VECTOR

MICROSAR IP Base

Technical Reference

IP Base Module

Version 1.02.02

Authors	Alex Lunkenheimer
Status	Released

Document Information

History

Author	Date	Version	Remarks
Alex Lunkenheimer	2011-05-20	1.0	Creation of the document
Alex Lunkenheimer	2011-12-05	1.1	Sock sub component added
Alex Lunkenheimer	2011-12-05	1.01.01	Released
Alex Lunkenheimer	2013-03-12	1.01.02	Update
Alex Lunkenheimer	2014-01-03	1.01.03	- New API IpBase_CalcTcpIpChecksum32 - Review integration
Alex Lunkenheimer	2014-01-03	1.01.04	- New API IpBase_CalcTcpIpChecksumAdd - Review integration
Alex Lunkenheimer	2014-02-07	1.01.05	- New API IpBase_CalcTcpIpChecksumAdd replaces IpBase_CalcTcpIpChecksum32
Alex Lunkenheimer	2014-02-07	1.01.06	- AUTOSAR version dependent architecture in 2.1 Architecture Overview
Alex Lunkenheimer	2015-02-27	1.01.07	- Adapted struct IpBase_SockAddrIn6Type and define IPBASE_AF_INET6
Alex Lunkenheimer	2015-03-02	1.02.00	- IpBase_CopySmallData introduced
Alex Lunkenheimer	2015-05-06	1.02.01	- IpBase_Copy as macro
Alex Lunkenheimer	2015-05-06	1.02.02	- Review integration

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_DET.pdf	2.2.1
[2]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	2.2.0
[3]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0

Scope of the Document

This technical reference describes the general use of the IpBase base 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
01.00.xx	Initial component version
01.01.xx	Extension by string length
01.02.xx	Data types adapted, ASN.1 / BER decoder added, Bugfixing
02.00.xx	Adapted struct IpBase_SockAddrIn6Type and define IPBASE_AF_INET6
02.01.xx	IpBase_Copy as macro from VStdLib (performance improvement)

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release:	not relevant	
Supported Configuration Variants:	not relevant	
Vendor ID:	IpBase_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	IpBase_MODULE_ID	255 decimal (according to ref. [4])

The IpBase component provides general functions used within MICROSAR IP. Its functionality covers copy, buffer and string handling as well as type definitions.

