MICROSAR I-PDU Multiplexer

Technical Reference

Version 1.2.0

Authors	Safiulla Shakir
Status	Released

1 Document Information

1.1 History

Author	Date	Version	Remarks
Safiulla Shakir	26.03.2009	1.0	Initial version
Safiulla Shakir	28.06.2010	1.1.0	Updated to support IPDUM system description 3.1.4 format
Safiulla Shakir	02.02.2011	1.2.0	ESCAN00046128

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR_SWS_IPDUM.pdf	1.2.1
[2]	AUTOSAR_SWS_IPDUM.doc	1.3.0
[3]	AUTOSAR_BasicSoftwareModules.pdf	1.0.0
[4]	AUTOSAR_SWS_IPDUM ASR3.2.pdf	1.3.0

Table 1-2 Reference documents

Please note
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.

Info
[2] is a draft version of AUTOSAR release 4.0.

Contents

Illustrations

Tables

2 Introduction

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

Supported AUTOSAR Release*:	**3.x**	
Supported Configuration Variants:	**pre-compile, link-time, post-build**	
Vendor ID:	**IPDUM_VENDOR_ID**	**30 decimal (= Vector-Informatik GmbH, according to HIS)**
Module ID:	**IPDUM_MODULE_ID**	**52 decimal (according to ref. [3])**

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

Multiplexing is a concept generally used to save CAN identifiers where an I-PDU with a CAN ID is used to carry different I-PDUs there by saving the CAN IDs for the I-PDUs it carries.

The implementation is based on the AUTOSAR IPDUM specifications [1] and [2]. It is assumed that the reader is familiar with this document and other relevant AUTOSAR related specifications.

I-PDU multiplexing means using the same PCI (Protocol Control Information) of a PDU (Protocol Data Unit) with more than one unique layouts of its SDU (Service Data Unit).
The SDU in a multiplexed I-PDU contains a selector field which is used to determine the layout of the SDU being multiplexed or de-multiplexed.
The AUTOSAR I-PDU multiplexer multiplexes a dynamic part along with or without a static part depending on whether a static part is configured to the multiplex I-PDU or not.
Each dynamic or static part is an independent signal-I-PDU in Com.

Info
Dynamic part or static part is the IPDUM vocabulary for an upper layer I-PDU which is to be multiplexed or de-multiplexed.

A given multiplex I-PDU can have 1 or no static part configured to it and it can multiplex 1 to 255 dynamic parts (layouts) depending on the selector field value along with the static part if configured.
A dynamic or static part may be further divided into sub parts. Each sub part is a bit field with the start and end bit positions. The dynamic part contains the selector field.
The selector field can be thought of as a signal in every dynamic part in Com whose value is set at configuration time and is never changed at runtime.

2.1 Architecture Overview

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

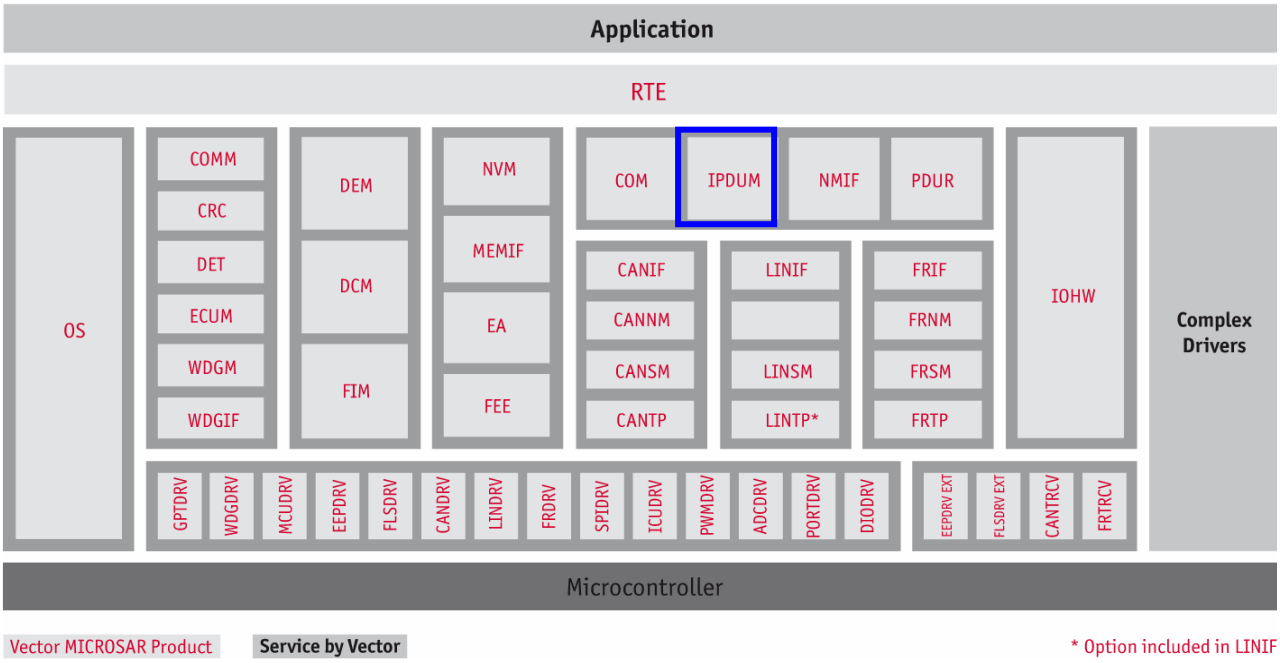


Figure 2-1 AUTOSAR architecture

The next figure shows the interfaces of the IPDUM to its adjacent module(s). These interfaces are described in chapter 5.

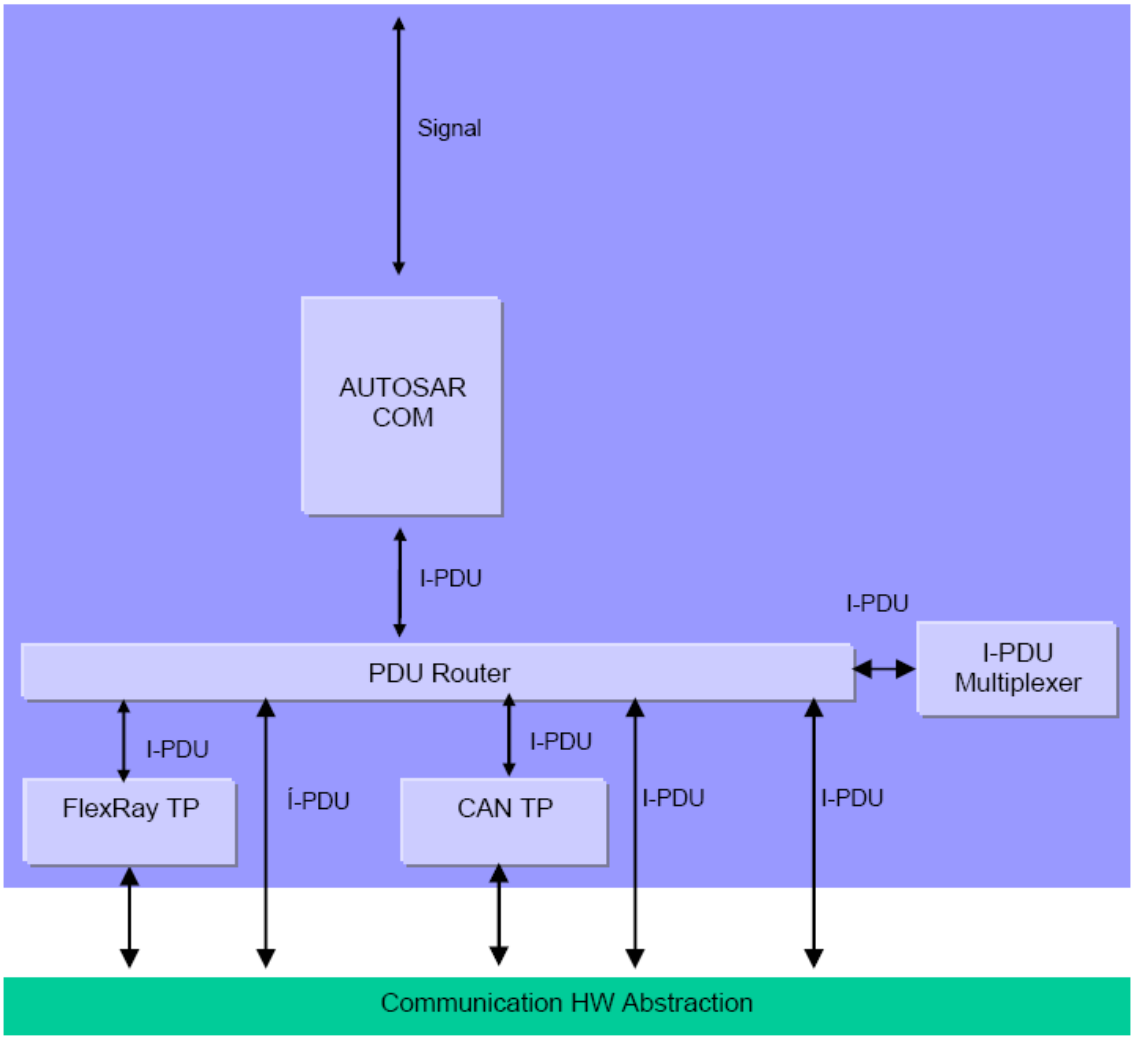


Figure 2-2 IPDUM Interfaces to adjacent modules

Normally the I-PDU multiplexer only interacts with the PduR. If required the I-PDU multiplexer can interface Com directly bypassing the PduR while indicating receptions and confirming transmissions to Com as demanded in [1].

