MICROSAR VttCntrl

Technical Reference

vVIRTUALtarget

Version 0.1.1

Authors	Damian Philipp, Max-Ferdinand Suffel
Status	Released

Document Information

History

Author	Date	Version	Remarks
Philipp, Damian	2015-11-30	0.1.0	Initial version
Suffel, Max-Ferdinand	2016-01-11	0.1.1	Updated list of static/generated files.

Reference Documents

No.	Source	Title	Version
[1]	Vector	TechnicalReference_vVIRTUALtarget.pdf	V1.0.2

Scope of the Document

This technical reference describes the end-user API of the VttCntrl basis software module. More general information about Vector vVIRTUALtarget is contained in a separate document [1], which is also part of the delivery.

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 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
[14,0]	Release for Vector MICROSAR4 R14.

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the MICROSAR module VttCntrl.

Supported AUTOSAR Release:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	VttCntrl_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)

VttCntrl implements the interface between the MICROSAR stack and the microcontroller emulation in Vector CANoe.

2.1 Architecture Overview

The following figure shows where VttCntrl is located in the MICROSAR stack.

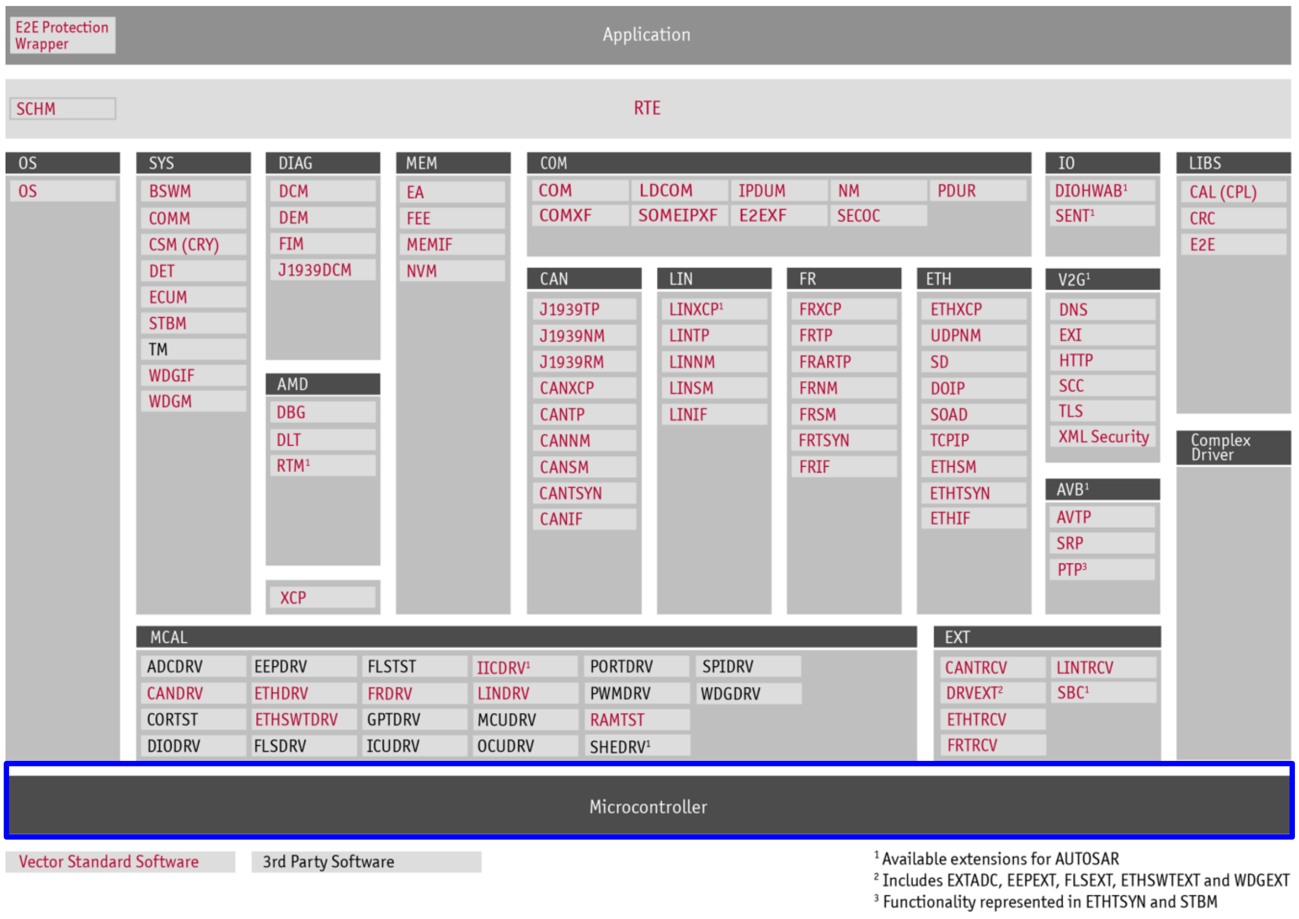


Figure 2-1 AUTOSAR 4.2 Architecture Overview

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for VttCntrl. As VttCntrl is not specified by AUTOSAR, information regarding the implementation of the AUTOSAR standard is omitted. Features provided by VttCntrl are listed in the table

