MICROSAR Identity Manager

Technical Reference

Post-Build Selectable

Version 1.1.1

Authors	Hannes Haas
Status	Released

Document Information

History

Author	Date	Version	Remarks
Hannes Haas	2014-10-29	1.0.0	Creation
Hannes Haas	2015-01-26	1.1.0	ESCAN00080589: global root structure is now a structure with one element for each variant. Global Root Structure Customization now possible by configuring the ECUC module.
Hannes Haas	2015-03-03	1.1.1	Adapted file names of referred documentations.

Reference Documents

No.	Source	Title	Version
[1]	Vector	TechnicalReference_PostBuildLoadable.pdf	as delivered
[2]	Vector	TechnicalReference_EcuM.pdf	as delivered
[3]	Vector	TechnicalReference_BswM.pdf	as delivered

Scope of the Document

This technical reference describes the general aspects of the MICROSAR Identity Manager. The document does not describe BSW module specific functionality or limitations unless this is essential to understand the overall concept. Module-specific details can be found in the documentation of each BSW module.

This document focuses on BSW functionality. Configuration aspects are described in the help system of DaVinci Configurator Pro.

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Introduction

The MICROSAR Identity Manager is an implementation of the AUTOSAR 4 post-build selectable concept. It allows the ECU manufacturer to include several BSW configurations within one ECU. At start-up the application determines the BSW variant to be activated and initializes the BSW accordingly.

The ultimate goal of the Identity Manager is to reduce the number of ECU variants that have to be built, stored and distributed. The Identity Manager thereby focuses on communication and diagnostic stack variance. Variance of the application (that may or may not be required) has to be realized within the application internally.

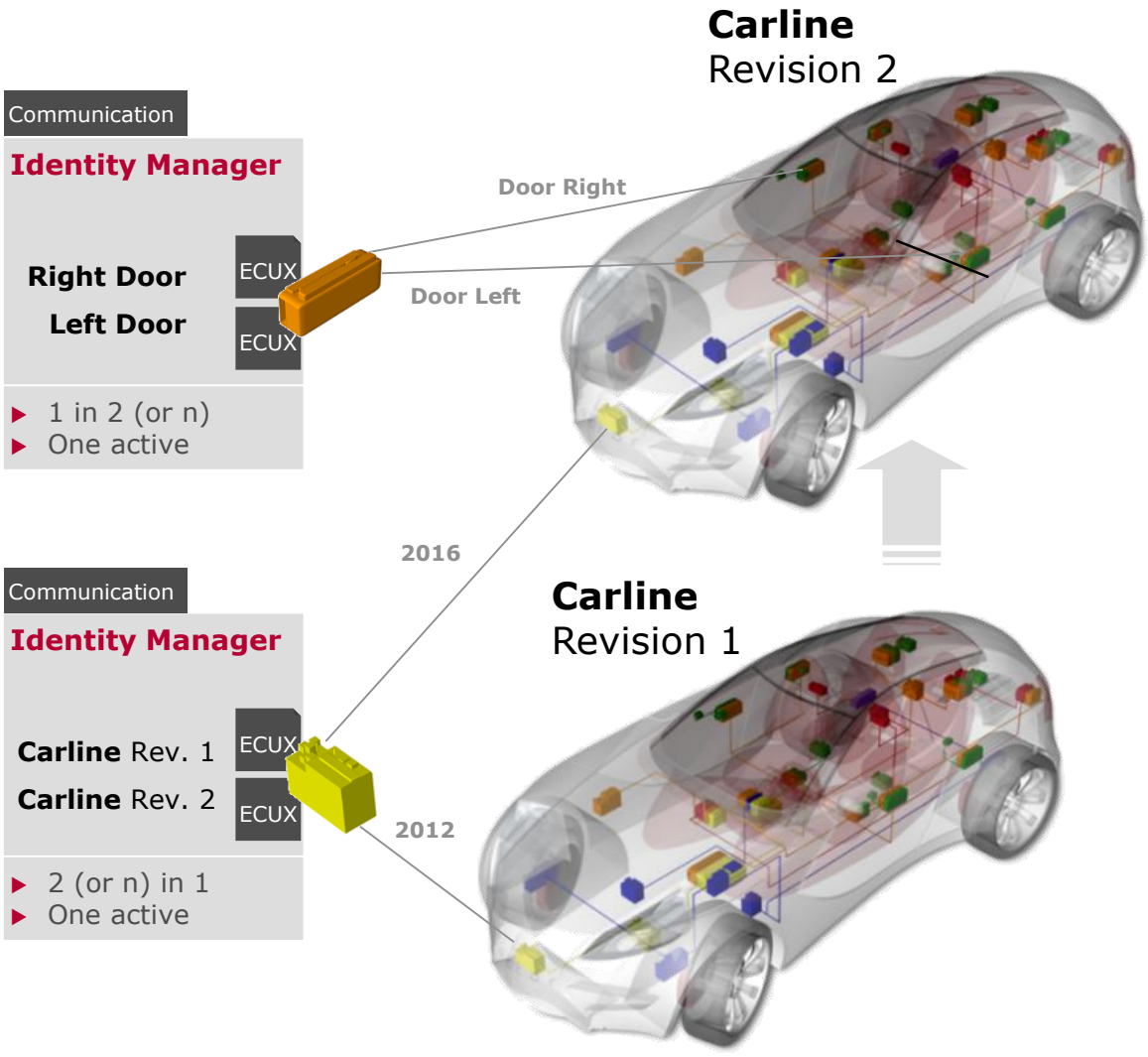

Figure 1-1 Identity Manager use-case exampels

Typical use-cases for the Identity Manager:

> Usage of one ECU multiple times within one vehicle. Typically the variants have a very similar behavior and communication description. An example may be the usage of an ECU that is used as both, driver and passenger door module.