The IPDUM also uses the interfaces to the BSW Scheduler and Det.

3 Functional Description

3.1 Features

The features listed in this chapter cover the complete functionality specified in [1].

The "supported" and "not supported" features are presented in the following two tables. For further information on features not supported see chapter 7.

The main purpose of the AUTOSAR BSW module IPDUM is to multiplex and de-multiplex IPDUs between upper (AUTOSAR Com) and the interface layers (CanIf, LinIf, FrIf).

The following features described in [1] are supported:

Supported Feature
Configuration
> Multiplexing of the static and dynamic parts and their transmission.
> Event base transmission triggering of the multiplex I-PDU(s).
> Confirming transmissions of the multiplexed I-PDUs to the BSW module AUTOSAR Com.
> Timeout handling while confirming transmissions.
> De-multiplexing and indicating receptions of the de-multiplexed dynamic parts and static part to the BSW module AUTOSAR Com.
> Static part configuration support in multiplex I-PDU(s).
> AUTOSAR EcuC format
> Initialization of buffers of the multiplex I-PDUs in case their transmission is triggered by interface layer BSW modules (relevant for FrIf, LinIf only)

Table 3-1 Supported SWS features

3.2 Initialization

The IPDUM is normally initialized by calling the API `IpduM_Init`. This is done by the EcuM. If this is not the case, an adequate and suitable solution has to be provided. During initialization the IPDUM initializes all the transmission multiplex I-PDU(s) buffer(s) and all runtime module relevant variables.

Info
If the BSW module Det is not used and the IPDUM is not initialized before its functionalities are used, the IPDUM's behavior is undefined. The IPDUM should be initialized before using its functionalities.

3.3 States

The IPDUM has the states: uninitialized and initialized.

3.4 Main Functions

IPDUM provides all functions described in [1]. The function `IpduM_MainFunction` should be called cyclically by the Basic Software Scheduler or a similar component. The `IpduM_MainFunction` primarily takes care of the timeout handling of the multiplex I-PDU(s) transmission confirmations.

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 `IPDUM_DEV_ERROR_DETECT == STD_ON`). The DET reporting can be enabled during pre-compile time using the tool GENy.

To do this set the check box against the parameter Development Error Detection as shown in Figure 3-1:

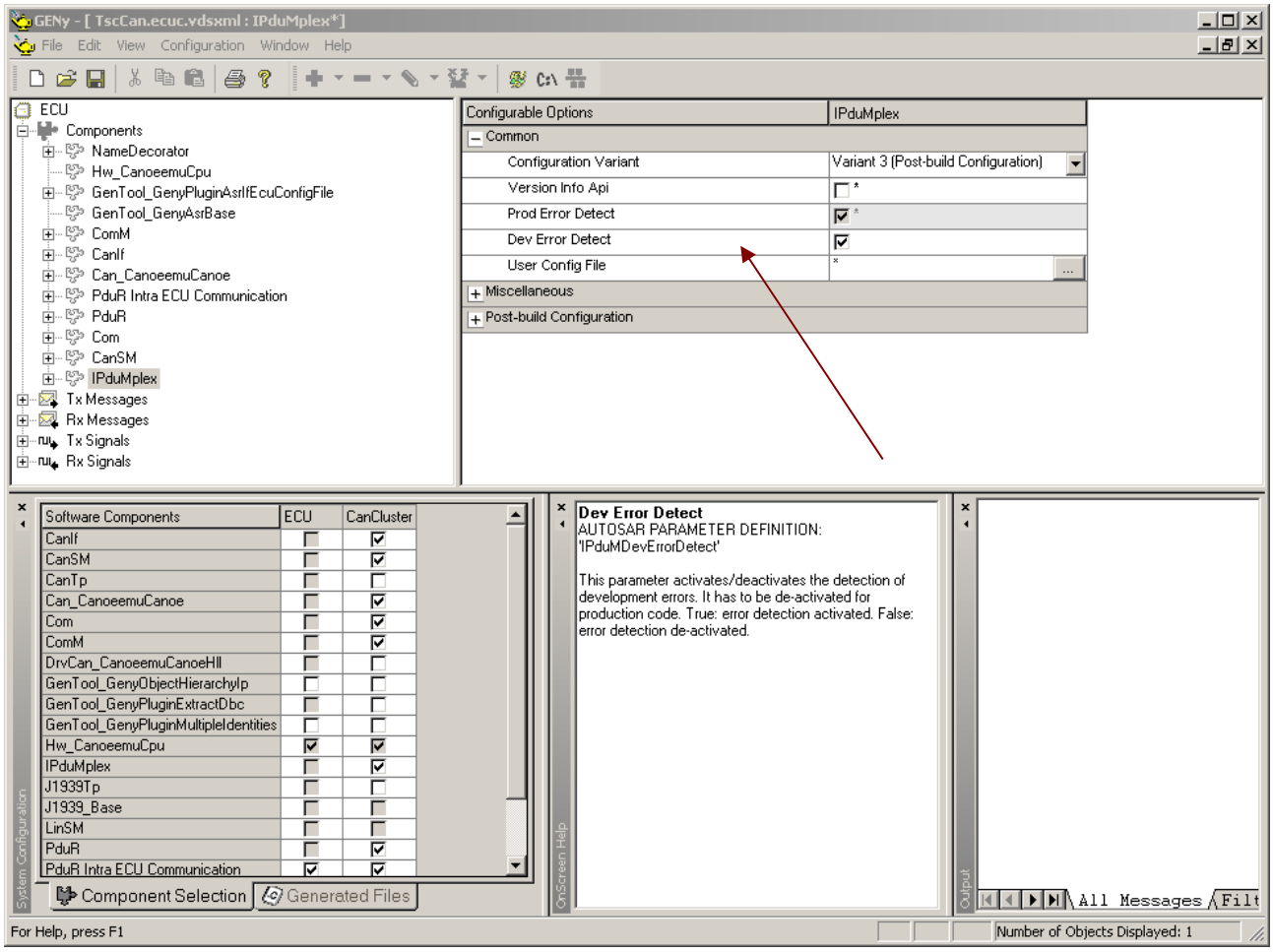


Figure 3-1 DET Activation

If another module is used for error reporting instead of DET, 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 IPDUM module ID is 52.

The reported services IDs identify the IPDUM services as specified by [1]. For the service IDs and their related service names see section 8.3 of [1].

The errors reported to DET are described in section 7.6 of [1]:

3.5.2 Production Error Reporting

No production error codes are currently defined for I-PDU multiplexer. For whatever reasons [1] demands the detection of production code errors should not be switched off. Hence as shown in the Figure 3-1 Dem is always on and cannot be de-activated.

4 Integration

This chapter gives information required for the integration of the IPDUM into an ECU application environment.

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
IpduM.c	Source file if ordered as source code.
IpduM.h	Header file if ordered as source code
IpduM.lib	Library file. The extension of the library may change depending on the used compiler

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool [GENy].

File Name	Description
IpduM_Cfg.h	Contains pre-compile switches and symbolic defines.
IpduM_Lcfg.c	Contains link time parameters.
IpduM_PBcfg.c	Contains post build parameters.

Table 4-2 Generated files

4.2 Include Structure

The I-PDU multiplexer's include structure is as illustrated in 5.4.2 in [1]. Apart from that IpduM.c also includes Com_Cbk.h due to the demanded optimization of interacting with BSW module Com directly bypassing the PDU-Router.

