

The Universal Gateway ECU

Flexible post-build configuration of AUTOSAR gateways

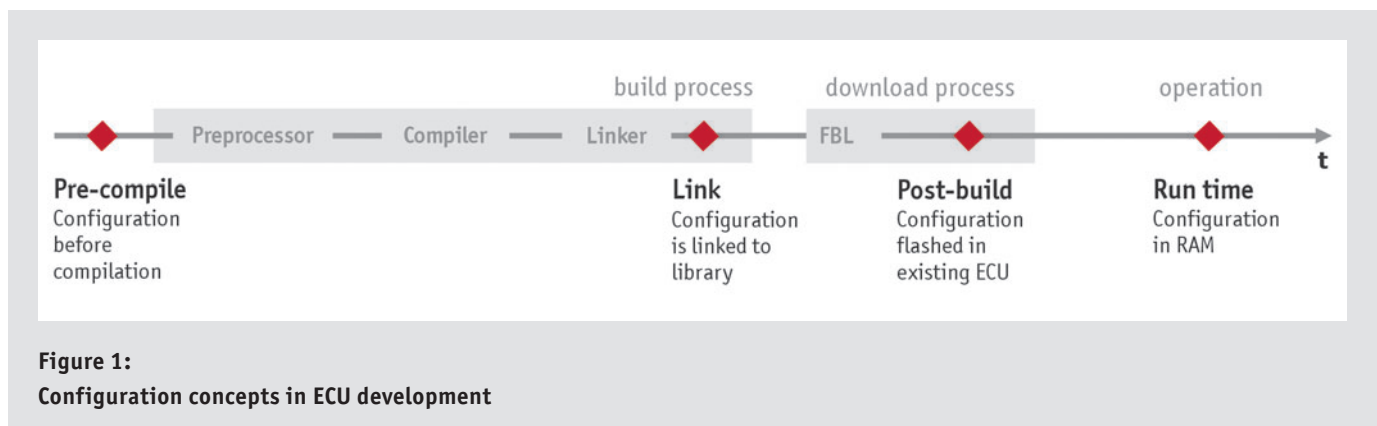
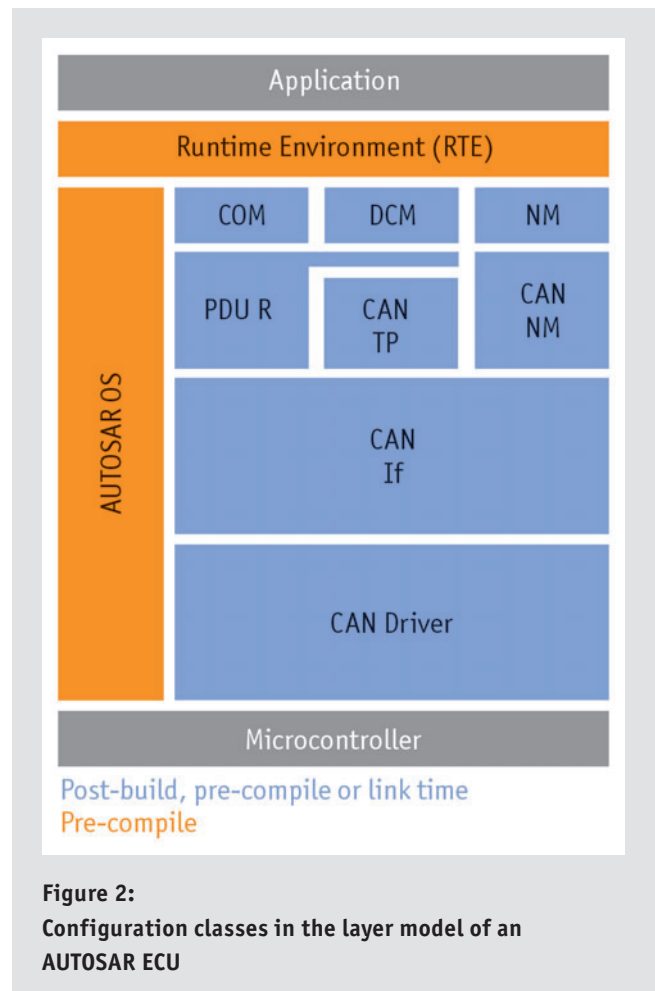
High flexibility of gateway ECUs is achieved by the post-build principle, since it permits a later configuration at any time – even in late development phases during ECU integration or even in the field. This results in universally deployable ECUs. Based on the AUTOSAR standard, a method is presented here that describes how the gateway functionality in the finished ECU can be adapted to new requirements.