2.1 Architecture Overview

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

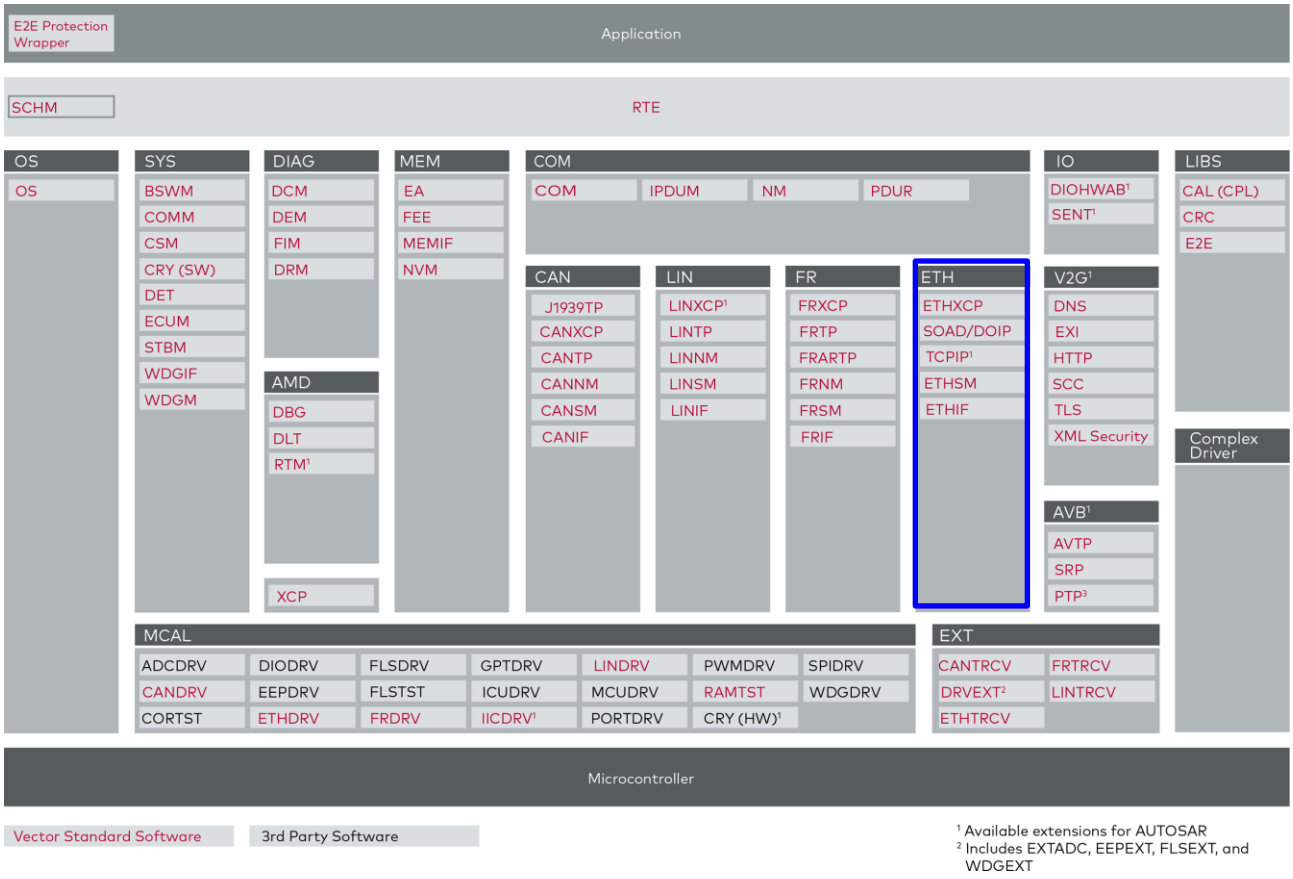

Figure 2-1 AUTOSAR 4.x Architecture Overview

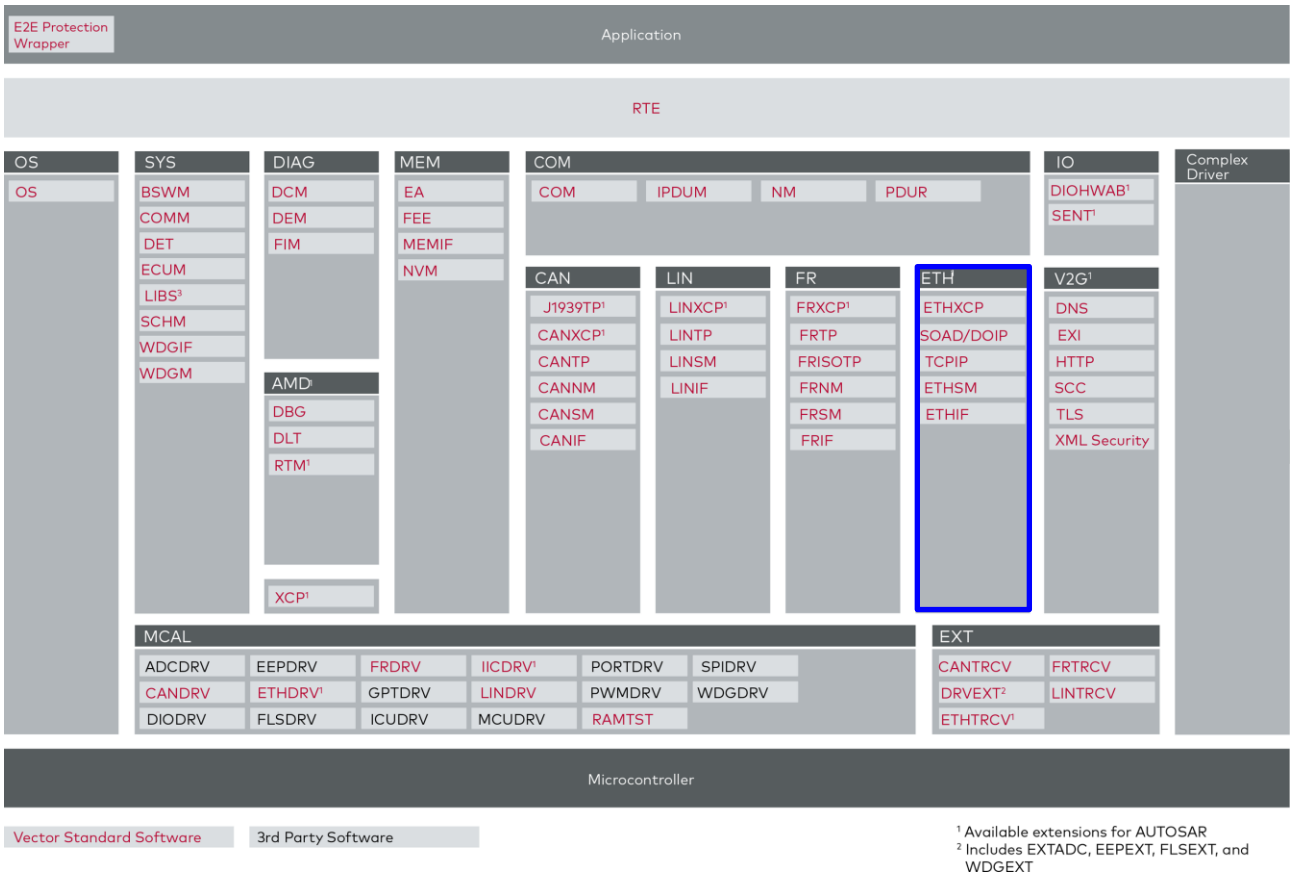

Figure 2-2 AUTOSAR 3.x Architecture Overview

The next figure shows the interfaces of IpBase provided to its users. These interfaces are described in chapter 5.

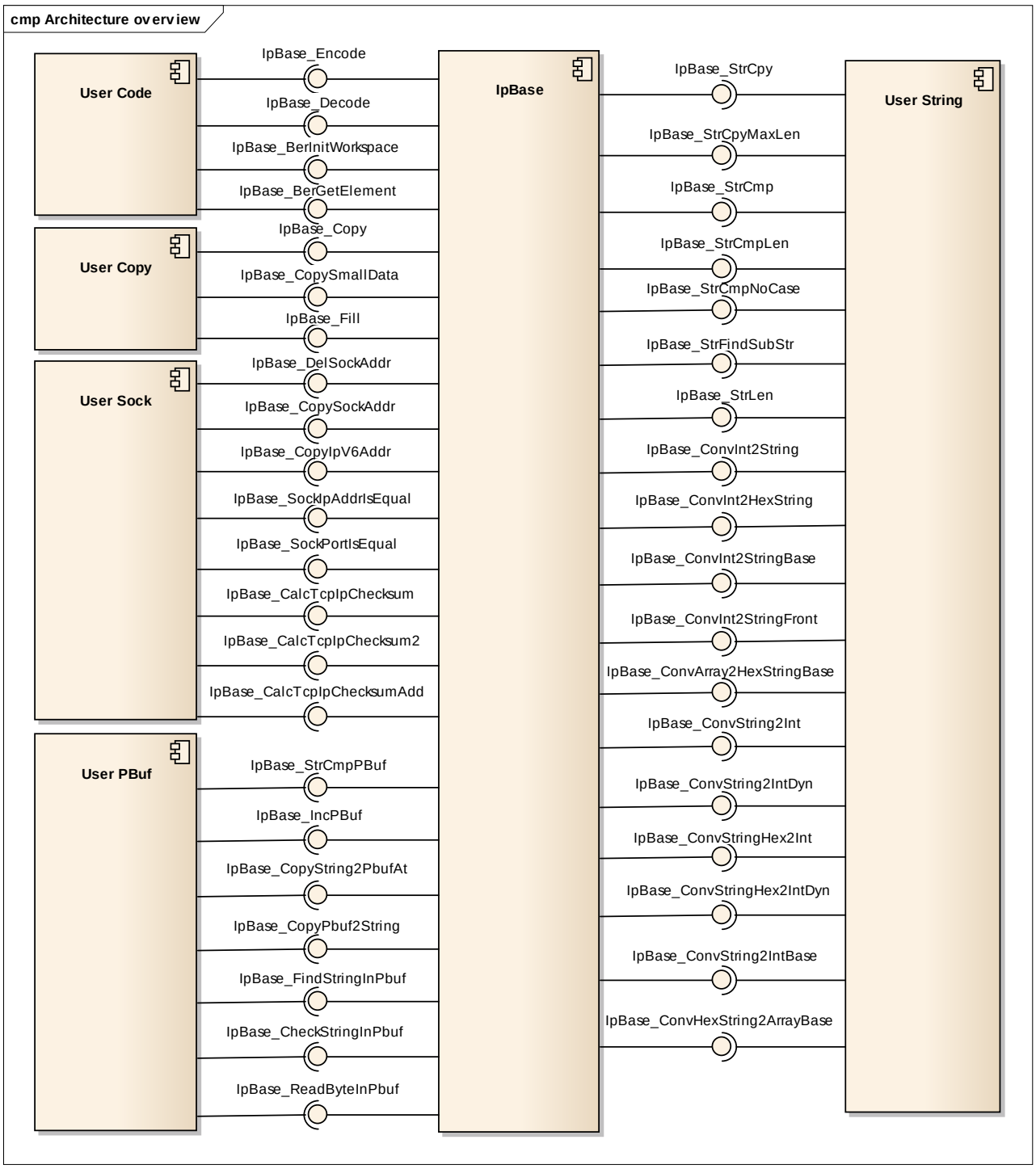

Figure 2-3 Interfaces of IpBase

3 Functional Description

3.1 Features

The features listed in this chapter cover the complete functionality provided by the module.

The "supported" and "not supported" features are presented in the following two tables.

The following features are supported:

Supported Feature
Base64 and ASN.1 (BER) encoding and decoding
Generic base copy
Linked buffer handling
String handling (copy, compare, conversion)
Socket handling (compare, copy, reset and checksum calculation)

Table 3-1 Supported IpBase features

3.2 Initialization

The IpBase component does not require initialization.

3.2.1 Configuration Variants 1 and 2 (Pre-Compile and Link-Time Configuration)

IpBase does not provide configuration.

3.2.2 Configuration Variant 3 (Post-build Configuration)

IpBase does not provide configuration.