4.3 Compiler Abstraction and Memory Mapping

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

The following table contains the IPDUM memory sections, its compiler abstraction definitions and their relationship among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	IPDUM_CODE	IPDUM_PBCFG	IPDUM_VAR_INIT	IPDUM_VAR_NOINIT	IPDUM_PBCFG_ROOT	IPDUM_APPL_DATA
IPDUM_START_SEC_CODE IPDUM_STOP_SEC_CODE	■					
IPDUM_START_SEC_PBCFG IPDUM_STOP_SEC_PBCFG		■				
IPDUM_START_SEC_PBCFG_ROOT IPDUM_STOP_SEC_PBCFG_ROOT					■	
IPDUM_START_SEC_VAR_INIT_UNSPECIFIED IPDUM_STOP_SEC_VAR_INIT_UNSPECIFIED			■			
IPDUM_START_SEC_VAR_NOINIT_8BIT IPDUM_STOP_SEC_VAR_NOINIT_8BIT				■		
IPDUM_START_SEC_VAR_NOINIT_UNSPECIFIED IPDUM_STOP_SEC_VAR_NOINIT_UNSPECIFIED				■		

Table 4-3 Compiler abstraction and memory mapping

4.4 Critical Sections

The IPDUM can use the BSW Scheduler to handle its critical sections. It can also use the Vector solution VStdLib to handle its critical sections.

The critical sections are entered in the context of de-multiplexing, multiplexing and the IpduM_MainFunction().

4.4.1 BSW Scheduler

If the BSW Scheduler is used to handle critical sections, the IPDUM uses the following APIs for optimized runtime.

SchM_Enter_IpduM (IPDUM_EXCLUSIVE_AREA_0)

SchM_Exit_IpduM (IPDUM_EXCLUSIVE_AREA_0)

These are configured in the template `_SchM_IpduMS.h` of the BSW Scheduler. IPDUM_EXCLUSIVE_AREA_0 should be defined to SCHM_EA_SUSPENDALLINTERRUPTS.

4.4.2 Vector Standard Library

If the Vector Standard Library (VStdLib) is used to handle the critical section, the IPDUM uses the following API: `VStdSuspendAllInterrupts()` which suspends the interrupts globally and `VStdResumeAllInterrupts()` which resumes the interrupts globally.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Services provided by IPDUM

The IPDUM API consists of services, which are realized by function calls.

5.1.1 IpduM_InitMemory

Prototype	
`void IpduM_InitMemory (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
The function initializes variables, which cannot be initialized with the startup code.	
Particularities and Limitations	
■ The function is used by the application.	
Expected Caller Context	
■ The function must be called on task level.	

Table 5-1 IpduM_InitMemory

5.1.2 IpduM_Init

Prototype	
`void IpduM_Init (const IpduM_ConfigType* ConfigPtr)`	
Parameter	
`ConfigPtr`	Pointer to the IpduM configuration data, if IPDUM_CONFIG_VARIANT 3 is configured.
Return code	
void	none
Functional Description	
This service initializes all internal and module related variables of the IPDU Multiplexer.	
Particularities and Limitations	
■ The function is used by the ECU Manager	
Expected Caller Context	

■ The function must be called during the initialization phase of the stack or before the intended use of any of the IPDU Multiplexer's services.

Table 5-2 IpduM_Init

5.1.3 IpduM_RxIndication

Prototype	

`# if(IPDUM_USE_PDUINFOTYPE == STD_ON)`
`#  define IPDUM_RXIND_PARA  PduInfoPtr`
`#  define IPDUM_RXIND_TYPE  P2CONST(PduInfoType, AUTOMATIC, IPDUM_APPL_DATA)`
`# else`
`#  define IPDUM_RXIND_PARA  SduPtr`
`#  define IPDUM_RXIND_TYPE  P2CONST(uint8, AUTOMATIC, IPDUM_APPL_DATA)`
`# endif`
`void IpduM_RxIndication (PduIdType PdumRxPduId, IPDUM_RXIND_TYPE IPDUM_RXIND_PARA)`	

Parameter	
`SduPtr`	Contains the length (SduLength) for the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
`PduInfoPtr`	Payload information of the received I-PDU (pointer to data and data length). This Parameter is used, if IPDUM_USE_PDUINFOTYPE is defined to STD_ON.
`PdumRxPduId`	ID of received multiplexed I-PDU. Identifies the payload. Range: 0 to count of multiplex I-PDU IDs received –1
Return code	
void	none
Functional Description	
This service de-multiplexes the received multiplexed IPDU.	
Particularities and Limitations	
■ The function is called by the PDU-Router	
Expected Caller Context	
■ The function is called in an interrupt context or at the task level	

Table 5-3 IpduM_RxIndication

5.1.4 IpduM_Transmit

Prototype
`StdReturnType IpduM_Transmit (PduIdType PdumTxPduId, const PduInfoType* PduInfoPtr)`

Parameter	
PduInfoPtr	Pointer to the payload of the transmit request.
PdumTxPduId	ID of transmit request (static or dynamic part). Identifies the payload. Range: 0 to count of dynamic/static I-PDU ID/parts multiplexed by IPDU Multiplexer –1
Return code	
StdReturnType	E_OK: request accepted, E_NOT_OK: request rejected
Functional Description	
This service multiplexes the received dynamic/static part	
Particularities and Limitations	
■ The function is called by the PDU-Router	
Expected Caller Context	
■ The function is called at the task level	

Table 5-4 IpduM_Transmit

5.1.5 IpduM_TxConfirmation

Prototype	
void **IpduM_TxConfirmation** (PduIdType PdumTxPduId)	
Parameter	
PdumTxPduId	ID of transmitted multiplexed IPDU. Range: 0 to count of multiplex I-PDU IDs transmitted –1
Return code	
void	none
Functional Description	
This service confirms the transmissions of the dynamic and static parts contained in the multiplex IPDU whose transmission is confirmed.	
Particularities and Limitations	
■ The function is called by the PDU-Router	
Expected Caller Context	
■ The function is called in an interrupt context or at the task level	

Table 5-5 IpduM_TxConfirmation

5.1.6 IpduM_TriggerTransmit

Prototype	

<pre># if(IPDUM_USE_PDUINFOTYPE == STD_ON) 	
# define IPDUM_TT_PARA PduInfoPtr 	
# define IPDUM_TT_TYPE CONSTP2VAR(PduInfoType, AUTOMATIC, IPDUM_APPL_DATA) 	
# define IPDUM_TT_RETURN Std_ReturnType 	
# else 	
# define IPDUM_TT_PARA SduPtr 	
# define IPDUM_TT_TYPE P2VAR(uint8, AUTOMATIC, IPDUM_APPL_DATA) 	
# define IPDUM_TT_RETURN void 	
# endif 	
 	
Std_ReturnType **IpduM_TriggerTransmit** (PduIdType PdumTxPduId, IPDUM_TT_TYPE IPDUM_TT_PARA)</pre>	

Parameter	
PdumTxPduId	ID of transmission multiplexed IPDU. Range: 0 to count of multiplex I-PDU IDs transmitted –1
SduPtr	Pointer to the received I-PDU data. This Parameter is used, if IPDUM_USE_PDUINFOTYPE is defined to STD_OFF.
PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
Return code	
Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. SduLength has not been set.
Functional Description	
This service copies the transmission multiplexed IPDU to the provided buffer.	
Particularities and Limitations	
■ The function is called by the interface layers (LIN/FR)	
Expected Caller Context	
■ The function is called in an interrupt context or at the task level	

Table 5-6 IpduM_TriggerTransmit

5.1.7 IpduM_MainFunction

Prototype	
`void IpduM_MainFunction (void)`	
Parameter	
`void`	none
Return code	
`void`	none
Functional Description	
This service takes care of the timeout handling of the transmission confirmations.	
Particularities and Limitations	
■ The function is called cyclically by the BSW scheduler	
Expected Caller Context	
■ The function is called at the task level	

Table 5-7 IpduM_MainFunction

5.1.8 IpduM_GetVersionInfo

Prototype	
`void IpduM_GetVersionInfo (Std_VersionInfoType* versioninfo)`	
Parameter	
`Std_VersionInfoType`	Pointer to the structure to write the versioninfo
Return code	
`void`	none
Functional Description	
Provides the version information of the IPDU Multiplexer.	
Particularities and Limitations	
■ none	
Expected Caller Context	
■ The function can be called at any time at the application' discretion, interrupt or task level	

Table 5-8 IpduM_GetVersionInfo

5.2 Services used by IPDUM

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

Component	API
Det	Det_ReportError
PduR	PduR_IpduMTransmit
PduR	PduR_IpduMTxConfirmation
PduR	PduR_IpduMRxIndication
Com	Com_RxIndication
Com	Com_TxConfirmation
BSW Scheduler	SchM_Enter_IpduM
BSW Scheduler	SchM_Exit_IpduM

Table 5-9 Services used by the IPDUM

6 Configuration

Info
Currently the IPDUM implementation supports only an AUTOSAR EcuC.

The IPDUM can be configured using an EcuC file through the Vector configuration and generation tool GENy. A few of the parameters and their configurations have been described in the section 3.5.1 (see above). The following chapter provides a detailed description of the IPDUM EcuC parameters.

6.1 EcuC configuration with GENy

A detail description of all the IPDUM parameters is given in the following sections.