> Usage of an ECU in multiple vehicle architectures that have each distinct requirements to the communication and diagnostic description

FAQ

One variant represents one complete set of functionality as required e.g. for the driver's door module. A variant is thereby an ECU global variant definition that can affect all BSW modules and potentially the application.

1.1 Comparison to Post-Build Loadable

Post-Build Loadable allows downloading configuration sets to the ECU at a time after the ECU has been build (e.g. end of line or in a workshop). This chapter points out in short the major differences between post-build loadable and post-build selectable (Identity Manager). Post-build loadable is a MICROSAR option that is available upon request.

Using the Identity Manager (an implementation of post-build selectable) all ECU variants are downloaded to the ECUs non-volatile memory (e.g. flash) at ECU build time. The Identity Manager does not allow modification of BSW aspects after ECU build time. If changes are required a new ECU build has to be created.

Post-build loadable in contrast allows the modification of selected ECU parameters after the ECU build time: at post-build time. The goal is to enable OEMs to update parts of the ECU BSW behavior independently of the Tier1, who controls the ECU build environment.

On request Vector can provide BSW packages that support both, the Identity Manager (post-build selectable) and post-build loadable. This will allow a post-build time modification of variant ECU projects.

2 Functional Description

2.1 Technical Background

MICROSAR Identity Manager is based on the post-build selectable variant handling defined by AUTOSAR 4. It uses variation points of binding time **PostBuild** in the arxml description files to define variance.

Variants – or according to AUTOSAR *Predefined Variants* – are defined based on a set of criteria settings within the DaVinci Configurator Pro Variants Editor. One criterion is an enumeration that allows formal definition of differences in the ECU configuration. At present DaVinci Configurator Pro supports a single criterion only. The default name is Communication. In a use-case with several input files, one Criteria Value is created for each (non-variant) input file. Using the Variant Editor it is possible to combine the criteria values to ECU Variants that are also visible as BSW configuration structures.

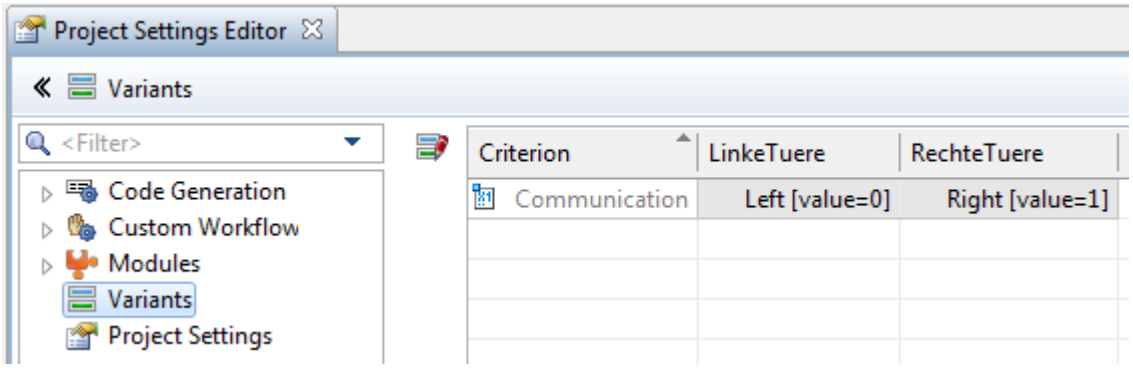

Figure 2-1 **Variants** configurator editor in DaVinci Configurator Pro

Note
Criteria are a modelling aid of the AUTOSAR schema and do not reflect directly on the generated BSW code and the BSW configuration structures used for BSW initialization.

The generated configuration structures instead reflect the ECU variants that are based on the selected criteria values.

Workaround
DaVinci Configurator Pro is currently limited to a single criterion. The default name is **Communication**. Despite the name (that may be changed by the user), this criterion can be used for both diagnostic and communication variance.

It is still possible to define multiple ECU variants by defining one criterion value for each variant.

