ECU-C File Handling

Technical Reference

Version 1.13

Authors:	Michael Schuele, Matthias Wernicke
Version:	1.13
Status:	released (in preparation/completed/inspected/released)

History

Author	Date	Version	Remarks
M. Schuele	2009-02-19	0.1.0	Initial version
M. Schuele	2009-04-11	1.0.0	final version
M. Wernicke	2009-04-17	1.1	Review and update
M. Schuele	2009-08-03	1.2	Added description of difference dialog
M. Schuele	2010-01-27	1.3	Added new ECU-C parameters for DaVinci 3.0
M. Wernicke	2010-01-28	1.4	Review and update
M. Schuele	2010-05-05	1.5	Introduced different but equivalent parameter values
M. Schuele	2010-05-14	1.6	Added new ECU-C parameters for DaVinci 3.0 SP2
M. Schuele	2010-07-20	1.7	Added new ECU-C parameters for DaVinci 3.0 SP3
M. Schuele	2010-11-29	1.8	Added new ECU-C parameters for DaVinci 3.0 SP4
M. Schuele	2011-05-06	1.9	ComTimeoutFactor synchronization is configurable
M. Schuele	2012-02-02	1.10	Added new ECU-C parameters for DaVinci 3.0 SP5 and 3.1
M. Schuele	2012-08-30	1.11	Added a note about AUTOSAR 4
M. Schuele	2013-05-03	1.12	Added new ECU-C parameters for DaVinci 3.5
M. Schuele	2014-03-27	1.13	Added SchM config to Rte section

Contents

1 Overview

DaVinci Developer is part of Vector’s solution for AUTOSAR compatible ECU development. It is used to configure and generate the Rte in AUTOSAR 3.x based projects and therefore interacts with other BSW configurators through the ECU-Configuration file.

This document describes the configuration process related to DaVinci Developer from a technical point of view, trying to give the user a better understanding of the internal processes and how the tool reacts in different situations.

Note

Starting with DaVinci Developer 3.3, AUTOSAR 4.0 based software designs can be created and edited. However, the configuration and generation of the AUTOSAR 4.0 Rte has been moved from DaVinci Developer to DaVinci Configurator Pro. Therefore this document is only relevant for AUTOSAR 3.x based projects.

1.1 Intended Audience

This document aims at ECU developers who are involved in the AUTOSAR compatible ECU-Configuration process and use DaVinci Developer to configure and generate the Rte module.

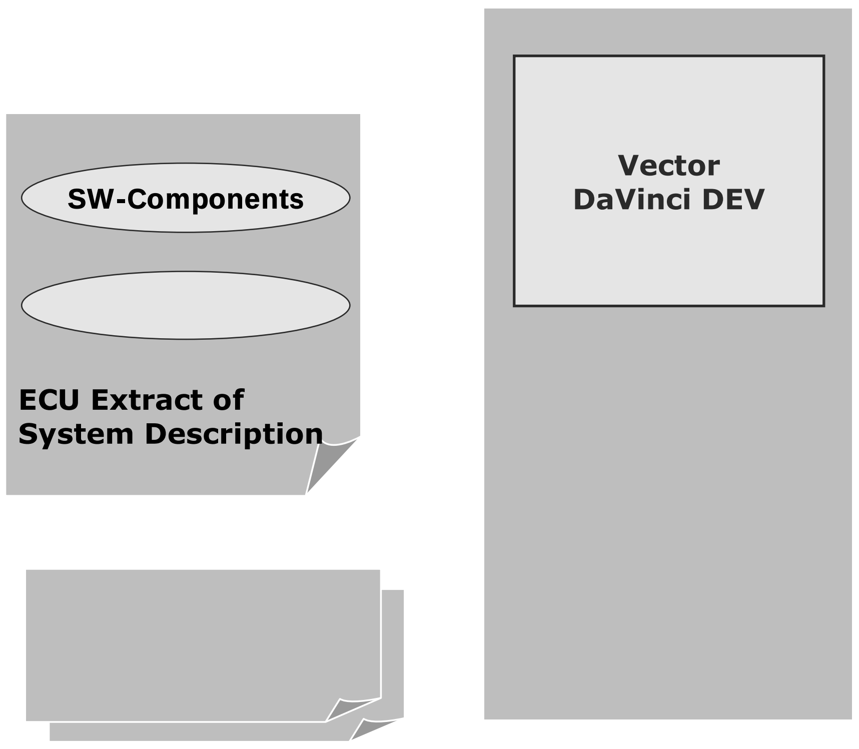
As DaVinci DEV updates the ECU-Configuration file automatically during save and load of a workspace, the presented information is not essential when working with the tools but provides some additional information how the ECU-Configuration process is handled behind the scenes.

1.2 Terms and Acronyms

Term	Definition
DaVinci DEV	DaVinci Developer
NWD	Network Designer
AR	AUTOSAR – Automotive Open System Architecture
GUI	Graphical user interface
Rte	Runtime environment
BSW	Basic Software
BSWMD	Basic Software Module Description

ECU-C file	ECU-Configuration file
ECU-C-Synchronization	Synchronization of DaVinci DEV workspace with the data in an ECU-C file

2 The ECU-Configuration Process



2.1 The ECU-Configuration File

AUTOSAR specifies all configuration parameters of a BSW module as a so called „Standardized Module Definition“. These definitions are contained in an XML-Document (AUTOSAR_EcucParamDef.arxml) and are modeled according to the „ECU Configuration Parameter Definition Meta model“.

To support vendor specific configuration parameters, BSW modules may provide a BSWMD („Vendor Specific Module Definition“) file which includes the standardized parameter definitions for the particular module and defines additional parameters. Additionally, a BSWMD may contain a section with a pre-configuration, i.e. parameter values which cannot be changed, or a section with a recommended configuration, i.e. parameter values which are appropriate for most use cases but can still be changed.

BSWMD files define which parameters are available to configure a BSW module, an example is given in Figure 2-2. The ECU-Configuration file contains the actual parameter values, i.e. the configuration of the modules used on a specific ECU, an example is given in Figure 2-1.

```
<AUTOSAR>
  <TOP-LEVEL-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>MICROSAR</SHORT-NAME>
      <ELEMENTS>
        <MODULE-DEF>
          <SHORT-NAME>Com</SHORT-NAME>
          <CONTAINERS>
            <PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>ComConfig</SHORT-NAME>
              <SUB-CONTAINERS>
                <PARAM-CONF-CONTAINER-DEF>
                  <SHORT-NAME>ComSignal</SHORT-NAME>
                  <PARAMETERS>
                    <INTEGER-PARAM-DEF>
                      <SHORT-NAME>ComBitPosition</SHORT-NAME>
                      <MAX>63</MAX>
                      <MIN>0</MIN>
```

