VECTOR

Stream Registration Protocol

Technical Reference

AVB-Stack

Version 3.0.0

Authors	Michael Seidenspinner
Status	Released

Document Information

History

Author	Date	Version	Remarks
Michael Seidenspinner	2015-06-02	1.0.0	Creation of document
Michael Seidenspinner	2015-07-03	1.1.0	ESCAN00083774
Michael Seidenspinner	2016-12-14	2.0.0	Update to new CI-Layout
Michael Seidenspinner	2017-01-13	3.0.0	Added dynamic Srp

Reference Documents

No.	Source	Title	Version
[1]	IEEE	IEEE Std 802.1Q Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks	2011
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	4.2.1
[3]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	4.2.1
[4]	AUTOSAR	AUTOSAR_SWS_EthernetInterface.pdf	4.2.1
[5]	Vector	TechnicalReference_EthIf.pdf	3.6.0
[6]	Vector	TechnicalReference_EthRh850.pdf	2.2.0

Scope of the Document

This technical reference describes the general use of the Srp basis software. The Srp module can only be used in conjunction with the EthIf (see [5]) and Eth (see [6], the use of the Srp is not restricted to the referenced RH850 Platform. This Technical Reference is just used as an example) basis software module which are also part of the delivery.

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
1.00.xx	Initial implementation according to IEEE802.1Q
2.00.xx	Update to R14 (Cfg5 Breaking Change)
3.00.xx	Update to R17
3.01.xx	Dynamic Srp support

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module Srp as specified in [1].

Supported AUTOSAR Release*:	4.2.1	
Supported Configuration Variants:	Pre-compile	
Vendor ID:	SRP_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	SRP_MODULE_ID	255 decimal

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The Srp module is used to propagate attributes throughout the Network. Therefore it utilizes the three MRP Applications:

> MSRP
> MVRP
> MMRP (optional)

Using MSRP the Srp module can reserve bandwidth for Streams (see [1] IEEE Std 802.1Q Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks).

The Srp module offers the functionality to

> Declare and register membership of Multi-cast MAC-Address (MMRP)
> Declare and register membership of VLAN (MVRP)
> Declare and register streams (MSRP)

2.1 Architecture Overview

The following figure shows where the Srp is located in the AUTOSAR architecture.

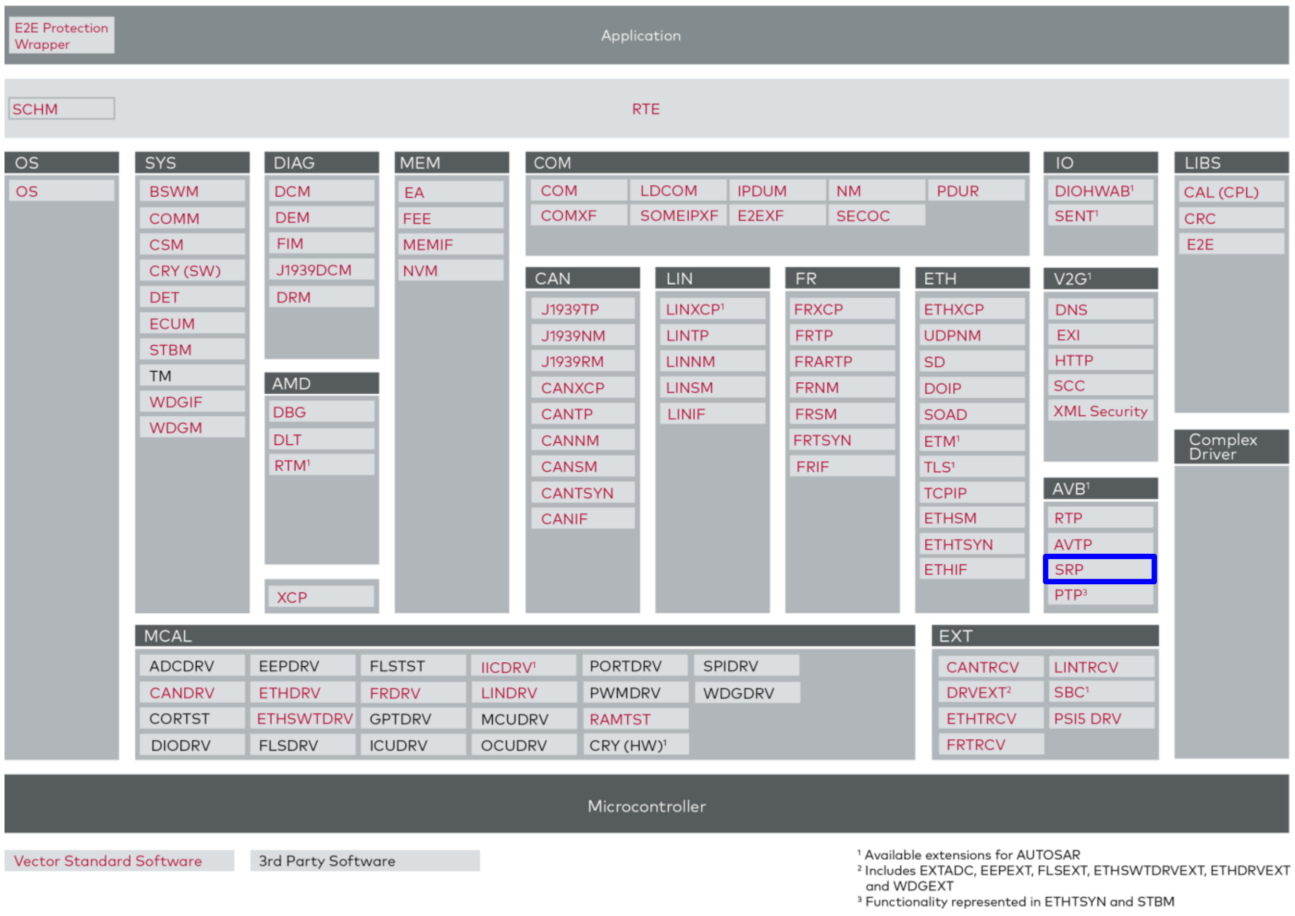


Figure 2-1 AUTOSAR 4.2 Architecture Overview

The next figure shows the interfaces to adjacent modules of the Srp. These interfaces are described in chapter 5.

