VECTOR

MICROSAR RTE

VECTOR

Document Information

History

Author	Date	Version	Remarks
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		

	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		-
	- -		-
	- -		
	- -		

	- -		
	- -		
	- -		
	- -		
	- -		-
	- -		-
	- -		
	- -		-
	- -		
	- -		
	- -		
	- -		- -

	- -		
	- -		-
	- -		- -
	- -		
	- -		
	- -		- - - Explicit order of ModeDeclarationGroups

	- -		- -
	- -		

- -

	- -		
	- -		-
	- -		
	- -		
	- -		
	- -		
	- -		- -

-

Reference Documents

No.	Title	Version
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		
[10]		
[11]		
[12]		
[13]		
[14]		
[15]		
[16]		
[17]		
[18]		
[19]		
[20]		
[21]		
[22]		
[23]		
[24]		

-

Scope of the Document

-

Please note

VECTOR

Contents

VECTOR

Illustrations

Tables

1 Component History

Component Version	New Features
	▶ ▶
	▶ - - ▶ ▶ ▶ ▶
	▶ ▶ ▶ - ▶ -g i -g c
	▶ - ▶ ▶ -
	▶ ▶ ▶
	▶ ▶ ▶
	▶
	▶ ▶ ▶ ▶ ▶ ▶ ▶ -

Component Version	New Features
	-
	-
	-
	-
	-

Component Version	New Features
	-
	-m
	-
	-
	-
	-
	-
	-
	-
	-
	-

Component Version	New Features

Component Version	New Features
	▶ ▶ ▶ ▶ ▶
	▶ ▶ ▶ ▶ ▶ ▶ ▶ -
	▶ ▶ ▶ ▶ ▶ ▶ ▶
	▶ ▶ ▶ - ▶ ▶ -
	▶ ▶ ▶ ▶
	▶ ▶
	▶
	▶
	▶ - ▶ ▶

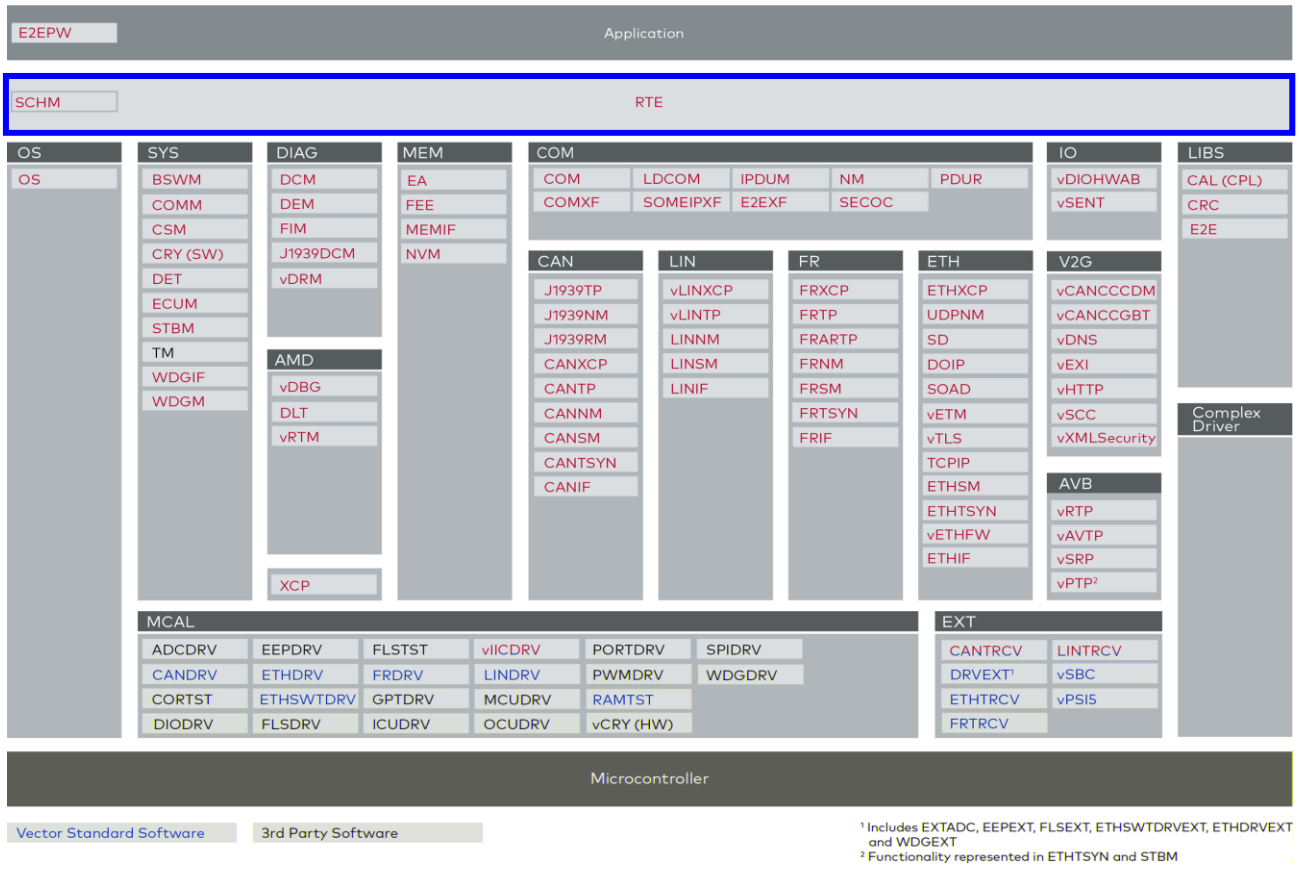
-

2 Introduction

-

Supported AUTOSAR Release*:		
Supported Configuration Variants:	-	
Vendor ID:		-
Module ID:		
AR Version:		
SW Version:		

2.1 Architecture Overview

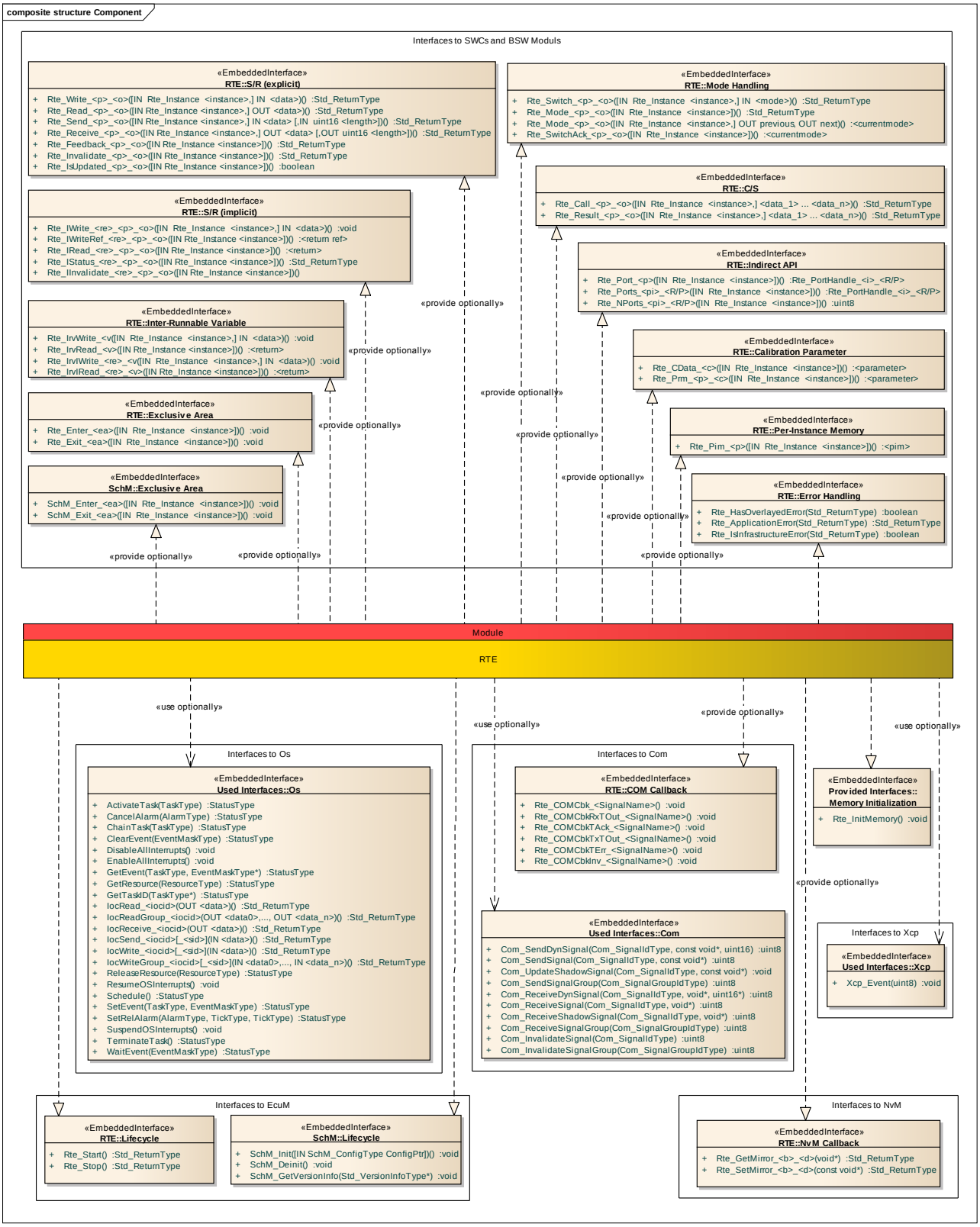


-

-
-
-
-

VECTOR

- -
-
-
-
-
-
-
-


composite structure Component
Interfaces to SWCs and BSW Moduls
«EmbeddedInterface»
RTE::S/R (explicit)
+ Rte_Write_<p>_<o>([IN Rte_Instance <instance>,] IN <data>)() :Std_ReturnType
+ Rte_Read_<p>_<o>([IN Rte_Instance <instance>,] OUT <data>)() :Std_ReturnType
+ Rte_Send_<p>_<o>([IN Rte_Instance <instance>,] IN <data> [,IN uint16 <length>])() :Std_ReturnType
+ Rte_Receive_<p>_<o>([IN Rte_Instance <instance>,] OUT <data> [,OUT uint16 <length>])() :Std_ReturnType
+ Rte_Feedback_<p>_<o>([IN Rte_Instance <instance>])() :Std_ReturnType
+ Rte_Invalidate_<p>_<o>([IN Rte_Instance <instance>])() :Std_ReturnType
+ Rte_IsUpdated_<p>_<o>([IN Rte_Instance <instance>])() :boolean
«EmbeddedInterface»
RTE::Mode Handling
+ Rte_Switch_<p>_<o>([IN Rte_Instance <instance>,] IN <mode>)() :Std_ReturnType
+ Rte_Mode_<p>_<o>([IN Rte_Instance <instance>])() :Std_ReturnType
+ Rte_Mode_<p>_<o>([IN Rte_Instance <instance>,] OUT previous, OUT next)() :<currentmode>
+ Rte_SwitchAck_<p>_<o>([IN Rte_Instance <instance>])() :<currentmode>
«EmbeddedInterface»
RTE::S/R (implicit)
+ Rte_IWrite_<re>_<p>_<o>([IN Rte_Instance <instance>,] IN <data>)() :void
+ Rte_IWriteRef_<re>_<p>_<o>([IN Rte_Instance <instance>])() :<return ref>
+ Rte_IRead_<re>_<p>_<o>([IN Rte_Instance <instance>])() :<return>
+ Rte_IStatus_<re>_<p>_<o>([IN Rte_Instance <instance>])() :Std_ReturnType
+ Rte_IInvalidate_<re>_<p>_<o>([IN Rte_Instance <instance>])()
«EmbeddedInterface»
RTE::C/S
+ Rte_Call_<p>_<o>([IN Rte_Instance <instance>,] <data_1> ... <data_n>)() :Std_ReturnType
+ Rte_Result_<p>_<o>([IN Rte_Instance <instance>,] <data_1> ... <data_n>)() :Std_ReturnType
«EmbeddedInterface»
RTE::Indirect API
+ Rte_Port_<p>([IN Rte_Instance <instance>])() :Rte_PortHandle_<i>_<R/P>
+ Rte_Ports_<pi>_<R/P>([IN Rte_Instance <instance>])() :Rte_PortHandle_<i>_<R/P>
+ Rte_NPorts_<pi>_<R/P>([IN Rte_Instance <instance>])() :uint8
«EmbeddedInterface»
RTE::Inter-Runnable Variable
+ Rte_IrvWrite_<v>([IN Rte_Instance <instance>,] IN <data>)() :void
+ Rte_IrvRead_<v>([IN Rte_Instance <instance>])() :<return>
+ Rte_IrvIWrite_<re>_<v>([IN Rte_Instance <instance>,] IN <data>)() :void
+ Rte_IrvIRead_<re>_<v>([IN Rte_Instance <instance>])() :<return>
«EmbeddedInterface»
RTE::Calibration Parameter
+ Rte_CData_<c>([IN Rte_Instance <instance>])() :<parameter>
+ Rte_Prm_<p>_<c>([IN Rte_Instance <instance>])() :<parameter>
«EmbeddedInterface»
RTE::Exclusive Area
+ Rte_Enter_<ea>([IN Rte_Instance <instance>])() :void
+ Rte_Exit_<ea>([IN Rte_Instance <instance>])() :void
«EmbeddedInterface»
RTE::Per-Instance Memory
+ Rte_Pim_<p>([IN Rte_Instance <instance>])() :<pim>
«EmbeddedInterface»
SchM::Exclusive Area
+ SchM_Enter_<ea>([IN Rte_Instance <instance>])() :void
+ SchM_Exit_<ea>([IN Rte_Instance <instance>])() :void
«EmbeddedInterface»
RTE::Error Handling
+ Rte_HasOverlayedError(Std_ReturnType) :boolean
+ Rte_ApplicationError(Std_ReturnType) :Std_ReturnType
+ Rte_IsInfrastructureError(Std_ReturnType) :boolean
«provide optionally»
Module
RTE
«use optionally»
«provide optionally»
«use optionally»
«use optionally»
Interfaces to Os
«EmbeddedInterface»
Used Interfaces::Os
+ ActivateTask(TaskType) :StatusType
+ CancelAlarm(AlarmType) :StatusType
+ ChainTask(TaskType) :StatusType
+ ClearEvent(EventMaskType) :StatusType
+ DisableAllInterrupts() :void
+ EnableAllInterrupts() :void
+ GetEvent(TaskType, EventMaskType*) :StatusType
+ GetResource(ResourceType) :StatusType
+ GetTaskID(TaskType*) :StatusType
+ IocRead_<iocid>(OUT <data>)() :Std_ReturnType
+ IocReadGroup_<iocid>(OUT <data0>,..., OUT <data_n>)() :Std_ReturnType
+ IocReceive_<iocid>(OUT <data>)() :Std_ReturnType
+ IocSend_<iocid>[_<sid>](IN <data>)() :Std_ReturnType
+ IocWrite_<iocid>[_<sid>](IN <data>)() :Std_ReturnType
+ IocWriteGroup_<iocid>[_<sid>](IN <data0>,..., IN <data_n>)() :Std_ReturnType
+ ReleaseResource(ResourceType) :StatusType
+ ResumeOSInterrupts() :void
+ Schedule() :StatusType
+ SetEvent(TaskType, EventMaskType) :StatusType
+ SetRelAlarm(AlarmType, TickType, TickType) :StatusType
+ SuspendOSInterrupts() :void
+ TerminateTask() :StatusType
+ WaitEvent(EventMaskType) :StatusType
Interfaces to Com
«EmbeddedInterface»
RTE::COM Callback
+ Rte_COMCbk_<SignalName>() :void
+ Rte_COMCbkRxTOut_<SignalName>() :void
+ Rte_COMCbkTAck_<SignalName>() :void
+ Rte_COMCbkTxTOut_<SignalName>() :void
+ Rte_COMCbkTErr_<SignalName>() :void
+ Rte_COMCbkInv_<SignalName>() :void
«EmbeddedInterface»
Used Interfaces::Com
+ Com_SendDynSignal(Com_SignalIdType, const void*, uint16) :uint8
+ Com_SendSignal(Com_SignalIdType, const void*) :uint8
+ Com_UpdateShadowSignal(Com_SignalIdType, const void*) :void
+ Com_SendSignalGroup(Com_SignalGroupIdType) :uint8
+ Com_ReceiveDynSignal(Com_SignalIdType, void*, uint16*) :uint8
+ Com_ReceiveSignal(Com_SignalIdType, void*) :uint8
+ Com_ReceiveShadowSignal(Com_SignalIdType, void*) :uint8
+ Com_ReceiveSignalGroup(Com_SignalGroupIdType) :uint8
+ Com_InvalidateSignal(Com_SignalIdType) :uint8
+ Com_InvalidateSignalGroup(Com_SignalGroupIdType) :uint8
«EmbeddedInterface»
Provided Interfaces::
Memory Initialization
+ Rte_InitMemory() :void
«provide optionally»
Interfaces to Xcp
«EmbeddedInterface»
Used Interfaces::Xcp
+ Xcp_Event(uint8) :void
Interfaces to EcuM
«EmbeddedInterface»
RTE::Lifecycle
+ Rte_Start() :Std_ReturnType
+ Rte_Stop() :Std_ReturnType
«EmbeddedInterface»
SchM::Lifecycle
+ SchM_Init([IN SchM_ConfigType ConfigPtr])() :void
+ SchM_Deinit() :void
+ SchM_GetVersionInfo(Std_VersionInfoType*) :void
Interfaces to NvM
«EmbeddedInterface»
RTE::NvM Callback
+ Rte_GetMirror_<b>_<d>(void*) :Std_ReturnType
+ Rte_SetMirror_<b>_<d>(const void*) :Std_ReturnType