Figure 2-2 Configuration parameter definition of “ComBitPosition” in the BSWMD file

```
<AUTOSAR>
  <TOP-LEVEL-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>MyProject</SHORT-NAME>
      <ELEMENTS>
        <ECU-CONFIGURATION>
          <SHORT-NAME>MyEcu</SHORT-NAME>
          <ECU-EXTRACT-REF DEST="SYSTEM">
              /VehicleProject/ReceiverEcu
          </ECU-EXTRACT-REF>
          <MODULE-REFS>
            <MODULE-REF DEST="MODULE-CONFIGURATION">
              /MyProject/Com
            </MODULE-REF>
          </MODULE-REFS>
        </ECU-CONFIGURATION>

        <MODULE-CONFIGURATION>
          <SHORT-NAME>Com</SHORT-NAME>
          <DEFINITION-REF DEST="MODULE-DEF">
              /AUTOSAR/Com
          </DEFINITION-REF>
          <CONTAINERS>
            <CONTAINER>
              <SHORT-NAME>ComConfig</SHORT-NAME>
              <DEFINITION-REF DEST="PARAM-CONF-CONTAINER-DEF">
                  /AUTOSAR/Com/ComConfig
              </DEFINITION-REF>
              <SUB-CONTAINERS>
                <CONTAINER>
                  <SHORT-NAME>Signal_1</SHORT-NAME>
                  <DEFINITION-REF DEST="PARAM-CONF-CONTAINER-DEF">
                      /AUTOSAR/Com/ComConfig/ComSignal
                  </DEFINITION-REF>
                  <PARAMETER-VALUES>
                    <INTEGER-VALUE>
                      <DEFINITION-REF DEST="INTEGER-PARAM-DEF">
                        /AUTOSAR/Com/ComConfig/ComSignal/ComBitPosition
                      </DEFINITION-REF>
                      <VALUE>7</VALUE>
                    </INTEGER-VALUE>
[…]
```

Figure 2-3 Value specification for configuration parameter “ComBitPosition” in the ECU-C file

The two files are connected by means of standard AUTOSAR references as defined in the specification „Model Persistence Rules for XML“. A parameter is configured in the ECU-C file by referencing its parameter definition element included in the BSWMD file (via `<DEFINITION-REF>`) and defining the actual value (highlighted in red in the given examples).

As each BSW module's parameters are defined in a separate `<MODULE-DEF>`, one `<MODULE-CONFIGURATION>` is needed for each of the BSW modules. The complete configuration of the ECU is defined by `<ECU-CONFIGURATION>` which references all relevant BSW module configurations.

All “Standardized Module Definitions” are included in a package named “AUTOSAR” and only those are allowed to be placed there. “Vendor Specific Module Definitions” must be included in a different package; Vector modules use “MICROSAR” as the package name. Since it is required that custom BSWMD files must include all standardized parameters as well, the package name is visible in all references to the parameter definitions. Hence, switching to another BSWMD requires adaption of all corresponding references in the ECU-C file.

As already mentioned, a module's configuration may depend on the settings of another module. In this case a reference parameter contains an AUTOSAR reference as its value, pointing into the configuration of the other module, e.g. the runnable mapping of the Rte module configuration references the task of the Os configuration it is mapped to via `<VALUE-REF DEST="CONTAINER">/MyProject/Os/Task1</VALUE-REF>`.

3 DaVinci DEV and the ECU-C file

As displayed in Figure 2-1, DaVinci DEV is used to configure and eventually generate an AUTOSAR compatible RTE. Obviously this requires reading from and writing to the Rte module configuration in the ECU-C file, but also to other modules, which directly interface the RTE (called depending modules within this document). Depending modules are Com, NvM and Os. These modules may be read and/or changed, too. Some of these foreign parameters are directly visible and changeable; some are automatically derived from other settings or the software design. Some parameters are needed to ensure that the generated code of different BSW generators matches regarding used and/or provided APIs, handles and other interface related code. Details about the affected parameters and their sources or relation to the Rte configuration are given in chapter 4.

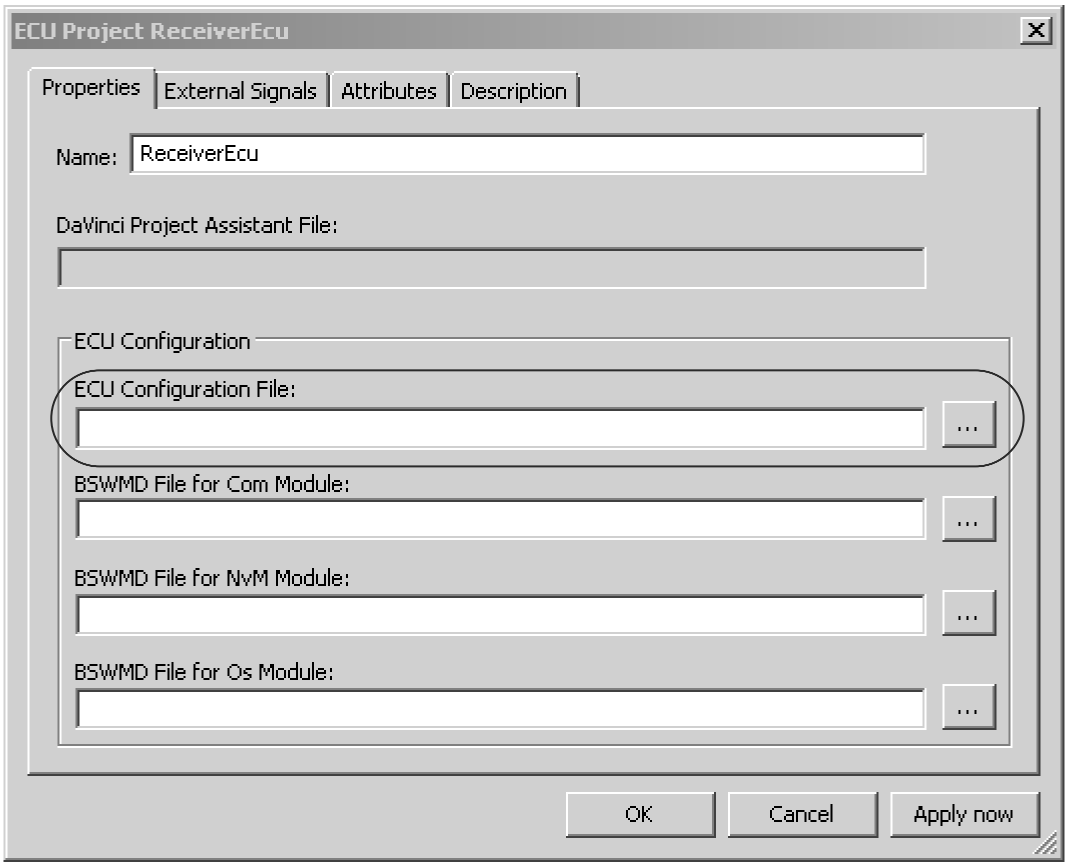
3.1 Project Assistant

The most convenient way to create a new ECU-C file is the Project Assistant (available via the menu item File | Project Assistant). The Project Assistant automatically creates one or more[1] ECU-C files based on an ECU Extract of System Description and a SIP (Software Integration Package) from Vector, and sets up an ECU-Project in DaVinci DEV.