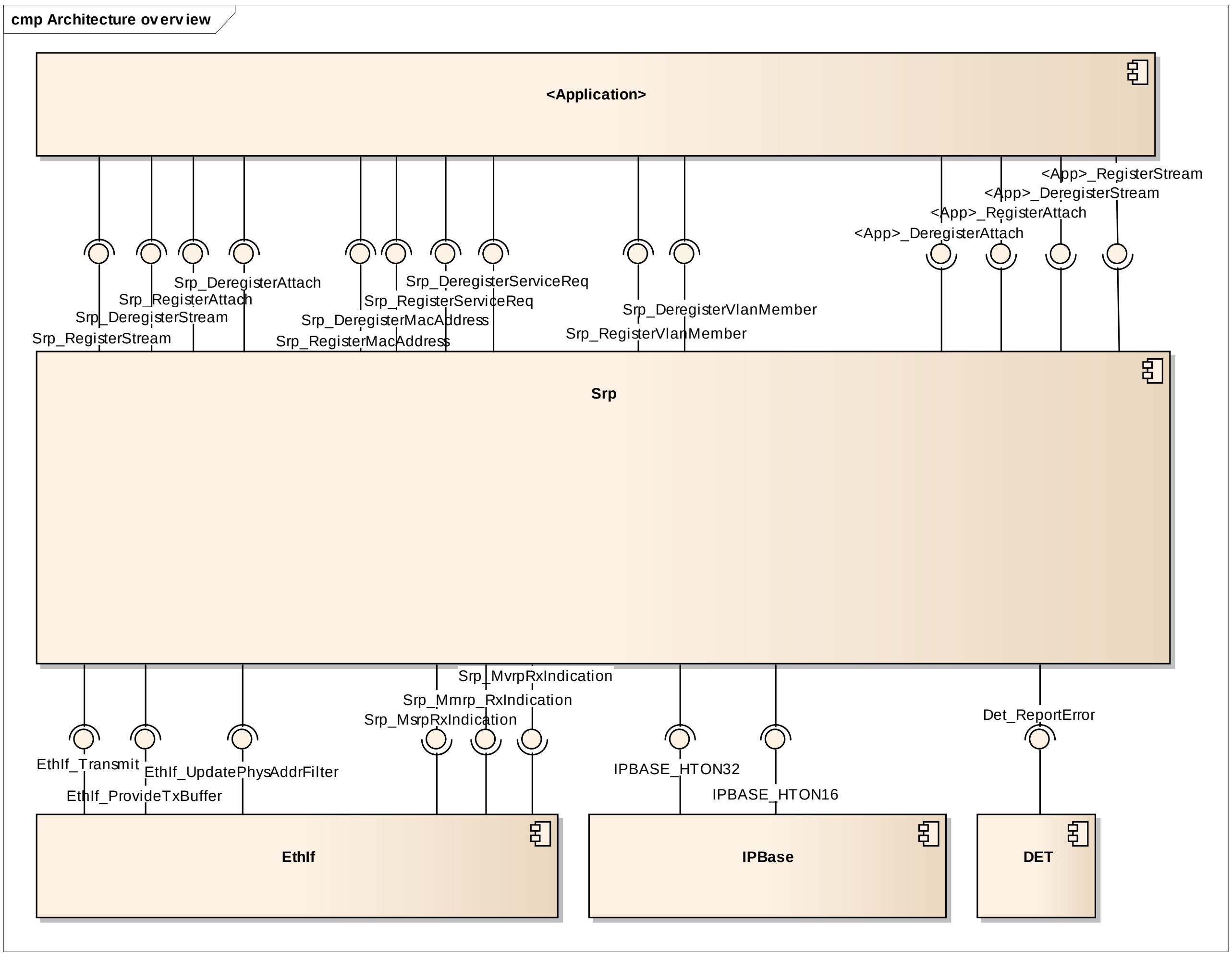


Figure 2-2 Interfaces to adjacent modules of the Srp

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the Srp.

The standard functionality according to IEEE802.1Q is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported IEEE802.1Q features

> Table 3-2 Not supported IEEE802.1Q features

The following features specified in [1] are supported:

Supported IEEE802.1Q Features
Multiple MAC Registration Protocol
Multiple VLAN Registration Protocol
Multiple Stream Registration Protocol
Full Participant

Table 3-1 Supported IEEE802.1Q features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported IEEE802.1Q Features
Dynamical update of the VID filter database (MVRP)
Service Requirement 'All Groups' and 'All Unregistered Groups' (MMRP)
Bridge behavior: MRP Attribute Propagation (MAP)
Applicant-Only Participant

Table 3-2 Not supported IEEE802.1Q features

3.1.2 General Limitations

The following general limitations apply to the Srp.

3.1.2.1 Supported MSRP Domains

Only one MSRP Domain is supported per Srp Port.

3.1.2.2 Bandwidth reservation

For a Listener, it is only checked if the bandwidth required for every received stream managed by the Srp exceeds the total bandwidth supported by the transceiver.

A real bandwidth check is only performed on Talker side (see also 3.6).

Note
The Bandwidth is reserved within the call of the `Srp_RegisterStream` API if enough bandwidth is available. In the current implementation no check if a Listener is available for the Stream is performed before the bandwidth reservation.

3.1.2.3 MVRP VLAN membership declaration

The dynamic declaring and withdrawing of membership to a VLAN is not supported. When the Application is declaring membership to a VLAN not supported by the Ethernet Controller, the corresponding Srp API will return E_NOT_OK. However the MVRP messages will be sent and processed by Srp as described in [1] IEEE Std 802.1Q Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks.

3.1.2.4 MMRP Service Requirement

Generally the MMRP Service Requirement functionality is supported by Srp. But declaring membership to ‘All Groups’ or ‘All Unregistered Groups’ will not affect the physical address filter.

3.2 Initialization

The Srp is initialized by calling the `Srp_InitMemory()` service followed by `Srp_Init()`.

3.3 States

The Srp is operational after initialization.

3.4 Main Functions

The Srp has a `Srp_MainFunction()` that handles cyclic tasks like timers and processing of state machines needed for Srp operation. Table 3-3 shows the state machines that are processed in the `Srp_MainFunction()`.

State machine	Task
`Srp_ProcessSmApplicant()`	Declaration of attribute
`Srp_ProcessSmRegistrar()`	Registration of attributes
`Srp_ProcessSmLeaveAll()`	Generates cyclic deregistration of attributes
`Srp_ProcessSmPeriodicTransmission()`	Generates Periodic Transmission events

Table 3-3 List of processed state machines in Srp_MainFunction()

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `Srp_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported Srp ID is 255.

The reported service IDs identify the services which are described in Table 3-5. The following table presents the service IDs and the related services:

Service ID	Service
`0x01`	`Srp_MainFunction()`
`0x02`	`Srp_GetVersionInfo()`
`0x03`	`Srp_Init()`
`0x04`	`Srp_InitPort()`
`0x05`	`Srp_InitSmLeaveAll()`
`0x06`	`Srp_InitSmPeriodicTransmission()`
`0x07`	`Srp_InitSmApplicant()`
`0x08`	`Srp_InitSmRegistrar()`
`0x10`	`Srp_RegisterStream()`
`0x11`	`Srp_DeregisterStream()`
`0x12`	`Srp_RegisterAttach()`
`0x13`	`Srp_DeregisterAttach()`
`0x14`	`Srp_RegisterMacAddress()`
`0x15`	`Srp_DeregisterMacAddress()`
`0x16`	`Srp_RegisterServiceRequirement()`
`0x17`	`Srp_DeregisterServiceRequirement()`
`0x18`	`Srp_RegisterVlanMember()`
`0x19`	`Srp_DeregisterVlanMember()`
`0x20`	`Srp_Msrp_RxIndication()`
`0x21`	`Srp_Mvrp_RxIndication()`
`0x22`	`Srp_Mmrp_RxIndication()`
`0x23`	`Srp_TxConfirmation()`
`0x24`	`Srp_Cbk_TrcvLinkStateChg()`
`0x30`	`Srp_ProcessSmApplicant()`
`0x31`	`Srp_ProcessSmRegistrar()`
`0x32`	`Srp_ProcessSmLeaveAll()`
`0x33`	`Srp_ProcessSmPeriodicTransmission()`