Embedded systems can be configured at different points in time: Before the source code runs through the build process, the so-called pre-compile configuration occurs. This enables efficient implementation of configurations, e.g. variants, by macros or C pre-processor switches. The link-time configuration is typically used to generate a library and to link it with ROM constants. Furthermore, there is the option of configuration during the run-time of the ECU (run-time configuration), e.g. by calibration or diagnostic commands. In such cases, the configuration parameters must be stored in RAM. In contrast to the configuration types already named, post-build configuration is performed on ECUs that have already been built by loading the configuration data into the ECU via a flash bootloader (Figure 1).

The AUTOSAR standard [1] defines three so-called Configuration Conformance Classes (CCC), which cover these different configuration times:

- > CCC0 refers to a Pre-Compile Configuration
- > CCC1 Link-Time Configuration
- > CCC2 Post-Build Configuration.

Just which of these configuration options is in principle available for a specific basic software module depends on the character of the module. The RTE (Runtime Environment) supports only CCC0, because it is closely linked to the applications and consists of fully generated code. The operating system is also configured only at pre-compile time. Figure 2 shows – in the context of the AUTOSAR layer model – which configuration options exist for CAN communication modules.



For the most part, the modules belonging to the communication stack beneath the RTE support the post-build configuration per CCC2. Nonetheless, these modules have some pre-compile parameters such as the number of channels, use of data buffers (queues) and activation of debug modes. These settings must be known at the time the library is generated. In designing an ECU, a sensible strategy must be developed for using the post-build capabilities of neighboring modules. For example, it would make little sense to provide post-build capability to an individual module such as the PDU Router (PDU-R), while only granting pre-compile configuration to all other modules.

When is a post-build implemented for gateways?

If the functionality of individual ECUs changes, e.g. during a vehicle facelift, often changes must also be made to the communication matrix of one or more networks. This is where post-build comes into play and enables simple adaptation of the basic software responsible for communication between all ECUs of the affected network. This eliminates the need for recompiling and linking the code. If a gateway ECU is designed to be post-build capable, it is easier to fit it in as a standard ECU in different vehicle models. Gateway ECUs perform a majority of their tasks via the communications basic software. A vehicle facelift for such ECUs often consists of just new or modified routing relationships, for which all that is needed is a reconfiguration of the basic software.

The primary motivation for using a post-build configuration is the fact that it avoids the need for a new build process, and the configuration can be performed in late development phases during ECU integration or even in the field. This approach is of special interest for gateway ECUs.

The gateway ECU as flexible switchboard

The main purpose of a gateway ECU is to distribute communication data among the individual networks in a vehicle. According to the AUTOSAR standard, various basic software modules of the ECU perform this task. Which modules are used depends on the specific gateway functionality that is needed:

PDU gateway

The PDU gateway is a part of the PDU router (Figure 3). It routes entire data packets, known as Protocol Data Units (PDUs), from one network to another. This principle assumes that the PDUs are defined identically on both the source and target networks, i.e. they must agree in length and contents. This means that data can also be exchanged between different bus systems such as CAN, LIN or FlexRay.

