VECTOR

MICROSAR AvTp

Technical Reference

AVB-Stack

Version 3.0.0

Authors	Dennis Müller, Jeroen Laverman, Michael Seidenspinner
Status	Released

Document Information

History

Author	Date	Version	Remarks
Dennis Müller	2014-01-27	1.0.0	Creation of document
Jeroen Laverman	2014-03-19	1.0.1	Include Review
Jeroen Laverman	2014-04-08	1.0.2	Add: Memory Mapping, section Multi-Controller, new APIs and new Configuration view, include structure
Jeroen Laverman	2014-05-15	1.0.3	Add: new Configuration view, new APIs, updated Interfaces (EthStbM)
Jeroen Laverman	2014-06-25	1.0.4	Add Diagnostics and Counter feature
Michael Seidenspinner	2014-10-30	1.0.5	Update DET descriptions, updated Figure 2-2
Jeroen Laverman	2015-03-20	1.0.6	ESCAN00080450
Jeroen Laverman	2015-04-17	1.0.7	ESCAN00082366
Michael Seidenspinner	2015-08-11	1.1.0	ESCAN00084236, ESCAN00084431
Michael Seidenspinner	2017-01-30	2.0.0	IEEE1722a-2015-d16
Michael Seidenspinner	2017-03-13	3.0.0	R18 Update

Reference Documents

No.	Source	Title	Version
[1]	IEEE	IEEE 1722-2011: Standard for Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks	2011
[2]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	v3.4.0
[3]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	V5.0.0
[4]	Vector	TechnicalReference_GPtp.pdf	V1.0.0
[5]	Vector	TechnicalReference_IpBase.pdf	v1.1.3
[6]	Vector	TechnicalReference_EthIf.pdf	v3.0.0
[7]	IEEE	IEEE 1722a-2014-d8: Standard for Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks	2014-d8
[8]	IEEE	IEEE 1722a-2015-d13: Standard for Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks	2015-d13
[9]	IEEE	IEEE 1722a-2015-d16: Draft Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks	P1722-rev1/D16

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
1.00.XX	Created
1.01.XX	ESCAN00080450, ESCAN00082366
1.02.XX	ESCAN00083860
1.03.XX	ESCAN00084356
2.00.XX	Configurator 5 Breaking Change. No functional changes
3.00.XX	Update to MICROSAR R17. No functional changes
4.00.XX	Update to IEEE1722a-2015-d16
5.00.XX	Update to R18 (removed dependency to EthTSyn and GPtp, added possibility to use StbM as TimeSource)

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module AvTp.

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	AVTP_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	AVTP_MODULE_ID	255 decimal
Instance ID:	AVTP_INSTANCE_ID	115 decimal

* For the precise AUTOSAR Release 3.x please see the release specific documentation.

The AvTp module handles the stream transmission and reception of Audio/Video data that conform to Audio/Video Transport Protocol AVTP (see [1] IEEE 1722-2011: Standard for Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks).

2.1 Architecture Overview

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

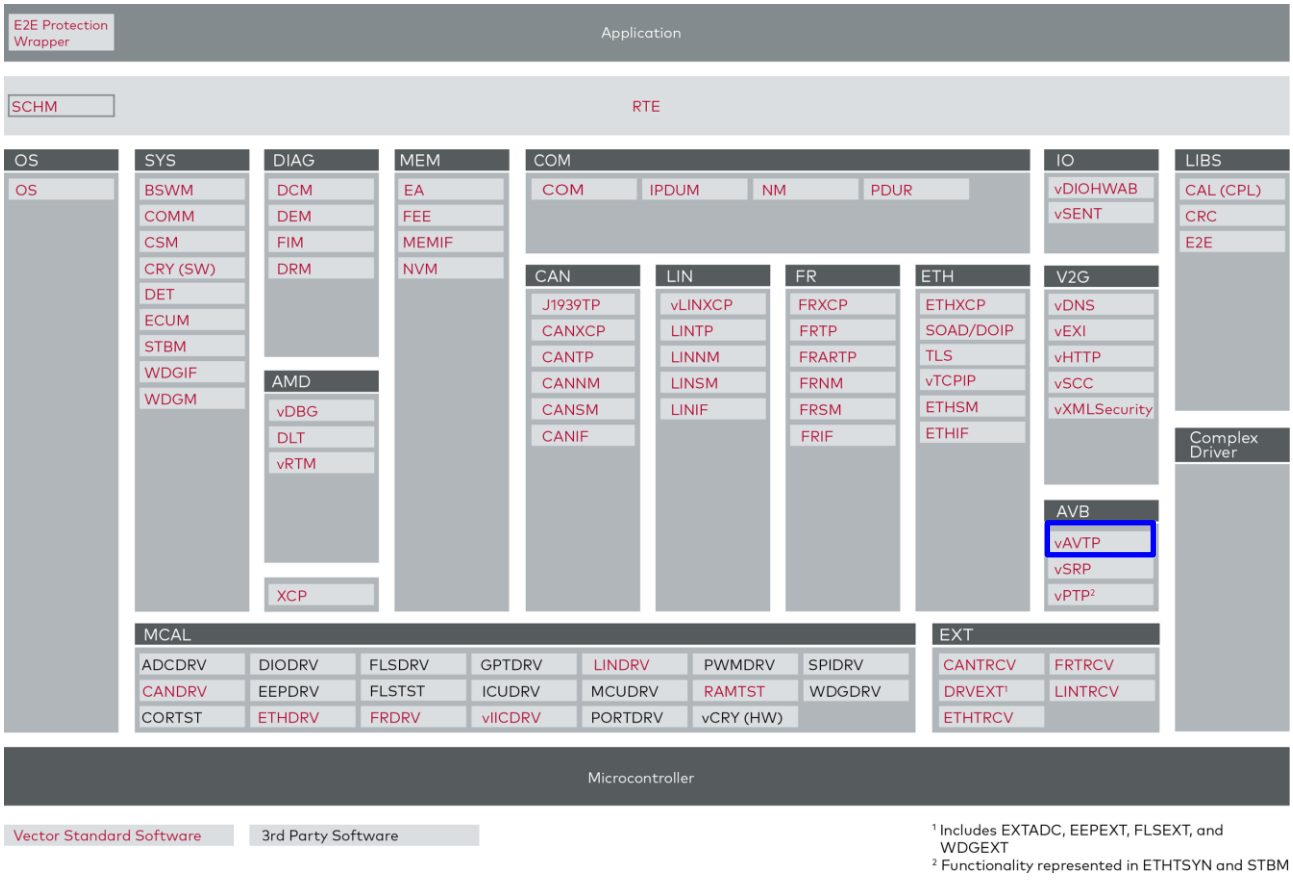

Figure 2-1 AUTOSAR 4.x Architecture Overview

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

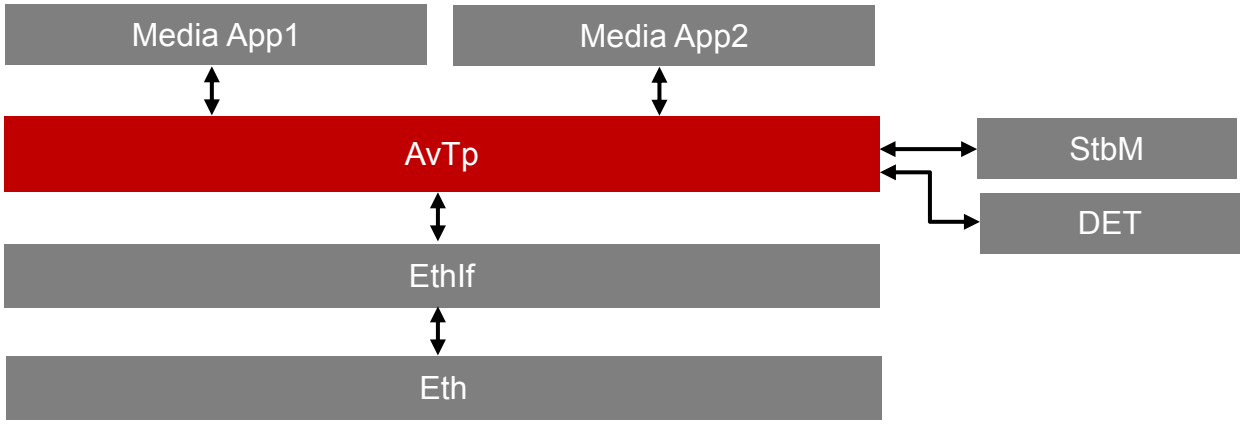

Figure 2-2 Interfaces to adjacent modules of the AvTp

The AvTp module handles the Audio Video Transport Protocol (AVTP) described in [1] (IEEE 1722-2011: Standard for Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks). The AVTP is based on layer 2 above the EthIf using Ethernet frames with certain frame type (EtherType). The main task of the AVTP is the encapsulation of media data (e.g. audio samples) with associated additional information.