-

3 Functional Description

3.1 Features

- -
- -

- -

Supported AUTOSAR Standard Conform Features
- -
Variable length arrays
-
-
-
- - -
-
-
-
-

Supported AUTOSAR Standard Conform Features
-
Nv Block Software Components
-
Multiple trace clients
-
Background triggered runnable and scheduleable entities (BackgroundEvent)
-
-
-
-
-
- - -
-

Supported AUTOSAR Standard Conform Features
-
-
Optimized S/R communication [API: Rte_DRead]
Variant Handling support (Postbuild selectable for variant data mappings and COM signals)
Data prototype mapping
Subelement mapping for Rx GroupSignals
Bit field texttable mapping
Activation reason for runnable entities (no support for multicore and memory protection)
Service BSW multiple partition distribution
Data conversion (limited to S/R communication with integer network signal(s) mapped to floating point data types on SWC ports, compu methods of type LINEAR or IDENTICAL and data type policy LEGACY or OVERRIDE)

-

3.1.1 Deviations

Not Supported AUTOSAR Standard Conform Features
-
-
external Trigger (via port) [API: Rte_Trigger]
Inter-Runnable Trigger (SWC internal) [API: Rte_IrTrigger]
Tx-Ack for implicit communication [API: Rte_IFeedback]
BSW-Scheduler Mode Handling [API: SchM_Mode, SchM_Switch, SchM_SwitchAck]
external Trigger between BSW modules [API: SchM_Trigger]
BSW-Scheduler Trigger [API: SchM_ActMainFunction]
BSW-Scheduler Calibration Parameter Access [API: SchM_CData]
BSW-Scheduler queued S/R communication [API: SchM_Send, SchM_Receive]
BSW-Scheduler C/S communication [API: SchM_Call, SchM_Result]

Not Supported AUTOSAR Standard Conform Features
BSW-Scheduler Per-Instance Memory Access [API: SchM_Pim]
Enhanced Rte Lifecycle API [API: Rte_StartTiming]
Post Build Variant Sets
Debugging and Logging Support
Variant Handling support (Pre-Compile variability, Postbuild variability for Connectors and ComponentPrototypes)
-
Activation reason in multicore and memory protection systems
Restarting of partitions
Re-scaling of ports / Data conversion
Pre-Build data set generation phase
Post-Build data set generation phase
Initialization of PerInstanceMemories
Asynchronous Mode Handling
MC data support
Generated BSWMD
Range checks
RTE memory section initialization strategies
Configuration of coherency groups for implicit communication
Immediate Buffer update for implicit communication
ScaleLinear and ScaleLinearTexttable CompuMethods with more than one CompuScale

-

3.1.2 Additions/ Extensions

Features Provided Beyond The AUTOSAR Standard
Rte_InitMemory API function. See Chapter 5.14.3 for details.
Init-Runnables. See Chapter 3.6.9 for details.
VFB Trace Hooks for SchM APIs. See Chapter 5.16.3 and 5.16.4 for details.
Measurement support for mode communication. See Chapter 6.6 for details.
Measurement with XCP Events. See Chapter 6.6 for details.

-

3.1.3 Limitations

3.2 Initialization

Rte_Start

SchM_Init

3.3 AUTOSAR ECUs

3.4 AUTOSAR Software Components

- - - -
- -
-
-

-

-

- -
- -
-
-

3.5 Runnable Entities

3.6 Triggering of Runnable Entities

-
-
-
-
-
-
-
-
-
-

-
-

```
Activation of runnable entity
wakeup of waitpoint
```

3.6.1 Time Triggered Runnables

```
                    TimingEvent
TimingEvent
                                  TimingEvent
```

```
 void <RunnableName>([IN Rte_Instance instance]
              [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.6.2 Data Received Triggered Runnables

DataReceivedEvent
- -
Rte_Receive

```
void <RunnableName>([IN Rte_Instance instance]
             [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.6.3 Data Reception Error Triggered Runnables

DataReceiveErrorEvent
aliveTimeout
DataReceiveErrorEvent

DataReceiveErrorEvent

```
void <RunnableName>([IN Rte_Instance instance]
             [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.6.4 Data Send Completed Triggered Runnables

DataSendCompletedEvent
DataSendCompletedEvent
Rte_Feedback

```
void <RunnableName>([IN Rte_Instance instance]
             [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.6.5 Mode Switch Triggered Runnables

ModeSwitchEvent
ModeSwitchEvent
ModeSwitchEvent

```
void <RunnableName>([IN Rte_Instance instance]
             [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.6.6 Mode Switched Acknowledge Triggered Runnables

ModeSwitchedAckEvent
ModeSwitchedAckEvent

```
void <RunnableName>([IN Rte_Instance instance]
             [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.6.7 Operation Invocation Triggered Runnables

OperationInvokedEvent

```
{void|Std_ReturnType} <Runnable>([IN Rte_Instance inst] {,paramlist}*)
```

3.6.8 Asynchronous Server Call Return Triggered Runnables

AsynchronousServerCallReturnsEvent

Rte_Result Rte_Result

```
void <RunnableName>([IN Rte_Instance instance]
             [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.6.9 Init Triggered Runnables

```
void <RunnableName>([IN Rte_Instance instance])
```

3.6.10 Background Triggered Runnables

```
void <RunnableName>([IN Rte_Instance instance]
             [,IN Rte_ActivatingEvent_<RunnableEntity> activation])
```

3.7 Exclusive Areas

-
-
-
-

Info

NONE

CUSTOM SchM_Enter
SchM_Exit

Caution

NONE CUSTOMER
SchM_Enter SchM_Exit

3.7.1 OS Interrupt Blocking

OS_INTERRUPT_BLOCKING
SuspendOSInterrupts()
ResumeOSInterrupts()

Rte_Receive
Rte_Feedback Rte_SwitchAck Rte_Result

3.7.2 All Interrupt Blocking

ALL_INTERRUPT_BLOCKING
SuspendAllInterrupts()
ResumeAllInterrupts()

Rte_Receive
Rte_Feedback Rte_SwitchAck Rte_Result

3.7.3 OS Resource

OS_RESOURCE
GetResource() ReleaseResource()

WaitEvent()

3.7.4 Cooperative Runnable Placement

COOPERATIVE_RUNNABLE_PLACEMENT

3.8 Error Handling

3.8.1 Development Error Reporting

Det_ReportError()
RteGeneration DevErrorDetect
DevErrorDetectUninit).

Det_ReportError()

Service ID	Service

Service ID	Service

-

Error Code	Description

-

DevErrorDetectUninit
DevErrorDetect.

Caution

DevErrorDetect
Det_ReportError

4 RTE Generation and Integration

4.1 Scope of Delivery

Files	Description

-

Info

4.2 RTE Generation

DVCfgCmd.exe

4.2.1 Command Line Options

Option		Description
--project <file>	-p <file>	<file>
--generate	-g	-g
--modulesToGenerate	-m <module>	<module>
--genArg="<module>: <params>"		<params> <module> -
--help	-h	DVCfgCmd.exe

-

4.2.2 RTE Generator Command Line Options

Option	Description
-m <obj>	<obj> <ECUProjectName> <ComponentTypeName> -g i -g c <ComponentTypeName> <ECUProjectName> -m -m
-g [r|c|i|h]	r -m <ECUProjectName> -g -g r

	c	-m <ComponentTypeName/BswModuleName> -m <ComponentType1Name/BswModule1Name>; <ComponentType2Name/BswModule2Name> - -m <ECUProjectName> i -m <ComponentTypeName> -m <ComponentType1Name>;<ComponentType2Name> - -m <ECUProjectName>. -f <file> -f <file> -m <ComponentTypeName>.c
	h	-m <ECUProjectName>. -f <file> -f <file> VFBTraceHook_<ECUProjectName>.c
-o <path> -o r=<path> -o c=<path> -o i=<path> -o h=<path> -o s=<path> -o a=<path> -f <file>		-g <path> -o -g i -g h -g i -m

-v -h	

-

4.2.3 Generation Path

-o

-o

-g -g r

Components

-o

“s=<path>”

-o “a=<path>”

4.3 MICROSAR RTE generation modes

4.3.1 RTE Generation Phase

-g -g r

File	Description
	Components Components Components

	- typeEmitter mak

-

```
DVCfgCmd -p "InteriorLight.dpa" –m /MICROSAR/Rte –g
```

4.3.2 RTE Contract Phase Generation

`-g c`

File	Description
	- `typeEmitter`

-

```
DVCfgCmd -p "InteriorLight.dpa"
    -m /MICROSAR/Rte
    -g
    --genArg=”Rte: -g c –m SenderComponent”
```

Caution

4.3.3 Template Code Generation for Application Software Components

-g i

File	Description
	-g i -f <ComponentTypeName>.c

-

```
DVCfgCmd -p "InteriorLight.dpa"
    -m /MICROSAR/Rte
    -g
    --genArg=”Rte: -g i -m SenderComponent -f Component1.c”
```

Caution

-

4.3.4 VFB Trace Hook Template Code Generation

-g h

File	Description
	-g h -f VFBTraceHook_< ECUProjectName >.c

-

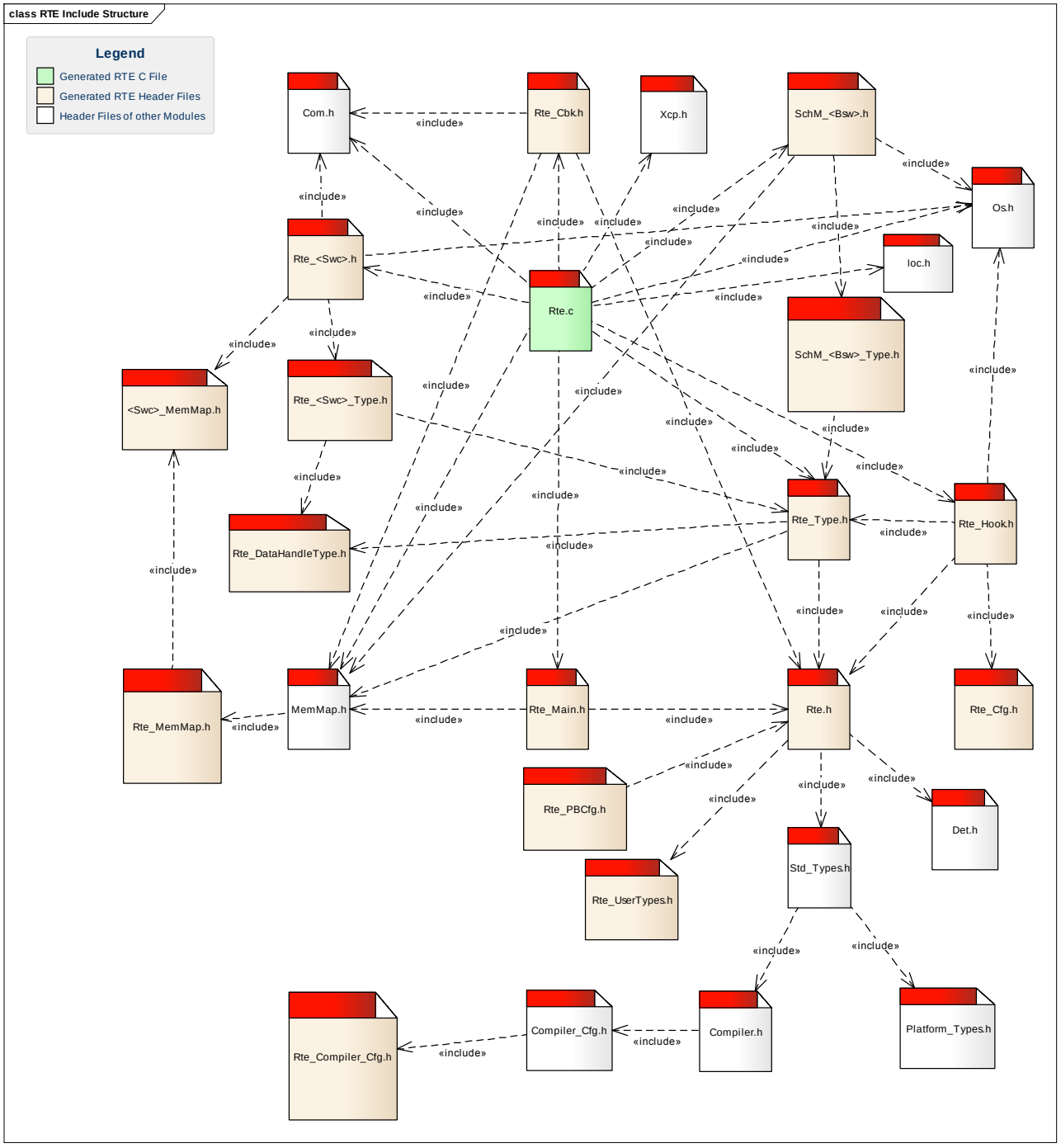
Example:

```
DVCfgCmd -p "InteriorLight.dpa"
    -m /MICROSAR/Rte
    -g
    --genArg=”Rte: -g h –f VFBTraceHook_myEcu.c”
```

Caution

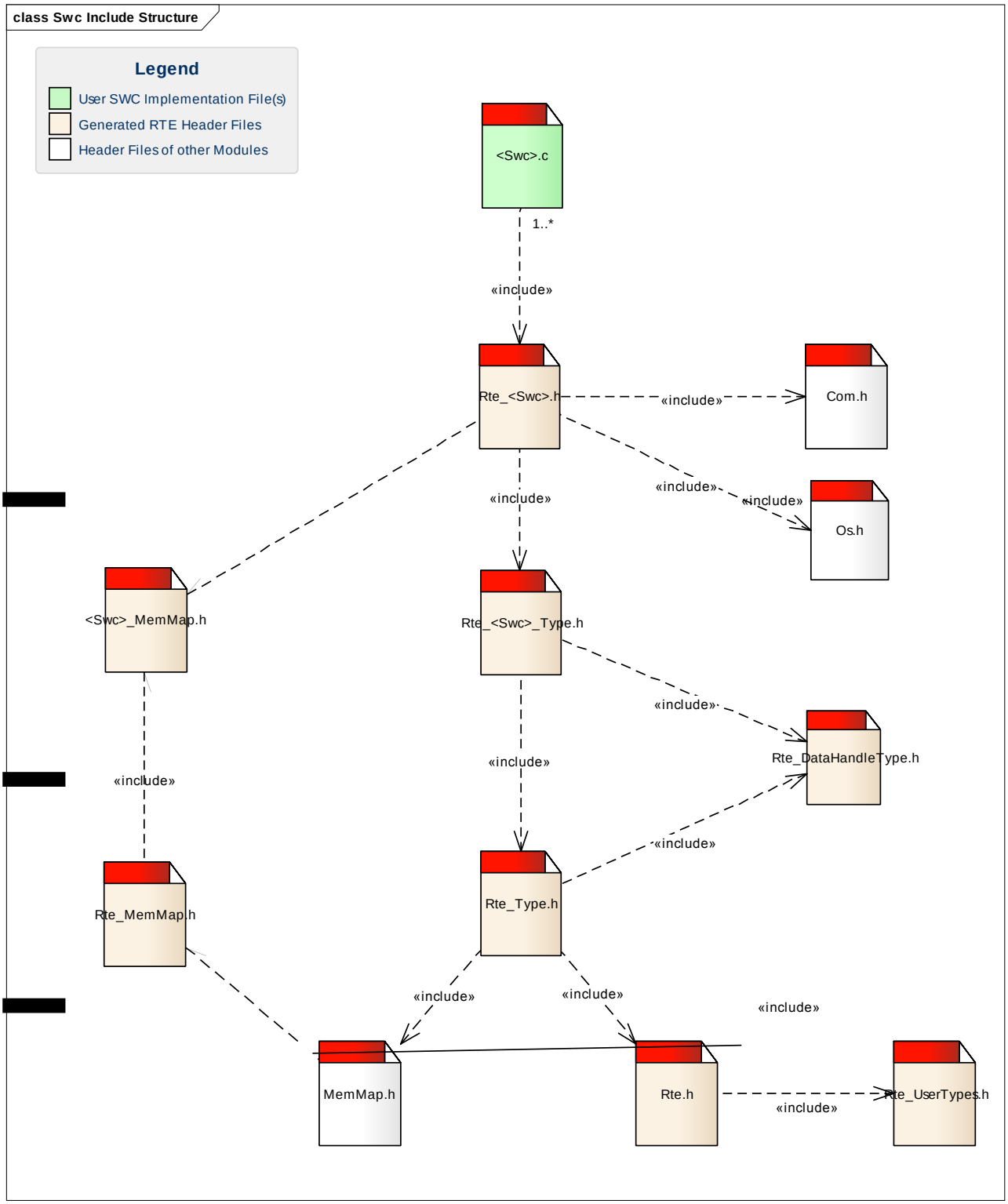
4.4 Include Structure

4.4.1 RTE Include Structure

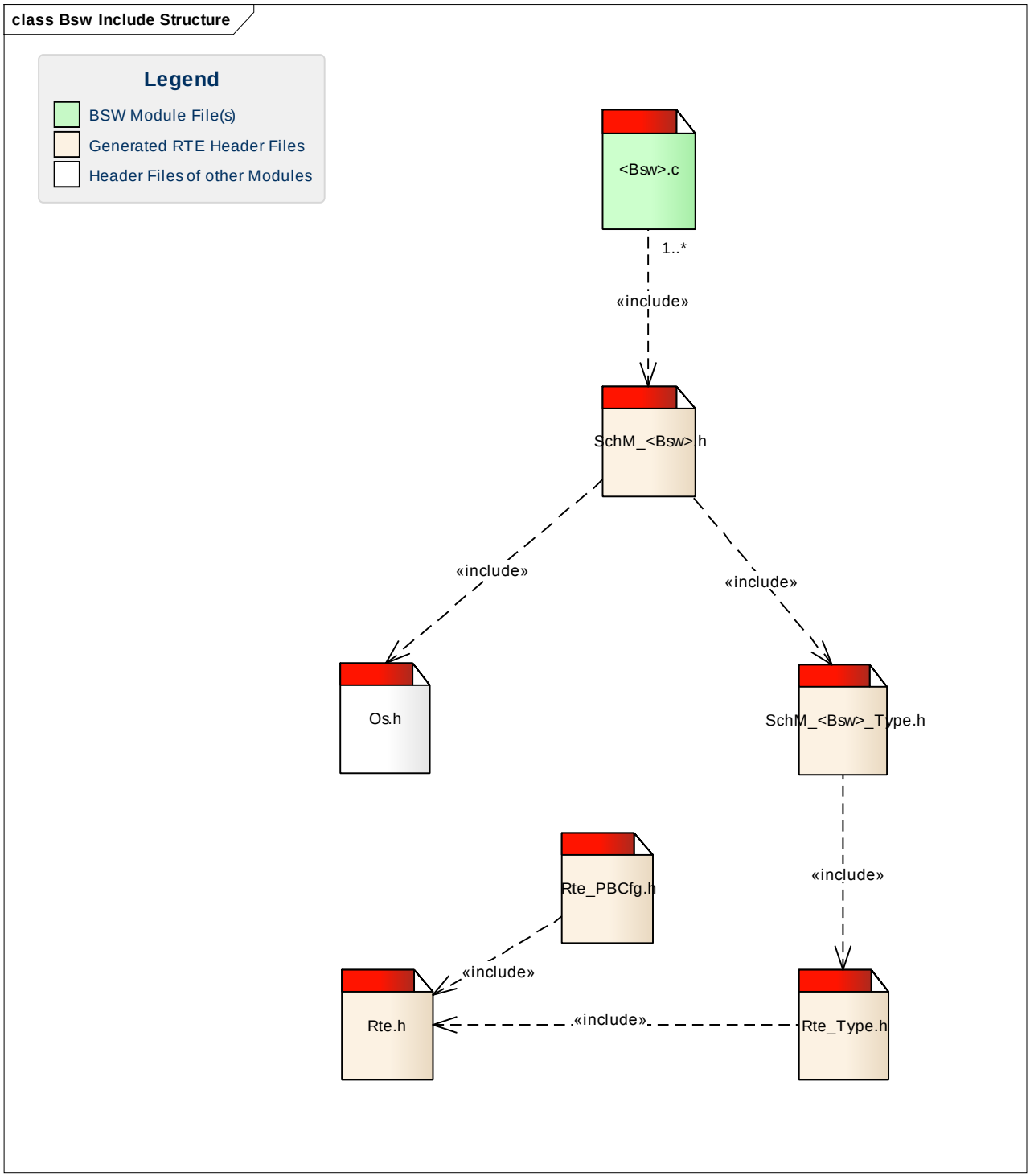


-

4.4.2 SWC Include Structure

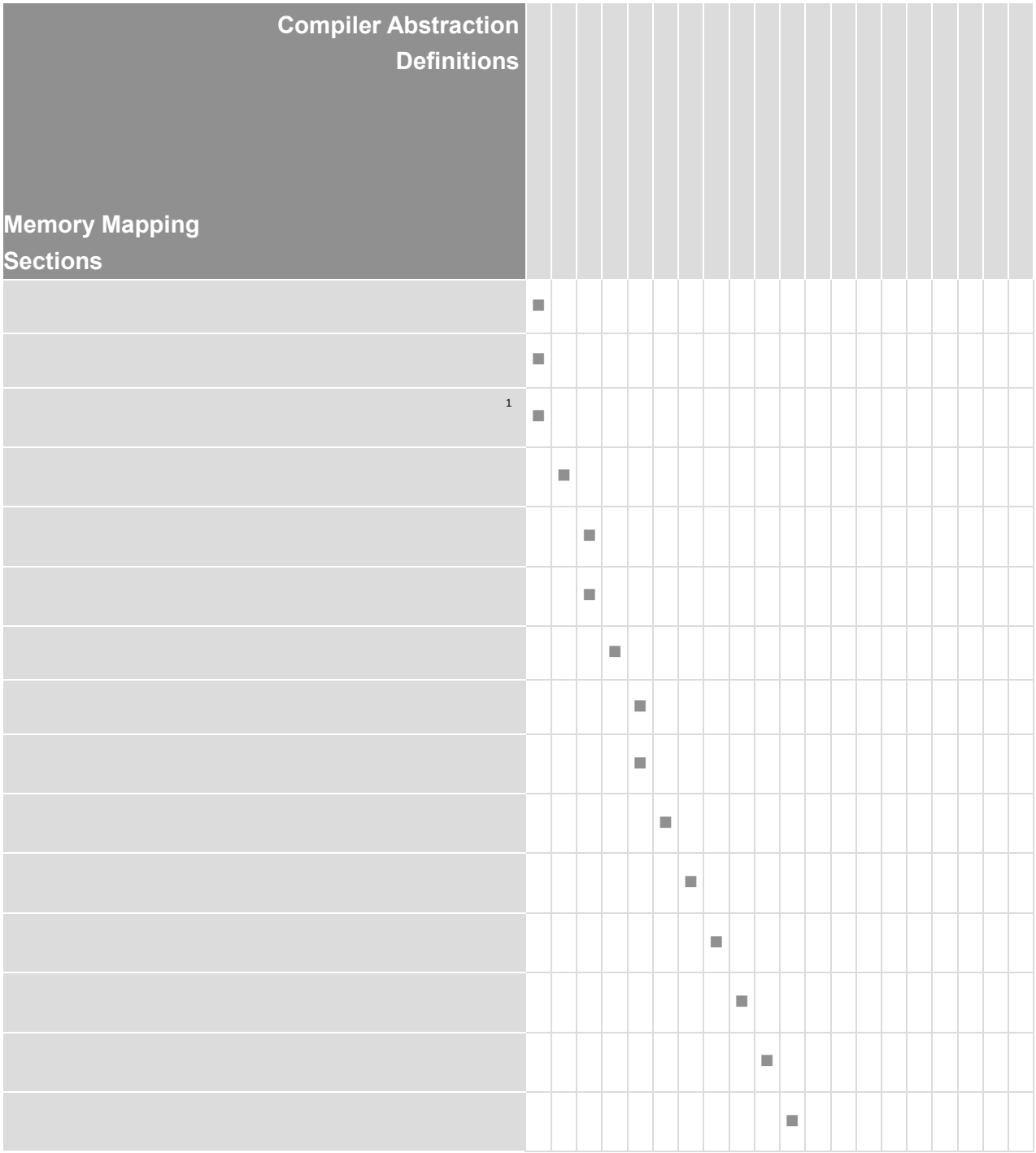


4.4.3 BSW Include Structure

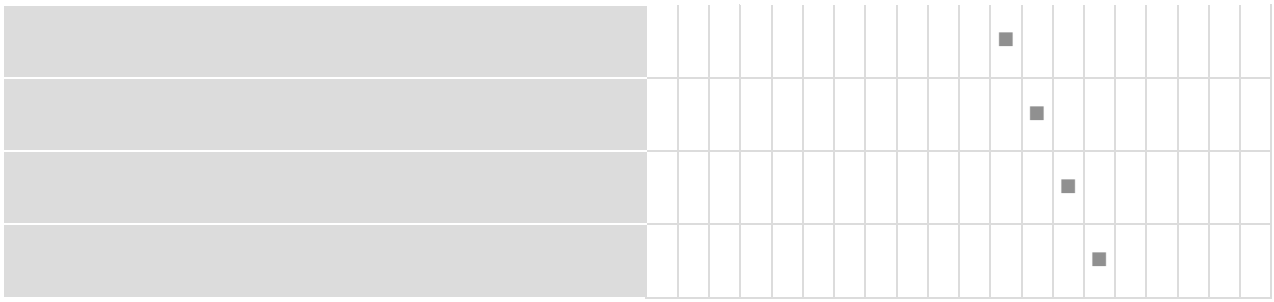


-

4.5 Compiler Abstraction and Memory Mapping



[1] This memory mapping sections are only used if memory protection support is enabled



-

Memory Mapping Sections	Compiler Abstraction Definitions		
	■		
	■		
	■		
	■		
		■	
		■	
			■
			■

- -

-

-

Compiler_Cfg.h MemMap.h

-
-

4.5.1 Memory Sections for Calibration Parameters and Per-Instance Memory

<Cal>

- <Pim>

-

Object Type	Attribute Name	Attribute Type
-		

-

Example for Calibration Parameters:

PAR_GROUP_CAL CalGroupA
CalGroupB

```
RTE_START_SEC_CONST_          _UNSPECIFIED
RTE_STOP_SEC_CONST_          _UNSPECIFIED
RTE_START_SEC_CONST_          _UNSPECIFIED
RTE_STOP_SEC_CONST_          _UNSPECIFIED
```

```
RTE_START_SEC_VAR_CalGroupA_UNSPECIFIED
RTE_STOP_SEC_VAR_CalGroupA_UNSPECIFIED
RTE_START_SEC_VAR_CalGroupB_UNSPECIFIED
RTE_STOP_SEC_VAR_CalGroupB_UNSPECIFIED
```

Example for Per-Instance Memory:

```
            PAR_GROUP_PIM                                   PimGroupA
PimGroupB       -

RTE_START_SEC_VAR_          _UNSPECIFIED
RTE_STOP_SEC_VAR_          _UNSPECIFIED
RTE_START_SEC_VAR_          _UNSPECIFIED
RTE_STOP_SEC_VAR_          _UNSPECIFIED
```

4.5.2 Memory Sections for Software Components

```
              <Swc>
                                                    -g  i