> Table 3-1 Features provided beyond the AUTOSAR standard

Features Provided Beyond The AUTOSAR Standard
Access to custom Vector CANoe System Variables
Access to notification function for simulated ECU state changes

Table 3-1 Features provided beyond the AUTOSAR standard

These features are described in more detail in the following sections.

3.1.1 Access to custom Vector CANoe System Variables

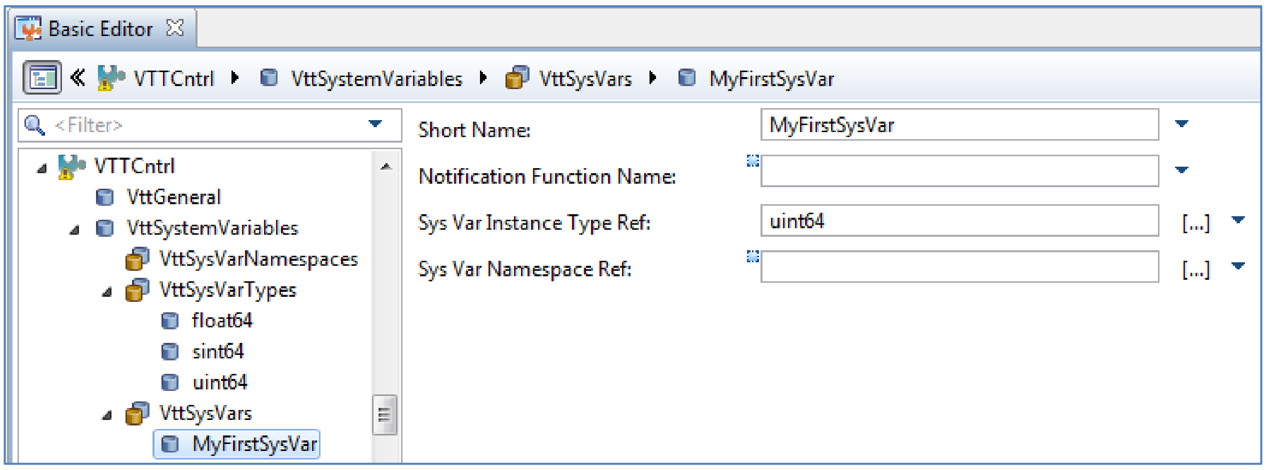
VttCntrl offers configuration options to create additional user-defined CANoe system variables for the simulated ECU. These system variables can be used to create vVIRTUALtarget implementations of custom drivers. For example, using user-defined system variables allows exporting the IO channels of IO expansion modules connected via SPI, instead of simulating the raw SPI communication only. Furthermore, user-defined system variables may be used to communicate between CAPL-implementations of ASIC modules present in the physical ECU and corresponding stub modules implemented as CDDs or SWCs in the ECU software or between tests implemented in Vector CANoe and corresponding test application SWCs implemented in the ECU software.

User-defined system variables can be configured as follows.

3.1.1.1 Definition of user-defined system variables

To let VttCntrl register a new user-defined system variable in CANoe, proceed as follows:

1. Configure a system variable “MyFirstSysVar” in DaVinci Configurator 5. Be sure to specify a “Sys Var Instance Type Ref”.



Short Name: MyFirstSysVar
Notification Function Name:
Sys Var Instance Type Ref: uint64
Sys Var Namespace Ref:

Figure 3-1 Configuration of a new user-defined system variable in VttCntrl

2. Generate the BSW configuration for VttCntrl using DaVinci Configurator 5.
3. Add the respective implementation files and configuration files to your Microsoft Visual Studio solution.

   If you are using the Vector vVIRTUALtarget tool to create your Microsoft Visual Studio solution, simply recreate the solution. The Vector vVIRTUALtarget tool will automatically detect the configuration of user-defined system variables and add the appropriate source files to your solution.