2.2 Features

The following high-level features are supported:

Supported Features	
Selection of the ECU variant at startup time of the ECU. Thereby one out of N available configuration variants can be selected by the application. Implementing the logic that determines the variant to be chosen is an application responsibility.	■
The Identity Manager is supported by MICROSAR BSW modules that support the Feature **MICROSAR Identity Manager using Post-Build Selectable** (as it is listed in the technical reference of the BSW module: Chapter **Functional Description → Features**).	■
Variance of communication aspects such as signals, PDUs and their properties.	■
Variance of communication channels such as the properties and the number of channels.	■
Variance of the RTE data mapping that links the communication stack with the application (SWCs).	■
Variance of diagnostic aspects such as DTCs and DIDs as well as some of their properties.	■
Data deduplication that reduces RAM and ROM footage by eliminating redundant data from the generated BSW configuration files.	■
Combination with MICROSAR Post-Build Loadable [1]. Post-build loadable requires dedicated licensing.	■
The recommended maximum number of variants is 16. The maximum number of supported variants is 32. On request higher numbers of variants may be supported.	■

Table 2-1 Supported features

2.3 Deviations

The following features specified by AUTOSAR are currently not supported:

Not Supported Features	
Variance of SWC and service-SWCs is currently not supported. As a consequence variant SWC must provide a superset API and manage the variance internally. The application must not invoke service-SWC and BSW APIs that are not defined in the currently active variant.	-
DaVinci Configurator Pro supports a single criterion only. As this criterion can have multiple values all use-cases can be covered.	-

Table 2-2 Not supported features

3 Workflow Overview

This chapter highlights the most important aspects of the workflow that is required to realize a variant ECU. The chapter shall be seen as an overview but already provides important information on the usage of this functionality.

More details can be found in the following chapters and the online help of DaVinci Configurator Pro.

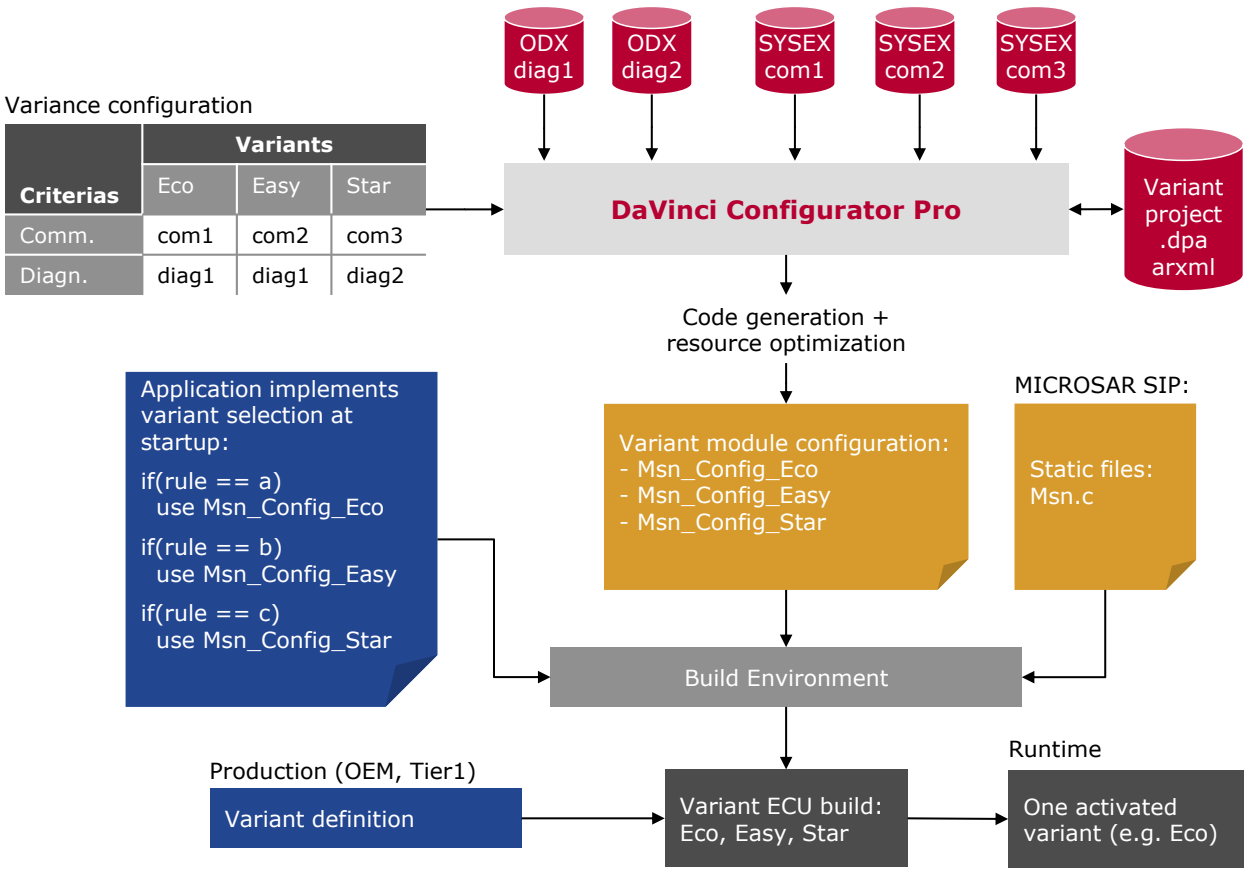

Figure 3-1 Workflow overview

3.1 Setting Up a Variant Project

Setting up a project with variance requires several steps which are summarized in Figure 3-2. The major difference between a workflow without variants is that variants need to be defined before databases are added to the configuration. For this purpose DaVinci Configurator Pro provides the **Variants** editor (see screenshot in Figure 2-1).

Adding additional variants at a later point remains possible, however.