The most important information provided by the AVTP header is shown in Table 2-1.

AvTp Header Information	Description
Sequence Number	Continuous incremented number to detect lost or missing frames on listener endpoints
Stream ID	Network-wide unique number to identify a certain media stream
Timestamp	Determines the presentation time of transported media data
Subtype	Determines the transported media type and format (e.g. IEC 61883 media format)

Table 2-1 Important AVTP header information

3 Functional Description

3.1 Features

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

3.1.1 Overview

The AvTp module handles the transmission and reception of AVTP frames. When media data should be transmitted by AvTp, the correct AVTP header Information (see Table 2-1) is set. The AVTP header Timestamp (Presentation time) can be obtained from the StbM module which achieve a time synchronization to a clock master. Additionally the Timestamp can be provided by a User-Callout. If no time source is available, the AVTP header Timestamp is set to zero and marked as invalid and uncertain.

The multicast destination MAC address of the AVTP frames can be dynamically assigned by the MAC Address Acquisition Protocol (MAAP). If no MAAP support is configured, the configured multicast destination MAC address is used.

3.1.2 IEEE 1722-2011 Conformance

The following features specified in [1] IEEE 1722-2011: Standard for Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks are supported:

Supported Features
AVTPDU common header format
AVTPDU common control header format
AVTP common stream data AVTPDU header format
Timing and synchronization
IEC 61883/IIDC over AVTP
MAC Address Acquisition Protocol (MAAP)

Table 3-1 Supported IEEE 1722-2011 features

3.1.3 IEEE 1722-2011 Deviations

Section in Document	Deviation
B.3.4 Protocol timers	No timer variance is supported
B.3.6.1 Generate_address	No randomly selection of address ranges is supported
B.3.6.4 Compare_MAC	No Compare_MAC function is supported

Table 3-2 Not supported IEEE 1722-2011 features

3.1.4 IEEE 1722a-2014-d8 Conformance

Supported Features
AVTP Audio format
Compressed Video Format
Clock Reference Format
Diagnostics and Counters

Table 3-3 Supported IEEE 1722a-2014-d8 features

3.1.5 IEEE 1722a-2014-d8 Deviations

Section in Document	Deviation
E.2.1 Media_Locked	No counter Media_Locked is supported
E.2.2 Media_Unlocked	No counter Media_Unlocked is supported
E.2.3 Stream_Interrupted	No counter Stream_Interrupted is supported
E.2.10 Late_Timestamp	No counter Late_Timestamp is supported
E.2.11 Early_Timestamp	No counter Early_Timestamp is supported

Table 3-4 Not supported IEEE 1722a-2014-d8 features

3.1.6 IEEE 1722a-2015-d13 Conformance

Supported Features
Vendor Specific Format

Table 3-5 Supported IEEE 1722a-2015-d3 features

3.1.7 General Limitations

Table 3-6 shows general limitations that apply to the AvTp.

Limitations
No general limitations known

Table 3-6 General Limitations

3.2 Initialization

The AvTp is initialized by calling the `AvTp_InitMemory()` and `AvTp _Init()` services without parameter. The `AvTp_InitMemory()` function has to be called before `AvTp_Init()` to initialize used memory of the AvTp module.

The AvTp stream configuration is pre-defined by Configurator Pro configuration process.

3.3 States

The AvTp is operational after initialization. If MAAP support is enabled, a multicast MAC address is allocated after module initialization. Until the completion of MAAP process no transmit of AVTP frames is possible. The upper layer Media App gets a callback if the MAAP process finished successful and AvTp is ready to transmit frames.

3.4 Main Functions

The AvTp has a `AvTp_MainFunction()` that handles cyclic tasks like timeout handling and processing of state machines needed for MAAP operation.

3.5 Stream Reception

If a valid AVTPDU is received, the Stream ID is compared to pre-configured RxStreams. If the Stream ID is equal to a pre-configured RxStreams ID the associated pre-configured receive callback function of the Media App is called with the contained media data and additional information about the received stream (e.g. Presentation Timestamp, see Table 5-2 for additional information).

3.6 Stream Transmission

The transmission of media data can be first initiated after the state-changed callback of the Media App is invoked by the AvTp with the state `AVTP_STATE_TX_READY`. After that a transmit buffer is provided by the call of `AvTp_ProvideTxBuffer()`. After the buffer is filled with media data the transmission is executed by calling `AvTp_Transmit()`.

Caution
For each transmit stream only one transmit buffer is available. Due to that fact the transmission of media data is required (by calling `AvTp_Transmit()`) after calling `AvTp_ProvideTxBuffer()` before getting a new transmit buffer.

3.7 Multi-Controller

AvTp is multi-controller capable. Due to the fact that AvTp is and EthIf upper layer and therefor unable to distinguish between PHY and VLAN controllers but only virtual controllers (see [6]) every virtual controller has its instance of MAAP, if enabled. MAAP then allocates multicast MAC addresses for all TxStreams configured on this virtual controller.

3.8 Include Structure

Figure 3-1 shows the include structure of AvTp.

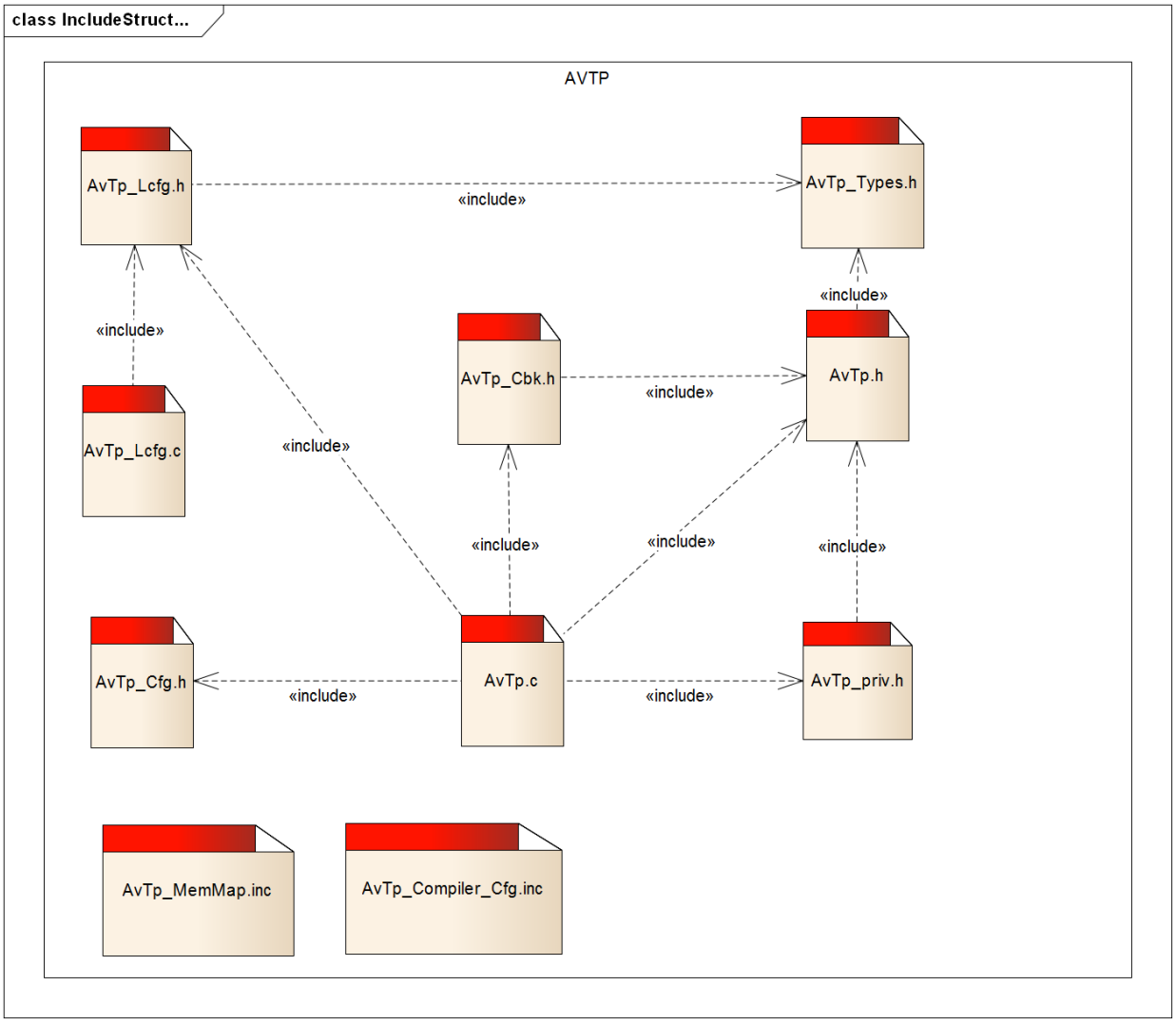

Figure 3-1 include structure

3.9 Error Handling

3.9.1 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 `AVTP_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 AvTp ID is 255. The reported AvTp instance ID is 115.

