MICROSAR RTP

Technical Reference

Real-Time Transport Protocol

Version 1.0.0

Authors	Philipp Christmann
Status	Not Released

Document Information

History

Author	Date	Version	Remarks
Philipp Christmann	2016-09-07	1.0.0	Creation of document

Reference Documents

No.	Source	Title	Version
[1]	RFC3550	RTP: A Transport Protocol for Real-Time Applications	July 2003
[2]	IEEE	IEEE 1733-2011: IEEE Standard for Layer 3 Transport Protocol for Time-Sensitive Applications in Local Area Networks	2011
[3]	AUTOSAR	AUTOSAR_SWS_SocketAdaptor.pdf	4.2.2
[4]	AUTOSAR	AUTOSAR_SWS_DefaultErrorTracer.pdf	4.2.2
[5]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	4.2.2

Scope of the Document

This technical reference describes the general use of the Real-Time Transport Protocol (RTP) basis software.

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 Beta version

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the BSW module RTP as specified in [3]. The RTP Control Protocol (RTCP) is extended as specified in see [2].

Supported AUTOSAR Release*:	4.2.2	
Supported Configuration Variants:	pre-compile	
Vendor ID:	RTP_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	RTP_MODULE_ID	255 decimal (according to [5])

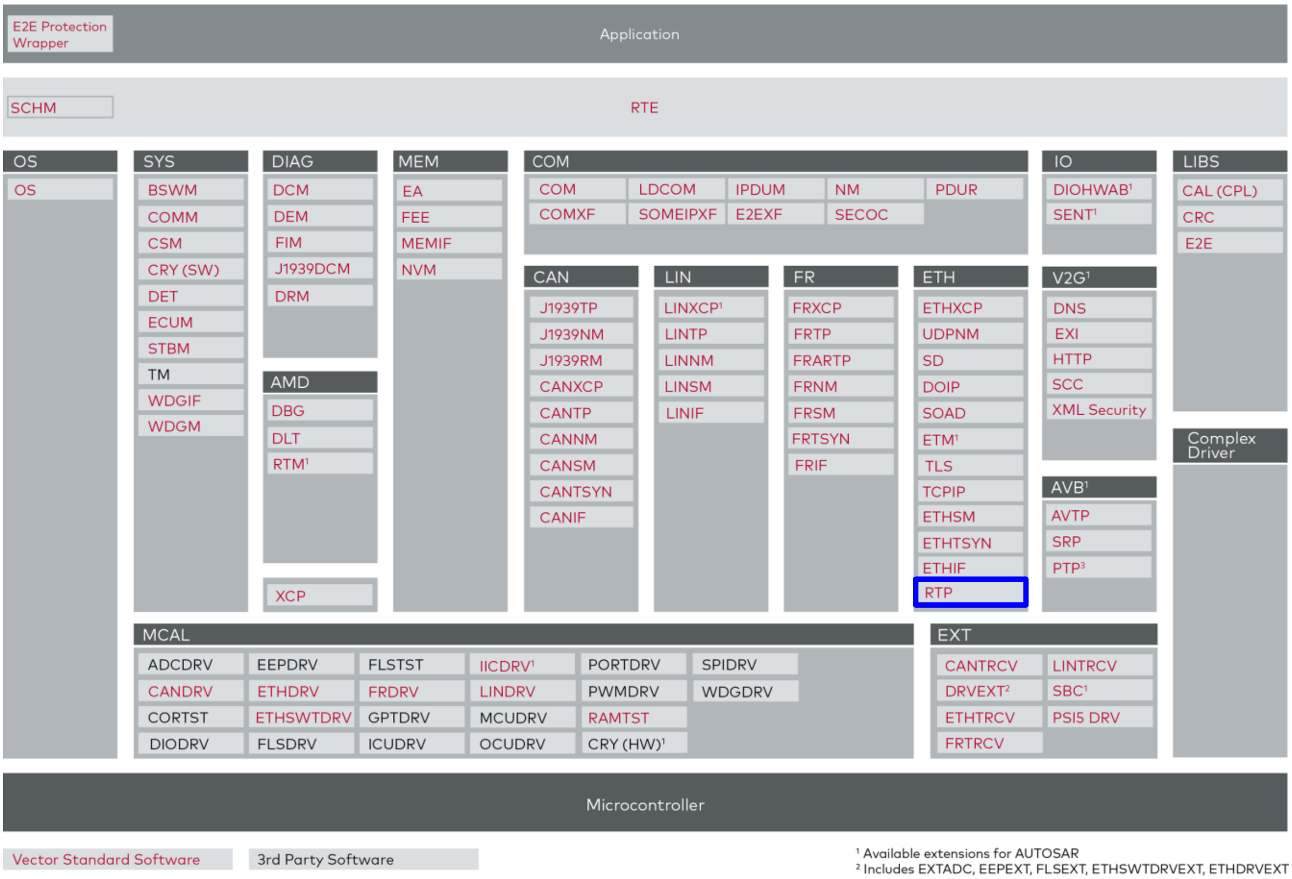
* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

RTP provides an implementation of the Real-Time Transport Protocol (a transport protocol primary for audio and video data, according to RFC3550). The protocol uses UDP messages in order to transport the payload. In addition, the module implements the RTCP (RTP Control Protocol), which can be used to provide feedback of the data distribution as well as additional user information.

In [2] IEEE 1733-2011: IEEE Standard for Layer 3 Transport Protocol for Time-Sensitive Applications in Local Area Networks, an additional RTCP packet type is specified, which allows the reuse of Audio/Video Bridging (AVB) protocols for meeting Quality of Service (QoS) requirements for time-sensitive applications.