3.2 Initial synchronization (bi-directional update)

To update an existing ECU-C file created by any third party tool without involving DaVinci DEV, you have to open the properties dialog of the ECU-Project (see Figure 3-1). The “select file” button (labeled “…”) shows a “File Open” dialog where an existing ECU-C file can be selected. The selected file is analyzed and if the configuration does not match the workspace settings, a synchronization dialog is displayed (this is explained in detail in section3.6.1). When the properties dialog is closed the ECU-C file will be assigned to the ECU-Project. To perform the synchronization process again, you can then select “Synchronize ECU-Configuration” from the context menu of the ECU-Project.

[1] Several ECU-C files are created if „one file per module“ is selected in the „Output Paths“ section.


ECU Project ReceiverEcu

Name: ReceiverEcu

DaVinci Project Assistant File:

ECU Configuration

ECU Configuration File:

BSWMD File for Com Module:

BSWMD File for NvM Module:

BSWMD File for Os Module:

Figure 3-1 ECU-C file reference in the ECU-Project properties dialog

Before the Rte generator is started, DaVinci DEV checks if the ECU-Project has an ECU-C file assigned and automatically executes the synchronization process. This ensures that the current workspace settings reflect the settings in the ECU-C file and that the ECU-C file contains the current configuration of the workspace. Both configurations must match because all code generators of the different modules have to work with exactly the same ECU configuration data.

3.3 Automatic synchronization

Once the ECU-C file is assigned to the ECU-Project it is checked for consistency during every load or save of the workspace. This ensures that other tools always work with a consistent configuration.

3.4 ECU-C file locking

The tools DaVinci Configurator, DaVinci Developer and GENy implement a locking mechanism to ensure that the user edits a specific configuration only in a single tool at a time. If the configuration is changed and not yet saved in one tool the other tools consider the configuration as read-only and display this state accordingly. This mechanism targets on a single user’s daily work, it does not support distributed development or multi-user scenarios.

3.5 BSWMD files

The ECU-Project properties dialog allows selecting the BSWMD files for each of the potentially modified modules. These files are needed for two reasons:

- Since configuration parameters are identified through an absolute reference to their parameter definition, the name of the root package in the BSWMD file has to be known.
- DaVinci DEV only exports Vector specific configuration parameters if the corresponding BSWMD file is assigned to the ECU-Project.

The selection of a specific BSWMD file for the Rte is not required because DaVinci DEV only configures the Vector's MICROSAR RTE, which is based on the standard AUTOSAR BSWMD file of the RTE.

If the ECU-C file follows the AUTOSAR Releases 2.1 Specification, DaVinci DEV uses the standard module definitions, `<DEFINITION-REF>` values therefore begin with "`/AUTOSAR`".

If the ECU-C file follows the AUTOSAR Releases 3.x Specification, DaVinci DEV implicitly uses the Rte BSWMD file, `<DEFINITION-REF>` values therefore begin with "`/MICROSAR`".

If there's a mismatch between the configured BSWMD file and the module configuration in the ECU-C file (e.g. the BSWMD file's package name is `MICROSAR` and the ECU-C file's module configuration references an `AUTOSAR` package), the definition of the ECU-C file is used. A message about this misconfiguration is written to the "Action Log" window saying:

ActionLog output

```
Inconsistent BSWMD configuration detected:
  /MICROSAR/Os is defined by BSWMD file D:\BSWMD_files\Os_bswmd.arxml
  /AUTOSAR/Os is used by ECU-C file D:\ECU_1_ecuc.arxml

using ECU-C definition
```

The misconfiguration can be detected because the module names are standardized, i.e. the `<DEFINITION-REF>` of a `<MODULE-CONFIGURATION>` always follows the rule `/[MODULE-DEF-PACKAGE-NAME]/[standardized module name].`

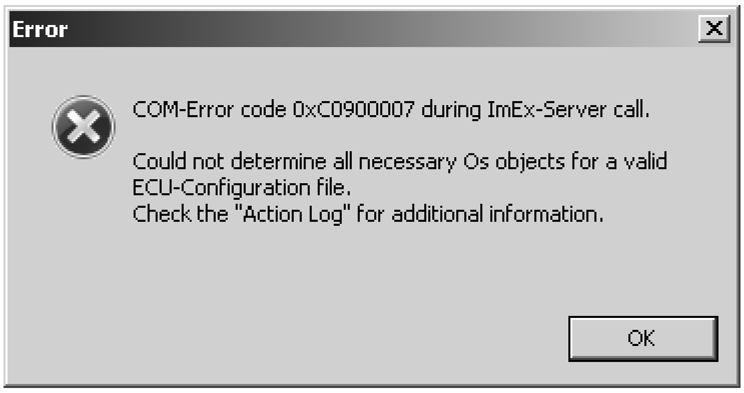
3.5.1 Pre- and Recommended config sections

DaVinci DEV in general does not evaluate Pre- and/or Recommended Module Configuration sections in the BSWMD or any other file. However, one exception exists for the Os where a preconfigured OsCounter is set as an Alarm's OsAlarmCounterRef by default (see 4.5.1).

3.6 Synchronizing an ECU-C file

Synchronizing an ECU-C file is necessary if the configuration of the RTE does not match to the configuration of the depending modules (see section 4 for a detailed description of the parameters for each module). The synchronization process can be divided into severel steps which will be explained in the following sections.

3.6.1 Step 1: Analysis of the RTE configuration in the workspace


Error

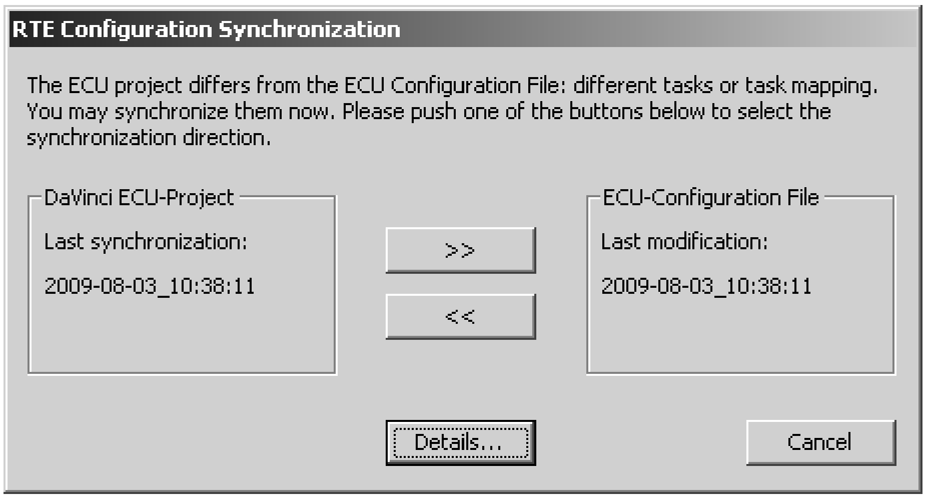
COM-Error code 0xC0900007 during ImEx-Server call.

