TCNOpen TRDP Light

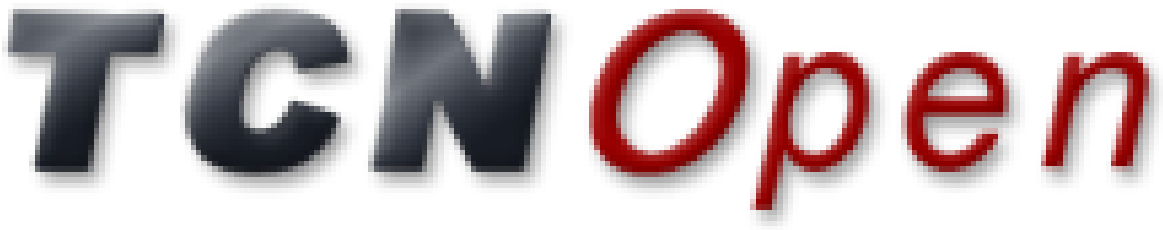
ReleaseV1.3

Generated by Doxygen 1.8.12

Contents

Chapter 1

The TRDP Light Library API Specification



1.1 General Information

1.1.1 Purpose

The TRDP protocol has been defined as the standard communication protocol in IP-enabled trains. It allows communication via process data (periodically transmitted data using UDP/IP) and message data (client - server messaging using UDP/IP or TCP/IP) This document describes the light API of the TRDP Library.

1.1.2 Scope

The intended audience of this document is the developers and project members of the TRDP project. TRDP Client Applications are programs using the TRDP protocol library to access the services of TRDP. Programmers developing such applications are the main target audience for this documentation.

1.1.3 Related documents

TCN-TRDP2-D-BOM-004-01 IEC61375-2-3_CD_ANNEXA Protocol definition of the TRDP standard

1.1.4 Abbreviations and Definitions

-*API* Application Programming Interface

-*ECN* Ethernet Consist Network

-*TRDP* Train Real-time Data Protocol

-*TCMS* Train Control Management System

1.2 Terminology

The API documented here is mainly concerned with three bodies of code:

- *TRDP Client Applications* (or 'client applications' for short): These are programs using the API to access the services of TRDP. Programmers developing such applications are the main target audience for this documentation.
- *TRDP Light Implementations* (or just 'TRDP implementation'): These are libraries realising the API as documented here. Programmers developing such implementations will find useful definitions about syntax and semantics of the API wihtin this documentation.
- *VOS Subsystem* (Virtual Operating System): An OS and hardware abstraction layer which offers memory, networking, threading, queues and debug functions. The VOS API is documented here.

The following diagram shows how these pieces of software are interrelated.

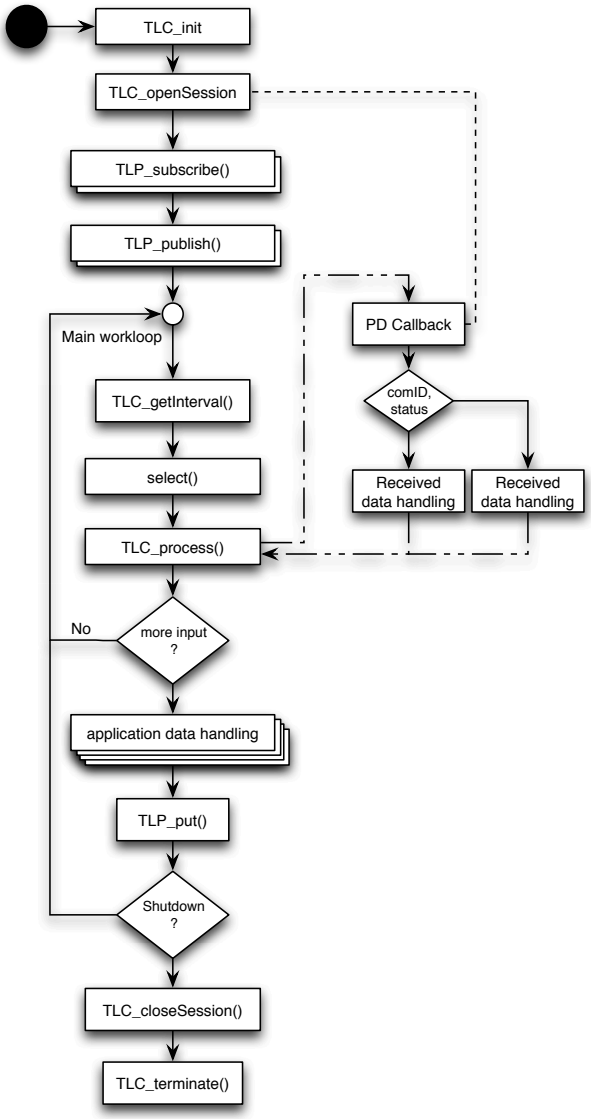


Figure 1.1 Sample client workflow

1.3 Conventions of the API

The API comprises a set of C header files that can also be used from client applications written in C++. These header files are contained in a directory named `trdp/api` and a subdirectory called `trdp/vos/api` with declarations not topical to TRDP but needed by the stack. Client applications shall include these header files like:

```
#include "trdp_if_light.h"
```

and, if VOS functions are needed, also the corresponding headers:

```
#include "vos_thread.h"
```

for example.

The subdirectory `trdp/doc` contains files needed for the API documentation.

Generally client application source code including API headers will only compile if the parent directory of the `trdp` directory is part of the include path of the used compiler. No other subdirectories of the API should be added to the compiler's include path.

The client API doesn't support a "catch-all" header file that includes all declarations in one step; rather the client application has to include individual headers for each feature set it wants to use.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 GNU_PACKED Struct Reference

Types for ETB control.

`#include <trdp_proto.h>`

Data Fields

- UINT8 trnVehNo

  vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5 value range: 0..63 a value of 0 indicates that this vehicle has been inserted by correction
- ANTIVALENT8 isLead

  vehicle is leading
- UINT8 leadDir

  vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2
- UINT8 vehOrient

  vehicle orientation 0 = not known (corrected vehicle) 1 = same as operational train direction 2 = inverse to operational train direction
- TRDP_SHORT_VERSION_T version

  telegram version information, main_version = 1, sub_version = 0
- UINT16 reserved01

  reserved (=0)
- UINT8 trnCstNo

  own TCN consist number (= 1..32)
- UINT8 reserved02

  reserved (=0)
- UINT8 ownOpCstNo

  own operational address (= 1..32) = 0 if unknown (e.g.
- UINT8 reserved03

  reserved (=0)
- UINT32 cstTopoCount

  Consist topology counter.
- UINT32 trnTopoCount

  Train directory topology counter.

- UINT32 opTrnTopoCount

  Operational Train topology counter.
- ANTIVALENT8 wasLead

  consist was leading, '01'B = false, '10'B = true
- ANTIVALENT8 reqLead

  leading request, '01'B = false, '10'B = true
- UINT8 reqLeadDir

  (request) leading direction, '01'B = consist direction 1, '10'B = consist direction 2
- ANTIVALENT8 accLead

  accept remote leading request, '01'B = false/not accepted, '10'B = true/accepted
- ANTIVALENT8 clearConfComp

  clear confirmed composition, '01'B = false, '10'B = true
- ANTIVALENT8 corrRequest

  request confirmation, '01'B = false, '10'B = true
- ANTIVALENT8 corrInfoSet

  correction info set, '01'B = false, '10'B = true
- ANTIVALENT8 compStored

  corrected composition stored, '01'B = false, '10'B = true
- ANTIVALENT8 sleepRequest

  request sleep mode, '01'B = false, '10'B = true
- UINT8 leadVehOfCst

  position of leading vehicle in consist, 0..31 (1: first vehicle in consist in Direction 1, 2: second vehicle, etc.)
- UINT8 reserved04

  reserved (=0)
- UINT16 reserved05

  reserved (=0)
- UINT8 reserved06

  reserved (=0)
- UINT8 confVehCnt

  number of confirmed vehicles in train (1..63)
- TRDP_CONF_VEHICLE_T confVehList [TRDP_MAX_VEH_CNT]

  dynamic ordered list of confirmed vehicles in train, starting with vehicle at train head, see sub-clause 5.3.3.2.6
- TRDP_ETB_CTRL_VDP_T safetyTrail

  ETBCTRL-VDP trailer, completely set to 0 == not used.
- UINT8 reserved01

  reserved (=0)
- TRDP_LABEL_T deviceName

  function device of ECSC which sends the telegram
- UINT8 inhibit

  inauguration inhibit 0 = no inhibit request 1 = inhibit request
- UINT8 leadingReq

  leading request 0 = no leading request 1 = leading request
- UINT8 leadingDir

  leading direction 0 = no leading request 1 = leading request direction 1 2 = leading request direction 2
- UINT8 sleepReq

  sleep request 0 = no sleep request 1 = sleep request
- UINT16 lifesign

  wrap-around counter, incremented with each produced datagram.
- UINT8 ecspState

  ECSP state indication 0 = ECSP not operational(initial value) 1 = ECSP in operation.
- UINT8 etbInhibit

inauguration inhibit indication 0 = n/a (default) 1 = inhibit not requested on ETB 2 = inhibit set on local ETBN 3 = inhibit set on remote ETBN 4 = inhibit set on local and remote ETBN

- UINT8 etbLength

  indicates train lengthening in case train inauguration is inhibit 0 = no lengthening (default) 1 = lengthening detected
- UINT8 etbShort

  indicates train shortening in case train inauguration is inhibit 0 = no shortening (default) 1 = shortening detected
- UINT16 reserved02

  reserved (=0)
- UINT8 etbLeadState

  indication of local consist leadership 5 = consist not leading (initial value) 6 = consist is leading requesting 9 = consist is leading 10 = leading conflict other values are not allowed
- UINT8 etbLeadDir

  direction of the leading end car in the local consist 0 = unknown (default) 1 = TCN direction 1 2 = TCN direction 2 other values are not allowed
- UINT8 ttdbSrvState

  TTDB server state indication 0 = n/a (initial value) 1 = Leader (default) 2 = Follower 3 = Error.
- UINT8 dnsSrvState

  DNS server state indication 0 = n/a (initial value) 1 = Leader (default) 2 = Follower 3 = Error.
- UINT8 trnDirState

  train directory state 1 = UNCONFIRMED 2 = CONFIRMED other values are not allowed
- UINT8 opTrnDirState

  train directory state 1 = INVALID 2 = VALID 4 = SHARED other values are not allowed
- UINT8 sleepCtrlState

  sleep control state (option) 0 = option not available 1 = RegularOperation 2 = WaitForSleepMode 3 = PrepareFor↩ SleepMode
- UINT8 sleepReqCnt

  number of sleep requests (option) value range: 0..63, not used = 0
- UINT32 opTrnTopoCnt

  operational train topology counter
- UINT8 command

  confirmation order 1 = confirmation/correction request 2 = un-confirmation request
- UINT16 confVehCnt

  number of confirmed vehicles in the train (1..63).
- TRDP_OP_VEHICLE_T confVehList [TRDP_MAX_VEH_CNT]

  ordered list of confirmed vehicles in the train, starting with vehicle at train head, see chapter 5.3.3.2.10.
- UINT8 status

  status of storing correction info 0 = correctly stored 1 = not stored
- UINT32 reqSafetyCode

  SC-32 value of the request message.
- UINT8 byPassCtrl

  ETBN bypass control 0 = no action (keep old state) 1 = no bypass 2 = activate bypass.
- UINT8 txCtrl

  ETBN transmission control 0 = no action (keep old state) 1 = activate sending on ETB (default) 2 = stop sending on ETB.
- UINT8 slCtrl

  sleep mode control (option) 0 = no action (keep old state) 1 = deactivate sleep mode 2 = activate sleep mode (line activity sensing)
- UINT8 etbnState

  state indication of the (active) ETBN 0 = ETBN not operational(initial value) 1 = ETBN in operation
- UINT8 etbnInaugState

  ETBN inauguration state as defined in IEC61375-2-5 0 = init 1 = not inaugurated 2 = inaugurated 3 = ready for inauguration.

- UINT8 etbnPosition

  position of the ETBN 0 = unknown (default) 1 = single node 2 = middle node 3 = end node TCN direction 1 4 = end node TCN direction 2
- UINT8 etbnRole

  ETBN node role as defined in IEC61375-2-5 0 = undefined 1 = master (redundancy leader) 2 = backup (redundancy follower) 3 = not redundant.
- BITSET8 etbLineState

  indication of ETB line status (FALSE == not trusted, TRUE == trusted) bit0 = line A ETBN direction 1 bit1 = line B ETBN direction 1 bit2 = line C ETBN direction 1 bit3 = line D ETBN direction 1 bit4 = line A ETBN direction 2 bit5 = line B ETBN direction 2 bit6 = line C ETBN direction 2 bit7 = line D ETBN direction 2
- UINT8 byPassState

  state of bypass function 0 = bypass disabled 1 = bypass enabled
- UINT8 slState

  sleep mode state (option) 0 = no sleep mode 1 = sleep mode active (line activity sensing)
- UINT32 etbTopoCnt

  ETB topography counter.
- TRDP_TRAIN_NET_DIR_T trnNetDir

  dynamic train info
- UINT8 ver

  Version - incremented for incompatible changes.
- UINT8 rel

  Release - incremented for compatible changes.
- UINT32 reserved01

  reserved (=0)
- TRDP_SHORT_VERSION_T userDataVersion

  version of the vital ETBCTRL telegram mainVersion = 1, subVersion = 0
- UINT32 safeSeqCount

  safe sequence counter, as defined in B.9
- UINT32 safetyCode

  checksum, as defined in B.9
- TRDP_UUID_T cstUUID

  UUID of the consist, provided by ETBN (TrainNetworkDirectory) Reference to static consist attributes 0 if not available (e.g.
- UINT32 cstTopoCnt

  consist topology counter provided with the CSTINFO 0 if no CSTINFO available
- UINT8 cstOrient

  consist orientation '01'B = same as train direction '10'B = inverse to train direction
- UINT8 cstCnt

  number of consists in train; range: 1..63
- TRDP_CONSIST_T cstList [TRDP_MAX_CST_CNT]

  consist list.
- UINT32 trnTopoCnt

  trnTopoCnt value ctrlType == 0: actual value ctrlType == 1: set to 0
- UINT8 etbId

  identification of the ETB the TTDB is computed for bit0: ETB0 (operational network) bit1: ETB1 (multimedia network) bit2: ETB2 (other network) bit3: ETB3 (other network)
- TRDP_LABEL_T vehId

  Unique vehicle identifier, application defined (e.g.
- UINT8 opVehNo

  operational vehicle sequence number in train value range 1..63
- UINT8 opCstNo

  operational consist number in train (1..63)

- UINT8 opCstOrient

  consist orientation '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
- TRDP_LABEL_T trnId

  train identifier, application defined (e.g.
- TRDP_LABEL_T trnOperator

  train operator, e.g.
- UINT32 crc

  sc-32 computed over record (seed value: 'FFFFFFFF'H)
- UINT8 opTrnOrient

  operational train orientation '00'B = unknown '01'B = same as train direction '10'B = inverse to train direction
- UINT8 opCstCnt

  number of consists in train (1..63)
- TRDP_OP_CONSIST_T opCstList [TRDP_MAX_CST_CNT]

  operational consist list starting with op.
- UINT8 reserved05

  reserved for future use (= 0)
- UINT8 opVehCnt

  number of vehicles in train (1..63)
- TRDP_OP_VEHICLE_T opVehList [TRDP_MAX_CST_CNT]

  operational vehicle list starting with op.
- TRDP_OP_TRAIN_DIR_STATE_T state

  operational state of the train
- UINT32 cstNetProp

  consist network properties bit0..1: consist orientation bit2..7: 0 bit8..13: ETBN Id bit14..15: 0 bit16..21: subnet Id bit24..29: CN Id bit30..31: 0
- UINT16 entryCnt

  number of entries in train network directory
- TRDP_TRAIN_NET_DIR_ENTRY_T trnNetDir [TRDP_MAX_CST_CNT]

  train network directory
- TRDP_OP_TRAIN_DIR_T opTrnDir

  operational directory
- TRDP_TRAIN_DIR_T trnDir

  train directory
- UINT32 sequenceCounter

  Unique counter (autom incremented)
- UINT16 protocolVersion

  fix value for compatibility (set by the API)
- UINT16 msgType

  of datagram: PD Request (0x5072) or PD_MSG (0x5064)
- UINT32 comId

  set by user: unique id
- UINT32 datasetLength

  length of the data to transmit 0...1432
- UINT32 reserved

  before used for ladder support
- UINT32 replyComId

  used in PD request
- UINT32 replyIpAddress

  used for PD request
- UINT32 frameCheckSum

CRC32 of header.

- INT32 replyStatus

  0 = OK
- UINT8 sessionID [16]

  UUID as a byte stream.
- UINT32 replyTimeout

  in us
- UINT8 sourceURI [32]

  User part of URI.
- UINT8 destinationURI [32]

  User part of URI.

4.1.1 Detailed Description

Types for ETB control.

TRDP message data header - network order and alignment.

TRDP process data header - network order and alignment.

Complete TTDB structure.

Train network directory structure.

Train network directory entry structure acc.

Operational Train directory status info structure.

Operational train structure.

Operational train directory state.

Operational consist structure.

Operational vehicle structure.

TCN train directory.

CSTINFO Control telegram.

TCN consist structure.

Version information for communication buffers.

to IEC61375-2-5

4.1.2 Field Documentation

4.1.2.1 confVehCnt

```
UINT16 GNU_PACKED::confVehCnt
```

number of confirmed vehicles in the train (1..63).

4.1.2.2 confVehList

```
TRDP_OP_VEHICLE_T GNU_PACKED::confVehList[TRDP_MAX_VEH_CNT]
```

ordered list of confirmed vehicles in the train, starting with vehicle at train head, see chapter 5.3.3.2.10.

Parameters 'isLead' and 'leadDir' to be set to 0

4.1.2.3 cstList

```
TRDP_CONSIST_T GNU_PACKED::cstList
```

consist list.

consist list ordered list starting with trnCstNo == 1 Note: This is a variable size array, only opCstCnt array elements are present on the network and for crc computation

If trnCstNo $>$ 0 this shall be an ordered list starting with trnCstNo == 1 (exactly the same as in structure TRAIN↩ _DIRECTORY). If trnCstNo == 0 it is not mandatory to list all consists (only consists which should send CSTINFO telegram). The parameters 'trnCstNo' and 'cstOrient' are optional and can be set to 0.

4.1.2.4 cstUUID

```
TRDP_UUID_T GNU_PACKED::cstUUID
```

UUID of the consist, provided by ETBN (TrainNetworkDirectory) Reference to static consist attributes 0 if not available (e.g.

unique consist identifier

Reference to static consist attributes, 0 if not available (e.g.

correction)

4.1.2.5 datasetLength

```
UINT32 GNU_PACKED::datasetLength
```

length of the data to transmit 0...1432

defined by user: length of data to transmit

4.1.2.6 deviceName

```
TRDP_LABEL_T GNU_PACKED::deviceName
```

function device of ECSC which sends the telegram

function device of ED which sends the telegram

4.1.2.7 etbId

`UINT8 GNU_PACKED::etbId`

identification of the ETB the TTDB is computed for bit0: ETB0 (operational network) bit1: ETB1 (multimedia network) bit2: ETB2 (other network) bit3: ETB3 (other network)

identification of the ETB the TTDB is computed for 0: ETB0 (operational network) 1: ETB1 (multimedia network) 2: ETB2 (other network) 3: ETB3 (other network)

4.1.2.8 etbTopoCnt

`UINT32 GNU_PACKED::etbTopoCnt`

ETB topography counter.

set by user: ETB to use, '0' for consist local traffic

train network directory CRC

4.1.2.9 inhibit

`UINT8 GNU_PACKED::inhibit`

inauguration inhibit 0 = no inhibit request 1 = inhibit request

ETBN inhibit 0 = no action (keep old state) 1 = no inhibit request 2 = inhibit request.