2.1 Architecture Overview

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



[1] Available extensions for AUTOSAR
[2] Includes EXTADC, EEPEXT, FLSEXT, ETHSWTDRVEXT, ETHDRVEXT and WDGEXT
[3] Functionality represented in ETHTSYN and STBM

Figure 2-1 AUTOSAR 4.2 Architecture Overview

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

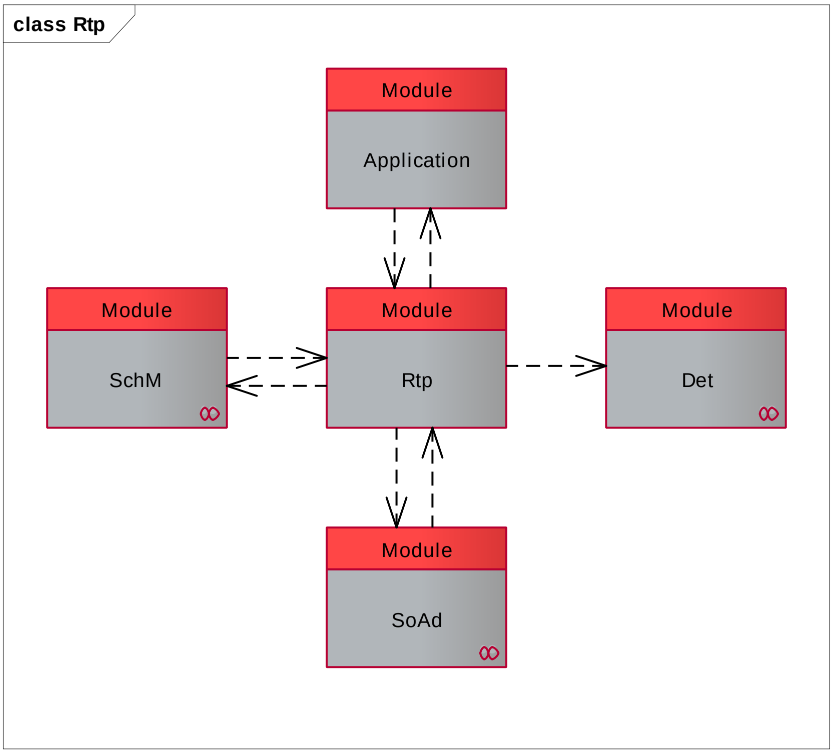


Figure 2-2 Interfaces to adjacent modules of the RTP

3 Functional Description

3.1 Features

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

The standard functionality is specified in [3], the extended AVB functionalities in [2] . The corresponding features are listed in the tables

- Table 3-1 Supported standard conform features
- Table 3-2 Not supported RTP standard conform features
- Table 3-3 Not supported RTCP standard conform features

The following features specified in [3], [2] are supported:

Supported Standard Conform Features
Transmission and reception of RTP packets as specified in [1].
Transmission and reception of extended RTCP packets as specified in [2].
Collision resolution of SSRC value.

Table 3-1 Supported standard conform features

3.1.1 Deviations

The following features specified in [3], [2] are not supported:

Not Supported RTP Standard Conform Features
Functionality of Mixers, Translators and Monitors.
Managing and transmitting a list of contributing sources (CSRC).
Usage of RTP Header Extensions. (Will be ignored on reception.)
Loop detection.

Table 3-2 Not supported RTP standard conform features

Not Supported RTCP Standard Conform Features
Feedback of data transmission/reception by Sender and Receiver Reports (SR, RR) is not supported because RTP is used in combination with IEEE 802.1AS and IEEE 802.1Q features.
Dynamic adaption of the transmission interval and packet content based on the available bandwidth.
Limited set of supported Source Description RTCP packets (SDES). Only the mandatory CNAME is set during transmission.
Encryption of control packets.

Table 3-3 Not supported RTCP standard conform features

3.2 Initialization

The RTP module is initialized via an `Rtp_InitMemory()` call followed by a call of `Rtp_Init()`. Currently only the configuration variant 'pre-compile' is supported, so the configuration pointer used for `Rtp_Init()` has to be a `NULL_PTR`.

3.3 States

After initialization of the module, each configured `RtpTxStream` is disabled and has to be configured with additional information. Therefore, the `Rtp_SetControlInformation()` API has to be called. In addition to the timing parameters, this API has to be used to enable and disable the stream and hence, the periodic transmission of RTCP packets.

Each configured `RtpRxStream` will be enabled automatically after initialization. If corresponding RTP or RTCP packets are received, the upper layer is informed via configurable notifications (see chapter 5.5.2 Notifications).

3.4 Main Functions

The RTP module has a main function that needs to be called periodically. This is normally done by the RTE. If no RTE is used, please call the `Rtp_MainFunction()` manually.

The main tasks are:

- Periodic transmission of RTCP packets.
- Renewal of SSRC if collision was detected.

Note
SSRC collisions can only be detected for communication which is forwarded to the Rtp module. This requires the configuration of at least one `RtpRxStream` as well as a matching IP and Port configuration.

3.5 RTP Message Transmission

After successful configuration and activation of an `RtpTxStream`, the first RTCP message will be transmitted in the context of the next `Rtp_MainFunction()` execution. All succeeding RTCP messages are transmitted based on the configured `RtpRtcpCycleTime`.

RTP data can be transmitted via the `Rtp_Transmit()` API. This API supports two kinds of data provision.