Could not determine all necessary Os objects for a valid ECU-Configuration file.
Check the "Action Log" for additional information.

Figure 3-3 The Rte generator detected an inconsistent system design

3.6.2 Step 2: Comparison of workspace and ECU-C file

If all parameters could be derived, the given ECU-C file is imported and the settings of the workspace are compared to the settings of the ECU-C file. If differences were detected the dialog in Figure 3-4 is shown. The user can then select whether the settings of the ECU-C file should be imported (direction “<<”), i.e. the ECU-Project should be changed according to the settings in the ECU-C file, or if the settings of the ECU-Project should be exported, i.e. the ECU-C file should be changed according to the settings in the ECU-Project (direction “>>”). Remember that an export might remove existing Os elements from the ECU-C file if the ECU-Project is inconsistent but the previously shown dialog (Figure 3-2) was closed with “Yes” (“export anyway”).


RTE Configuration Synchronization

The ECU project differs from the ECU Configuration File: different tasks or task mapping. You may synchronize them now. Please push one of the buttons below to select the synchronization direction.

DaVinci ECU-Project
Last synchronization:
2009-08-03_10:38:11

ECU-Configuration File
Last modification:
2009-08-03_10:38:11

Figure 3-4 ECU-C-Synchronization dialog

On the left side the dialog displays the timestamp of the last synchronization of the ECU-Project, and on the right side the “last modified” timestamp of the ECU-C file is displayed. Since both the ECU-C file and the ECU-Project can be edited in parallel there’s no rule of thumb to decide which synchronization direction has to be chosen without considering additional information. Therefore the “Details…” button shows a dialog with the actual differences of both configurations. Section 3.6.5 describes the dialog in detail.

Chapter 5 shows some practical approaches, which makes it easier for the user to avoid errors like selecting the wrong synchronization direction or to run a synchronization process even though it is not required to do so.

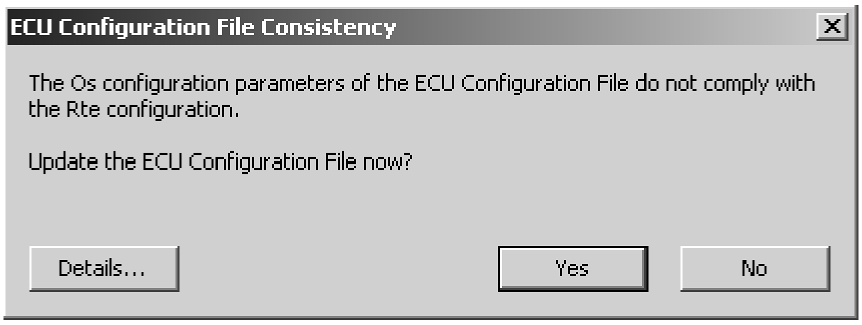
Chapter 4 shows in details, which parameters are affected or relevant for DaVinci DEV.

The next sections describe what happens if the user selects “import” or “export”.

3.6.3 Synchronization direction “import”

If “import” is selected, the ECU-Project is changed according to the settings of the ECU-C file. If one of the modules Rte, Os, Com, NvM is not configured in the ECU-C file the user is asked if he wants to delete the corresponding settings in the ECU-Project as well or if they should be kept as they are. This is useful if the user already added a task mapping in DaVinci DEV but did not yet configure the Os module. When the import has completed the new ECU-Project settings are analyzed again and the resulting configuration parameters are compared to the ECU-C file settings. This is necessary because changes of the Rte configuration might result in additional changes of the Com or Os configuration. If this is the case the user is asked if he wants to update the corresponding module configuration in the ECU-C file or not (Figure 3-5). Depending on the user’s role he might be allowed to change the Os or Com configuration or he has to confer with the responsible person at first.

Again, the “Details…” button allows to further investigate the actual differences between the ECU-Project’s implicit configuration and the settings from the ECU-C file.


ECU Configuration File Consistency

The Os configuration parameters of the ECU Configuration File do not comply with the Rte configuration.

Update the ECU Configuration File now?

Figure 3-5 Module specific ECU-Configuration update dialog

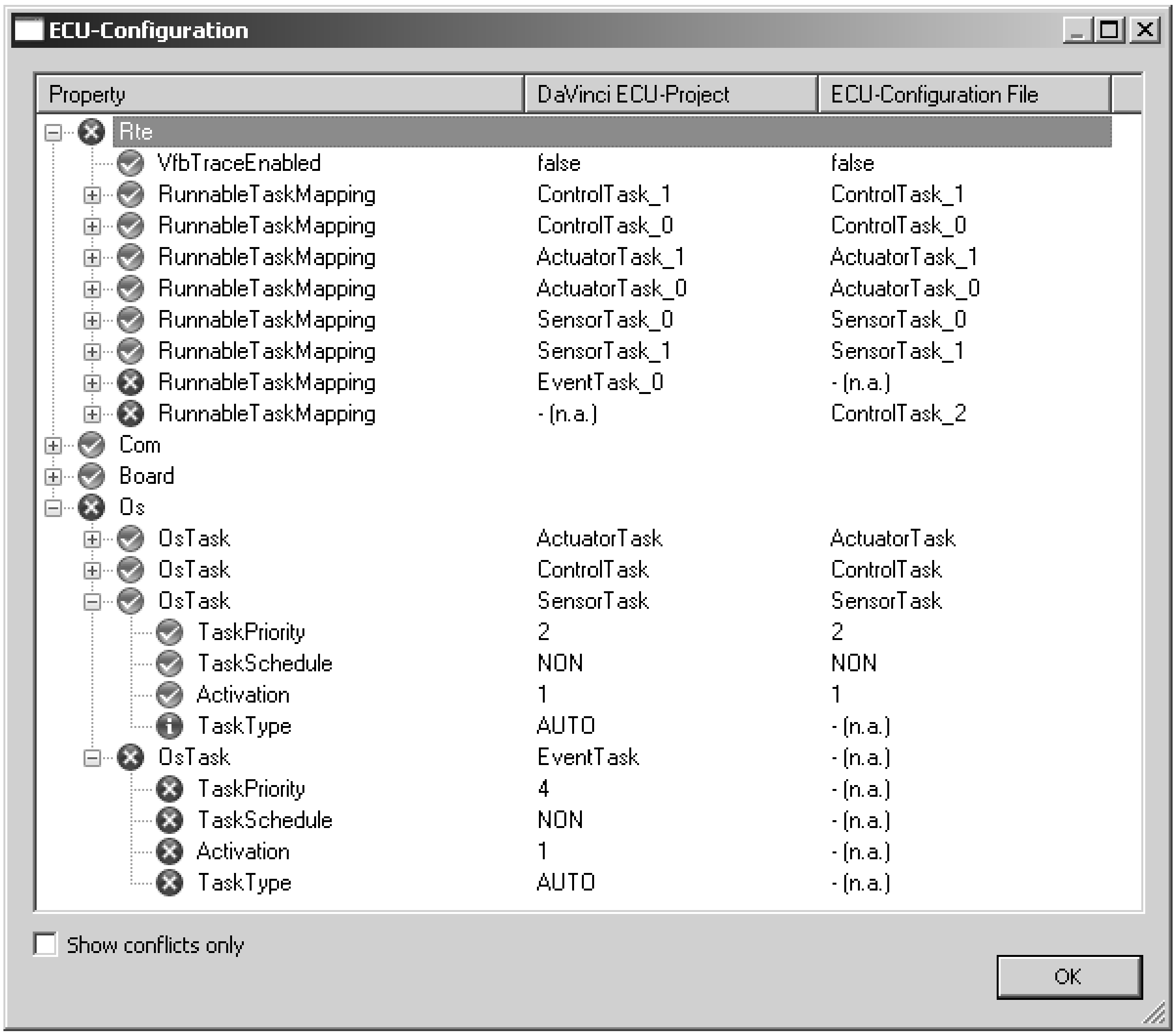
3.6.4 Synchronization direction “export”

