their own configuration parameters. However, the AUTOSAR configuration approach is fundamentally able to handle this dynamism with its formally described module structures and parameters. This makes it possible to quickly and easily define the Basic Software Module Descriptions (BSWMD) contained in the delivered BSW. At first, this formal method sounds enticing as an approach for tools: the tool company would develop a generic tool with a reasonable amount of effort, and then all parameters could be individually configured – those parameters that are known
today as well as all future parameters. However, no one would want to configure the thousands of necessary parameters in such an editor without further support.

**Specific Editors and Assistance Functions**

There are a number of approaches for developing a software tool that can simplify this work. A specially developed editor displays the relationships between parameters and simplifies the configuration process, e.g., by bulk operations. In addition, such an editor can group parameters thematically—i.e., even extending beyond module boundaries. The editor’s graphic views make it easier to understand the complex configuration. Such tools are helpful and necessary in all domains of the basic software such as communication, mode management, diagnostics, memory management, and I/O. When using an editor in the area of memory domains, for example, the developer can rather simply insert a memory block, which is then consistently configured in the non-volatile RAM manager (NvM) as well as in the flash EEPROM emulation (Fee). It is also very easy to estimate the overhead from the ratio of user data to block size in a graphic display (Figure 2).

Assistance functions are also helpful in the configuration process. When the developer changes a parameter, for example, other parameters that depend on it are automatically corrected by a rules system. For more complex tasks, such as mapping runnable entities to operating system tasks, assistants may be of help: The developer can perform tasks like selecting runnable entities of the SWCs based on similar trigger conditions, assign a task and finally define the execution sequence within the task.

Another example from the mode management domain: The BswM module in AUTOSAR 4 permits completely user-defined configuration of arbitration rules, logical expressions and actions, for reacting to mode changes in other BSW modules or for requesting mode changes itself. An assistance function now helps to configure the BswM such that it behaves similarly to the relatively easy to handle ECU State Manager (EcuM) from AUTOSAR 3.

**Deriving Parameters from the System Description**

Configuration of the communication modules for CAN, LIN, FlexRay or Ethernet must match the system description that originates from the OEM. AUTOSAR provides for derivation of a basic configuration of the modules (the Base EcuC) from an ECU-specific extract of the system description (System Extract). In practice, however, the situation is somewhat more complicated (Figure 3): AUTOSAR has defined a standard format for the system descriptions. But in addition to this format, OEMs also use the traditional DBC, LDF and FIBEX formats. Furthermore, the TIER1 supplier might use its own system descriptions, e.g., for private CAN buses within the ECU or for LIN buses that connect sensors with the ECU. The software tool must therefore identify all possible types of input data, convert traditional formats by suitable preprocessing methods and generate ECU-specific extracts. It must also be possible to merge multiple, separate system descriptions into a common description. Only then is it possible to generate the Base EcuC.

Similar considerations apply to the diagnostic modules. The configuration of the modules must match the ODX specification of the ECU. Therefore, the ODX file must also be included into the Base EcuC. Sometimes the TIER1 also has a standard configuration for its projects that is available directly in EcuC format. These configuration elements should also be contained in the Base EcuC. The BSWMD files that the tool needs in order to check the configuration can originate from various sources. They are either provided by the TIER2-BSW supplier,
or they might be supplied by the microcontroller manufacturer so that they match the MCAL modules. Finally, the SWCs might be contained in the system extract which the OEM provides to its suppliers. According to the AUTOSAR 4 method, the system extract is first used to generate an ECU extract, which represents a flat perspective of the SWCs.

Afterwards, for example, the ECU extract is extended to include other SWCs which might be provided by external suppliers (TIER2-SWC).

Using the ECU Extracts and the Base EcuC, the ECU developer creates the overall configuration of the BSW and RTE (Runtime Environment). Here, the tool should treat those parameters taken from the Base EcuC as write-protected to avoid deviations from the system extract.