Service ID	Service
0x40	Srp_Transmit()
0x41	Srp_AssembleMsg()
0x42	Srp_AssembleMsgInOrMt()
0x43	Srp_AssembleMsgJoin()
0x44	Srp_RequestTransmitOpportunity()
0x50	Srp_ProcessAttributeMac()
0x51	Srp_ProcessAttributeServiceReq()
0x52	Srp_ProcessAttributeVid()
0x53	Srp_ProcessAttributeTalker()
0x54	Srp_ProcessAttributeListener()
0x55	Srp_ProcessAttributeDomain()
0x60	Srp_MrpHasAtLeastOneValidAttribute()

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

Error Code	Description
0x01	SRP_E_UNINIT
0x02	SRP_E_ALREADY_INITIALIZED
0x03	SRP_E_NULL_POINTER
0x04	SRP_E_INV_PARAM
0x05	SRP_E_INV_STATE_MACHINE
0x06	SRP_E_TX_FAILED
0x07	SRP_E_INV_MRP_APPLICATION
0x08	SRP_E_INV_EVENT
0x09	SRP_E_MMRP_NOT_SUPPORTED
0x0A	SRP_E_INV_MSG_LENGTH
0x0B	SRP_E_INV_BUF_IDX
0x0C	SRP_E_INV_BUF_PTR
0x0D	SRP_E_NO_TX_BUFFER

Table 3-5 Errors reported to DET

3.5.2 Production Code Error Reporting

No production error code reporting to the DEM (see [3]) is supported.

Error Code	Description
None	

Table 3-6 Errors reported to DEM

3.6 Static and dynamic Srp

The Srp module supports two different ways to manage the bandwidth on transmission side. Table 3-7 shows the difference between both methods. The used type can be chosen by the configuration parameter `/MICROSAR/Srp/SrpGeneral/SrpType`.

Value of `/MICROSAR/Srp/SrpGeneral/SrpType`	**Mode description**
STATIC	The bandwidth is managed by the Srp only. It is possible to decide a maximum allowed bandwidth by configuration of the Eth. The Srp is calculating and holding the required bandwidth for each relevant stream and is checking the bandwidth. In this mode no traffic other than the streams managed by the Srp are supported on the corresponding Queue.
DYNAMIC	The bandwidth is managed by Eth. The Srp dynamically reserves and releases bandwidth required for the stream managed by the Srp.

Table 3-7 Srp Type comparison

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
`Srp.h`	API declaration
`Srp.c`	Implementation of the Srp functionality
`Srp_Cbk.h`	API Callback declaration
`Srp_Types.h`	Type definitions for the Srp module
`Srp_Priv.h`	Internal (private) API declaration

Table 4-1 Static files

4.1.2 Dynamic Files

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

File Name	Description
`Srp_Cfg.h`	Pre-compile time parameter configuration
`Srp_Lcfg.h`	Link-time parameter configuration declaration
`Srp_Lcfg.c`	Link-time parameter configuration

Table 4-2 Generated files

4.2 Critical Sections

To ensure data consistency and a correct function of the Srp module the exclusive area SRP_EXCLUSIVE_AREA_0 has to be provided during the integration.

Considering the timing behavior of your system (e.g. depending on the CPU load of your system, priorities and interruptibility of interrupts and OS tasks and their jitter and delay times) the integrator has to choose and configure a critical section solution in such was that it is ensured that the API functions do not interrupt each other.

It is recommended to use the functions SuspendAllInterrupts() and ResumeAllInterrupts() for SRP_EXCLUSIVE_AREA_0 to ensure data consistency.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the Srp are described in this chapter.

Type Name	C-Type	Description	Value Range
Srp_MsrpAttributeType	uint8	Type of MSRP Attribute	SRP_MSRP_TALKER_ADVERTISE Talker Advertise
			SRP_MSRP_TALKER_FAILED Talker Failed
			SRP_MSRP_LISTENER Listener
			SRP_MSRP_DOMAIN Domain
Srp_PortIdxType	uint8	Srp Port Index	0 – 255
Srp_PriorityType	uint8	The Data Frame Priority	0 – 7
Srp_RankType	uint8	The Rank of a Stream	SRP_RANK_EMERGENCY Emergency Rank
			SRP_RANK_NON_EMERGENCY Non-Emergency Rank
Srp_AccumulatedLatencyType	uint32	Worst-case latency that a Stream can encounter in its path from the Talker to a given Listener	0 – 4.294.967.295
Srp_VLanIdType	uint16	VLAN Identifier	1 – 4094
Srp_MacAddressType	uint8[6]	6 Byte MAC-Address	00:00:00:00:00:00 – FF:FF:FF:FF:FF:FF
Srp_MmrpServiceRequirementAttributeType	uint8	The Service Requirement Attribute	SRP_MMRP_ALL_GROUPS All groups
			SRP_MMRP_ALL_UNREGISTERED_GROUPS All unregistered groups
Srp_MrpApplicationType	uint8	Specifying the MRP Application	SRP_MMRP_APPLICATION
			SRP_MVRP_APPLICATION
			SRP_MSRP_APPLICATION
Srp_MsrpAttributeType	uint8	Specifying the MSRP Attribute Type	SRP_MSRP_TALKER_ADVERTISE
			SRP_MSRP_TALKER_FAILED
			SRP_MSRP_LISTENER

Type Name	C-Type	Description	Value Range
			SRP_MSRP_DOMAIN

Table 5-1 Type definitions

Srp_StreamIdType

This structure contains the information clearly identifying a Stream.

Struct Element Name	C-Type	Description	Value Range
SourceAddress	uint8[6]	Array containing the 6 Byte MAC-Address of the Participant sourcing the Stream	00:00:00:00:00:00 – FF:FF:FF:FF:FF:FF
UniqueId	uint16	Unique identifier to differentiate between several Streams sourced by the same Participant	0 – 65535

Table 5-2 Srp_StreamIdType

Srp_DataFrameParametersType

This structure contains the DataFrameParameters of a Stream

Struct Element Name	C-Type	Description	Value Range
DestinationAddress	uint8[6]	Array containing the 6 Byte destination MAC-Address	00:00:00:00:00:00 – FF:FF:FF:FF:FF:FF
VLanIdentifier	uint16	The ID of the VLAN the Stream is sourced within	1 – 4094

Table 5-3 Srp_DataFrameParametersType

Srp_TSpecType

This structure contains the TSpec of a Stream

Struct Element Name	C-Type	Description	Value Range
MaxFrameSize	uint16	Maximum frame size that the Talker will produce, excluding any overhead for media specific framing	0 – 65535
MaxIntervalFrames	uint16	Maximum number of frames the Talker may transmit in one “class measurement interval”	0 – 65535