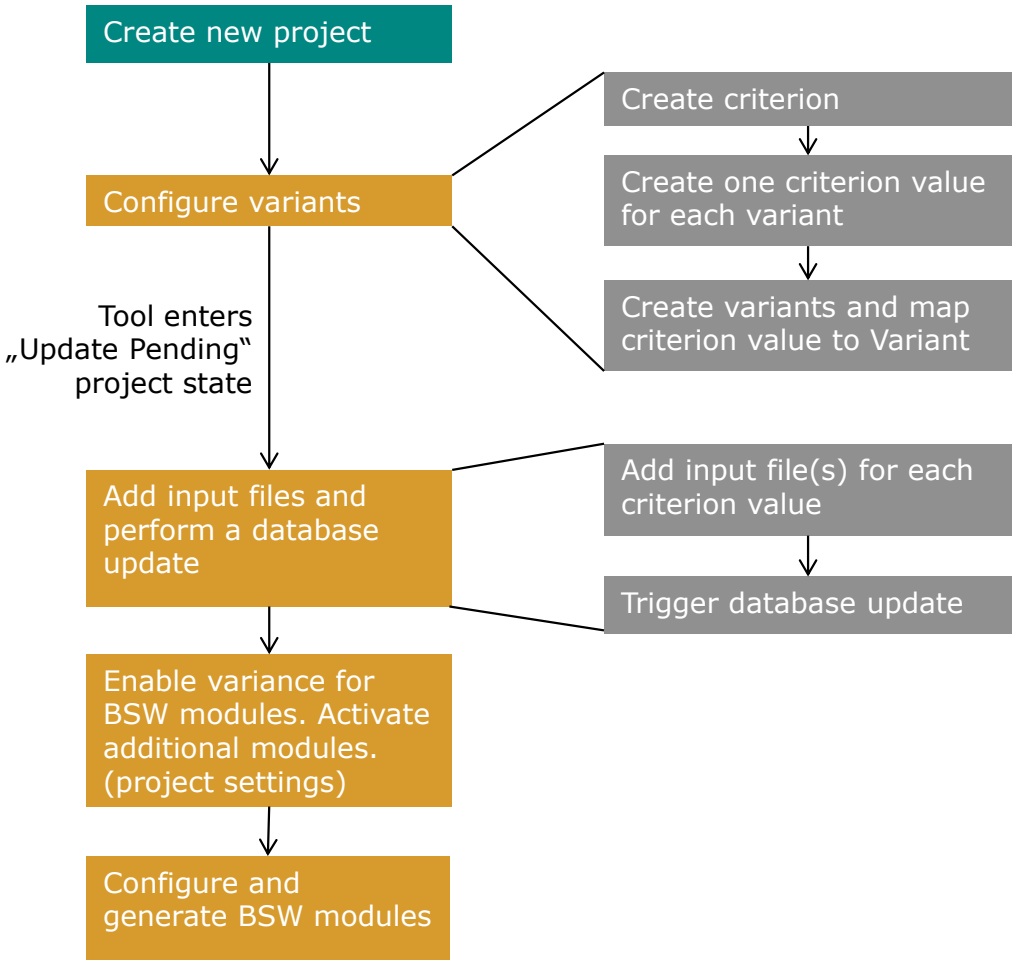

Figure 3-2 Major steps to setup a variant project with DaVinci Configurator Pro

In a next step input files are assigned to each variant. Therefore a file set of input files is created for each criteria value (e.g. Communication = Left). A file set consists of e.g. multiple legacy files (one for each channel) that is relevant for one variant or a single system extract that describes all channels of one variant.

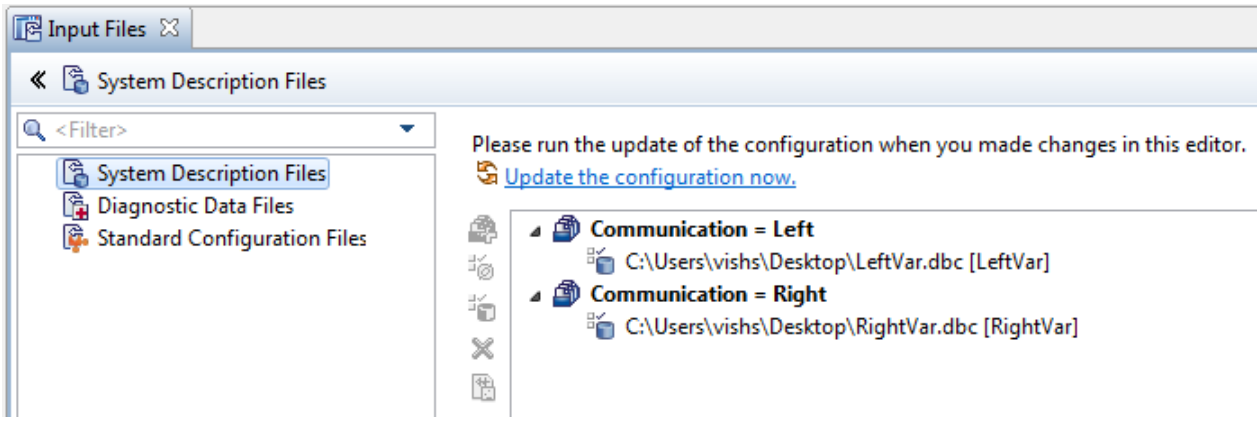

Figure 3-3 DaVinci Configurator Pro: Variant specific definition of input files

3.2 BSW Configuration

DaVinci Configurator Pro highlights variant parameter and container using a brown *[V]*. When changing the configuration it has to be considered whether the change shall be applied to all variants *[V]* or if the change shall be done only for one variant.

