Third Party Modules

Version 1.1.0

2018-08-02

Application Note AN-ISC-8-1153

Author Sven Hesselmann, Klaus Emmert

Restrictions Customer Confidential – Vector decides

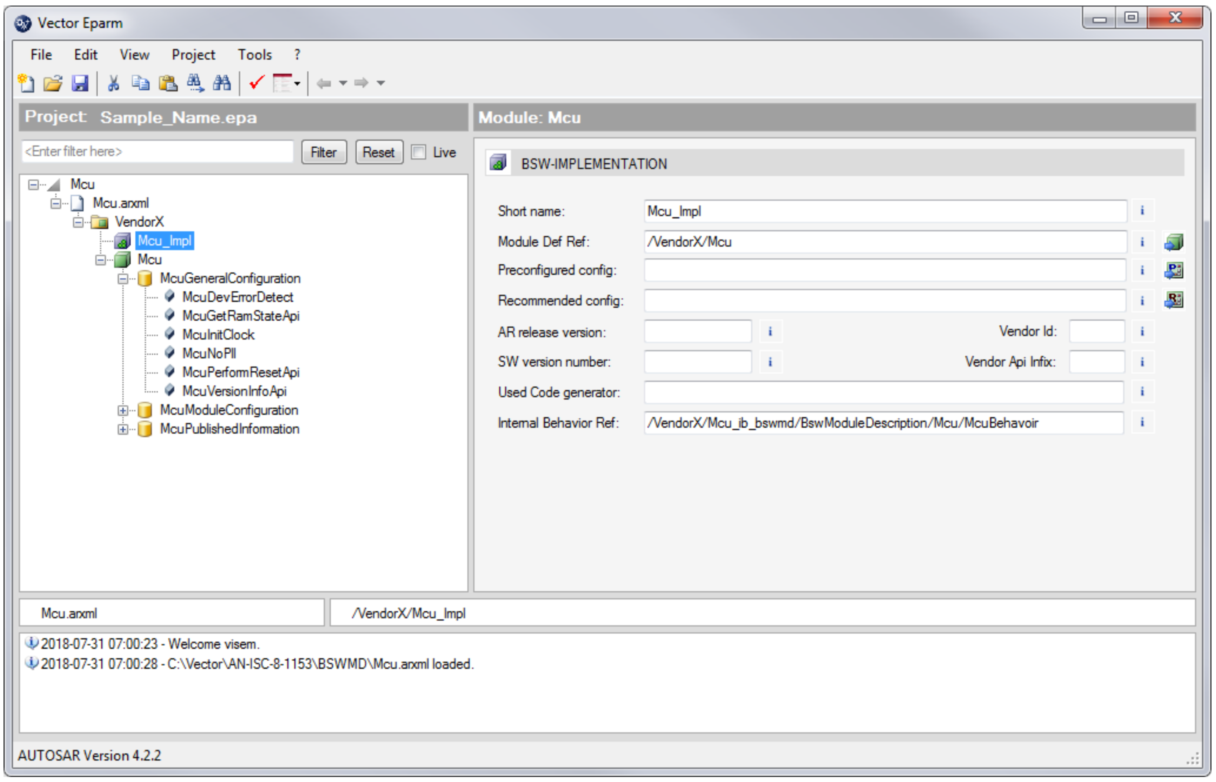
Abstract Introduction how to integrate 3rd partly modules into the MICROSAR4 stack.

Table of Contents

1 Overview

This application note describes the integration of third party modules (i.e. MCAL) into DaVinci Configurator Pro (Version 5 or higher) and the MICROSAR stack for AUTOSAR Release 4.x.

This application note uses the MCU module as an example.



Project: Sample_Name.epa

<Enter filter here>

```
Mcu
  Mcu.arxml
    VendorX
      Mcu_Impl
      Mcu
        McuGeneralConfiguration
          McuDevErrorDetect
          McuGetRamStateApi
          McuInitClock
          McuNoPll
          McuPerformResetApi
          McuVersionInfoApi
        McuModuleConfiguration
        McuPublishedInformation
```

Module: Mcu

BSW-IMPLEMENTATION

Short name:	Mcu_Impl
Module Def Ref:	/VendorX/Mcu
Preconfigured config:	
Recommended config:	
AR release version:	
Vendor Id:	
SW version number:	
Vendor Api Infix:	
Used Code generator:	
Internal Behavior Ref:	/VendorX/Mcu_ib_bswmd/BswModuleDescription/Mcu/McuBehavoir

Mcu.arxml   /VendorX/Mcu_Impl

```
2018-07-31 07:00:23 - Welcome visem.
2018-07-31 07:00:28 - C:\Vector\AN-ISC-8-1153\BSWMD\Mcu.arxml loaded.
```

AUTOSAR Version 4.2.2

Figure 1-1 - Overview of BSWMD file of MCU

2 Integration in DaVinci Configurator Pro

For the integration of a module into a MICROSAR stack, different things have to be done.