                          MemMap.h
```

4.5.3 Compiler Abstraction Symbols for Software Components and RTE

-

<Swc>_CODE

<Swc>_CODE
<Swc>_CONST
<Swc>_VAR_NOINIT
<Swc>_VAR_INIT
<Swc>_VAR_ZERO_INIT

RTE_APPL_VAR

RTE_<SWC>_APPL_DATA
RTE_APPL_DATA

Caution

<Swc>_MemMap.h, Rte_MemMap.h Rte_Compiler_Cfg.h

-

4.6 Memory Protection Support

These wrapper functions can be implemented manually by following Providing Trusted Functions of the AUTOSAR SWS OS (Version 4.0-4.3).

4.6.1 Partitioning of SWCs

4.6.2 OS Applications

- -
-

-

Caution

Caution

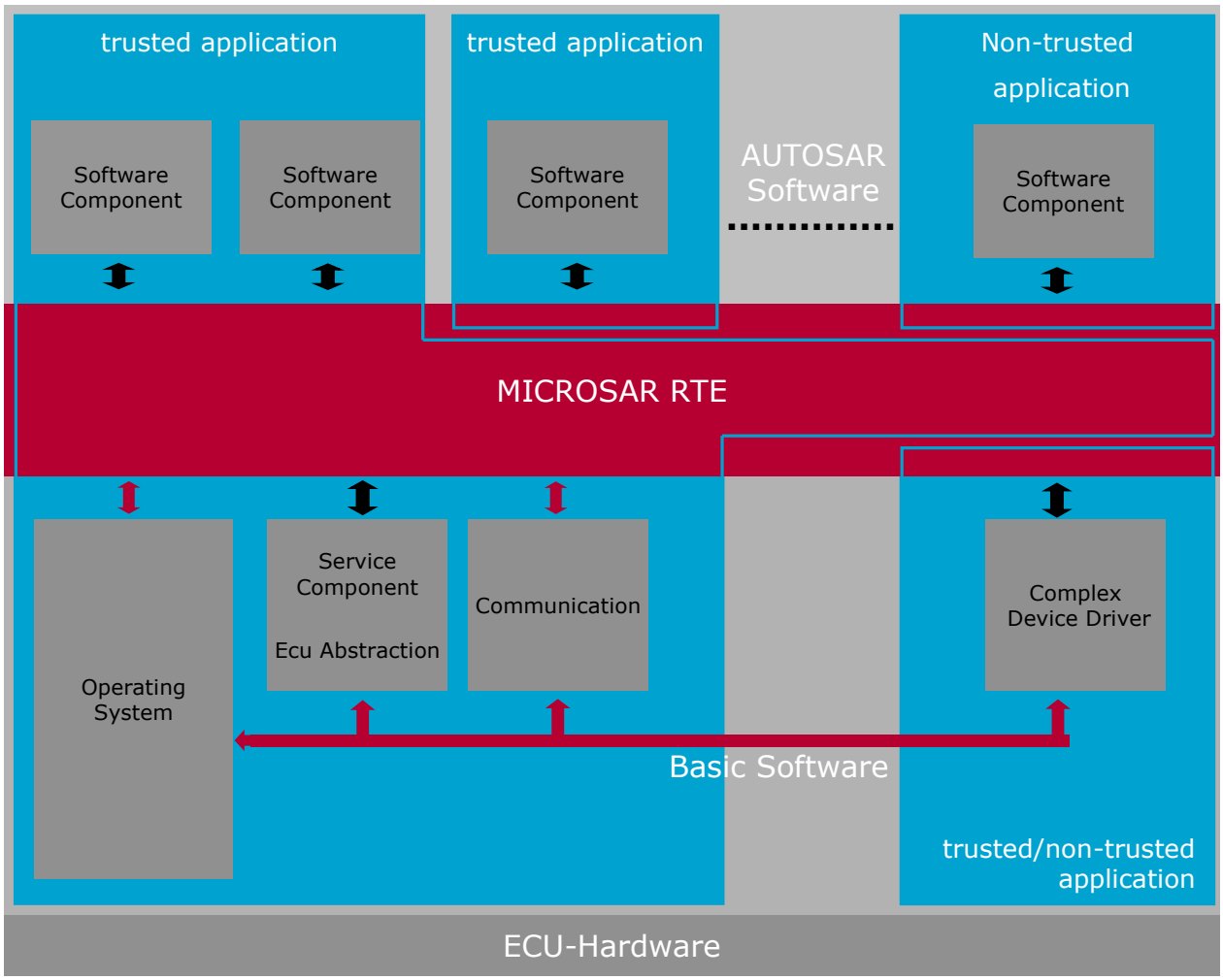
4.6.3 Partitioning Architecture

Caution

-

-

4.6.3.1 Trusted RTE and BSW



-

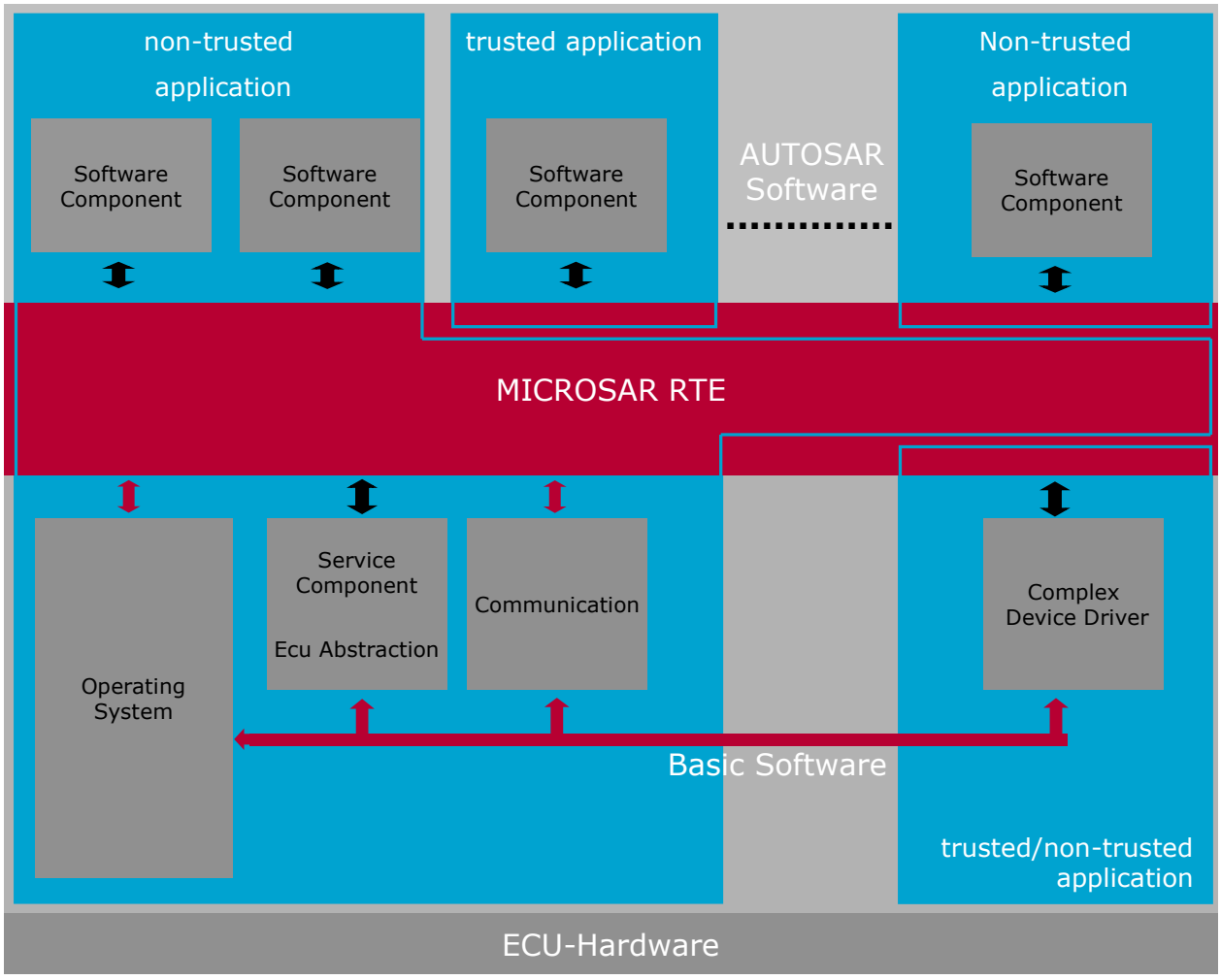
-

-

`Rte_Start()`

-

4.6.3.2 Non-Trusted RTE and BSW



- -

-

-

-

-

-

-

-

-

4.6.4 Conceptual Aspects

-

4.6.5 Memory Protection Integration Hints

4.6.5.1 Enabling of Memory Protection support

4.6.5.2 Memory mapping in Linker Command File

-

Rte_<OsApplicationName>.c

4.6.5.3 OS Configuration extension

-

ActivateTask SetEvent

Rte_Start

4.7 Multicore support

4.7.1 Partitioning of SWCs

4.7.2 BSW in Multicore Systems

Rte_Start(), Rte_Stop(), Rte_InitMemory(), SchM_Init(), SchM_Deinit()

Rte_Start()

Rte_Start()

4.7.3 Service BSW in Multicore Systems

Service SWC: WdgMCore0

- **WdgM_Mainfunction**
- **WdgM_CheckPointReached**

Service SWC: WdgMCore1

- **WdgM_Mainfunction**
- **WdgM_CheckPointReached**

Service SWC: WdgMCore1ASIL

- **WdgM_CheckPointReached**

`Rte_Call` `WdgM_CheckPointReached`

`WdgM_Mainfunction`

Caution

4.7.4 IOC Usage

-

-

-

EnforceIoc

RteGeneration

Caution

NOCACHE

-

4.8 BSW Access in Partitioned systems

4.8.1 Inter-ECU Communication

Rte_ComSendSignalProxyPeriodic

Rte_ComSendSignalProxyPeriodic

Com_MainFunctionTx

Rte_ComSendSignalProxyPeriodic

Com_MainFunctionTx

4.8.2 Client Server Communication

`Rte_Pim()`

5 API Description

Rte_<ComponentType>.h

Info

-

-

Info

- -
enableTakeAddress

-

5.1 Data Type Definition

Std_Types.h Platform_Types.h

typeEmitter

RTE

Rte_UserTypes.h

IncludedDataTypeSet

5.1.1 Invalid Value

`InvalidValue_<literalPrefix><DataType>`

Caution

`Rte_`

5.2 API Error Status

```
Rte_IsInfrastructureError(status)
Rte_HasOverlayedError(status)
Rte_ApplicationError(status)
```

boolean Rte_IsInfrastructure
Rte_HasOverlayedError

Rte_ApplicationError

5.3 Runnable Entities

5.3.1 <RunnableEntity>

Prototype

```
void <              > ( [IN Rte_Instance instance][,
IN Rte_ActivatingEvent_<RunnableEntity> activation])

{Std_ReturnType|void} <              > ( [IN Rte_Instance instance] {,
IN type [*]inputparam}* {, OUT type *outputparam}* )
```

Parameter

supportsMultipleInstantiation	

Return code

Existence

Functional Description

<RunnableEntity>()

▶

-
-
-
-
-
-
  Rte_Result()
-
-
-
-

Call Context

5.4 SWC Exclusive Areas

5.4.1 Rte_Enter

Prototype
void ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
–
Existence
canEnterExclusiveArea
Functional Description
Rte_Enter_<ea>() Rte_Exit_<ea>() Rte_Enter_<ea>() Rte_Exit_<ea>()
Call Context

5.4.2 Rte_Exit

Prototype
void ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
-
Existence
canEnterExclusiveArea
Functional Description
Rte_Exit_<ea>() Rte_Enter_<ea>() Rte_Enter_<ea>() Rte_Exit_<ea>()
Call Context

5.5 BSW Exclusive Areas

5.5.1 SchM_Enter

Prototype

```
void                                    ( void )
```

Parameter

–

Return code

–

Existence

```
canEnterExclusiveArea
```

Functional Description

```
          SchM_Enter_<bsw>_<ea>()


                          SchM_Exit_<bsw>_<ea>()
SchM_Enter_<bsw>_<ea>()                       SchM_Exit_<bsw>_<ea>()
```

Call Context

5.5.2 SchM_Exit

Prototype	
void (void)	
Parameter	
–	
Return code	
–	
Existence	
canEnterExclusiveArea	
Functional Description	
SchM_Exit_<bsw>_<ea>() SchM_Enter_<bsw>_<ea>() SchM_Enter_<bsw>_<ea>() SchM_Exit_<bsw>_<ea>()	
Call Context	

5.6 Sender-Receiver Communication

5.6.1 Rte_Read

Prototype

```
Std_ReturnType                  ( [IN Rte_Instance instance,] OUT <DataType> *data
[, OUT Rte_TransformerError transformerError] )
```

Parameter

supportsMultipleInstantiation

transformerErrorHandling

Return code

aliveTimeout

Existence

dataReceivePointByArgument

isQueued=false

Functional Description

Rte_Read_<p>_<d>()

isQueued=false Rte_Read

Call Context

5.6.2 Rte_DRead

Prototype

```
<DataType>                    ( [IN Rte_Instance instance][, OUT Rte_TransformerError
transformerError] )
```

Parameter

supportsMultipleInstantiation

transformerErrorHandling

Return code

Existence

dataReceivePointByValue

isQueued=false

Functional Description

Rte_DRead_<p>_<d>()

isQueued=false

Rte_DRead

Call Context

5.6.3 Rte_Write

Prototype

```
Std_ReturnType                    ( [IN Rte_Instance instance,] IN <DataType> data
[, OUT Rte_TransformerError transformerError] )

Std_ReturnType                    ( [IN Rte_Instance instance,] IN <DataType> *data
[, OUT Rte_TransformerError transformerError] )
```

Parameter

supportsMultipleInstantiation

transformerErrorHandling

Return code

Existence

isQueued=false

Functional Description

Rte_Write_<p>_<d>()
isQueued=false

Call Context

5.6.4 Rte_Receive

Prototype

```
Std_ReturnType                    ( [IN Rte_Instance instance,] OUT <DataType> *data
[, OUT uint16 *length][, OUT Rte_TransformerError transformerError] )
```

Parameter

supportsMultipleInstantiation

transformerErrorHandling

Return code

-

Existence

isQueued=true

Functional Description

Rte_Receive_<p>_<d>()

isQueued=true

Call Context

5.6.5 Rte_Send

Prototype

```
Std_ReturnType                  ( [IN Rte_Instance instance,] IN <DataType> data
[, OUT Rte_TransformerError transformerError] )

Std_ReturnType                  ( [IN Rte_Instance instance,] IN <DataType> *data
[, IN uint16 length] [, OUT Rte_TransformerError transformerError] )
```

Parameter

supportsMultipleInstantiation

transformerErrorHandling

Return code

Existence

isQueued=true

Functional Description

Rte_Send_<p>_<d>()
isQueued=true

Call Context

5.6.6 Rte_IRead

Prototype
<DataType> ([IN Rte_Instance instance])
<DataType> ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
Existence
Functional Description
Rte_IRead_<r>_<p>_<d>()
isQueued=false. Rte_IRead
Call Context

5.6.7 Rte_IWrite

Prototype