If the data to be transmitted is already serialized and ready for transmission, the API can be called with set `PayloadPtr`. In this case, the RTP module copies the payload in an Ethernet buffer and triggers the transmission.

If the data is not yet serialized, the upper layer has also the possibility to request an Ethernet buffer by calling the `Rtp_Transmit()` API with `PayloadPtr` set to `NULL_PTR`. In this case, the `<UpperLayer>_CopyTxData()` callout will be triggered and the upper layer can serialize its RTP data directly to the Ethernet buffer. This procedure may reduce the processing overhead of copying data to the buffer.

Both kinds of data provision are shown in Figure 3-1 RTP message transmission.

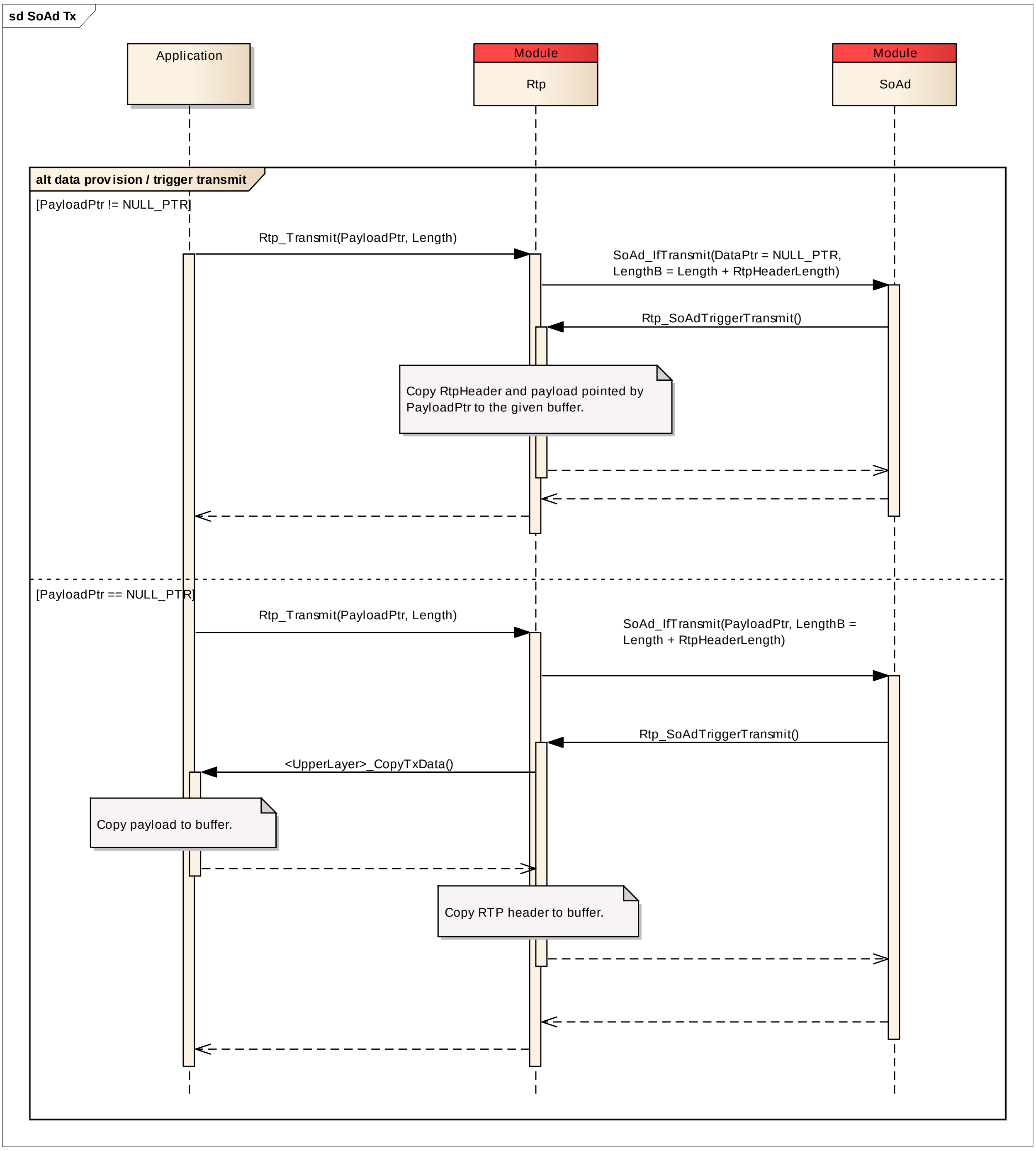

sd SoAd Tx
Application
Module
Rtp
Module
SoAd
alt data provision / trigger transmit
[PayloadPtr != NULL_PTR]
Rtp_Transmit(PayloadPtr, Length)
SoAd_IfTransmit(DataPtr = NULL_PTR, LengthB = Length + RtpHeaderLength)
Rtp_SoAdTriggerTransmit()
Copy RtpHeader and payload pointed by PayloadPtr to the given buffer.
[PayloadPtr == NULL_PTR]
Rtp_Transmit(PayloadPtr, Length)
SoAd_IfTransmit(PayloadPtr, LengthB = Length + RtpHeaderLength)
Rtp_SoAdTriggerTransmit()
<UpperLayer>_CopyTxData()
Copy payload to buffer.
Copy RTP header to buffer.

Figure 3-1 RTP message transmission

The information of the grand master clock as well as the timestamps can be updated by additional `Rtp_SetControlInformation()` calls. The timestamps are also updated with each call of the `Rtp_Transmit()` API.