If the module fulfills one of the following list points, check this chapter for the description.

The module:

> has parts, generated based on the configuration (i.e. ECUC file)
> requires the SCHM API for Exclusive Area handling
> has cyclic MainFunction calls
> needs access to communication PDUs

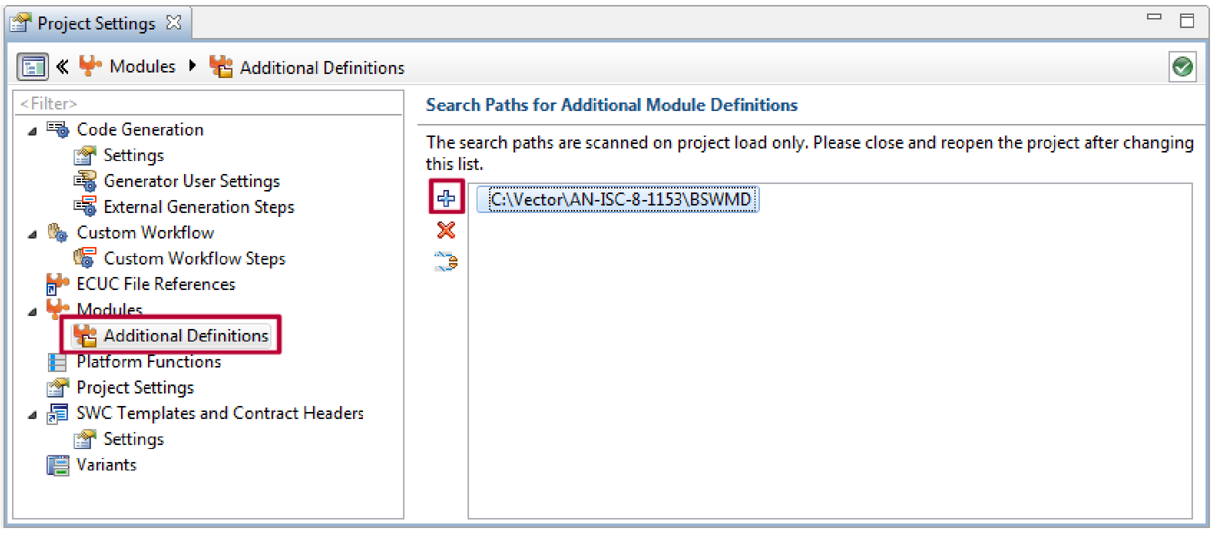
2.1 Configuration With DaVinci Configurator Pro

If the module shall be configured within the DaVinci Configurator Pro, the tool requires its module descriptions in a BSWMD file (basis software module description). Also, the module has to be added to the configuration.

2.1.1 Adding of a BSWMD File

Provide an additional search path for BSWMD files within your configuration. The BSWMD file must end with **.arxml** to be noticed by DaVinci Configurator Pro.

Open **Project|Project Setting|Modules|Additional Definitions** and **Add** the path to the module's BSWMD file as shown in the following screenshots.



The search paths are scanned on project load only. Please close and reopen the project after changing this list.

