

Post-build configuration in AUTOSAR

Dipl.-Ing. Hartmut Hörner

Abstract

Embedded systems can be configured at different points in time. The AUTOSAR standard defines three categories, pre-compile time, link time and post-build time configuration. While pre-compile and link time configurations are common for setting up source code or libraries the post-build configuration has the capability to apply configuration settings to a complete ECU.

The properties of these concepts as specified by AUTOSAR are shown in detail, for all concepts typical examples are given and some restrictions are outlined. For the post-build concept the impact on the distributed development process of an automotive ECU is assessed and a possible approach for a post-build tool chain is shown.

After discussing the impacts of the post-build concept on the design of the embedded software components the effects on the resource requirements regarding RAM, ROM and run-time are analyzed. Finally specific requirements on the test procedure of the ECU are derived.

1 Introduction

Embedded software components can be configured at different points in time. The so-called pre-compile configuration is applied before the source code is put into the build process. This allows the implementation of some configuration settings (e.g. handling of variants) by macros or C pre-processor switches. Link-time configuration is typically used to link tables with ROM constants to a library. In contrast to the configuration concepts which are applied during the build process the post-build configuration is applied on an existing ECU. Here the configuration data is downloaded via a flash boot loader. Finally configuration settings may be applied during run-time, in such a case the configuration parameters must be located in RAM.

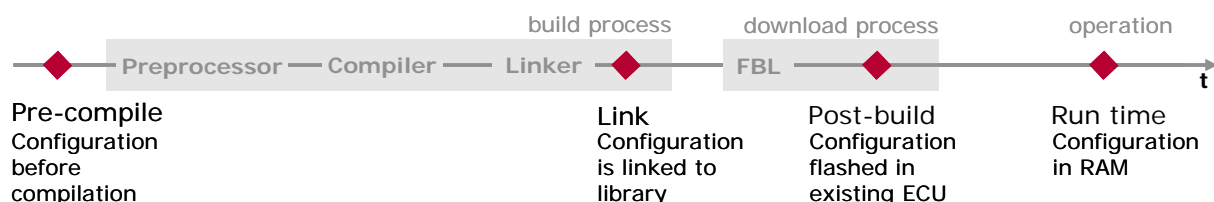


Figure 1: Configuration concepts

More comprehensive configuration techniques like the usage of interpreters, compilers or databases are very uncommon on deeply embedded devices and are only mentioned here for the sake of completeness.

The main motivation for the usage of the post-build configuration is that it allows a late modification of configuration settings without running the build process. Typical applications for the post-build configuration are changes in a communication data basis such as changing CAN identifiers. In gateways routed signals often pass the application completely so they can be added or removed without changing and re-building the application.

2 Configuration Concept in AUTOSAR

AUTOSAR [1] defines three categories of conformance classes as shown in the following figure. In the implementation conformance class ICC3 the basic software modules appear like specified in the AUTOSAR specs. In ICC2 some basic software modules are merged into clusters which allow some special optimizations at the modules' interfaces. In ICC1 the whole basic software appears as one large block.



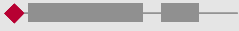



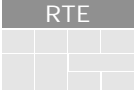

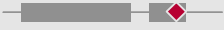
Implementation Conformance Class	Functional Conformance Class	Configuration Conformance Class
ICC1 (system with RTE interface) 	FCC1 (core feature set) 	CCC0 Pre-compile 
ICC2 (split-up in clusters) 	FCC2 (BSW module specific options) 	CCC1 Link-time 
ICC3 (split-up in BSW modules) 	FCC3 (system spanning options) 	CCC2 Post-build 

Figure 2: Conformance classes in AUTOSAR

The possible configuration concepts depend on the individual basic software module. The Runtime Environment (RTE) supports only pre-compile configuration because it is tightly coupled with the application. If the configuration changes (e.g. by adding a signal), the application needs to be re-built, because in the application no post-build concepts are supported. So it would provide no benefit if the RTE would support post-build.