3.3 States

The IpBase component is always operational.

3.4 Main Functions

The IpBase does not provide a main function.

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 `IPBASE_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 IpBase ID is 255.

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

Service ID	Service
0x01	IpBase_GetVersionInfo
0x11	IpBase_Encode
0x12	IpBase_Decode
0x13	IpBase_BerInitWorkspace
0x14	IpBase_BerGetElement
0x21	IpBase_Copy
0x22	IpBase_Fill
0x31	IpBase_StrCopy
0x32	IpBase_StrCopyMaxLen
0x33	IpBase_StrCmp
0x34	IpBase_StrCmpLen
0x35	IpBase_StrCmpNoCase
0x36	IpBase_StrFindSubStr
0x37	IpBase_StrLen
0x38	IpBase_ConvIntToStr
0x39	IpBase_ConvIntToHexStr
0x3A	IpBase_ConvIntToStrBase
0x3B	IpBase_ConvArrayToStrBase
0x3C	IpBase_ConvIntToStrFrong
0x3D	IpBase_ConvStrToInt
0x3E	IpBase_ConvStrToIntDyn
0x3F	IpBase_ConvHexStrToInt
0x40	IpBase_ConvHexStrToIntDyn
0x41	IpBase_ConvStrToIntBase
0x42	IpBase_ConvStrToSignedIntBase
0x43	IpBase_ConvHexStrToArrayBase
0x51	IpBase_StrCmpPbuf
0x52	IpBase_IncPbuf
0x53	IpBase_CopyStrToPBufAt
0x54	IpBase_CopyPbufToStr
0x55	IpBase_FindStrInPbuf
0x56	IpBase_ChkStrInPbuf
0x57	IpBase_ReadByteInPbuf
0x60	IpBase_DelSockAddr
0x61	IpBase_CopySockAddr
0x62	IpBase_CopyIpV6Addr

Service ID	Service
0x63	IpBase_SockIpAddrIsEqual
0x64	IpBase_SockPortIsEqual
0x65	IpBase_CalcTcpIpChecksum
0x66	IpBase_CalcTcpIpChecksum2
0x67	IpBase_CalcTcpIpChecksumAdd

Table 3-2 Service IDs

The IpBase component does not perform development error checks for internal APIs.

Service ID	Service

Table 3-3 Service IDs for internal APIs

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

Error Code		Description
0x01	IPBASE_E_INV_POINTER	Invalid pointer
0x02	IPBASE_E_INV_PARAM	Invalid parameter

Table 3-4 Errors reported to DET

3.5.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled separately. The configuration of en-/disabling the checks is described in chapter 6.2. En-/disabling of single checks is an addition to the AUTOSAR standard which requires to en-/disable the complete parameter checking via the parameter IPBASE_DEV_ERROR_DETECT.

The following table shows which parameter checks are performed on which services:

Service \ Check	IPBASE_E_INV_POINTER	IPBASE_E_INV_PARAM
IpBase_GetVersionInfo	■	
IpBase_Encode	■	
IpBase_Decode	■	
IpBase_BerInitWorkspace	■	
IpBase_BerGetElement	■	
IpBase_Copy	■	
IpBase_Fill	■	

Service \ Check	IPBASE_E_INV_POINTER	IPBASE_E_INV_PARAM
IpBase_StrCpy	■	
IpBase_StrCpyMaxLen	■	
IpBase_StrCmp	■	
IpBase_StrCmpLen	■	
IpBase_StrCmpNoCase	■	
IpBase_StrFindSubStr	■	■
IpBase_StrLen	■	
IpBase_ConvInt2String	■	
IpBase_ConvInt2HexString	■	
IpBase_ConvInt2StringBase	■	
IpBase_ConvArray2HexStringBase	■	
IpBase_ConvInt2StringFront	■	
IpBase_ConvString2Int	■	■
IpBase_ConvString2IntDyn	■	
IpBase_ConvStringHex2Int	■	■
IpBase_ConvStringHex2IntDyn	■	
IpBase_ConvString2IntBase	■	
IpBase_ConvString2SignedIntBase	■	
IpBase_ConvHexString2ArrayBase	■	
IpBase_StrCmpPBuf	■	
IpBase_IncPBuf	■	
IpBase_CopyString2PbufAt	■	■
IpBase_CopyPbuf2String	■	■
IpBase_FindStrInPbuf	■	■
IpBase_CheckStringInPbuf	■	■
IpBase_ReadByteInPbuf	■	■
IpBase_DelSockAddr	■	
IpBase_CopySockAddr	■	
IpBase_CopyIpV6Addr	■	
IpBase_SockIpAddrIsEqual	■	
IpBase_SockPortIsEqual	■	
IpBase_CalcTcpIpChecksum	■	
IpBase_CalcTcpIpChecksum2	■	
IpBase_CalcTcpIpChecksumAdd	■	

Table 3-5 Development Error Reporting: Assignment of checks to services

3.5.2 Production Code Error Reporting

Not used.

4 Integration

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

4.1 Scope of Delivery

The delivery of the IpBase contains the files which are described in the chapters 4.1.1 and 4.1.2.

4.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
IpBase.a		■	IpBase library.
IpBase.c	■		Static source for core.
IpBase.h	■	■	Static header for API.
IpBase_Cfg.h	■	■	Static header for configuration.
IpBase_Code.c	■		Static source for coding.
IpBase_Code.h	■	■	Static header for coding.
IpBase_Copy.c	■		Static source for copy.
IpBase_Copy.h	■	■	Static header for copy.
IpBase_PBuf.c	■		Static source for buffer.
IpBase_PBuf.h	■	■	Static header for buffer.
IpBase_Sock.c	■		Static source for socket.
IpBase_Sock.h	■	■	Static header for socket.
IpBase_String.c	■		Static source for string.
IpBase_String.h	■	■	Static header for string.
IpBase_Priv.h	■	■	Static header for internal macro and variable declaration.
IpBase_Types.h	■	■	Static header for type definitions.
_Appl_Rand.c	■	■	Template static source for random functions.
_Appl_Rand.h	■	■	Template static header for random functions.
_Appl_Time.c	■	■	Template static source for time functions.
_Appl_Time.h	■	■	Template static header for time functions.

Table 4-1 Static files

4.1.2 Dynamic Files

No dynamic files used.