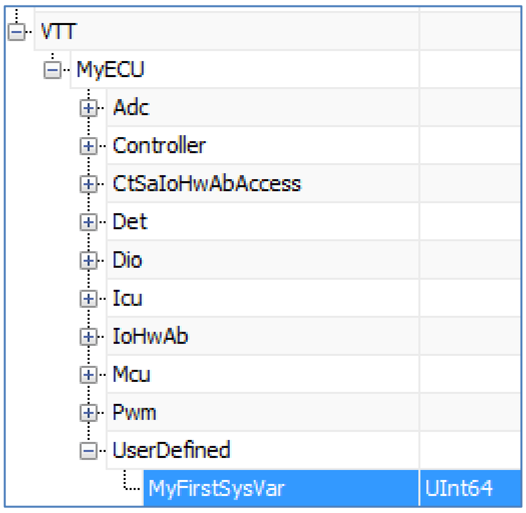
   If the Microsoft Visual Studio solution is maintained manually, add the following files to your solution:

- <SIP Directory>/BSW/VttCntrl/VttCntrl_SysVar.h
- <SIP Directory>/BSW/VttCntrl/VttCntrl_SysVar.c
- <SIP Directory>/BSW/VttCntrl/VttCntrl_SysVar_Cfg.h
- <SIP Directory>/BSW/VttCntrl/VttCntrl_SysVar_Cfg.c
- <SIP Directory>/BSW/VttCntrl/VttCntrl_SysVar_Cbk.h

4. In your CDD driver or SWC, access the value of the system variable as follows:

```
#include "VttCntrl_SysVar.h"
void demo(void) {
   uint64 incomingValue;
   uint64 outgoingValue = 42;
   VttSysVar_Write_uint64(VttCntrlConf_VttCntrl_SysVar_\
     MyFirstSysVar, outgoingValue);
   VttSysVar_Read_uint64(VttCntrlConf_VttCntrl_SysVar_\
     MyFirstSysVar, &incomingValue);
}
```

5. Recompile the ECU project.
6. Access the system variable in Vector CANoe. The system variable is placed in the namespace VTT::MyECU::UserDefined.



```
VTT
  MyECU
    Adc
    Controller
    CtSaIoHwAbAccess
    Det
    Dio
    Icu
    IoHwAb
    Mcu
    Pwm
    UserDefined
      MyFirstSysVar        UInt64
```

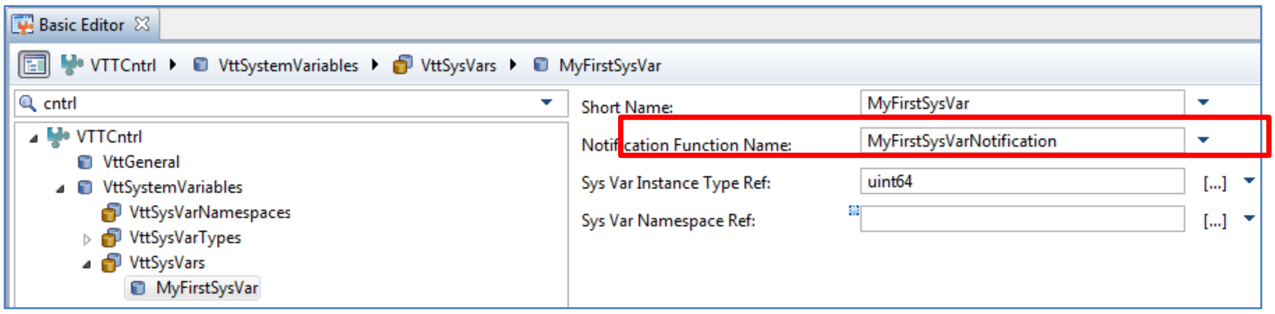
Figure 3-2 User-defined system variable as seen in CANoe

3.1.1.2 Enabling a notification function for a user-defined system variable

To be notified via a notification function whenever the value of a system variable changes, proceed as follows.

Caution
All notification functions will be called in interrupt context from the same interrupt handler. However, you must implement a separate notification function for each system variable that should support notifications.

1. Set the parameter “VttSysVarNotificationFunctionName“ to the name of a notification function for the system variable.



Short Name: MyFirstSysVar
Notification Function Name: MyFirstSysVarNotification
Sys Var Instance Type Ref: uint64
Sys Var Namespace Ref:

Figure 3-3 Configuration of a notification function for a user-defined system variable

2. Generate the BSW configuration for VttCntrl in DaVinci Configurator 5.
3. Implement your notification function as detailed in Section 5.6.1.1. A function prototype for the notification function is generated in VttCntrl_SysVar_Cbk.h. See the following code for an example notification function:

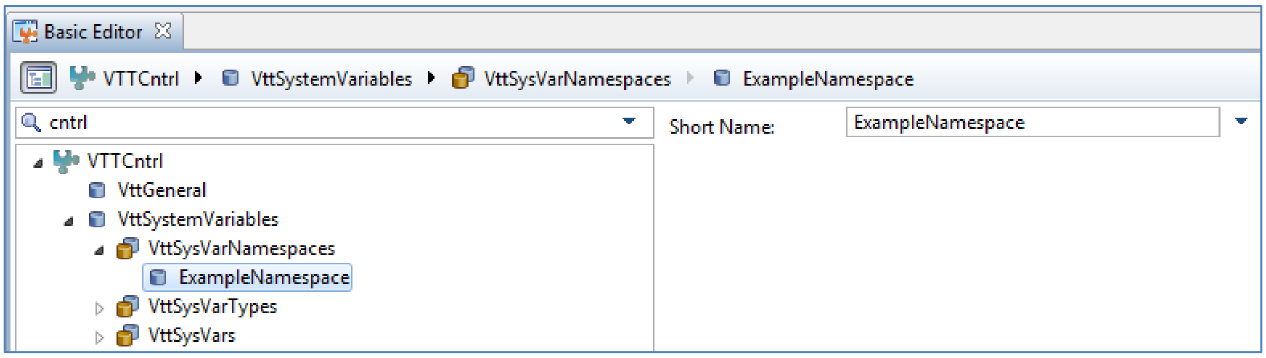
```
#include "VttCntrl_SysVar_Cbk.h"
static uint64 incomingValue;

void MyFirstSysVarNotification(uint64 value) {
  incomingValue = value;
}
```

4. Recompile the ECU project. The notification function is now enabled.

3.1.1.3 Placing user-defined system variables in custom namespaces

For convenience, user-defined system variables may be grouped into sub-namespaces of VTT::<EcuName>::UserDefined in Vector CANoe. To place a system variable in a namespace, proceed as follows.

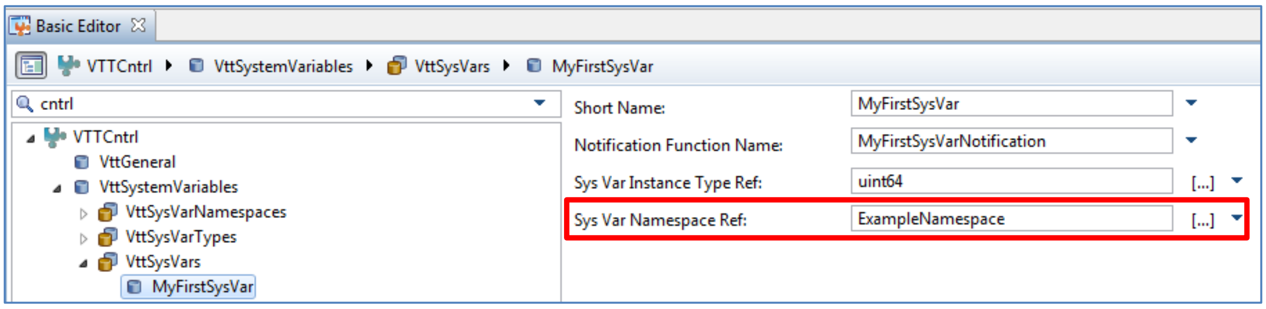
1. Add a namespace in DaVinci Configurator 5.



Short Name: ExampleNamespace

Figure 3-4 System variable namespace in DaVinci Configurator 5

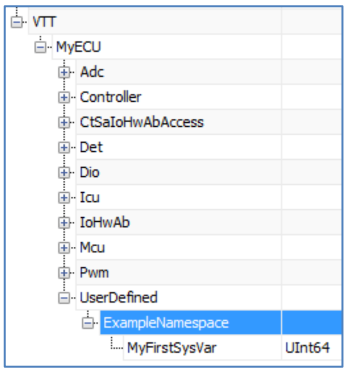
2. Add a namespace reference to the respective system variable



Short Name: MyFirstSysVar
Notification Function Name: MyFirstSysVarNotification
Sys Var Instance Type Ref: uint64
Sys Var Namespace Ref: ExampleNamespace

Figure 3-5 System variable with namespace reference in DaVinci Configurator 5

3. Regenerate the BSW configuration of VttCntrl in DaVinci Configurator 5.
4. Recompile the ECU project. The system variable is now available in the defined namespace.



```
VTT
  MyECU
    Adc
    Controller
    CtSaIoHwAbAccess
    Det
    Dio
    Icu
    IoHwAb
    Mcu
    Pwm
    UserDefined
      ExampleNamespace
        MyFirstSysVar        UInt64
```

Table 3-2 User-defined system variable placed in user-defined namespace as seen in CANoe.

3.1.2 Access to notification function for simulated ECU state changes

vVIRTUALtarget performs an internal handling of the state of the simulated ECU. To support the development of custom vVIRTUALtarget MCAL modules, VttCntrl can call a handler function to notify user code when a state change occurs.