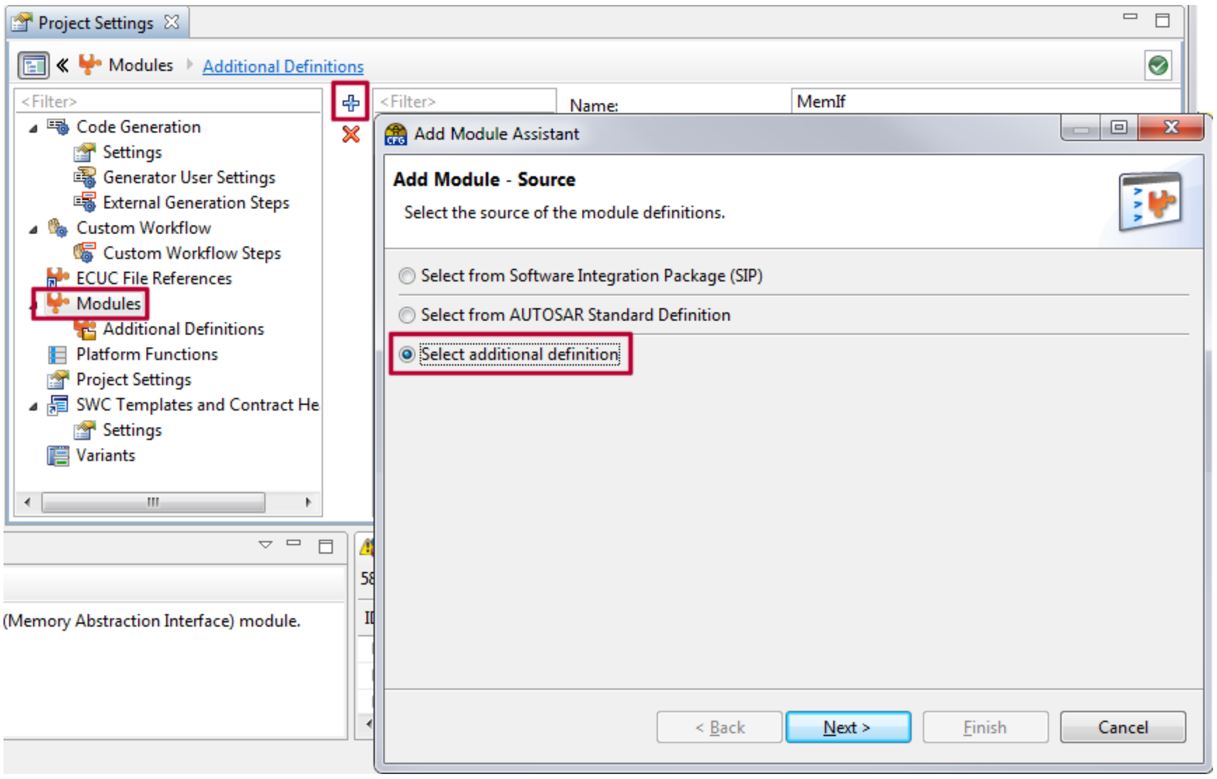
C:\Vector\AN-ISC-8-1153\BSWMD

Figure 2-1 - Modules|Additional Definitions

Now DaVinci Configurator Pro knows the module and it can be added to the configuration. But before, the configuration has to be closed and opened again.

2.1.2 Adding a Module to the Current Project

For adding the module to the current configuration, open **Project|Project Settings|Modules** and **Add** it with the blue plus. If the path to the module is within the delivered SIP, you will find it in **Select from SIP** otherwise in **Select additional definition** (see screenshots below).



Name: MemIf

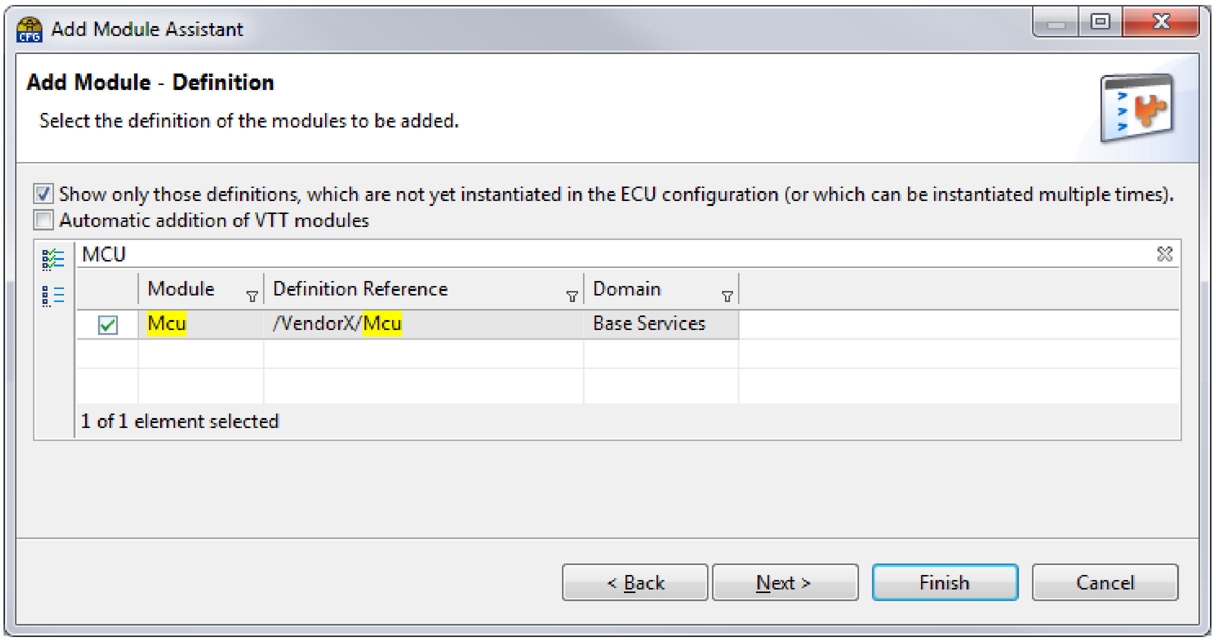
(Memory Abstraction Interface) module.