This instance ID indicates, that the error is reported by the AvTp.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x01	`AvTp_GetVersionInfo()`

Service ID	Service
0x02	AvTp_Init()
0x10	AvTp_ProcessStreamPdu()
0x11	AvTp_ProcessControlPdu()
0x12	AvTp_Cbk_TrcvLinkStateChg()
0x13	AvTp_ProvideTxBuffer()
0x14	AvTp_Transmit()
0x15	AvTp_RxIndication()
0x16	AvTp_GetRxStreamConfig()
0x17	AvTp_GetTxStreamConfig()
0x18	AvTp_SetEvent()
0x19	AvTp_SetMarker()
0x1A	AvTp_SetMr()
0x1B	AvTp_SetSy()
0x1C	AvTp_SetFs()
0x1D	AvTp_ReadDiagnosticCounterListener()
0x1E	AvTp_ReadDiagnosticCounterTalker()
0x1F	AvTp_ResetDiagnosticCounter()
0x20	AvTp_Maap_RxIndication()
0x21	AvTp_Maap_CompareMac()
0x22	AvTp_Maap_CheckConflict()
0x23	AvTp_Maap_CalculateConflictCount()
0x24	AvTp_Maap_TxDefend()
0x30	AvTp_Aaf_WriteSample()
0x40	AvTp_GetCurrentTime()

Table 3-7 Service IDs

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

Error Code	Description
0x01	AVTP_E_NOT_INITIALIZED
0x02	AVTP_E_INV_POINTER
0x03	AVTP_E_INV_LENGTH
0x04	AVTP_E_INV_SUBTYPE
0x05	AVTP_E_NOT_TX_READY
0x06	AVTP_E_INV_STREAM_IDX
0x07	AVTP_E_INV_VCTRL_IDX
0x08	AVTP_E_INV_CTRL_IDX
0x09	AVTP_E_MULTIPLE_CALLS
0x0A	AVTP_E_INV_FRAME_TYPE

Error Code	Description
0x0B	AVTP_E_INV_TIME_SOURCE
0x20	AVTP_MAAP_E_NOT_INITIALIZED
0x21	AVTP_MAAP_E_INV_POINTER
0x22	AVTP_MAAP_E_INV_LENGTH
0x30	AVTP_AUDIO_E_INV_FORMAT
0x40	AVTP_CRF_E_INV_TYPE

Table 3-8 Errors reported to DET

3.9.2 Production Code Error Reporting

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

Error Code	Description
None	

Table 3-9 Errors reported to DEM

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
AvTp.h	API declaration
AvTp.c	Implementation of the AvTp core functionality.
AvTp_Cbk.h	API Callback declaration
AvTp_Types.h	Internal type definitions for AvTp modules
AvTp_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
AvTp_Cfg.h	Pre-compile time parameter configuration
AvTp_Lcfg.h	Link-time parameter configuration declaration
AvTp_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 AvTp module the exclusive area AVTP_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 way that it is ensured that the API functions do not interrupt each other.

It is recommended to use the functions SuspendAllInterrupts() and ResumeAllInterrupts() for AVTP_EXCLUSIVE_AREA_0 to ensure data consistency.

4.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction definitions which are defined for the Audio Video Transport Protocol and illustrates their assignment among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	AVTP_CONST	AVTP_VAR	AVTP_CODE
AVTP_START_SEC_CONST_UNSPECIFIED	■		
AVTP_START_SEC_VAR_NOINIT_UNSPECIFIED		■	
AVTP_START_SEC_VAR_NOINIT_8BIT		■	
AVTP_START_SEC_VAR_NOINIT_32BIT		■	
AVTP_START_SEC_CODE			■

Table 4-3 Compiler abstraction and memory mapping

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

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

Type Name	C-Type	Description	Value Range
AvTp_TimestampType	uint32	AvTp Timestamp	0x00000000 – 0xFFFFFFFF
AvTp_StreamIdxType	uint8	AvTp Stream Index	0 – 255
AvTp_DataType	uint8	AvTp Data	0 – 255
AvTp_CounterType	uint32	AvTp Diagnostic Counter	0x00 – 0xFFFFFFFF
boolean	uint8	Boolean	TRUE / FALSE

Table 5-1 Type definitions

AvTp_RxPduInfoType

This generic structure contains the received AVTPDU information provided to upper layer. According to the Subtype this generic structure has to be casted to the specific RxInfoType.

Struct Element Name	C-Type	Description	Value Range
Subtype	uint8	Media subtype	-

Table 5-2 AvTp_RxPduInfoType

AvTp_RxIECInfoType

This structure contains the received AVTPDU information provided to upper layer receiving a IEC61883-IIDC PDU.

Struct Element Name	C-Type	Description	Value Range
Subtype	uint8	Media subtype	0x00
Sy	uint8	Sy-field	0-15
Following Parameters are only available if 'RxPduInfo' is enabled			
SequenceNum	uint8	Sequence Number	0 - 255
MediaClockReset	boolean	Media Clock Reset	TRUE / FALSE
IsValidTimestamp	boolean	Valid Timestamp indication	TRUE / FALSE
IsUncertainTimestamp	boolean	Uncertain Timestamp indication	TRUE / FALSE

Struct Element Name	C-Type	Description	Value Range
Timestamp	AvTp_TimestampType	Presentation Time	-

Table 5-3 AvTp_RxIECInfoType

AvTp_RxAudioInfoType

This structure contains the received AVTPDU information provided to upper layer receiving a AVTP Audio PDU.

Struct Element Name	C-Type	Description	Value Range
Subtype	uint8	Media subtype	0x02
Event	uint8	Event	0 - 15
Following Parameters are only available if 'RxPduInfo' is enabled			
SequenceNum	uint8	Sequence Number	0 - 255
MediaClockReset	boolean	Media Clock Reset	TRUE / FALSE
IsValidTimestamp	boolean	Valid Timestamp indication	TRUE / FALSE
IsUncertainTimestamp	boolean	Uncertain Timestamp indication	TRUE / FALSE
Timestamp	AvTp_TimestampType	Presentation Time	-

Table 5-4 AvTp_RxAudioInfoType

AvTp_RxCmpVideoInfoType

This structure contains the received AVTPDU information provided to upper layer receiving a Compressed Video PDU

Struct Element Name	C-Type	Description	Value Range
Subtype	uint8	Media subtype	0x03
Event	uint8	Event	0 - 15
M	boolean	Marker Bit	TRUE / FALSE
Following Parameters are only available if 'RxPduInfo' is enabled			
SequenceNum	uint8	Sequence Number	0 - 255
MediaClockReset	boolean	Media Clock Reset	TRUE / FALSE
IsValidTimestamp	boolean	Valid Timestamp indication	TRUE / FALSE
IsUncertainTimestamp	boolean	Uncertain Timestamp indication	TRUE / FALSE
Timestamp	AvTp_TimestampType	Presentation Time	-

Table 5-5 AvTp_RxCmpVideoInfoType

AvTp_RxClockReferenceInfoType

This structure contains the received AVTPDU information provided to upper layer receiving a Clock Reference PDU

Struct Element Name	C-Type	Description	Value Range
Subtype	uint8	Media subtype	0x04
Fs	boolean	Frame Sync	TRUE / FALSE
Following Parameters are only available if 'RxPduInfo' is enabled			
SequenceNum	uint8	Sequence Number	0 - 255
MediaClockReset	boolean	Media Clock Reset	TRUE / FALSE
IsValidTimestamp	boolean	Valid Timestamp indication	TRUE / FALSE
IsUncertainTimestamp	boolean	Uncertain Timestamp indication	TRUE / FALSE

Table 5-6 AvTp_RxCmpVideoInfoType

AvTp_DiagnosticCounterListenerType

This structure contains all diagnostic counters related to a Listener stream. This type is only available if Diagnostic Counters are enabled.