```
void                    ( [IN Rte_Instance instance,] IN <DataType> data )
void                    ( [IN Rte_Instance instance,] IN <DataType> *data )
```

Parameter

supportsMultipleInstantiation

Return code

-

Existence

Functional Description

```
        Rte_IWrite_<r>_<p>_<d>()
   isQueued=false
```

Call Context

Caution

Rte_IWrite Rte_IWriteRef

Rte_IWrite Rte_IWriteRef

5.6.8 Rte_IWriteRef

Prototype

<DataType> ([IN Rte_Instance instance])

Parameter

supportsMultipleInstantiation

Return code

Existence

Functional Description

Rte_IWriteRef_<r>_<p>_<d>()
isQueued=false

Call Context

Caution

Rte_IWrite Rte_IWriteRef

5.6.9 Rte_IStatus

Prototype

```
Std_ReturnType                          ( [IN Rte_Instance instance] )
```

Parameter

supportsMultipleInstantiation

Return code

aliveTimeout

Existence

- aliveTimeout > 0
- data element or
- handleNeverReceived

Functional Description

Rte_IStatus_<r>_<p>_<d>()
Rte_IRead

Call Context

5.6.10 Rte_Feedback

Prototype
`Std_ReturnType ( [IN Rte_Instance instance] )`
Parameter
`supportsMultipleInstantiation`
Return code
-
-
-
Existence
Functional Description
`Rte_Feedback_<p>_<d>()` `Rte_Write()` `Rte_Send()` -
Call Context

5.6.11 Rte_IsUpdated

Prototype
boolean ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
Existence
EnableUpdate
Functional Description
Rte_IsUpdated_<p>_<d>()
Call Context

5.7 Data Element Invalidation

5.7.1 Rte_Invalidate

Prototype

```
Std_ReturnType                          ( [IN Rte_Instance instance]
[, OUT Rte_TransformerError transformerError] )
```

Parameter

supportsMultipleInstantiation

transformerErrorHandling

Return code

Existence

-

CanInvalidate=true

Functional Description

Rte_Invalidate_<p>_<d>()

-

Call Context

5.7.2 Rte_IInvalidate

Prototype
void ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
-
Existence
CanInvalidate=true
Functional Description
Rte_IInvalidate_<r>_<p>_<d>()
Call Context

5.8 Mode Management

5.8.1 Rte_Switch

Prototype

```
Std_ReturnType                    ( [IN Rte_Instance instance,]
IN Rte_ModeType_<ModeDeclarationGroup> mode )
```

Parameter

supportsMultipleInstantiation	
	Rte_ModeType_<m>

Return code

Existence

Functional Description

Rte_Switch_<p>_<m>()

Call Context

5.8.2 Rte_Mode

Prototype
`Rte_ModeType_<ModeDeclarationGroup>  ( [IN Rte_Instance instance] )`
Parameter
`supportsMultipleInstantiation`
Return code
<m> indicates the current
Existence
Functional Description
`Rte_Mode_<p>_<m>()`
Call Context

5.8.3 Enhanced Rte_Mode

Prototype

```
Rte_ModeType_<ModeDeclarationGroup>                ( [IN Rte_Instance instance],
                                OUT Rte_ModeType_<ModeDeclarationGroup> previousMode,
                                OUT Rte_ModeType_<ModeDeclarationGroup> nextMode )
```

Parameter

	supportsMultipleInstantiation
previousMode	
nextMode	

Return code

	<m> indicates the current

Existence

Functional Description

```
Rte_Mode_<p>_<m>()
```

Call Context

5.8.4 Rte_SwitchAck

Prototype
`Std_ReturnType ( [IN Rte_Instance instance] )`
Parameter
`supportsMultipleInstantiation`
Return code
-
Existence
Functional Description
`Rte_SwitchAck_<p>_<m>()`
`Rte_Switch` -
Call Context

5.9 Inter-Runnable Variables

5.9.1 Rte_IrvRead

Prototype
<DataType> ([IN Rte_Instance instance])
void ([IN Rte_Instance instance,] OUT <DataType> *data)
Parameter
supportsMultipleInstantiation -
Return code
- -
Existence
-
Functional Description
Rte_IrvRead_<r>_<v>() - - . Rte_IrvRead
Call Context

5.9.2 Rte_IrvWrite

Prototype	
void	([IN Rte_Instance instance,] IN <DataType> data)
void	([IN Rte_Instance instance,] IN <DataType> *data)
Parameter	
	supportsMultipleInstantiation

5.9.3 Rte_IrvIRead

Prototype
<DataType> ([IN Rte_Instance instance])
<DataType> * ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
-
-
-
-
Existence
-
Functional Description
Rte_IrvIRead_<r>_<v>() -
- . Rte_IrvIRead
Call Context

5.9.4 Rte_IrvIWrite

Prototype

```
void                    ( [IN Rte_Instance instance,] IN <DataType> data )
void                    ( [IN Rte_Instance instance,] IN <DataType> *data)
```

Parameter

supportsMultipleInstantiation

-

-

Return code

-

Existence

-

Functional Description

Rte_IrvIWrite_<r>_<v>() -

Call Context

Caution

-

-

Rte_IrvIWrite

-

5.10 Per-Instance Memory

5.10.1 Rte_Pim

Prototype
<C-type> ([IN Rte_Instance instance])
<DataType> ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
- - - -
Existence
-
Functional Description
Rte_Pim_<n>() - -
Call Context
-

Caution

uint8

Rte_UserTypes.h

- -

5.11 Calibration Parameters

5.11.1 Rte_CData

Prototype
<DataType> ([IN Rte_Instance instance])
<DataType> ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
Existence
Functional Description
Rte_CData_<cp>() shared perInstance
Call Context

5.11.2 Rte_Prm

Prototype
<DataType> ([IN Rte_Instance instance])
<DataType> ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
Existence
Functional Description
Rte_Prm_<p>_<cp>()
Call Context

5.12 Client-Server Communication

5.12.1 Rte_Call

Prototype

```
Std_ReturnType                    ( [IN Rte_Instance instance,]
{IN type [*]inputparam,}* {OUT type *outputparam,}* {INOUT type *inoutputparam,}* )
```

Parameter

supportsMultipleInstantiation

-

Return code

Existence

Functional Description

Rte_Call_<p>_<o>()

Rte_Call

Call Context

5.12.2 Rte_Result

Prototype

```
Std_ReturnType                    ( [IN Rte_Instance instance,]
{OUT type *outputparam,}* {INOUT type *inoutputparam,}* )
```

Parameter

supportsMultipleInstantiation

-

Return code

-

Existence

Functional Description

Rte_Result_<p>_<o>()

Call Context

5.13 Indirect API

5.13.1 Rte_Ports

Prototype
Rte_PortHandle_<i>_<R/P> ([IN Rte_Instance instance])
Parameter
supportsMultipleInstantiation
Return code
Existence
Functional Description
Rte_Ports_<i>_<R/P>
Call Context

5.13.2 Rte_NPorts

Prototype
`uint8 ( [IN Rte_Instance instance] )`
Parameter
`supportsMultipleInstantiation`
Return code
`uint8 Rte_Ports`
Existence
Functional Description
`Rte_NPorts_<i>_<R/P>`
Call Context

5.13.3 Rte_Port

Prototype
`Rte_PortHandle_<i>_<R/P>  ( [IN Rte_Instance instance] )`
Parameter
`supportsMultipleInstantiation`
Return code
Existence
Functional Description
`Rte_Port_<p>`
Call Context

5.14 RTE Lifecycle API

Rte_Main.h

5.14.1 Rte_Start

Prototype

Std_ReturnType (void)

Parameter

-

Return code

Functional Description

Rte_Start

Call Context

5.14.2 Rte_Stop

Prototype

Std_ReturnType (void)

Parameter

-

Return code

Functional Description

Rte_Stop

Rte_Stop

Call Context

5.14.3 Rte_InitMemory

Prototype
`void  ( void )`
Parameter
-
Return code
-
Functional Description
`Rte_InitMemory`
Call Context

Caution

5.15 SchM Lifecycle API

Rte_Main.h

5.15.1 SchM_Init

Prototype

```
void            ( [IN SchM_ConfigType ConfigPtr] )
```

Parameter

Return code

-

Functional Description

Call Context

5.15.2 SchM_Deinit

Prototype

```
void              ( void )
```

Parameter

-

Return code

-

Functional Description

Call Context

5.15.3 SchM_GetVersionInfo

Prototype
void (Std_VersionInfoType *versioninfo)
Parameter
Return code
-
Existence
RteSchMVersionInfoApi
Functional Description
SchM_GetVersionInfo() -
Call Context

5.16 VFB Trace Hooks

5.16.1 Rte_[<client>_]<API>Hook_<cts>_<ap>_Start

Prototype

```
void                                              ( [IN const Rte_CDS_<cts>* inst,]
params )
```

Parameter

supportsMultipleInstantiation

Return code

-

Existence

Functional Description

-
-
-
-
-

Call Context

5.16.2 Rte_[<client>_]<API>Hook_<cts>_<ap>_Return

Prototype

```
void                                              ( [IN const Rte_CDS_<cts> *inst,]
params )
```

Parameter

supportsMultipleInstantiation

Return code

-

Existence

Functional Description

-
-
-
-
-

Call Context

Caution

Caution

-
- -
- -
-
- -

5.16.3 SchM_[<client>_]<API>Hook_<Bsw>_<ap>_Start

Prototype
`void ( params )`
Parameter
Return code
-
Existence
Functional Description
    -
Call Context

Caution

5.16.4 SchM_[<client>_]<API>Hook_<Bsw>_<ap>_Return

Prototype
`void ( params )`
Parameter
Return code
-
Existence
Functional Description
▶
Call Context

Caution

5.16.5 Rte_[<client>_]ComHook_<SignalName>_SigTx

Prototype

```
void                                        ( <DataType> *data )
```

Parameter

Rte_Send
Rte_Write Rte_IWrite

Return code

-

Existence

-

Functional Description

Com_SendSignal or
Com_UpdateShadowSignal.

Call Context

Rte_Send Rte_Write

Rte_IWrite

5.16.6 Rte_[<client>_]ComHook_<SignalName>_SigIv

Prototype
void (void)
Parameter
-
Return code
-
Existence
- canInvalidate
Functional Description
Com_InvalidateSignal.
Call Context
Rte_Invalidate Rte_IInvalidate

5.16.7 Rte_[<client>_]ComHook_<SignalName>_SigGroupIv

Prototype
void (void)
Parameter
-
Return code
-
Existence
- canInvalidate
Functional Description
Com_InvalidateSignalGroup.
Call Context
Rte_Invalidate Rte_IInvalidate

5.16.8 Rte_[<client>_]ComHook_<SignalName>_SigRx

Prototype

```
void                                        ( <DataType> *data )
```

Parameter

Rte_Receive Rte_Read Rte_DRead Rte_IRead

Return code

-

Existence

Functional Description

Com_ReceiveSignal or Com_ReceiveShadowSignal.

Call Context

Rte_Read Rte_DRead

Rte_IRead

Rte_Receive

5.16.9 Rte_[<client>_]ComHook<Event>_<SignalName>

Prototype