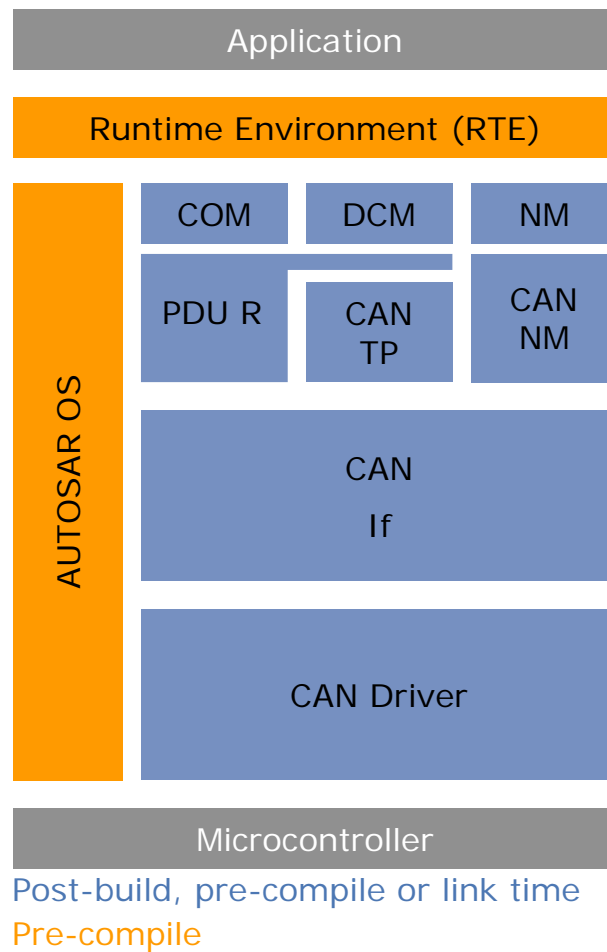


Figure 3: Example sub-set of the AUTOSAR layered architecture for CAN

In figure 3 an easy subset for the CAN bus is shown. The modules below the RTE belonging to the communication stack support different configuration concepts. However, also the post-build configurable modules have some pre-compile parameters such as the number of channels, usage of queues and activation of debug modes. These settings must be known when the library is built. During system design care must be taken that the configuration concepts of adjacent layers match. It is obviously not very useful to have the post-build capability for modules like the PDU Router only, while all others support only pre-compile configuration.

3 Process and Tool Chain Aspects

In a distributed development project the main advantage of the post-build configuration becomes more obvious. If the configuration concept requires a re-build of the software this cannot be performed by the OEM, therefore the supplier needs to be involved and several steps in the process must be repeated. In the post-build process the OEM is able to perform configuration changes.

Pre-compile link time process



Post-build process

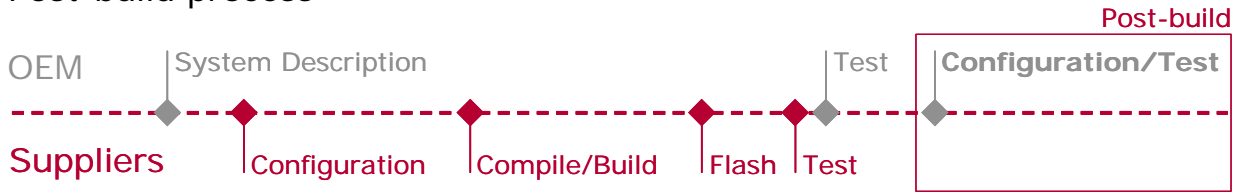


Figure 4: Roles of OEM and supplier in the configuration process

The following figure 5 shows an example of Vector's current approach for a post-build tool chain. In step 1 the communication data base is handed over from the OEM to the supplier. The supplier uses the configuration tool GENy to generate C code for the build process of the ECU. In the final step 3 the tool GENy is used by the OEM to generate a configuration hex file.