Info
Should the GENy support for IPDUM configuration be intended? Please ensure the basic software module definition file (IpduM_bswmd.xml) and the Il_AsrIpduM.dll are available in the GENy components folder

All of the IPDUM configuration parameters are imported from the EcuC into GENy. However some non AUTOSAR parameters (e.g IPduMUserConfigFile, IpduMVStdLibUsed, IPduMPBStartAddress, IPduMBufferSize, IPduMMaxRxBufferSize, IPduMMaxNumberOfTxRequests see section 6.2.1.22) may not be available by default in the EcuC. Such parameters will be exported to the EcuC file through GENy.

Only a few IPDUM parameters imported from the EcuC file may have to be configured in GENy such parameters are editable.

Caution
The IPDUM EcuC parameters which are not editable in GENy are directly derived from the systems description and should not be manually edited / configured by the user.

The IPDUM can be enabled in the component selection section of the GENy GUI as shown in the Figure 6-1.

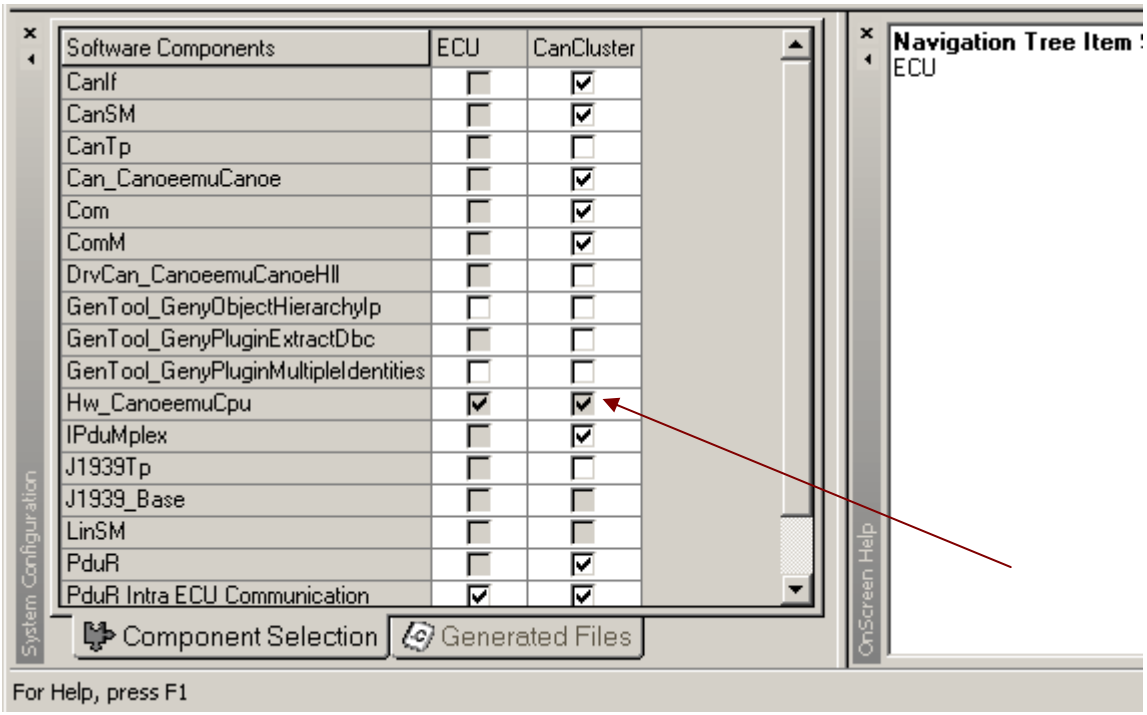


Figure 6-1 GENy component selection

6.1.1 General parameters

Figure 6-2 shows the activation of static parts support in the IPDUM. If the IPDUM EcuC multiplex I-PDUs are configured with static parts or static part support in expected in a post build configuration, the parameter IpduMStaticPartExists should be enabled in GENy at pre-compile time.

Caution
If this parameter is not enabled the IPDUM will not support static parts.

Likewise if the IPDUM version info is required at runtime, the parameter IPduMVersionInfoApi should be activated at pre-compile time.

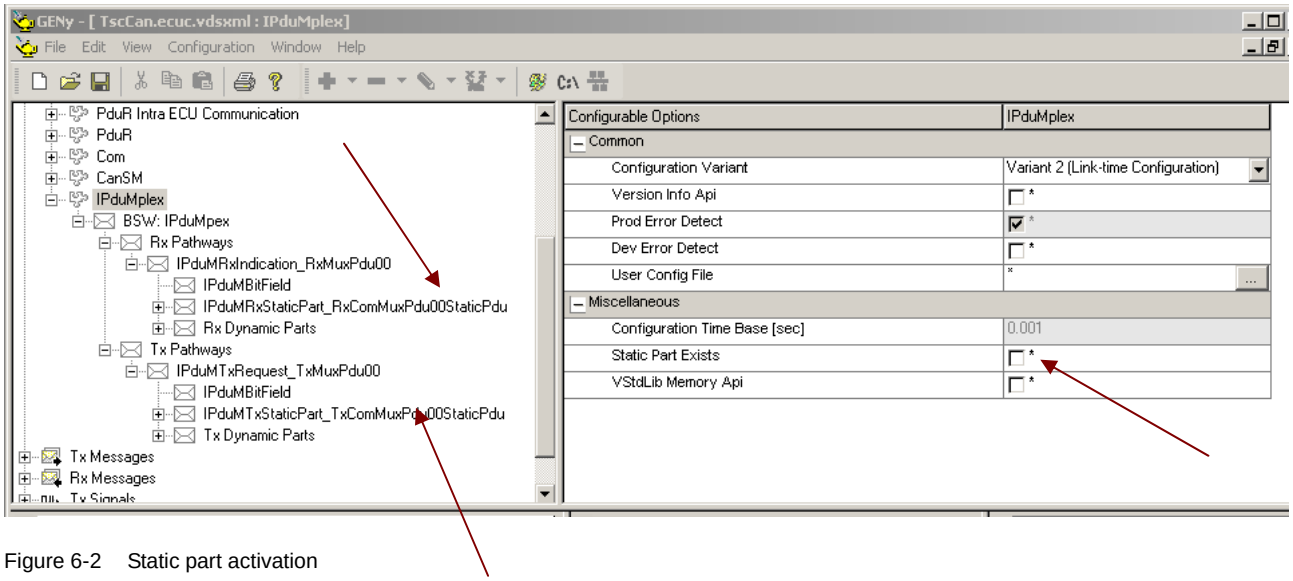


Figure 6-2 Static part activation

6.1.2 Link time and Post build configuration

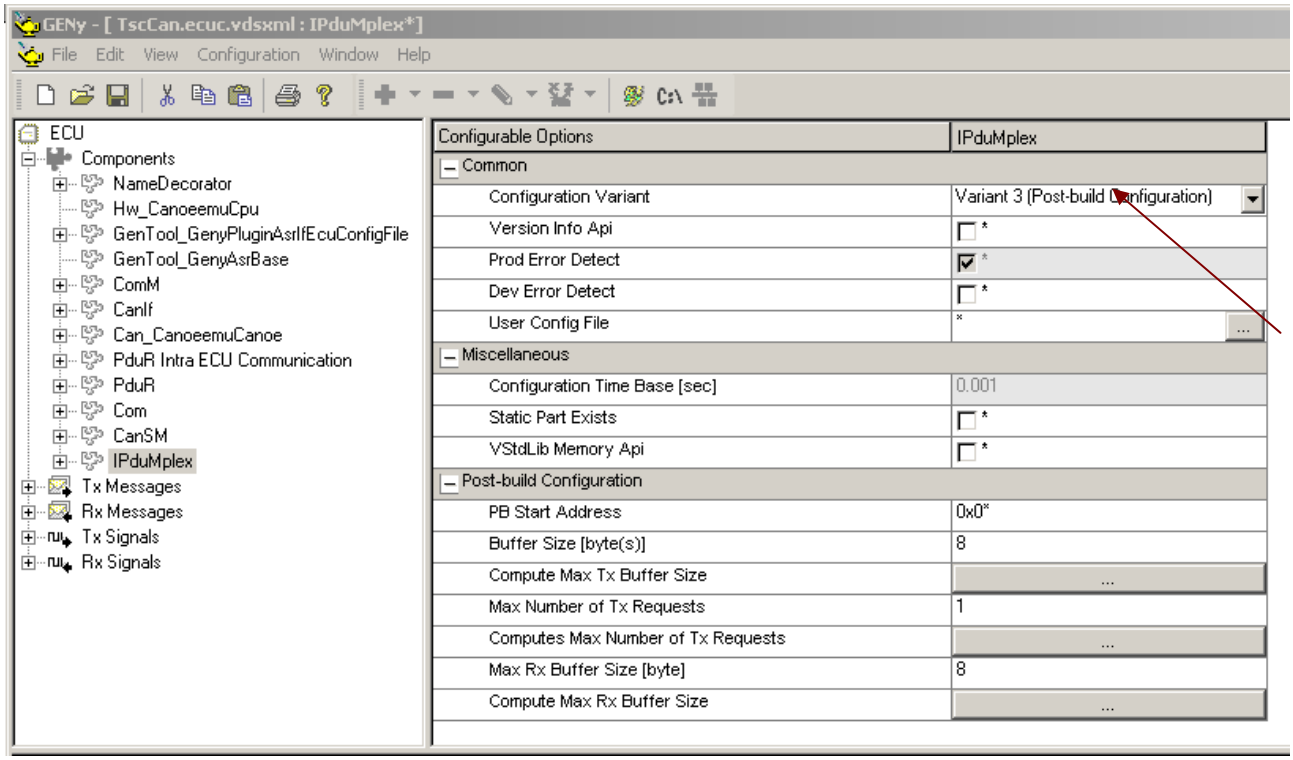


Figure 6-3 Link time and post build parameters

Figure 6-3 shows the link time and post build parameters. The desired configuration can be chosen using the configuration variant drop down menu.