```
void                                                ( void )
```

Parameter

-

Return code

-

Existence

Functional Description

-
-
-
-
-
-

Call Context

5.16.10 Rte_[<client>_]Task_Activate

Prototype
void (TaskType task)
Parameter
ActivateTask
Return code
-
Existence
ActivateTask
Functional Description
ActivateTask
Call Context
Rte_Start

5.16.11 Rte_[<client>_]Task_Dispatch

Prototype
void (TaskType task)
Parameter
Return code
-
Existence
Functional Description
Call Context

5.16.12 Rte_[<client>_]Task_SetEvent

Prototype
void (TaskType task, EventMaskType event)
Parameter
SetEvent
SetEvent
Return code
-
Existence
SetEvent
Functional Description
SetEvent
Call Context

5.16.13 Rte_[<client>_]Task_WaitEvent

Prototype
void (TaskType task, EventMaskType event)
Parameter
WaitEvent
WaitEvent
Return code
-
Existence
WaitEvent
Functional Description
WaitEvent
Call Context

5.16.14 Rte_[<client>_]Task_WaitEventRet

Prototype
void (TaskType task, EventMaskType event)
Parameter
WaitEvent
WaitEvent
Return code
-
Existence
WaitEvent
Functional Description
WaitEvent
Call Context

5.16.15 Rte_[<client>_]Runnable_<cts>_<re>_Start

Prototype
void ([IN const Rte_CDS_<cts> *inst])
Parameter
supportsMultipleInstantiation
Return code
-
Existence
Functional Description
Rte_[<client>_]Runnable_<cts>_<re>_Return.
Call Context

5.16.16 Rte_[<client>_]Runnable_<cts>_<re>_Return

Prototype
`void ( [IN const Rte_CDS_<cts> *inst] )`
Parameter
`supportsMultipleInstantiation`
Return code
-
Existence
Functional Description
`Rte_[<client>_]Runnable_<cts>_<re>_Start.`
Call Context

5.17 RTE Interfaces to BSW

5.17.1 Interface to COM / LDCOM

Used COM API

Used LDCOM API

`Rte_Cbk.h`

Caution

Com.h/LdCom.h

5.17.2 Interface to Transformer

Used Transformer API

Caution

5.17.3 Interface to OS

Used OS API

Os.h

Rte_Needs.ecuc.arxml

Rte.oil

Caution

Rte_Needs.ecuc.arxml Rte.oil

5.17.4 Interface to NVM

NvM_MainFunction

Rte_Cbk.h

Caution

NvM_MainFunction

5.17.5 Interface to XCP

Xcp.h

Used Xcp API

5.17.6 Interface to SCHM

`Rte_ComSendSignalProxyPeriodic`

Provided Schedulable Entity

5.17.7 Interface to DET

Used DET API

6 RTE Configuration