3.6 RTP Message Reception

If a RTP message is received which matches the configuration of an `RtpRxStream`, the RTP module forwards the payload to the upper layer via the configurable `<UpperLayer>_RtpRxIndication()` API.

3.7 RTCP Message Reception

If a RTCP message is received which matches the configuration of an `RtpRxStream`, the RTP modules extracts the information and forwards it to the upper layer via the configurable `<UpperLayer>_RtcpRxIndication()` and `<UpperLayer>_RtcpSdesInformation()` APIs. The additional SDES information callout can be enabled and disabled by the configuration parameter `RtpSdesInformationCalloutEnabled`.

3.8 Error Handling

3.8.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [4] , if development error reporting is enabled (i.e. pre-compile parameter `RTP_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 RTP ID is 255.

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
`0x10u`	`Rtp_Init()`
`0x11u`	`Rtp_GetVersionInfo()`
`0x12u`	`Rtp_MainFunction()`
`0x13u`	`Rtp_Transmit()`
`0x14u`	`Rtp_SetControlInformation()`
`0x20u`	`Rtp_SoAdRxIndication ()`
`0x21u`	`Rtp_SoAdTriggerTransmit()`

Table 3-4 Service IDs

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

Error Code		Description
`0x00u`	`RTP_E_NO_ERROR`	used to check if no error occurred

Error Code		Description
0x0Au	RTP_E_PARAM_CONFIG	API service Rtp_Init() called with wrong parameter.
0x0Bu	RTP_E_PARAM_VALUE	API called with invalid parameter value.
0x0Cu	RTP_E_PARAM_POINTER	API service used with invalid pointer parameter (NULL).
0x0Du	RTP_E_API_DISABLED	API not available. Configuration mismatch.
0x10u	RTP_E_UNINIT	API service used without module initialization.
0x11u	RTP_E_ALREADY_INITIALIZED	The service Rtp_Init() is called while the module is already initialized.

Table 3-5 Errors reported to DET

3.8.2 Production Code Error Reporting

The RTP module does not use DEM error reporting.

Error Code	Description
none	

Table 3-6 Errors reported to DEM

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
Rtp.c	This is the source file of the RTP module
Rtp.h	API declaration of the module
Rtp_Cbk.h	API declaration of RTP callback functions
Rtp_Priv.h	Component local macro and variable declaration
Rtp_Types.h	Data type declarations

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
Rtp_Cfg.h	Pre-compile time parameter configuration
Rtp_Lcfg.c	Link-time parameter configuration
Rtp_Lcfg.h	Link-time parameter configuration declaration

Table 4-2 Generated files

4.2 Critical Sections

To ensure data consistency and a correct function of the RTP module an exclusive area is used and 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() 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 RTP are described in this chapter.

Type Name	C-Type	Description
Rtp_GmIdentityType	uint8[10]	Array containing the Identity of the grand master clock.

Table 5-1 Type definitions

Rtp_TxRtpInfoType

This struct contains relevant data during transmission of RTP frames.

Struct Element Name	C-Type	Description
RtpTimestamp	uint32	RTP specific timestamp.
AsTimestamp	uint32	IEEE 802.1AS timestamp in [ns]
Padding	boolean	Header flag “padding” indicates if the payload is padded.
Marker	boolean	Header flag “marker” indicates application specific behavior.

Table 5-2 Rtp_TxRtpInfoType

Rtp_RxRtpInfoType

This struct contains relevant data during reception of RTP frames.

Struct Element Name	C-Type	Description
RtpTimestamp	uint32	RTP specific timestamp.
SequenceNumber	uint16	Sequence number of RTP packet.
Padding	boolean	Header flag “padding” indicates if the payload is padded.
Marker	boolean	Header flag “marker” indicates application specific behavior.

Table 5-3 Rtp_RxRtpInfoType

Rtp_RxAvbRtcpInfoType

This struct contains relevant data during reception of RTCP frames.

Struct Element Name	C-Type	Description
NameAscii	uint32	Name to identify RTCP packets.
AsTimestamp	uint32	IEEE 802.1AS timestamp in [ns]
RtpTimestamp	uint32	RTP specific timestamp.
GmTimeBaseIndicator	uint16	Index of the current time base.

Struct Element Name	C-Type	Description
GmIdentity	Rtp_GmIdentityType	Identity of the grand master clock.

Table 5-4 Rtp_RxAvbRtcpInfoType

5.2 Services provided by RTP

5.2.1 Rtp_InitMemory

Prototype	
`void Rtp_InitMemory (void)`	
Parameter	
none	
Return code	
void	void
Functional Description	
Function for Rtp-variable initialization.	
Particularities and Limitations	
Module is uninitialized. Service to initialize module global variables at power up. This function initializes the variables in Rtp sections. Used in case they are not initialized by the startup code.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-5 Rtp_InitMemory

5.2.2 Rtp_Init

Prototype	

`void Rtp_Init (`
`  const Rtp_ConfigType *ConfigPtr)`	

Parameter	
ConfigPtr [in]	Configuration structure for initializing the module
Return code	
void	void
Functional Description	
Initialization function.	
Particularities and Limitations	
Specification of module initialization > Interrupts are disabled. Module is uninitialized. Rtp_InitMemory has been called unless Rtp_ModuleInitialized is initialized by start-up code. This function initializes the module Rtp. It initializes all variables and sets the module state to initialized.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-6 Rtp_Init

5.2.3 Rtp_Transmit

Prototype	

`Std_ReturnType Rtp_Transmit (`
`  uint16 StreamHandleId,`
`  const Rtp_TxRtpInfoType *InfoPtr,`
`  const uint8 *PayloadPtr,`
`  uint16 Length)`	