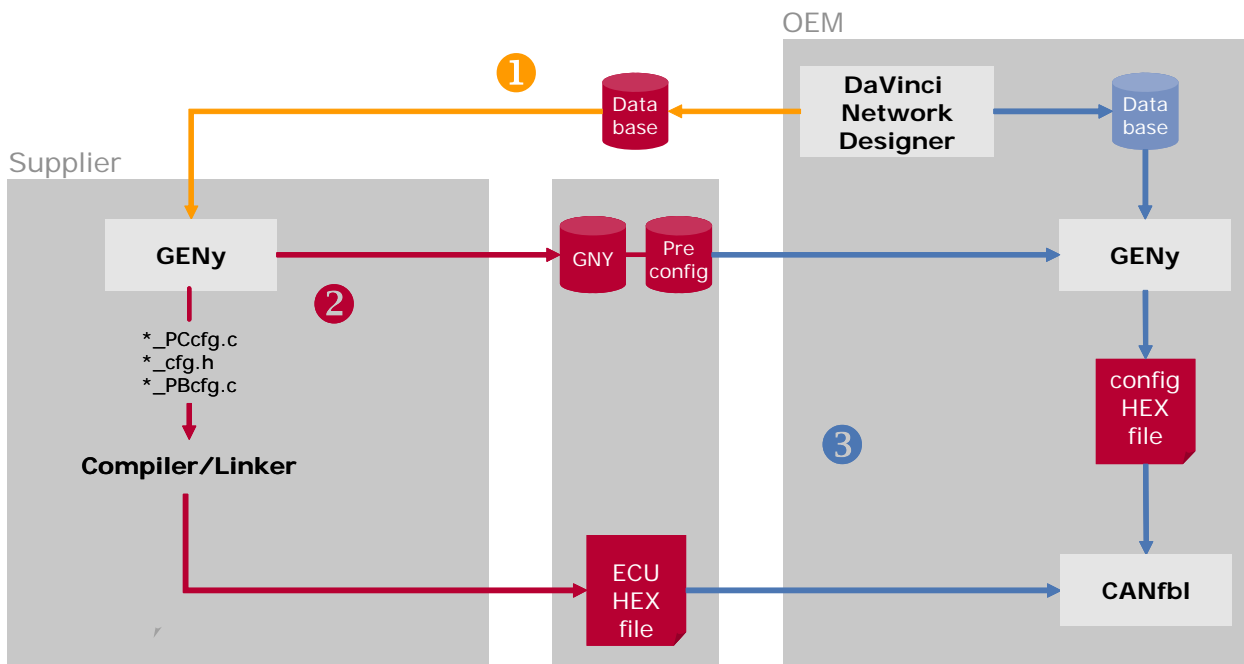


Figure 5: Implementation concept of the post-build tool chain

4 Technical Characteristics of Post-build Configuration

Data structures for the post-build data can be designed in the two different ways. In the non fragmented variant 1 the remaining memory is available in one block and can be used for other purposes. In the fragmented variant 2 all structures are designed under worst case assumptions. Variant 2 is more efficient since it requires no pointer indirections. Note that these considerations apply not only for single basic software modules but also for clusters. If modules of different vendors are combined variant 1 is not usable.