-

-

6.1 Configuration Variants

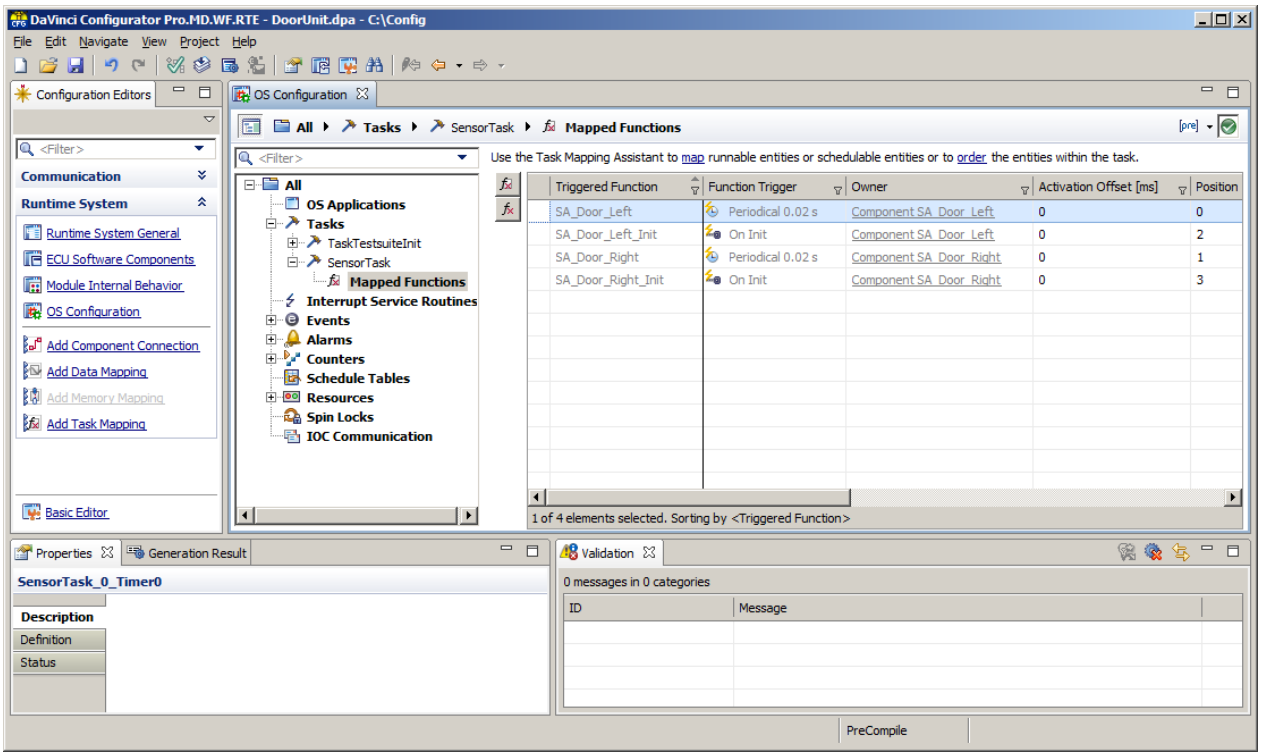
- VARIANT-PRE-COMPILE
- VARIANT-POST-BUILD-SELECTABLE

Rte_bswmd.arxml

6.2 Task Configuration

TimingEvent
OperationInvokedEvent
CanBeInvokedConcurrently

Task Mapping Assistant



-

BASIC EXTENDED

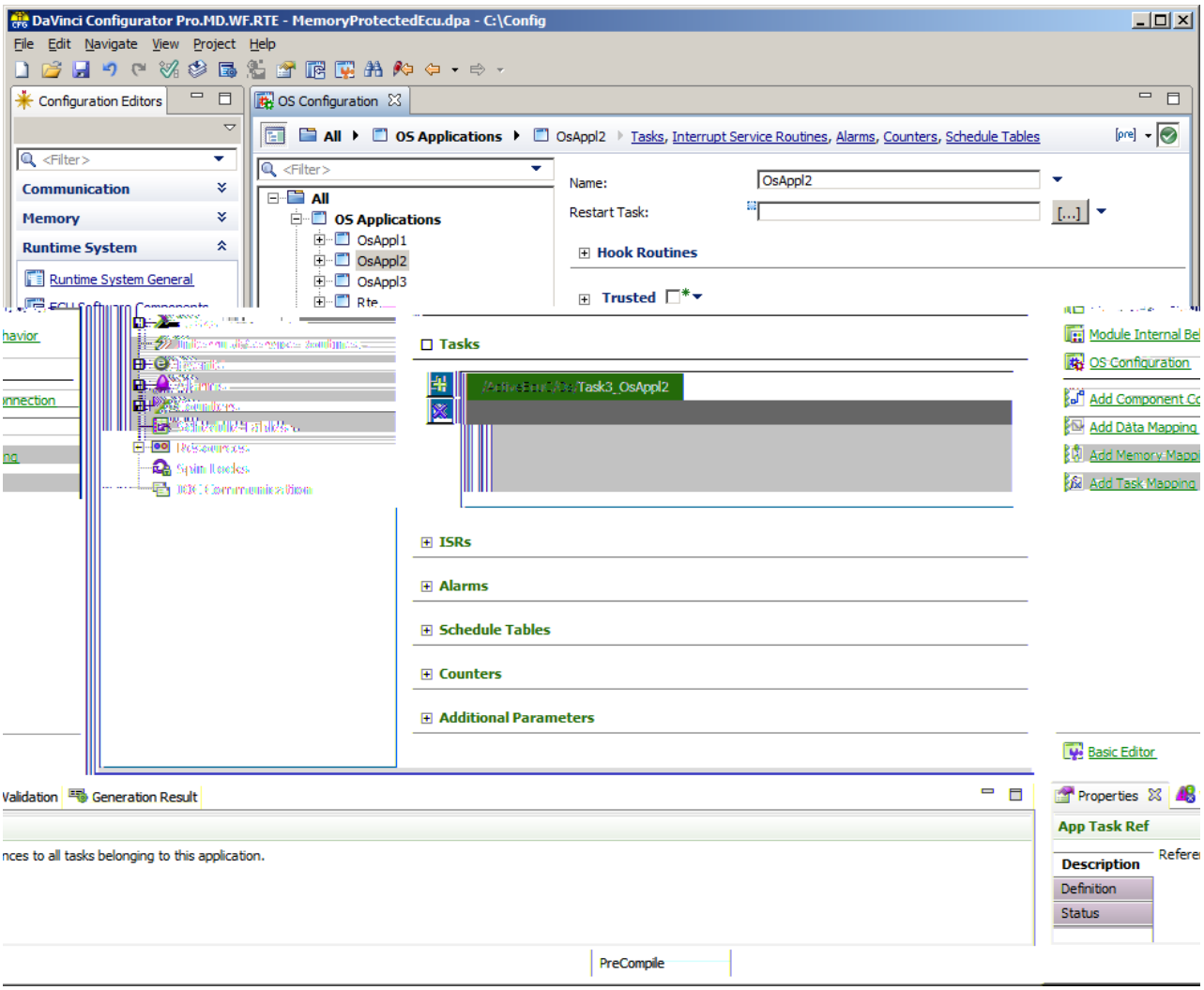
AUTO

BASIC

Rte_Feedback()

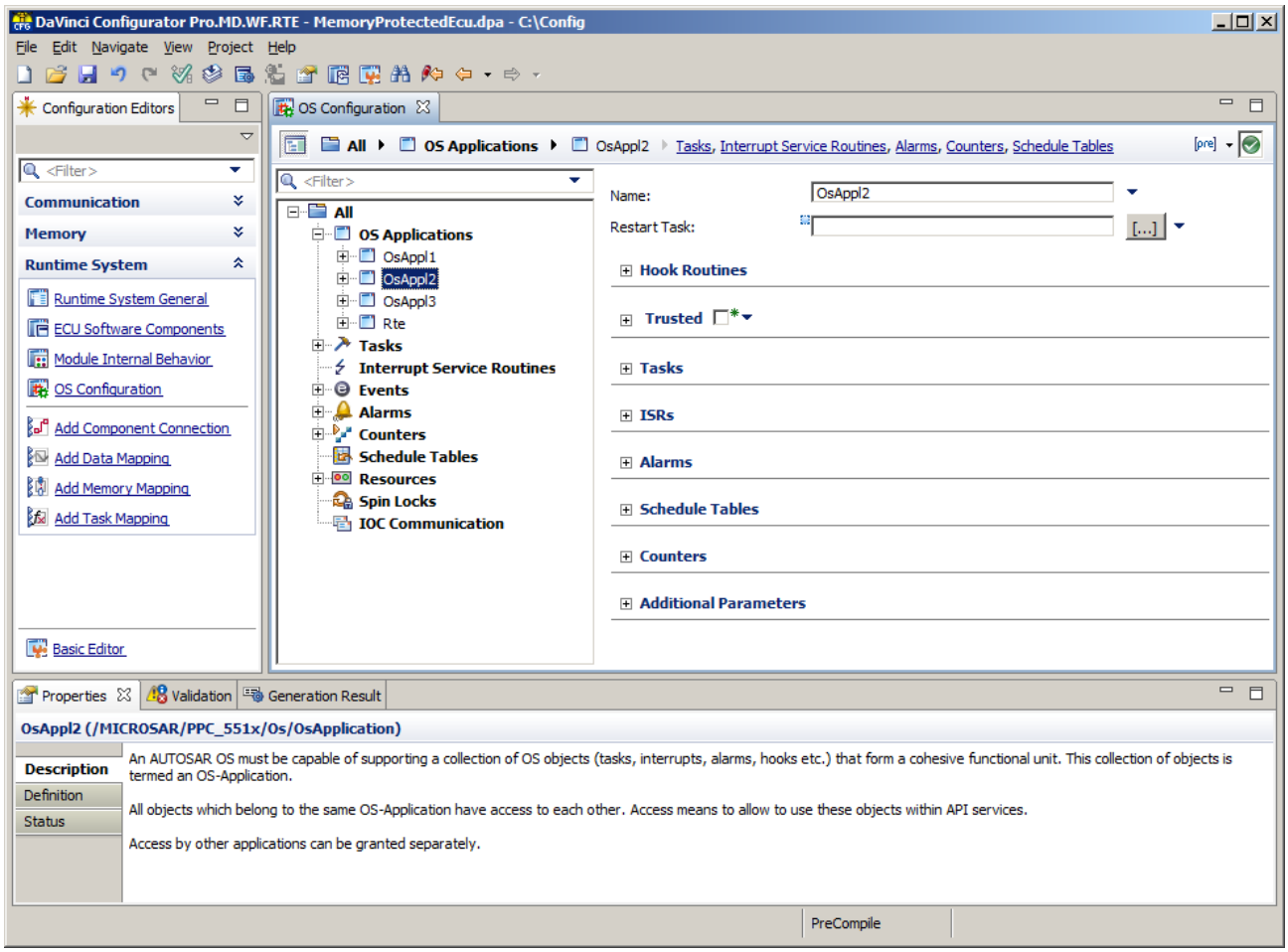
Caution

6.3 Memory Protection and Multicore Configuration


DaVinci Configurator Pro.MD.WF.RTE - MemoryProtectedEcu.dpa - C:\Config
File Edit Navigate View Project Help
Configuration Editors
OS Configuration
All OS Applications OsAppl2 Tasks, Interrupt Service Routines, Alarms, Counters, Schedule Tables
Communication
Memory
Runtime System
Runtime System General
OsAppl1
OsAppl2
OsAppl3
Name: OsAppl2
Restart Task:
Hook Routines
Trusted
Tasks
Task3_OsAppl2
ISRs
Alarms
Schedule Tables
Counters
Additional Parameters
Module Internal Be
OS Configuration
Add Component Co
Add Data Mapping
Add Memory Mappi
Add Task Mapping
Basic Editor
Validation
Generation Result
nces to all tasks belonging to this application.
Properties
App Task Ref
Description
Definition
Status
PreCompile

-

Caution


DaVinci Configurator Pro.MD.WF.RTE - MemoryProtectedEcu.dpa - C:\Config
File Edit Navigate View Project Help
Configuration Editors
Communication
Memory
Runtime System
Runtime System General
ECU Software Components
Module Internal Behavior
OS Configuration
Add Component Connection
Add Data Mapping
Add Memory Mapping
Add Task Mapping
Basic Editor
OS Configuration
All OS Applications OsAppl2 Tasks, Interrupt Service Routines, Alarms, Counters, Schedule Tables
All
OS Applications
OsAppl1
OsAppl2
OsAppl3
Rte
Tasks
Interrupt Service Routines
Events
Alarms
Counters
Schedule Tables
Resources
Spin Locks
IOC Communication
Name: OsAppl2
Restart Task:
Hook Routines
Trusted
Tasks
ISRs
Alarms
Schedule Tables
Counters
Additional Parameters
Properties
Validation
Generation Result
OsAppl2 (/MICROSAR/PPC_551x/Os/OsApplication)
Description
Definition
Status
An AUTOSAR OS must be capable of supporting a collection of OS objects (tasks, interrupts, alarms, hooks etc.) that form a cohesive functional unit. This collection of objects is termed an OS-Application.