Parameter	
StreamHandleId [in]	Handle ID of the relevant stream
InfoPtr [in]	Structure containing relevant header and packet information
PayloadPtr [in]	NULL_PTR Data will be requested by additional [UL]_CopyTxData() function call. !NULL_PTR Pointer to data which shall be transmitted.
Length [in]	Length of the data to be transmitted.
Return code	
Std_ReturnType	E_OK Transmission was successful. E_NOT_OK Transmission failed.
Functional Description	
Triggers transmission of a RTP frame.	
Particularities and Limitations	
Module is initialized, corresponding stream is configured with Rtp_SetControlInformation() and activated. The relevant payload can be passed by the API via the PayloadPtr or a buffer of the required size can be requested if PayloadPtr == NULL_PTR.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-7 Rtp_Transmit

5.2.4 Rtp_SetControlInformation

<table>
<tr><th colspan="3">Prototype</th></tr>
<tr><td colspan="3">

```
Std_ReturnType Rtp_SetControlInformation (
  uint16 StreamHandleId,
  boolean StreamEnabled,
  const uint32 *AsTimestampPtr,
  const uint32 *RtpTimestampPtr,
  const uint16 *GmTimeBaseIndicatorPtr,
  const Rtp_GmIdentityType *GmIdentityPtr)
```

</td></tr>
<tr><th colspan="3">Parameter</th></tr>
<tr><td>StreamHandleId [in]</td><td colspan="2">Handle ID of the relevant stream.</td></tr>
<tr><td>StreamEnabled [in]</td><td colspan="2">Enables of disables transmission of RTCP control frames.</td></tr>
<tr><td>AsTimestampPtr [in]</td><td colspan="2">Updates AsTimestamp value if Pointer != NULL_PTR</td></tr>
<tr><td>RtpTimestampPtr [in]</td><td colspan="2">Updates RtpTimestamp value if Pointer != NULL_PTR</td></tr>
<tr><td>GmTimeBaseIndicatorPtr [in]</td><td colspan="2">Updates GmTimeBaseIndicator value if Pointer != NULL_PTR</td></tr>
<tr><td>GmIdentityPtr [in]</td><td colspan="2">Updates GmIdentity value if Pointer != NULL_PTR</td></tr>
<tr><th colspan="3">Return code</th></tr>
<tr><td>Std_ReturnType</td><td>E_OK
E_NOT_OK</td><td>Control information successfully set.
Failed to set control information.</td></tr>
<tr><th colspan="3">Functional Description</th></tr>
<tr><td colspan="3">Updates the RTCP control information.</td></tr>
<tr><th colspan="3">Particularities and Limitations</th></tr>
<tr><td colspan="3">Module is initialized.
Stream has to be configured and enabled in order to start periodic transmission of RTCP frames. The timestamp values will be updated with the values passed at Rtp_Transmit() call.</td></tr>
<tr><th colspan="3">Call context</th></tr>
<tr><td colspan="3">&gt; TASK
> This function is Synchronous
> This function is Reentrant</td></tr>
</table>

Table 5-8 Rtp_SetControlInformation

5.2.5 Rtp_MainFunction

Prototype	
`void` **`Rtp_MainFunction`** `(void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
none	
Particularities and Limitations	
Module is initialized.	
Call context	
TASK	

Table 5-9 Rtp_MainFunction

5.3 Services used by RTP

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

Component	API
DET	Det_ReportError
SoAd	SoAd_IfTransmit

Table 5-10 Services used by the RTP

5.4 Callback Functions

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

5.4.1 Rtp_SoAdRxIndication

Prototype	

`void` **`Rtp_SoAdRxIndication`** `(`
`  PduIdType RxPduId,`
`  const PduInfoType *PduInfoPtr)`	

Parameter	
RxPduId [in]	ID of the received I-PDU.
PduInfoPtr [in]	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Return code	
void	void
Functional Description	
Indication of a received I-PDU from the SoAd lower layer communication interface.	
Particularities and Limitations	
Module is initialized. API is used for reception of RTP and RTCP packets.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-11 Rtp_SoAdRxIndication

5.4.2 Rtp_SoAdTriggerTransmit

Prototype		

`Std_ReturnType` **`Rtp_SoAdTriggerTransmit`** `(`
`PduIdType TxPduId,`
`PduInfoType *PduInfoPtr)`		

Parameter		
TxPduId [in]	ID of the I-PDU to transmit.	
PduInfoPtr [in]	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.	
Return code		
Std_ReturnType	E_OK	Data was successfully copied to the buffer.
	E_NOT_OK	Failed to copy data.
Functional Description		
Trigger RTP to copy its payload in the provided buffer.		
Particularities and Limitations		
Module is initialized. Function is called by SoAd module if ETH buffer is successfully requested and ready to copy data to.		
Call context		
> TASK > This function is Synchronous > This function is Reentrant		

Table 5-12 Rtp_SoAdTriggerTransmit

5.5 Configurable Interfaces

5.5.1 Rtp_GetVersionInfo

<table>
<tr><th colspan="2">Prototype</th></tr>
<tr><td colspan="2"><code>void <b>Rtp_GetVersionInfo</b> (</code>
<code> Std_VersionInfoType *VersionInfoPtr)</code></td></tr>
<tr><th colspan="2">Parameter</th></tr>
<tr><td>VersionInfoPtr [out]</td><td>Pointer to where to store the version information. Parameter must not be NULL.</td></tr>
<tr><th colspan="2">Return code</th></tr>
<tr><td>void</td><td>void</td></tr>
<tr><th colspan="2">Functional Description</th></tr>
<tr><td colspan="2">Returns the version information.</td></tr>
<tr><th colspan="2">Particularities and Limitations</th></tr>
<tr><td colspan="2">none
Rtp_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component.</td></tr>
<tr><td colspan="2">Call context</td></tr>
<tr><td colspan="2">> TASK|ISR2
> This function is Synchronous
> This function is Reentrant</td></tr>
</table>