4.1.2.10 isLead

`ANTIVALENT8 GNU_PACKED::isLead`

vehicle is leading

consist contains leading vehicle, '01'B = false, '10'B = true

4.1.2.11 leadDir

`UINT8 GNU_PACKED::leadDir`

vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2

'vehicle leading direction 0 = not relevant 1 = leading direction 1 2 = leading direction 2

4.1.2.12 leadVehOfCst

`UINT8 GNU_PACKED::leadVehOfCst`

position of leading vehicle in consist, 0..31 (1: first vehicle in consist in Direction 1, 2: second vehicle, etc.)

position of leading vehicle in consist range 0...32 0 = not defined 1 = first vehicle in consist in direction 1 2 = second vehicle etc.

4.1.2.13 lifesign

```
UINT16 GNU_PACKED::lifesign
```

wrap-around counter, incremented with each produced datagram.

4.1.2.14 msgType

```
UINT16 GNU_PACKED::msgType
```

of datagram: PD Request (0x5072) or PD_MSG (0x5064)

of datagram: Mn, Mr, Mp, Mq, Mc or Me

4.1.2.15 opCstList

```
TRDP_OP_CONSIST_T GNU_PACKED::opCstList[TRDP_MAX_CST_CNT]
```

operational consist list starting with op.

consist #1 Note: This is a variable size array, only opCstCnt array elements are present

4.1.2.16 opTrnDirState

```
UINT8 GNU_PACKED::opTrnDirState
```

train directory state 1 = INVALID 2 = VALID 4 = SHARED other values are not allowed

Operational train directory status: '01'B == invalid, '10'B == valid, '100'B == shared.

4.1.2.17 opTrnTopoCnt

```
UINT32 GNU_PACKED::opTrnTopoCnt
```

operational train topology counter

set by user: direction/side critical, '0' if ignored

operational train topology counter computed as defined in 5.3.3.2.16 (seed value : trnTopoCnt)

operational train topology counter set to 0 if opTrnDirState == invalid

operational train topocounter value of the operational train directory the correction is based on

4.1.2.18 opVehList

```
TRDP_OP_VEHICLE_T GNU_PACKED::opVehList[TRDP_MAX_CST_CNT]
```

operational vehicle list starting with op.

vehicle #1 Note: This is a variable size array, only opCstCnt array elements are present

4.1.2.19 ownOpCstNo

`UINT8 GNU_PACKED::ownOpCstNo`

own operational address (= 1..32) = 0 if unknown (e.g.

operational consist number the vehicle belongs to

after Inauguration)

4.1.2.20 protocolVersion

`UINT16 GNU_PACKED::protocolVersion`

fix value for compatibility (set by the API)

fix value for compatibility

4.1.2.21 reserved01 [1/2]

`UINT16 GNU_PACKED::reserved01`

reserved (=0)

reserved for future use (= 0)

4.1.2.22 reserved01 [2/2]

`UINT8 GNU_PACKED::reserved01`

reserved (=0)

reserved for future use (= 0)

4.1.2.23 reserved02 [1/2]

`UINT16 GNU_PACKED::reserved02`

reserved (=0)

reserved (= 0)

reserved for future use (= 0)

4.1.2.24 reserved02 [2/2]

`UINT16 GNU_PACKED::reserved02`

reserved (=0)

reserved (= 0)

4.1.2.25 reserved03

```
UINT8 GNU_PACKED::reserved03
```

reserved (=0)

reserved for future use (= 0)

4.1.2.26 reserved04

```
UINT8 GNU_PACKED::reserved04
```

reserved (=0)

reserved for future use (= 0)

4.1.2.27 reserved06

```
UINT8 GNU_PACKED::reserved06
```

reserved (=0)

reserved for future use (= 0)

4.1.2.28 safetyTrail

```
TRDP_ETB_CTRL_VDP_T GNU_PACKED::safetyTrail
```

ETBCTRL-VDP trailer, completely set to 0 == not used.

ETBCTRL-VDP trailer, parameter 'safeSequCount' == 0 completely set to 0 == not used.

ETBCTRL-VDP trailer, parameter 'safeSequCount' == 0 completely set to 0 == SDTv2 not used.

ETBCTRL-VDP trailer, completely set to 0 == SDTv2 not used.

4.1.2.29 trnCstNo

```
UINT8 GNU_PACKED::trnCstNo
```

own TCN consist number (= 1..32)

train consist number telegram control type 0 = with trnTopoCnt tracking 1 = without trnTopoCnt tracking

Sequence number of consist in train (1..63)

4.1.2.30 trnDirState

`UINT8 GNU_PACKED::trnDirState`

train directory state 1 = UNCONFIRMED 2 = CONFIRMED other values are not allowed

TTDB status: '01'B == unconfirmed, '10'B == confirmed.

4.1.2.31 trnId

`TRDP_LABEL_T GNU_PACKED::trnId`

train identifier, application defined (e.g.

'ICE75', 'IC346'), informal

4.1.2.32 trnNetDir

`TRDP_TRAIN_NET_DIR_T GNU_PACKED::trnNetDir`

dynamic train info

network directory

4.1.2.33 trnOperator

`TRDP_LABEL_T GNU_PACKED::trnOperator`

train operator, e.g.

'trenitalia.it', informal

4.1.2.34 trnTopoCnt

`UINT32 GNU_PACKED::trnTopoCnt`

trnTopoCnt value ctrlType == 0: actual value ctrlType == 1: set to 0

computed as defined in 5.3.3.2.16 (seed value: etbTopoCnt)

4.1.2.35 trnVehNo

`UINT8 GNU_PACKED::trnVehNo`

vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5 value range: 0..63 a value of 0 indicates that this vehicle has been inserted by correction

vehicle sequence number within the train with vehicle 01 being the first vehicle in ETB reference direction 1 as defined in IEC61375-2-5, value range: 1..63, a value of 0 indicates that this vehicle has been inserted by correction

4.1.2.36 vehId

`TRDP_LABEL_T GNU_PACKED::vehId`

Unique vehicle identifier, application defined (e.g.

UIC Identifier)

4.1.2.37 vehOrient

`UINT8 GNU_PACKED::vehOrient`

vehicle orientation 0 = not known (corrected vehicle) 1 = same as operational train direction 2 = inverse to operational train direction

vehicle orientation, '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction

4.1.2.38 version

`TRDP_SHORT_VERSION_T GNU_PACKED::version`

telegram version information, main_version = 1, sub_version = 0

Train info structure version.

TrainDirectoryState data structure version parameter 'mainVersion' shall be set to 1.

TrainDirectory data structure version parameter 'mainVersion' shall be set to 1.

Consist Info Control structure version parameter 'mainVersion' shall be set to 1.

The documentation for this struct was generated from the following files:

- tau_ctrl_types.h
- tau_tti_types.h
- trdp_proto.h

4.2 TRDP_CLTR_CST_INFO_T Struct Reference

Closed train consists information.

`#include <tau_tti_types.h>`

Data Fields

- TRDP_UUID_T cltrCstUUID

  closed train consist UUID
- UINT8 cltrCstOrient

  closed train consist orientation '01'B = same as closed train direction '10'B = inverse to closed train direction
- UINT8 cltrCstNo

  sequence number of the consist within the closed train, value range 1..32
- UINT16 reserved01

  reserved for future use (= 0)

4.2.1 Detailed Description

Closed train consists information.

The documentation for this struct was generated from the following file:

- tau_tti_types.h

4.3 TRDP_COMID_DSID_MAP_T Struct Reference

ComId - data set mapping element definition.

```
#include <trdp_types.h>
```

Data Fields

- UINT32 comId

  comId
- UINT32 datasetId

  corresponding dataset Id

4.3.1 Detailed Description

ComId - data set mapping element definition.

The documentation for this struct was generated from the following file:

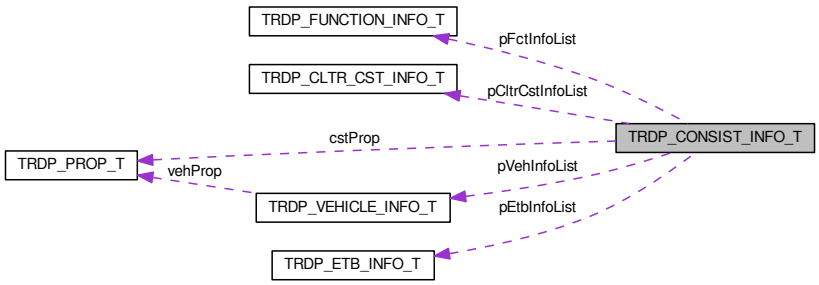
- trdp_types.h

4.4 TRDP_CONSIST_INFO_T Struct Reference

consist information structure

`#include <tau_tti_types.h>`

Collaboration diagram for TRDP_CONSIST_INFO_T:



Data Fields

- TRDP_SHORT_VERSION_T version

  ConsistInfo data structure version, application defined mainVersion = 1, subVersion = 0.
- UINT8 cstClass

  consist info classification 1 = (single) consist 2 = closed train 3 = closed train consist
- UINT8 reserved01

  reserved for future use (= 0)
- TRDP_LABEL_T cstId

  application defined consist identifier, e.g.
- TRDP_LABEL_T cstType

  consist type, application defined
- TRDP_LABEL_T cstOwner

  consist owner, e.g.
- TRDP_UUID_T cstUUID

  consist UUID
- UINT32 reserved02

  reserved for future use (= 0)
- TRDP_PROP_T cstProp

  static consist properties
- UINT16 reserved03

  reserved for future use (= 0)
- UINT16 etbCnt

  number of ETB's, range: 1..4
- TRDP_ETB_INFO_T * pEtbInfoList

  ETB information list for the consist Ordered list starting with lowest etbId.
- UINT16 reserved04

  reserved for future use (= 0)
- UINT16 vehCnt

number of vehicles in consist 1..32

- TRDP_VEHICLE_INFO_T * pVehInfoList

  vehicle info list for the vehicles in the consist Ordered list starting with cstVehNo==1

- UINT16 reserved05

  reserved for future use (= 0)

- UINT16 fctCnt

  number of consist functions value range 0..1024

- TRDP_FUNCTION_INFO_T * pFctInfoList

  function info list for the functions in consist lexicographical ordered by fctName

- UINT16 reserved06

  reserved for future use (= 0)

- UINT16 cltrCstCnt

  number of original consists in closed train value range: 0..32, 0 = consist is no closed train

- TRDP_CLTR_CST_INFO_T * pCltrCstInfoList

  info on closed train composition Ordered list starting with cltrCstNo == 1

- UINT32 cstTopoCnt

  consist topology counter computed as defined in 5.3.3.2.16, seed value: 'FFFFFFFF'H

4.4.1 Detailed Description

consist information structure

4.4.2 Field Documentation

4.4.2.1 cstId

```
TRDP_LABEL_T TRDP_CONSIST_INFO_T::cstId
```

application defined consist identifier, e.g.

UIC identifier

4.4.2.2 cstOwner

```
TRDP_LABEL_T TRDP_CONSIST_INFO_T::cstOwner
```

consist owner, e.g.

"trenitalia.it", "sncf.fr", "db.de"

The documentation for this struct was generated from the following file:

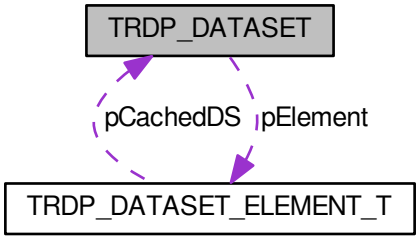
- tau_tti_types.h

4.5 TRDP_DATASET Struct Reference

Dataset definition.

```
#include <trdp_types.h>
```

Collaboration diagram for TRDP_DATASET:



Data Fields

- UINT32 id

  dataset identifier > 1000
- UINT16 reserved1

  Reserved for future use, must be zero.
- UINT16 numElement

  Number of elements.
- TRDP_DATASET_ELEMENT_T pElement [ ]

  Pointer to a dataset element, used as array.

4.5.1 Detailed Description

Dataset definition.

The documentation for this struct was generated from the following file:

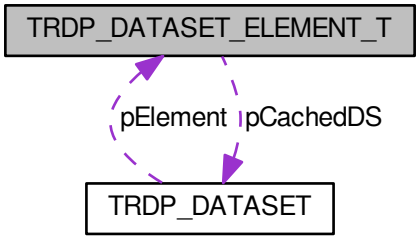
- trdp_types.h

4.6 TRDP_DATASET_ELEMENT_T Struct Reference

Dataset element definition.

`#include <trdp_types.h>`

Collaboration diagram for TRDP_DATASET_ELEMENT_T:



Data Fields

- UINT32 type
  Data type (TRDP_DATA_TYPE_T 1...99) or dataset id > 1000.
- UINT32 size
  Number of items or TDRP_VAR_SIZE (0)
- CHAR8 * unit
  Unit text for visualisation.
- REAL32 scale
  Factor for visualisation.
- INT32 offset
  *Offset for visualisation (val = scale * x + offset)*
- struct TRDP_DATASET * pCachedDS
  Used internally for marshalling speed-up.

4.6.1 Detailed Description

Dataset element definition.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.7 TRDP_DBG_CONFIG_T Struct Reference

Control for debug output device/file on application level.

`#include <tau_xml.h>`

Data Fields

- TRDP_DBG_OPTION_T option
  Debug printout options for application use.
- UINT32 maxFileSize
  Maximal file size.
- TRDP_FILE_NAME_T fileName
  Debug file name and path.

4.7.1 Detailed Description

Control for debug output device/file on application level.

The documentation for this struct was generated from the following file:

- tau_xml.h

4.8 TRDP_ETB_INFO_T Struct Reference

Types for train configuration information.

`#include <tau_tti_types.h>`

Data Fields

- UINT8 etbId
  identification of train backbone; value range: 0..3
- UINT8 cnCnt
  number of CNs within consist connected to this ETB value range 1..16 referring to cnId 0..15 acc.
- UINT16 reserved01
  reserved for future use (= 0)

4.8.1 Detailed Description

Types for train configuration information.

ETB information

4.8.2 Field Documentation

4.8.2.1 cnCnt

`UINT8 TRDP_ETB_INFO_T::cnCnt`

number of CNs within consist connected to this ETB value range 1..16 referring to cnId 0..15 acc.

IEC61375-2-5

The documentation for this struct was generated from the following file:

- tau_tti_types.h

4.9 TRDP_FUNCTION_INFO_T Struct Reference

function/device information structure

`#include <tau_tti_types.h>`

Data Fields

- TRDP_LABEL_T fctName
  function device or group label
- UINT16 fctId
  host identification of the function device or group as defined in IEC 61375-2-5, application defined.
- BOOL8 grp
  is a function group and will be resolved as IP multicast address
- UINT8 reserved01
  reserved for future use (= 0)
- UINT8 cstVehNo
  Sequence number of the vehicle in the consist the function belongs to.
- UINT8 etbId
  number of connected train backbone.
- UINT8 cnId
  identifier of connected consist network in the consist, related to the etbId.
- UINT8 reserved02
  reserved for future use (= 0)

4.9.1 Detailed Description

function/device information structure

4.9.2 Field Documentation

4.9.2.1 cnId

`UINT8 TRDP_FUNCTION_INFO_T::cnId`

identifier of connected consist network in the consist, related to the etbId.

Value range: 0..31

4.9.2.2 cstVehNo

`UINT8 TRDP_FUNCTION_INFO_T::cstVehNo`

Sequence number of the vehicle in the consist the function belongs to.

Value range: 1..16, 0 = not defined

4.9.2.3 etbId

`UINT8 TRDP_FUNCTION_INFO_T::etbId`

number of connected train backbone.

Value range: 0..3

4.9.2.4 fctId

`UINT16 TRDP_FUNCTION_INFO_T::fctId`

host identification of the function device or group as defined in IEC 61375-2-5, application defined.

Value range: 1..16383 (device), 256..16383 (group)

The documentation for this struct was generated from the following file:

- tau_tti_types.h

4.10 TRDP_LIST_STATISTICS_T Struct Reference

Information about a particular MD listener.

`#include <trdp_types.h>`

Data Fields

- UINT32 comId
  ComId to listen to.
- TRDP_URI_USER_T uri
  URI user part to listen to.
- TRDP_IP_ADDR_T joinedAddr
  Joined IP address.
- UINT32 callBack
  Call back function if used.
- UINT32 userRef
  User reference if used.
- UINT32 numSessions
  Number of sessions.

4.10.1 Detailed Description

Information about a particular MD listener.

The documentation for this struct was generated from the following file:

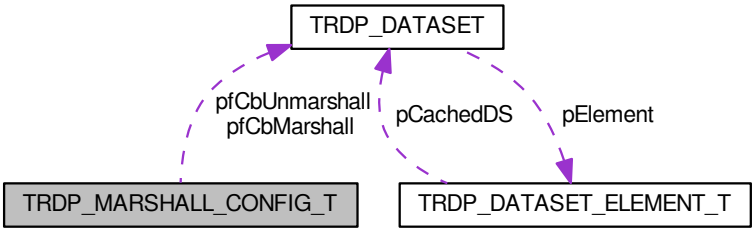
- trdp_types.h

4.11 TRDP_MARSHALL_CONFIG_T Struct Reference

Marshaling/unmarshalling configuration.

`#include <trdp_types.h>`

Collaboration diagram for TRDP_MARSHALL_CONFIG_T:



Data Fields

- TRDP_MARSHALL_T pfCbMarshall

  Pointer to marshall callback function.
- TRDP_UNMARSHALL_T pfCbUnmarshall

  Pointer to unmarshall callback function.
- void * pRefCon

  Pointer to user context for call back.

4.11.1 Detailed Description

Marshaling/unmarshalling configuration.

The documentation for this struct was generated from the following file:

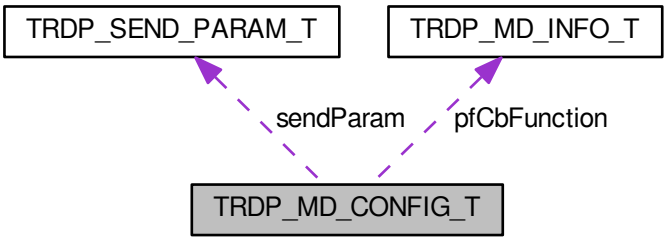
- trdp_types.h

4.12 TRDP_MD_CONFIG_T Struct Reference

Default MD configuration.

`#include <trdp_types.h>`

Collaboration diagram for TRDP_MD_CONFIG_T:



Data Fields

- TRDP_MD_CALLBACK_T pfCbFunction
  Pointer to MD callback function.
- void * pRefCon
  Pointer to user context for call back.
- TRDP_SEND_PARAM_T sendParam
  Default send parameters.
- TRDP_FLAGS_T flags
  Default flags for MD packets.
- UINT32 replyTimeout
  Default reply timeout in us.
- UINT32 confirmTimeout
  Default confirmation timeout in us.
- UINT32 connectTimeout
  Default connection timeout in us.
- UINT32 sendingTimeout
  Default sending timeout in us.
- UINT16 udpPort
  Port to be used for UDP MD communication.
- UINT16 tcpPort
  Port to be used for TCP MD communication.
- UINT32 maxNumSessions
  Maximal number of replier sessions.