Add Module - Source

Select the source of the module definitions.

( ) Select from Software Integration Package (SIP)
( ) Select from AUTOSAR Standard Definition
[✓ Select additional definition]

Figure 2-2 - Module Assistant


Add Module Assistant

Add Module - Definition

Select the definition of the modules to be added.

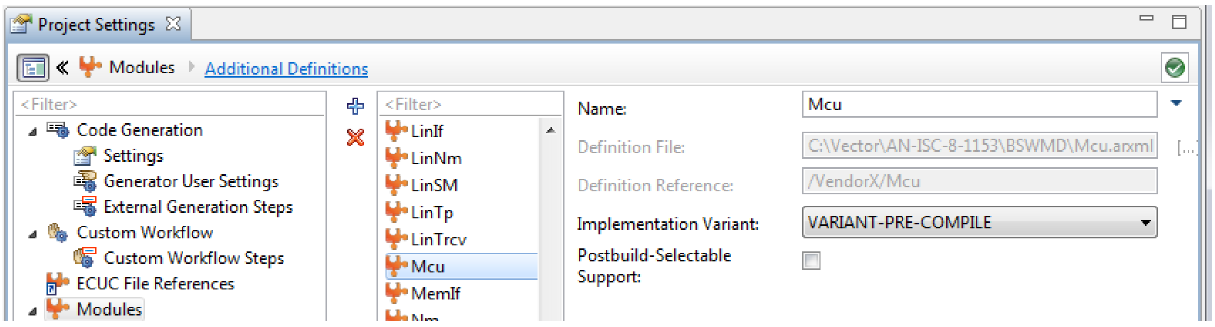
☑ Show only those definitions, which are not yet instantiated in the ECU configuration (or which can be instantiated multiple times).
☐ Automatic addition of VTT modules

MCU

	Module	Definition Reference	Domain
☑	Mcu	/VendorX/Mcu	Base Services

1 of 1 element selected

Figure 2-3 - Module Definitions


Project Settings

Modules › Additional Definitions

Name: Mcu
Definition File: C:\Vector\AN-ISC-8-1153\BSWMD\Mcu.arxml
Definition Reference: /VendorX/Mcu
Implementation Variant: VARIANT-PRE-COMPILE
Postbuild-Selectable Support: ☐

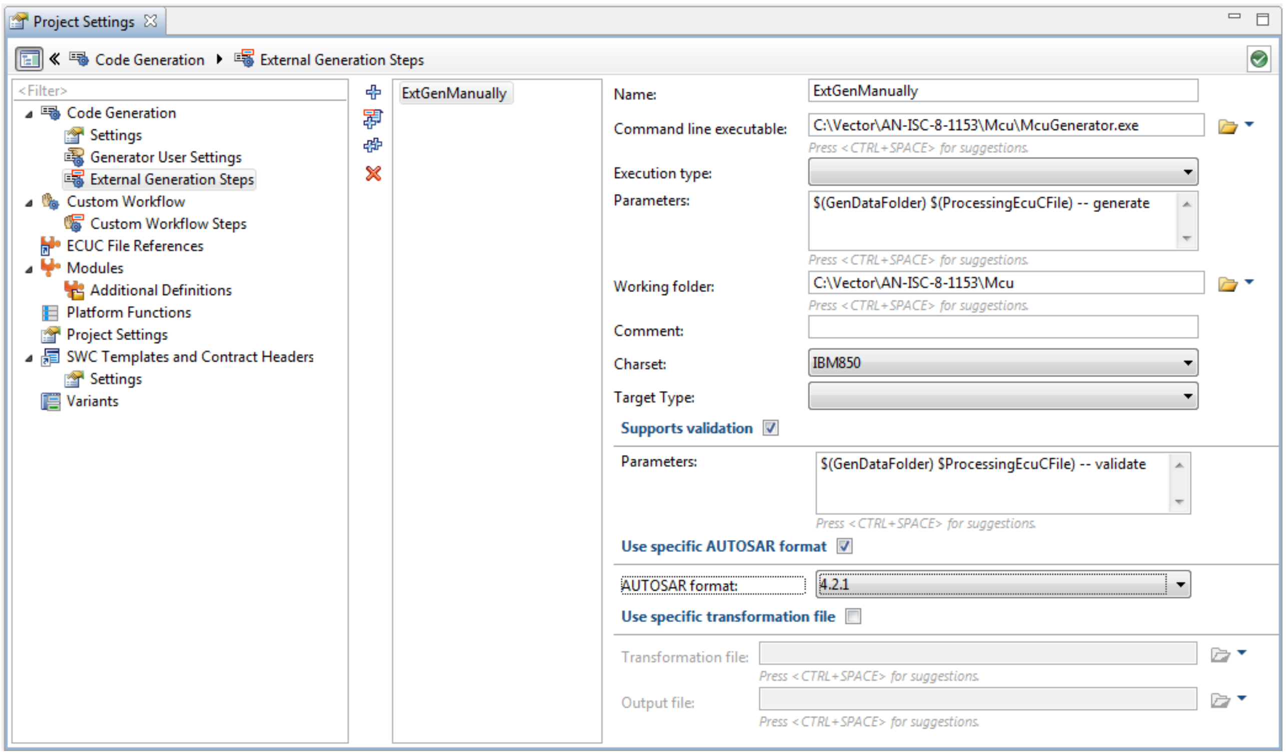
Figure 2-4 - Now the module is within your project, configure it using the **Basic Editor**