Table 5-13 Rtp_GetVersionInfo

5.5.2 Notifications

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

5.5.2.1 <UpperLayer>_RtpRxIndication

Prototype	
void **<UpperLayer>_RtpRxIndication** (uint16 StreamHandleId, const Rtp_RxRtpInfoType *InformationPtr, const uint8 *PayloadPtr, uint16 Length)	
Parameter	
StreamHandleId [in]	Handle ID of the stream.
InformationPtr [in]	Structure containing relevant header information.
PayloadPtr [in]	Pointer to received RTP payload.
Length [in]	Length of RTP payload.
Return code	
void	void
Functional Description	
Forward relevant RTP information to the upper layer.	
Particularities and Limitations	
Callout function will be called if a RTP message was received.	
Call context	
> ISR > This function is Synchronous > This function is Reentrant	

Table 5-14 <UpperLayer>_RtpRxIndication

5.5.2.2 <UpperLayer>_RtcpRxIndication

Prototype	

`void <UpperLayer>_RtcpRxIndication (`
`  uint16 StreamHandleId,`
`  const Rtp_RxAvbRtcpInfoType *InformationPtr)`	

Parameter	
StreamHandleId [in]	Handle ID of the stream.
InformationPtr [in]	Structure containing relevant header and packet information.
Return code	
void	void
Functional Description	
Forward relevant RTCP information to the upper layer.	
Particularities and Limitations	
Callout function will be called if a RTCP message was received.	
Call context	
> ISR > This function is Synchronous > This function is Reentrant	

Table 5-15 <UpperLayer>_RtcpRxIndication

5.5.2.3 <UpperLayer>_RtcpSdesInformation

Prototype	

`void <UpperLayer>_RtcpSdesInformation (`
`uint16 StreamHandleId,`
`const uint8 *SdesChunkItemsPtr,`
`uint16 Length)`	

Parameter	
StreamHandleId [in]	Handle ID of the stream.
SdesChunkItemsPtr [in]	Pointer to the received SDES chunk.
Length [in]	Length of the SDES chunk.
Return code	
void	void
Functional Description	
Forwards received SDES chunk items to upper layer.	
Particularities and Limitations	
Callout function will be called if a RTCP message was received which contains SDES items and RtpSdesInformationCalloutEnabled is configured for the stream.	
Call context	
> ISR > This function is Synchronous > This function is Reentrant	

Table 5-16 <UpperLayer>_RtcpSdesInformation

5.5.3 Callout Functions

At its configurable interfaces the RTP defines callout functions. The declarations of the callout functions are provided by the BSW module, i.e. the RTP. It is the integrator's task to provide the corresponding function definitions. The definitions of the callouts can be adjusted to the system's needs. The RTP callout function declarations are described in the following tables:

5.5.3.1 <UpperLayer>_CopyTxData

Prototype	

void **<UpperLayer>_CopyTxData** (
  uint16 StreamHandleId,
  uint8 *PayloadPtr,
  uint16 *Length)	

Parameter	
StreamHandleId [in]	Handle ID of the stream.
PayloadPtr [in]	Pointer to transmission-buffer.
in/out] [in]	Length [in] Length of the available buffer. [out] Length copied by the upper layer.
Return code	
Std_ReturnType	E_OK Data successfully copied. E_NOT_OK Failed to copy data.
Functional Description	
Callout providing a buffer the upper layer has to copy data to.	
Particularities and Limitations	
Module is initialized, corresponding stream is configured with Rtp_SetControlInformation() and activated. Callout function will be called if the upper layer requested a RTP message transmission by Rtp_Transmit() API and RtpCopyTxDataCalloutEnabled is configured for the stream.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-17 <UpperLayer>_CopyTxData

6 Configuration

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

6.1 Configuration Variants

The RTP supports the configuration variants

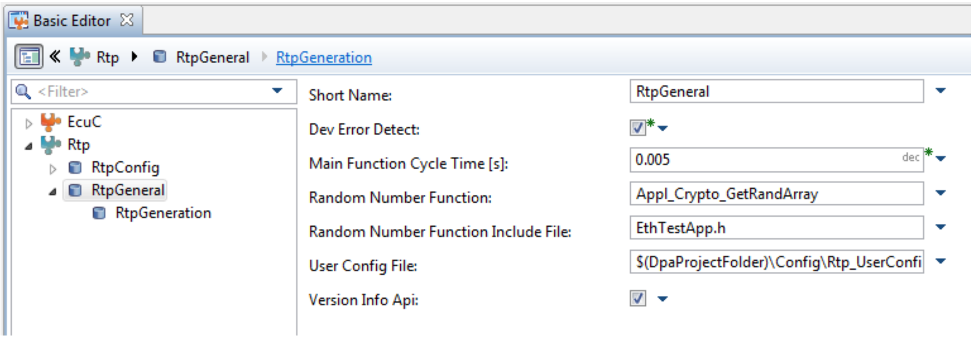
- `VARIANT-PRE-COMPILE`

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

6.2 Configuration with DaVinci Configurator Pro

The configuration of the RTP module is separated into general and stream-dependent configuration parameters.