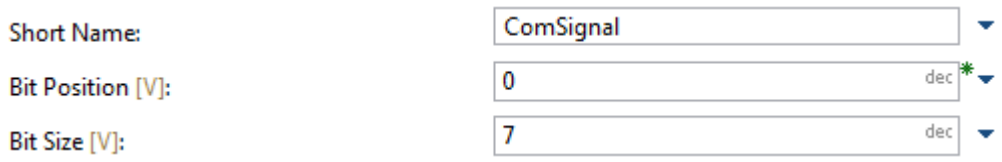

Figure 3-4 Illustration of Variant Parameters in DaVinci Configurator Pro

If required, variation points can be added to the configuration and linked with a set of criteria values that must apply to include that configuration element into one variant. If required also custom criteria and criteria values can be defined. The criteria configuration of a variation point will implicitly link the setting with one or several variants depending on the variance configuration table.

3.3 Variance in Application Code

The application code may or may not be affected by the variance in the BSW. If only the CAN IDs or Tx modes of messages are different in each variant this is handled in a transparent way for the application within the BSW.

Typically ECU variance is more complex and also involves application interfaces. For example the application may transmit or receive more signals in one variant than in the other. Equally there may be different requirements to the diagnostic handling such as one variant may have more features and thus may have more fault memory entries (DTCs) to be managed.

As soon as the interface of the BSW provides variant-dependent functionality (e.g. a signal or a DTC that exists in some of the variants only) the application will have to consider the BSW variance when dealing with BSW APIs.

Caution

The application must not invoke any APIs that are not available in the variant that is currently realized by the BSW. Even with enabled development error detection, the BSW will not be able to detect all kinds of invalid API usage!

Undefined ECU behavior can be the result of calling APIs that are not available in the active variant.

If development error detection (DET) is enabled, the MICROSAR BSW will be able to detect some faults related to accessing functionality that is not available in the activated variants. Due to technical reasons, not all faults can be detected and be handled gracefully.

When accessing BSW functionality that is not available in one variant (but in another) the following faults may happen:

> Reporting of a development error if DET is enabled.

- Activation of a BSW functionality that is not related to the API call (e.g. a different message is transmitted as requested by the API call).
- BSW API returns an error such as E_INVALID_REQUEST or E_UNINIT
- Undefined ECU behavior (with and without usage of development error detection)

The following sections give some advice, how the application can deal with variance in the BSW.

3.3.1 Variance of COM Signals

The MICROSAR RTE supports variant data mapping. This allows e.g. to map one application SWC S/R port to different signals depending on the activated variant. If required an S/R port can remain unconnected in one variant while in another variant a data mapping is configured.

When accessing COM signals directly from a complex driver, only these signals must be accessed at runtime that are configured for the activated variant.

3.3.2 Variance in SWC and BSW APIs

MICROSAR BSW modules of the service layer generate a service SWC description that describes the API in a formal way. MICROSAR service-SWCs define the BSW variance in a non-formal way by providing the API superset of all BSW variants. As a consequence the accessing application SWC must only invoke APIs that are enabled in the current BSW variant. The same applies for complex drivers that access BSW functionality directly.

Example

Both examples realize the same ECU behavior and point out how BSW variance may be considered during application development.

This is an incomplete example only. Of course there are many other ways of taking care that the application invokes only these BSW APIs that are valid in the variant that is currently enabled.

- Example 1 (SWC implementation):

```
if(Rte_IrvRead_Swc_Runnable_EcuVariant == V_ECO)
{
   /* Service mapping to DEM ErrorX configured */
   Rte_Call_DiagnosticMonitor_Port_SetEventStatus(DEM_EVENT_STATUS_PASSED);
}
else if(Rte_IrvRead_Swc_Runnable_EcuVariant == V_STAR)
{
   /* DEM ErrorX is not defined in this variant and must not be reported */
}
/* Variant Data Mapping to SigA for ECO and SigB for START managed by RTE */
Rte_Write_Swc_Port(data);
```

- Example 2 (CDD implementation):

```
if(gVariant == V_ECO)
{
```

```
   /* Report DEM ErrorX on V_ECO only */
   Dem_SetEventStatus(ErrorX, DEM_EVENT_STATUS_PASSED);
   Com_SendSignal(SigA, &data); /* SigA is defined in this variant only */
}
else if(gVariant == V_STAR)
{
   /* DEM ErrorX is not defined in this variant and must not be reported */
   Com_SendSignal(SigB, &data); /* SigB is defined in this variant only */
}
```

3.4 Validation and Code Generation

The ECU configuration is validated and generated for all configured variants at the same time.

Practical Procedure
To reduce overall complexity when setting up a new project it may be helpful to realize a single ECU variant first. After having the first variant tested, additional variants may be added and linked with the appropriate input files that are now imported for the first time to this project.