2.2 External Generation Step

If the module has generated parts based on the configuration of the ECUC file and the generation shall be started from DaVinci Configurator Pro, the generation list has to be extended. The configuration of the generation steps for third party modules can either be done manually or by a configuration file, making it easier to reuse your module for further projects.

2.2.1 Manual Setup

Open **Project|Project Settings|Code Generation|External Generation Steps** and **Add** the generation settings using the blue plus.


Project Settings

Code Generation › External Generation Steps

<Filter>
- Code Generation
  - Settings
  - Generator User Settings
  - External Generation Steps
- Custom Workflow
  - Custom Workflow Steps
- ECUC File References
- Modules
  - Additional Definitions
- Platform Functions
- Project Settings
- SWC Templates and Contract Headers
  - Settings
- Variants

ExtGenManually

Name:	ExtGenManually
Command line executable:	C:\Vector\AN-ISC-8-1153\Mcu\McuGenerator.exe
Execution type:	
Parameters:	$(GenDataFolder) $(ProcessingEcuCFile) -- generate
Working folder:	C:\Vector\AN-ISC-8-1153\Mcu
Comment:	
Charset:	IBM850
Target Type:	

Press <CTRL+SPACE> for suggestions.

Supports validation ✓

Parameters:	$(GenDataFolder) $ProcessingEcuCFile) -- validate

Use specific AUTOSAR format ✓

AUTOSAR format:	4.2.1

Use specific transformation file ☐

Transformation file:	
Output file:	

Figure 2-5 - External Generation Steps

Specify the module generator settings (i.e. parameters to be handed over). If the module generator also supports validation or requires a transformation of the input file, this can also be configured. For further information also see the Help Content of DaVinci Configurator Pro.

2.2.2 Automatic Setup

The settings described in 2.2.1 can also be done automatically by a so-called **Settings.xml**. For configuration options of the Settings.xml please refer to 3.

2.3 Internal Behavior Description

Most AUTOSAR modules require Exclusive Area and / or MainFunction handling by the RTE. The MICROSAR RTE reads this information from the so-called Internal Behavior description, which is a part of a BSWMD file. This file has to be provided to DaVinci Configurator Pro by placing it into the folder for InternalBehavior files (default is **./Config/InternalBehavior**).

The `<BSW-IMPLEMENTATION>` container within the BSWMD file must have a reference to the Internal Behavior (i.e. `<BEHAVIOR-REF DEST="BSW-INTERNAL-BEHAVIOR">/VendorX/Mcu_ib_bswmd/BswModuleDescriptions/Mcu/McuBehavior</BEHAVIOR-REF>`, see also Figure 1-1). The RTE reads the Internal Behavior of the module from this file and provides a solving action to create an **RteBswModuleInstance** with this information.

2.3.1 Example for Internal Behavior Description

The following example for an Internal Behavior description defines an exclusive area called `MCU_EXCLUSIVE_AREA_0` and a MainFunction called `Mcu_MainFunction`, which has to be called with a cycle time of `0.01` seconds. The file should be placed into the folder for InternalBehavior files (default is ./Config/InternalBehavior) or the content can even be in the BSWMD file itself.