Struct Element Name	C-Type	Description	Value Range
SeqNumMismatch	AvTp_CounterType	Counts non-sequential sequence_num	0-0xFFFFFFFF
MediaReset	AvTp_CounterType	Counts toggle of Mr-Bit	0-0xFFFFFFFF
TimestampUncertain	AvTp_CounterType	Counts toggle of Tu-Bit	0-0xFFFFFFFF
TimestampValid	AvTp_CounterType	Incr. on frame received with Tv-Bit set	0-0xFFFFFFFF
TimestampNotValid	AvTp_CounterType	Incr. on frame received with Tv-Bit cleared	0-0xFFFFFFFF
UnsupportedFormat	AvTp_CounterType	Counts frames received with unsupported format	0-0xFFFFFFFF
FramesRx	AvTp_CounterType	Counts received frames	0-0xFFFFFFFF
LateTimestamp	AvTp_CounterType	Counts frames received containing timestamp in	0-0xFFFFFFFF

Struct Element Name	C-Type	Description	Value Range
		the past	
FirstFrame	AvTp_CounterType	Indicates of an initial frame has been received	TRUE / FALSE

Table 5-7 AvTp_DiagnosticCounterListenerType

AvTp_DiagnosticCounterTalkerType

This structure contains all diagnostic counters related to a Listener stream. This type is only available if Diagnostic Counters are enabled.

Struct Element Name	C-Type	Description	Value Range
MediaReset	AvTp_CounterType	Counts toggle of Mr-Bit	0–0xFFFFFFFF
TimestampUncertain	AvTp_CounterType	Counts toggle of Tu-Bit	0–0xFFFFFFFF
TimestampValid	AvTp_CounterType	Incr. on frame transmitted with Tv-Bit set	0–0xFFFFFFFF
TimestampNotValid	AvTp_CounterType	Incr. on frame transmitted with Tv-Bit cleared	0–0xFFFFFFFF
FrameTx	AvTp_CounterType	Counts transmitted frames	0–0xFFFFFFFF

Table 5-8 AvTp_DiagnosticCounterTalkerType

5.2 Services provided by AvTp

5.2.1 AvTp_GetVersionInfo

Prototype	
`void AvTp_GetVersionInfo (AVTP_P2VAR(Std_VersionInfoType) VersionInfoPtr)`	
Parameter	
VersionInfoPtr	Pointer to a memory location where the AvTp version information shall be stored.
Return Code	
void	none
Functional Description	
Return the BCD-coded version information of the AvTp module.	
Particularities and Limitations	
none	
Pre-Conditions	
> Availability: This function is only available if AvTpVersionInfoApi is enabled.	
Call Context	
This function can be called in any context.	

Table 5-9 AvTp_GetVersionInfo

5.2.2 AvTp_InitMemory

Prototype	
`void` **`AvTp_InitMemory`** `(void)`	
Parameter	
none	
Return Code	
void	none
Functional Description	
Memory initialization of AvTp module.	
Particularities and Limitations	
This function has to be called before AvTp_Init.	
Pre-Conditions	
none	
Call Context	
This function can be called in any context.	

Table 5-10 AvTp_InitMemory

5.2.3 AvTp_Init

Prototype	
`void` **`AvTp_Init`** `(void)`	
Parameter	
none	
Return Code	
void	none
Functional Description	
Initialization of AvTp module.	
Particularities and Limitations	
This function has to be called before using the module.	
Pre-Conditions	
Function AvTp_InitMemory() has to be called first.	
Call Context	
This function can be called in any context.	

Table 5-11 AvTp_Init

5.2.4 AvTp_MainFunction

Prototype	
`void AvTp_MainFunction (void)`	
Parameter	
none	
Return Code	
void	none
Functional Description	
Handles the periodic tasks of state machines, timers and timeouts of the AvTp module.	
Particularities and Limitations	
This function has to be called periodically.	
Pre-Conditions	
none	
Call Context	
This function can be called in any context.	

Table 5-12 AvTp_MainFunction

5.2.5 AvTp_ProvideTxBuffer

Prototype	
`BufReq_ReturnType AvTp_ProvideTxBuffer (AvTp_StreamIdxType StreamIdx, AVTP_P2VAR(AvTp_DataType)*BufPtr, AVTP_P2VAR(uint16) LenBytePtr)`	
Parameter	
StreamIdx	Pointer to the received data
BufPtr	Pointer to the address where the buffer is placed
LenBytePtr	is an in/out parameter. [in] The requested buffer length. [out] The actual buffer length reduced by Ethernet header and AvTp header length. The actual buffer length is equal or bigger than the requested payload length as far as the return value is BUFREQ_OK.
Return Code	
BufReq_ReturnType	BUFREQ_OK Buffer request accomplished successful. BUFREQ_E_NOT_OK Buffer request not successful. Buffer cannot be accessed. BUFREQ_E_BUSY Temporarily no buffer available. It's up the requester to retry for a certain time. BUFREQ_E_OVFL No Buffer of the required length can be provided.
Functional Description	
Provides a transmit buffer to a upper layer module.	

Particularities and Limitations
none
Pre-Conditions
none
Call Context
This function can be called in task context.

Table 5-13 AvTp_ProvideTxBuffer

5.2.6 AvTp_Transmit

Prototype	
`Std_ReturnType` **`AvTp_Transmit`** `(AvTp_StreamIdxType StreamIdx, uint16 LenByte)`	
Parameter	
StreamIdx	Index of TxStream
LenByte	Byte count of the data to transmit
Return Code	
Std_ReturnType	E_OK Data transmission was successful. E_NOT_OK Data could not be transmitted
Functional Description	
Transmits a AvTp stream pdu.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
Function could be called from interrupt level or from task level	

Table 5-14 AvTp_Transmit

5.2.7 AvTp_Aaf_WriteSamples

Prototype	
`Std_ReturnType` **`AvTp_Aaf_WriteSamples`** `(AvTp_StreamIdxType StreamIdx, AVTP_P2CONST(AvTp_DataType) SamplePtr)`	
Parameter	
StreamIdx	Index of Stream
SamplePtr	Pointer to current sample frame (containing samples of all channel)

Return Code	
Std_ReturnType	E_OK Writing Sample was successful E_NOT_OK Event could not be set
Functional Description	
Write AVTP Audio samples to Buffer. If the configured number of samples per frame has been written the Frame is transmitted. This function can only be used with subtype AVTP Audio and other formats than USERSPEC.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
Function could be called from interrupt level or from task level	

Table 5-15 AvTp_Aaf_WriteSamples

5.2.8 AvTp_SetMr

Prototype	
`Std_ReturnType AvTp_SetMr (AvTp_StreamIdxType StreamIdx, boolean Mr)`	
Parameter	
StreamIdx	Index of Stream
Mr	Value of Mr-bit
Return Code	
Std_ReturnType	E_OK Setting Mr-bit was successful E_NOT_OK Mr-bit could not be set
Functional Description	
Set Media-Clock-Restart-Bit (Mr) which will be transmitted with next sent Pdu of corresponding stream.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
Function could be called from interrupt level or from task level	

Table 5-16 AvTp_SetMr

5.2.9 AvTp_SetEvent

Prototype	
`Std_ReturnType` **`AvTp_SetEvent`** `(AvTp_StreamIdxType StreamIdx, uint8 Event)`	
Parameter	
StreamIdx	Index of Stream
Event	Event
Return Code	
Std_ReturnType	E_OK Setting Event was successful E_NOT_OK Event could not be set
Functional Description	
Set Event which will be transmitted with next sent Pdu.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
Function could be called from interrupt level or from task level	

Table 5-17 AvTp_SetEvent

5.2.10 AvTp_SetMarker

Prototype	
`Std_ReturnType` **`AvTp_SetMarker`** `(AvTp_StreamIdxType StreamIdx, boolean M)`	
Parameter	
StreamIdx	Index of Stream
M	Marker Bit
Return Code	
Std_ReturnType	E_OK Setting Marker Bit was successful E_NOT_OK Marker Bit could not be set
Functional Description	
Set Marker Bit which will be transmitted with next sent Pdu.	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	