- PB Start Address is the start address of the post build configuration structure of the I-PDU multiplexer module in the ROM. This address is required by the IPDUM post build configuration.
- The buffer size required by the transmitted multiplex I-PDU(s) (Tx Pathways) in the IPDUM configuration can be computed by clicking the button against the field “Compute Max Tx Buffer Size”. If more Tx Pathways are anticipated in a post build configuration the size can be manually edited
- The count of the transmitted multiplex I-PDU(s) (Tx Pathways) in the IPDUM configuration can be computed by clicking the button against the field “Max Number of Tx Requests”. If more Tx Pathways are anticipated in a post build configuration the count can be manually edited
- The buffer size required by the received multiplex I-PDU(s) (Rx Pathways) in the IPDUM configuration can be computed by clicking the button against the field “Compute Max Rx Buffer Size”. If an increment in length of the largest received multiplex is anticipated in a post build configuration the size can be manually edited

6.1.3 Multiplexing and De-multiplexing

The IPDUM multiplexes or de-multiplexes different dynamic parts associated to a multiplex I-PDU along with the configured static part if any. The multiplexing and de-multiplexing is executed based on the configuration derived from the system description and therefore should not be edited.

Figure 6-4 shows the configuration of a dynamic part to be multiplexed. The start bit and end bit give the the bit field positions in the dynamic part. The destination bit is the bit position in the multiplex I-PDU to which the bit field has to be copied. The IPDUM multiplexes accordingly and triggers the transmission of the multiplex I-PDU(s) (Tx Pathway(s)).

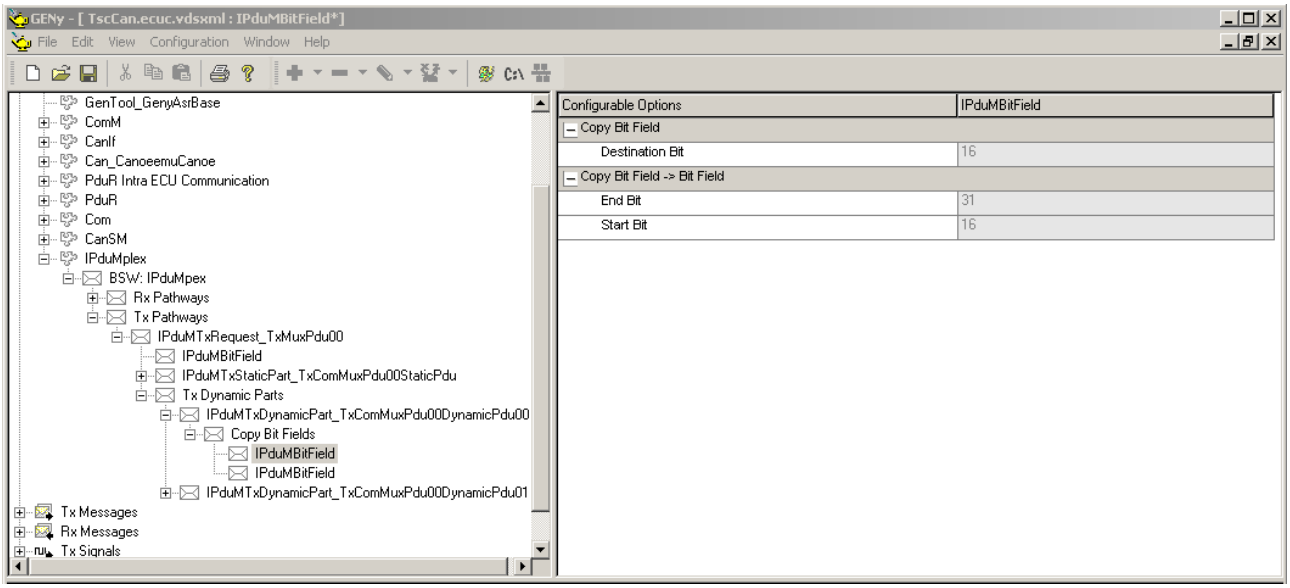


Figure 6-4 Copy bit field configuration of a dynamic part

Figure 6-5 shows the configuration of a static part to be de-multiplexed. The start bit and end bit give the bit field positions in the multiplexed I-PDU. The destination bit is the bit position in the static part to which the bit field has to be copied. The IPDUM de-multiplexes accordingly and indicates the reception of the static part to Com.

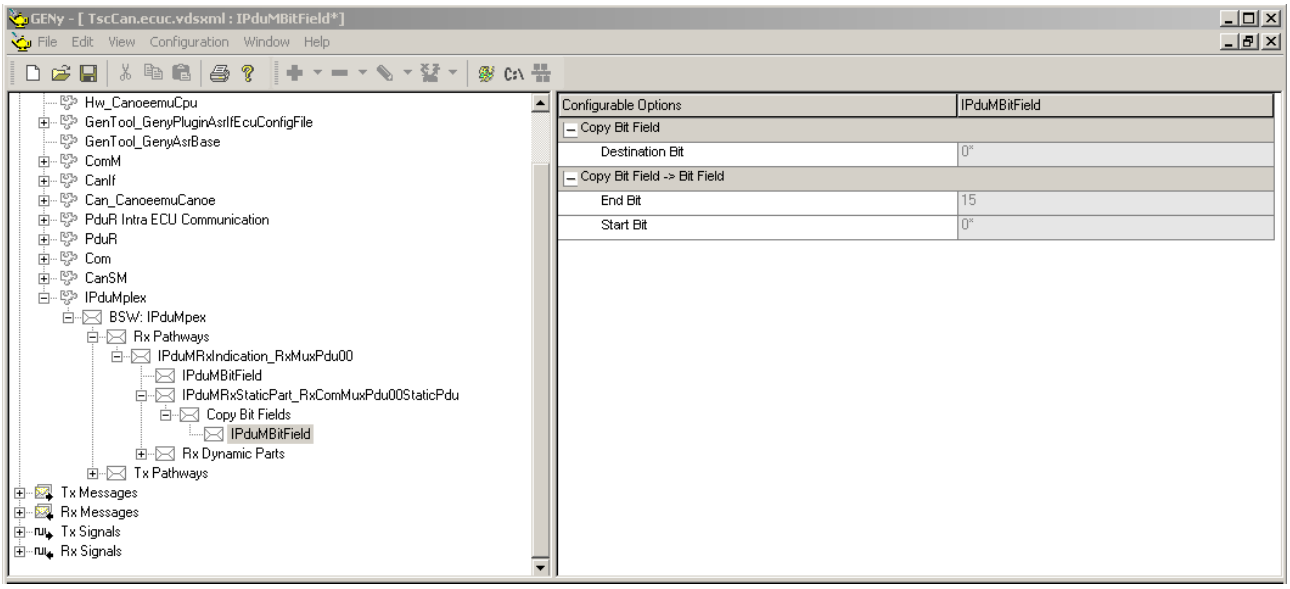


Figure 6-5 Copy bit field configuration of a static part

While de-multiplexing a dynamic part contained in a received multiplex I-PDU. The IPDUM chooses the associated dynamic part layout based on the parameter “Rx Selector Value” which is derived from the system description and should not be edited.