If “export” is selected, the ECU-C file is changed according to the settings of the ECU-project. This means that the explicit parameters are written to the file. Then it is checked if other module configurations have to be updated with implicit parameters, e.g. ComCallbacks or Os elements. If this is the case, DaVinci DEV asks the user to decide if the corresponding module configuration in the ECU-C file should be updated or not (Figure 3-5).

Again, the “Details…” button allows to further investigate the actual differences between the ECU-Project’s implicit configuration and the settings from the ECU-C file.

3.6.5 ECU-Configuration difference dialog

The ECU-Configuration difference dialog helps the user to decide about his further actions whenever the configurations of the ECU-Project and the ECU-C file are inconsistent. It may be opened from the ECU-C synchronization dialog described in section 3.6.2 as well as the dialogs from sections 3.6.3 and 3.6.4 which are displayed for each module configuration.



ECU-Configuration

Property	DaVinci ECU-Project	ECU-Configuration File
✗ Rte		
✓ VfbTraceEnabled	false	false
✓ RunnableTaskMapping	ControlTask_1	ControlTask_1
✓ RunnableTaskMapping	ControlTask_0	ControlTask_0
✓ RunnableTaskMapping	ActuatorTask_1	ActuatorTask_1
✓ RunnableTaskMapping	ActuatorTask_0	ActuatorTask_0
✓ RunnableTaskMapping	SensorTask_0	SensorTask_0
✓ RunnableTaskMapping	SensorTask_1	SensorTask_1
✗ RunnableTaskMapping	EventTask_0	- (n.a.)
✗ RunnableTaskMapping	- (n.a.)	ControlTask_2
✓ Com		
✓ Board		
✗ Os		
✓ OsTask	ActuatorTask	ActuatorTask
✓ OsTask	ControlTask	ControlTask
✓ OsTask	SensorTask	SensorTask
✓ TaskPriority	2	2
✓ TaskSchedule	NON	NON
✓ Activation	1	1
ℹ TaskType	AUTO	- (n.a.)
✗ OsTask	EventTask	- (n.a.)
✗ TaskPriority	4	- (n.a.)
✗ TaskSchedule	NON	- (n.a.)
✗ Activation	1	- (n.a.)
✗ TaskType	AUTO	- (n.a.)

Figure 3-6 ECU-Configuration difference dialog

The dialog displays the relevant module configurations and a status icon indicating if there are inconsistencies (✗) or if the settings in the ECU-Project and the ECU-C file are equal (✓) or equivalent (ℹ). Values which are not set in the ECU-Project (either explicit or implicit parameters) are not displayed in the dialog unless it is required that these values have to be removed from the ECU-C file as well.

If there are only small changes in a complex configuration the “Show conflicts only” option may be used to reduce the displayed parameters to those which are considered inconsistent.

Figure 3-6 shows an example where a new task “EventTask” was added in DaVinci DEV and a Runnable was moved from “ControlTask” to the new task. The SensorTask’s attribute “TaskType” is displayed as equivalent because “AUTO” is the default value if no specific attribute value is set in the ECU-C file.

4 ECU-Configuration parameters

The following sections list the parameters and/or containers which are read and/or written by DaVinci DEV. All other parameters/containers in the ECU-C file do not affect the RTE. They can be changed in the ECU-C file without need of synchronizing the workspace.

For some parameters it is allowed that the ECU-Project and the ECU-C file contain different values. These parameters and the corresponding rules are given in the following sections as well. Remember these values are not displayed in the difference dialog if the ECU-Project's parameter has no specific value.

4.1 Vendor Specific Configuration Parameters

Vendor specific configuration parameters for Microsar Modules are specified in each module's BSWMD file which is named <Modulename>_bswmd.arxml. These parameters are marked with (MICROSAR) in the following sections.

4.2 Rte module

4.2.1 Parameters

Rte module configuration		
Parameter	Read/Write	
RteVfbTrace	r/w	
RteVfbTraceFunction	r/w	
RteA2LVersion (MICROSAR)	r/w	
RteXcpEventSupport (MICROSAR)	r/w	
RteMeasurementSupport	r/w	
RteCalibrationSupport	r/w	
RteCalibrationBufferSize	r/w	
RteOptimizationMode	r/w	
RteTaskConfiguration (MICROSAR)	r/w	
RteTaskUsesSchedule (MICROSAR)	r/w	
RteTaskRef (MICROSAR)	r/w	
SwComponentInstance/	r/w	
ImplementationRef	w	

ServiceComponentPrototypeRef	w	
SoftwareComponentPrototypeRef	w	
ExclusiveAreaImplementation	r/w	
NvRamAllocation/	r/w	
SwNvRamMappingReference	r/w	
NvmBlockRef	r/w	
RunnableEntityMapping/	r/w	
ActivationOffset	r/w	
PositionInTask	r/w	
RTEEventRef (resolved to Runnable)	r/w	
MappedToTaskRef	r/w	
RteCyclicTriggerImplementation (MICROSAR)	r/w	
ComponentTypeCalibration/		
CalibrationSupportEnabled	r/w	

Table 4-1 Rte configuration parameters

These settings are configured in DaVinci DEV at ECU-Project level.

Please note:

- The implementation selection is not read since DaVinci DEV always assumes that only one implementation per atomic component type exists.
- RteCyclicTriggerImplementation is not visible in DaVinci DEV and cannot be modified.
- RteTaskConfiguration settings are displayed in the Os section in the ECU-Configuration difference dialog (3.6.5).

4.3 SchM module

4.3.1 General

The SchM module configuration is never changed by DaVinci DEV; it is only used as the read-only master configuration for RunnableEntityMappings of ServiceComponentTypes and the “Role” attribute of an OsTask.