File Name	Description

Table 4-2 Generated files

4.2 Include Structure

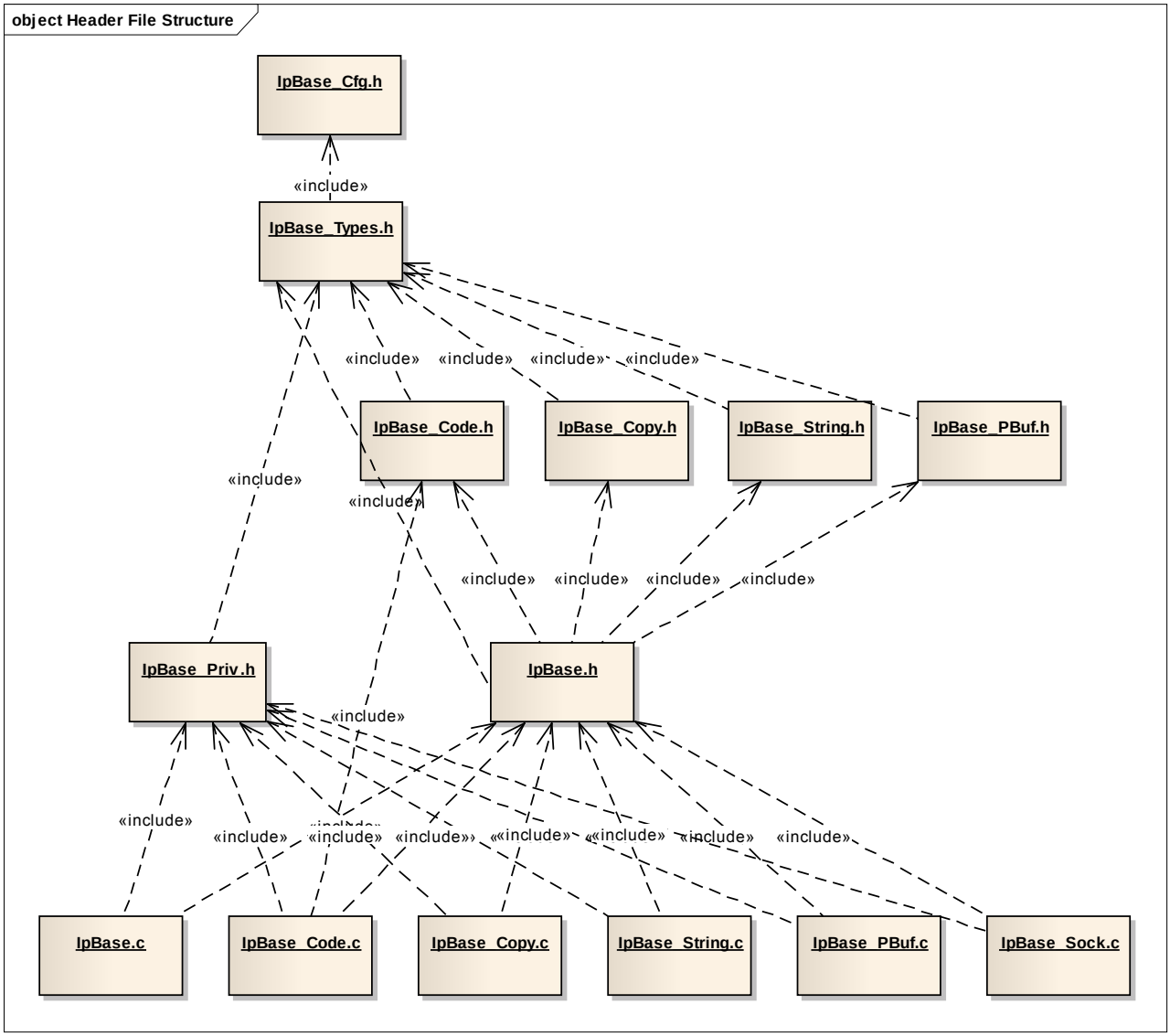

Figure 4-1 Include structure

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 of the IpBase and illustrates their assignment among each other.

Memory Mapping Sections / Compiler Abstraction Definitions	IPBASE_VAR_NOINIT	IPBASE_CONST	IPBASE_CODE	IPBASE_PBCFG
IPBASE_START_SEC_PBCFG IPBASE_STOP_SEC_PBCFG				■
IPBASE_START_SEC_CODE IPBASE_STOP_SEC_CODE			■	
IPBASE_START_SEC_CONST_UNSPECIFIED IPBASE_STOP_SEC_ CONST_UNSPECIFIED		■		
IPBASE_START_SEC_CONST_32BIT IPBASE_STOP_SEC_ CONST_32BIT		■		
IPBASE_START_SEC_CONST_16BIT IPBASE_STOP_SEC_ CONST_16BIT		■		
IPBASE_START_SEC_CONST_8BIT IPBASE_STOP_SEC_ CONST_8BIT		■		
IPBASE_START_SEC_VAR_NOINIT_UNSPECIFIED IPBASE_STOP_SEC_VAR_NOINT_UNSPECIFIED	■			
IPBASE_START_SEC_VAR_NOINIT_32BIT IPBASE_STOP_SEC_VAR_NOINT_32BIT	■			
IPBASE_START_SEC_VAR_NOINIT_16BIT IPBASE_STOP_SEC_VAR_NOINT_16BIT	■			
IPBASE_START_SEC_VAR_NOINIT_8BIT IPBASE_STOP_SEC_VAR_NOINT_8BIT	■			

Table 4-3 Compiler abstraction and memory mapping

4.4 Critical Sections

Currently, no critical sections are used.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

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

Type Name	C-Type	Description	Value Range
IpBase_AddrInType	uint32	Type used for IP addresses	0x00000000 – 0xFFFFFFFF
IpBase_IPAddressType	uint32	Type used for IP addresses (limited to IPv4)	0x00000000 – 0xFFFFFFFF
IpBase_CopyDataType	uint32	Type used for copy routines	0x00000000 – 0xFFFFFFFF
IpBase_FamilyType	uint16	Type used for IP address family	0x04 or 0x06
IpBase_PortType	uint16	Type used for port	0x0000 - 0xFFFF
IpBase_EthPhysAddrType	uint8	Array used for Ethernet physical address (MAC address)	0x000000000000 – 0xFFFFFFFFFFFF
IpBase_SockIdxType	uint8	Type used for socket index	0x00 – 0xFF
IpBase_ReturnType	uint8	Type used for return values	IPBASE_E_OK 0x00 IPBASE_E_NOT_OK 0x81 IPBASE_E_PENDING 0x82 IPBASE_E_MEM 0x83 IPBASE_E_BER_PARAM 0x84
IpBase_TcpIpEventType	uint8	Type used for TCP/IP events	IPBASE_TCP_EVENT_RESET 0x01 IPBASE_TCP_EVENT_CLOSED 0x02 IPBASE_TCP_EVENT_FIN_RECEIVED 0x03
IpBase_PbufType	struct	Type used for distributed buffers	Payload pointer, total length, segment length
IpBase_IpAddrPortType	struct	Type used for TCP/IP addressing (limited to IPv4)	Port, length, IPv4 address Hint: Type is deprecated due to limitation to IPv4
IpBase_SockAddrType	struct	Type used for TCP/IP addressing	Family, address data abstract base type for IpBase_SockAddrInType and IpBase_SockAddrIn6Type
IpBase_SockAddrInType	struct	Type used for TCP/IP addressing	Family, port, IPv4 address

Type Name	C-Type	Description	Value Range
IpBase_AddrIn6Type	Struct	Type used to store IPv6 address	IPv6 address
IpBase_SockAddrIn6Type	Struct	Type used to store socket address	Family, port, IPv6 address

Table 5-1 Type definitions

5.2 Services provided by IpBase

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

5.2.1 IpBase_GetVersionInfo

Prototype	
`void IpBase_GetVersionInfo (Std_VersionInfoType *VersionInfoPtr)`	
Parameter	
`VersionInfoPtr`	pointer for version information
Return code	
void	none
Functional Description	
Returns version information, vendor ID and AUTOSAR module ID of the component. The versions are decimal coded.	
Particularities and Limitations	
none	
Call Context	
task	