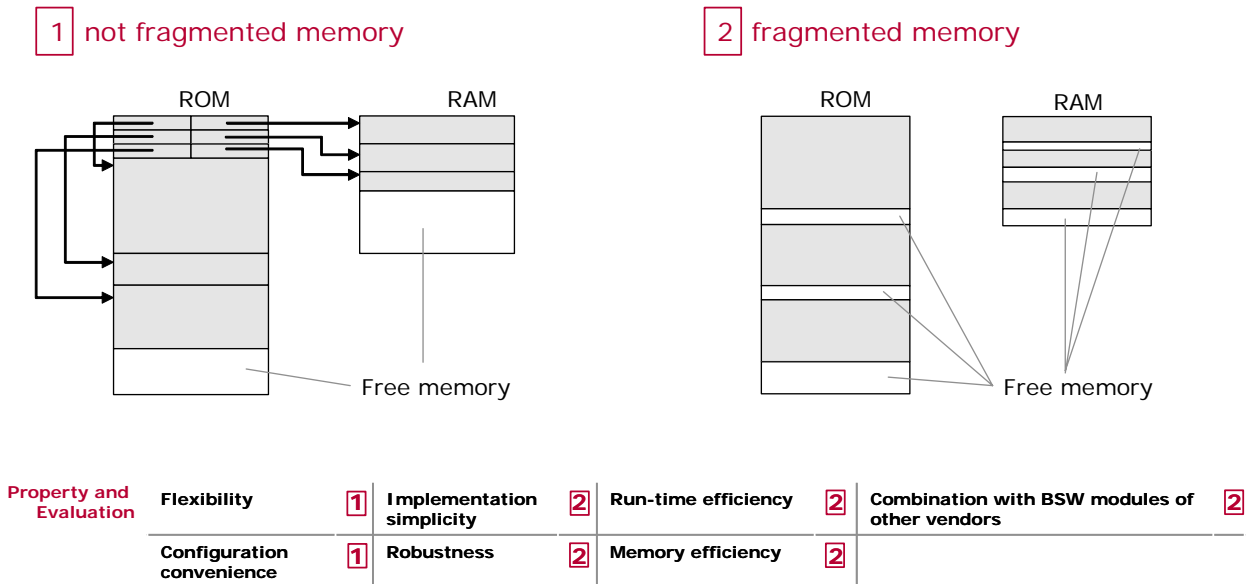


Figure 6: Data structures for post-build configuration

The post-build data structures also have a major impact on the design of a basic software module. In case of pre-compile configuration, no separation of code and data is required; therefore configuration settings can be implemented very efficiently with macros or pre-processor switches. It is also possible to generate C functions by the code generator. Post-build configuration requires a strict separation of code and data for the post-build parameters. The generator can only generate constant tables and no C functions. In consequence, the selection of the configuration concept can lead to a different design of the data structures and to different architectural concepts. In the following figure some example source code fragments are shown.

	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>

Table 1: Code examples for different configuration concepts

To assess the effects of the different configuration concepts on the performance an example is shown in the following figure 7. Reading a one-bit signal from the COM layer can be implemented by a macro in a pre-compile optimized solution. Depending on the target system this could lead to just one assembler instruction. In a post-build solution a generic signal read function is needed which has to perform several table lookups.