Note
It is not necessary to implement a custom ECU state change handler to make use of user-defined system variables.

To use this functionality, set the parameter `/MICROSAR/VTT/VTTCntrl/VttGeneral/VttUserCalloutOnStateChange` to `TRUE`. You must then provide an implementation of the handler function as described in Section 5.6.1.2.

3.2 Initialization

Initialization of VttCntrl is handled internally.

3.3 Main Functions

VttCntrl does not offer any Main Function. The implementation of VttCntrl is entirely event-driven. Actions are performed either due to calls from a vVIRTUALtarget MCAL module or due to simulation events triggered by Vector CANoe.

3.4 Error Handling

3.4.1 Development Error Reporting

No DET errors are defined for VttCntrl.

3.4.2 Production Code Error Reporting

No DEM errors are defined for VttCntrl.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR VttCntrl into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the VttCntrl contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
VttCntrl_Adc.c	This is the source file for the Adc implementation of the simulated microcontroller.
VttCntrl_Adc.h	This is the header file for the Adc implementation of the simulated microcontroller.
VttCntrl_Base.c	This is the source file for the base VttCntrl implementation.
VttCntrl_Base.h	This is the header file for the base VttCntrl implementation.
VttCntrl_Can.c	This is the source file for the Can implementation of the simulated microcontroller.
VttCntrl_Can.h	This is the header file for the Can implementation of the simulated microcontroller.
VttCntrl_Det.c	This is the source file for the CANoe Det Error database configuration. It contains the Module ID, Name, Service ID, Service Name, Error ID and Error Message of every BSW module.
VttCntrl_Det.h	This is the header file for the CANoe Det Error database configuration.
VttCntrl_Dio.c	This is the source file for the Dio implementation of the simulated microcontroller.
VttCntrl_Dio.h	This is the header file for the Dio implementation of the simulated microcontroller.
VttCntrl_Eep.c	This is the source file for the Eep implementation of the simulated microcontroller.
VttCntrl_Eep.h	This is the header file for the Eep implementation of the simulated microcontroller.
VttCntrl_Eth.c	This is the source file for the Eth implementation of the simulated microcontroller.
VttCntrl_Eth.h	This is the header file for the Eth implementation of the simulated microcontroller.
VttCntrl_Fls.c	This is the source file for the Fls implementation of the simulated microcontroller.
VttCntrl_Fls.h	This is the header file for the Fls implementation of the simulated microcontroller.
VttCntrl_Fr.c	This is the source file for the Fr implementation of the simulated microcontroller.
VttCntrl_Fr.h	This is the header file for the Fr implementation of the simulated

File Name	Description
	microcontroller.
VttCntrl_Gpt.c	This is the source file for the Gpt implementation of the simulated microcontroller.
VttCntrl_Gpt.h	This is the header file for the Gpt implementation of the simulated microcontroller.
VttCntrl_Icu.c	This is the source file for the Icu implementation of the simulated microcontroller.
VttCntrl_Icu.h	This is the header file for the Icu implementation of the simulated microcontroller.
VttCntrl_Mcu.c	This is the source file for the Mcu implementation of the simulated microcontroller.
VttCntrl_Mcu.h	This is the header file for the Mcu implementation of the simulated microcontroller.
VttCntrl_Pwm.c	This is the source file for the Pwm implementation of the simulated microcontroller.
VttCntrl_Pwm.h	This is the header file for the Pwm implementation of the simulated microcontroller.
VttCntrl_SysVar.c	This is the source file for the user-defined Vector CANoe system variables.
VttCntrl_SysVar.h	This is the header file for the user-defined Vector CANoe system variables. Include this file to access the system variables from your application code.
VttCntrl_Wdg.c	This is the source file for the Wdg implementation of the simulated microcontroller.
VttCntrl_Wdg.h	This is the header file for the Wdg implementation of the simulated microcontroller.
VttCntrl_ModuleIncludes.h	This is the header file that (conditionally) includes all Vector vVIRTUALtarget MCAL header files, if the respective module is part of the ECU configuration.
VTTCntrl_Compiler_Cfg.inc	This is the include file containing the compiler abstraction of the vVIRTUALtarget.

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator 5.