4.12.1 Detailed Description

Default MD configuration.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.13 TRDP_MD_INFO_T Struct Reference

Message data info from received telegram; allows the application to generate responses.

`#include <trdp_types.h>`

Data Fields

- TRDP_IP_ADDR_T srcIpAddr
  source IP address for filtering
- TRDP_IP_ADDR_T destIpAddr
  destination IP address for filtering
- UINT32 seqCount
  sequence counter
- UINT16 protVersion
  Protocol version.
- TRDP_MSG_T msgType
  Protocol ('PD', 'MD', ...)
- UINT32 comId
  ComID.
- UINT32 etbTopoCnt
  received topocount
- UINT32 opTrnTopoCnt
  received topocount
- BOOL8 aboutToDie
  session is about to die
- UINT32 numRepliesQuery
  number of ReplyQuery received
- UINT32 numConfirmSent
  number of Confirm sent
- UINT32 numConfirmTimeout
  number of Confirm Timeouts (incremented by listeners
- UINT16 userStatus
  error code, user stat
- TRDP_REPLY_STATUS_T replyStatus
  reply status
- TRDP_UUID_T sessionId
  for response
- UINT32 replyTimeout
  reply timeout in us given with the request
- TRDP_URI_USER_T srcUserURI
  source URI user part from MD header
- TRDP_URI_HOST_T srcHostURI
  source URI host part (unused)
- TRDP_URI_USER_T destUserURI
  destination URI user part from MD header
- TRDP_URI_HOST_T destHostURI
  destination URI host part (unused)
- UINT32 numExpReplies
  number of expected replies, 0 if unknown

- UINT32 numReplies
  actual number of replies for the request
- const void * pUserRef
  User reference given with the local call.
- TRDP_ERR_T resultCode
  error code

4.13.1 Detailed Description

Message data info from received telegram; allows the application to generate responses.

Note: Not all fields are relevant for each message type!

The documentation for this struct was generated from the following file:

- trdp_types.h

4.14 TRDP_MD_STATISTICS_T Struct Reference

Structure containing all general MD statistics information.

`#include <trdp_types.h>`

Data Fields

- UINT32 defQos
  default QoS for MD
- UINT32 defTtl
  default TTL for MD
- UINT32 defReplyTimeout
  default reply timeout in us for MD
- UINT32 defConfirmTimeout
  default confirm timeout in us for MD
- UINT32 numList
  number of listeners
- UINT32 numRcv
  number of received MD packets
- UINT32 numCrcErr
  number of received MD packets with CRC err
- UINT32 numProtErr
  number of received MD packets with protocol err
- UINT32 numTopoErr
  number of received MD packets with wrong topo count
- UINT32 numNoListener
  number of received MD packets without listener
- UINT32 numReplyTimeout
  number of reply timeouts
- UINT32 numConfirmTimeout
  number of confirm timeouts
- UINT32 numSend
  number of sent MD packets

4.14.1 Detailed Description

Structure containing all general MD statistics information.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.15 TRDP_MEM_CONFIG_T Struct Reference

Enumeration type for memory pre-fragmentation, reuse of VOS definition.

`#include <trdp_types.h>`

Data Fields

- UINT8 * p
  pointer to static or allocated memory
- UINT32 size
  size of static or allocated memory
- UINT32 prealloc [VOS_MEM_NBLOCKSIZES]
  memory block structure

4.15.1 Detailed Description

Enumeration type for memory pre-fragmentation, reuse of VOS definition.

Structure describing memory (and its pre-fragmentation)

The documentation for this struct was generated from the following file:

- trdp_types.h

4.16 TRDP_MEM_STATISTICS_T Struct Reference

Structure containing all general memory statistics information.

`#include <trdp_types.h>`

Data Fields

- UINT32 total
  total memory size
- UINT32 free
  free memory size
- UINT32 minFree
  minimal free memory size in statistics interval
- UINT32 numAllocBlocks
  allocated memory blocks
- UINT32 numAllocErr
  allocation errors
- UINT32 numFreeErr
  free errors
- UINT32 blockSize [VOS_MEM_NBLOCKSIZES]
  preallocated memory blocks
- UINT32 usedBlockSize [VOS_MEM_NBLOCKSIZES]
  used memory blocks

4.16.1 Detailed Description

Structure containing all general memory statistics information.

The documentation for this struct was generated from the following file:

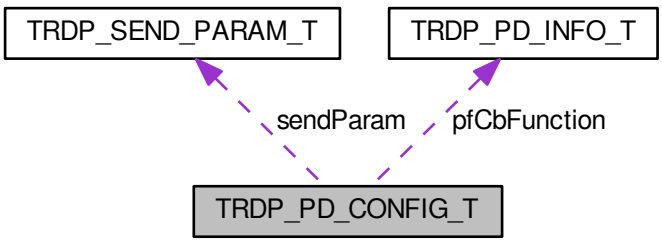
- trdp_types.h

4.17 TRDP_PD_CONFIG_T Struct Reference

Default PD configuration.

```
#include <trdp_types.h>
```

Collaboration diagram for TRDP_PD_CONFIG_T:



Data Fields

- TRDP_PD_CALLBACK_T pfCbFunction
  Pointer to PD callback function.
- void * pRefCon
  Pointer to user context for call back.
- TRDP_SEND_PARAM_T sendParam
  Default send parameters.
- TRDP_FLAGS_T flags
  Default flags for PD packets.
- UINT32 timeout
  Default timeout in us.
- TRDP_TO_BEHAVIOR_T toBehavior
  Default timeout behavior.
- UINT16 port
  Port to be used for PD communication.

4.17.1 Detailed Description

Default PD configuration.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.18 TRDP_PD_INFO_T Struct Reference

Process data info from received telegram; allows the application to generate responses.

```
#include <trdp_types.h>
```

Data Fields

- TRDP_IP_ADDR_T srcIpAddr
  source IP address for filtering
- TRDP_IP_ADDR_T destIpAddr
  destination IP address for filtering
- UINT32 seqCount
  sequence counter
- UINT16 protVersion
  Protocol version.
- TRDP_MSG_T msgType
  Protocol ('PD', 'MD', ...)
- UINT32 comId
  ComID.
- UINT32 etbTopoCnt
  received ETB topocount
- UINT32 opTrnTopoCnt

received operational train directory topocount

- UINT32 replyComId

  ComID for reply (request only)
- TRDP_IP_ADDR_T replyIpAddr

  IP address for reply (request only)
- const void * pUserRef

  User reference given with the local subscribe.
- TRDP_ERR_T resultCode

  error code
- TRDP_URI_HOST_T srcHostURI

  source URI host part (unused)
- TRDP_URI_HOST_T destHostURI

  destination URI host part (unused)

4.18.1 Detailed Description

Process data info from received telegram; allows the application to generate responses.

Note: Not all fields are relevant for each message type!

The documentation for this struct was generated from the following file:

- trdp_types.h

4.19 TRDP_PD_STATISTICS_T Struct Reference

Structure containing all general PD statistics information.

```
#include <trdp_types.h>
```

Data Fields

- UINT32 defQos

  default QoS for PD
- UINT32 defTtl

  default TTL for PD
- UINT32 defTimeout

  default timeout in us for PD
- UINT32 numSubs

  number of subscribed ComId's
- UINT32 numPub

  number of published ComId's
- UINT32 numRcv

  number of received PD packets
- UINT32 numCrcErr

  number of received PD packets with CRC err
- UINT32 numProtErr

  number of received PD packets with protocol err

- UINT32 numTopoErr
  number of received PD packets with wrong topo count
- UINT32 numNoSubs
  number of received PD push packets without subscription
- UINT32 numNoPub
  number of received PD pull packets without publisher
- UINT32 numTimeout
  number of PD timeouts
- UINT32 numSend
  number of sent PD packets
- UINT32 numMissed
  number of packets skipped

4.19.1 Detailed Description

Structure containing all general PD statistics information.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.20 TRDP_PROCESS_CONFIG_T Struct Reference

Various flags/general TRDP options for library initialization.

`#include <trdp_types.h>`

Data Fields

- TRDP_LABEL_T hostName
  Host name.
- TRDP_LABEL_T leaderName
  Leader name dependant on redundancy concept.
- UINT32 cycleTime
  TRDP main process cycle time in us.
- UINT32 priority
  TRDP main process cycle time (0-255, 0=default, 255=highest)
- TRDP_OPTION_T options
  TRDP options.

4.20.1 Detailed Description

Various flags/general TRDP options for library initialization.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.21 TRDP_PROP_T Struct Reference

Application defined properties.

`#include <tau_tti_types.h>`

Data Fields

- TRDP_SHORT_VERSION_T ver

  properties version information, application defined
- UINT16 len

  properties length in number of octets, application defined, must be a multiple of 4 octets for alignment reasons value range: 0..32768
- UINT8 prop [1]

  properties, application defined

4.21.1 Detailed Description

Application defined properties.

The documentation for this struct was generated from the following file:

- tau_tti_types.h

4.22 TRDP_PUB_STATISTICS_T Struct Reference

Table containing particular PD publishing information.

`#include <trdp_types.h>`

Data Fields

- UINT32 comId

  Published ComId.
- TRDP_IP_ADDR_T destAddr

  IP address of destination for this publishing.
- UINT32 cycle

  Publishing cycle in us.
- UINT32 redId

  Redundancy group id.
- UINT32 redState

  Redundant state.Leader or Follower.
- UINT32 numPut

  Number of packet updates.
- UINT32 numSend

  Number of packets sent out.

4.22.1 Detailed Description

Table containing particular PD publishing information.

4.22.2 Field Documentation

4.22.2.1 destAddr

```
TRDP_IP_ADDR_T TRDP_PUB_STATISTICS_T::destAddr
```

IP address of destination for this publishing.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.23 TRDP_RED_STATISTICS_T Struct Reference

A table containing PD redundant group information.

```
#include <trdp_types.h>
```

Data Fields

- UINT32 id
  Redundant Id.
- TRDP_RED_STATE_T state
  Redundant state.Leader or Follower.

4.23.1 Detailed Description

A table containing PD redundant group information.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.24 TRDP_SDT_PAR_T Struct Reference

Types to read out the XML configuration.

```
#include <tau_xml.h>
```

Data Fields

- UINT32 smi1

  Safe message identifier - unique for this message at consist level.
- UINT32 smi2

  Safe message identifier - unique for this message at consist level.
- UINT32 cmThr

  Channel monitoring threshold.
- UINT16 udv

  User data version.
- UINT16 rxPeriod

  Sink cycle time.
- UINT16 txPeriod

  Source cycle time.
- UINT16 nGuard

  Initial timeout cycles.
- UINT8 nrxSafe

  Timout cycles.
- UINT8 reserved1

  Reserved for future use.
- UINT16 reserved2

  Reserved for future use.

4.24.1 Detailed Description

Types to read out the XML configuration.

The documentation for this struct was generated from the following file:

- tau_xml.h

4.25 TRDP_SEND_PARAM_T Struct Reference

Quality/type of service and time to live.

```
#include <trdp_types.h>
```

Data Fields

- UINT8 qos

  Quality of service (default should be 5 for PD and 3 for MD)
- UINT8 ttl

  Time to live (default should be 64)
- UINT8 retries

  Retries from XML file.

4.25.1 Detailed Description

Quality/type of service and time to live.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.26 TRDP_STATISTICS_REQUEST_T Struct Reference

TRDP statistics type definitions.

```
#include <trdp_types.h>
```

Data Fields

- UINT32 comId

  ComId to request: 35...41.

4.26.1 Detailed Description

TRDP statistics type definitions.

Statistical data regarding the former info provided via SNMP the following information was left out/can be implemented additionally using MD:

- PD subscr table: ComId, sourceIpAddr, destIpAddr, cbFct?, timout, toBehavior, counter
- PD publish table: ComId, destIpAddr, redId, redState cycle, ttl, qos, counter
- PD join table: joined MC address table
- MD listener table: ComId destIpAddr, destUri, cbFct?, counter
- Memory usageStructure containing comId for MD statistics request (ComId 32).

The documentation for this struct was generated from the following file:

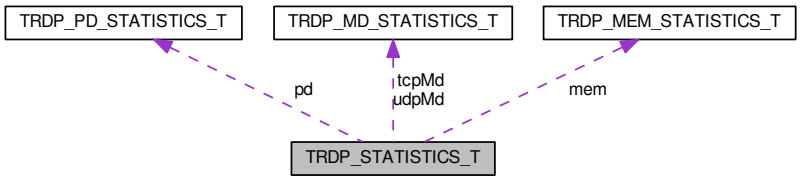
- trdp_types.h

4.27 TRDP_STATISTICS_T Struct Reference

Structure containing all general memory, PD and MD statistics information.

`#include <trdp_types.h>`

Collaboration diagram for TRDP_STATISTICS_T:



Data Fields

- UINT32 version
  TRDP version.
- TIMEDATE64 timeStamp
  actual time stamp
- TIMEDATE32 upTime
  time in sec since last initialisation
- TIMEDATE32 statisticTime
  time in sec since last reset of statistics
- TRDP_LABEL_T hostName
  host name
- TRDP_LABEL_T leaderName
  leader host name
- TRDP_IP_ADDR_T ownIpAddr
  own IP address
- TRDP_IP_ADDR_T leaderIpAddr
  leader IP address
- UINT32 processPrio
  priority of TRDP process
- UINT32 processCycle
  cycle time of TRDP process in microseconds
- UINT32 numJoin
  number of joins
- UINT32 numRed
  number of redundancy groups
- TRDP_MEM_STATISTICS_T mem
  memory statistics
- TRDP_PD_STATISTICS_T pd
  pd statistics
- TRDP_MD_STATISTICS_T udpMd
  UDP md statistics.
- TRDP_MD_STATISTICS_T tcpMd
  TCP md statistics.

4.27.1 Detailed Description

Structure containing all general memory, PD and MD statistics information.

The documentation for this struct was generated from the following file:

- trdp_types.h

4.28 TRDP_SUBS_STATISTICS_T Struct Reference

Table containing particular PD subscription information.

`#include <trdp_types.h>`

Data Fields

- UINT32 comId
  Subscribed ComId.
- TRDP_IP_ADDR_T joinedAddr
  Joined IP address.
- TRDP_IP_ADDR_T filterAddr
  Filter IP address, i.e IP address of the sender for this subscription, 0.0.0.0 in case all senders.
- UINT32 callBack
  call back function if used
- UINT32 userRef
  User reference if used.
- UINT32 timeout
  Time-out value in us.
- TRDP_ERR_T status
  Receive status information TRDP_NO_ERR, TRDP_TIMEOUT_ERR.
- TRDP_TO_BEHAVIOR_T toBehav
  Behavior at time-out.
- UINT32 numRecv
  Number of packets received for this subscription.
- UINT32 numMissed
  number of packets skipped for this subscription

4.28.1 Detailed Description

Table containing particular PD subscription information.

4.28.2 Field Documentation

4.28.2.1 filterAddr

`TRDP_IP_ADDR_T TRDP_SUBS_STATISTICS_T::filterAddr`

Filter IP address, i.e IP address of the sender for this subscription, 0.0.0.0 in case all senders.

4.28.2.2 timeout

```
UINT32 TRDP_SUBS_STATISTICS_T::timeout
```

Time-out value in us.

0 = No time-out supervision

4.28.2.3 toBehav

```
TRDP_TO_BEHAVIOR_T TRDP_SUBS_STATISTICS_T::toBehav
```

Behavior at time-out.

Set data to zero / keep last value

The documentation for this struct was generated from the following file:

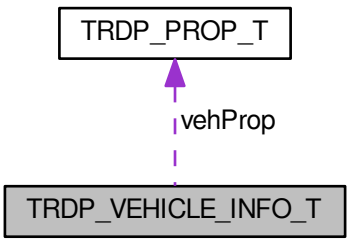
- trdp_types.h

4.29 TRDP_VEHICLE_INFO_T Struct Reference

vehicle information structure

```
#include <tau_tti_types.h>
```

Collaboration diagram for TRDP_VEHICLE_INFO_T:



Data Fields

- TRDP_LABEL_T vehId
  vehicle identifier label,application defined (e.g.
- TRDP_LABEL_T vehType
  vehicle type,application defined
- UINT8 vehOrient
  vehicle orientation '01'B = same as consist direction '10'B = inverse to consist direction
- UINT8 cstVehNo
  Sequence number of vehicle in consist(1..16)
- ANTIVALENT8 tractVeh
  vehicle is a traction vehicle '01'B = vehicle is not a traction vehicle '10'B = vehicle is a traction vehicle
- UINT8 reserved01
  for future use (= 0)
- TRDP_PROP_T vehProp
  static vehicle properties

4.29.1 Detailed Description

vehicle information structure

4.29.2 Field Documentation

4.29.2.1 vehId

```
TRDP_LABEL_T TRDP_VEHICLE_INFO_T::vehId
```

vehicle identifier label,application defined (e.g.

UIC vehicle identification number) vehId of vehicle with vehNo==1 is used also as cstId

The documentation for this struct was generated from the following file:

- tau_tti_types.h

4.30 TRDP_XML_DOC_HANDLE_T Struct Reference

Parsed XML document handle.

```
#include <tau_xml.h>
```

Data Fields

- struct XML_HANDLE * pXmlDocument
  XML document context.

4.30.1 Detailed Description

Parsed XML document handle.

The documentation for this struct was generated from the following file:

- tau_xml.h

4.31 VOS_SOCK_OPT_T Struct Reference

Common socket options.

`#include <vos_sock.h>`

Data Fields

- UINT8 qos
  quality/type of service 0...7
- UINT8 ttl
  time to live for unicast (default 64)
- UINT8 ttl_multicast
  time to live for multicast
- BOOL8 reuseAddrPort
  allow reuse of address and port
- BOOL8 nonBlocking
  use non blocking calls
- BOOL8 no_mc_loop
  no multicast loop back
- BOOL8 no_udp_crc
  supress udp crc computation

4.31.1 Detailed Description

Common socket options.

The documentation for this struct was generated from the following file:

- vos_sock.h

4.32 VOS_TIME_T Struct Reference

Timer value compatible with timeval / select.

`#include <vos_types.h>`

Data Fields

- UINT32 tv_sec
  full seconds
- INT32 tv_usec
  Micro seconds (max.

4.32.1 Detailed Description

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage Assume 32 Bit system, if not defined

4.32.2 Field Documentation

4.32.2.1 tv_usec

```
INT32 VOS_TIME_T::tv_usec
```

Micro seconds (max.

value 999999)

The documentation for this struct was generated from the following file:

- vos_types.h

4.33 VOS_VERSION_T Struct Reference

Version information.

```
#include <vos_types.h>
```

Data Fields

- UINT8 ver
  Version - incremented for incompatible changes.
- UINT8 rel
  Release - incremented for compatible changes.
- UINT8 upd
  Update - incremented for bug fixes.
- UINT8 evo
  Evolution - incremented for build.

4.33.1 Detailed Description

Version information.

The documentation for this struct was generated from the following file:

- vos_types.h

Chapter 5

File Documentation

5.1 iec61375-2-3.h File Reference

TTDB, CSTINFO Frame typedefs, Telegram definitions.