Table 5-4 Srp_TSpecType

Srp_FailureInformationType

This structure contains Failure Information

Struct Element Name	C-Type	Description	Value Range
BridgeID	uint64	The ID of the Bridge the failure occurred	`0 – (2^64)-1`
ReservationFailureCode	uint8	Error Code	`SRP_FAILURE_CODE_UNKNOWN`
			`SRP_FAILURE_CODE_INSUFFICIENT_BANDWITH`
			`SRP_FAILURE_CODE_INSUFFICIENT_BRIDGE_RESOURCES`
			`SRP_FAILURE_CODE_INSUFFICIENT_BANDWITH_FOR_TRAFFIC_CLASS`
			`SRP_FAILURE_CODE_STREAM_ID_IN_USE_BY_ANOTHER_TALKER`
			`SRP_FAILURE_CODE_STREAM_DESTINATION_ADDRESS_ALREADY_IN_USE`
			`SRP_FAILURE_CODE_STREAM_PREEMPTED_BY_HIGHER_RANK`
			`SRP_FAILURE_CODE_REPORTED_LATENCY_HAS_CHANGED`
			`SRP_FAILURE_CODE_EGRESS_PORT_IS_NOT_AVB_CAPABLE`
			`SRP_FAILURE_CODE_USE_A_DIFFERENT_DESTINATION_ADDRESS`
			`SRP_FAILURE_CODE_OUT_OF_MSRP_RESOURCES`
			`SRP_FAILURE_CODE_OUT_OF_MMRP_RESOURCES`
			`SRP_FAILURE_CODE_CANNOT_STORE_DESTINATION_ADDRESS`
			`SRP_FAILURE_CODE_REQUESTED_PRIORITY_IS_NOT_AN_SR_CLASS`
			`SRP_FAILURE_CODE_MAX_FRAME_SIZE_IS_TOO_LARGE_FOR_MEDIA`
			`SRP_FAILURE_CODE_MSRP_MAX_FAN_IN_PORTS_LIMIT_HAS_BEEN_REACHED`
			`SRP_FAILURE_CODE_CHANGES_IN_FIRST_VALUE_FOR_A_REGISTERED_STREAM_ID`

Struct Element Name	C-Type	Description	Value Range
			SRP_FAILURE_CODE_VLAN_IS_BLOCKED_ON_THIS_EGRESS_PORT
			SRP_FAILURE_CODE_VLAN_TAGGING_IS_DIABLED_ON_THIS_EGRESS_PORT
			SRP_FAILURE_CODE_SR_CLASS_PRIORITY_MISMATCH

Table 5-5 Srp_FailureInformationType

Srp_MsrpRegisterStreamInfoType

This structure contains the necessary information to declare/register a Stream

Struct Element Name	C-Type	Description	Value Range
StreamId	uint8[6]	SourceAddress	see Table 5-2
	uint16	UniqueId	
DataFrameParameters	uint8[6]	DestinationAddress	see Table 5-3
	uint16	VLanIdentifier	
TSpec	uint16	MaxFrameSize	see Table 5-4
	uint16	MaxIntervalFrames	
Priority	uint8	Data Frame Priority	0 – 7
Rank	uint8	Ranke Type	SRP_RANK_EMERGENCY
			SRP_RANK_NON_EMERGENCY
AccumulatedLatency	uint32	Worst-case latency that a Stream can encounter in its path from the Talker to a given Listener	0 – 4.294.967.295

Table 5-6 Srp_MsrpRegisterStreamInfoType

Srp_MsrpStreamInfoType

This structure contains all information about a Stream

Struct Element Name	C-Type	Description	Value Range
StreamId	uint8[6]	SourceAddress	see Table 5-2
	uint16	UniqueId	
DataFrameParameters	uint8[6]	DestinationAddress	see Table 5-3
	uint16	VLanIdentifier	
TSpec	uint16	MaxFrameSize	see Table 5-4
	uint16	MaxIntervalFrames	
Priority	uint8	Data Frame Priority	0 – 7
Rank	uint8	Ranke Type	SRP_RANK_EMERGENCY

Struct Element Name	C-Type	Description	Value Range
			SRP_RANK_NON_EMERGENCY
AccumulatedLatency	uint32	Worst-case latency that a Stream can encounter in its path from the Talker to a given Listener	0 - 4.294.967.295
FailureInformation	uint64	BridgeID	see Table 5-5
	uint8	ReservationFailureCode	

Table 5-7 Srp_ MsrpStreamInfoType

Srp_MsrpAttributeInfoType

This structure contains all information necessary within the indication callback notification functions

Struct Element Name	C-Type	Description	Value Range
AttributeType	uint8	Specifying the MSRP Attribute Type	SRP_MSRP_TALKER_ADVERTISE Talker Advertise
			SRP_MSRP_TALKER_FAILED Talker Failed
			SRP_MSRP_LISTENER Listener
			SRP_MSRP_DOMAIN Domain
PortIdx	uint8	The Srp Port Index	0 - 255
StreamInfoPtr	uint8[6]	SourceAddress	see Table 5-7
	uint16	UniqueId	
	uint8[6]	DestinationAddress	
	uint16	VLanIdentifier	
	uint16	MaxFrameSize	
	uint16	MaxIntervalFrames	
	uint8	Priority	
	uint8	Rank	
	uint32	AccumulatedLatency	
	uint64	BridgeID	
	uint8	ReservationFailureCode	

Table 5-8 Srp_MsrpAttributeInfoType

5.2 Services provided by Srp

5.2.1 Srp_InitMemory

Prototype	
`void Srp_InitMemory (void)`	
Parameter	
none	
Return code	
void	none
Functional Description	
Function for *_INIT_*-variable initialization. Service to initialize module global variables at power up. This function can be used to initialize the variables in *_INIT_* sections in case they are not initialized by the startup code.	
Particularities and Limitations	
This function must be called before using the module > Module must not be initialized Function shall be called from task level	
Call context	

Table 5-9 Srp_InitMemory

5.2.2 Srp_Init

Prototype	
`void Srp_Init (SRP_P2CONSTCFG(Srp_ConfigType) ConfigPtr)`	
Parameter	
ConfigPtr [in]	Configuration structure for initializing the module
Return code	
void	none
Functional Description	
Initialization function. This function initializes the module Srp. It initializes all variables and sets the module state to initialized.	
Particularities and Limitations	
Specification of module initialization > Interrupts have to be disabled. The module has to be uninitialized. Srp_InitMemory has been called unless Srp_State is initialized by start-up code.	
Call context	

Table 5-10 Srp_Init

5.2.3 Srp_GetVersionInfo

Prototype	
`void **Srp_GetVersionInfo** (SRP_P2VAR(Std_VersionInfoType) Versioninfo)`	
Parameter	
Versioninfo [out]	Pointer to where to store the version information
Return code	
void	none
Functional Description	
Returns the version information. Srp_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component.	
Particularities and Limitations	
> Input parameter must not be NULL. Function shall be called from task level	
Call context	