File Name	Description
VttCntrl.h	This is a header file to include all header files relevant for the implementation of VttCntrl.
VttCntrl_Adc_Cfg.c	This is the source file containing the pre-compile configuration of the Adc implementation of the simulated microcontroller.
VttCntrl_Adc_Cfg.h	This is the header file containing the pre-compile configuration of the Adc implementation of the simulated microcontroller.
VttCntrl_Can_Cfg.c	This is the source file containing the pre-compile configuration of the

File Name	Description
	Can implementation of the simulated microcontroller.
VttCntrl_Can_Cfg.h	This is the header file containing the pre-compile configuration of the Can implementation of the simulated microcontroller.
VttCntrl_CanTrcv.c	This is the source file containing the CanTrcv implementations of the simulated ECU.
VttCntrl_CanTrcv.h	This is the header file declaring the CanTrcvs of the simulated ECU.
VttCntrl_Cfg.h	This is the header file that contains #defines to detect whether a given Vector vVIRTUALtarget MCAL module is part of the ECU configuration.
VttCntrl_Defines.h	This is the header file containing the INIT-Pointer declarations for the modules of the vVIRTUALtarget.
VttCntrl_Dio_Cfg.c	This is the source file containing the pre-compile configuration of the Dio implementation of the simulated microcontroller.
VttCntrl_Dio_Cfg.h	This is the header file containing the pre-compile configuration of the Dio implementation of the simulated microcontroller.
VttCntrl_Eep_Cfg.c	This is the source file containing the pre-compile configuration of the Eep implementation of the simulated microcontroller.
VttCntrl_Eep_Cfg.h	This is the header file containing the pre-compile configuration of the Eep implementation of the simulated microcontroller.
VttCntrl_Eth_Cfg.c	This is the source file containing the pre-compile configuration of the Eth implementation of the simulated microcontroller.
VttCntrl_Eth_Cfg.h	This is the header file containing the pre-compile configuration of the Eth implementation of the simulated microcontroller.
VttCntrl_Fls_Cfg.c	This is the source file containing the pre-compile configuration of the Fls implementation of the simulated microcontroller.
VttCntrl_Fls_Cfg.h	This is the header file containing the pre-compile configuration of the Fls implementation of the simulated microcontroller.
VttCntrl_Fr_Cfg.c	This is the source file containing the pre-compile configuration of the Fr implementation of the simulated microcontroller.
VttCntrl_FrTrcv.c	This is the source file containing the FrTrcv implementations of the simulated ECU.
VttCntrl_FrTrcv.h	This is the header file declaring the FrTrcvs of the simulated ECU.
VttCntrl_FrTrcv_Cbk.h	This is the header file declaring all FrTrcvs callback functions, e.g. for error notifications.
VttCntrl_Gpt_Cfg.c	This is the source file containing the pre-compile configuration of the Gpt implementation of the simulated microcontroller.
VttCntrl_Gpt_Cfg.h	This is the header file containing the pre-compile configuration of the Gpt implementation of the simulated microcontroller.
VttCntrl_Icu_Cfg.c	This is the source file containing the pre-compile configuration of the Icu implementation of the simulated microcontroller.
VttCntrl_Icu_Cfg.h	This is the header file containing the pre-compile configuration of the Icu implementation of the simulated microcontroller.
VttCntrl_Lin.c	This is the source file for the Lin implementation of the simulated microcontroller.

File Name	Description
VttCntrl_Lin.h	This is the header file for the Lin implementation of the simulated microcontroller.
VttCntrl_LinTrcv.c	This is the source file containing the LinTrcv implementations of the simulated ECU.
VttCntrl_LinTrcv.h	This is the header file declaring the LinTrcvs of the simulated ECU.
VttCntrl_Pwm_Callout_Stubs.c	This is the source file containing the pre-compile configuration of the configured Pwm callout functions.
VttCntrl_Pwm_Cfg.c	This is the source file containing the pre-compile configuration of the Pwm implementation of the simulated microcontroller.
VttCntrl_Pwm_Cfg.h	This is the header file containing the pre-compile configuration of the Pwm implementation of the simulated microcontroller.
VttCntrl_SysVar_Cbk.h	This is the header file that defines the function prototypes for all configured system variable notification functions.
VttCntrl_SysVar_Cfg.c	This is the source file that contains the pre-compile configuration of the configured system variables.
VttCntrl_SysVar_Cfg.h	This is the header file that contains the pre-compile configuration of the configured system variables.

Table 4-2 Generated files

5 API Description

5.1 Type Definitions

VttCntrl does not define any type.

5.2 Interrupt Service Routines provided by VttCntrl

VttCntrl uses a single interrupt service routine to send notifications for system variable value changes.

5.2.1 VttSysVarIsr_0

Prototype	
void **VttSysVarIsr_0** (void)	
Parameter	
--	--
Return code	
--	--
Functional Description	
This interrupt service routine is used to execute notifications for value changes to system variables. See 5.6.1.1 VttSysVarNotificationFunctionName.	
Particularities and Limitations	
> None.	