Macros

- #define ETB_CTRL_COMID 1
  ETB Control telegram.
- #define ETB_CTRL_CYC 500
  0.5s
- #define ETB_CTRL_TO 3000
  3s
- #define CSTINFO_COMID 2
  Consist Info telegram (Message data notification 'Mn')
- #define CSTINFOCTRL_COMID 3
  Consist Info control/request telegram (Message data notification 'Mn')
- #define TTDB_STATUS_COMID 100
  TTDB manager telegram PD.
- #define TTDB_STATUS_CYC 1000
  Push.
- #define TTDB_STATUS_TO 5000
  5s
- #define TTDB_OP_DIR_INFO_COMID 101
  TTDB manager telegram MD: Push the OP_TRAIN_DIRECTORY.
- #define TTDB_OP_DIR_INFO_DS "TTDB_OP_TRAIN_DIRECTORY_INFO"
  OP_TRAIN_DIRECTORY.
- #define TTDB_TRN_DIR_REQ_COMID 102
  TTDB manager telegram MD: Get the TRAIN_DIRECTORY.
- #define TTDB_TRN_DIR_REQ_TO 3000
  3s timeout
- #define TTDB_TRN_DIR_REP_COMID 103
  MD reply.
- #define TTDB_TRN_DIR_REP_DS "TTDB_TRAIN_DIRECTORY_INFO_REPLY"
  TRAIN_DIRECTORY.

- #define TTDB_STAT_CST_REQ_COMID 104

  TTDB manager telegram MD: Get the static consist information.
- #define TTDB_STAT_CST_REQ_TO 3000

  3s timeout
- #define TTDB_STAT_CST_REP_DS "TTDB_STATIC_CONSIST_INFO_REPLY"

  CONSIST_INFO.
- #define TTDB_NET_DIR_REQ_COMID 106

  TTDB manager telegram MD: Get the NETWORK_TRAIN_DIRECTORY.
- #define TTDB_NET_DIR_REQ_TO 3000

  3s timeout
- #define TTDB_NET_DIR_REP_COMID 107

  MD reply.
- #define TTDB_NET_DIR_REP_DS "TTDB_TRAIN_NETWORK_DIRECTORY_INFO_REPLY"

  TRAIN_NETWORK_DIRECTORY.
- #define TTDB_OP_DIR_INFO_REQ_COMID 108

  TTDB manager telegram MD: Get the OP_TRAIN_DIRECTORY.
- #define TTDB_OP_DIR_INFO_REQ_TO 3000

  3s timeout
- #define TTDB_OP_DIR_INFO_REP_DS "TTDB_OP_TRAIN_DIR_INFO"

  OP_TRAIN_DIRECTORY.
- #define TTDB_READ_CMPLT_REQ_COMID 110

  TTDB manager telegram MD: Get the TTDB.
- #define TTDB_READ_CMPLT_REQ_DS "TTDB_READ_COMPLETE_REQUEST"

  ETBx.
- #define TTDB_READ_CMPLT_REQ_TO 3000

  3s timeout
- #define TTDB_READ_CMPLT_REP_COMID 111

  MD reply.
- #define TTDB_READ_CMPLT_REP_DS "TTDB_READ_COMPLETE_REPLY"

  TRDP_READ_COMPLETE_REPLY_T.
- #define ECSP_CTRL_COMID 120

  ECSP Control telegram.
- #define ECSP_CTRL_CYC 1000

  1s
- #define ECSP_CTRL_TO 5000

  5s
- #define ECSP_CTRL_DEST_URI "devECSP.anyVeh.lCst.lClTrn.lTrn"

  10.0.0.1
- #define ECSP_STATUS_COMID 121

  ECSP status telegram.
- #define ECSP_STATUS_CYC 1000

  1s
- #define ECSP_STATUS_TO 5000

  5s
- #define ECSP_STATUS_DEST_URI "devECSC.anyVeh.lCst.lClTrn.lTrn"

  10.0.0.100
- #define ECSP_CONF_REQ_COMID 122

  ECSP Confirmation Request telegram MD:
- #define ECSP_CONF_REQ_URI "devECSP.anyVeh.lCst.lClTrn.lTrn"

  10.0.0.1
- #define ETBN_CTRL_REQ_COMID 130

ETBN Control & Status Telegram MD.

- #define ETBN_CTRL_REQ_DS "ETBN_CTRL"

  ETBx.
- #define ETBN_CTRL_REQ_TO 3000

  3s timeout
- #define ETBN_CTRL_REP_DS "ETBN_STATUS"

  ETBN status reply.
- #define ETBN_TRN_NET_DIR_REQ_COMID 132

  ETBN Control Telegram MD.
- #define ETBN_TRN_NET_DIR_REQ_TO 3000

  3s timeout
- #define TCN_DNS_REQ_COMID 140

  TCN-DNS Request Telegram MD.
- #define TCN_DNS_REQ_TO 3000

  3s timeout

5.1.1 Detailed Description

TTDB, CSTINFO Frame typedefs, Telegram definitions.

Note

Project: TCNOpen TRDP

Author

Bernd Loehr, NewTec GmbH, 2015-09-11

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.

Id

```
BL 2016-05-04: Ticket #118: Fix defines to match IEC IS 2015
```

5.1.2 Macro Definition Documentation

5.1.2.1 TTDB_NET_DIR_REQ_COMID

```
#define TTDB_NET_DIR_REQ_COMID 106
```

TTDB manager telegram MD: Get the NETWORK_TRAIN_DIRECTORY.

MD request

5.1.2.2 TTDB_OP_DIR_INFO_COMID

```
#define TTDB_OP_DIR_INFO_COMID 101
```

TTDB manager telegram MD: Push the OP_TRAIN_DIRECTORY.

MD notification

5.1.2.3 TTDB_STAT_CST_REQ_COMID

```
#define TTDB_STAT_CST_REQ_COMID 104
```

TTDB manager telegram MD: Get the static consist information.

MD request

5.1.2.4 TTDB_TRN_DIR_REQ_COMID

```
#define TTDB_TRN_DIR_REQ_COMID 102
```

TTDB manager telegram MD: Get the TRAIN_DIRECTORY.

MD request

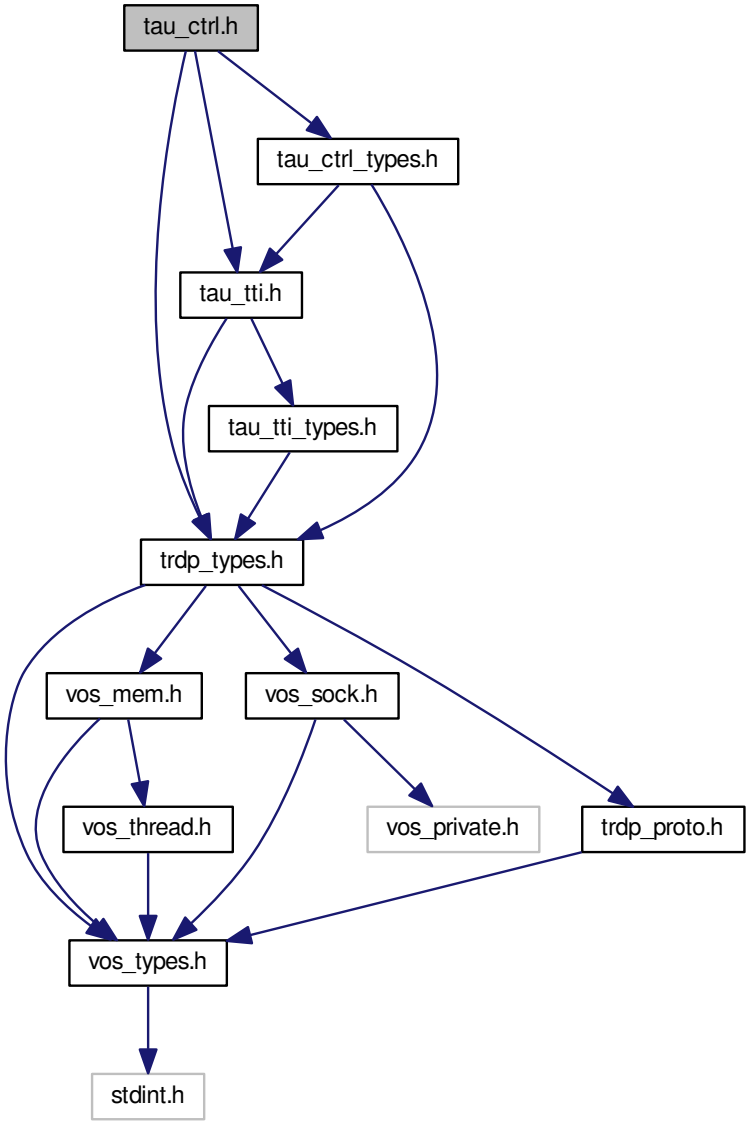
5.2 tau_ctrl.h File Reference

TRDP utility interface definitions.

```
#include "trdp_types.h"
#include "tau_tti.h"
```

```
#include "tau_ctrl_types.h"
```

Include dependency graph for tau_ctrl.h:



Functions

- EXT_DECL TRDP_ERR_T tau_initEcspCtrl (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_↩ T ecspIpAddr)

  Function to init ECSP control interface.
- EXT_DECL TRDP_ERR_T tau_terminateEcspCtrl (TRDP_APP_SESSION_T appHandle)

  Function to close ECSP control interface.
- EXT_DECL TRDP_ERR_T tau_setEcspCtrl (TRDP_APP_SESSION_T appHandle, TRDP_ECSP_CTRL_T ∗pEcspCtrl)

  Function to set ECSP control information.

- EXT_DECL TRDP_ERR_T tau_getEcspStat (TRDP_APP_SESSION_T appHandle, TRDP_ECSP_STAT_T *pEcspStat, TRDP_PD_INFO_T *pPdInfo)

  Function to get ECSP status information.
- EXT_DECL TRDP_ERR_T tau_requestEcspConfirm (TRDP_APP_SESSION_T appHandle, const void *pUserRef, TRDP_MD_CALLBACK_T pfCbFunction, TRDP_ECSP_CONF_REQUEST_T *pEcspConf↩ Request)

  Function for ECSP confirmation/correction request, reply will be received via call back.

5.2.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- ETB control

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

Id

5.2.2 Function Documentation

5.2.2.1 tau_getEcspStat()

```
EXT_DECL TRDP_ERR_T tau_getEcspStat (
            TRDP_APP_SESSION_T appHandle,
            TRDP_ECSP_STAT_T * pEcspStat,
            TRDP_PD_INFO_T * pPdInfo )
```

Function to get ECSP status information.

Parameters

in	*appHandle*	Application Handle
in,out	*pEcspStat*	Pointer to the ECSP status structure
in,out	*pPdInfo*	Pointer to PD status information

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	module not initialised
TRDP_PARAM_ERR	Parameter error

5.2.2.2 tau_initEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_initEcspCtrl (
             TRDP_APP_SESSION_T appHandle,
             TRDP_IP_ADDR_T ecspIpAddr )
```

Function to init ECSP control interface.

Parameters

in	*appHandle*	Application handle
in	*ecspIpAddr*	ECSP address

Return values

TRDP_NO_ERR	no error
TRDP_INIT_ERR	initialisation error

5.2.2.3 tau_requestEcspConfirm()

```
EXT_DECL TRDP_ERR_T tau_requestEcspConfirm (
             TRDP_APP_SESSION_T appHandle,
             const void * pUserRef,
             TRDP_MD_CALLBACK_T pfCbFunction,
             TRDP_ECSP_CONF_REQUEST_T * pEcspConfRequest )
```

Function for ECSP confirmation/correction request, reply will be received via call back.

Parameters

in	*appHandle*	Application Handle
in	*pUserRef*	user reference returned with reply
in	*pfCbFunction*	Pointer to callback function, NULL for default
in	*pEcspConfRequest*	Pointer to confirmation data

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	module not initialised
TRDP_PARAM_ERR	Parameter error

5.2.2.4 tau_setEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_setEcspCtrl (
             TRDP_APP_SESSION_T appHandle,
             TRDP_ECSP_CTRL_T * pEcspCtrl )
```

Function to set ECSP control information.

Parameters

in	*appHandle*	Application handle
in	*pEcspCtrl*	Pointer to the ECSP control structure

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	module not initialised
TRDP_PARAM_ERR	Parameter error

5.2.2.5 tau_terminateEcspCtrl()

```
EXT_DECL TRDP_ERR_T tau_terminateEcspCtrl (
             TRDP_APP_SESSION_T appHandle )
```

Function to close ECSP control interface.

Parameters

in	*appHandle*	Application handle

Return values

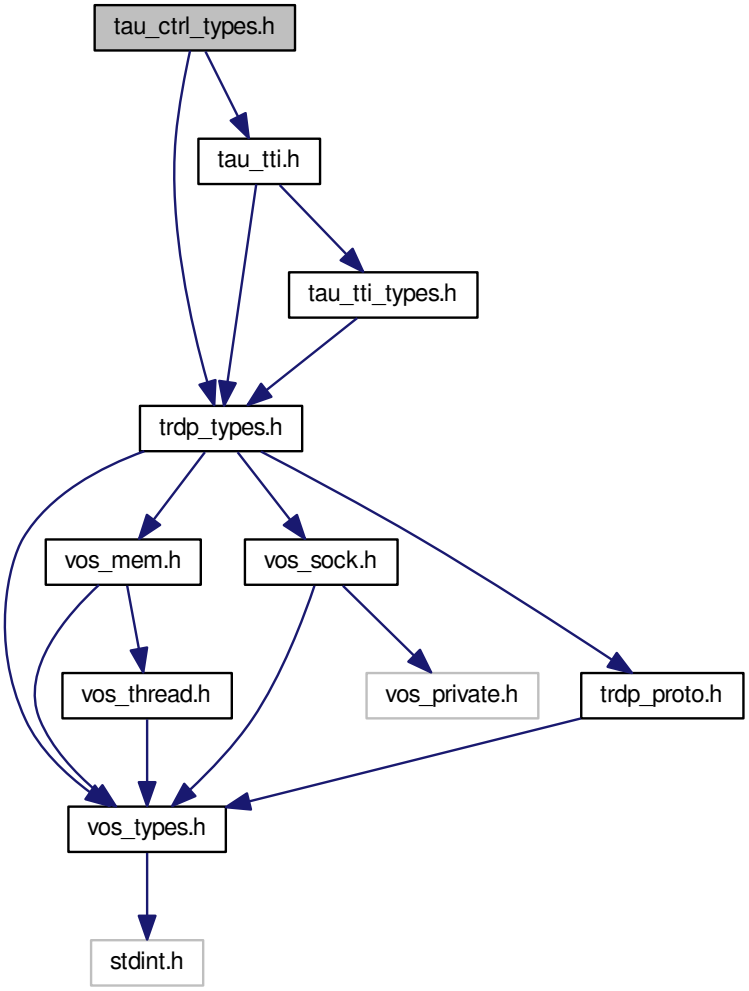
TRDP_NO_ERR	no error
TRDP_UNKNOWN_ERR	undefined error

5.3 tau_ctrl_types.h File Reference

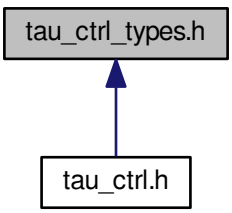
TRDP utility interface definitions.

```
#include "trdp_types.h"
#include "tau_tti.h"
```

Include dependency graph for tau_ctrl_types.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.

5.3.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following

- ETB control type definitions acc. to IEC61375-2-3

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

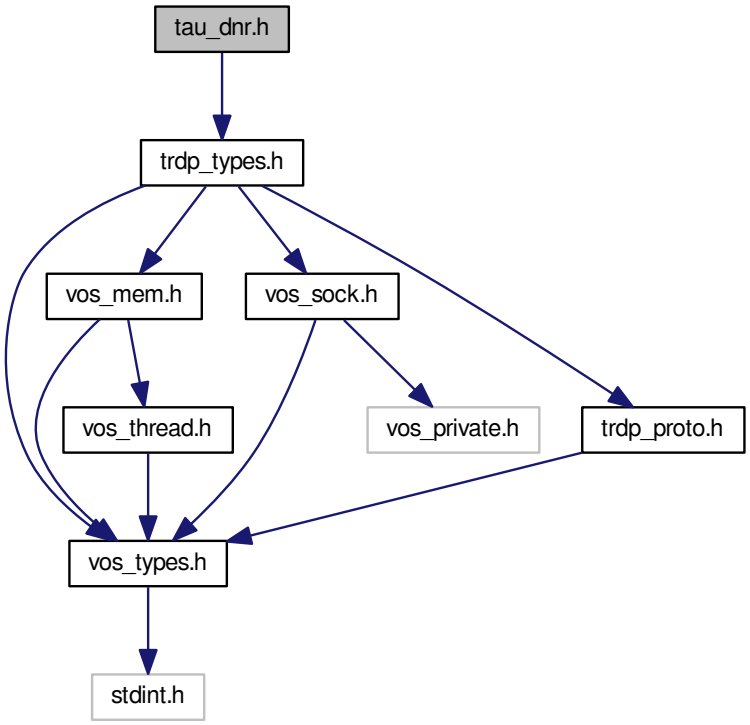
Id

5.4 tau_dnr.h File Reference

TRDP utility interface definitions.

```
#include "trdp_types.h"
```

Include dependency graph for tau_dnr.h:



Functions

- EXT_DECL TRDP_ERR_T tau_initDnr (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_T dnsIp↩ Addr, UINT16 dnsPort, const CHAR8 ∗hostsFileName)

  Function to init DNR.
- EXT_DECL void tau_deInitDnr (TRDP_APP_SESSION_T appHandle)

  Release any resources allocated by DNR.
- EXT_DECL TRDP_DNR_STATE_T tau_DNRstatus (TRDP_APP_SESSION_T appHandle)

  Function to get the status of DNR.
- EXT_DECL TRDP_ERR_T tau_getOwnIds (TRDP_APP_SESSION_T appHandle, TRDP_LABEL_T devId, TRDP_LABEL_T vehId, TRDP_LABEL_T cstId)

  Who am I ?.
- EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr (TRDP_APP_SESSION_T appHandle)

  Function to get the own IP address.
- EXT_DECL TRDP_ERR_T tau_uri2Addr (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_T ∗p↩ Addr, const TRDP_URI_T pUri)

  Function to convert a URI to an IP address.
- EXT_DECL TRDP_ERR_T tau_addr2Uri (TRDP_APP_SESSION_T appHandle, TRDP_URI_HOST_T pUri, TRDP_IP_ADDR_T addr)

  Function to convert an IP address to a URI.

5.4.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- IP - URI address translation

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

Id

```
BL 2015-12-14: Ticket #8: DNR client
```

5.4.2 Function Documentation

5.4.2.1 tau_addr2Uri()

```
EXT_DECL TRDP_ERR_T tau_addr2Uri (
             TRDP_APP_SESSION_T appHandle,
             TRDP_URI_HOST_T pUri,
             TRDP_IP_ADDR_T addr )
```

Function to convert an IP address to a URI.

Receives an IP-Address and translates it into the host part of the corresponding URI. Both unicast and multicast addresses are accepted.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pUri*	Pointer to a string to return the URI host part
in	*addr*	IP address, 0==own address

Return values

TRDP_NO_ERR	no error

Return values

TRDP_PARAM_ERR	Parameter error

5.4.2.2 tau_deInitDnr()

```
EXT_DECL void tau_deInitDnr (
            TRDP_APP_SESSION_T appHandle )
```

Release any resources allocated by DNR.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().