Table 5-11 Srp_GetVersionInfo

5.2.4 Srp_RegisterStream

Prototype	
`Std_ReturnType **Srp_RegisterStream** (Srp_PortIdxType PortIdx, SRP_P2CONST(Srp_MsrpRegisterStreamInfoType) StreamInfoPtr)`	
Parameter	
PortIdx [in]	Port Index
StreamInfoPtr [in]	Pointer to a structure containing the following Information of the offered Stream: StreamID The unique identifier of the offered Stream DataFrameParameters Identifying all frames belonging to the same Stream TSpec The Traffic Specification for the Stream containing the MaxFrameSize and the MaxIntervalFrames DataFramePriority The Priority of the Stream Rank Marking emergency data AccumulatedLatency Marking worst-case latency in its path from Talker to Listener
Return code	
Std_ReturnType	E_OK Successfully registered new Stream E_NOT_OK Failed to register new Stream
Functional Description	
Allows the Application to offer a new Stream.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-12 Srp_RegisterStream

5.2.5 Srp_DeregisterStream

Prototype	
`void **Srp_DeregisterStream** (Srp_PortIdxType PortIdx, Srp_StreamIdType StreamID)`	
Parameter	
PortIdx [in]	Port Index
StreamID [in]	The unique identifier of the Stream
Return code	
void	none
Functional Description	
Allows the Application to withdraw the offer of a Stream.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-13 Srp_DeregisterStream

5.2.6 Srp_RegisterAttach

Prototype	
`Std_ReturnType **Srp_RegisterAttach** (Srp_PortIdxType PortIdx, Srp_StreamIdType StreamID)`	
Parameter	
PortIdx [in]	Port Index
StreamID [in]	The unique identifier of the Stream
Return code	
Std_ReturnType	E_OK Successfully registered the offered Stream E_NOT_OK Failed to register the offered Stream
Functional Description	
Allows the Application to register a offered Stream.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-14 Srp_RegisterAttach

5.2.7 Srp_DeregisterAttach

Prototype	
`void **Srp_DeregisterAttach** (Srp_PortIdxType PortIdx, Srp_StreamIdType StreamID)`	
Parameter	
StreamID [in]	The unique identifier of the Stream
PortIdx [in]	Port Index
Return code	
void	none
Functional Description	
Allows the Application to withdraw the registration of a Stream.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-15 Srp_DeregisterAttach

5.2.8 Srp_RegisterVlanMember

Prototype	
`Std_ReturnType **Srp_RegisterVlanMember** (Srp_PortIdxType PortIdx, Srp_VLanIdType VID)`	
Parameter	
VID [in]	The unique identifier of the VLan
PortIdx [in]	Port Index
Return code	
Std_ReturnType	E_OK Successfully declared membership of the VLan E_NOT_OK Failed to declare membership of the VLan
Functional Description	
Allows the Application to register membership of the specified VLan.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-16 Srp_RegisterVlanMember

5.2.9 Srp_DeregisterVlanMember

Prototype	
`void` **`Srp_DeregisterVlanMember`** `(Srp_PortIdxType PortIdx, Srp_VLanIdType VID)`	
Parameter	
VID [in]	The unique identifier of the VLan
PortIdx [in]	Port Index
Return code	
void	none
Functional Description	
Allows the Application to withdraw the membership of the specified VLan.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-17 Srp_DeregisterVlanMember

5.2.10 Srp_RegisterMacAddress

Prototype	
`Std_ReturnType` **`Srp_RegisterMacAddress`** `(Srp_PortIdxType PortIdx, SRP_P2CONST(uint8) MacAddressPtr)`	
Parameter	
MacAddressPtr [in]	Pointer to the Multi-cast MacAddress
PortIdx [in]	Port Index
Return code	
Std_ReturnType	E_OK Successfully declared membership of the Multi-cast MAC Address E_NOT_OK Failed to declare membership of the Multi-cast MAC Address
Functional Description	
Allows the Application to register membership of the specified Multi-cast MAC Address.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-18 Srp_RegisterMacAddress

5.2.11 Srp_DeregisterMacAddress

Prototype	
`void` **`Srp_DeregisterMacAddress`** `(Srp_PortIdxType PortIdx, SRP_P2CONST(uint8) MacAddressPtr)`	
Parameter	
MacAddressPtr [in]	Pointer to the Multi-cast MacAddress
PortIdx [in]	Port Index
Return code	
void	none
Functional Description	
Allows the Application to withdraw membership of the specified Multi-cast MAC Address.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-19 Srp_DeregisterMacAddress

5.2.12 Srp_RegisterServiceRequirement

Prototype	
`Std_ReturnType` **`Srp_RegisterServiceRequirement`** `(Srp_PortIdxType PortIdx, Srp_MmrpServiceRequirementAttributeType ServiceRequirement)`	
Parameter	
PortIdx [in]	Port Index
ServiceRequirement [in]	The Service Requirement
Return code	
Std_ReturnType	E_OK Successfully registered the Service Requirement E_NOT_OK Failed to register the Service Requirement
Functional Description	
Allows the Application to register the specified Service Requirement.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-20 Srp_RegisterServiceRequirement

5.2.13 Srp_DeregisterServiceRequirement

Prototype	
`void` **`Srp_DeregisterServiceRequirement`** `(Srp_PortIdxType PortIdx, Srp_MmrpServiceRequirementAttributeType ServiceRequirement)`	
Parameter	
PortIdx [in]	Port Index
ServiceRequirement [in]	The Service Requirement
Return code	
void	none
Functional Description	
Allows the Application to withdraw registration of the specified Service Requirement.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-21 Srp_DeregisterServiceRequirement

5.2.14 Srp_SetPeriodic

Prototype	
`void` **`Srp_SetPeriodic`** `(Srp_PortIdxType PortIdx, Srp_MrpApplicationType MrpApplication, boolean PeriodicEnabled)`	
Parameter	
PortIdx [in]	Port Index
MrpApplication [in]	The MrpApplicationType: SRP_MMRP_APPLICATION: MMRP Application SRP_MVRP_APPLICATION: MVRP Application SRP_MSRP_APPLICATION: MSRP Application
PeriodicEnabled [in]	Enable/Disable PeriodicTransmission state machine TRUE: Enable PeriodicTransmission state machine FALSE: Disable PeriodicTransmission state machine
Return code	
void	none
Functional Description	
Allows the Application to enable or disable the PeriodicTransmission state machine.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-22 Srp_SetPeriodic

5.3 Services used by Srp

In the following table services provided by other components, which are used by the Srp are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
Det	Det_ReportError()
EthIf	EthIf_ProvideTxBuffer()
EthIf	EthIf_Transmit
EthIf	EthIf_UpdatePhysAddrFilter()

Table 5-23 Services used by the Srp

5.4 Callback Functions

This chapter describes the callback functions that are implemented by the Srp and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `Srp_Cbk.h` by the Srp.