4.3.2 Parameters

SchM module configuration		
Parameter	Read/Write	
SchMMainFunctionMapping/	r	
SchMMainFunctionSymbol	r	
SchMBswActivationOffset	r	
SchMMappedToTask	r	
SchMPositionInTask	r	

Table 4-2 SchM configuration parameters

4.3.3 Rte

The SchM module configuration contains a task mapping for BSW functions similar to the RTEEvent mapping in the Rte module configuration. If these BSW functions are represented by a Runnable of a corresponding ServiceComponentType the mapping of its RTEEvents to an OsTask has to be added to the Rte module configuration.

If DaVinci DEV detects a SchMMainFunctionMapping for a Runnable[2] of a ServiceComponentType used in the ECUProject, a corresponding virtual RunnableEntityMapping is derived automatically. The mapping is called virtual because it is only displayed in the ECU-Configuration difference dialog and not automatically added to the Rte ECU-Configuration section. This has to be done by the usual synchronization mechanism, i.e. the new RunnableEntityMapping has to be imported in DaVinci DEV and exported to the Rte ECU-Configuration section.

If the Runnable of the ServiceComponentType is already mapped to another OsTask in DaVinci DEV, the mapping will be adapted to ensure both SchM and the Rte use the same OsTask. As the actual position of a runnable within a task is not used by the Rte generator in this context, DaVinci DEV does not modify PositionInTask if it differs from SchMPositionInTask.

Whenever DaVinci DEV adds a virtual RunnableEntityMapping or modifies the OsTask of an existing RunnableEntityMapping a log message is written to the Action Log.

4.3.4 Os

DaVinci DEV sets the “Role” attribute of an OsTask to “BSW Scheduler” if it is referenced by any SchMMainFunctionMapping container.

[2] the Runnable is found by matching its name with the value of SchMMainFunctionSymbol

4.4 Com module

4.4.1 Parameters

Com module configuration		
Parameter	Read/Write	
ComSignal/	w	
ShortName	(r)	
ComDataInvalidAction	w	
ComInvalidNotification	w	
ComSignalDataInvalidValue	w	
SystemTemplateSystemSignalRef	(r)/w	
ComSignalInitValue	w	
ComErrorNotification	w	
ComNotification	w	
ComTimeoutNotification	w	
ComTimeoutFactor	w	
ComSignalType	w	
ComFilter/		
ComFilterAlgorithm	w	
ComFilterMask	w	
ComSignalGroup/	w	
ShortName	(r)	
ComDataInvalidAction	w	
ComErrorNotification	w	
ComInvalidNotification	w	
ComNotification	w	
ComTimeoutFactor	w	
ComTimeoutNotification	w	
SystemTemplateSignalGroupRef	(r)/w	

ComGroupSignal/	(r)/w	
ShortName	(r)	
ComSignalDataInvalidValue	w	
ComSignalInitValue	w	
ComSignalType	w	
SystemTemplateSystemSignalRef	(r)/w	
ComFilter/		
ComFilterAlgorithm	w	
ComFilterMask	w	

Table 4-3 Com configuration parameters

For simplicity, only the term “ComSignal” is used in the following explanation but the basic principle is true for ComSignalGroups, too.

DaVinci DEV assumes that the same communication database has been used to setup the workspace as well as the ComSignals in the ECU-C file (which is done by the BSW Configuration Tool by importing e.g. a dbc file or an AUTOSAR ECU Extract of System Description). Therefore, DaVinci DEV does not import ComSignals.

“(r)” means that either the ShortName or the SystemTemplateSignalRef is read to identify the ComSignal and store its name to be able to use correct ComSignal handles in the generated Rte code.

Please see section 4.8 for details about ComSignals and usage of their symbolic handles.

4.4.2 Equivalent parameter values

Com module configuration	
Parameter	Equivalence rule
ComSignal/	
ComDataInvalidAction	Parameter values “NONE” and “” (undefined attribute value) are equivalent
ComInvalidNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback
ComSignalInitValue	ECU-Project parameter is not set (InvalidAction is not set to “Replace”)
ComErrorNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback
ComNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback
ComTimeoutNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback

ComTimeoutFactor	ECU-C file value is greater than 0 but less than ECU-Project value Starting with DaVinci Developer 3.0 SP3 this depends on the workspace setting “SWC Alive Timeout overrides ComSignal” (see 4.4.2.1)
ComSignalGroup/	
ComDataInvalidAction	Parameter values “NONE” and “” (undefined attribute value) are equivalent
ComErrorNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback
ComInvalidNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback
ComNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback
ComTimeoutFactor	ECU-C file value is greater than 0 but less than ECU-Project value Starting with DaVinci Developer 3.0 SP3 this depends on the workspace setting “SWC Alive Timeout overrides ComSignal” (see 4.4.2.1)
ComTimeoutNotification	ECU-Project parameter is set but ECU-C file contains a non-Rte callback
ComGroupSignal/	
ComSignalInitValue	ECU-Project parameter is not set (InvalidAction is not set to “Replace”)

Table 4-4 Com equivalent parameter value rules

If a ComSignal’s callback is already set to a non-Rte callback this configuration is accepted but a message is displayed which contains the ComSignal name and the callback name. In this configuration the user is in charge to ensure that the Rte’s callback function is called.

4.4.2.1 ComTimeoutFactor

Depending on the design approach, the ComTimeoutFactor may be specified by the communication design or should be updated based on the software design. To support both strategies, a workspace specific setting allows specifying how to synchronize the ComTimeoutFactor: “never”, only if the value of the software design is lower than the current ComTimeoutFactor value or “always”.

4.5 Os module

4.5.1 Parameters

The column “Category“ of the following table specifies if the user can edit the corresponding parameter (“DEV“) or if it is automatically derived by the Rte code generator (“Rte generator“).

Os module configuration		
Parameter	Read/Write	Category

OsAlarm/	w	Implicit
OsAlarmAccessingApplication	w	Implicit
OsAlarmAction/		Implicit
OsAlarmActivateTask/		Implicit
OsAlarmActivateTaskRef	w	Implicit
OsAlarmSetEvent/		Implicit
OsAlarmSetEventRef	w	Implicit
OsAlarmSetEventTaskRef	w	Implicit
OsAlarmCounterRef	w	Explicit
OsEvent/	w	Implicit
OsEventMask	w	Implicit
OsApplication/	w	Explicit / Implicit
OsTrusted	r/w	Explicit / Implicit
OsAppAlarmRef	w	Implicit
OsAppResourceRef	w	Implicit
OsAppTaskRef	w	Implicit
OsApplicationTrustedFunction/		Implicit
OsTrustedFunctionName	w	Implicit
OsApplicationParams (MICROSAR)	w	Implicit
OsApplicationReturnType (MICROSAR)	w	Implicit
OsApplicationGenerateStub (MICROSAR)	w	Implicit
OsCounter	w	Explicit
OsResource/	w	Implicit
OsResourceProperty	w	Implicit
OsTask/	r/w	Explicit
OsTaskPriority	r/w	Explicit
OsTaskSchedule	r/w	Explicit