Return values

none	

5.4.2.3 tau_DNRstatus()

```
EXT_DECL TRDP_DNR_STATE_T tau_DNRstatus (
            TRDP_APP_SESSION_T appHandle )
```

Function to get the status of DNR.

Parameters

in	*appHandle*	Handle returned by tlc_openSession()

Return values

TRDP_DNR_NOT_AVAILABLE	no error
TRDP_DNR_UNKNOWN	enabled, but cache is empty
TRDP_DNR_ACTIVE	enabled, cache has values
TRDP_DNR_HOSTSFILE	enabled, hostsfile used (static mode)

5.4.2.4 tau_getOwnAddr()

```
EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr (
            TRDP_APP_SESSION_T appHandle )
```

Function to get the own IP address.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().

Return values

own	IP address

5.4.2.5 tau_getOwnIds()

```
EXT_DECL TRDP_ERR_T tau_getOwnIds (
             TRDP_APP_SESSION_T appHandle,
             TRDP_LABEL_T devId,
             TRDP_LABEL_T vehId,
             TRDP_LABEL_T cstId )
```

Who am I ?.

Realizes a kind of 'Who am I' function. It is used to determine the own identifiers (i.e. the own labels), which may be used as host part of the own fully qualified domain name.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*devId*	Returns the device label (host name)
out	*vehId*	Returns the vehicle label
out	*cstId*	Returns the consist label

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.4.2.6 tau_initDnr()

```
EXT_DECL TRDP_ERR_T tau_initDnr (
             TRDP_APP_SESSION_T appHandle,
             TRDP_IP_ADDR_T dnsIpAddr,
             UINT16 dnsPort,
             const CHAR8 * hostsFileName )
```

Function to init DNR.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
in	*dnsIpAddr*	DNS/ECSP IP address.
in	*dnsPort*	DNS port number.
in	*hostsFileName*	Optional host file name as ECSP replacement/addition.

Return values

TRDP_NO_ERR	no error
TRDP_INIT_ERR	initialisation error

5.4.2.7 tau_uri2Addr()

```
EXT_DECL TRDP_ERR_T tau_uri2Addr (
             TRDP_APP_SESSION_T appHandle,
             TRDP_IP_ADDR_T * pAddr,
             const TRDP_URI_T pUri )
```

Function to convert a URI to an IP address.

Receives a URI as input variable and translates this URI to an IP-Address. The URI may specify either a unicast or a multicast IP-Address. The caller may specify a topographic counter, which will be checked.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pAddr*	Pointer to return the IP address
in	*pUri*	Pointer to a URI or an IP Address string, NULL==own URI

Return values

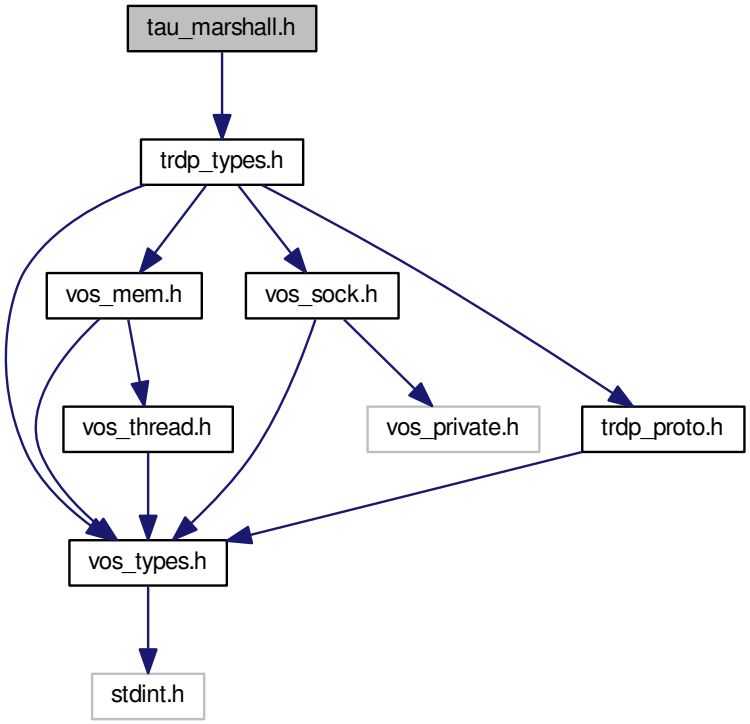
TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.5 tau_marshall.h File Reference

TRDP utility interface definitions.

```
#include "trdp_types.h"
```

Include dependency graph for tau_marshall.h:



Functions

- EXT_DECL TRDP_ERR_T tau_initMarshall (void **ppRefCon, UINT32 numComId, TRDP_COMID_DSID↩ _MAP_T *pComIdDsIdMap, UINT32 numDataSet, TRDP_DATASET_T *pDataset[ ])

  Types for marshalling / unmarshalling.
- EXT_DECL TRDP_ERR_T tau_marshall (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, TRDP_DATASET_T **ppDSPointer)

  marshall function.
- EXT_DECL TRDP_ERR_T tau_marshallDs (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, TRDP_DATASET_T **ppDSPointer)

  marshall data set function.
- EXT_DECL TRDP_ERR_T tau_unmarshall (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, TRDP_DATASET_T **ppDSPointer)

  unmarshall function.
- EXT_DECL TRDP_ERR_T tau_unmarshallDs (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDest, UINT32 *pDestSize, TRDP_DATASET_T **ppDSPointer)

  unmarshall data set function.
- EXT_DECL TRDP_ERR_T tau_calcDatasetSize (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT32 src↩ Size, UINT32 *pDestSize, TRDP_DATASET_T **ppDSPointer)

  Calculate data set size by given data set id.
- EXT_DECL TRDP_ERR_T tau_calcDatasetSizeByComId (void *pRefCon, UINT32 comId, UINT8 *pSrc, U↩ INT32 srcSize, UINT32 *pDestSize, TRDP_DATASET_T **ppDSPointer)

  Calculate data set size by given ComId.

5.5.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- marshalling/unmarshalling

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss

Remarks

Id

```
BL 2015-12-14: Ticket #33: source size check for marshalling
```

5.5.2 Function Documentation

5.5.2.1 tau_calcDatasetSize()

```
EXT_DECL TRDP_ERR_T tau_calcDatasetSize (
             void * pRefCon,
             UINT32 dsId,
             UINT8 * pSrc,
             UINT32 srcSize,
             UINT32 * pDestSize,
             TRDP_DATASET_T ** ppDSPointer )
```

Calculate data set size by given data set id.

Parameters

in	*pRefCon*	Pointer to user context
in	*dsId*	Dataset id to identify the structure out of a configuration
in	*pSrc*	Pointer to received original message
in	*srcSize*	size of the source buffer
out	*pDestSize*	Pointer to the size of the data set
in,out	*ppDSPointer*	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

TRDP_NO_ERR	no error
TRDP_INIT_ERR	marshalling not initialised
TRDP_PARAM_ERR	data set id not existing

5.5.2.2 tau_calcDatasetSizeByComId()

```
EXT_DECL TRDP_ERR_T tau_calcDatasetSizeByComId (
             void * pRefCon,
             UINT32 comId,
             UINT8 * pSrc,
             UINT32 srcSize,
             UINT32 * pDestSize,
             TRDP_DATASET_T ** ppDSPointer )
```

Calculate data set size by given ComId.

Parameters

in	*pRefCon*	Pointer to user context
in	*comId*	ComId id to identify the structure out of a configuration
in	*pSrc*	Pointer to received original message
in	*srcSize*	size of the source buffer
out	*pDestSize*	Pointer to the size of the data set
in,out	*ppDSPointer*	pointer to pointer to cached dataset, set NULL if not used, set content NULL if unknown

Return values

TRDP_NO_ERR	no error
TRDP_INIT_ERR	marshalling not initialised
TRDP_PARAM_ERR	data set id not existing

5.5.2.3 tau_initMarshall()

```
EXT_DECL TRDP_ERR_T tau_initMarshall (
             void ** ppRefCon,
             UINT32 numComId,
             TRDP_COMID_DSID_MAP_T * pComIdDsIdMap,
             UINT32 numDataSet,
             TRDP_DATASET_T * pDataset[] )
```

Types for marshalling / unmarshalling.

Function to initialise the marshalling/unmarshalling.

Parameters

in,out	*ppRefCon*	Returns a pointer to be used for the reference context of marshalling/unmarshalling
in	*numComId*	Number of datasets found in the configuration
in	*pComIdDsIdMap*	Pointer to an array of structures of type TRDP_DATASET_T
in	*numDataSet*	Number of datasets found in the configuration
in	*pDataset*	Pointer to an array of pointers to structures of type TRDP_DATASET_T

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_PARAM_ERR	Parameter error

5.5.2.4 tau_marshall()

```
EXT_DECL TRDP_ERR_T tau_marshall (
             void * pRefCon,
             UINT32 comId,
             UINT8 * pSrc,
             UINT32 srcSize,
             UINT8 * pDest,
             UINT32 * pDestSize,
             TRDP_DATASET_T ** ppDSPointer )
```

marshall function.

Parameters

in	*pRefCon*	pointer to user context
in	*comId*	ComId to identify the structure out of a configuration
in	*pSrc*	pointer to received original message
in	*srcSize*	size of the source buffer
in	*pDest*	pointer to a buffer for the treated message
in,out	*pDestSize*	size of the provide buffer / size of the treated message
in,out	*ppDSPointer*	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_INIT_ERR	marshalling not initialised
TRDP_COMID_ERR	comid not existing
TRDP_PARAM_ERR	Parameter error

5.5.2.5 tau_marshallDs()

```
EXT_DECL TRDP_ERR_T tau_marshallDs (
             void * pRefCon,
             UINT32 dsId,
             UINT8 * pSrc,
             UINT32 srcSize,
             UINT8 * pDest,
             UINT32 * pDestSize,
             TRDP_DATASET_T ** ppDSPointer )
```

marshall data set function.

Parameters

in	*pRefCon*	pointer to user context
in	*dsId*	Data set id to identify the structure out of a configuration
in	*pSrc*	pointer to received original message
in	*srcSize*	size of the source buffer
in	*pDest*	pointer to a buffer for the treated message
in,out	*pDestSize*	size of the provide buffer / size of the treated message
in,out	*ppDSPointer*	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_INIT_ERR	marshalling not initialised
TRDP_COMID_ERR	comid not existing
TRDP_PARAM_ERR	Parameter error

5.5.2.6 tau_unmarshall()

```
EXT_DECL TRDP_ERR_T tau_unmarshall (
             void * pRefCon,
             UINT32 comId,
             UINT8 * pSrc,
             UINT32 srcSize,
             UINT8 * pDest,
             UINT32 * pDestSize,
             TRDP_DATASET_T ** ppDSPointer )
```

unmarshall function.

Parameters

in	*pRefCon*	pointer to user context
in	*comId*	ComId to identify the structure out of a configuration
in	*pSrc*	pointer to received original message
in	*srcSize*	size of the source buffer
in	*pDest*	pointer to a buffer for the treated message
in,out	*pDestSize*	size of the provide buffer / size of the treated message
in,out	*ppDSPointer*	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_INIT_ERR	marshalling not initialised
TRDP_COMID_ERR	comid not existing

5.5.2.7 tau_unmarshallDs()

```
EXT_DECL TRDP_ERR_T tau_unmarshallDs (
             void * pRefCon,
             UINT32 dsId,
             UINT8 * pSrc,
             UINT32 srcSize,
             UINT8 * pDest,
             UINT32 * pDestSize,
             TRDP_DATASET_T ** ppDSPointer )
```

unmarshall data set function.

Parameters

in	*pRefCon*	pointer to user context
in	*dsId*	Data set id to identify the structure out of a configuration
in	*pSrc*	pointer to received original message
in	*srcSize*	size of the source buffer
in	*pDest*	pointer to a buffer for the treated message
in,out	*pDestSize*	size of the provide buffer / size of the treated message
in,out	*ppDSPointer*	pointer to pointer to cached dataset set NULL if not used, set content NULL if unknown

Return values

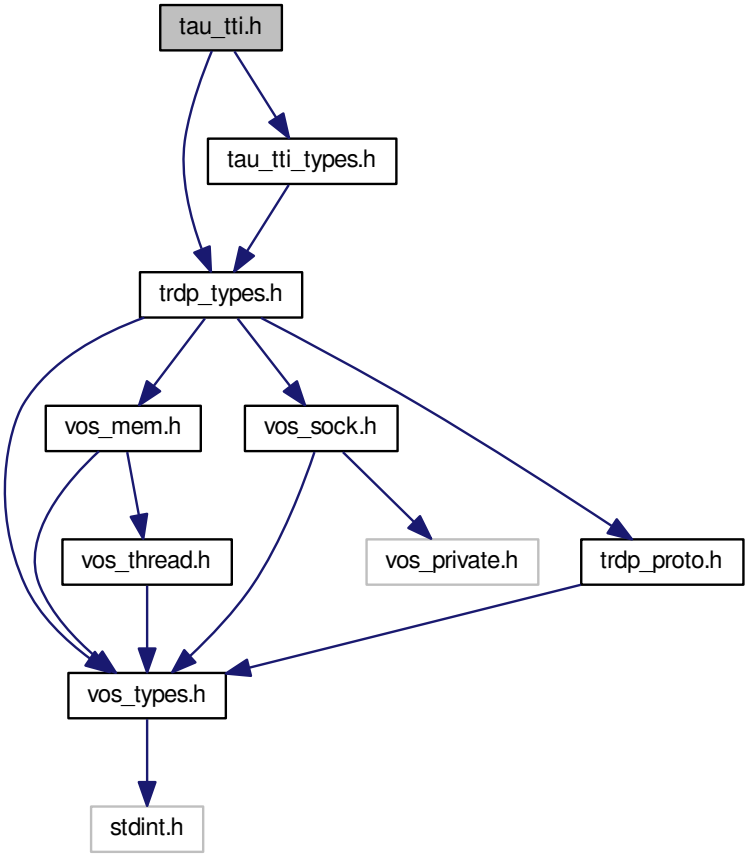
TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_INIT_ERR	marshalling not initialised
TRDP_COMID_ERR	comid not existing

5.6 tau_tti.h File Reference

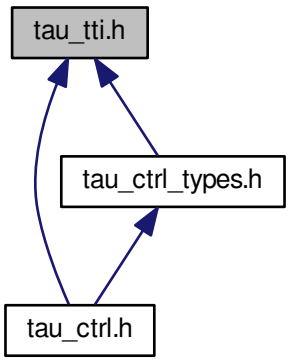
TRDP utility interface definitions.

```
#include "trdp_types.h"
#include "tau_tti_types.h"
```

Include dependency graph for tau_tti.h:



This graph shows which files directly or indirectly include this file:



Functions

- EXT_DECL TRDP_ERR_T tau_initTTIaccess (TRDP_APP_SESSION_T appHandle, VOS_SEMA_T user↩ Action, TRDP_IP_ADDR_T ecspIpAddr, CHAR8 ∗hostsFileName)

  Function to init TTI access.
- EXT_DECL void tau_deInitTTI (TRDP_APP_SESSION_T appHandle)

  Function to terminate TTI access.
- EXT_DECL TRDP_ERR_T tau_getOpTrDirectory (TRDP_APP_SESSION_T appHandle, TRDP_OP_TRA↩ IN_DIR_STATE_T ∗pOpTrDirState, TRDP_OP_TRAIN_DIR_T ∗pOpTrDir)

  Function to retrieve the operational train directory state.
- EXT_DECL TRDP_ERR_T tau_getOpTrnDirectoryStatusInfo (TRDP_APP_SESSION_T appHandle, TRD↩ P_OP_TRAIN_DIR_STATUS_INFO_T ∗pOpTrnDirStatusInfo)

  Function to retrieve the operational train directory state info.
- EXT_DECL TRDP_ERR_T tau_getTrDirectory (TRDP_APP_SESSION_T appHandle, TRDP_TRAIN_DIR↩ _T ∗pTrDir)

  Function to retrieve the operational train directory.
- EXT_DECL TRDP_ERR_T tau_getStaticCstInfo (TRDP_APP_SESSION_T appHandle, TRDP_CONSIST↩ _INFO_T ∗pCstInfo, TRDP_UUID_T const cstUUID)

  Function to retrieve the operational train directory.
- EXT_DECL TRDP_ERR_T tau_getTTI (TRDP_APP_SESSION_T appHandle, TRDP_OP_TRAIN_DIR_S↩ TATE_T ∗pOpTrDirState, TRDP_OP_TRAIN_DIR_T ∗pOpTrDir, TRDP_TRAIN_DIR_T ∗pTrDir, TRDP_TR↩ AIN_NET_DIR_T ∗pTrNetDir)

  Function to retrieve the operational train directory.
- EXT_DECL TRDP_ERR_T tau_getTrnCstCnt (TRDP_APP_SESSION_T appHandle, UINT16 ∗pTrnCstCnt)

  Function to retrieve the total number of consists in the train.
- EXT_DECL TRDP_ERR_T tau_getTrnVehCnt (TRDP_APP_SESSION_T appHandle, UINT16 ∗pTrnVehCnt)

  Function to retrieve the total number of vehicles in the train.
- EXT_DECL TRDP_ERR_T tau_getCstVehCnt (TRDP_APP_SESSION_T appHandle, UINT16 ∗pCstVehCnt, const TRDP_LABEL_T pCstLabel)

  Function to retrieve the total number of vehicles in a consist.
- EXT_DECL TRDP_ERR_T tau_getCstFctCnt (TRDP_APP_SESSION_T appHandle, UINT16 ∗pCstFctCnt, const TRDP_LABEL_T pCstLabel)

  Function to retrieve the total number of functions in a consist.
- EXT_DECL TRDP_ERR_T tau_getCstFctInfo (TRDP_APP_SESSION_T appHandle, TRDP_FUNCTION_↩ INFO_T ∗pFctInfo, const TRDP_LABEL_T pCstLabel, UINT16 maxFctCnt)

  Function to retrieve the function information of the consist.
- EXT_DECL TRDP_ERR_T tau_getVehInfo (TRDP_APP_SESSION_T appHandle, TRDP_VEHICLE_INF↩ O_T ∗pVehInfo, const TRDP_LABEL_T pVehLabel, const TRDP_LABEL_T pCstLabel)

  Function to retrieve the vehicle information of a consist's vehicle.
- EXT_DECL TRDP_ERR_T tau_getCstInfo (TRDP_APP_SESSION_T appHandle, TRDP_CONSIST_INF↩ O_T ∗pCstInfo, const TRDP_LABEL_T pCstLabel)

  Function to retrieve the consist information of a train's consist.
- EXT_DECL TRDP_ERR_T tau_getVehOrient (TRDP_APP_SESSION_T appHandle, UINT8 ∗pVehOrient, UINT8 ∗pCstOrient, TRDP_LABEL_T pVehLabel, TRDP_LABEL_T pCstLabel)

  Function to retrieve the orientation of the given vehicle.

5.6.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- train topology information access

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2014. All rights reserved.

Id

```
BL 2016-02-18: Ticket #7: Add train topology information support
```

5.6.2 Function Documentation

5.6.2.1 tau_deInitTTI()

```
EXT_DECL void tau_deInitTTI (
            TRDP_APP_SESSION_T appHandle )
```

Function to terminate TTI access.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().