Function could be called from interrupt level or from task level

Table 5-18 AvTp_SetMarker

5.2.11 AvTp_SetTu

Prototype	
Std_ReturnType **AvTp_SetTu**(AvTp_StreamIdxType StreamIdx, boolean Tu)	
Parameter	
`StreamIdx`	Index of Stream
`Tu`	Timing uncertain
Return code	
Std_ReturnType	E_OK Setting Tu was successful E_NOT_OK Tu could not be set
Functional Description	
Set Tu for a Clock Reference Format PDU which will be transmitted with next sent Pdu	
Particularities and Limitations	
none	
Call context	
> Function could be called from interrupt level of from task level	

Table 5-19 AvTp_SetTu

5.2.12 AvTp_SetH264Timestamp

Prototype	
Std_ReturnType **AvTp_SetH264Timestamp**(AvTp_StreamIdxType StreamIdx, uint32* H264TimestampPtr	
Parameter	
`StreamIdx`	Index of Stream
`H264TimestampPtr`	Pointer to the H264 Timestamp
Return code	
Std_ReturnType	E_OK Setting the H264 Timestamp was successful E_NOT_OK H264 Timestamp could not be set
Functional Description	
Set H264 Timestamp for a Compressed Video Format H264 PDU which will be transmitted with next sent Pdu	
Particularities and Limitations	
none	
Call context	
Function could be called from interrupt level or from task level	

Table 5-20 AvTp_SetH264Timestamp

5.2.13 AvTp_GetTxStreamConfig

Prototype	
`Std_ReturnType` **`AvTp_GetTxStreamConfig`**`(AvTp_StreamIdxType StreamIdx, AVTP_P2VAR(AVTP_P2CONSTCFG(AvTp_StreamCfgType)) StreamCfgPtr)`	
Parameter	
StreamIdx	Index of Stream
StreamCfgPtr	Pointer to config of corresponding values
Return Code	
Std_ReturnType	E_OK Pointer is valid E_NOT_OK Pointer is not valid
Functional Description	
Get a pointer to configured values of corresponding TxStream	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
Function could be called from interrupt level or from task level	

Table 5-21 AvTp_GetTxStreamConfig

5.2.14 AvTp_GetRxStreamConfig

Prototype	
`Std_ReturnType` **`AvTp_GetRxStreamConfig`**`(AvTp_StreamIdxType StreamIdx, AVTP_P2VAR(AVTP_P2CONSTCFG(AvTp_StreamCfgType)) StreamCfgPtr)`	
Parameter	
StreamIdx	Index of Stream
StreamCfgPtr	Pointer to config of corresponding values
Return Code	
Std_ReturnType	E_OK Pointer is valid E_NOT_OK Pointer is not valid
Functional Description	
Get a pointer to configured values of corresponding RxStream	
Particularities and Limitations	
none	
Pre-Conditions	
none	

Call Context
Function could be called from interrupt level or from task level

Table 5-22 AvTp_GetRxStreamConfig

5.2.15 AvTp_ReadDiagnosticCounterListener

Prototype	
`Std_ReturnType` **`AvTp_ReadDiagnosticCounterListener`**`( AvTp_StreamIdxType StreamIdx, AVTP_P2VAR(AVTP_P2VAR( AvTp_DiagnosticCounterListenerType)) DiagnosticCounterListenerPtr)`	
Parameter	
StreamIdx	Index of Rx-Stream
DiagnosticCounterListenerPtr	Pointer to set of Diagnostic Counters of a Listener stream
Return Code	
Std_ReturnType	E_OK Pointer is valid E_NOT_OK Pointer is not valid
Functional Description	
Readout current state of a Diagnostic Counter for Listener stream	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
Function could be called from interrupt level or from task level	

Table 5-23 AvTp_ReadDiagnosticCounterListener

5.2.16 AvTp_ReadDiagnosticCounterTalker

Prototype	
`Std_ReturnType` **`AvTp_ReadDiagnosticCounterTalker`**`( AvTp_StreamIdxType StreamIdx, AVTP_P2VAR(AVTP_P2VAR( AvTp_DiagnosticCounterTalkerType)) DiagnosticCounterTalkerPtr)`	
Parameter	
StreamIdx	Index of Rx-Stream
DiagnosticCounterTalkerPtr	Pointer to set of Diagnostic Counters of a Talker stream
Return Code	
Std_ReturnType	E_OK Pointer is valid E_NOT_OK Pointer is not valid

Functional Description
Readout current state of a Diagnostic Counter for Talker stream
Particularities and Limitations
none
Pre-Conditions
none
Call Context
Function could be called from interrupt level or from task level

Table 5-24 AvTp_ReadDiagnosticCounterTalker

5.2.17 AvTp_ResetDiagnosticCounter

Prototype	
`Std_ReturnType` **`AvTp_ResetDiagnosticCounter`**`( AvTp_StreamIdxType StreamIdx,boolean IsListener)`	
Parameter	
StreamIdx	Index of Rx-Stream
IsListener	TRUE: StreamIdx is a Listener(Rx)-StreamIdx FALSE: StreamIdx is a Talker(Tx)-StreamIdx
Return Code	
Std_ReturnType	E_OK Pointer is valid E_NOT_OK Pointer is not valid
Functional Description	
Readout current state of a Diagnostic Counter for Talker stream	
Particularities and Limitations	
none	
Pre-Conditions	
none	
Call Context	
Function could be called from interrupt level or from task level	

Table 5-25 AvTp_ResetDiagnosticCounter

5.3 Services used by AvTp

In the following table services provided by other components, which are used by the AvTp 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
SchM	SchM_Enter_AvTp
SchM	SchM_Exit_AvTp
StbM	StbM_GetCurrentTime

Table 5-26 Services used by the AvTp

5.4 Callback Functions

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

1.1.1 AvTp_RxIndication

Prototype	
void **AvTp_RxIndication** (uint8 CtrlIdx, Eth_FrameType FrameType, boolean IsBroadcast, AVTP_P2VAR(uint8) PhysAddrPtr, AVTP_P2VAR(uint8) DataPtr, uint16 LenByte)	
Parameter	
CtrlIdx	Index of the controller that has received the frame.
FrameType	Ethertype of the frame
IsBroadcast	Determines that the frame was transmitted as broadcast
PhysAddrPtr	Pointer to the physical address of the transmitted interface
DataPtr	Pointer to the received data.
LenByte	Byte count of the received frame.
Return Code	
void	none
Functional Description	
Handles processing of received AvTp frames.	
Particularities and Limitations	
none	
Pre-Conditions	
Call Context	
This function can be called in any context.	

Table 5-27 AvTp_RxIndication

1.1.2 AvTp_Cbk_TrcvLinkStateChg

Prototype	
`void AvTp_Cbk_TrcvLinkStateChg ( uint8 CtrlIdx, AvTp_LinkStateType TrcvLinkState)`	
Parameter	
CtrlIdx	Index of the controller that link status changed
TrcvLinkState	New link state of the transceiver
Return Code	
void	none
Functional Description	
Callback function that indicates a changed transceiver link state.	
Particularities and Limitations	
none	
Pre-Conditions	
Call Context	
Function must be called from task level	

Table 5-28 AvTp_Cbk_TrcvLinkStateChg

6 Configuration

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

6.1 Configuration Variants

The AvTp supports the configuration variants

- VARIANT-PRE-COMPILE

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

6.2 Configuration with DaVinci Configurator Pro

6.2.1 AvTpGeneral Container

Figure 6-1 shows an overview of the AvTpGeneral container in an existing project. Due to the fact that AvTp is not specified by AUTOSAR it is shown as Complex Device Driver (CDD) module in the Basic Editor. The AvTpGeneral 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.