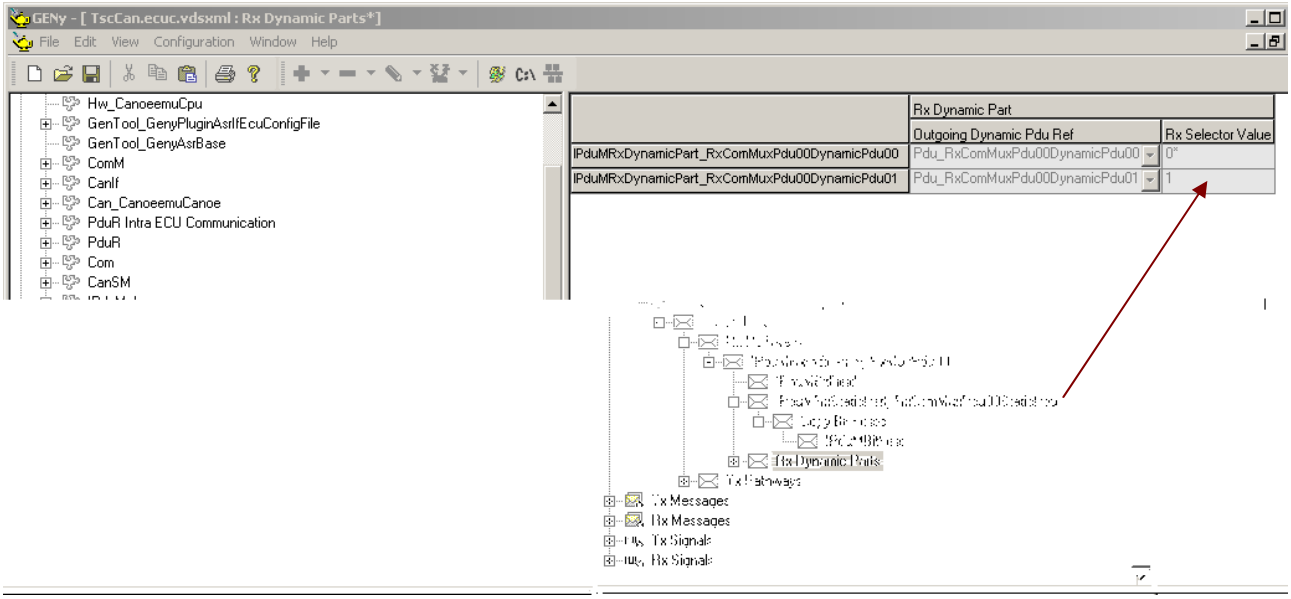


Figure 6-6 Dynamic part layout selection

Caution
It should be noted here that:

- The parameters Start Bit, End bit and Destination Bit are derived from the system description.
- The AUTOSAR system template versions prior to 3.1.4 do not support the configuration of more than 1 sub parts (IPduMBitField) per static or dynamic part(s).

6.1.4 Tx Pathway

All parameters for a multiplex I-PDU (TxPathway/TxRequest) configuration are shown in the Figure 6-7.

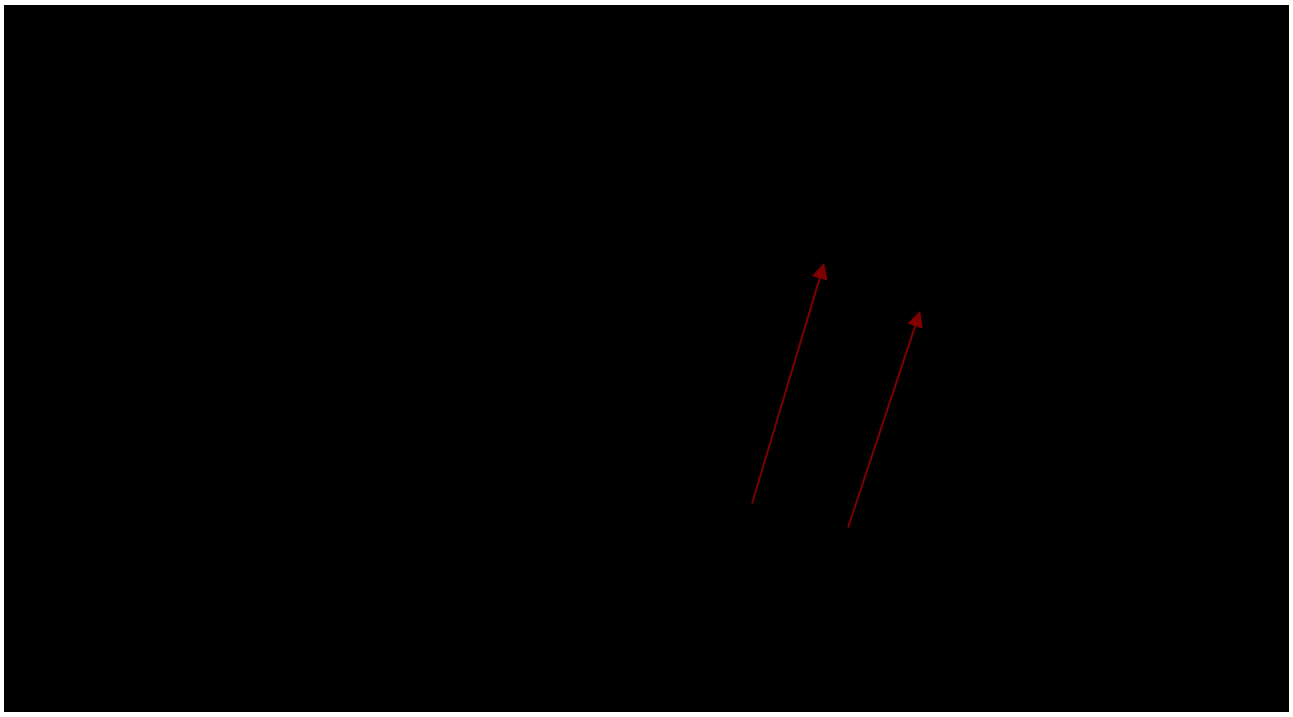


Figure 6-7 Tx Pathway configuration parameters

- All the parameters except “Tx Confirmation Timeout” are derived from the system description and hence should not be edited. While configuring the value for this parameter the user has to consider the “Tx Mode” of the associated dynamic and / or static parts in Com. If configured the IPDUM performs a time out handling for every multiplex I-PDU (Tx Pathway) it transmits. No transmission requests of the associated static or dynamic parts are accepted for a transmitted multiplex I-PDU as long as the transmission

confirmation is received or the configured transmission confirmation timeout hasn't elapsed.

- In the system description with format 3.0.1 along with above parameter, another parameter "Tx Trigger Mode" also seen in the figure above can be configured using the drop down. The parameter "Tx Trigger Mode" is the event which triggers the transmission of a multiplex I-PDU by IPDUM. The transmission of a multiplex IPDU can be configured following the below events:
  - DYNAMIC_PART_TRIGGER: Transmission of the multiplex I-PDU is triggered by the IPDUM when Com requests the transmission of a multiplexed dynamic part.
  - STATIC_OR_DYNAMIC_PART_TRIGGER: Transmission of the multiplex I-PDU is triggered by the IPDUM when Com requests the transmission of a multiplexed dynamic part or the configured static part.
  - STATIC_ PART_TRIGGER: Transmission of the multiplex I-PDU is triggered by the IPDUM when Com requests the transmission of the configured static part.
  - NONE: Transmission of the multiplex I-PDU is never triggered by the IPDUM.

Info
In case of the last event the static or dynamic part is copied to its associated buffer but no transmission is triggered.

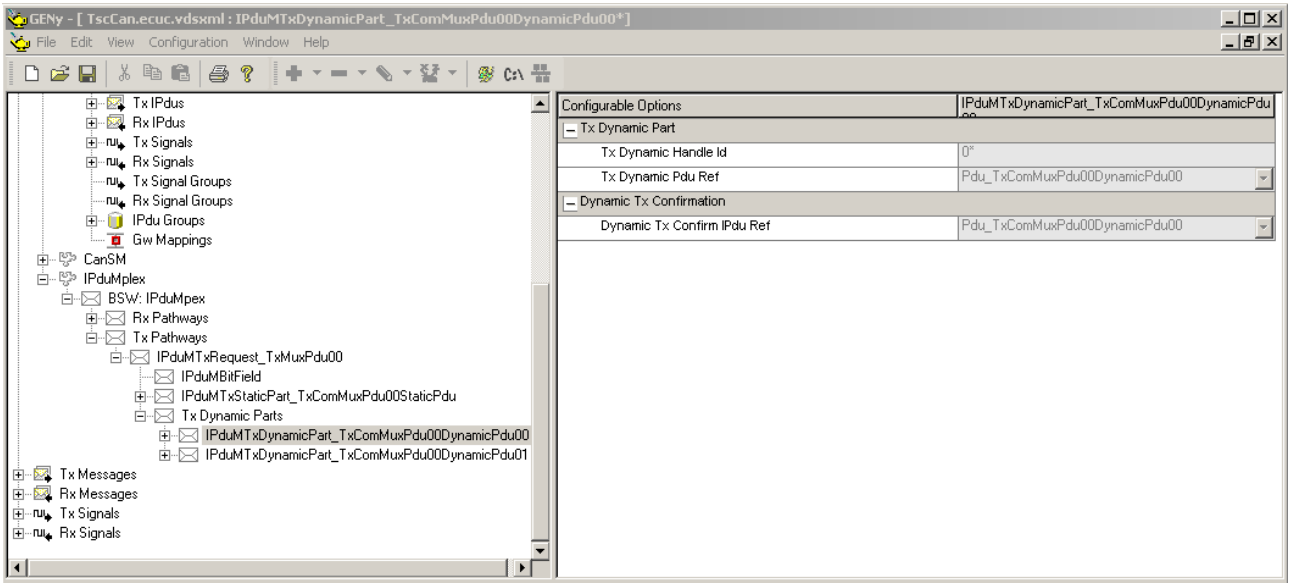
The trigger event NONE is useful in case the transmission of a multiplex I-PDU is triggered by a lower interface layer (FrIf, LinIf) layer. The IPDU Multiplexer provides the API `IpduM_TriggerTransmit` for such a scenario.