**Project Update**

Over the life of the project, the developer continually receives updated input data. Typically, these updates arrive at different points in time and with varying frequency. The OEM distributes new system descriptions according to the specific milestones of vehicle development, while the diagnostic description is typically updated much more frequently. Often, the time span between receiving the system description and the delivery date of the ECU is very short. A tool must therefore be able to update the configuration with the new input data as automatically as possible. A project update function can do this by generating a new Base EcuC, and by updating the actual configuration to the new Base EcuC.

But what happens when there are errors in the system description? Even when the problems have been clarified with the OEM, a corrected system description is usually not available immediately. In such cases, the ECU developer might want to intentionally ignore some of the derived parameters and overwrite them with other values. This is a way to correct the error directly in the ECU configuration. Since such a deviation is always critical, the ECU developer should be able to explicitly set the parameter status as “user-defined” in the tool. As long as the parameter is in this status, the tool may not overwrite its value when performing a project update. Only when the ECU developer removes the status “user-defined”, the parameter adopts the derived value.

The tool also helps in coordination between the ECU developer and the OEM, e.g. by generating a report on overwritten parameters.

![Workflow for creating and updating an ECU configuration based on the example of DaVinci Configurator Pro from Vector](image-url)
“Merge” Function for Multiple Developers Working in Parallel

Even on smaller ECU projects, multiple developers are always working on the project simultaneously. If competencies are clearly defined (e.g., “Colleague xy is always responsible for the operating system.”), then it is possible to avoid making parallel changes to the same module or to the same SWC. Typically, however, the developers are working in parallel, and they develop ECU functions that run vertically through all of the architecture layers of the ECU software (Figure 4). In the times of manual C-coding, the integrator would then merge the different code branches on textual level. With AUTOSAR, however, this would mean that the integrator would have to merge XML files in AUTOSAR format that could be many megabytes in size. It would be practically impossible to compare or merge an AUTOSAR configuration using general XML tools. The structure of the file and the many references within the file are much too complicated for that.

Only an AUTOSAR-tool that clearly displays the differences in the AUTOSAR configuration and offers merge options would be of help here (Figure 5). Ideally, the user could view the differences directly in the same editors with which the user created the configuration. Then the user would feel “at home” and would not have to work with a separate tool with different, unfamiliar view of the data. Nonetheless, the following rule of thumb applies: The fewer the changes that need to be merged the better. Massive changes to the configuration might result from a project update to a new communications matrix (C-matrix), for example. Therefore, the following practical procedure is recommended: Starting from an integration milestone MS0, the feature developers each do their development work on a separate branch (Figure 6). The branches are merged, one after another, in the main branch – possibly in several intermediate steps (MS1 to MS3). Only then can the integrator update a project to a new C-Matrix (MS4).
Afterwards, new branches can be added and functions developed that match the new C-Matrix. All of this can be supported by a Configuration Management (CM) system, which also manages the implementation files of the SWCs.

Available Tools
The tool DaVinci Configurator Pro from Vector already enables the described working method. Soon, DaVinci Configurator Pro will also permit the creation of configuration variants, which can be dynamically switched in the ECU. They are also known as Post-Build Selectable Variants. Here too it is important to intelligently implement the available AUTOSAR concepts in tool functions. Then the developer gets further valuable support in the development of AUTOSAR projects.

Translation of a German publication in Elektronik automotive, Issue 4 / 2014

Image rights:
Vector Informatik GmbH

Links:
Vector AUTOSAR Solution: www.vector.com/autosar

Matthias Wernicke (Dipl.-Ing. (FH))
After completing his studies in Industrial Electronics at the University of Applied Science at Ulm, he was employed at the Daimler Research Institute in Ulm for four years. Since early 2000, he has been employed at Vector Informatik in Stuttgart working on the development of methods and tools for designing distributed electronic functions in motor vehicles. Today, he is responsible for product management of the DaVinci AUTOSAR tools.