All objects which belong to the same OS-Application have access to each other. Access means to allow to use these objects within API services.
Access by other applications can be granted separately.
PreCompile

-

6.4 NV Memory Mapping

- **Needs memory mapping**

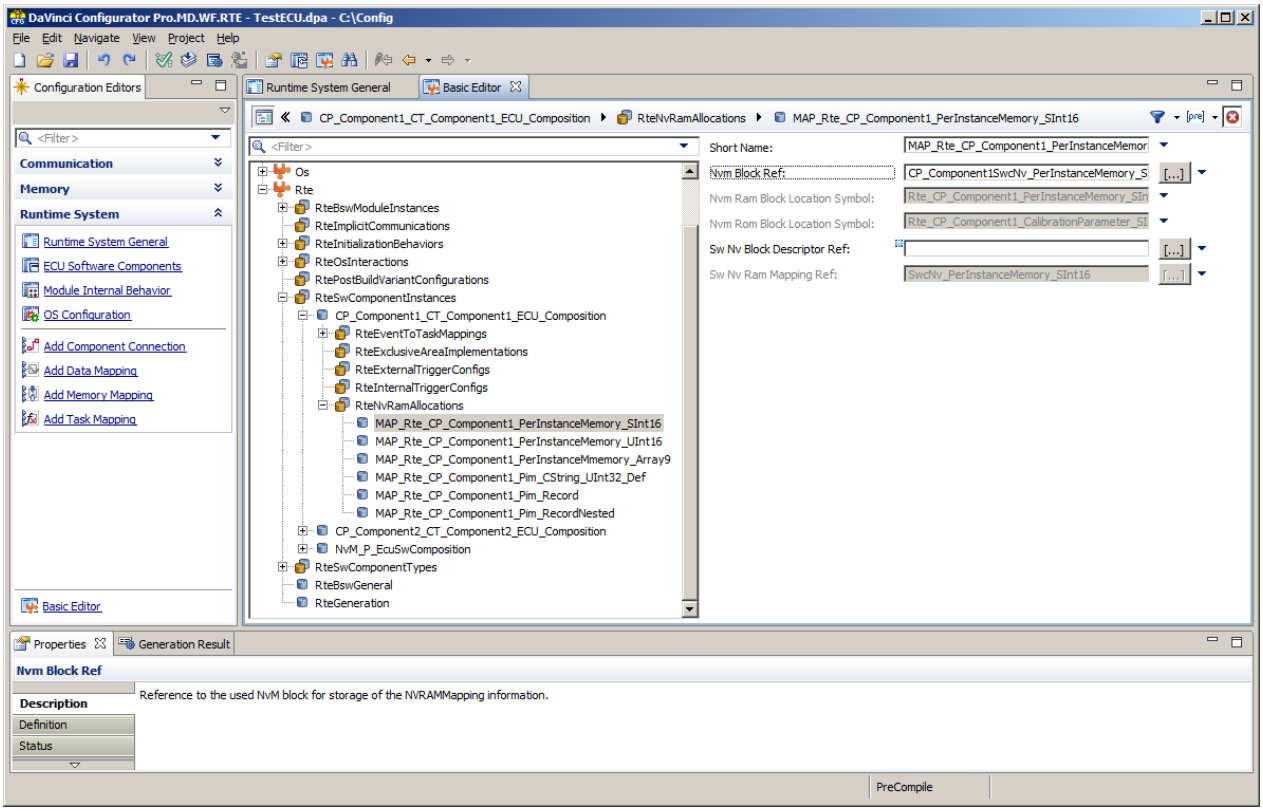
-

`NvM_ReadAll`

`NvM_WriteAll`

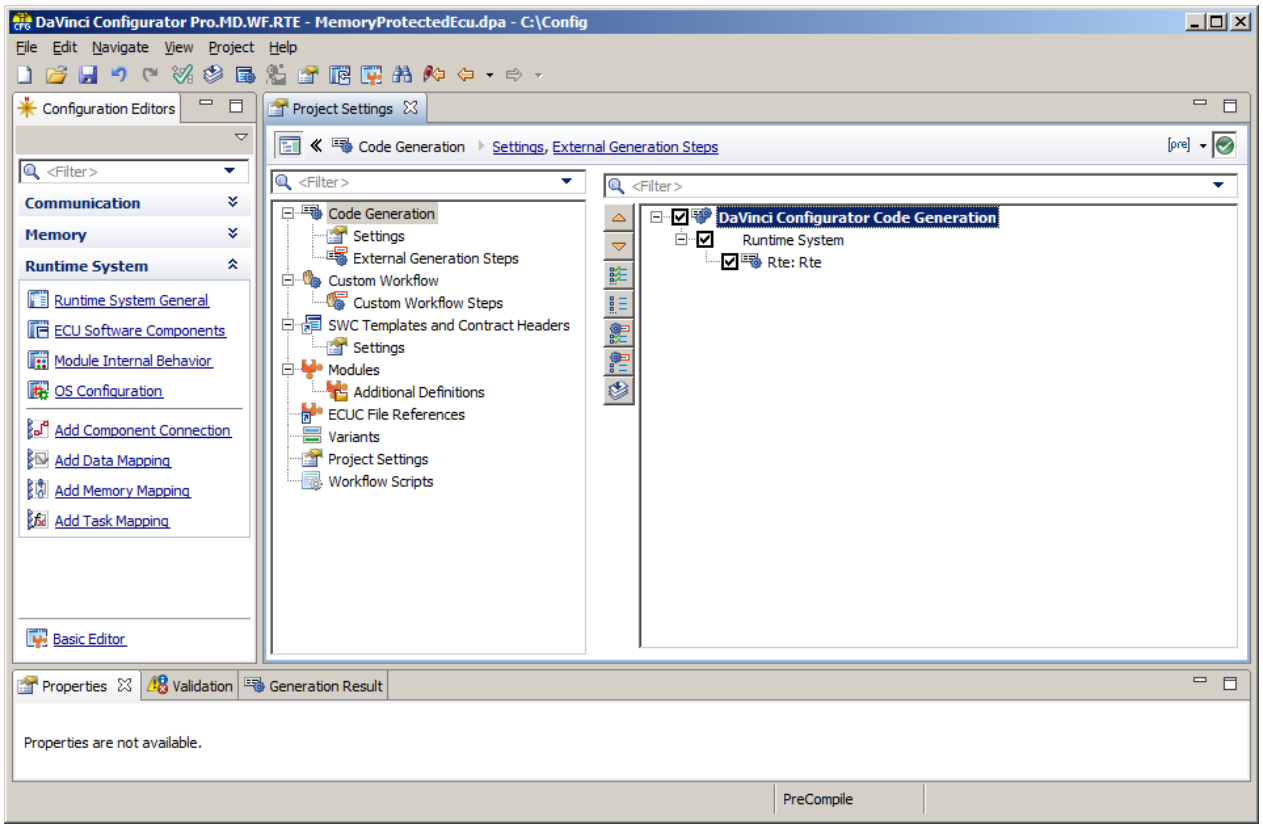
Memory Mapping

-



- -

6.5 RTE Generator Settings

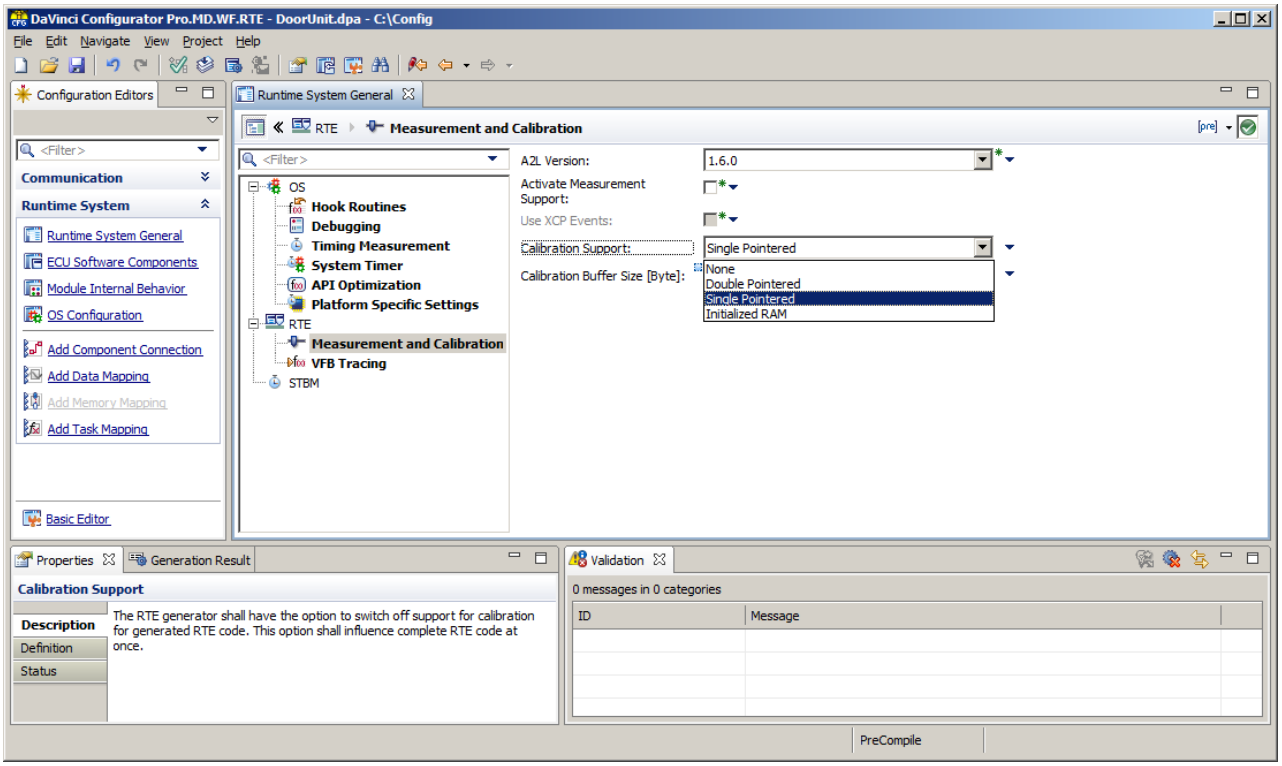


6.6 Measurement and Calibration

-

Rte.a2l

- -



-

- Rte.a2l

Rte.a2l

-

-

Rte.a2l

Rte.a2l -

Rte.a2l

-

-

-

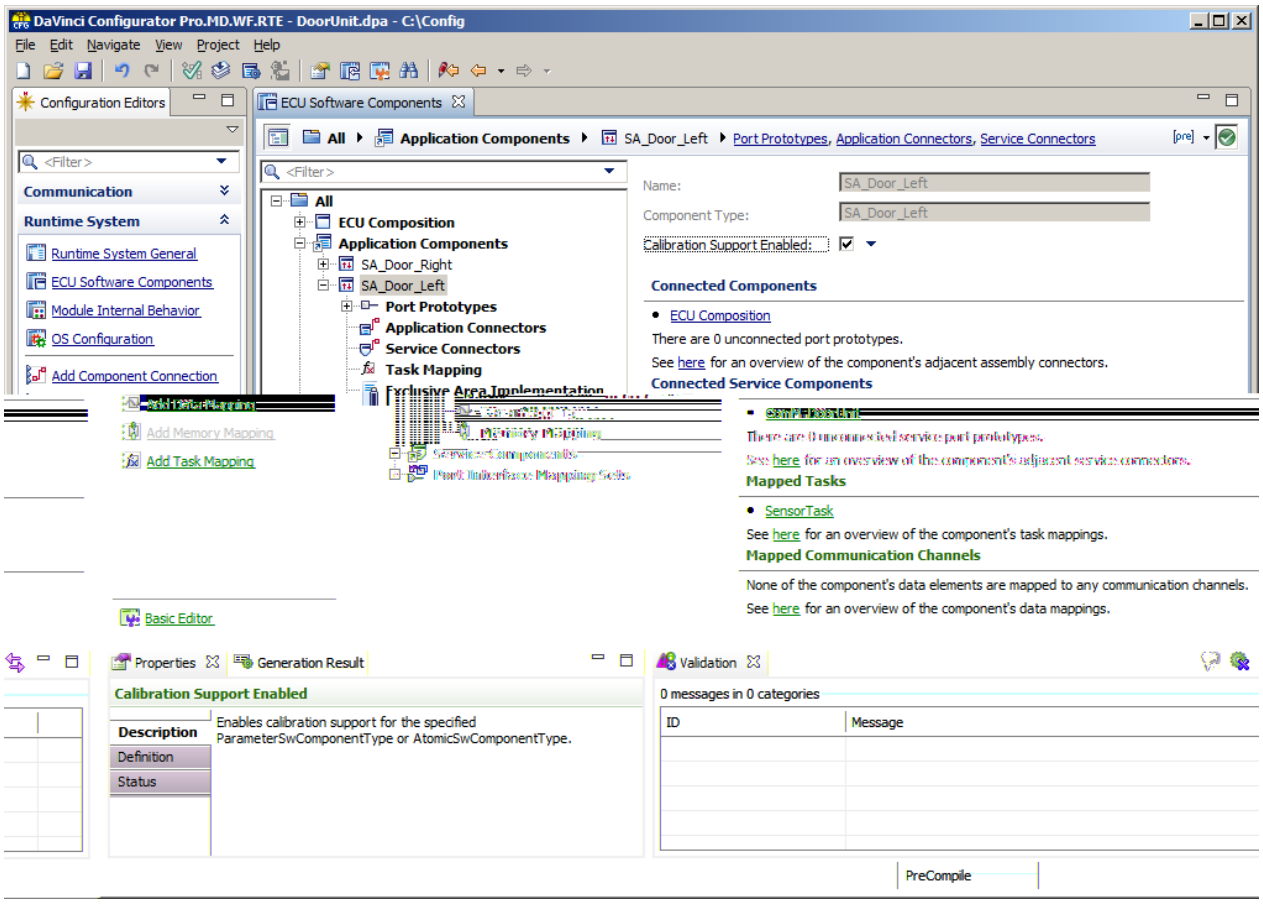
XCP_events.a2l

Rte_CData

Rte_Prm

-
-
-
-

Calibration Support



-

Rte_MemSeg.a2l

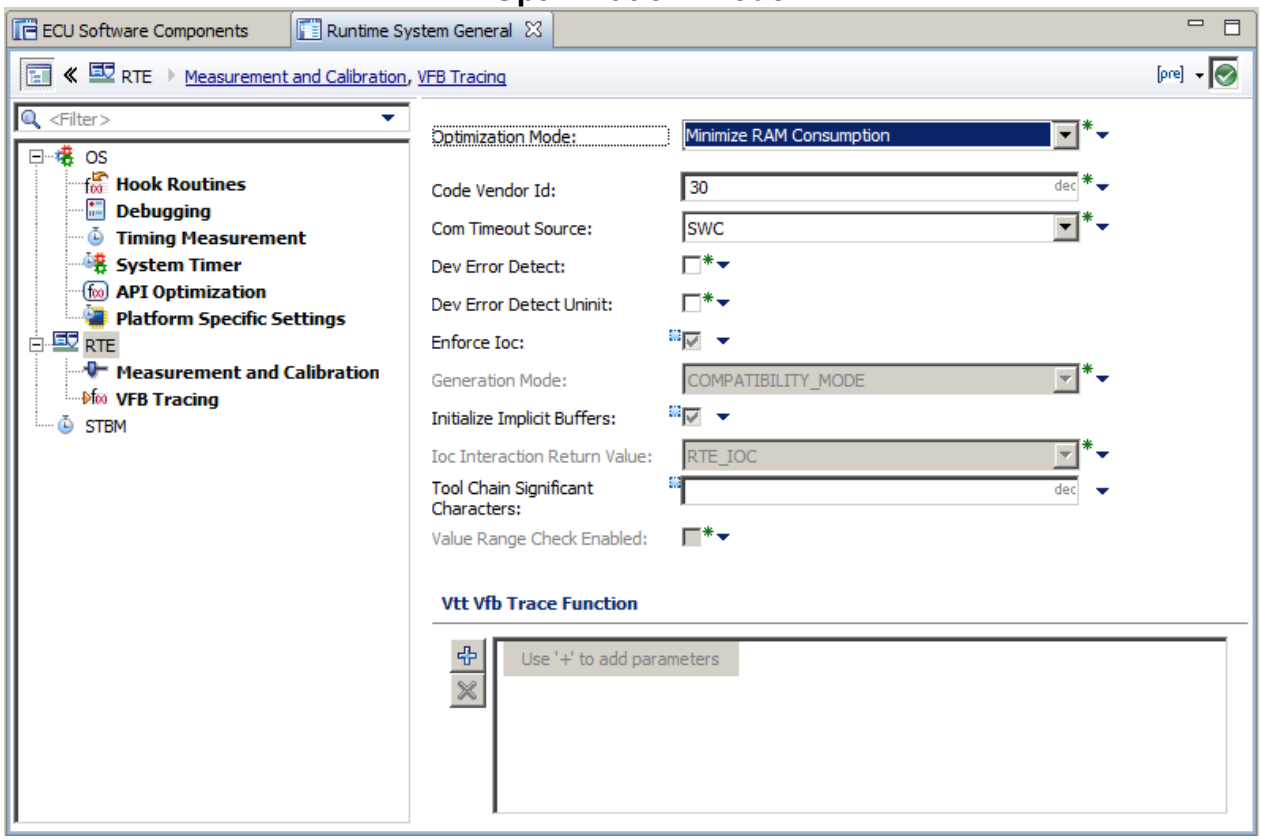
Rte_MemSeg.a2l

CalibrationBufferSize

6.7 Optimization Mode Configuration

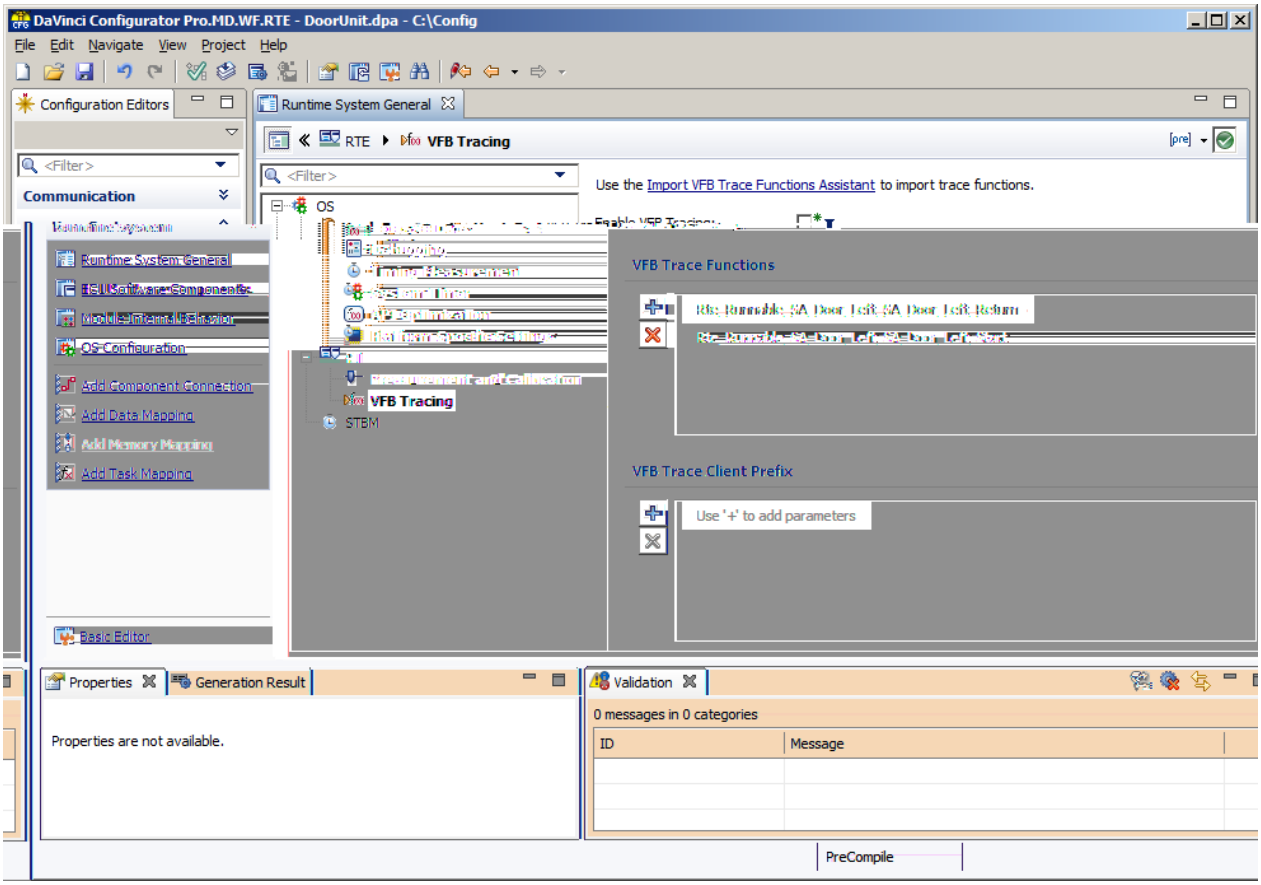
- MEMORY
- RUNTIME

Optimization Mode



-

6.8 VFB Tracing Configuration



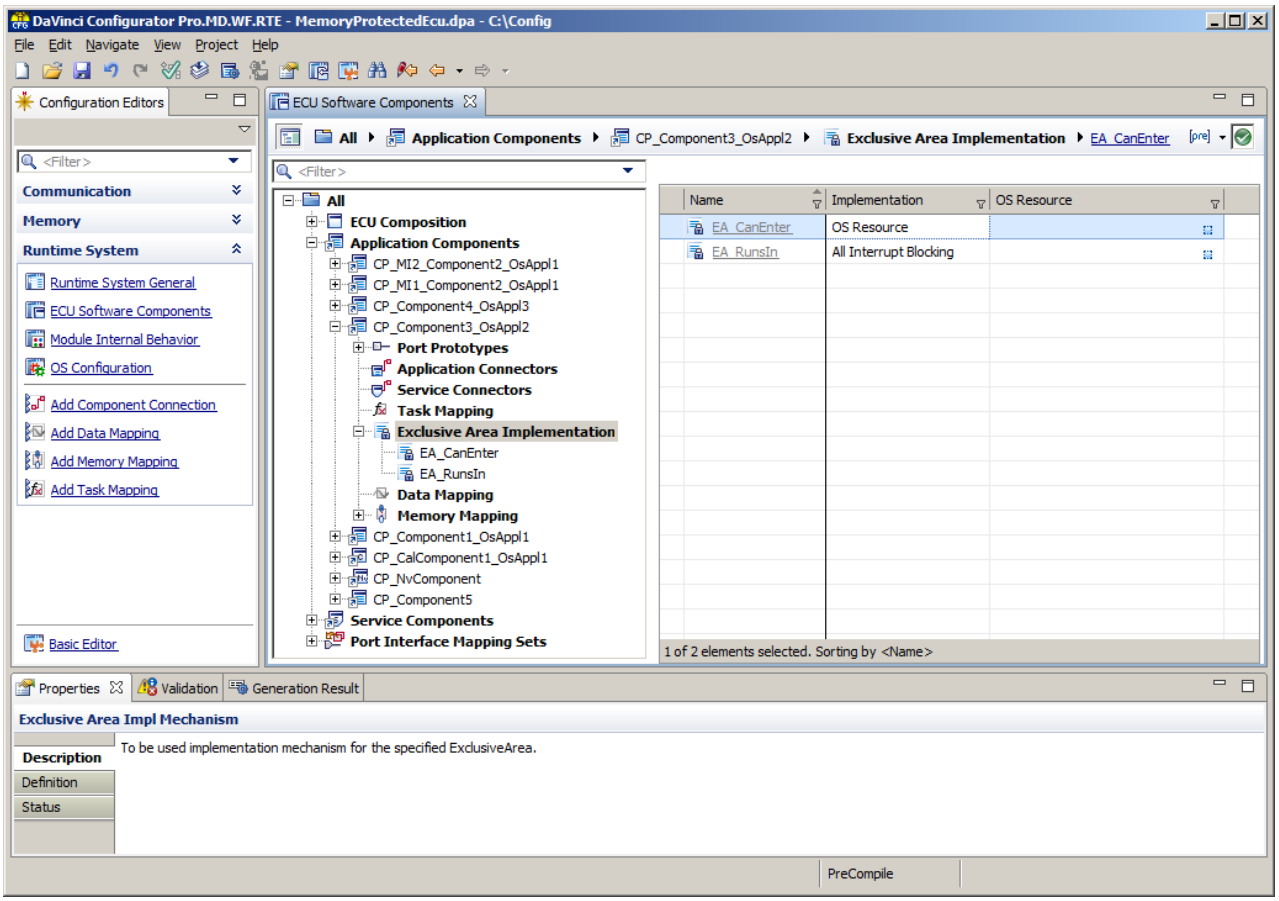
-

`Rte_Hook.h`

`Rte_Hook.h`.

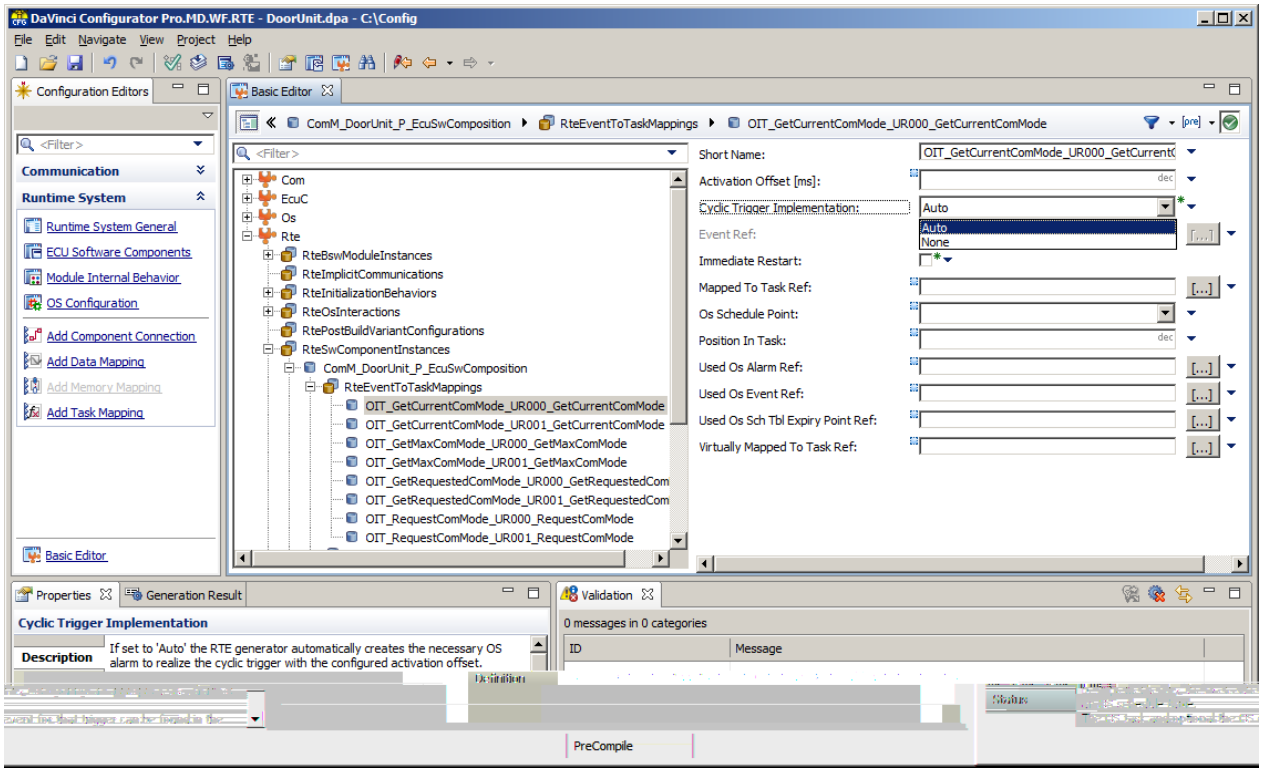
Info

6.9 Exclusive Area Implementation



-

6.10 Periodic Trigger Implementation



-

Caution

`Auto` `None`

`Auto`

`None`

Rte.html

5 Task List

Task	Type	Schedule	Priority
T1	Extended	NON	1
T2	Basic	NON	2

Back

6 Trigger List

Trigger	Runnable	Task	OS Event	OS Alarm
TimingEvent Cyclic 2ms	Runnable1	T1	Rte_Ev_Run1_c_Runnable1	
TimingEvent Cyclic 2ms	Runnable2	T2	n/a	
TimingEvent Cyclic 5ms	RunnableCyclic	T1	Rte_Ev_Run_c_RunnableCyclic	Rte_Al_TE_c_RunnableCyclic
TimingEvent Cyclic 5ms	Runnable3	T1	Rte_Ev_Run1_c_Runnable3	

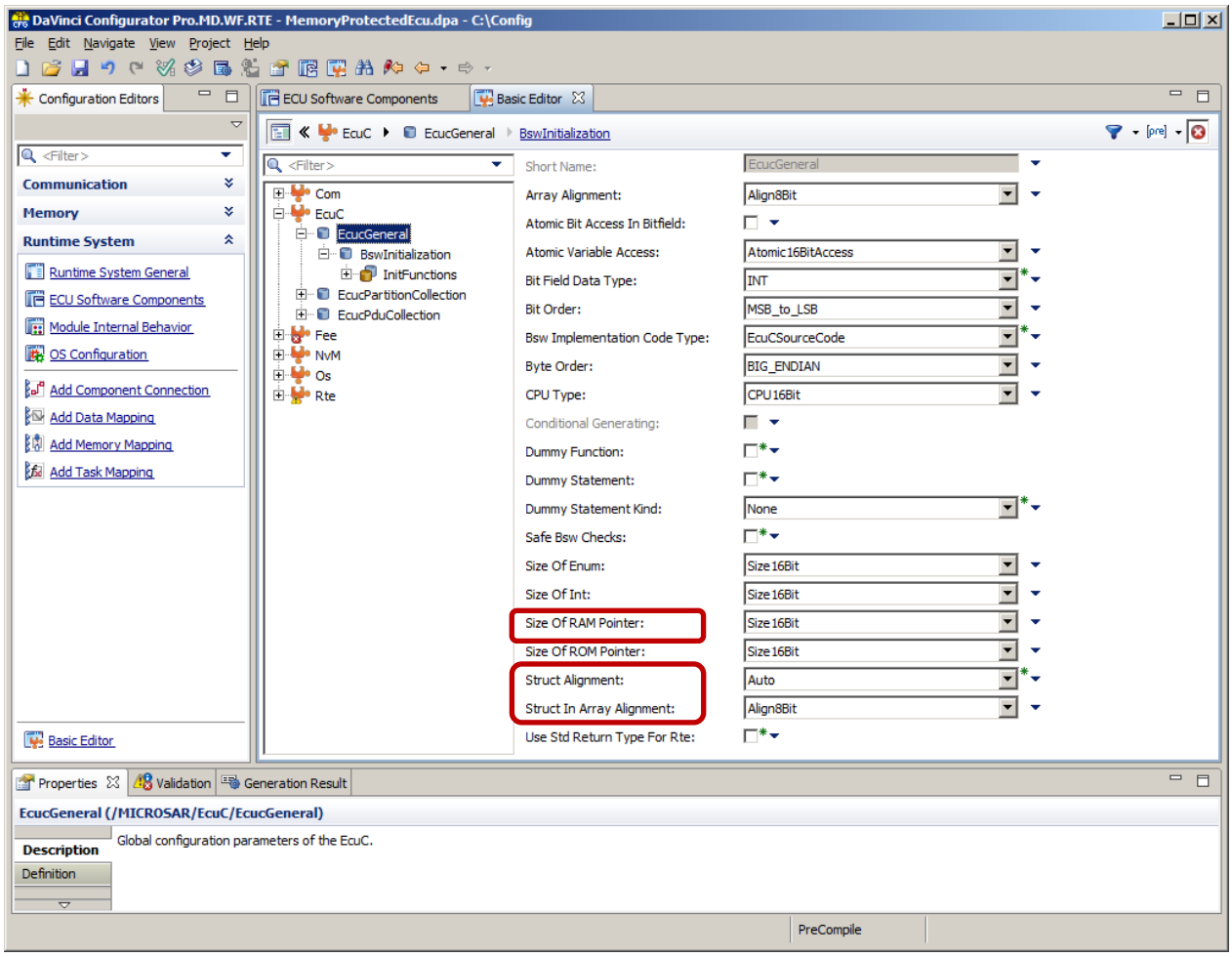
-

SETEVENT
ACTIVATETASK

Caution

6.11 Resource Calculation

- Size Of RAM Pointer
- TaskType
- EventMaskType
- Struct Alignment Struct In Array Alignment
- boolean PlatformTypes.h



-

7 Glossary and Abbreviations

7.1 Glossary

Term	Description

-

7.2 Abbreviations

Abbreviation	Description
	-
	- -
	- -
	-
	-
	-

	-
	-

-

8 Additional Copyrights

Free and Open Source Software

9 Contact

-
-
-
-
-
-

www.vector.com