5.4.1 Srp_Mmrp_RxIndication

Prototype	
`void` **`Srp_Mmrp_RxIndication`** `(uint8 VCtrlIdx, Eth_FrameType FrameType, boolean IsBroadcast, SRP_P2VAR(uint8) PhysAddrPtr, SRP_P2VAR(uint8) DataPtr, uint16 LenByte)`	
Parameter	
VCtrlIdx [in]	Index of the virtual controller that has received the frame.
FrameType [in]	Ethertype of the frame
IsBroadcast [in]	Determines that the frame was transmitted as broadcast
PhysAddrPtr [in]	Pointer to the physical address of the transmitted interface
DataPtr [in]	Pointer to the received data.
LenByte [in]	Byte count of the received frame.
Return code	
void	none
Functional Description	
Handles processing of received MMRP frames.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-24 Srp_Mmrp_RxIndication

5.4.2 Srp_Mvrp_RxIndication

Prototype	
`void Srp_Mvrp_RxIndication (uint8 VCtrlIdx, Eth_FrameType FrameType, boolean IsBroadcast, SRP_P2VAR(uint8) PhysAddrPtr, SRP_P2VAR(uint8) DataPtr, uint16 LenByte)`	
Parameter	
VCtrlIdx [in]	Index of the virtual controller that has received the frame.
FrameType [in]	Ethertype of the frame
IsBroadcast [in]	Determines that the frame was transmitted as broadcast
PhysAddrPtr [in]	Pointer to the physical address of the transmitted interface
DataPtr [in]	Pointer to the received data.
LenByte [in]	Byte count of the received frame.
Return code	
void	none
Functional Description	
Handles processing of received MVRP frames.	
Particularities and Limitations	
Call context	
> This function can be called in any context.	

Table 5-25 Srp_Mvrp_RxIndication

5.4.3 Srp_Msrp_RxIndication

Prototype	
`void Srp_Msrp_RxIndication (uint8 VCtrlIdx, Eth_FrameType FrameType, boolean IsBroadcast, SRP_P2VAR(uint8) PhysAddrPtr, SRP_P2VAR(uint8) DataPtr, uint16 LenByte)`	
Parameter	
VCtrlIdx [in]	Index of the virtual controller that has received the frame.
FrameType [in]	Ethertype of the frame
IsBroadcast [in]	Determines that the frame was transmitted as broadcast
PhysAddrPtr [in]	Pointer to the physical address of the transmitted interface
DataPtr [in]	Pointer to the received data.
LenByte [in]	Byte count of the received frame.
Return code	
void	none

Functional Description
Handles processing of received MSRP frames.
Particularities and Limitations
Call context
> This function can be called in any context.

Table 5-26 Srp_Msrp_RxIndication

5.4.4 Srp_Cbk_TrcvLinkStateChg

Prototype	
`void Srp_Cbk_TrcvLinkStateChg (uint8 CtrlIdx, Srp_LinkStateType TrcvLinkState)`	
Parameter	
CtrlIdx [in]	Index of the controller that changed its state
TrcvLinkState [in]	New link state of the transceiver
Return code	
void	none
Functional Description	
Callback function that notifies a changed state of the transceiver link.	
Particularities and Limitations	
Call context	
> This function can be called in task context.	

Table 5-27 Srp_Cbk_TrcvLinkStateChg

5.5 Configurable Interfaces

5.5.1 Notifications

At its configurable interfaces the Srp defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the Srp 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.

5.5.1.1 Register Stream Indication Callback

Prototype	
`void <Configurable_Cbk_Name>(const Srp_MsrpAttributeInfoType* AttributeInfoPtr)`	
Parameter	
`AttributeInfoPtr`	Pointer to a structure containing all information about the offered Stream.

Return code	
`void`	-
Functional Description	
This callback will be called when a new Stream is declared by another MSRP participant.	
Particularities and Limitations	
Call context	
> interrupt or task context	

Table 5-28 Register Stream Indication Callback

5.5.1.2 Deregister Stream Indication Callback

Prototype	
`void` **`<Configurable_Cbk_Name>`**`(const Srp_MsrpAttributeInfoType* AttributeInfoPtr)`	
Parameter	
`AttributeInfoPtr`	Pointer to a structure containing all information about the Stream
Return code	
`void`	-
Functional Description	
This callback will be called when the declaration of a Stream is withdrawn.	
Particularities and Limitations	
Call context	
> Interrupt or task context	

Table 5-29 Deregister Stream Indication Callback

5.5.1.3 Register Attach Indication Callback

Prototype	
`void` **`<Configurable_Cbk_Name>`**`(const Srp_MsrpAttributeInfoType* AttributeInfoPtr)`	
Parameter	
`AttributeInfoPtr`	Pointer to a structure containing all information about the Stream
Return code	
`void`	-
Functional Description	
This callback will be called when a Stream is registered by another MSRP participant.	

Particularities and Limitations
Call context
> Interrupt or task context

Table 5-30 Register Attach Indication Callback

5.5.1.4 Deregister Attach Indication Callback

Prototype	
`void <Configurable_Cbk_Name>(const Srp_MsrpAttributeInfoType* AttributeInfoPtr)`	
Parameter	
`AttributeInfoPtr`	Pointer to a structure containing all information about the Stream
Return code	
`void`	-
Functional Description	
This callback will be called when the registration of a Stream is withdrawn.	
Particularities and Limitations	
Call context	
> Interrupt or task context	

Table 5-31 Deregister Attach Indication Callback

6 Configuration

In the Srp the attributes can be configured with the tool DaVinci Configurator Pro.