Table 5-1 VttSysVarIsr_0

5.3 Services provided by VttCntrl

5.3.1 VttSysVar_Read_<SysVarType>

Prototype	
Std_ReturnType VttSysVar_Read_<SysVarType>(uint32 sysVar, SysVarType *value)	
Parameter	
sysVar	ID of the system variable to be written.
*value	Pointer to a variable of type SysVarType. The Value read from the system variable will be written to the given memory location. The data type is the type of the system variable as configured in DaVinci Configurator 5
Return code	
Std_Return	E_OK if the value was successfully read. E_NOT_OK if the type of the function does not match the type of the system variable, if the system variable does not exist or the system variable could not be read.

Functional Description
This function reads the value from the system variable with the given ID and stores it in the memory location defined by SysVarType, provided that the type of the system variable matches the type of the function. An instance of this function is generated for every type of system variable instantiated in the ECU configuration.
Particularities and Limitations
> This function is synchronous. > This function is reentrant.
Expected Caller Context
> There are no limitations to the caller context.

Table 5-2 VttSysVar_Read_<SysVarType>

5.3.2 VttSysVar_Write_<SysVarType>

Prototype	
`Std_ReturnType VttSysVar_Write_<SysVarType>(uint32 sysVar, SysVarType value)`	
Parameter	
`sysVar`	ID of the system variable to be written.
`value`	The value to be written to the system variable. The data type is the type of the system variable as configured in DaVinci Configurator 5
Return code	
`Std_Return`	E_OK if the value was successfully written. E_NOT_OK if the type of the function does not match the type of the system variable, if the system variable does not exist or the system variable could not be written.
Functional Description	
This function writes the given value to the system variable with the given ID, provided that the type of the system variable matches the type of the function. An instance of this function is generated for every type of system variable instantiated in the ECU configuration.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> There are no limitations to the caller context.	

Table 5-3 VttSysVar_Write_<SysVarType>

5.4 Services used by VttCntrl

VttCntrl does not use any services.

5.5 Callback Functions

VttCntrl does not provide any callback functions.

5.6 Configurable Interfaces

5.6.1 Notifications

At its configurable interfaces VttCntrl defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by VttCntrl but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

5.6.1.1 VttSysVarNotificationFunctionName

Prototype	
`void` **`VttSysVarNotificationFunctionName`** `( SysVarType value )`	
Parameter	
`value`	The new value of the system variable.
Return code	
`retCode`	description
Functional Description	
This notification function is called for a system variable when the value of the system variable changes.	
Particularities and Limitations	
> If the rate of changes to the value of the system variable is greater than the rate at which this notification is sent, notifications will be lost. Only the notification for the latest value will be sent. > If a change to the value of a system variable occurs while the notification interrupt service routine is executed, the notification may be sent a second time with the new value.	
Call context	
> This notification is always called in interrupt context. > This notification is sent after a change in the value of the corresponding system variable.	

Table 5-4 VttSysVarNotificationFunctionName

5.6.1.2 VttCntrl_Base_UserCallout_OnStateChange

Prototype
`void` **`VttCntrl_Base_UserCallout_OnStateChange`** `(uint8 action, uint8 oldState, uint8 newState)`

Parameter	

`action`	The state change action performed. The value may be one of the following:
	`CANOEAPI_ECUACTION_NOACTION`
	`CANOEAPI_ECUACTION_LOAD`
	`CANOEAPI_ECUACTION_UNLOAD`
	`CANOEAPI_ECUACTION_INITMEASUREMENT`
	`CANOEAPI_ECUACTION_STARTMEASUREMENT`
	`CANOEAPI_ECUACTION_STOPMEASUREMENT`
	`CANOEAPI_ECUACTION_SWITCHON`
	`CANOEAPI_ECUACTION_SWITCHOFF`
	`CANOEAPI_ECUACTION_GOTOSLEEP`
	`CANOEAPI_ECUACTION_WAKEUP`
	`CANOEAPI_ECUACTION_RESET`

`oldState`	The last ECU state the simulated ECU was in. See `newState` for possible values.

`newState`	The new ECU state the simulated ECU is in. The value may be one of the following:
	`CANOEAPI_ECUSTATE_INITIAL`
	`CANOEAPI_ECUSTATE_NOMEASUREMENT`
	`CANOEAPI_ECUSTATE_PRESTART`
	`CANOEAPI_ECUSTATE_POWEROFF`
	`CANOEAPI_ECUSTATE_PROCESSING`
	`CANOEAPI_ECUSTATE_SLEEPING`
	`CANOEAPI_ECUSTATE_FINAL`

Return code	
`void`	
Functional Description	
When enabled, this notification function is called for a system variable when the value of the system variable changes.	
Particularities and Limitations	
> The name of the callback function cannot be configured. > This notification is called in CANoe context. It is provided as a convenience function for users that wish to extend the simulation model of Vector vVIRTUALtarget. No ECU application code may be executed in the context of this notification function. > When data items are accessed in the context of this notification, the data items must be protected against concurrent access from Vector CANoe and Vector MICROSAR using CANoeAPI_AtomicBegin() and CANoeAPI_AtomicEnd()	
Call context	
> This notification is called in CANoe context.	