Return values

none	

5.6.2.2 tau_getCstFctCnt()

```
EXT_DECL TRDP_ERR_T tau_getCstFctCnt (
            TRDP_APP_SESSION_T appHandle,
            UINT16 * pCstFctCnt,
            const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the total number of functions in a consist.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pCstFctCnt*	Pointer to the number of functions to be returned
in	*pCstLabel*	Pointer to a consist label. NULL means own consist.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.3 tau_getCstFctInfo()

```
EXT_DECL TRDP_ERR_T tau_getCstFctInfo (
             TRDP_APP_SESSION_T appHandle,
             TRDP_FUNCTION_INFO_T * pFctInfo,
             const TRDP_LABEL_T pCstLabel,
             UINT16 maxFctCnt )
```

Function to retrieve the function information of the consist.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pFctInfo*	Pointer to function info list to be returned. Memory needs to be provided by application. Set NULL if not used.
in	*pCstLabel*	Pointer to a consist label. NULL means own consist.
in	*maxFctCnt*	Maximal number of functions to be returned in provided buffer.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.4 tau_getCstInfo()

```
EXT_DECL TRDP_ERR_T tau_getCstInfo (
             TRDP_APP_SESSION_T appHandle,
             TRDP_CONSIST_INFO_T * pCstInfo,
             const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the consist information of a train's consist.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pCstInfo*	Pointer to the consist info to be returned.
in	*pCstLabel*	Pointer to a consist label. NULL means own consist.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.5 tau_getCstVehCnt()

```
EXT_DECL TRDP_ERR_T tau_getCstVehCnt (
             TRDP_APP_SESSION_T appHandle,
             UINT16 * pCstVehCnt,
             const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the total number of vehicles in a consist.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pCstVehCnt*	Pointer to the number of vehicles to be returned
in	*pCstLabel*	Pointer to a consist label. NULL means own consist.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.6 tau_getOpTrDirectory()

```
EXT_DECL TRDP_ERR_T tau_getOpTrDirectory (
             TRDP_APP_SESSION_T appHandle,
             TRDP_OP_TRAIN_DIR_STATE_T * pOpTrDirState,
             TRDP_OP_TRAIN_DIR_T * pOpTrDir )
```

Function to retrieve the operational train directory state.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pOpTrDirState*	Pointer to an operational train directory state structure to be returned.
out	*pOpTrDir*	Pointer to an operational train directory structure to be returned.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.7 tau_getOpTrnDirectoryStatusInfo()

```
EXT_DECL TRDP_ERR_T tau_getOpTrnDirectoryStatusInfo (
             TRDP_APP_SESSION_T appHandle,
             TRDP_OP_TRAIN_DIR_STATUS_INFO_T * pOpTrnDirStatusInfo )
```

Function to retrieve the operational train directory state info.

Return a copy of the last received PD 100 telegram. Note: The values are in host endianess! When validating (SDTv2), network endianess must be ensured.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pOpTrnDirStatusInfo*	Pointer to an operational train directory state structure to be returned.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.8 tau_getStaticCstInfo()

```
EXT_DECL TRDP_ERR_T tau_getStaticCstInfo (
             TRDP_APP_SESSION_T appHandle,
             TRDP_CONSIST_INFO_T * pCstInfo,
             TRDP_UUID_T const cstUUID )
```

Function to retrieve the operational train directory.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pCstInfo*	Pointer to a consist info structure to be returned.
in	*cstUUID*	UUID of the consist the consist info is rquested for.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.9 tau_getTrDirectory()

```
EXT_DECL TRDP_ERR_T tau_getTrDirectory (
             TRDP_APP_SESSION_T appHandle,
             TRDP_TRAIN_DIR_T * pTrDir )
```

Function to retrieve the operational train directory.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pTrDir*	Pointer to a train directory structure to be returned.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.10 tau_getTrnCstCnt()

```
EXT_DECL TRDP_ERR_T tau_getTrnCstCnt (
             TRDP_APP_SESSION_T appHandle,
             UINT16 * pTrnCstCnt )
```

Function to retrieve the total number of consists in the train.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pTrnCstCnt*	Pointer to the number of consists to be returned

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.11 tau_getTrnVehCnt()

```
EXT_DECL TRDP_ERR_T tau_getTrnVehCnt (
             TRDP_APP_SESSION_T appHandle,
             UINT16 * pTrnVehCnt )
```

Function to retrieve the total number of vehicles in the train.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pTrnVehCnt*	Pointer to the number of vehicles to be returned

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.12 tau_getTTI()

```
EXT_DECL TRDP_ERR_T tau_getTTI (
             TRDP_APP_SESSION_T appHandle,
             TRDP_OP_TRAIN_DIR_STATE_T * pOpTrDirState,
             TRDP_OP_TRAIN_DIR_T * pOpTrDir,
             TRDP_TRAIN_DIR_T * pTrDir,
             TRDP_TRAIN_NET_DIR_T * pTrNetDir )
```

Function to retrieve the operational train directory.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pOpTrDirState*	Pointer to an operational train directory state structure to be returned.
out	*pOpTrDir*	Pointer to an operational train directory structure to be returned.
out	*pTrDir*	Pointer to a train directory structure to be returned.
out	*pTrNetDir*	Pointer to a train network directory structure to be returned.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.13 tau_getVehInfo()

```
EXT_DECL TRDP_ERR_T tau_getVehInfo (
             TRDP_APP_SESSION_T appHandle,
             TRDP_VEHICLE_INFO_T * pVehInfo,
             const TRDP_LABEL_T pVehLabel,
             const TRDP_LABEL_T pCstLabel )
```

Function to retrieve the vehicle information of a consist's vehicle.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pVehInfo*	Pointer to the vehicle info to be returned.
in	*pVehLabel*	Pointer to a vehicle label. NULL means own vehicle if cstLabel refers to own consist.
in	*pCstLabel*	Pointer to a consist label. NULL means own consist.

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.14 tau_getVehOrient()

```
EXT_DECL TRDP_ERR_T tau_getVehOrient (
             TRDP_APP_SESSION_T appHandle,
             UINT8 * pVehOrient,
             UINT8 * pCstOrient,
             TRDP_LABEL_T pVehLabel,
             TRDP_LABEL_T pCstLabel )
```

Function to retrieve the orientation of the given vehicle.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
out	*pVehOrient*	Pointer to the vehicle orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
out	*pCstOrient*	Pointer to the consist orientation to be returned '00'B = not known (corrected vehicle) '01'B = same as operational train direction '10'B = inverse to operational train direction
in	*pVehLabel*	vehLabel = NULL means own vehicle if cstLabel == NULL
in	*pCstLabel*	cstLabel = NULL means own consist

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	Parameter error

5.6.2.15 tau_initTTIaccess()

```
EXT_DECL TRDP_ERR_T tau_initTTIaccess (
             TRDP_APP_SESSION_T appHandle,
             VOS_SEMA_T userAction,
             TRDP_IP_ADDR_T ecspIpAddr,
             CHAR8 * hostsFileName )
```

Function to init TTI access.

Parameters

in	*appHandle*	Handle returned by tlc_openSession().
in	*userAction*	Semaphore to fire if inauguration took place.
in	*ecspIpAddr*	ECSP IP address.
in	*hostsFileName*	Optional host file name as ECSP replacement.

Return values

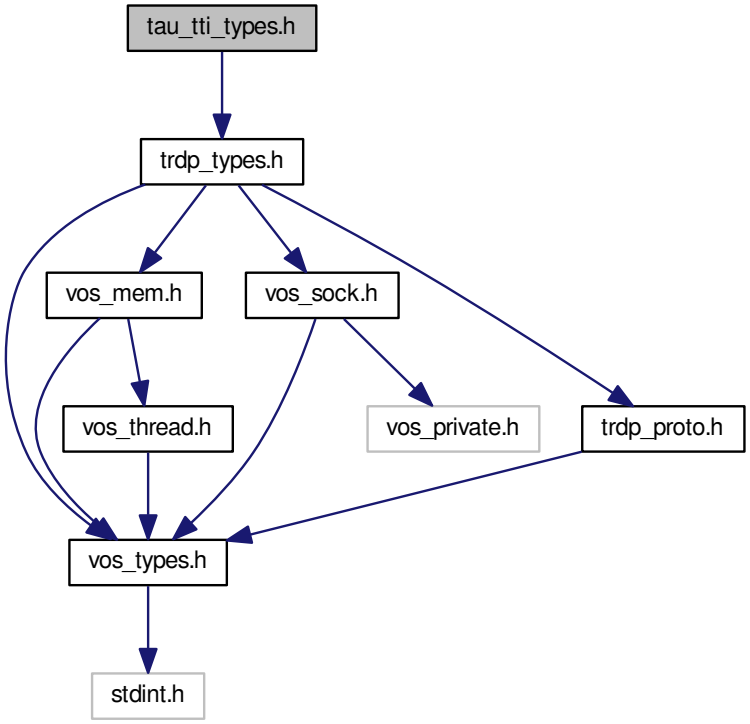
TRDP_NO_ERR	no error
TRDP_INIT_ERR	initialisation error

5.7 tau_tti_types.h File Reference

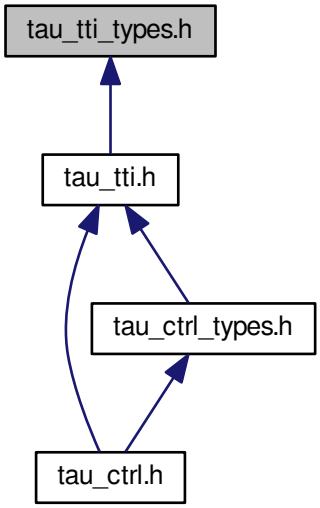
TRDP utility interface definitions.

```
#include "trdp_types.h"
```

Include dependency graph for tau_tti_types.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct GNU_PACKED
  Types for ETB control.
- struct TRDP_ETB_INFO_T
  Types for train configuration information.
- struct TRDP_CLTR_CST_INFO_T
  Closed train consists information.
- struct TRDP_PROP_T
  Application defined properties.
- struct TRDP_FUNCTION_INFO_T
  function/device information structure
- struct TRDP_VEHICLE_INFO_T
  vehicle information structure
- struct TRDP_CONSIST_INFO_T
  consist information structure
- struct GNU_PACKED
  Types for ETB control.
- struct GNU_PACKED
  Types for ETB control.
- struct GNU_PACKED
  Types for ETB control.
- struct GNU_PACKED
  Types for ETB control.
- struct GNU_PACKED
  Types for ETB control.
- struct GNU_PACKED

Types for ETB control.

- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.

Macros

- #define TRDP_MAX_CST_CNT 63

  max number of consists per train
- #define TRDP_MAX_VEH_CNT 63

  max number of vehicles per train

5.7.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- train topology information access type definitions acc. to IEC61375-2-3

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

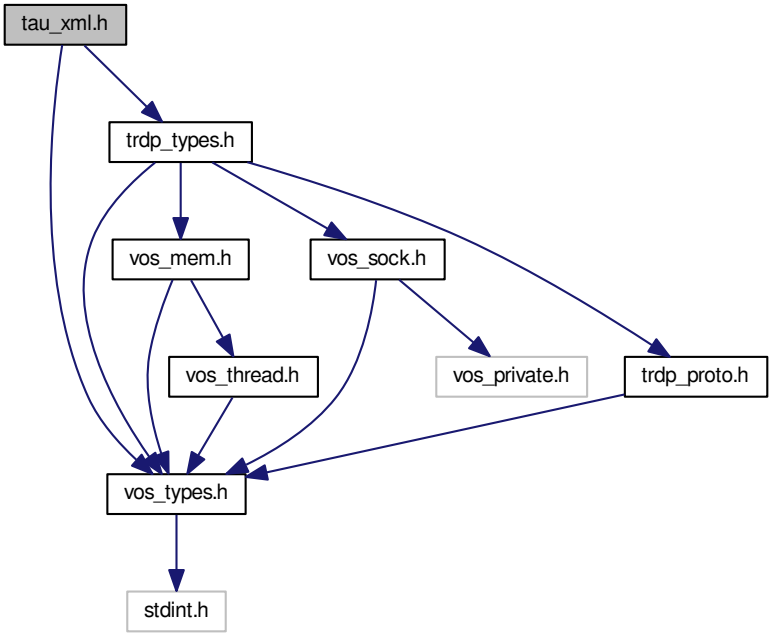
Id

5.8 tau_xml.h File Reference

TRDP utility interface definitions.

```
#include "vos_types.h"
#include "trdp_types.h"
```

Include dependency graph for tau_xml.h:



Data Structures

- struct TRDP_SDT_PAR_T

  Types to read out the XML configuration.
- struct TRDP_DBG_CONFIG_T

  Control for debug output device/file on application level.
- struct TRDP_XML_DOC_HANDLE_T

  Parsed XML document handle.

Enumerations

- enum TRDP_EXCHG_OPTION_T {
  TRDP_EXCHG_UNSET = 0,
  TRDP_EXCHG_SOURCE = 1,
  TRDP_EXCHG_SINK = 2,
  TRDP_EXCHG_SOURCESINK = 3 }

  Type attribute for telegrams.

- enum TRDP_DBG_OPTION_T {
  TRDP_DBG_DEFAULT = 0,
  TRDP_DBG_OFF = 0x01,
  TRDP_DBG_ERR = 0x02,
  TRDP_DBG_WARN = 0x04,
  TRDP_DBG_INFO = 0x08,
  TRDP_DBG_DBG = 0x10,
  TRDP_DBG_TIME = 0x20,
  TRDP_DBG_LOC = 0x40,
  TRDP_DBG_CAT = 0x80 }

  Control for debug output format on application level.

Functions

- EXT_DECL TRDP_ERR_T tau_prepareXmlDoc (const CHAR8 *pFileName, TRDP_XML_DOC_HANDLE_T *pDocHnd)

  Load XML file into DOM tree, prepare XPath context.
- EXT_DECL void tau_freeXmlDoc (TRDP_XML_DOC_HANDLE_T *pDocHnd)

  Free all the memory allocated by tau_prepareXmlDoc.
- EXT_DECL TRDP_ERR_T tau_readXmlDeviceConfig (const TRDP_XML_DOC_HANDLE_T *pDocHnd, TRDP_MEM_CONFIG_T *pMemConfig, TRDP_DBG_CONFIG_T *pDbgConfig, UINT32 *pNumComPar, TRDP_COM_PAR_T **ppComPar, UINT32 *pNumIfConfig, TRDP_IF_CONFIG_T **ppIfConfig)

  Function to read the TRDP device configuration parameters out of the XML configuration file.
- EXT_DECL TRDP_ERR_T tau_readXmlInterfaceConfig (const TRDP_XML_DOC_HANDLE_T *pDocHnd, const CHAR8 *pIfName, TRDP_PROCESS_CONFIG_T *pProcessConfig, TRDP_PD_CONFIG_T *pPdConfig, TRDP_MD_CONFIG_T *pMdConfig, UINT32 *pNumExchgPar, TRDP_EXCHG_PAR_T **ppExchgPar)

  Read the interface relevant telegram parameters (except data set configuration) out of the configuration file .
- EXT_DECL TRDP_ERR_T tau_readXmlDatasetConfig (const TRDP_XML_DOC_HANDLE_T *pDocHnd, UINT32 *pNumComId, TRDP_COMID_DSID_MAP_T **ppComIdDsIdMap, UINT32 *pNumDataset, papTRDP_DATASET_T papDataset)

  Function to read the DataSet configuration out of the XML configuration file.
- EXT_DECL void tau_freeXmlDatasetConfig (UINT32 numComId, TRDP_COMID_DSID_MAP_T *pComIdDsIdMap, UINT32 numDataset, TRDP_DATASET_T **pNumDataset)

  Function to free the memory for the DataSet configuration.
- EXT_DECL void tau_freeTelegrams (UINT32 numExchgPar, TRDP_EXCHG_PAR_T *pExchgPar)

  Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

5.8.1 Detailed Description

TRDP utility interface definitions.

This module provides the interface to the following utilities

- read xml configuration interpreter

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss (initial version)

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

Id

```
BL 2016-02-11: Ticket #102: Custom XML parser, libxml2 not needed anymore
```

5.8.2 Enumeration Type Documentation

5.8.2.1 TRDP_DBG_OPTION_T

```
enum TRDP_DBG_OPTION_T
```

Control for debug output format on application level.

Enumerator

TRDP_DBG_DEFAULT	Printout default.
TRDP_DBG_OFF	Printout off.
TRDP_DBG_ERR	Printout error.
TRDP_DBG_WARN	Printout warning and error.
TRDP_DBG_INFO	Printout info, warning and error.
TRDP_DBG_DBG	Printout debug, info, warning and error.
TRDP_DBG_TIME	Printout timestamp.
TRDP_DBG_LOC	Printout file name and line.
TRDP_DBG_CAT	Printout category (DBG, INFO, WARN, ERR)

5.8.2.2 TRDP_EXCHG_OPTION_T

```
enum TRDP_EXCHG_OPTION_T
```

Type attribute for telegrams.

Enumerator

TRDP_EXCHG_UNSET	default, direction is not defined
TRDP_EXCHG_SOURCE	telegram shall be published
TRDP_EXCHG_SINK	telegram shall be subscribed
TRDP_EXCHG_SOURCESINK	telegram shall be published and subscribed

5.8.3 Function Documentation

5.8.3.1 tau_freeTelegrams()

```
EXT_DECL void tau_freeTelegrams (
            UINT32 numExchgPar,
            TRDP_EXCHG_PAR_T * pExchgPar )
```

Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

Parameters

in	*numExchgPar*	Number of telegram configurations in the array
in	*pExchgPar*	Pointer to array of telegram configurations

5.8.3.2 tau_freeXmlDatasetConfig()

```
EXT_DECL void tau_freeXmlDatasetConfig (
            UINT32 numComId,
            TRDP_COMID_DSID_MAP_T * pComIdDsIdMap,
            UINT32 numDataset,
            TRDP_DATASET_T ** pNumDataset )
```

Function to free the memory for the DataSet configuration.

Free the memory for the DataSet configuration which was allocated when parsing the XML configuration file.

Parameters

in	*numComId*	The number of entries in the ComId DatasetId mapping list
in	*pComIdDsIdMap*	Pointer to an array of structures of type TRDP_COMID_DSID_MAP_T
in	*numDataset*	The number of datasets found in the configuration
in	*pNumDataset*	Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values

none	

5.8.3.3 tau_freeXmlDoc()

```
EXT_DECL void tau_freeXmlDoc (
            TRDP_XML_DOC_HANDLE_T * pDocHnd )
```

Free all the memory allocated by tau_prepareXmlDoc.

Parameters

in	*pDocHnd*	Handle of the parsed XML file

5.8.3.4 tau_prepareXmlDoc()

```
EXT_DECL TRDP_ERR_T tau_prepareXmlDoc (
            const CHAR8 * pFileName,
            TRDP_XML_DOC_HANDLE_T * pDocHnd )
```

Load XML file into DOM tree, prepare XPath context.

Parameters

in	*pFileName*	Path and filename of the xml configuration file
out	*pDocHnd*	Handle of the parsed XML file

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	File does not exist

5.8.3.5 tau_readXmlDatasetConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlDatasetConfig (
            const TRDP_XML_DOC_HANDLE_T * pDocHnd,
            UINT32 * pNumComId,
            TRDP_COMID_DSID_MAP_T ** ppComIdDsIdMap,
            UINT32 * pNumDataset,
            papTRDP_DATASET_T papDataset )
```

Function to read the DataSet configuration out of the XML configuration file.