Table 5-2 IpBase_GetVersionInfo

5.2.2 IpBase_Encode

Prototype	
`Std_ReturnType IpBase_Encode (uint8 Code, uint8 *TgtDataPtr, const uint8 *SrcDataPtr, uint32 *TgtLenBytePtr, uint32 SrcLenByte)`	
Parameter	
Code	defines the code used for encoding
TgtDataPtr	pointer for the encoded data
SrcDataPtr	pointer to the raw data
TgtLenBytePtr	pointer for the encoded data length in bytes
SrcLenByte	raw data length in bytes
Return code	
Std_ReturnType	E_ OK data encoded E_NOT_OK encoding failed

Functional Description
encodes the given data using the specified code
Particularities and Limitations
intended as general encoding
Call Context
task

Table 5-3 IpBase_Encode

5.2.3 IpBase_Decode

Prototype	
`Std_ReturnType` **`IpBase_Decode`** `(uint8 Code, uint8 *TgtDataPtr, const uint8 *SrcDataPtr, uint32 *TgtLenBytePtr, uint32 SrcLenByte)`	
Parameter	
Code	defines the code used for decoding
TgtDataPtr	pointer for the decoded data
SrcDataPtr	pointer to the raw data
TgtLenBytePtr	pointer for the decoded data length in bytes
SrcLenByte	raw data length in bytes
Return code	
Std_ReturnType	E_ OK data decoded E_NOT_OK decoding failed
Functional Description	
decodes the given data using the specified code	
Particularities and Limitations	
intended as general decoding	
Call Context	
task	

Table 5-4 IpBase_Decode

5.2.4 IpBase_BerInitWorkspace

Prototype	
`void` **`IpBase_BerInitWorkspace`** `(CONST IpBase_BerWorkspaceType *WorkspacePtr, CONSTIpBase_BerStackElementType *StackPtr, const uint8 Depth)`	
Parameter	
WorkspacePtr	the workspace to initialize
StackPtr	the stack to use
Depth	the depth of the stack

Return code	
void	none
Functional Description	
initialize the ASN.1/BER parser workspace	
Particularities and Limitations	
none	
Call Context	
task	

Table 5-5 IpBase_BerInitWorkspace

5.2.5 IpBase_BerGetElement

Prototype	
`IpBase_ReturnType` **`IpBase_BerGetElement`** `(CONSTIpBase_BerWorkspaceType *WorkspacePtr, CONSTIpBase_BerElementType *ElementPtr, CONSTconst uint8 *ElementNrPtr, const uint8 ElementDepth, CONSTconst uint8 *DataPtr, const uint32 DataSize)`	
Parameter	
WorkspacePtr	the internally used workspace
ElementPtr	the found element
ElementNrPtr	the element number (chapter.section.subsection. ...)
ElementDepth	the depth of the element (chapter = 1, chapter.section = 2, ...)
DataPtr	the data
DataSize	the size of the data
Return code	
IpBase_ReturnType	IPBASE_E_OK element found IPBASE_E_NOT_OK element not found IPBASE_E_INV_PARAM data corrupt IPBASE_E_MEM memory exceeded
Functional Description	
get an ASN.1/BER element with a given number out of ASN.1/BER encoded data	
Particularities and Limitations	
none	
Call Context	
task	

Table 5-6 IpBase_BerGetElement

5.2.6 IpBase_Copy

Prototype
`void` **`IpBase_Copy`** `(IpBase_CopyDataType *TgtDataPtr, const IpBase_CopyDataType *SrcDataPtr, uint32 LenByte)`

Parameter	
TgtDataPtr	pointer for target data
SrcDataPtr	pointer to source data
LenByte	data length in bytes
Return code	
void	none
Functional Description	
copy data (memcpy)	
Particularities and Limitations	
Depending on Enable Copy Macro in IpBase_Cfg.h the function is implemented as macro and using VStdLib_MemCpy	
Call Context	
interrupt or task level	

Table 5-7 IpBase_Copy

5.2.7 IpBase_CopySmallData

Prototype	
`void IpBase_CopySmallData (IpBase_CopyDataType *TgtDataPtr, const IpBase_CopyDataType *SrcDataPtr, uint32 LenByte)`	
Parameter	
TgtDataPtr	pointer for target data
SrcDataPtr	pointer to source data
LenByte	data length in bytes
Return code	
void	none
Functional Description	
copy data (memcpy) for less than 32 data chunks. Chunk size depends on the alignment -> i.e. 32 bytes for 1 byte aligned data, 64 bytes chunks for 2 byte aligned data, 128 bytes for 4 byte aligned data	
Particularities and Limitations	
Depending on Enable Copy Macro in IpBase_Cfg.h the function is implemented as macro and using VStdLib_MemCpy	
Call Context	
interrupt or task level	

Table 5-8 IpBase_Copy

5.2.8 IpBase_Fill

Prototype
`void IpBase_Fill (IpBase_CopyDataType *TgtDataPtr, uint8 Pattern, uint32 LenByte)`

Parameter	
TgtDataPtr	pointer for target data
Pattern	fill pattern
LenByte	data length in bytes
Return code	
void	none
Functional Description	
fill data (memset)	
Particularities and Limitations	
none	
Call Context	
interrupt or task level	

Table 5-9 IpBase_Fill

5.2.9 IpBase_StrCmpPBuf

Prototype
`uint8 IpBase_StrCmpPBuf (const IpBase_PbufType **SrcPBufPtr, const char *PatternPtr, uint16 *CurByteIdxPtr, uint32 *TotByteIdxPtr, uint32 *RestLenBytePtr)`
Parameter
SrcPBufPtr — pointer to source data
PatternPtr — string pattern
CurByteIdxPtr — local start index
TotByteIdxPtr — total start index
RestLenBytePtr — unread sniplet
Return code
IpBase_ReturnType — 0 string found IPBASE_CMP_NOT_EQUAL string not found
Functional Description
local PBuf string compare routine
Particularities and Limitations
supports PBuf, only for short string comparisons (bytewise access)
Call Context
interrupt or task level

Table 5-10 IpBase_StrCmpPBuf

5.2.10 IpBase_IncPBuf

Prototype	
`void IpBase_IncPBuf (IpBase_PbufType **PBufPtr, uint16 *CurByteIdxPtr, uint32 *TotByteIdxPtr)`	
Parameter	
PBufPtr	pointer to PBuf struct
CurByteIdxPtr	pointer to current byte idx within PBuf struct
TotByteIdxPtr	pointer to total byte idx within PBuf struct
Return code	
void	none
Functional Description	
increment PBuf	
Particularities and Limitations	
increments the pbuf byte access. Switches to next PBuf at end of segment	
Call Context	
task level	

Table 5-11 IpBase_IncPBuf

5.2.11 IpBase_CopyString2PbufAt

Prototype	
`Std_ReturnType IpBase_CopyString2PbufAt (const uint8 *StrPtr, const uint16 StrLen, IpBase_PbufType *PbufPtr, uint32 StartPos)`	
Parameter	
StrPtr	pointer to source string
StrLen	length of the source string [byte]
PbufPtr	pointer to destination Pbuf struct
StartPos	start position in Pbuf
Return code	
Std_ReturnType	E_OK string could be copied E_NOT_OK string could not be copied
Functional Description	
copy a string to a pbuf at a defined position	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-12 IpBase_CopyString2PbufAt