The BSW module configuration must be valid according to the BSWMD file for each variant (e.g. variant parameters with a multiplicity of 1:1 must be available in all variants but possibly with different values).

When generating code, all BSW modules that are enabled for variance (DaVinci Configurator Pro **Project Settings** editor) will produce one configuration set for each configured ECU variant. Variance may be implemented by the BSW modules as part of post-build tables (linked by the related configuration structures) or within generated source code.

Validation rules at configuration and generation time ensure that MICROSAR modules that do not support variance do not access variant data. In this case an error is thrown. To solve this issue, variance may have to be enabled for the affected BSW module (see chapter 4.1) or the variance has to be removed from the configuration.

3.5 BSW Initialization

At ECU start-up, the application has to determine the expected ECU behavior and choose the appropriate BSW configuration structures to initialize the BSW.

Determining the required variant can be done in multiple ways and is typically defined by the vehicle manufacturer. Examples are:

- External pin coding of the ECU e.g. by using specific connectors that allow the ECU to detect its variant
- Definition of a non-volatile memory pattern that may be set end of line or during the first start-up of the ECU

Caution
If the variant of the ECU shall be changed, it may be required that some hardware devices are disabled as these are no longer used in the variant to be enabled. The BSW does not provide any functionality to disable hardware functionality that had been enabled by a previous variant. Therefore it is recommended to completely restart the ECU before activating a new BSW variant. Alternatively the application may also disable hardware functionality required by the BSW prior of starting the BSW initialization.

3.5.1 Manual BSW Initialization

If for some reason the initialization through the BSWM “Module Initialization” Auto Configuration assistant is not possible the BSW initialization can also be implemented manually.

For post-build loadable modules the header `EcuM_Init_PBcfg.h` is used to publish initialization relevant data types. To access the module configuration pointers always use `EcuM_GlobalPBConfig_Ptr`.

For variant modules that do not support post-build loadable the header `EcuM_Init_Cfg.h` is used to publish initialization relevant data types. To access the module configuration pointers always use `EcuM_GlobalPCConfig_Ptr`.

```
#include "EcuM_Init_Cfg.h"
…
void MyInitFunction()
{
…
  Com_Init(EcuM_GlobalPCConfig_Ptr->CfgPtr_Com_Init);
  ComM_Init(EcuM_GlobalPCConfig_Ptr->CfgPtr_ComM_Init);
…
}
```

Table 3-1 Manual BSW initialization using the configuration structure address provided by ECUM

The `EcuM_GlobalPBConfig_Ptr` and `EcuM_GlobalPCConfig_Ptr` are initialized by the ECUM during `EcuM_Init()` with the global root structure address returned by the callout `EcuM_DeterminePbConfiguration()`.

4 Configuration

The DaVinci Configurator Pro help system (press F1 within the tool) will provide you with information how to set up a configuration that includes variants.

This chapter highlights the most important aspects only.

4.1 Activating Variance for BSW Modules

In the project settings editor of DaVinci Configurator Pro you can define each BSW module whether it shall support variance or not. It is recommended to enable variance for a complete cluster (e.g. communication stack or diagnostic stack) at once.

Basic Knowledge
As the BSW initialization sequence will handle the BSW modules variance, modules in charge for BSW initialization (BSWM and ECUM) have to support variance in any case.

To enable variance for a BSW module choose either

- `VARIANT-POST-BUILD-SELECTABLE`: Variants are supported. No post-build loadable update of the configuration at post-build time. All variants are configured at pre-compile time.
- `VARIANT-POST-BUILD (LOADABLE and SELECTABLE)`: Variants are supported and a post-build loadable update of the configuration data is supported using MICROSAR Post-Build Loadable. This feature required dedicated licensing.

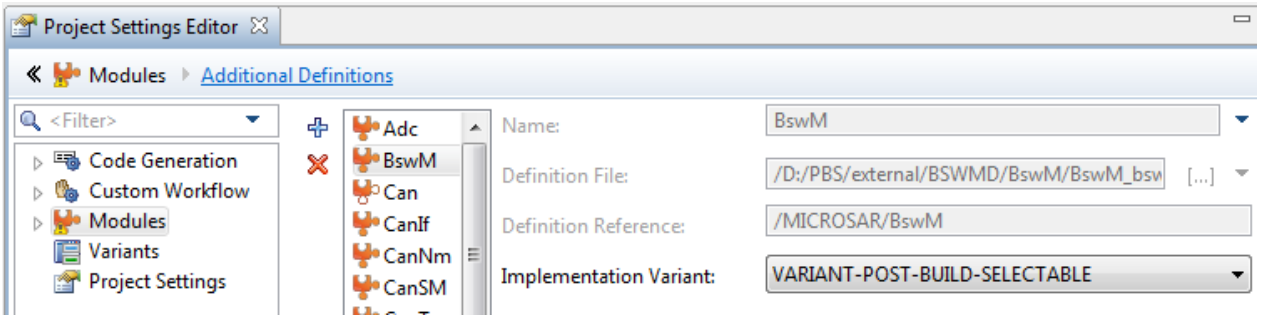

Figure 4-1 Enabling Variance for BSW Modules

Hint
You can use multi select in DaVinci Configurator Pro to enabled variance for several BSW modules at the same time.