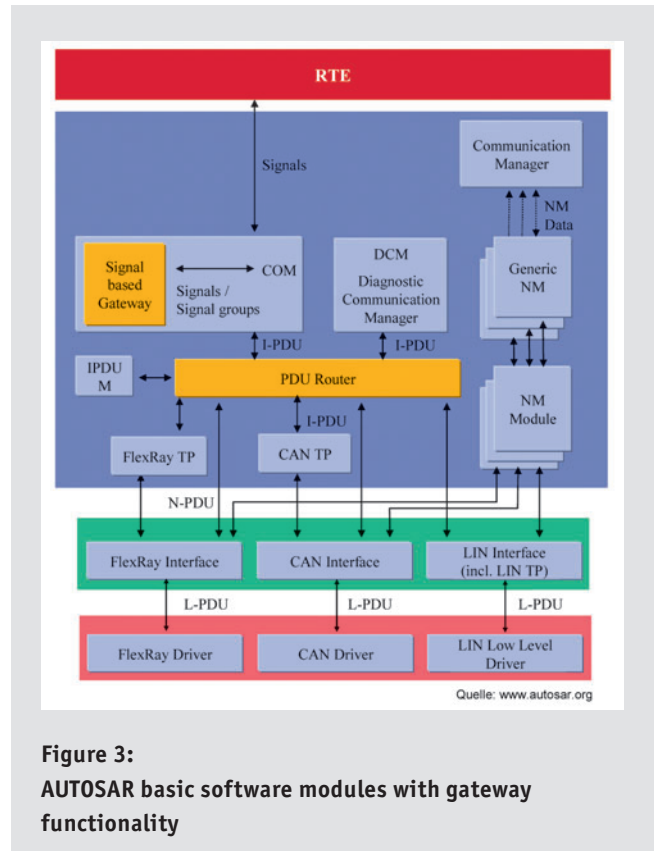


Figure 3:
AUTOSAR basic software modules with gateway functionality

Nonetheless, it should be noted that in accordance with the AUTOSAR specification the PDUs must be routed immediately after they are received. Consequently, the PDU router cannot perform send type or cycle time conversions. In some cases, however, such a conversion is necessary. An example of this might be a FlexRay-CAN gateway that routes a PDU from a FlexRay cluster to a CAN network as a CAN message. In this case, the gateway ECU must for example conform to minimum transmission delay requirements (debounce time) for the CAN message. In such cases, the PDUs are therefore routed directly to the COM layer, which then assumes this task.

TP gateway

Another task of the PDU router is to route transport protocol data. This comes into play, for example, if the extensive diagnostic data of an ECU need to be automatically transferred from one network to another. This involves receiving the data via the Transport Protocol (TP) and resending it. At first, the transmission occurs above the TP (Layer 4 in the ISO/OSI layers model), and this enables a conversion to different addressing methods and various bus systems. To keep delays and RAM requirements in the gateway as low as possible,

the TP gateway supports so-called “on-the-fly routing”: The gateway does not wait until all TP data have been received, rather it already begins to resend the data at an earlier time point. Consequently, it receives and sends simultaneously.

Signal gateway

Often just individual signals are needed on the other network. In this case, the gateway does not transfer the entire PDU, but only sends individual signals to the other bus. To do this, it first disassembles a received PDU into individual signals, so that it can then assemble them into one or more send PDUs. Besides modifying the signal composition and signal positions within a PDU, the send type and cycle time can also be changed. This method is also used when a PDU should contain both routed signals and signals generated directly in the gateway ECU.

Technical aspects of post-build configuration

Data structures for the post-build configurable data essentially have two different types of layouts (Figure 4). In the non-fragmented variant 1, the data structures are arranged one directly after another in memory. The individual data structures are accessed via an indirection table located at a static position. The data structures on the other hand may vary in size depending on the post-build configuration. The remaining memory is a contiguous block that is

available for other purposes. However, the basic software modules are highly dependent with respect to their implementation. If the basic software originates from different software suppliers, this variant generates a high coordination needs and is therefore impractical.

In the fragmented variant 2, the data structures are always placed at a static position in memory. Here, at the time of design, an assumption is made concerning the largest possible memory usage of individual data structures. In practice, some unused memory between the data structures remains. With regard to runtime behavior, variant 2 is more efficient, since no indirection is needed in data access. Despite the fragmentation, this variant also offers advantages in terms of memory efficiency, since an indirection table is unnecessary.

The use of post-build data structures has an enormous effect on the design of the basic software modules. In the case of the pre-compile configuration, for example, the separation of code and data is not required. This makes it easier to implement configuration settings very efficiently with macros or preprocessor switches and to generate optimized C functions with the help of code generators. The principle of the post-build configuration, on the other hand, requires strict separation of code and data for post-build parameters. A generator is only available for generating constants tables. The C functions are static. Figure 5 shows how the different configuration concepts appear based on code examples.