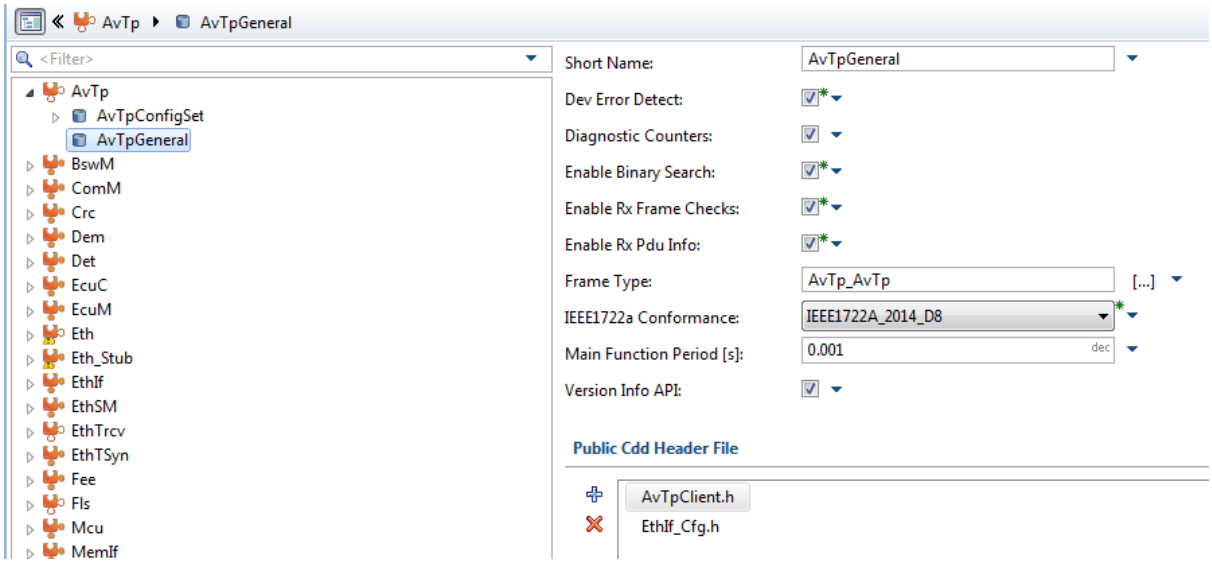

Figure 6-1 Configuration overview of AvTpGeneral container

6.2.2 AvTpRxStreamConfigs Container

Figure 6-2 shows an overview of the AvTpRxStreamConfigs container in an existing project. The AvTpRxStreamConfigs container holds all configuration parameter regarding the reception of streams. A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

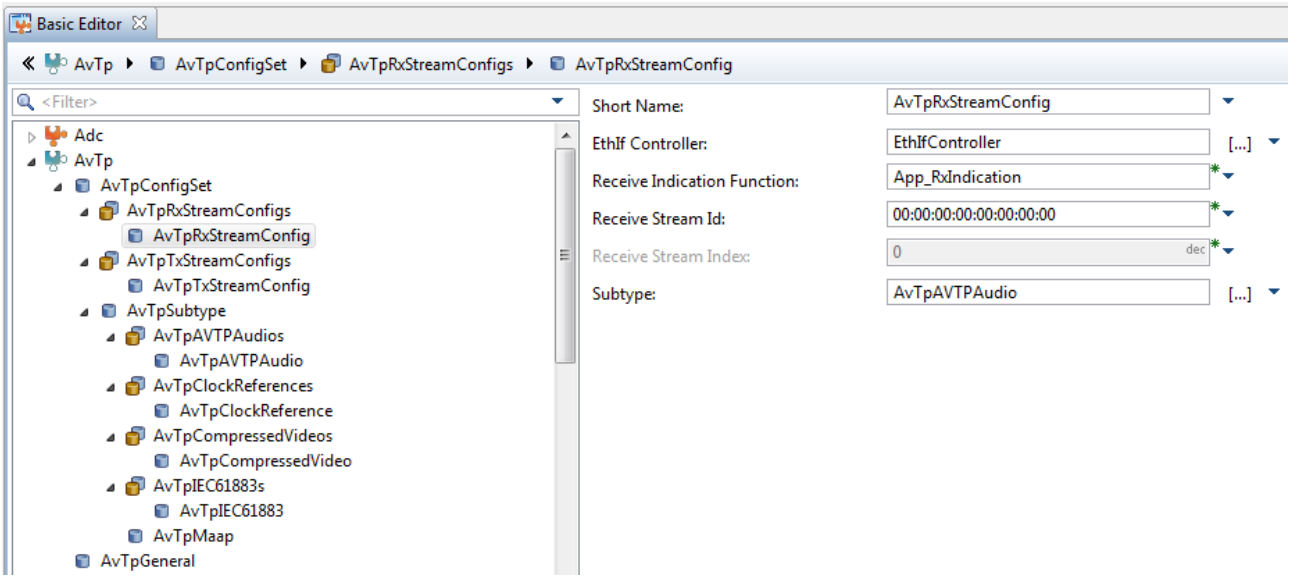

Figure 6-2 Configuration overview of AvTpRxStreamConfigs container

6.2.3 AvTpTxStreamConfigs Container

Figure 6-3 shows an overview of the AvTpTxStreamConfigs container in an existing project. The AvTpTxStreamConfigs container holds all configuration parameter regarding the transmission of streams. A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

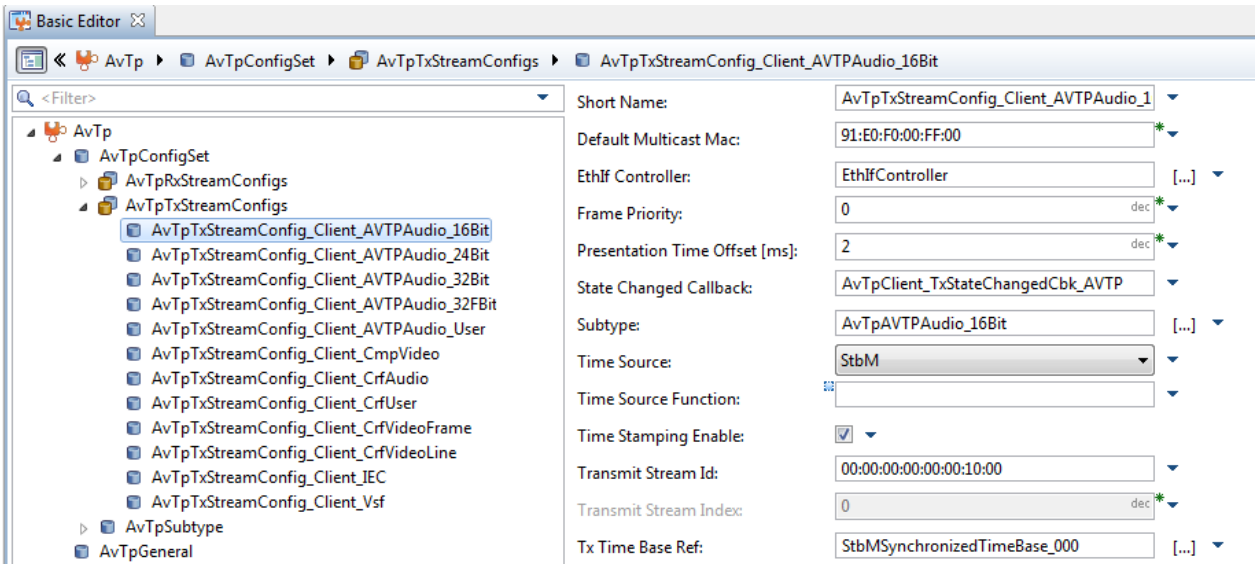

Figure 6-3 Configuration overview of AvTpTxStreamConfigs container

6.2.4 AvTpIEC61883 Container

Figure 6-4 shows an overview of the AvTpIEC61883 container in an existing project. The AvTpIEC61883 container holds all configuration parameter regarding the IEC61883-IIDC format (see [7]). A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