6.1 Configuration Variants

The Srp supports the configuration variants

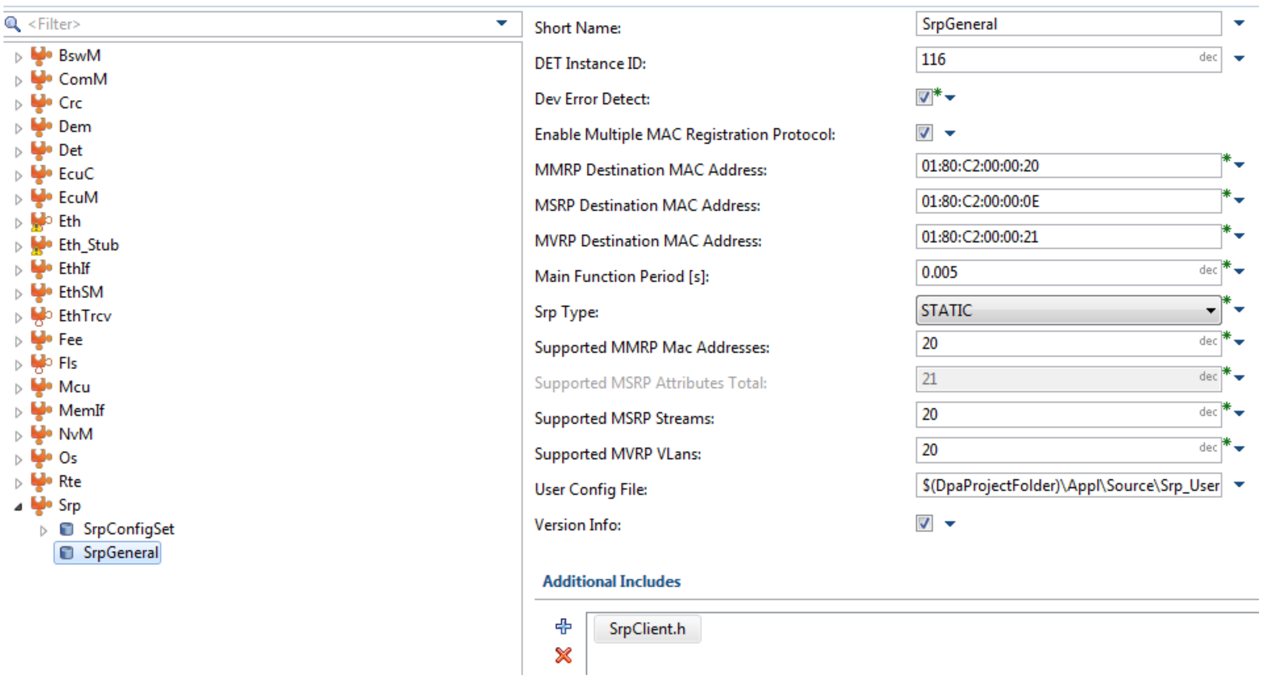
> `VARIANT-PRE-COMPILE`

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

6.2 Configuration with DaVinci Configurator Pro

6.2.1 SrpGeneral container

Figure 6-1 shows an overview of the SrpGeneral container in an existing project. The SrpGeneral container holds all common used configuration parameter. A detailed description of each configuration parameter can be found in the description view of the parameter properties.



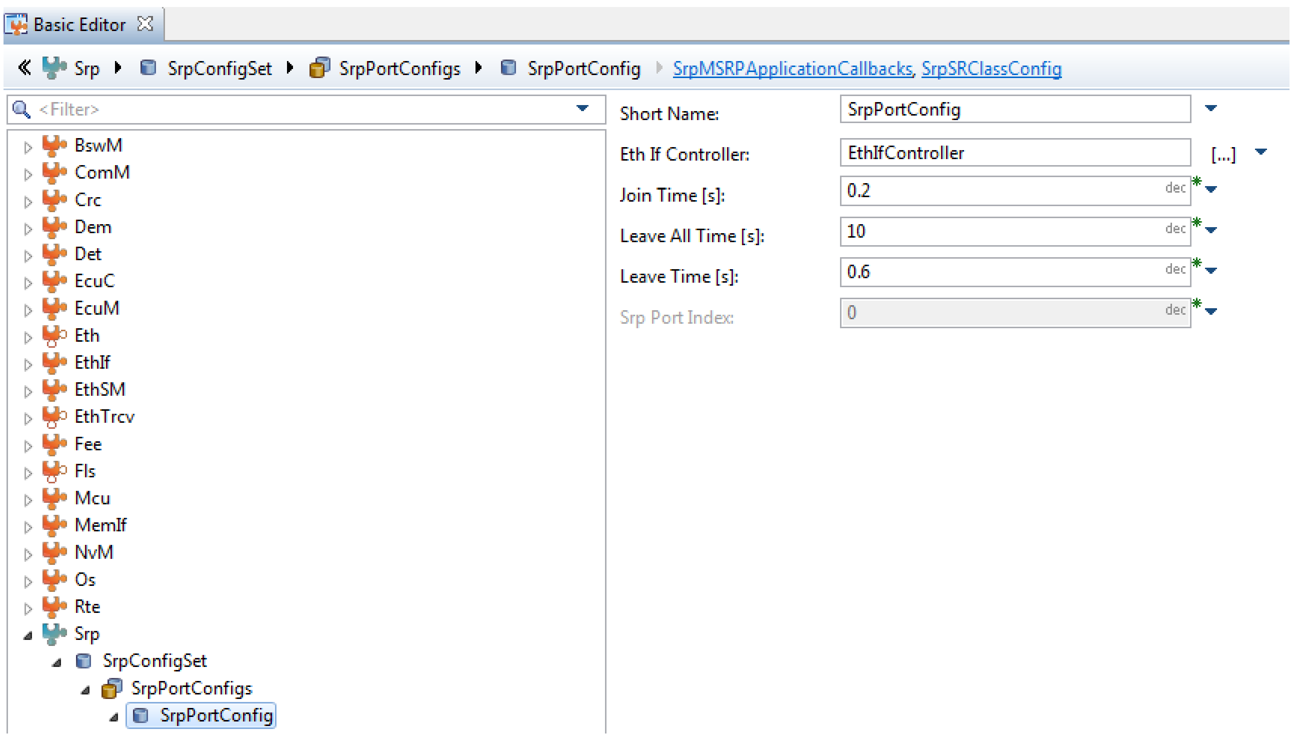
Short Name:	SrpGeneral
DET Instance ID:	116
Dev Error Detect:	✓
Enable Multiple MAC Registration Protocol:	✓
MMRP Destination MAC Address:	01:80:C2:00:00:20
MSRP Destination MAC Address:	01:80:C2:00:00:0E
MVRP Destination MAC Address:	01:80:C2:00:00:21
Main Function Period [s]:	0.005
Srp Type:	STATIC
Supported MMRP Mac Addresses:	20
Supported MSRP Attributes Total:	21
Supported MSRP Streams:	20
Supported MVRP VLans:	20
User Config File:	$(DpaProjectFolder)\Appl\Source\Srp_User
Version Info:	✓

SrpClient.h

Figure 6-1 Configuration overview of the SrpGeneral container

6.2.2 SrpPortConfig container

Figure 6-2 shows an overview of a SrpPortConfig container in an existing project. The SrpPortConfig holds all configuration parameters regarding one Srp port. A detailed description of each configuration parameter can be found in the description view of the parameter properties.


Basic Editor

Srp ▸ SrpConfigSet ▸ SrpPortConfigs ▸ SrpPortConfig ▸ SrpMSRPApplicationCallbacks, SrpSRClassConfig