Table 5-5 VttCntrl_Base_UserCallout_OnStateChange

6 Configuration

VttCntrl is configured using DaVinci Configurator 5. The following sections summarize all configuration parameters.

6.1 Configuration Variants

The VttCntrl supports the configuration variants

> `VARIANT-PRE-COMPILE`

The configuration classes of the VttCntrl parameters depend on the supported configuration variants. For their definitions please see the VTT_bswmd.arxml file.

6.2 Configuration of XML Attributes

6.2.1 VttCntrl

Module Name	VttCntrl
Module Description	Configuration of the VttCntrl (Vector vVIRTUALtarget ECU simulation interface layer) module.
Post-Build Variant Support	false

Included Containers		
Container Name	Multiplicity	Sope / Dependency
VttGeneral	1	This container contains the configuration parameters of the VttCntrl module.
VttSystemVariables	0..1	This container contains the configuration parameters and sub containers for access to user-defined system variables.

6.2.2 VttGeneral

Name	VttEntryPointName
Description	Defines the entry point of the node layer DLL.
Multiplicity	1
Type	Function name
Default Value	main

Name	VttHeaderFile
Description	Defines the header file for the entry point of the node layer DLL.
Multiplicity	1
Type	String
Default Value	-

Name	VttPreferredInputAsrVersion
Description	Specifies the AUTOSAR version which should be used for input extracts. This parameter is only relevant for Vector vVIRTUALtarget Pro.
Multiplicity	1
Type	String
Default Value	4.2.1

Name	VttRequiredEcucGeneratorVersion
Description	Specifies the version of vVIRTUALtarget's built-in ECU-C generator which is required for the SIP. This parameter is only relevant for Vector vVIRTUALtarget Pro.
Multiplicity	1
Type	String
Default Value	1.0

Name	VttUserCalloutOnStateChange
Description	Specified whether VTTCntrl should provide the callout VttCntrl_UserCallout_OnStateChange (see 5.6.1.2 VttCntrl_Base_UserCallout_OnStateChange). This callout allows user code to, e.g., perform work when the VT DLL is loaded.
Multiplicity	1
Type	Boolean
Default Value	false

6.2.3 VttSystemVariables

Included Containers		
Container Name	Multiplicity	Sope / Dependency
VttSysVarNamespace	0..n	This container contains the namespaces used to group system variables for access in Vector CANoe.
VttSysVarType	0..n	This container contains the supported data types of system variables.

Included Containers		
Container Name	Multiplicity	Sope / Dependency
VttSysVar	0..n	This container contains the definition of instances of system variables.

6.2.4 VttSysVarNamespace

Name	ShortName
Description	Defines the name of the system variable namespace.
Multiplicity	1
Type	String
Default Value	VttSysVar

6.2.5 VttSysVarType

Note that VttCntrl currently does not support the definition of additional VttSysvarTypes beyond the preconfigured values.

Name	ShortName
Description	Defines the name of the system variable type.
Multiplicity	1
Type	String
Default Value	VttSysVar

6.2.6 VttSysVar

Name	ShortName
Description	Defines the name of the system variable.
Multiplicity	1
Type	String
Default Value	VttSysVar

Name	VttSysVarNotificationFunctionName
Description	Name of the notification callback function. When this value is empty or undefined, notifications are disabled for this system variable.
Multiplicity	0..1
Type	Linker symbol
Default Value	-

Name	VttSysVarInstanceTypeRef
Description	Specifies the data type of this system variable.
Multiplicity	1
Type	Reference to VttSystemVariables/VttSysVarType
Default Value	-

Name	VttSysVarNamespaceRef
Description	Specifies an optional namespace under which this system variable will be displayed in Vector CANoe.
Multiplicity	0..1
Type	Reference to VttSystemVariables/VttSysVarNamespace
Default Value	-

Included Containers		
Container Name	Multiplicity	Sope / Dependency
VttSysVarInitialValue	0..1	Specifies an optional initial value for this system variable.

6.2.7 VttSysVarInitialValue

VttSysVarInitialValue is a choice container that allows the specification of initial values for system variables depending on the type of the system variable.

Initial	VttInitValue<SysVarType>Value
Description	Specifies the initial value. <SysVarType> is dependent on the type of the system variable.
Multiplicity	1
Type	Same as <SysVarType>
Default Value	0, 0.0, or empty string.

7 Glossary and Abbreviations

7.1 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)

Table 7-1 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com