6.2.1 General Configuration



Short Name: RtpGeneral
Dev Error Detect: ☑
Main Function Cycle Time [s]: 0.005
Random Number Function: Appl_Crypto_GetRandArray
Random Number Function Include File: EthTestApp.h
User Config File: $(DpaProjectFolder)\Config\Rtp_UserConfi
Version Info Api: ☑

Figure 6-1 RtpGeneral Container

The `RtpGeneral` container shown in Figure 6-1 contains configuration parameters which are stream-independent and valid for the entire RTP module. A detailed description of each configuration parameter can be found in the description view of the parameter properties.

The `RtpRandomNumberFunctionIncludeFile` and `RtpRandomNumberFunction` have to be provided by the customer if an `RtpTxStream` is configured. The random number function is used by the RTP module to generate random SSRC values as well as random initial sequence numbers. The random number function has to fill a pointed memory range with random data.

- `void <RandNoFct>(uint8* TgtDataPtr, uint16 TgtLen)`

The `RtpUserConfigFile` allows the user to define additional code, which will be generated to the end of the `Rtp_Cfg.h` file.

The configuration parameters in the `RtpGeneration` container control the generation of dynamic files and configure for example additional out of bounds read/write sanitizer or several data reduction strategies.

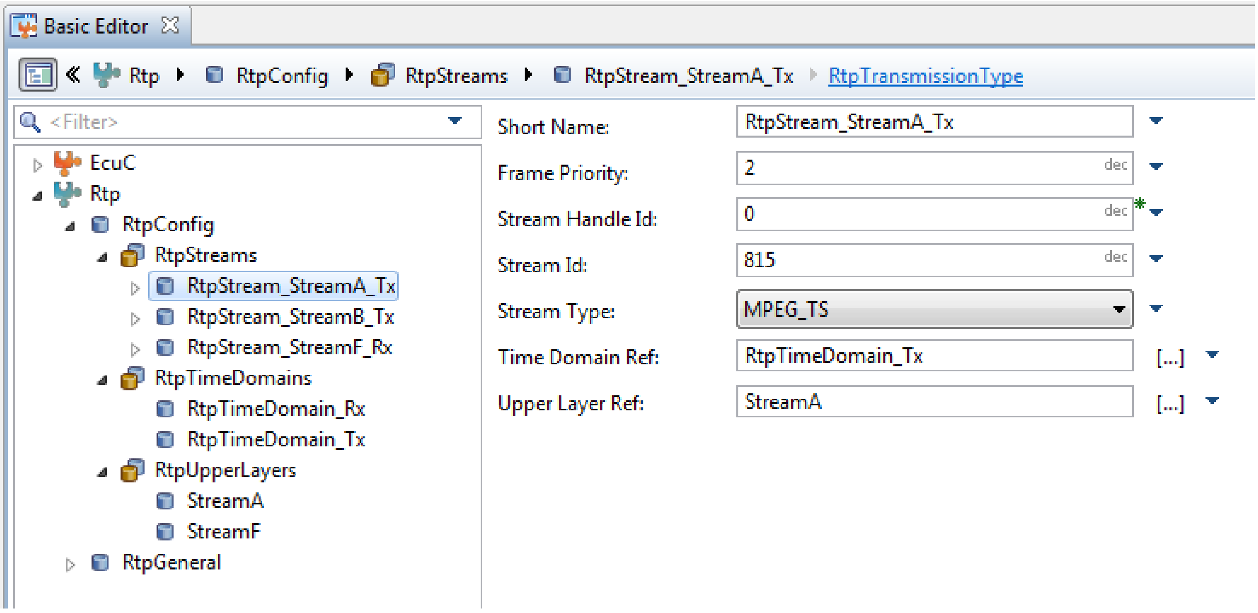
6.2.2 Stream Configuration

Each stream is configured in an `RtpStream` container, which contains general configuration parameters as shown in Figure 6-2. The `RtpUpperLayerRef` defines the `<UpperLayer>` of the stream and influences the name of the generated APIs.

The `RtpStreamHandleId` is used as API parameter in order to identify the corresponding stream. For each stream, a symbolic name value `#define` is generated to the `Rtp_Cfg.h`.

The `Rtp_Cfg.h` contains also global constant macros to access the configured:

- Stream ID: `RTP_UL_GETSTREAMID(HandleId)`
- Payload Type: `RTP_UL_GETTYPEOFSTREAM(HandleId)`