6.1.5 Transmission Confirmations

The IPDUM confirms the transmission of every dynamic part it multiplexes through a multiplex I-PDU and a static part if configured to the multiplex I-PDU to Com.

Figure 6-8 shows the transmission confirmation configurations of different dynamic parts multiplexed by a multiplex I-PDU (Tx Pathway).

Info
The IPDUM confirms the transmission of all the dynamic parts multiplexed by a multiplex I-PDU and also confirms the transmission of the static part if configured to the multiplex I-PDU.


GENy - [TscCan.ecuc.vdsxml : IPduMTxDynamicPart_TxComMuxPdu00DynamicPdu00*]
File Edit View Configuration Window Help
Tx IPdus
Rx IPdus
Tx Signals
Rx Signals
Tx Signal Groups
Rx Signal Groups
IPdu Groups
Gw Mappings
CanSM
IPduMplex
BSW: IPduMpex
Rx Pathways
Tx Pathways
IPduMTxRequest_TxMuxPdu00
IPduMBitField
IPduMTxStaticPart_TxComMuxPdu00StaticPdu
Tx Dynamic Parts
IPduMTxDynamicPart_TxComMuxPdu00DynamicPdu00
IPduMTxDynamicPart_TxComMuxPdu00DynamicPdu01
Tx Messages
Rx Messages
Tx Signals
Rx Signals
Configurable Options
IPduMTxDynamicPart_TxComMuxPdu00DynamicPdu
Tx Dynamic Part
Tx Dynamic Handle Id
0*
Tx Dynamic Pdu Ref
Pdu_TxComMuxPdu00DynamicPdu00
Dynamic Tx Confirmation
Dynamic Tx Confirm IPdu Ref
Pdu_TxComMuxPdu00DynamicPdu00

6.2 ECU Configuration parameters:

Container Name	IPduMplex
Multiplicity	0...1
Description	Configuration of the IPduMplex (IPdu Multiplexer) module.

6.2.1.1 IPduMConfig

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMConfig	1...1			This container contains the sub containers of the IPduMplex module. The IPduMTxPathway subcontainer includes information about sent I-PDUs. The IPduMRxPathway includes information about received I-PDUs. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.

Sub Container	Multiplicity
IPduMRxPathway	0...*
IPduMTxPathway	0...*

6.2.1.2 IPduMRxPathway

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMRxPathway	0...*			Contains the configuration parameters of the I-PDUs received by the IPduM module.

Sub Container	Multiplicity
IPduMRxIndication	1...1

6.2.1.3 IPduMRxIndication

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMRxIndication	1...1			Contains the configuration for incoming RxIndication calls.

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMRxHandleId	1...1	INTEGER		This is the I-PDU ID of the incoming I-PDU. If an incoming RxIndication's I-PDU ID matches this value then it is unpacked according to the specification in this container.
IPduMRxIndicationPduRef	1...1	REFERENCE		Reference to the received Pdu representation in the ECU Configuration Description exchange file.

Sub Container	Multiplicity
IPduMBitField	1...1
IPduMRxDynamicPart	1...*
IPduMRxStaticPart	0...1

6.2.1.4 IPduMBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMBitField	1...1			This is used to specify a contiguous range of bits within an I-PDU. The range is inclusive.
IPduMEndBit	1...1	INTEGER		Bit position in an I-PDU of the end of the bit field. Value must fit inside the I-PDU. Value must be the same as or higher than Ipdum_StartBit.
IPduMStartBit	1...1	INTEGER		Bit position in an I-PDU of the start of the bit field. Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit.

6.2.1.5 IPduMRxDynamicPart

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMRxDynamicPart	1...*			This container contains the configuration for the dynamic part of incoming RxIndication calls. When an received I-PDU's selector field matches the IPduM_Selector_Value the I-PDU is unpacked according to the values in the IPduMCopyBitfield and then the new I-PDU constructed and sent out with the I-PDU ID referenced by IPduMOutgoingDyamicPduRef.
IPduMRxSelectorValue	1...1	INTEGER		This is the selector value in the received IPDU

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMOutgoingDynamicPduRef	1...1	REFERENCE		When the new I-PDU is sent out it is sent with this I-PDU ID. Reference to the sent PDU representation in the ECU Configuration Description exchange file.

Sub Container	Multiplicity
IPduMCopyBitField	1...*

6.2.1.6 IPduMCopyBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMCopyBitField	1...*			Specifies the source bit fields and the destination bit position, so that the bits in the source can be copied to the bits in the destination. Within one I-PDU multiple instances of this container are used to specify the bit fields in that I-PDU. Adjacent bit fields could be merged in order to reduce the number of instances of this container.
IPduMDestinationBit	1...1	INTEGER		Bit position in an I-PDU of the start of the destination bit field for the copy. The resulting destination field must fit inside the I-PDU.

Sub Container	Multiplicity
IPduMBitField	1...1

6.2.1.7 IPduMBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMBitField	1...1			This is used to specify a contiguous range of bits within an I-PDU. The range is inclusive.
IPduMEndBit	1...1	INTEGER		Bit position in an I-PDU of the end of the bit field. Value must fit inside the I-PDU. Value must be the same as or higher than Ipdum_StartBit.

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMStartBit	1...1	INTEGER		Bit position in an I-PDU of the start of the bit field. Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit.

6.2.1.8 IPduMRxStaticPart

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMRxStaticPart	0...1			This container contains the information on how to unpack the static part of an incoming I-PDU.
IPduMOutgoingStaticPduRef	1...1	REFERENCE		When the new I-PDU is sent out it is sent with this I-PDU ID. Reference to the sent Pdu representation in the ECU Configuration Description exchange file.

Sub Container	Multiplicity
IPduMCopyBitField	1...*

6.2.1.9 IPduMCopyBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMCopyBitField	1...*			Specifies the source bit fields and the destination bit position, so that the bits in the source can be copied to the bits in the destination. Within one I-PDU multiple instances of this container are used to specify the bit fields in that I-PDU. Adjacent bit fields could be merged in order to reduce the number of instances of this container.
IPduMDestinationBit	1...1	INTEGER		Bit position in an I-PDU of the start of the destination bit field for the copy. The resulting destination field must fit inside the I-PDU.

Sub Container	Multiplicity
IPduMBitField	1...1

6.2.1.10 IPduMBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMBitField	1...1			This is used to specify a contiguous range of bits within an I-PDU. The range is inclusive.
IPduMEndBit	1...1	INTEGER		Bit position in an I-PDU of the end of the bit field. Value must fit inside the I-PDU. Value must be the same as or higher than Ipdum_StartBit.
IPduMStartBit	1...1	INTEGER		Bit position in an I-PDU of the start of the bit field. Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit.

6.2.1.11 IPduMTxPathway

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMTxPathway	0...*			Contains the configuration parameters transmitted I-PDUs by the IPduM module.

Sub Container	Multiplicity
IPduMTxConfirmation	0...1
IPduMTxRequest	1...1

6.2.1.12 IPduMTxConfirmation

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMTxConfirmation	0...1			A transmit request can be confirmed by the lower layer. This container is used to generate the associated confirmations for the static and dynamic parts of a multiplexed I-PDU. When an I-PDU is transmitted by the IPduM, the selector field value in that PDU needs to be stored in the IPduM so that the confirmation for the correct dynamic part can be generated. This is state internal to the IPduM at run-time. For the purposes of this container and IPduMDynamicTxConfirmation this stored state is called Stored_Selector.

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMStaticTxConfirmationIPduRef	0...1	REFERENCE		This references the I-PDU to use in the TxConfirmation for the static part. This entity does not appear if there is no static part.

Sub Container	Multiplicity
IPduMDynamicTxConfirmation	1...*

6.2.1.13 IPduMDynamicTxConfirmation

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMDynamicTxConfirmation	1...*			The dynamic part of an I-PDU can have more than one I-PDU IDs for confirmations. The correct I-PDU ID for the confirmation is found from the selector field value of a previously transmitted I-PDU. It is assumed that this selector field is stored in some internal value called Stored_Selector. When a transmit confirmation is received the Stored_Selector is used to select an instance of IPduMDynamicTxConfirmation by matching the Stored_Selector with the IPduMSelectorValue.
IPduMSelectorValue	1...1	INTEGER		When the selector field of the confirmed I-PDU matches the value in here then generate a TxConfirmation for the I-PDU referenced by IPduMDynamicTxConfirmIPduRef.
IPduMDynamicTxConfirmIPduRef	1...1	REFERENCE		This is the I-PDU ID to use in the outgoing confirmation (confirmation for the COM I-PDU) when an incoming confirmation (for an IPduM I-PDU) is received and matches the stored Stored_Selector.