Parameters

in	*pDocHnd*	Handle of the XML document prepared by tau_prepareXmlDoc
out	*pNumComId*	Pointer to the number of entries in the ComId DatasetId mapping list
out	*ppComIdDsIdMap*	Pointer to an array of a structures of type TRDP_COMID_DSID_MAP_T
out	*pNumDataset*	Pointer to the number of datasets found in the configuration
out	*papDataset*	Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_PARAM_ERR	File not existing

5.8.3.6 tau_readXmlDeviceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlDeviceConfig (
            const TRDP_XML_DOC_HANDLE_T * pDocHnd,
```

```
                TRDP_MEM_CONFIG_T * pMemConfig,
                TRDP_DBG_CONFIG_T * pDbgConfig,
                UINT32 * pNumComPar,
                TRDP_COM_PAR_T ** ppComPar,
                UINT32 * pNumIfConfig,
                TRDP_IF_CONFIG_T ** ppIfConfig )
```

Function to read the TRDP device configuration parameters out of the XML configuration file.

Parameters

in	*pDocHnd*	Handle of the XML document prepared by tau_prepareXmlDoc
out	*pMemConfig*	Memory configuration
out	*pDbgConfig*	Debug printout configuration for application use
out	*pNumComPar*	Number of configured com parameters
out	*ppComPar*	Pointer to array of com parameters
out	*pNumIfConfig*	Number of configured interfaces
out	*ppIfConfig*	Pointer to an array of interface parameter sets

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_PARAM_ERR	File not existing

5.8.3.7 tau_readXmlInterfaceConfig()

```
EXT_DECL TRDP_ERR_T tau_readXmlInterfaceConfig (
             const TRDP_XML_DOC_HANDLE_T * pDocHnd,
             const CHAR8 * pIfName,
             TRDP_PROCESS_CONFIG_T * pProcessConfig,
             TRDP_PD_CONFIG_T * pPdConfig,
             TRDP_MD_CONFIG_T * pMdConfig,
             UINT32 * pNumExchgPar,
             TRDP_EXCHG_PAR_T ** ppExchgPar )
```

Read the interface relevant telegram parameters (except data set configuration) out of the configuration file .

Parameters

in	*pDocHnd*	Handle of the XML document prepared by tau_prepareXmlDoc
in	*pIfName*	Interface name
out	*pProcessConfig*	TRDP process (session) configuration for the interface
out	*pPdConfig*	PD default configuration for the interface
out	*pMdConfig*	MD default configuration for the interface
out	*pNumExchgPar*	Number of configured telegrams
out	*ppExchgPar*	Pointer to array of telegram configurations

Return values

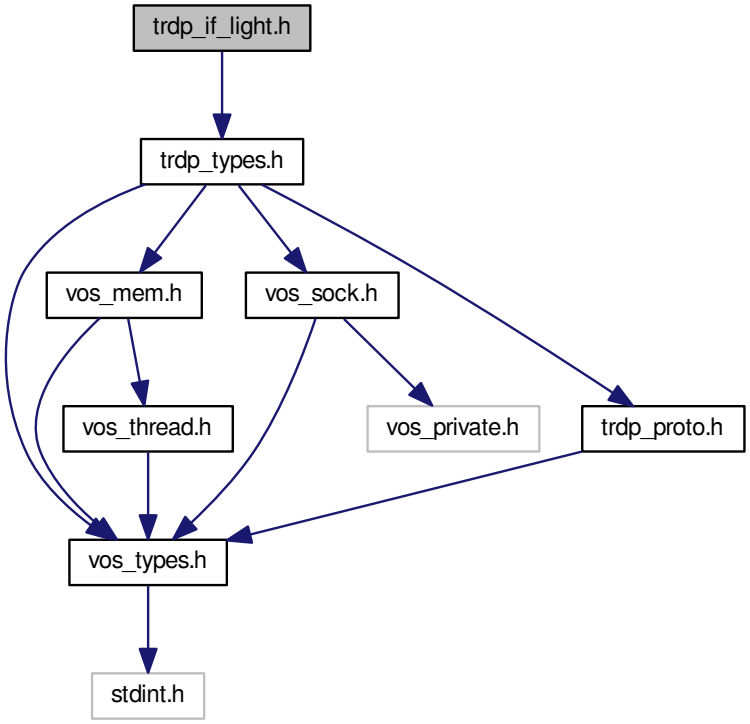
TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_PARAM_ERR	File not existing

5.9 trdp_if_light.h File Reference

TRDP Light interface functions (API)

```
#include "trdp_types.h"
```

Include dependency graph for trdp_if_light.h:



Functions

- EXT_DECL TRDP_ERR_T tlc_init (const TRDP_PRINT_DBG_T pPrintDebugString, void *pRefCon, const TRDP_MEM_CONFIG_T *pMemConfig)

  Support for message data can only be excluded during compile time!
- EXT_DECL TRDP_ERR_T tlc_openSession (TRDP_APP_SESSION_T *pAppHandle, TRDP_IP_ADDR_T ownIpAddr, TRDP_IP_ADDR_T leaderIpAddr, const TRDP_MARSHALL_CONFIG_T *pMarshall, const TRDP_PD_CONFIG_T *pPdDefault, const TRDP_MD_CONFIG_T *pMdDefault, const TRDP_PROCESS_CONFIG_T *pProcessConfig)

Open a session with the TRDP stack.

- EXT_DECL TRDP_ERR_T tlc_reinitSession (TRDP_APP_SESSION_T appHandle)

  Re-Initialize.

- EXT_DECL TRDP_ERR_T tlc_configSession (TRDP_APP_SESSION_T appHandle, const TRDP_MARSHALL_CONFIG_T *pMarshall, const TRDP_PD_CONFIG_T *pPdDefault, const TRDP_MD_CONFIG_T *pMdDefault, const TRDP_PROCESS_CONFIG_T *pProcessConfig)

  (Re-)configure a session.

- EXT_DECL TRDP_ERR_T tlc_closeSession (TRDP_APP_SESSION_T appHandle)

  Close a session.

- EXT_DECL TRDP_ERR_T tlc_terminate (void)

  Un-Initialize.

- EXT_DECL TRDP_ERR_T tlc_setETBTopoCount (TRDP_APP_SESSION_T appHandle, UINT32 etbTopoCnt)

  Set new topocount for trainwide communication.

- EXT_DECL TRDP_ERR_T tlc_setOpTrainTopoCount (TRDP_APP_SESSION_T appHandle, UINT32 opTrnTopoCnt)

  Set new operational train topocount for direction/orientation sensitive communication.

- EXT_DECL TRDP_ERR_T tlc_freeBuf (TRDP_APP_SESSION_T appHandle, char *pBuf)

  Frees the buffer reserved by the TRDP layer.

- EXT_DECL TRDP_ERR_T tlc_getInterval (TRDP_APP_SESSION_T appHandle, TRDP_TIME_T *pInterval, TRDP_FDS_T *pFileDesc, INT32 *pNoDesc)

  Get the lowest time interval for PDs.

- EXT_DECL TRDP_ERR_T tlc_process (TRDP_APP_SESSION_T appHandle, TRDP_FDS_T *pRfds, INT32 *pCount)

  Work loop of the TRDP handler.

- EXT_DECL TRDP_IP_ADDR_T tlc_getOwnIpAddress (TRDP_APP_SESSION_T appHandle)

  Get the interface address.

- EXT_DECL TRDP_ERR_T tlp_publish (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T *pPubHandle, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, UINT32 interval, UINT32 redId, TRDP_FLAGS_T pktFlags, const TRDP_SEND_PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize)

  Prepare for sending PD messages.

- EXT_DECL TRDP_ERR_T tlp_republish (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T pubHandle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr)

  Prepare for sending PD messages.

- EXT_DECL TRDP_ERR_T tlp_unpublish (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T pubHandle)

  Stop sending PD messages.

- EXT_DECL TRDP_ERR_T tlp_put (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T pubHandle, const UINT8 *pData, UINT32 dataSize)

  Update the process data to send.

- EXT_DECL TRDP_ERR_T tlp_setRedundant (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 leader)

  Do not send redundant PD's when we are follower.

- EXT_DECL TRDP_ERR_T tlp_getRedundant (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 *pLeader)

  Get status of redundant ComIds.

- EXT_DECL TRDP_ERR_T tlp_request (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T subHandle, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, UINT32 redId, TRDP_FLAGS_T pktFlags, const TRDP_SEND_PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize, UINT32 replyComId, TRDP_IP_ADDR_T replyIpAddr)

  Initiate sending PD messages (PULL).

- EXT_DECL TRDP_ERR_T tlp_subscribe (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T *pSub↩ Handle, const void *pUserRef, TRDP_PD_CALLBACK_T pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_↩ T pktFlags, UINT32 timeout, TRDP_TO_BEHAVIOR_T toBehavior)

  Prepare for receiving PD messages.
- EXT_DECL TRDP_ERR_T tlp_resubscribe (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T sub↩ Handle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_↩ T destIpAddr)

  Reprepare for receiving PD messages.
- EXT_DECL TRDP_ERR_T tlp_unsubscribe (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T sub↩ Handle)

  Stop receiving PD messages.
- EXT_DECL TRDP_ERR_T tlp_get (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T subHandle, TR↩ DP_PD_INFO_T *pPdInfo, UINT8 *pData, UINT32 *pDataSize)

  Get the last valid PD message.
- EXT_DECL TRDP_ERR_T tlm_notify (TRDP_APP_SESSION_T appHandle, const void *pUserRef, TRD↩ P_MD_CALLBACK_T pfCbFunction, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_↩ IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_T pktFlags, const TRDP_SEND_↩ PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T sourceURI, const TRDP_URI_USER_T destURI)

  Initiate sending MD notification message.
- EXT_DECL TRDP_ERR_T tlm_request (TRDP_APP_SESSION_T appHandle, const void *pUserRef, T↩ RDP_MD_CALLBACK_T pfCbFunction, TRDP_UUID_T *pSessionId, UINT32 comId, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T srcIpAddr, TRDP_IP_ADDR_T destIpAddr, TRDP_FLAGS_↩ T pktFlags, UINT32 numReplies, UINT32 replyTimeout, UINT32 maxNumRetries, const TRDP_SEND_↩ PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize, const TRDP_URI_USER_T sourceURI, const TRDP_URI_USER_T destURI)

  Initiate sending MD request message.
- EXT_DECL TRDP_ERR_T tlm_confirm (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *p↩ SessionId, UINT16 userStatus, const TRDP_SEND_PARAM_T *pSendParam)

  Initiate sending MD confirm message.
- EXT_DECL TRDP_ERR_T tlm_abortSession (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId)

  Cancel an open session.
- EXT_DECL TRDP_ERR_T tlm_addListener (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T *pListen↩ Handle, const void *pUserRef, TRDP_MD_CALLBACK_T pfCbFunction, UINT32 comId, UINT32 etbTopo↩ Cnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T mcDestIpAddr, TRDP_FLAGS_T pktFlags, const TRDP_↩ URI_USER_T destURI)

  Subscribe to MD messages.
- EXT_DECL TRDP_ERR_T tlm_readdListener (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T listen↩ Handle, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, TRDP_IP_ADDR_T mcDestIpAddr)

  Resubscribe to MD messages.
- EXT_DECL TRDP_ERR_T tlm_delListener (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T listen↩ Handle)

  Remove Listener.
- TRDP_ERR_T tlm_reply (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId, UINT32 comId, UINT16 userStatus, const TRDP_SEND_PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize)

  Send a MD reply message.
- TRDP_ERR_T tlm_replyQuery (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId, UINT32 comId, UINT16 userStatus, UINT32 confirmTimeout, const TRDP_SEND_PARAM_T *pSendParam, const UINT8 *pData, UINT32 dataSize)

  Send a MD reply query message.
- TRDP_ERR_T tlm_replyErr (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T *pSessionId, UI↩ NT32 comId, TRDP_REPLY_STATUS_T replyStatus, const TRDP_SEND_PARAM_T *pSendParam)

Send a MD reply message.

- EXT_DECL const CHAR8 * tlc_getVersionString (void)

  Return a human readable version representation.
- EXT_DECL const TRDP_VERSION_T * tlc_getVersion (void)

  Return version.
- EXT_DECL TRDP_ERR_T tlc_getStatistics (TRDP_APP_SESSION_T appHandle, TRDP_STATISTICS_T *pStatistics)

  Return statistics.
- EXT_DECL TRDP_ERR_T tlc_getSubsStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *pNum↩ Subs, TRDP_SUBS_STATISTICS_T *pStatistics)

  Return PD subscription statistics.
- EXT_DECL TRDP_ERR_T tlc_getPubStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *pNumPub, TRDP_PUB_STATISTICS_T *pStatistics)

  Return PD publish statistics.
- EXT_DECL TRDP_ERR_T tlc_getUdpListStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *p↩ NumList, TRDP_LIST_STATISTICS_T *pStatistics)

  Return UDP MD listener statistics.
- EXT_DECL TRDP_ERR_T tlc_getTcpListStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *pNum↩ List, TRDP_LIST_STATISTICS_T *pStatistics)

  Return TCP MD listener statistics.
- EXT_DECL TRDP_ERR_T tlc_getRedStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *pNumRed, TRDP_RED_STATISTICS_T *pStatistics)

  Return redundancy group statistics.
- EXT_DECL TRDP_ERR_T tlc_getJoinStatistics (TRDP_APP_SESSION_T appHandle, UINT16 *pNumJoin, UINT32 *pIpAddr)

  Return join statistics.
- EXT_DECL TRDP_ERR_T tlc_resetStatistics (TRDP_APP_SESSION_T appHandle)

  Reset statistics.

5.9.1 Detailed Description

TRDP Light interface functions (API)

Low level functions for communicating using the TRDP protocol

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

Id

```
BL 2015-11-24: Accessor for IP address of session
BL 2015-09-04: Ticket #99: refCon for tlc_init()

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
```

5.9.2 Function Documentation

5.9.2.1 tlc_closeSession()

```
EXT_DECL TRDP_ERR_T tlc_closeSession (
             TRDP_APP_SESSION_T appHandle )
```

Close a session.

Clean up and release all resources of that session

Parameters

in	*appHandle*	The handle returned by tlc_openSession

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	handle NULL

5.9.2.2 tlc_configSession()

```
EXT_DECL TRDP_ERR_T tlc_configSession (
             TRDP_APP_SESSION_T appHandle,
             const TRDP_MARSHALL_CONFIG_T * pMarshall,
             const TRDP_PD_CONFIG_T * pPdDefault,
             const TRDP_MD_CONFIG_T * pMdDefault,
             const TRDP_PROCESS_CONFIG_T * pProcessConfig )
```

(Re-)configure a session.

tlc_configSession is called by openSession, but may also be called later on to change the defaults.

Parameters

in	*appHandle*	A handle for further calls to the trdp stack
in	*pMarshall*	Pointer to marshalling configuration
in	*pPdDefault*	Pointer to default PD configuration
in	*pMdDefault*	Pointer to default MD configuration
in	*pProcessConfig*	Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values

TRDP_NO_ERR	no error
TRDP_INIT_ERR	not yet inited
TRDP_PARAM_ERR	parameter error

5.9.2.3 tlc_freeBuf()

```
EXT_DECL TRDP_ERR_T tlc_freeBuf (
             TRDP_APP_SESSION_T appHandle,
             char * pBuf )
```

Frees the buffer reserved by the TRDP layer.

Parameters

in	*appHandle*	The handle returned by tlc_openSession
in	*pBuf*	pointer to the buffer to be freed

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	buffer pointer invalid

5.9.2.4 tlc_getInterval()

```
EXT_DECL TRDP_ERR_T tlc_getInterval (
             TRDP_APP_SESSION_T appHandle,
             TRDP_TIME_T * pInterval,
             TRDP_FDS_T * pFileDesc,
             INT32 * pNoDesc )
```

Get the lowest time interval for PDs.

Return the maximum time interval suitable for 'select()' so that we can send due PD packets in time. If the PD send queue is empty, return zero time

Parameters

in	*appHandle*	The handle returned by tlc_openSession
out	*pInterval*	pointer to needed interval
in,out	*pFileDesc*	pointer to file descriptor set
out	*pNoDesc*	pointer to put no of used descriptors (for select())

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid

5.9.2.5 tlc_getJoinStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getJoinStatistics (
             TRDP_APP_SESSION_T appHandle,
```

```
            UINT16 * pNumJoin,
            UINT32 * pIpAddr )
```

Return join statistics.

Memory for statistics information must be provided by the user. The reserved length is given via pNumJoin implicitely.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in,out	*pNumJoin*	Pointer to the number of joined IP Adresses
out	*pIpAddr*	Pointer to a list with the joined IP adresses

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	there are more items than requested

5.9.2.6 tlc_getOwnIpAddress()

```
EXT_DECL TRDP_IP_ADDR_T tlc_getOwnIpAddress (
            TRDP_APP_SESSION_T appHandle )
```

Get the interface address.

Parameters

out	*appHandle*	A handle for further calls to the trdp stack

Return values

real↩ IP	

5.9.2.7 tlc_getPubStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getPubStatistics (
            TRDP_APP_SESSION_T appHandle,
            UINT16 * pNumPub,
            TRDP_PUB_STATISTICS_T * pStatistics )
```

Return PD publish statistics.

Memory for statistics information must be provided by the user. The reserved length is given via pNumPub implicitely.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in,out	*pNumPub*	Pointer to the number of publishers
out	*pStatistics*	pointer to a list with the publish statistics information

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	there are more subscriptions than requested

5.9.2.8 tlc_getRedStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getRedStatistics (
             TRDP_APP_SESSION_T appHandle,
             UINT16 * pNumRed,
             TRDP_RED_STATISTICS_T * pStatistics )
```

Return redundancy group statistics.

Memory for statistics information must be provided by the user. The reserved length is given via pNumRed implicitely.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in,out	*pNumRed*	Pointer to the number of redundancy groups
out	*pStatistics*	Pointer to a list with the redundancy group information

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	there are more subscriptions than requested

5.9.2.9 tlc_getStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getStatistics (
             TRDP_APP_SESSION_T appHandle,
             TRDP_STATISTICS_T * pStatistics )
```

Return statistics.

Memory for statistics information must be preserved by the user.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
out	*pStatistics*	Pointer to statistics for this application session

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error

5.9.2.10 tlc_getSubsStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getSubsStatistics (
             TRDP_APP_SESSION_T appHandle,
             UINT16 * pNumSubs,
             TRDP_SUBS_STATISTICS_T * pStatistics )
```

Return PD subscription statistics.