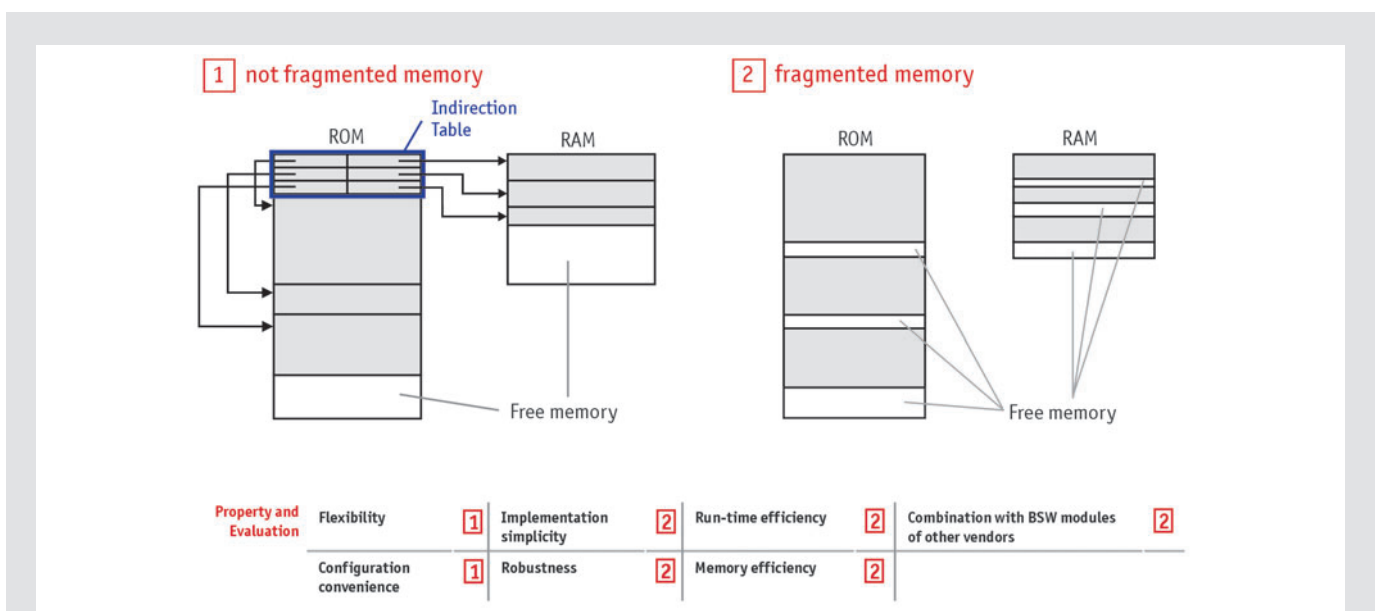


Figure 4:
Data structures for post-build configuration

| | Pre-compile time | Post-build/link time |
|-------------------|--|---|
| Enabling Features | <pre>#define FEATURE_ENABLED #if defined(FEATURE_ENABLED) ... #endif</pre> | <pre>static boolean FEATURE_ENABLED; if (FEATURE_ENABLED == true) { ... }</pre> |
| Scalar Values | <pre>#define VALUE 8</pre> | <pre>const uint8 value = 8;</pre> |

Figure 5:
Code examples for different configuration concepts

A gateway ECU requires knowledge of large numbers of signals or messages. This information exists in the form of data structures in the ECU's memory. As a result, in gateway ECUs the communication layers in the software architecture take up a considerable share of memory and runtime resources. Even when the more efficient variant 2 is selected, the post-build capable layout of the ECU typically causes increased resource requirements.

A tool chain for post-build configuration of gateway ECUs
Besides the aspects of memory and runtime resources in the ECU, the post-build principle also requires new processes to coordinate configuration parameters between participating development partners and exchange them reliably. One important source of support here is a well-functioning tool chain. Figure 6 shows the tool chain from Vector Informatik, which can also be used for post-build configuration of gateway ECUs.