OsTaskTYPE (MICROSAR)	r/w	Explicit
OsTaskActivation	w	Implicit
OsTaskAutostart/		Implicit
OsTaskAppModeRef	r/w	Implicit
OsTaskAppMode	w	Implicit
OsTaskEventRef	w	Implicit
OsTaskResourceRef	w	Implicit
OsTaskAccessingApplication	w	Explicit / Implicit
AdminData/		
DV_TaskTimingPreConfig (DaVinci DEV)	r/w	Explicit
DV_TaskTimingCycleTime (DaVinci DEV)	r/w	Explicit
DV_TaskTimingOffset (DaVinci DEV)	r/w	Explicit

Table 4-5 Os module parameters

Most of the Os objects are implicit, i.e. not directly visible and editable in DaVinci DEV. Only OsTask and OsApplication can be changed by the user. All other objects (OsAlarm, OsEvent, OsResource) are implicit. They reflect the implementation of the MICROSAR RTE.

The OsApplication is classified as “Explicit / Implicit”, because an additional implicit special OsApplication named “Rte” might be required beside the explicitly defined OsApplications. The existence of this implicit OsApplication depends on runnable and task mapping configuration.

Depending on the Os module version the configuration parameter OsTaskTYPE is stored in different configuration container layouts; therefore the module’s BSWMD file has to be configured at the ECU-Project to ensure that the parameter is written in the correct layout.

All implicit Os elements are prefixed with “Rte_”. Their name must not be changed.

OsCounter and OsAlarmCounterRef are only exported if a BSWMD is configured which contains a pre-config section with an OsCounter. The OsAlarmCounterRef is set to this OsCounter by default but may be changed by the user in the Os configuration tool.

The optional TaskTiming configuration is used by DaVinci DEV to check for consistent cyclic runnable triggers and is not evaluated by the Os module.

4.5.2 Equivalent parameter values

Os module configuration	
Parameter	Equivalence rule
OsAlarm/	
OsAlarmCounterRef	No Os-BSWMD file is referenced or the user has configured a specific OsCounter.
OsTask/	
OsTaskTYPE (MICROSAR)	ECU-C file does not contain a value and no Os-BSWMD file is referenced by the ECU-Project which contains the parameter definition
OsTaskAutostart	ECU-Project parameter is not set

Table 4-6 Os equivalent parameter value rules

4.6 NvM module

4.6.1 Parameters

NvM module configuration		
Parameter	Read/Write	Category
NvmBlockDescriptor/	r/w	Explicit
NvmNvBlockLength	w	Implicit/Explicit
NvmRamBlockDataAddress	w	Implicit
NvmRomBlockDataAddress	w	Implicit
NvmNvramBlockIdentifier	r/w	Explicit
NvmBlockUseSyncMechanism	w	Implicit
NvmWriteRamBlockToNvCallback	w	Implicit
NvmReadRamBlockFromNvCallback	w	Implicit

Table 4-7 NvM configuration parameters

To be able to do a memory mapping in DaVinci DEV, NvMBlocks are imported from an ECU-C file. However, other attributes cannot be changed and the above mentioned attributes are automatically derived from an existing memory mapping during ECU-C-Synchronization.

4.6.2 Equivalent parameter values

NvM module configuration	
Parameter	Equivalence rule
NvmBlockDescriptor/	
NvmNvBlockLength	ECU-Project parameter is not set (no MemoryMapping for NvMBlock)
NvmRamBlockDataAddress	ECU-Project parameter is not set (no MemoryMapping for NvMBlock)
NvmRomBlockDataAddress	ECU-Project parameter is not set (no MemoryMapping for NvMBlock)
NvmNvramBlockIdentifier	ECU-Project parameter is not set but the ECU-C file contains a value
NvmBlockUseSyncMechanism	ECU-Project parameter is not set to “true” or the Nvm BSWMD does not specify this parameter
NvmWriteRamBlockToNvCallback	ECU-Project parameter is set but ECU-C file contains a non-Rte callback or the Nvm BSWMD does not specify this parameter
NvmReadRamBlockFromNvCallback	ECU-Project parameter is set but ECU-C file contains a non-Rte callback or the Nvm BSWMD does not specify this parameter

Table 4-8 NvM equivalent parameter value rules

4.7 Board

4.7.1 Parameters

Board module configuration		
Parameter	Read/Write	Category
BoardGeneral/		
BoardAtomicVariableAccess (MICROSAR)	r	(Explicit)
BoardEnableSnvPrefixes (MICROSAR)	r	(Explicit)

Table 4-9 Board configuration parameters

The parameters are marked as “(Explicit)” because it is not visible in DaVinci DEV but the Rte generator adapts the Rte code according to the parameter’s value.

4.7.2 Equivalent parameter values

Board module configuration	
Parameter	Equivalence rule
BoardGeneral/	
BoardAtomicVariableAccess (MICROSAR)	ECU-C file does not contain a value and ECU-Project parameter is set to default value “Atomic16BitAccess”
BoardEnableSnvPrefixes (MICROSAR)	ECU-C file does not contain a value and ECU-Project parameter is set to value “false”

Table 4-10 Board equivalent parameter value rules

4.8 ComSignals and ComSignalGroups

Since the Rte utilizes the Com module to transfer application data over the communication bus it has to know the mapping of SystemSignals to ComSignal handles defined by the Com module. If the handles are completely different or even worse existing handles are used for the wrong SystemSignal the result may be a simple compile error or in the worst case the problem is only visible at run time. In general, it is not possible to always use the default approach and use the SystemSignal’s name as the ComSignal handle because the Rte has to support Gateway signals (receive) and FanOut signals (write). In this case, more than one ComSignal exists for each SystemSignal and the Rte has to call the Com_SendSignal-API for each of these ComSignals with the correct handle.

To ensure that the Rte knows about the correct ComSignal handles, there are different approaches depending on the AUTOSAR version and the source of the communication data.

4.8.1 dbc files

The signals in the dbc file are imported as SystemSignals and the ComSignal handles are derived from their name. In case of Gateway/FanOut signals the dbc file contains signal names following the rule <SystemSignal>_ISig_<number> to be able to identify the original SystemSignal and to provide a unique name for each of the ComSignals on the different communication buses.

When such a dbc file is imported in DaVinci DEV one SystemSignal is created for each unique SystemSignal name and the complete signal name, e.g. SigMyData_ISig_0, is internally stored to be able to find the correct ComSignal during ECU-C-Synchronization.

NWD creates dbc files with these signal names if the export option “Create GENy compatible signal names” is activated.

4.8.2 ECU-Extract

An ECU-Extract is imported as it is, i.e. SystemSignals are imported as SystemSignals and no special naming rules apply for these signals.