4.2 Semantics of ECU Configuration Variance

The location of the variation point does not imply any semantic and has no effect to the generated code. Figure 4-2 illustrates two ECU configuration hierarchies which would produce the same generated code as the content of configuration is equal when looking at one variant at a time. The right version however contains more redundant data as the variation point is located at a higher level.

Example

When editing B1 in the left example the change would apply for all variants as long as no additional variation point is introduced first. When editing B1 in the right example, the change is applied only to the variant where the change is made. The other instance of B1 remains as it is.

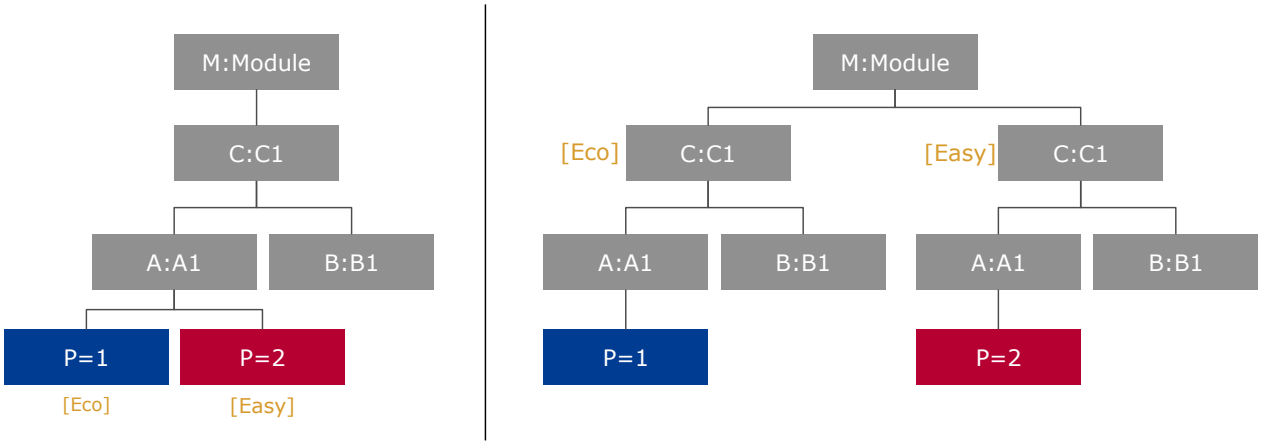

Figure 4-2 Location of variation points has no semantic meaning to the configuration

As the example shows, the location of the variation point mainly depends on the expectations what shall or shall not be equal for an object within the different variants. A general statement is therefore not possible.

Practical Procedure

If it is not clear where to create a variation point (e.g. as the requirements or the impact are not yet fully understood) it is recommended to create the variation points as low as possible in order to handle with as few redundant data as possible.

At a later point when it turns out that an object is assumed to be completely different in all variants a new variation point can be added at a higher level.

Note
The Base ECUC creation process of DaVinci Configurator Pro for example locates variation points as low as possible in the hierarchy in order to reduce the amount of redundant data within the project (left example in Figure 4-2). Having to deal with less redundant data can simplify the BSW configuration when dealing with variants that have also common settings.

5 Integration

5.1 BSW Module Initialization

During BSW initialization the application is required to provide information which variant the BSW shall implement. After ECUM initialization (that is variant independent) the ECUM will query the variant to be used from the application by using the callout `EcuM_DeterminePbConfiguration()` (chapter 5.1.1). Within this callout the application shall return the global root configuration structure (chapter 5.1.1) of the variant to be used.

The variant-specific global root structure references the configuration structures of each BSW module relevant of that variant.