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<AUTOSAR xmlns="http://autosar.org/schema/r4.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://autosar.org/schema/r4.0 autosar_4-0-3.xsd">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>VendorX</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>Mcu_ib_bswmd</SHORT-NAME>
          <AR-PACKAGES>
            <AR-PACKAGE>
              <SHORT-NAME>BswModuleDescriptions</SHORT-NAME>
              <ELEMENTS>
                <BSW-MODULE-DESCRIPTION>
                  <SHORT-NAME>Mcu</SHORT-NAME>
                  <PROVIDED-ENTRYS>
                    <BSW-MODULE-ENTRY-REF-CONDITIONAL>
                      <BSW-MODULE-ENTRY-REF DEST="BSW-MODULE-
ENTRY">/VendorX/Mcu_ib_bswmd/BswModuleDescriptions/Mcu_MainFunction</BSW-MODULE-
ENTRY-REF>
                    </BSW-MODULE-ENTRY-REF-CONDITIONAL>
                  </PROVIDED-ENTRYS>
                  <INTERNAL-BEHAVIORS>
                    <BSW-INTERNAL-BEHAVIOR>
                      <SHORT-NAME>McuBehavior</SHORT-NAME>
                      <EXCLUSIVE-AREAS>
                        <EXCLUSIVE-AREA>
                          <SHORT-NAME>MCU_EXCLUSIVE_AREA_0</SHORT-NAME>
                        </EXCLUSIVE-AREA>
                      </EXCLUSIVE-AREAS>
                      <ENTITYS>
                        <BSW-SCHEDULABLE-ENTITY>
                          <SHORT-NAME>Mcu_MainFunction</SHORT-NAME>
                          <IMPLEMENTED-ENTRY-REF DEST="BSW-MODULE-
ENTRY">/VendorX/Mcu_ib_bswmd/BswModuleDescriptions/Mcu_MainFunction</IMPLEMENTED-
ENTRY-REF>
                        </BSW-SCHEDULABLE-ENTITY>
                      </ENTITYS>
                      <EVENTS>
                        <BSW-TIMING-EVENT>
                          <SHORT-NAME>Mcu_MainFunctionTimingEvent0</SHORT-NAME>
                          <STARTS-ON-EVENT-REF DEST="BSW-SCHEDULABLE-
ENTITY">/VendorX/Mcu_ib_bswmd/BswModuleDescriptions/Mcu/McuBehavior/Mcu_MainFunctio
n</STARTS-ON-EVENT-REF>
                          <PERIOD>0.01</PERIOD>
                        </BSW-TIMING-EVENT>
                      </EVENTS>
                    </BSW-INTERNAL-BEHAVIOR>
                  </INTERNAL-BEHAVIORS>
                </BSW-MODULE-DESCRIPTION>
                <BSW-MODULE-ENTRY>
                  <SHORT-NAME>Mcu_MainFunction</SHORT-NAME>
                  <CALL-TYPE>SCHEDULED</CALL-TYPE>
                  <EXECUTION-CONTEXT>TASK</EXECUTION-CONTEXT>
                </BSW-MODULE-ENTRY>
              </ELEMENTS>
            </AR-PACKAGE>
          </AR-PACKAGES>
        </AR-PACKAGE>
      </AR-PACKAGES>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
```

2.3.2 Templates for Internal Behavior Description

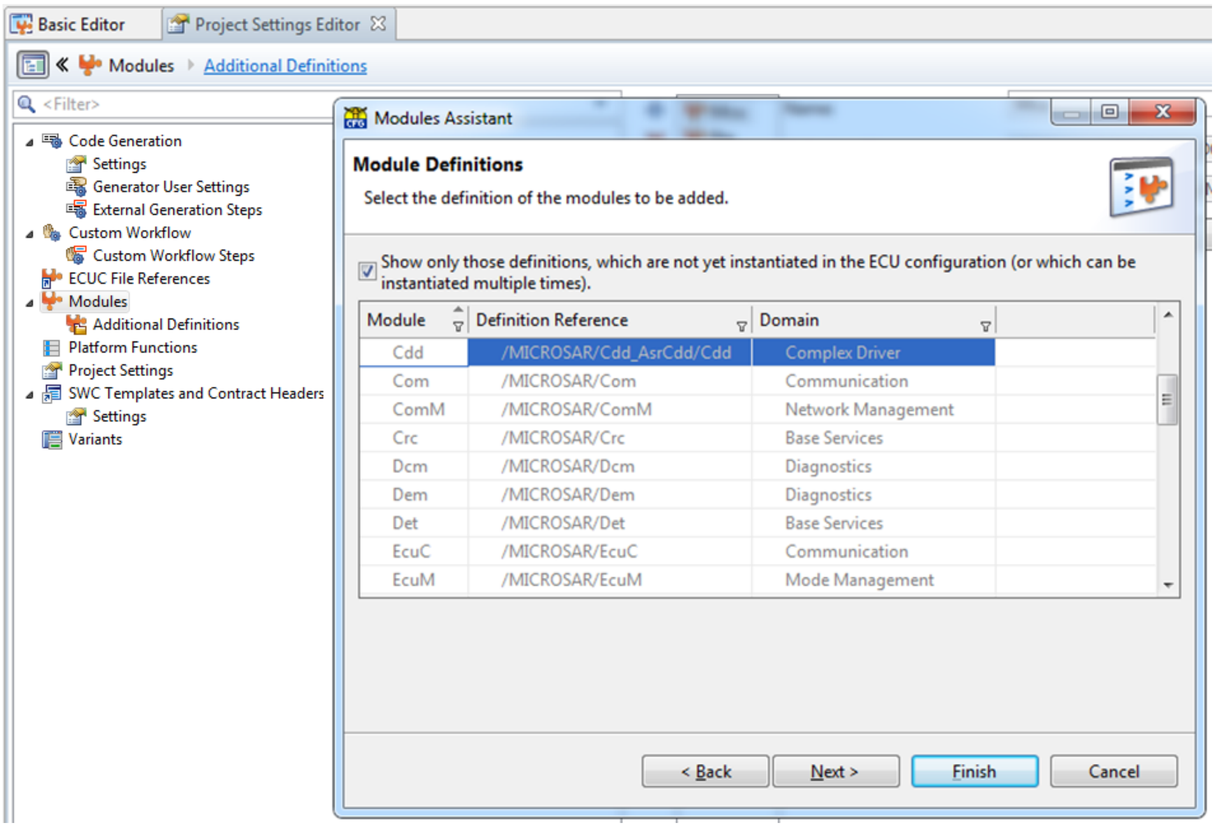
Some modules provided by a MICROSAR delivery are not developed by Vector and do not have an automatic Internal Behavior creation (i.e. MCALs). For easier integration some of them provide templates for the Internal Behavior description. If these files are available within the current delivery they are positioned at .\Misc\InternalBehaviorTemplates. Please remove the starting underscore (_) marking the files as templates, copy them to the InternalBehavior folder of your project and adapt them according to your configuration (i.e. changing the cycle time of the MainFunction).

2.4 RTE Configuration

If the Internal Behavior is configured as described in 2.3, the RTE will provide a solving action to automatically create the required RTE configuration. If a MainFunction has to be called by the RTE, perform the according **Task Mapping** afterwards. The RTE will then generate the Exclusive Area and MainFunction calls as described in the Internal Behavior.

2.5 CDD Configuration

For your own modules that need access to any PDU within the communication stack, use the so-called CDD module (Complex Driver), which is part of the SIP. For adding the CDD to the current configuration, open **Project|Project Settings|Modules** and **Add** it via the blue plus.



Module Definitions

Select the definition of the modules to be added.

☑ Show only those definitions, which are not yet instantiated in the ECU configuration (or which can be instantiated multiple times).

Module	Definition Reference	Domain
Cdd	/MICROSAR/Cdd_AsrCdd/Cdd	Complex Driver
Com	/MICROSAR/Com	Communication
ComM	/MICROSAR/ComM	Network Management
Crc	/MICROSAR/Crc	Base Services
Dcm	/MICROSAR/Dcm	Diagnostics
Dem	/MICROSAR/Dem	Diagnostics
Det	/MICROSAR/Det	Base Services
EcuC	/MICROSAR/EcuC	Communication
EcuM	/MICROSAR/EcuM	Mode Management

Figure 2-6 - Module Definitions CDD

In the Basic Editor you can define, where the CDD shall be placed within the communication stack.

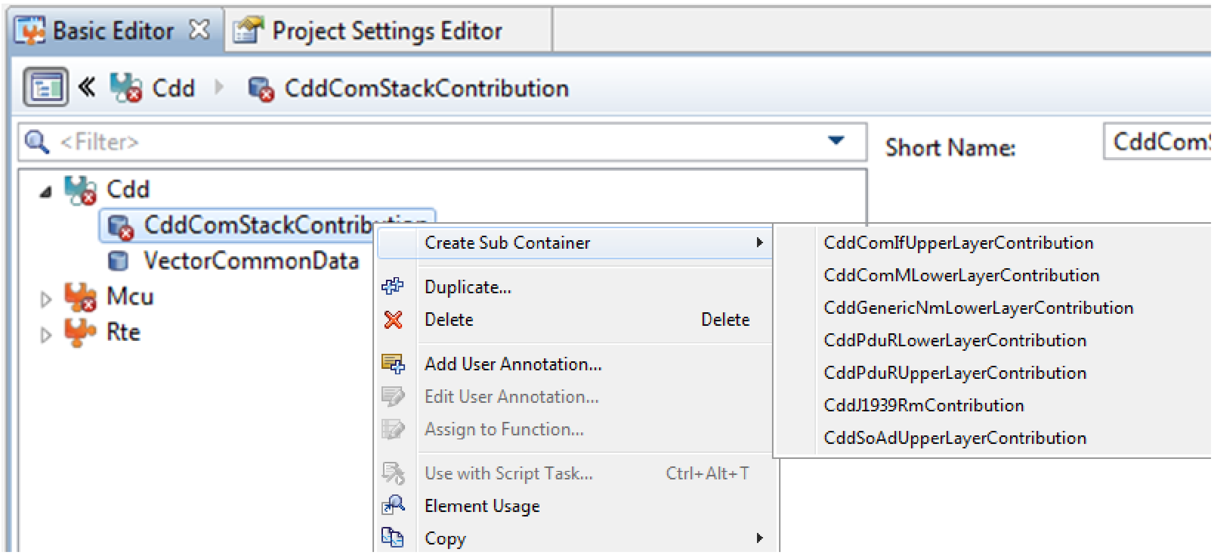


Figure 2-7 - Configuring CDD

For further information how to configure the CDD, please refer to its Technical Reference (TechnicalReference_Asr_CddCfg5.pdf).

3 Settings.xml

The **Settings.xml** is an open interface to the DaVinci Configurator Pro. With this file the following configuration can be done:

> Settings for external validation and generation steps

The file shall be placed to .\DaVinciConfigurator\Generators\<Msn> (i.e. C:\Vector\<yourPath>\DaVinciConfigurator\Generators\Mcu\Settings_ExtGen_Mcu.xml) and is automatically known at start of DaVinci Configurator Pro.

Example

The following example creates the same generator settings as the example in 2.2.1.