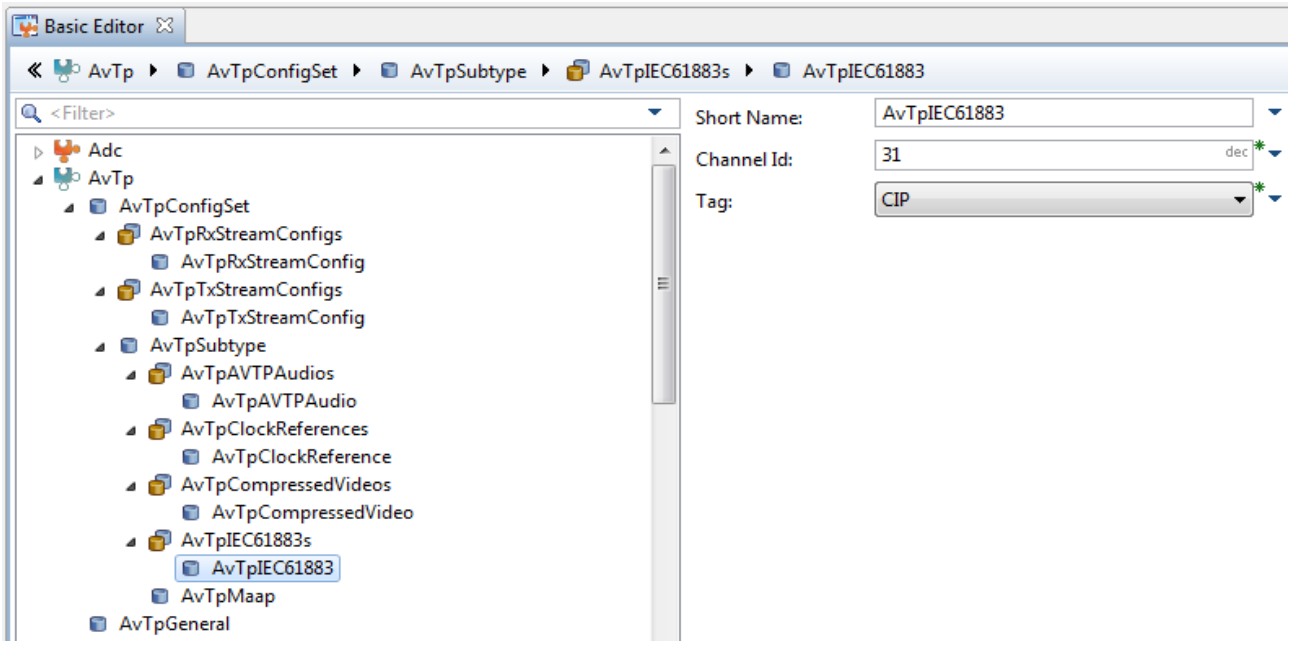

Figure 6-4 Configuration overview of IEC61883 Format

6.2.5 AvTpAudio Container

Figure 6-5 shows an overview of the AvTpAudio container in an existing project. The AvTpAudio container holds all configuration parameter regarding the AVTP Audio format (see [7]). A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

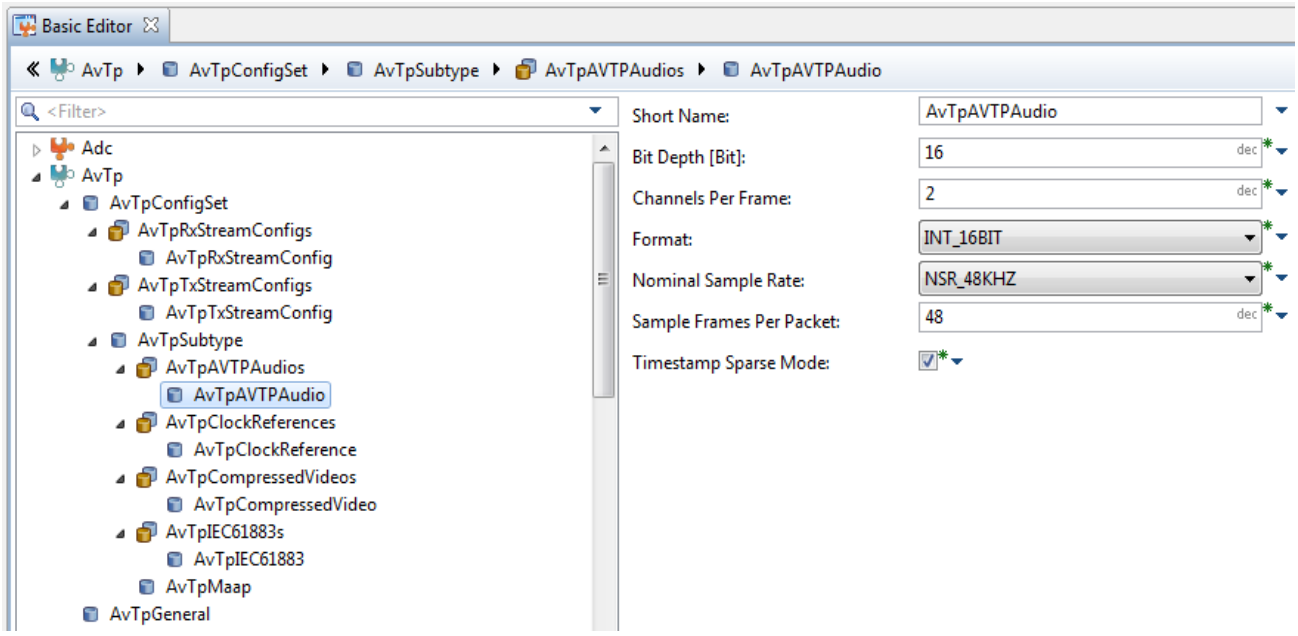

Figure 6-5 Configuration overview of AvTpAudio Format

6.2.6 AvTpCompressedVideo Container

Figure 6-6 shows an overview of the AvTpCompressedVideo container in an existing project. The AvTpCompressedVideo container holds all configuration parameter regarding the Compressed Video format (see [7]). A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

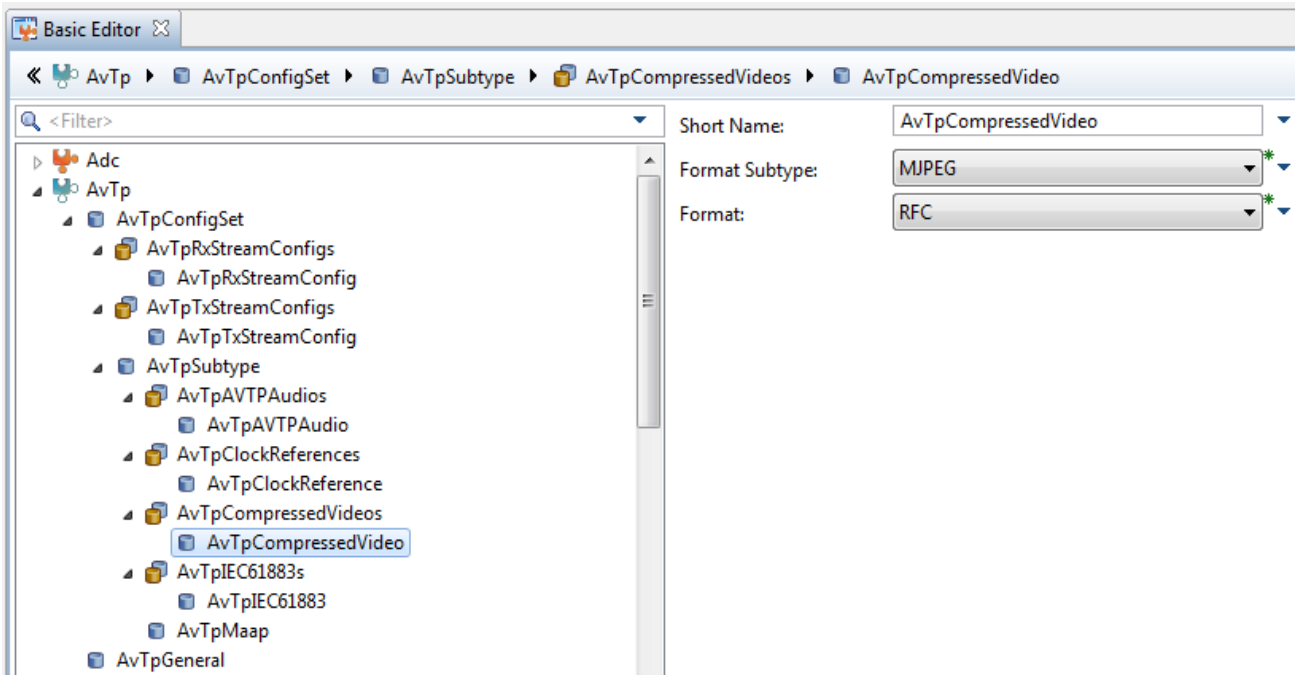

Figure 6-6 Configuration overview of AvTpCompressedVideo Format

6.2.7 AvTpClockReference Container

Figure 6-7 shows an overview of the AvTpClockReference container in an existing project. The AvTpClockReference container holds all configuration parameter regarding the Clock Reference format (see [7]). A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