If AUTOSAR 3.0 is used there's one limitation: The AUTOSAR ShortName path of a SignalToIPDUMapping has to follow the rule:
`DaVinci/PKG_Cluster<ClusterName>/<PDUName>/<SystemSignalName>_S` for Signals
`DaVinci/PKG_Cluster<ClusterName>/<PDUName>/<SystemSignalName>_SG` for SignalGroups
for CAN `<PDUName>` has to be the name of the CAN frame

ECU-Extracts of DaVinci DEV and NWD follow this rule.
If an ECU-Extract is used with a different package structure and/or naming of the relevant elements it is usually not possible to match the ComSignals of the ECU-C file with the SystemSignals of DaVinci DEV's workspace.

4.8.3 AUTOSAR 2.1

Since there is no explicit relation of a ComSignal to the corresponding SystemSignal the ECU-Synchronization identifies matching pairs of ComSignal/SystemSignal by their name.

4.8.4 AUTOSAR 3.x

With AUTOSAR 3.0 an indirect relation from ComSignal to the SystemSignal was introduced. The ComSignal references a SignalToIPduMapping which itself eventually references the SystemSignal.
To be able to resolve this reference from the ECU-Configuration into the ECU-Extract, the naming rule described in section 4.8.2 has to be fulfilled.

4.8.5 Relevant ComSignals

It is important to remember that the Rte does not need to know all ComSignals and their handles. Only those SystemSignal/ComSignal relations are relevant which are used by applications, i.e. where a data mapping of DataElements to SystemSignals exists. If attributes of other ComSignals or attributes which are not handled by DaVinci DEV are modified it is not necessary to perform an ECU-C-Synchronization.
To support the dbc based Com module configuration in a Com configurator like GENy, DaVinci DEV derives certain ComSignal attributes from the ECU-Extract because they are not contained in a dbc file, e.g. ComSignalType. These attributes are written to the ECU-C file during ECU-C-Synchronization. This causes modification of ComSignals which are not data mapped to a DataElement.

4.8.6 Callbacks

Depending on the configuration the Rte needs to be informed by the Com module about certain events, e.g. when a signal is received. The Rte's callback function name is set by DaVinci DEV at the corresponding ComSignal if the configuration parameter is not set. In case there is already a callback configured DaVinci DEV displays a corresponding warning message in the Messages window with all ComSignals, configured callbacks and additionally needed callbacks. In this scenario the user has to call the Rte function manually.

Message Details

```
ECUProject ECU1 contains the following ComSignals/ComSignalGroups with non-Rte
callbacks:

ComSignal Com_Signal_Temp_EnvData:

CallbackName
Needed Rte callback: Rte_COMCbk_Com_Signal_Temp_EnvData
Assigned callback  : Custom_Callback_Com_Signal_Temp_EnvData
```

The same mechanism is used for Nvm callbacks.

5 Best practices

Since the process of ECU-Configuration is rather complex you should follow some basic rules given in this chapter. These rules should be seen as a guideline to avoid unnecessary ECU-C-Synchronization processes in DaVinci DEV and nevertheless ensure a consistent overall configuration.

5.1 Always work on the latest communication databases

Always ensure that you are working on the latest communication databases in all tools. It is often a problem that updated dbc files are imported in a BSW configuration tool like GENy but not in DaVinci DEV. If both tools work on a different communication model it is very likely that they derive a different set of ComSignals and report various errors.

Example Error Message:

```
ComSignal with an invalid SystemSignal reference
/DaVinci/PKG_ClusterCanA/PKG_PDU/Frame1/Signal1_S, the according
ISignalToIPDUMapping object could not be identified.
```

The given example shows an error message which is displayed if the ECU-C file contains a ComSignal which is not known in DaVinci DEV, i.e. the Com configuration was updated by the Com configurator based on a certain communication database but DaVinci DEV does not have the same information available.

Another reason for this error message is the fact that DaVinci DEV can only work with ECU-C files based on ECU-Extracts if the ECU-Extract follows a naming rule concerning ISignalToIPDUMappings (see section 4.8).

5.2 Do not edit the same module configuration in different tools at the same time

Since several tools may edit the same module configuration (e.g. OsTasks can be edited in DaVinci DEV and third party BSW configuration tool), it is not wise to use both tools in parallel, especially if some of the configuration settings are automatically derived from another module's configuration. The decision what to do in DaVinci DEV, i.e. choose „import" or „export" during ECU-C-Synchronization, becomes much easier if the module configurations are edited exclusively by one tool at a time. So you might add a new OsTask in DaVinci DEV, synchronize the ECU-C file, change to BSW configuration tool and add another OsTask for a non-Rte purpose, save the ECU-C file, switch back to DaVinci DEV, synchronize the ECU-C file again and add some more OsTasks for additional runnables. You should not add these new OsTasks in both tools in parallel, since either the OsTasks defined in DaVinci DEV or the OsTaks defined in the BSW configuration tool will be lost when running the synchronization.

If you execute external generators and/or configurators from DaVinci DEV's generator or configurator lists you may enable "Options/Settings/ECU generation/Automatic synchronization of ECU Configuration file". In this case the ECU-C file is always synchronized before the executable is run and after it has been closed. Although this ensures that the ECU-Project and the ECU-C file are always in a consistent state, it might be better to not use this option and explicitly synchronize the ECU-C file to avoid unnecessary and probably time consuming synchronizations when you edit configuration parameters which are not relevant for DaVinci DEV and the executed generator/configurator.

As explained in 3.4 Vector tools implement a mechanism to ensure that the ECU configuration is editable in only one tool at the same time. So you don't have the problems mentioned above when just working with the Vector tools.

5.3 Ensure that all tools use the same max. SHORT-NAME length

Although AUTOSAR specifications up to releases 3.0.6 and 3.1.4 specify that a SHORT-NAME must not exceed the length of 32 characters, DaVinci DEV supports an extended length of up to 128 characters. If this setting is enabled (Options/Settings/AUTOSAR XML) it has to be ensured that all other tools support this extension, too. If this is not the case configuration elements won't be found during ECU-C-Synchronization and will be created a second time if the name is longer than 32 characters.

Starting with AUTOSAR releases 3.0.7, 3.1.6 and 3.2, SHORT-NAMEs with up to 128 characters are allowed by the standard.

5.4 ECU-C files have to be valid according to the AUTOSAR schema

DaVinci DEV validates the ECU-C file against the corresponding AUTOSAR schema

5.5 Configuration elements must have unique SHORT-NAMEs

To be able to uniquely identify a configuration element all elements of the same hierarchy level must have unique SHORT-NAMEs. If this is not the case references within the ECU-C file may point to several different elements which makes the configuration inconsistent. Since this constraint is not checked by the XML schema, an additional check is implemented in DaVinci DEV and an error message is printed in the „Action Log“ tab if such an error occurred.

6 Contact

Visit our website for more information on

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

www.vector-informatik.com