6.2.1.14 IPduMTxRequest

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMTxRequest	1...1			This is used to specify the configuration for Transmit requests. There will one instance of this container for each I-PDU that can be requested for transmission (the outgoing I-PDUs) by the IPduM.

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMIPduUnusedAreasDefault	0...1	INTEGER		IPduM module fills not used areas of an I-PDU with this bit-pattern If this attribute is omitted the IPduM module does not fill the I-PDU.
IPduMInitialSelectorValue	1...1	INTEGER		This value is used by the initialization function to set the initial value of the selector field.
IPduMSize	1...1	INTEGER		The size of the I-PDU in bytes. The maximum size is limited by the underlying communication interface. 0-8 for CAN and LIN 0-254 for FlexRay
IPduMTxConfirmationTimeout	0...1	FLOAT		This timeout (in seconds) defines the timeout period for monitoring the reception of the TxConfirmation. It is not used when an I-PDU is requested using the trigger transmit API.
IPduMTxTriggerMode	1...1	ENUMERATION		Sets the transmission trigger event of the multiplexed I-PDU, send immediately or at a later point in time.
IPduMOutgoingPduRef	1...1	REFERENCE		Reference to the PDU defining the outgoing I-PDU. When the outgoing I-PDU is sent this is the I-PDU ID to give it. It is the IPduM I-PDU ID of the assembled I-PDU.

Sub Container	Multiplicity
IPduMBitField	1...1
IPduMTxDynamicPart	1...*
IPduMTxStaticPart	0...1

6.2.1.15 IPduMBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMBitField	1...1			This is used to specify a contiguous range of bits within an I-PDU. The range is inclusive.
IPduMEndBit	1...1	INTEGER		Bit position in an I-PDU of the end of the bit field. Value must fit inside the I-PDU. Value must be the same as or higher than Ipdum_StartBit.
IPduMStartBit	1...1	INTEGER		Bit position in an I-PDU of the start of the bit field. Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit.

6.2.1.16 IPduMTxDynamicPart

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMTxDynamicPart	1...*			Configuration parameters for an instance of a TxRequest call into the IPduM. When a Tx Request with the IPduMTxDynamicHandleId is received by the IPduM, the bit fields in the incoming I-PDU are packed into the outgoing I-PDU buffer and then the send mode is honored. This container is used by the dynamic part of a TxRequest configuration. Therefore, for each outgoing I-PDU there will be one instance of this container for the dynamic part.
IPduMTxDynamicHandleId	1...1	INTEGER		This is an incoming handle id. When the handle of an incoming Tx Request matches this, the bits fields (see Ipdum_CopyBitField) are copied and the IpduMTxTriggerMode is honored.
IPduMTxDynamicPduRef	1...1	REFERENCE		Reference to the Pdu representation in the ECU

Sub Container	Multiplicity
IPduMBitField	1...1

6.2.1.18 IPduMBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMBitField	1...1			This is used to specify a contiguous range of bits within an I-PDU. The range is inclusive.
IPduMEndBit	1...1	INTEGER		Bit position in an I-PDU of the end of the bit field. Value must fit inside the I-PDU. Value must be the same as or higher than Ipdum_StartBit.
IPduMStartBit	1...1	INTEGER		Bit position in an I-PDU of the start of the bit field. Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit.

6.2.1.19 IPduMTxStaticPart

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMTxStaticPart	0...1			Configuration parameters for an instance of a Tx_Request call into the IPduM. When a Tx Request with the IPduMTxStaticHandleId is received by the IPduM, the bit fields in the incoming I-PDU are packed into the outgoing I-PDU buffer and then the send mode is honored. This container is used for the static part of a TxRequest configuration. Therefore, for each outgoing I-PDU there will be one instance of this container for the static part if it exists.
IPduMTxStaticHandleId	1...1	INTEGER		This is an incoming handle id. When the handle of an incoming Tx Request matches this, the bits fields (see IPduMCopyBitField) are copied and the IPduMTxTriggerMode is honored.
IPduMTxStaticPduRef	1...1	REFERENCE		Reference to the Pdu representation in the ECU Configuration Description exchange file to be transmitted.

Sub Container

6.2.1.20 IPduMCopyBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMCopyBitField	1...*			Specifies the source bit fields and the destination bit position, so that the bits in the source can be copied to the bits in the destination. Within one I-PDU multiple instances of this container are used to specify the bit fields in that I-PDU. Adjacent bit fields could be merged in order to reduce the number of instances of this container.
IPduMDestinationBit	1...1	INTEGER		Bit position in an I-PDU of the start of the destination bit field for the copy. The resulting destination field must fit inside the I-PDU.

Sub Container	Multiplicity
IPduMBitField	1...1

6.2.1.21 IPduMBitField

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMBitField	1...1			This is used to specify a contiguous range of bits within an I-PDU. The range is inclusive.
IPduMEndBit	1...1	INTEGER		Bit position in an I-PDU of the end of the bit field. Value must fit inside the I-PDU. Value must be the same as or higher than Ipdum_StartBit.
IPduMStartBit	1...1	INTEGER		Bit position in an I-PDU of the start of the bit field. Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit.

6.2.1.22 IPduMGeneral

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMGeneral	1...1			Contains the general configuration parameters of IPduMplex.
IPduMConfiguration TimeBase	1...1	FLOAT		The period between successive ticks of AUTOSAR COM in seconds.

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMDevErrorDetect	1...1	BOOLEAN		Active/Deactivate the detection of development errors, for production code this parameter has to be False. True: error detection activated False: error detection deactivated
IPduMStaticPartExists	1...1	BOOLEAN		This is to allow optimizations in the case the IPduM will never be used with a static part. Note that this is a pre-compile option. If this is set to False then it will not be possible to add static parts after compilation. True: A static part may exist. False: A static part will never exist.
IPduMVersionInfoApi	1...1	BOOLEAN		Active/Deactivate the version information API. true: version information activated false: version information deactivated
IPduMUserConfigFile	0...1	STRING		Configure a file reference to a user defined configuration file. The user defined configuration file is included at the end of the generated file IpduM_Cfg.h and can be used to extend or overwrite definitions.
IPduMPBStartAddress	1...1	INTEGER	0	This is the start address of the memory block in ROM where the post build configuration data is located.
IPduMBufferSize	1...1	INTEGER	0	As multiplexed Pdus can be added at post-build time, the buffer size (RAM) must be configured at link-time. The size of the buffer depends on the current IPduM configuration and on the scalability that should be available at post-build time.
IPduMMaxRxBufferSize	1...1	INTEGER	0	As multiplexed Pdus can be added at post-build time, the buffer size (RAM) must be configured at link-time. The size of the reception buffer depends on the current largest multiplex IPDU received by the IPduM and on the scalability that should be available at post-build time. (i.e if the size of the largest reception multiplex IPdu may change in future the buffer can be reserved in accordance to the expected size)
IPduMMaxNumberOfTxRequests	1...1	INTEGER	0	As multiplexed Pdus can be added at post-build time, the buffer size (RAM) must be configured at link-time. The size of the buffer depends on the number of multiplex IPdus transmitted in the current IPduM configuration and on the scalability that should be available at post-build time
IPduMUsePduInfoType	0…1	BOOLEAN	false	If this feature is active, PduInfoPtr instead of SduPtr is used for the APIs IpduM_RxIndication and IpduM_TriggerTransmit.

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMDynamicDlcSupport	0…1	BOOLEAN	false	If this feature is activ, the received data length (PduInfoPtr->SduLength) is checked by the function IpduM_RxIndication and set by function IpduM_TriggerTransmit only completely received static or dynamic parts are processed.

6.2.1.23 IPduMplexCh

Attribute Name	Multiplicity	Value Type	Default Value	Description
IPduMplexCh	0...1			
ComMChannelRef	0...1	REFERENCE		This is needed for GENy module activation

7 AUTOSAR Standard Compliance

7.1 Deviations

There are no deviations from the standard AUTOSAR requirements in the current implementation.

7.2 Additions/ Extensions

The current implementation incorporates 2 requirements (IPDUM155, IPDUM156) of AUTOSAR R4.0.

7.3 Limitations

N.A

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
GENy	Vector Configuration and Generation tool

Table 8-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	AUTOSAR BSW module Diagnostic Event Manager
DET	AUTOSAR BSW module Development Error Tracer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
SWS	Software Specification
Com	AUTOSAR BSW module Com
PduR	AUTOSAR BSW module Pdu-Router
IPDUM / IpduM	AUTOSAR BSW module I-PDU Multiplexer
EcuM	AUTOSAR BSW module ECU Manager

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector-informatik.com