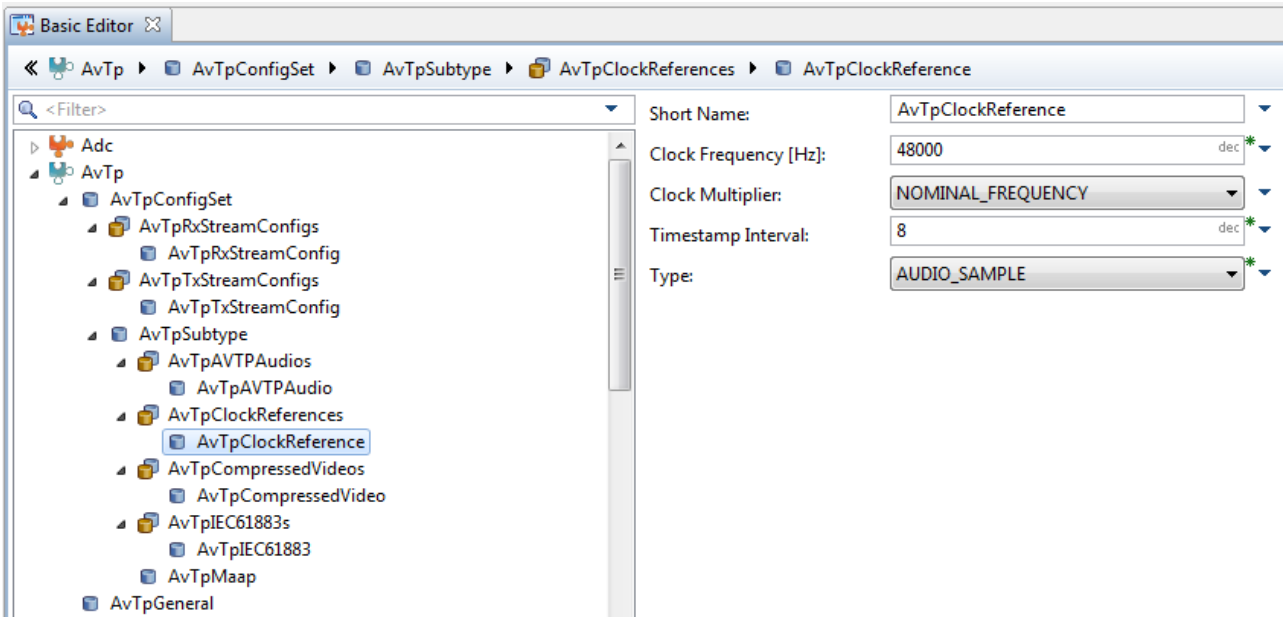

Figure 6-7 Configuration overview of AvTpClockReference container

6.2.8 AvTpVendorSpecificFormat Container

Figure 6-8 shows an overview of the AvTpVendorSpecificFormat container in an existing project. The AvTpVendorSpecificFormat container holds all configuration parameter regarding the Vendor Specific Format. A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

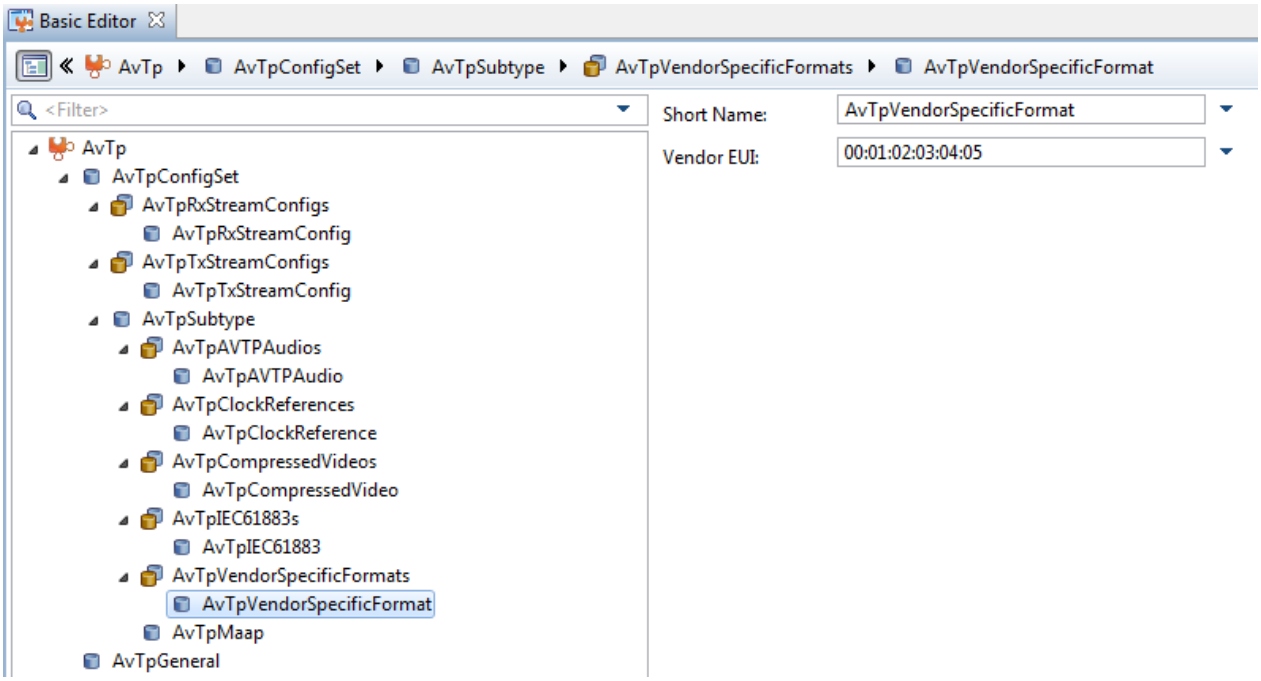

Figure 6-8 Configuration overview of AvTpVendorSpecificFormat container

6.2.9 AvTpMaapConfig Container

Figure 6-9 shows an overview of the AvTpMaapConfig container in an existing project. The AvTpMaapConfig container holds all configuration parameter regarding the MAC Address Acquisition Protocol (MAAP) (see[1]). A detailed description of each configuration parameter can be found in the Description View of the parameter properties.

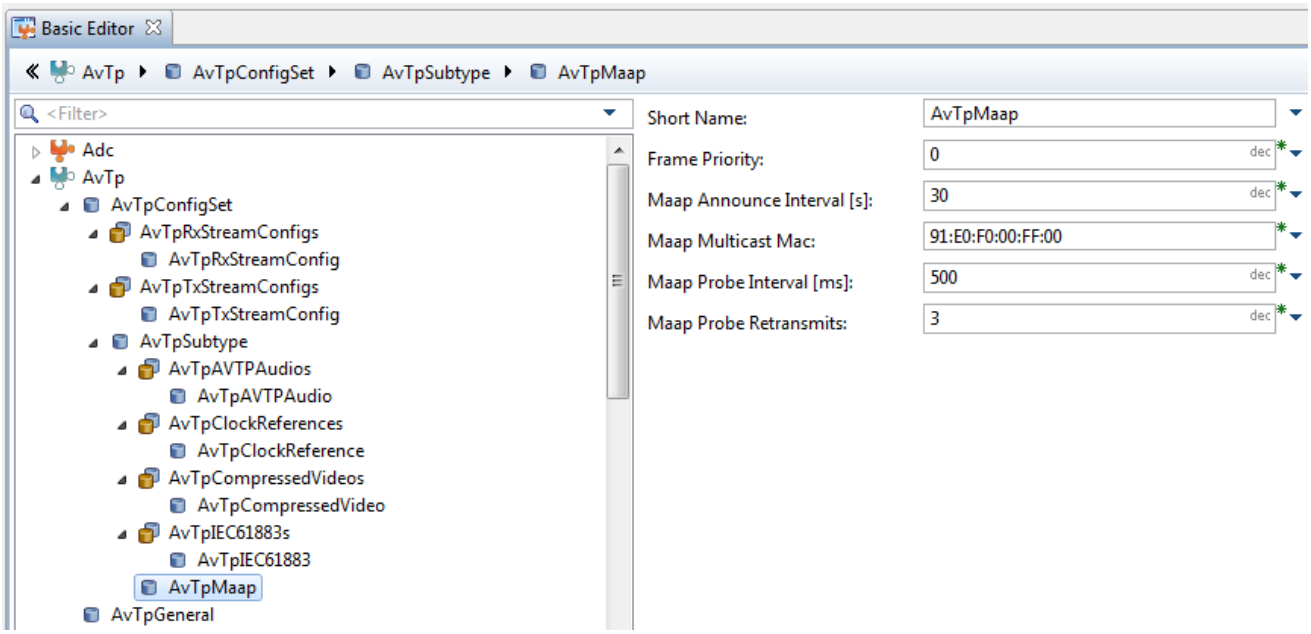

Figure 6-9 Configuration overview of AvTpMaapConfig container

7 Glossary and Abbreviations

7.1 Glossary

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

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AVTP	Audio/Video Transport Protocol
AVTPDU	Audio/Video Transport Protocol Data Unit
MAAP	MAC Address Acquisition Protocol
MMA	Midi Manufacturers Association
PDU	Protocol Data Unit
MAAP	Mac
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
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