```
<Settings>
  <!-- external generator -->
  <Settings Name="com.vector.cfg.gui.core.generators.ExtGenSteps">
    <Settings Name="ExtGenSettings_DrvMcu">
      <Setting Name="Active" Value="true"/>
      <Setting Name="CommandLine"
Value="C:\Vector\CBD1200333_D01_V85x_AddOn\Mcu\Mcu.exe"/>
      <Setting Name="GenerationParameters" Value="$(GenDataFolder)
$(ProcessingEcuCFile) --generate"/>
      <Setting Name="SupportsStandAloneValidation" Value="true"/>
      <Setting Name="ValidationParameters" Value="$(GenDataFolder)
$(ProcessingEcuCFile) --validate"/>
      <Setting Name="TransformationRequired" Value="false"/>
      <Setting Name="TransformationXsltFile" Value=""/>
      <Setting Name="TransformationOutput" Value=""/>
      <Setting Name="WorkingDir" Value="C:\Vector\CBD1200333_D01_V85x_AddOn\Mcu"/>
      <Setting Name="SpecificAsVersionRequired" Value="true"/>
      <Setting Name="RequiredAsVersion" Value="4.0.3"/>
    </Settings>
  </Settings>
</Settings>
```

4 Integration Into the Build Project

AUTOSAR has introduced a mechanism to integrate standardized code into different µC and Compilers. To adapt the modules into a project with specific compiler and linker settings the files `MemMap.h` and `Compiler_Cfg.h` have been introduced. If the module that shall be integrated into a build project that makes use of those mechanisms, these files have to be adapted.

For further information on this topic please also refer to TechnicalReference_Asr_MemoryMapping.pdf within the SIP.

Example code for the following chapters:

```
#define MCU_START_SEC_VAR_INIT_8BIT
#include "MemMap.h"

VAR (uint8, MCU_INIT_DATA) Mcu_InitStatus = 0;

#define MCU_STOP_SEC_VAR_INIT_8BIT
#include "MemMap.h"


#define MCU_START_SEC_PUBLIC_CODE
#include "MemMap.h"

FUNC(void, MCU_PUBLIC_CODE) Mcu_Init (P2CONST(Mcu_ConfigType, AUTOMATIC,
MCU_APPL_CONST) ConfigPtr)
{
…
}

#define MCU_STOP_SEC_PUBLIC_CODE
#include "MemMap.h"
```

4.1 Compiler_Cfg.h

Add all used compiler abstraction defines from your module to this file, even if they are defined to nothing.

Example

Required Compiler_Cfg.h content for above given example:

```
#define MCU_PUBLIC_CODE
#define MCU_APPL_CONST
#define MCU_INIT_DATA
```

4.2 MemMap.h

Add all used memory abstraction defines from your module to this file.

Example

Required MemMap.h content for above given example:

```
#ifdef MCU_START_SEC_VAR_INIT_8BIT
  #undef MCU_START_SEC_VAR_INIT_8BIT
  #define START_SEC_VAR_INIT_8BIT
#endif
#ifdef MCU_STOP_SEC_VAR_INIT_8BIT
  #undef MCU_STOP_SEC_VAR_INIT_8BIT
  #define STOP_SEC_VAR
#endif

#ifdef MCU_START_SEC_PUBLIC_CODE
  #undef MCU_START_SEC_PUBLIC_CODE
  #define START_SEC_CODE
#endif
#ifdef MCU_STOP_SEC_PUBLIC_CODE
  #undef MCU_STOP_SEC_PUBLIC_CODE
  #define STOP_SEC_CODE
#endif
```

5 Additional Resources

TechnicalReference_Asr_MemoryMapping.pdf

TechnicalReference_Asr_CddCfg5.pdf

6 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.