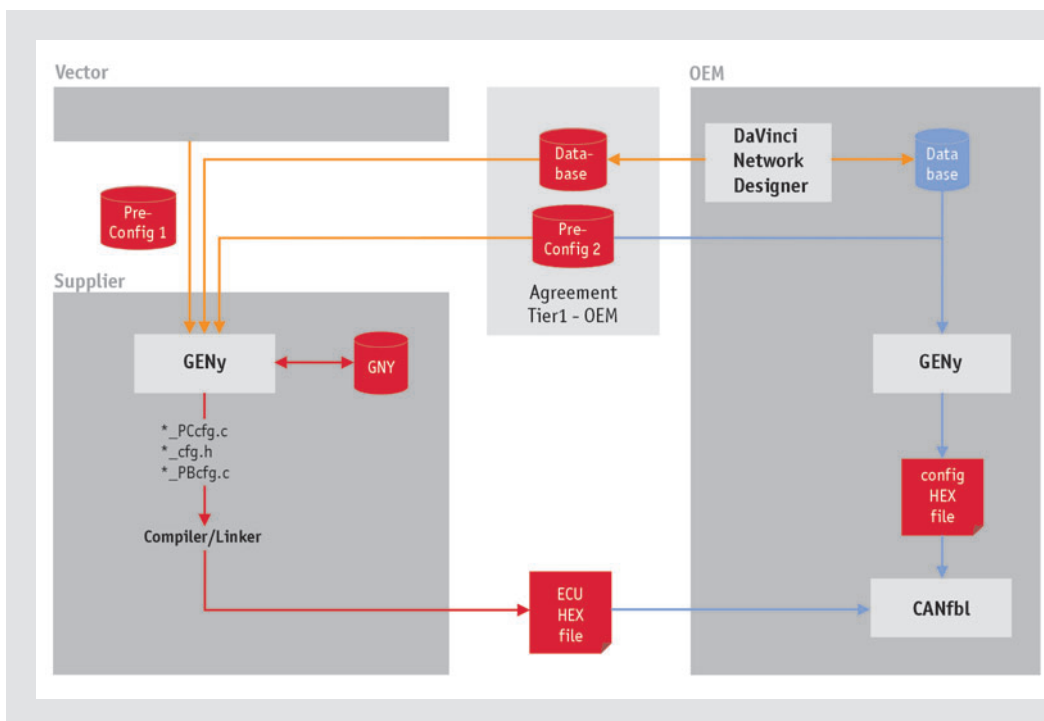


Figure 6:
Vector tool chain for the post-build configuration

At the beginning of a development project, the ECU supplier sets up the project based on the Pre-Config1 parameter set. Vector provides this parameter set along with the base software delivery for the ECU, and it contains pre-compile parameters that do not have any effect on the post-build configuration. Examples of this are general settings and use of the Development Error Tracer (DET). In coordination with the ECU supplier, the automotive OEM provides the Pre-Config2 parameter set and an initial network description (database). The Pre-Config2 parameter set contains those pre-compile and link-time parameters that have an effect on the post-build process and need to be defined in advance. Examples of this are the addresses of post-build data in the ECU, compiler options and maximum memory size (Flash and RAM). The initial network description, which the automotive OEM might create with the DaVinci Network Designer tool, for example, contains all signals relevant to the ECU. In the case of a gateway ECU, the routing relationships between the networks are also described there.

The ECU supplier uses the GENy tool to configure and generate the basic software using this input information. The routing information is prepared in the form of generated data structures (tables) for the individual basic software modules. This provides a foundation for developing the ECU based on the initial network description.

During the actual post-build process, these tables must be regenerated and exchanged in the ECU. The basis for this is a modified network description by the automotive OEM. If the updated tables now exist in binary format – and indeed in precisely the same form as the compiler would have created them – then another compiling and linking run is obsolete. The knowledge about the compiler behavior when generating the binary format is stored in GENy. This binary file is now converted to a standard hex format and is loaded into the ECU via the CANfbl flash bootloader. If the pre-config information is known to the automotive OEM, the OEM itself can also perform the post-build process directly.

Summary

The post-build process provides flexibility when changes are made in the communication description. Configuration is even possible in late development phases during ECU integration or in the field. This approach is especially useful for gateway ECUs, since it is possible to adapt them to modified network conditions without having to change the complete application code. However, the increased resource requirements must be taken into account. In any event, a gateway ECU is an interesting candidate for the post-build process, at least during the development phase.

Reference:

[1] AUTOSAR specifications: www.autosar.org



Hartmut Hörner

studied Electrical Engineering at the University of Stuttgart from 1987 to 1992. Afterwards, he worked as a software developer for ATM Test Systems. In 1998, Mr. Hörner came to Vector where he is the team leader responsible for the development of embedded software components. Hartmut Hörner represents Vector on various standards working committees (OSEK, ISO, AUTOSAR).