5.2.12 IpBase_CopyPbuf2String

Prototype	
`Std_ReturnType` **`IpBase_CopyPbuf2String`** `(uint8 *StrPtr, const IpBase_PbufType *PbufPtr, uint16 StrLen, uint32 StartPos)`	
Parameter	
StrPtr	pointer to string
PbufPtr	pointer to Pbuf struct
StrLen	length of the string [byte]
StartPos	absolute start position in Pbuf
Return code	
Std_ReturnType	E_OK string was copied E_NOT_OK string was not copied
Functional Description	
find a string in a pbuf	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-13 IpBase_CopyPbuf2String

5.2.13 IpBase_FindStringInPbuf

Prototype	
`Std_ReturnType` **`IpBase_FindStringInPbuf`** `(const uint8 *StrPtr, const IpBase_PbufType *PbufPtr, uint16 StrLen, uint32 StartPos, uint32 *StrPosPtr)`	
Parameter	
StrPtr	pointer to search string
PbufPtr	pointer to Pbuf struct
StrLen	length of the search string [byte]
StartPos	start position for search in Pbuf
StrPosPtr	index in Pbuf where the searched string starts
Return code	
Std_ReturnType	E_OK string was found E_NOT_OK string was not found or API parameters are invalid
Functional Description	
find a string in a pbuf	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-14 IpBase_FindStringInPbuf

5.2.14 IpBase_CheckStringInPbuf

Prototype	
`Std_ReturnType` **`IpBase_CheckStringInPbuf`** `(const uint8 *StrPtr, const IpBase_PbufType *PbufPtr, uint16 StrLen, uint32 StartPos)`	
Parameter	
StrPtr	pointer to search string
PbufPtr	pointer to Pbuf struct
StrLen	length of the search string [byte]
StartPos	start position for search in Pbuf
Return code	
Std_ReturnType	E_OK string was found E_NOT_OK string was not found or API parameters are invalid
Functional Description	
check whether a string is found in a pbuf at the given position	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-15 IpBase_CheckStringInPbuf

5.2.15 IpBase_ReadByteInPbuf

Prototype	
`Std_ReturnType` **`IpBase_ReadByteInPbuf`** `(const IpBase_PbufType *PbufPtr, uint32 BytePos, uint8 *SingleBytePtr)`	
Parameter	
PbufPtr	pointer to Pbuf struct
BytePos	absolute byte position in Pbuf
SingleBytePtr	pointer where the byte shall be copied to
Return code	
Std_ReturnType	E_OK byte was copied E_NOT_OK byte was not copied
Functional Description	
read a byte in a pbuf	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-16 IpBase_ReadByteInPbuf

5.2.16 IpBase_DelSockAddr

Prototype	
`Std_ReturnType` **`IpBase_DelSockAddr`** `(IpBase_SockAddrType *SockPtr, uint16 Family)`	
Parameter	
SockPtr	socket address
Family	supported family
Return code	
Std_ReturnType	E_OK SockAddr could be deleted E_NOT_OK deletion failed
Functional Description	
delete socket address	
Particularities and Limitations	
none	
Call Context	
interrupt or task level	

Table 5-17 IpBase_DelSockAddr

5.2.17 IpBase_CopySockAddr

Prototype	
`Std_ReturnType` **`IpBase_CopySockAddr`** `(IpBase_SockAddrType *TgtSockPtr, const IpBase_SockAddrType *SrcSockPtr)`	
Parameter	
TgtSockPtr	target socket address
SrcSockPtr	source socket address
Return code	
Std_ReturnType	E_OK SockAddr could be copied E_NOT_OK copy failed
Functional Description	
copy socket address (incl. family, port, ip-addr) from Src to Tgt	
Particularities and Limitations	
none	
Call Context	
interrupt or task level	

Table 5-18 IpBase_CopySockAddr

5.2.18 IpBase_CopyIpV6Addr

Prototype
`Std_ReturnType` **`IpBase_CopyIpV6Addr`** `(IpBase_AddrIn6Type *TgtIpAddrPtr, const IpBase_AddrIn6Type *SrcIpAddrPtr)`

Parameter	
TgtIpAddrPtr	target IP address
SrcIpAddrPtr	source IP address
Return code	
Std_ReturnType	E_OK IP addr could be copied E_NOT_OK copy failed
Functional Description	
copy socket address (incl. family, port, ip-addr) from Src to Tgt	
Particularities and Limitations	
none	
Call Context	
interrupt or task level	

Table 5-19 IpBase_CopyIpV6Addr

5.2.19 IpBase_SockIpAddrIsEqual

Prototype
`boolean` **`IpBase_SockIpAddrIsEqual`** `(IpBase_SockAddrType *SockAPtr, IpBase_SockAddrType *SockBPtr)`
Parameter
SockAPtr — socket address A
SockBPtr — socket address B
Return code
boolean — TRUE IP address is equal FALSE IP address is not equal
Functional Description
check if IP address of sockets is equal
Particularities and Limitations
none
Call Context
interrupt or task level

Table 5-20 IpBase_SockIpAddrIsEqual

5.2.20 IpBase_SockPortIsEqual

Prototype
`boolean` **`IpBase_SockPortIsEqual`** `(IpBase_SockAddrType *SockAPtr, IpBase_SockAddrType *SockBPtr)`
Parameter
SockAPtr — target socket address
SockBPtr — source socket address

Return code	
boolean	TRUE port is equal FALSE port is not equal
Functional Description	
check if port of sockets is equal	
Particularities and Limitations	
none	
Call Context	
interrupt or task level	

Table 5-21 IpBase_SockPortIsEqual

5.2.21 IpBase_CalcTcpIpChecksum

Prototype	
`uint16 IpBase_CalcTcpIpChecksum (uint8 *DataPtrStart, uint32 LenByte)`	
Parameter	
DataPtrStart	pointer to the data
LenByte	data length in bytes
Return code	
uint16	calculated checksum
Functional Description	
This API calculates the checksum over a given data range. The checksum is TcpIp specific. I.e. it expects 16bit data chunks and uses one's complement checksum algorithm.	
Particularities and Limitations	
Deprecated	
Call Context	
interrupt or task level	

Table 5-22 IpBase_CalcTcpIpChecksum

5.2.22 IpBase_CalcTcpIpChecksum2

Prototype	
`uint16 IpBase_CalcTcpIpChecksum2 (uint8 *DataPtrStart, uint32 LenByte, uint8 *PseudoHdrPtrStart, uint32 PseudoHdrLenByte)`	
Parameter	
DataPtrStart	pointer to the data
LenByte	data length in bytes
PseudoHdrPtrStart	pointer to the pseudo header
PseudoHdrLenByte	pseudo header length in bytes
Return code	
uint16	calculated checksum

Functional Description
This API calculates the checksum over two given data ranges. The checksum is TcpIp specific. I.e. it expects 16bit data chunks and uses one's complement checksum algorithm.
Particularities and Limitations
Deprecated
Call Context
interrupt or task level

Table 5-23 IpBase_CalcTcpIpChecksum2

5.2.23 IpBase_CalcTcpIpChecksumAdd

Prototype	
`uint16 IpBase_CalcTcpIpChecksumAdd (uint8 *DataPtr, uint32 LenByte, uint32 Checksum, boolean Stop)`	
Parameter	
DataPtrStart	pointer to the data
LenByte	data length in bytes
PseudoHdrPtrStart	pointer to the pseudo header
PseudoHdrLenByte	pseudo header length in bytes
Return code	
uint16	calculated checksum
Functional Description	
This API adds a range to TcpIp checkusm calculation. The checksum is TcpIp specific. I.e. it expects 16bit data chunks and uses one's complement checksum algorithm.	
Particularities and Limitations	
none	
Call Context	
interrupt or task level	