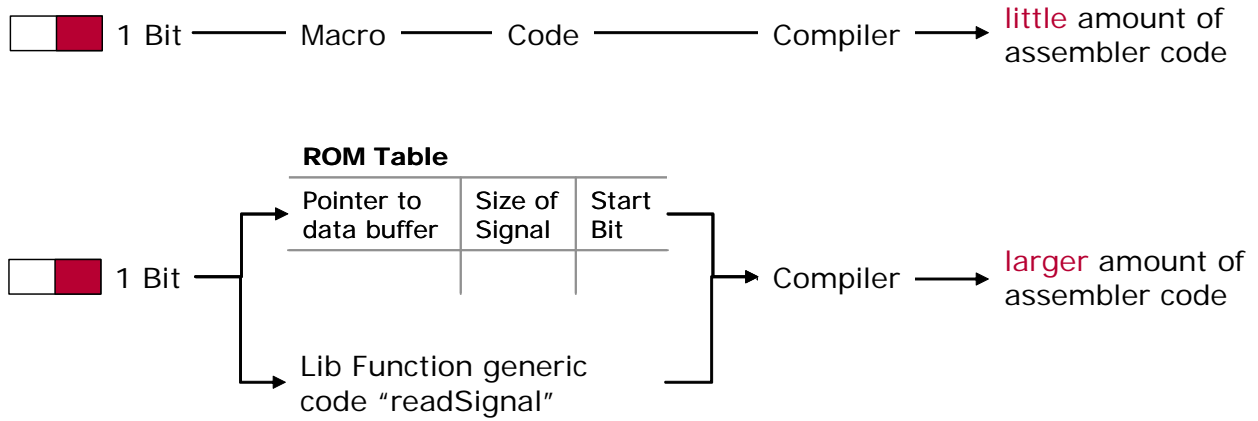


Figure 7: Example: Reading a 1-bit signal

So there is a significant difference in performance especially in case of signal oriented layers. The implementation of PDU oriented layers is usually based on tables so there is only a minor performance issue. In some cases it is useful to provide multiple implementations to provide the most optimal solution for all configuration scenarios.

5 Test Aspects

Special care has to be taken in the test process. If the configuration is changed post-build, the result is an ECU which has actually not been completely verified. While the impact of configuration changes on the external behavior is predictable by simulation models up to a certain degree, the impact on the internal behavior is usually difficult to predict. If for example the number of CAN messages is increased drastically post-build this will lead to a higher interrupt load which could influence parts of the application.

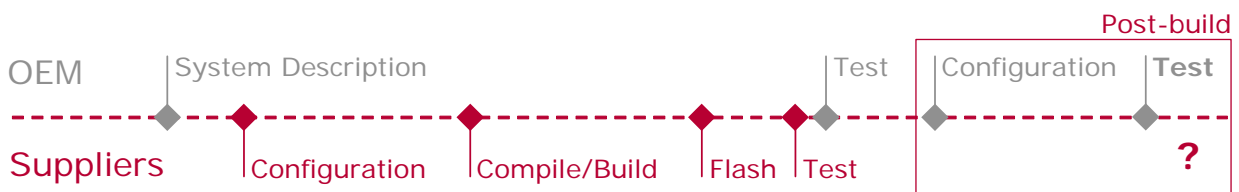


Figure 8: Test of post-build configuration

Additional test effort is necessary on the OEM side after a post-build configuration change has been performed.

Because the post-build configuration data is available in ROM tables, the usage of calibration tools is also possible to test configuration changes without running the post-build tool chain. This can be useful in rapid prototyping, e.g. to perform a parameter sweep.

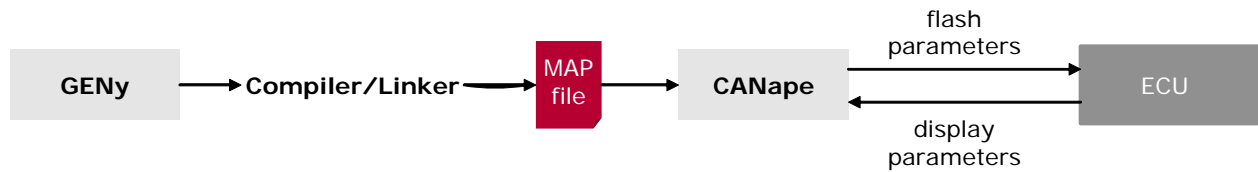


Figure 9: Usage of calibration tools

6 Summary

The use of the post-build configuration depends on the implementation cluster. The decision for the configuration concepts is a system decision which must be made early in the project. Post-build configuration adds flexibility but it goes along with increased resource consumption. A well-defined process is required and specific test concepts have to be in place.

References

[1] AUTOSAR specifications: www.autosar.org

Contact:

Hartmut Hörner
Vector Informatik GmbH
Ingersheimer Straße 24
70499 Stuttgart
hartmut.hoerner@vector-informatik.de
Tel: +49 711 80670-0
Fax: +49 711 80670-399

www.vector-informatik.de/autosar