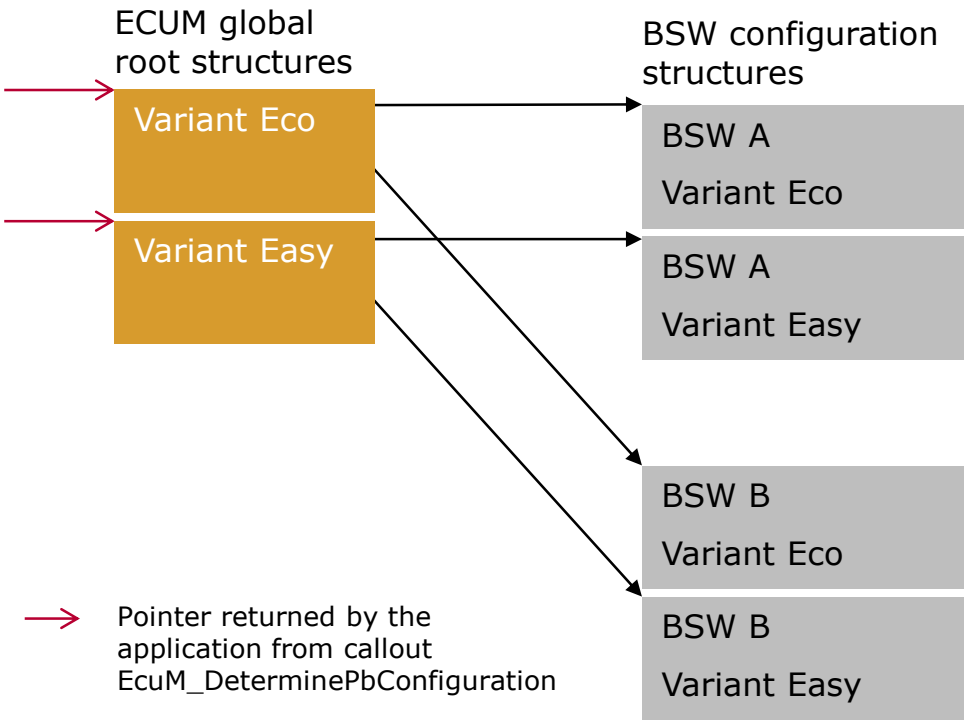

Figure 5-1 Hierarchy of BSW configuration structures for variant specific initialization

In typical BSW environments using the MICROSAR stack, no further actions or adaptations are required to the templates provided.

Reference
Please read the Technical References of ECUM ([2]) and BSWM ([3]) carefully to find out how the BSW initialization needs to be implemented in general terms.

If you also realize a post-build time update of your configuration data using MICROSAR Post-Build Loadable, please also follow [1].

5.1.1 Implementation of EcuM_DeterminePbConfiguration()

During initialization the ECUM invokes the callout `EcuM_DeterminePbConfiguration()` to determine the variant that shall be used for BSW initialization. This callout shall return the address of global root structure of the variant to be used.

To determine the address use e.g. the following code: `&EcuM_GlobalConfigRoot.<Variant>`. The symbol is published by `EcuM_Init_PBcfg.h`.

FAQ
`EcuM_DeterminePbConfiguration()` is expected to return the pointer to the global root structure (`&EcuM_GlobalConfigRoot.<Variant>`) published by `EcuM_Init_PBcfg.h` even in projects where post-build loadable is not supported.

Edit
A sample implementation of this function is provided by `EcuM_Callout_Stubs.c` as a template. Extend this implementation with the application specific rules to determine the root structure required for the variant that shall be used.

5.1.2 BSWM Module Initialization Auto Configuration

BSW modules with variance are initialized by the BSWM. To realize the BSW module initialization use the **Module Initialization** Auto Configuration assistant provided by the **BSW Management** editor of DaVinci Configurator Pro. This assistant will realize the BSW initialization as required by the Identity Manager using the global root structure as source for the BSW initialization pointers. Find more details in the BSWM technical reference [3].

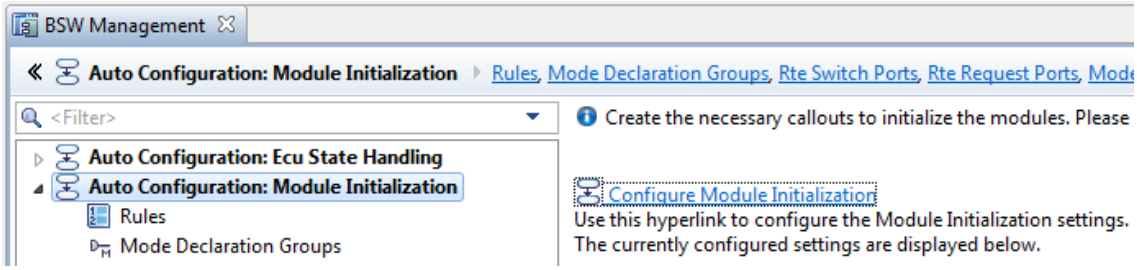

Figure 5-2 Module Initialization Auto Configuration Assistant of the DaVinci Configurator Pro BSWM configuration

5.1.3 Global Root Structure Customization

If an additional configuration pointer shall be added to the global root structure you can add the initialization details in the ECUC module configuration using DaVinci Configurator: `/MICROSAR/EcuC/EcucGeneral/BswInitialization/InitFunction`.

Caution

Within DaVinci Configurator only BSW modules that support the MICROSAR post-build loadable process must be configured as Implementation Variant VARIANT-POST-BUILD-LOADABLE or VARIANT-POST-BUILD. As a consequence these modules are added to the post-build loadable global root structure `EcuM_GlobalPBConfigRoot`.

Variant modules that do not support post-build loadable will be added to `EcuM_GlobalPCConfigRoot` instead.

5.2 Critical Sections

Critical sections are not variant-specific. As a consequence each critical section has exactly one implementation. One critical section can be used in just one variant but can also be used in several or even all ECU variants. The technical reference of the BSW modules will define what to consider when implementing the critical section handling.

5.3 Main Function Scheduling

The RTE Schedule Manager is not variant aware. Consequently the main function cycles are configured globally and are valid for all variants of the ECU.

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
MICROSAR Identity Manager	Implementation of the AUTOSAR 4 variance concept using post-build selectable and variation points in the configuration data.
DaVinci Configurator Pro	Vector´s configuration and generation tool for MICROSAR BSW and RTE
DaVinci Developer	Vector´s AUTOSAR SWC design editor

Table 6-1 Glossary

6.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CDD	Complex Drivers
DEM	Diagnostic Event Manager
DET	Development Error Tracer
IDM	MICROSAR Identity Manager
ECU	Electronic Control Unit
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com