<Filter>

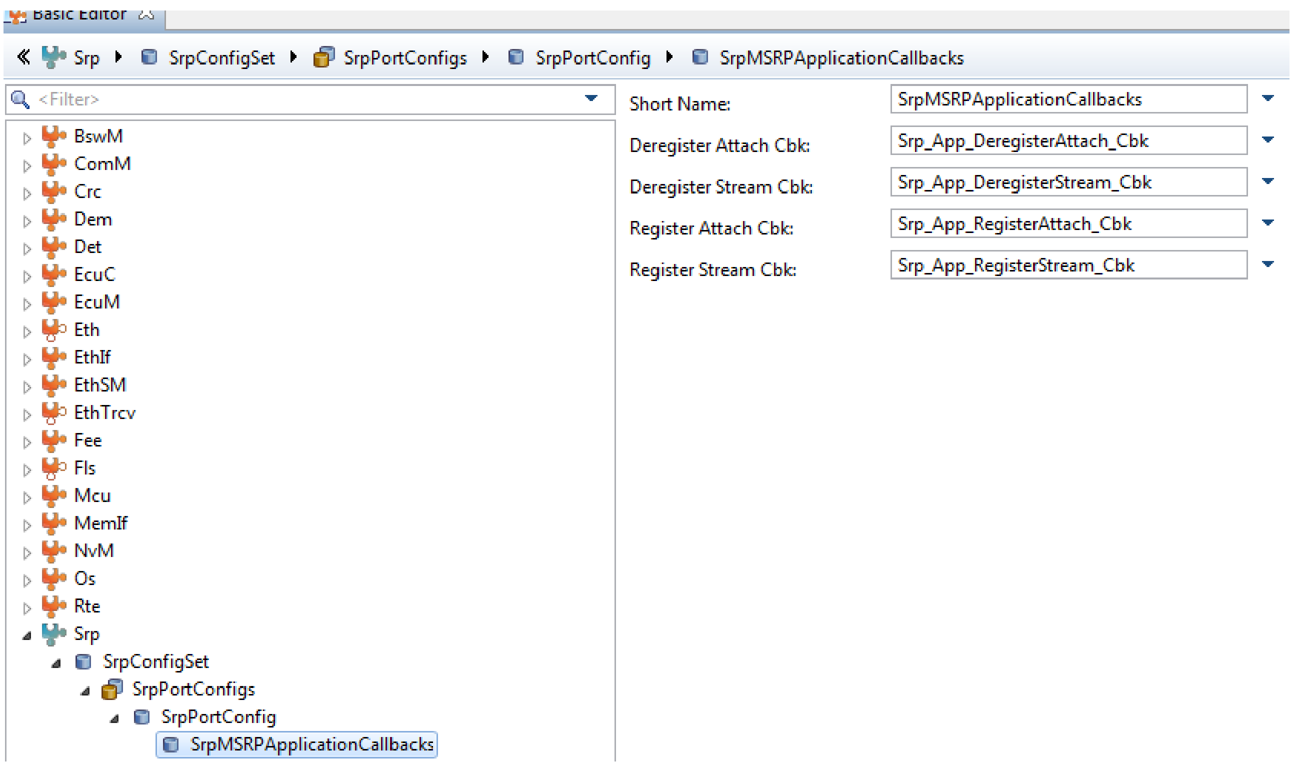
BswM, ComM, Crc, Dem, Det, EcuC, EcuM, Eth, EthIf, EthSM, EthTrcv, Fee, Fls, Mcu, MemIf, NvM, Os, Rte, Srp
- SrpConfigSet
  - SrpPortConfigs
    - SrpPortConfig

Short Name:	SrpPortConfig
Eth If Controller:	EthIfController
Join Time [s]:	0.2
Leave All Time [s]:	10
Leave Time [s]:	0.6
Srp Port Index:	0

Figure 6-2 Configuration overview of SrpPortConfig container

6.2.3 SrpMSRPApplicationCallbacks container

Figure 6-3 shows an overview of the SrpMSRPApplicationCallbacks container in an existing project. The SrpMSRPApplicationCallbacks container holds all configuration parameters used for the configuration of the notification callback functions. A detailed description of each configuration parameter can be found in the description view of the parameter properties.

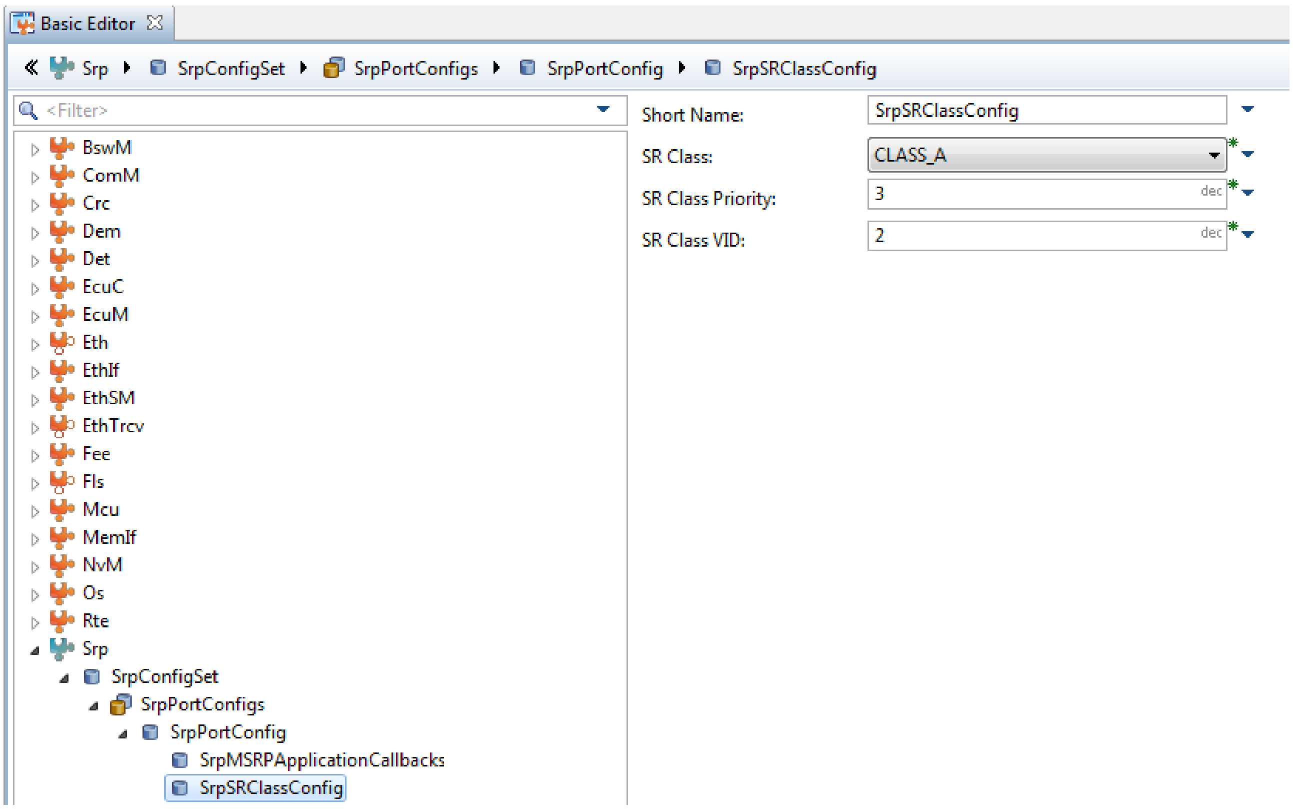


Short Name: SrpMSRPApplicationCallbacks
Deregister Attach Cbk: Srp_App_DeregisterAttach_Cbk
Deregister Stream Cbk: Srp_App_DeregisterStream_Cbk
Register Attach Cbk: Srp_App_RegisterAttach_Cbk
Register Stream Cbk: Srp_App_RegisterStream_Cbk

Figure 6-3 Configuration overview of SrpMSRPApplicationCallbacks container

6.2.4 SrpSRClassConfig container

Figure 6-4 shows an overview of the SrpSRClassConfig container in an existing project. The SrpSRClassConfig



Short Name: SrpSRClassConfig
SR Class: CLASS_A
SR Class Priority: 3 dec
SR Class VID: 2 dec

Figure 6-4 Configuration overview of SrpSRClassConfig container

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
DaVinci Configurator Pro	DaVinci Configurator Pro 5 generation tool for MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
AVB	Audio/Video Bridging
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
MAC	Medium Access Control
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
MMRP	Multiple MAC Registration Protocol
MRP	Multiple Registration Protocol
MSRP	Multiple Stream Registration Protocol
MVRP	Multiple V-LAN Registration Protocol
SWS	Software Specification

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com