Short Name: RtpStream_StreamA_Tx

Frame Priority: 2 dec

Stream Handle Id: 0 dec

Stream Id: 815 dec

Stream Type: MPEG_TS

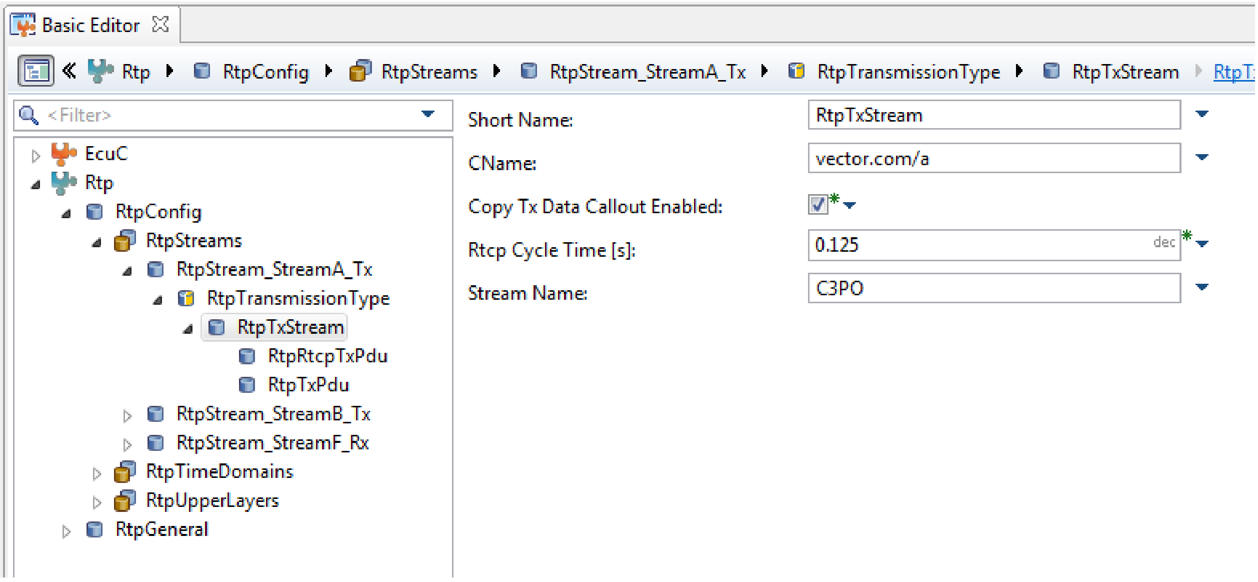
Time Domain Ref: RtpTimeDomain_Tx [...]

Upper Layer Ref: StreamA [...]

Figure 6-2 Stream Configuration

6.2.3 TX Stream Configuration

For each configured `RtpTxStream` the user can choose between two possible ways of data provision as described in Section 3.5. The possibility to use the `<UpperLayer>_CopyTxData()` callout during runtime requires the enabled `RtpCopyTxDataCalloutEnabled` parameter. The `RtpTxStream` configuration parameters can be seen in Figure 6-3.



Short Name: RtpTxStream

CName: vector.com/a

Copy Tx Data Callout Enabled: ☑

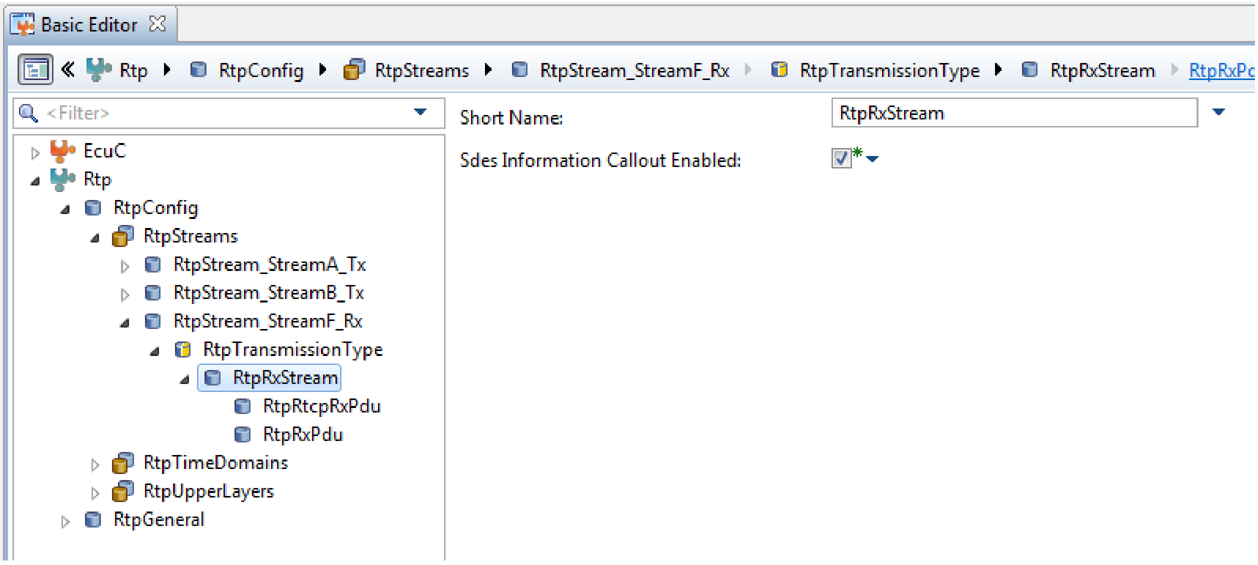
Rtcp Cycle Time [s]: 0.125

Stream Name: C3PO

Figure 6-3 Configuration of TX Stream

6.2.4 RX Stream Configuration

For each configured `RtpRxStream` the user can choose if `<UpperLayer>_RtcpSdesInformation()` callouts shall be triggered as described in Chapter 3.7. The callouts can be enabled with the `RtpSdesInformationCalloutEnabled` parameter. Figure 6-4 shows the configuration of each `RtpRxStream`.



Short Name: RtpRxStream

Sdes Information Callout Enabled: ☑

Figure 6-4 Configuration of RX Stream

7 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
AVB	Audio/Video Bridging
BSW	Basis Software
CSRC	Contributing Source
DET	Development Error Tracer
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
QoS	Quality of Service
RR	Receiver Report
RTCP	RTP Control Protocol
RTP	Real-Time Transport Protocol
SDES	Source Description
SR	Sender Report
SSRC	Synchronization Source
SWC	Software Component
SWS	Software Specification

Table 7-1 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com