Table 5-24 IpBase_CalcTcpIpChecksumAdd

5.2.24 IpBase_StrCpy

Prototype	
`uint8 IpBase_StrCpy (uint8 *TgtPtr, const uint8 *SrcPtr)`	
Parameter	
TgtPtr	pointer for target string
SrcPtr	pointer to source string
Return code	
uint8	number of copied bytes
Functional Description	
string copy (zero terminated strings)	

Particularities and Limitations
the source string has to be terminated by '\0'. Deprecated.
Call Context
task level

Table 5-25 IpBase_StrCpy

5.2.25 IpBase_StrCpyMaxLen

Prototype	
`uint8 IpBase_StrCpyMaxLen (uint8 *TgtPtr, const uint8 *SrcPtr, uint32 MaxLen)`	
Parameter	
TgtPtr	pointer for target string
SrcPtr	pointer to source string
MaxLen	maximum length
Return code	
uint8	number of copied bytes
Functional Description	
string copy (zero terminated strings) with length limitation	
Particularities and Limitations	
the source string has to be terminated by '\0'	
Call Context	
task level	

Table 5-26 IpBase_StrCpyMaxLen

5.2.26 IpBase_StrCmp

Prototype	
`uint8 IpBase_StrCmp (const uint8 *Str1Ptr, const uint8 *Str2Ptr)`	
Parameter	
Str1Ptr	pointer to first string
Str2Ptr	pointer to second string
Return code	
uint8	IPBASE_CMP_EQUAL strings are equal IPBASE_CMP_NOT_EQUAL string pattern not found
Functional Description	
compare 2 strings until end of the shorter string (accepting sub strings)	
Particularities and Limitations	
the strings have to be terminated by '\0'. String subsets are accepted (i.e. "Hello" == "Hello World")	
Call Context	

task level

Table 5-27 IpBase_StrCmp

5.2.27 IpBase_StrCmpLen

Prototype	
`uint8 IpBase_StrCmpLen (const uint8 *Str1Ptr, const uint8 *Str2Ptr, uint16 StrLen)`	
Parameter	
Str1Ptr	pointer to string 1
Str2Ptr	pointer to string 2
StrLen	length of the search string [byte]
Return code	
uint8	IPBASE_CMP_EQUAL strings are equal IPBASE_CMP_NOT_EQUAL strings are not equal, or other error condition occurred
Functional Description	
check whether the two strings are equal or not	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-28 IpBase_StrCmpLen

5.2.28 IpBase_StrCmpNoCase

Prototype	
`uint8 IpBase_StrCmpNoCase (const uint8 *Str1Ptr, const uint8 *Str2Ptr)`	
Parameter	
Str1Ptr	pointer to first string
Str2Ptr	pointer to second string
Return code	
uint8	IPBASE_CMP_EQUAL strings are equal IPBASE_CMP_NOT_EQUAL string pattern not found
Functional Description	
compare 2 strings until end of the shorter string ignoring the case (accepting sub strings)	
Particularities and Limitations	
the strings have to be terminated by '\0'. String subsets are accepted (i.e. "Hello" == "Hello World"). Case is ignored (i.e. "Hello" == "HELLO")	
Call Context	
task level	

Table 5-29 IpBase_StrCmpNoCase

5.2.29 IpBase_StrFindSubStr

Prototype	
`uint32 IpBase_StrFindSubStr (const uint8 *StrPtr, const uint8 *SubStrPtr, uint16 StrLen, uint16 SubStrLen)`	
Parameter	
StrPtr	pointer to string
SubStrPtr	pointer to sub string
StrLen	length of the string [byte]
SubStrLen	length of the sub string [byte]
Return code	
uint32	PosByte position in string where the sub-string starts IPBASE_STR_LEN_INVALID sub-string not found or error
Functional Description	
searches for the first occurence of sub-string within a string (e.g. string "hello world", sub-string "world")	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-30 IpBase_StrFindSubStr

5.2.30 IpBase_StrLen

Prototype	
`uint32 IpBase_StrLen (const uint8 *StrPtr, uint32 MaxLen)`	
Parameter	
StrPtr	pointer to string
MaxLen	maximum length of the search string [byte]
Return code	
uint32	0..MaxLen-1 length of the string
Functional Description	
check the length of the string	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-31 IpBase_StrLen

5.2.31 IpBase_ConvInt2String

Prototype	
`Std_ReturnType` **`IpBase_ConvInt2String`** `(uint32 IntVal, uint8 **StrPtr, uint8 *StrLenPtr)`	
Parameter	
IntVal	integer number
StrPtr	pointer to string
StrLenPtr	pointer to length of the string [byte]
Return code	
Std_ReturnType	E_OK integer converted E_NOT_OK integer conversion failed
Functional Description	
convert an integer number to an ASCII string (dec)	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-32 IpBase_ConvInt2String

5.2.32 IpBase_ConvInt2HexString

Prototype	
`Std_ReturnType` **`IpBase_ConvInt2HexString`** `(uint32 IntVal, uint8 **StrPtr, uint8 *StrLenPtr)`	
Parameter	
IntVal	integer number
StrPtr	pointer to string (hex coded)
StrLenPtr	pointer to length of the string [byte]
Return code	
Std_ReturnType	E_OK integer converted E_NOT_OK integer conversion failed
Functional Description	
convert an integer number to an ASCII string (hex)	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-33 IpBase_ConvInt2HexString

5.2.33 IpBase_ConvInt2StringBase

Prototype	
`Std_ReturnType` **`IpBase_ConvInt2StringBase`** `(uint32 IntVal, uint8 *StrPtr, uint8 StrLen)`	
Parameter	
IntVal	integer number
StrPtr	pointer to string
StrLenPtr	pointer to length of the string [byte]
Return code	
Std_ReturnType	E_OK integer converted E_NOT_OK integer conversion failed
Functional Description	
convert an integer number to an ASCII string (dec) without incrementing StrPtr but '\0' at end	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-34 IpBase_ConvInt2StringBase

5.2.34 IpBase_ConvInt2StringFront

Prototype	
`Std_ReturnType` **`IpBase_ConvInt2StringFront`** `(uint32 IntVal, uint8 **StrPtr, uint8 *StrLenPtr)`	
Parameter	
IntVal	integer number
StrPtr	pointer to string
StrLenPtr	pointer to length of the string [byte]
Return code	
Std_ReturnType	E_OK integer converted E_NOT_OK integer conversion failed
Functional Description	
convert an integer number to an ASCII string (dec) without incrementing StrPtr but '\0' at end	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-35 IpBase_ConvInt2StringFront

5.2.35 IpBase_ConvArray2HexStringBase

Prototype	
`Std_ReturnType` **`IpBase_ConvArray2HexStringBase`** `(uint8 *ArrayPtr, uint16 ArrayLen, uint8 *StrPtr)`	
Parameter	
ArrayPtr	pointer to array
ArrayLen	array length [byte]
StrPtr	pointer to string (has to provide (ArrayLen*2)+1 chars)
Return code	
Std_ReturnType	E_OK array converted E_NOT_OK integer conversion failed
Functional Description	
convert an array number to an ASCII string (hex), omits leading '00'	
Particularities and Limitations	
Array [a0a1], ArrayLen 2 -> 'A0A1'.Array [00a1], ArrayLen 2 -> 'A1'	
Call Context	
task level	

Table 5-36 IpBase_ConvArray2HexStringBase

5.2.36 IpBase_ConvString2Int

Prototype	
`Std_ReturnType` **`IpBase_ConvString2Int`** `(const uint8 *StrPtr, const uint8 StrLen, uint32 *IntValPtr)`	
Parameter	
StrPtr	pointer to string
StrLen	length of the string [byte]
IntValPtr	pointer to integer number
Return code	
Std_ReturnType	E_OK string could be converted to integer E_NOT_OK string could not be converted to integer
Functional Description	
convert an ASCII string (dec values) to an integer	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-37 IpBase_ConvString2Int

5.2.37 IpBase_ConvString2IntDyn

Prototype	
`Std_ReturnType` **`IpBase_ConvString2IntDyn`** `(const uint8 **StrPtr, uint8 *StrLenPtr, uint32 *IntValPtr)`	
Parameter	
StrPtr	pointer to string
StrLen	length of the string [byte]
IntValPtr	pointer to integer number
Return code	
Std_ReturnType	E_OK string could be converted to integer E_NOT_OK string could not be converted to integer
Functional Description	
convert an ASCII string (dec values) to an integer with dynamic length	
Particularities and Limitations	
Str '12', StrLen 2 -> 12. Str '12', StrLen 1 -> 1.	
Call Context	
task level	

Table 5-38 IpBase_ConvString2IntDyn

5.2.38 IpBase_ConvStringHex2Int

Prototype	
`Std_ReturnType` **`IpBase_ConvStringHex2Int`** `(const uint8 *StrPtr, const uint8 StrLen, uint32 *IntValPtr)`	
Parameter	
StrPtr	pointer to string
StrLen	length of the string [byte]
IntValPtr	pointer to integer number
Return code	
Std_ReturnType	E_OK string could be converted to integer E_NOT_OK string could not be converted to integer
Functional Description	
convert an ASCII string (hex values) to an integer	
Particularities and Limitations	
none	
Call Context	
task level	

Table 5-39 IpBase_ConvStringHex2Int

5.2.39 IpBase_ConvStringHex2IntDyn

Prototype	
`Std_ReturnType` **`IpBase_ConvStringHex2IntDyn`** `(const uint8 **StrPtr, uint8 *StrLenPtr, uint32 *IntValPtr)`	
Parameter	
StrPtr	pointer to string
StrLen	length of the string [byte]
IntValPtr	pointer to integer number
Return code	
Std_ReturnType	E_OK string could be converted to integer E_NOT_OK string could not be converted to integer
Functional Description	
convert an ASCII string (hex values) to an integer with dynamic length	
Particularities and Limitations	
Str '12', StrLen 2 -> 0x12. Str '12', StrLen 1 -> 0x1.	
Call Context	
task level	

Table 5-40 IpBase_ConvStringHex2IntDyn

5.2.40 IpBase_ConvString2IntBase

Prototype	
`uint32` **`IpBase_ConvString2IntBase`** `(const uint8 *StrPtr, uint8 StrMaxLen)`	
Parameter	
StrPtr	pointer to string
StrMaxLen	max length of the string [byte]
Return code	
uint32	0-4294967295 string converted to integer
Functional Description	
convert an ASCII string (dec values) to an integer	
Particularities and Limitations	
Str '12', StrMaxLen 2 -> 12. Str '12', StrMaxLen 1 -> 1.	
Call Context	
task level	

Table 5-41 IpBase_ConvString2IntBase

5.2.41 IpBase_ConvString2SignedIntBase

Prototype
`sint32` **`IpBase_ConvString2SignedIntBase`** `(const uint8 *StrPtr, uint8 StrMaxLen)`

Parameter	
StrPtr	pointer to string
StrMaxLen	max length of the string [byte]
Return code	
sint32	-2147483646-+2147483647 string converted to integer
Functional Description	
convert an ASCII string (dec values) to a signed integer	
Particularities and Limitations	
Str '12', StrMaxLen 2 -> 12. Str '12', StrMaxLen 1 -> 1. Str 'a' -> IPBASE_ULONG_MAX.	
Call Context	
task level	

Table 5-42 IpBase_ConvString2SignedIntBase

5.2.42 IpBase_ConvHexString2ArrayBase

Prototype	
`Std_ReturnType` **`IpBase_ConvHexString2ArrayBase`** `(uint8 *ArrayPtr, uint16 ArrayLen, CONSTconst uint8 *StrPtr)`	
Parameter	
ArrayPtr	pointer to array
ArrayLen	array length [byte]
StrPtr	pointer to string (has to provide (ArrayLen*2)+1 chars)
Return code	
Std_ReturnType	E_OK array converted E_NOT_OK array conversion failed
Functional Description	
convert an ASCII hex string to an array number, omits leading '00'	
Particularities and Limitations	
Array [a0a1], ArrayLen 2 -> 'A0A1'.Array [00a1], ArrayLen 2 -> 'A1'	
Call Context	
task level	

Table 5-43 IpBase_ConvHexString2ArrayBase

5.3 Configurable Interfaces

5.3.1 Notifications

At its configurable interfaces the IpBase defines no notifications that can be mapped to callback functions provided by other modules.

6 Configuration

In the IpBase attributes can be configured according with the following methods/ tools:

> Manual adaption of IpBase_Cfg.h, for a detailed description see 6.2

6.1 Configuration Variants

The IpBase supports no configuration variants.

6.2 Configuration with IpBase_Cfg.h

The IpBase is configured with the file IpBase_Cfg.h.

6.2.1 Component Configuration

The following attributes can be configured for IpBase, if it is delivered as source code.

Attribute Name	Value Type	Values The default value is written in bold	Description
Enable Version Info API	Boolean	**STD_ON**, STD_OFF	Enables the Version Information API.
Enable Development Error Detection	Boolean	STD_ON, **STD_OFF**	Enables the Development Error Tracing (DET).
Enable Code	Boolean	**STD_ON**, STD_OFF	Enables the encoding and decoding functionality.
Enable Copy	Boolean	**STD_ON**, STD_OFF	Enables the copy functionality.
Enable Copy Macro	Boolean	**STD_ON**, STD_OFF	Enables the copy/fill functionality as macro based on VStdMemCpy/VStdMemSet in VStdLib.
Enable PBuf	Boolean	**STD_ON**, STD_OFF	Enables the PBuf handling functionality.
Enable Sock	Boolean	**STD_ON**, STD_OFF	Enables the socket handling functionality.
Enable String	Boolean	**STD_ON**, STD_OFF	Enables the string handling functionality.
Enable Code BASE64	Boolean	**STD_ON**, STD_OFF	Enables the BASE64 encoding and decoding functionality.
Enable Code BER	Boolean	**STD_ON**, STD_OFF	Enables the BER (Basic Encoding Rules) decoding functionality.

Table 6-1 Configuration parameter descriptions

6.2.2 User Configuration

Not relevant.

7 AUTOSAR Standard Compliance

Currently, the component is not considered in AUTOSAR. However, the component is designed based on AUTOSAR principles.

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
SysService_IpBase	Vector Informatik component name of the IPBase module

Table 7-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)

Table 7-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector-informatik.com