Memory for statistics information must be provided by the user. The reserved length is given via pNumSub implicitely.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in,out	*pNumSubs*	In: The number of subscriptions requested Out: Number of subscriptions returned
in,out	*pStatistics*	Pointer to an array with the subscription statistics information

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	there are more subscriptions than requested

5.9.2.11 tlc_getTcpListStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getTcpListStatistics (
             TRDP_APP_SESSION_T appHandle,
             UINT16 * pNumList,
             TRDP_LIST_STATISTICS_T * pStatistics )
```

Return TCP MD listener statistics.

Memory for statistics information must be provided by the user. The reserved length is given via pNumLis implicitely.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in,out	*pNumList*	Pointer to the number of listeners
out	*pStatistics*	Pointer to a list with the listener statistics information

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	there are more subscriptions than requested

5.9.2.12 tlc_getUdpListStatistics()

```
EXT_DECL TRDP_ERR_T tlc_getUdpListStatistics (
             TRDP_APP_SESSION_T appHandle,
             UINT16 * pNumList,
             TRDP_LIST_STATISTICS_T * pStatistics )
```

Return UDP MD listener statistics.

Memory for statistics information must be provided by the user. The reserved length is given via pNumLis implicitely.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in,out	*pNumList*	Pointer to the number of listeners
out	*pStatistics*	Pointer to a list with the listener statistics information

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	there are more subscriptions than requested

5.9.2.13 tlc_getVersion()

```
EXT_DECL const TRDP_VERSION_T* tlc_getVersion (
             void  )
```

Return version.

Return pointer to version structure

Return values

const	TRDP_VERSION↩ _T

5.9.2.14 tlc_getVersionString()

```
EXT_DECL const CHAR8* tlc_getVersionString (
            void  )
```

Return a human readable version representation.

Return string in the form 'v.r.u.b'

Return values

const	string

5.9.2.15 tlc_init()

```
EXT_DECL TRDP_ERR_T tlc_init (
            const TRDP_PRINT_DBG_T pPrintDebugString,
            void * pRefCon,
            const TRDP_MEM_CONFIG_T * pMemConfig )
```

Support for message data can only be excluded during compile time!

Initialize the TRDP stack.

tlc_init initializes the memory subsystem and takes a function pointer to an output function for logging.

Parameters

in	*pPrintDebugString*	Pointer to debug print function
in	*pRefCon*	user context
in	*pMemConfig*	Pointer to memory configuration

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	memory allocation failed
TRDP_PARAM_ERR	initialization error

5.9.2.16 tlc_openSession()

```
EXT_DECL TRDP_ERR_T tlc_openSession (
            TRDP_APP_SESSION_T * pAppHandle,
```

```
            TRDP_IP_ADDR_T ownIpAddr,
            TRDP_IP_ADDR_T leaderIpAddr,
            const TRDP_MARSHALL_CONFIG_T * pMarshall,
            const TRDP_PD_CONFIG_T * pPdDefault,
            const TRDP_MD_CONFIG_T * pMdDefault,
            const TRDP_PROCESS_CONFIG_T * pProcessConfig )
```

Open a session with the TRDP stack.

tlc_openSession returns in pAppHandle a unique handle to be used in further calls to the stack.

Parameters

out	*pAppHandle*	A handle for further calls to the trdp stack
in	*ownIpAddr*	Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
in	*leaderIpAddr*	IP address of redundancy leader
in	*pMarshall*	Pointer to marshalling configuration
in	*pPdDefault*	Pointer to default PD configuration
in	*pMdDefault*	Pointer to default MD configuration
in	*pProcessConfig*	Pointer to process configuration only option parameter is used here to define session behavior all other parameters are only used to feed statistics

Return values

TRDP_NO_ERR	no error
TRDP_INIT_ERR	not yet inited
TRDP_PARAM_ERR	parameter error
TRDP_SOCK_ERR	socket error

5.9.2.17 tlc_process()

```
EXT_DECL TRDP_ERR_T tlc_process (
            TRDP_APP_SESSION_T appHandle,
            TRDP_FDS_T * pRfds,
            INT32 * pCount )
```

Work loop of the TRDP handler.

Search the queue for pending PDs to be sent Search the receive queue for pending PDs (time out)

Parameters

in	*appHandle*	The handle returned by tlc_openSession
in	*pRfds*	pointer to set of ready descriptors
in,out	*pCount*	pointer to number of ready descriptors

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid

5.9.2.18 tlc_reinitSession()

```
EXT_DECL TRDP_ERR_T tlc_reinitSession (
             TRDP_APP_SESSION_T appHandle )
```

Re-Initialize.

Should be called by the application when a link-down/link-up event has occured during normal operation. We need to re-join the multicast groups...

Parameters

in	*appHandle*	The handle returned by tlc_openSession

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	handle NULL

5.9.2.19 tlc_resetStatistics()

```
EXT_DECL TRDP_ERR_T tlc_resetStatistics (
             TRDP_APP_SESSION_T appHandle )
```

Reset statistics.

Parameters

in	*appHandle*	the handle returned by tlc_openSession

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error

5.9.2.20 tlc_setETBTopoCount()

```
EXT_DECL TRDP_ERR_T tlc_setETBTopoCount (
             TRDP_APP_SESSION_T appHandle,
             UINT32 etbTopoCnt )
```

Set new topocount for trainwide communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	*appHandle*	The handle returned by tlc_openSession
in	*etbTopoCnt*	New topocount value

5.9.2.21 tlc_setOpTrainTopoCount()

```
EXT_DECL TRDP_ERR_T tlc_setOpTrainTopoCount (
             TRDP_APP_SESSION_T appHandle,
             UINT32 opTrnTopoCnt )
```

Set new operational train topocount for direction/orientation sensitive communication.

This value is used for validating outgoing and incoming packets only!

Parameters

in	*appHandle*	The handle returned by tlc_openSession
in	*opTrnTopoCnt*	New operational topocount value

5.9.2.22 tlc_terminate()

```
EXT_DECL TRDP_ERR_T tlc_terminate (
             void  )
```

Un-Initialize.

Clean up and close all sessions. Mainly used for debugging/test runs. No further calls to library allowed

Return values

TRDP_NO_ERR	no error

5.9.2.23 tlm_abortSession()

```
EXT_DECL TRDP_ERR_T tlm_abortSession (
             TRDP_APP_SESSION_T appHandle,
             const TRDP_UUID_T * pSessionId )
```

Cancel an open session.

Abort an open session; any pending messages will be dropped

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*p↩ SessionId*	Session ID returned by request

Return values

TRDP_NO_ERR	no error
TRDP_NO_SESSION_ERR	no such session
TRDP_NOINIT_ERR	handle invalid

5.9.2.24 tlm_addListener()

```
EXT_DECL TRDP_ERR_T tlm_addListener (
             TRDP_APP_SESSION_T appHandle,
             TRDP_LIS_T * pListenHandle,
             const void * pUserRef,
             TRDP_MD_CALLBACK_T pfCbFunction,
             UINT32 comId,
             UINT32 etbTopoCnt,
             UINT32 opTrnTopoCnt,
             TRDP_IP_ADDR_T mcDestIpAddr,
             TRDP_FLAGS_T pktFlags,
             const TRDP_URI_USER_T destURI )
```

Subscribe to MD messages.

Add a listener to TRDP to get notified when messages are received

Parameters

in	*appHandle*	the handle returned by tlc_openSession
out	*pListenHandle*	Handle for this listener returned
in	*pUserRef*	user supplied value returned with received message
in	*pfCbFunction*	Pointer to listener specific callback function, NULL to use default function
in	*comId*	comId to be observed
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*mcDestIpAddr*	multicast group to listen on
in	*pktFlags*	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_MARSHALL, TRDP_PLAGS_TCP
in	*destURI*	only functional group of destination URI

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	out of memory
TRDP_NOINIT_ERR	handle invalid

5.9.2.25 tlm_confirm()

```
EXT_DECL TRDP_ERR_T tlm_confirm (
             TRDP_APP_SESSION_T appHandle,
```

```
            const TRDP_UUID_T * pSessionId,
            UINT16 userStatus,
            const TRDP_SEND_PARAM_T * pSendParam )
```

Initiate sending MD confirm message.

Send a MD confirmation message User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pSessionId*	Session ID returned by request
in	*userStatus*	Info for requester about application errors
in	*pSendParam*	Pointer to send parameters, NULL to use default send parameters

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	out of memory
TRDP_NO_SESSION_ERR	no such session
TRDP_NOINIT_ERR	handle invalid

5.9.2.26 tlm_delListener()

```
EXT_DECL TRDP_ERR_T tlm_delListener (
            TRDP_APP_SESSION_T appHandle,
            TRDP_LIS_T listenHandle )
```

Remove Listener.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
out	*listenHandle*	Handle for this listener

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_NOINIT_ERR	handle invalid

5.9.2.27 tlm_notify()

```
EXT_DECL TRDP_ERR_T tlm_notify (
            TRDP_APP_SESSION_T appHandle,
            const void * pUserRef,
```

```
                TRDP_MD_CALLBACK_T pfCbFunction,
                UINT32 comId,
                UINT32 etbTopoCnt,
                UINT32 opTrnTopoCnt,
                TRDP_IP_ADDR_T srcIpAddr,
                TRDP_IP_ADDR_T destIpAddr,
                TRDP_FLAGS_T pktFlags,
                const TRDP_SEND_PARAM_T * pSendParam,
                const UINT8 * pData,
                UINT32 dataSize,
                const TRDP_URI_USER_T sourceURI,
                const TRDP_URI_USER_T destURI )
```

Initiate sending MD notification message.

Send a MD notification message

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pUserRef*	user supplied value returned with reply
in	*pfCbFunction*	Pointer to listener specific callback function, NULL to use default function
in	*comId*	comId of packet to be sent
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*srcIpAddr*	own IP address, 0 - srcIP will be set by the stack
in	*destIpAddr*	where to send the packet to
in	*pktFlags*	OPTIONS: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_MARSHALL, TRDP_PLAGS_TCP
in	*pSendParam*	optional pointer to send parameter, NULL - default parameters are used
in	*pData*	pointer to packet data / dataset
in	*dataSize*	size of packet data
in	*sourceURI*	only functional group of source URI
in	*destURI*	only functional group of destination URI

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	out of memory
TRDP_NOINIT_ERR	handle invalid

5.9.2.28 tlm_readdListener()

```
EXT_DECL TRDP_ERR_T tlm_readdListener (
             TRDP_APP_SESSION_T appHandle,
             TRDP_LIS_T listenHandle,
             UINT32 etbTopoCnt,
             UINT32 opTrnTopoCnt,
             TRDP_IP_ADDR_T mcDestIpAddr )
```

Resubscribe to MD messages.

Readd a listener after topoCount changes to get notified when messages are received

Parameters

in	*appHandle*	the handle returned by tlc_openSession
out	*listenHandle*	Handle for this listener
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*mcDestIpAddr*	multicast group to listen on

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	out of memory
TRDP_NOINIT_ERR	handle invalid

5.9.2.29 tlm_reply()

```
TRDP_ERR_T tlm_reply (
            TRDP_APP_SESSION_T appHandle,
            const TRDP_UUID_T * pSessionId,
            UINT32 comId,
            UINT16 userStatus,
            const TRDP_SEND_PARAM_T * pSendParam,
            const UINT8 * pData,
            UINT32 dataSize )
```

Send a MD reply message.

Send a MD reply message after receiving an request User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pSessionId*	Session ID returned by indication
in	*comId*	comId of packet to be sent
in	*userStatus*	Info for requester about application errors
in	*pSendParam*	Pointer to send parameters, NULL to use default send parameters
in	*pData*	pointer to packet data / dataset
in	*dataSize*	size of packet data

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	Out of memory
TRDP_NO_SESSION_ERR	no such session
TRDP_NOINIT_ERR	handle invalid

5.9.2.30 tlm_replyErr()

```
TRDP_ERR_T tlm_replyErr (
            TRDP_APP_SESSION_T appHandle,
            const TRDP_UUID_T * pSessionId,
            UINT32 comId,
            TRDP_REPLY_STATUS_T replyStatus,
            const TRDP_SEND_PARAM_T * pSendParam )
```

Send a MD reply message.

Send a MD error reply message after receiving an request User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pSessionId*	Session ID returned by indication
in	*comId*	ComId for reply
in	*replyStatus*	Info for requester about stack errors
in	*pSendParam*	Pointer to send parameters, NULL to use default send parameters

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	out of memory
TRDP_NO_SESSION_ERR	no such session
TRDP_NOINIT_ERR	handle invalid

5.9.2.31 tlm_replyQuery()

```
TRDP_ERR_T tlm_replyQuery (
            TRDP_APP_SESSION_T appHandle,
            const TRDP_UUID_T * pSessionId,
            UINT32 comId,
            UINT16 userStatus,
            UINT32 confirmTimeout,
            const TRDP_SEND_PARAM_T * pSendParam,
            const UINT8 * pData,
            UINT32 dataSize )
```

Send a MD reply query message.

Send a MD reply query message after receiving a request and ask for confirmation. User reference, source and destination IP addresses as well as topo counts and packet flags are taken from the session

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pSessionId*	Session ID returned by indication

Parameters

in	*comId*	comId of packet to be sent
in	*userStatus*	Info for requester about application errors
in	*confirmTimeout*	timeout for confirmation
in	*pSendParam*	Pointer to send parameters, NULL to use default send parameters
in	*pData*	pointer to packet data / dataset
in	*dataSize*	size of packet data

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	out of memory
TRDP_NO_SESSION_ERR	no such session
TRDP_NOINIT_ERR	handle invalid

5.9.2.32 tlm_request()

```
EXT_DECL TRDP_ERR_T tlm_request (
             TRDP_APP_SESSION_T appHandle,
             const void * pUserRef,
             TRDP_MD_CALLBACK_T pfCbFunction,
             TRDP_UUID_T * pSessionId,
             UINT32 comId,
             UINT32 etbTopoCnt,
             UINT32 opTrnTopoCnt,
             TRDP_IP_ADDR_T srcIpAddr,
             TRDP_IP_ADDR_T destIpAddr,
             TRDP_FLAGS_T pktFlags,
             UINT32 numReplies,
             UINT32 replyTimeout,
             UINT32 maxNumRetries,
             const TRDP_SEND_PARAM_T * pSendParam,
             const UINT8 * pData,
             UINT32 dataSize,
             const TRDP_URI_USER_T sourceURI,
             const TRDP_URI_USER_T destURI )
```

Initiate sending MD request message.

Send a MD request message

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pUserRef*	user supplied value returned with reply
in	*pfCbFunction*	Pointer to listener specific callback function, NULL to use default function
out	*pSessionId*	return session ID
in	*comId*	comId of packet to be sent
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication

Parameters

in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*srcIpAddr*	own IP address, 0 - srcIP will be set by the stack
in	*destIpAddr*	where to send the packet to
in	*pktFlags*	OPTIONS: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_MARSHALL, TRDP_PLAGS_TCP
in	*numReplies*	number of expected replies, 0 if unknown
in	*replyTimeout*	timeout for reply
in	*maxNumRetries*	maximum number of retries (0 ... 2)
in	*pSendParam*	Pointer to send parameters, NULL to use default send parameters
in	*pData*	pointer to packet data / dataset
in	*dataSize*	size of packet data
in	*sourceURI*	only functional group of source URI
in	*destURI*	only functional group of destination URI

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	out of memory
TRDP_NOINIT_ERR	handle invalid

5.9.2.33 tlp_get()

```
EXT_DECL TRDP_ERR_T tlp_get (
             TRDP_APP_SESSION_T appHandle,
             TRDP_SUB_T subHandle,
             TRDP_PD_INFO_T * pPdInfo,
             UINT8 * pData,
             UINT32 * pDataSize )
```

Get the last valid PD message.

This allows polling of PDs instead of event driven handling by callback

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*subHandle*	the handle returned by subscription
in,out	*pPdInfo*	pointer to application's info buffer
in,out	*pData*	pointer to application's data buffer
in,out	*pDataSize*	in: size of buffer, out: size of data

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_SUB_ERR	not subscribed

Return values

TRDP_TIMEOUT_ERR	packet timed out
TRDP_NOINIT_ERR	handle invalid
TRDP_COMID_ERR	ComID not found when marshalling

5.9.2.34 tlp_getRedundant()

```
EXT_DECL TRDP_ERR_T tlp_getRedundant (
             TRDP_APP_SESSION_T appHandle,
             UINT32 redId,
             BOOL8 * pLeader )
```

Get status of redundant ComIds.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*redId*	will be set for all ComID's with the given redId, 0 for all redId
in,out	*pLeader*	TRUE if we send (leader)

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error / redId not existing
TRDP_NOINIT_ERR	handle invalid

5.9.2.35 tlp_publish()

```
EXT_DECL TRDP_ERR_T tlp_publish (
             TRDP_APP_SESSION_T appHandle,
             TRDP_PUB_T * pPubHandle,
             UINT32 comId,
             UINT32 etbTopoCnt,
             UINT32 opTrnTopoCnt,
             TRDP_IP_ADDR_T srcIpAddr,
             TRDP_IP_ADDR_T destIpAddr,
             UINT32 interval,
             UINT32 redId,
             TRDP_FLAGS_T pktFlags,
             const TRDP_SEND_PARAM_T * pSendParam,
             const UINT8 * pData,
             UINT32 dataSize )
```

Prepare for sending PD messages.

Queue a PD message, it will be send when tlc_publish has been called

Parameters

in	*appHandle*	the handle returned by tlc_openSession
out	*pPubHandle*	returned handle for related re/unpublish
in	*comId*	comId of packet to send
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*srcIpAddr*	own IP address, 0 - srcIP will be set by the stack
in	*destIpAddr*	where to send the packet to
in	*interval*	frequency of PD packet (>= 10ms) in usec
in	*redId*	0 - Non-redundant, > 0 valid redundancy group
in	*pktFlags*	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	*pSendParam*	optional pointer to send parameter, NULL - default parameters are used
in	*pData*	pointer to data packet / dataset, NULL if sending starts later with tlp_put()
in	*dataSize*	size of data packet >= 0 and <= TRDP_MAX_PD_DATA_SIZE

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	could not insert (out of memory)
TRDP_NOINIT_ERR	handle invalid

5.9.2.36 tlp_put()

```
EXT_DECL TRDP_ERR_T tlp_put (
             TRDP_APP_SESSION_T appHandle,
             TRDP_PUB_T pubHandle,
             const UINT8 * pData,
             UINT32 dataSize )
```

Update the process data to send.

Update previously published data. The new telegram will be sent earliest when tlc_process is called.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pubHandle*	the handle returned by publish
in,out	*pData*	pointer to application's data buffer
in,out	*dataSize*	size of data

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error on uninitialized parameter or changed dataSize compared to published one
TRDP_PUB_ERR	not published

Return values

TRDP_NOINIT_ERR	handle invalid
TRDP_COMID_ERR	ComID not found when marshalling

5.9.2.37 tlp_republish()

```
EXT_DECL TRDP_ERR_T tlp_republish (
             TRDP_APP_SESSION_T appHandle,
             TRDP_PUB_T pubHandle,
             UINT32 etbTopoCnt,
             UINT32 opTrnTopoCnt,
             TRDP_IP_ADDR_T srcIpAddr,
             TRDP_IP_ADDR_T destIpAddr )
```

Prepare for sending PD messages.

Reinitialize and queue a PD message, it will be send when tlc_publish has been called

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pubHandle*	handle for related unpublish
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*srcIpAddr*	own IP address, 0 - srcIP will be set by the stack
in	*destIpAddr*	where to send the packet to

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	could not insert (out of memory)
TRDP_NOINIT_ERR	handle invalid

5.9.2.38 tlp_request()

```
EXT_DECL TRDP_ERR_T tlp_request (
             TRDP_APP_SESSION_T appHandle,
             TRDP_SUB_T subHandle,
             UINT32 comId,
             UINT32 etbTopoCnt,
             UINT32 opTrnTopoCnt,
             TRDP_IP_ADDR_T srcIpAddr,
             TRDP_IP_ADDR_T destIpAddr,
             UINT32 redId,
             TRDP_FLAGS_T pktFlags,
             const TRDP_SEND_PARAM_T * pSendParam,
             const UINT8 * pData,
```

```
            UINT32 dataSize,
            UINT32 replyComId,
            TRDP_IP_ADDR_T replyIpAddr )
```

Initiate sending PD messages (PULL).

Send a PD request message

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*subHandle*	handle from related subscribe
in	*comId*	comId of packet to be sent
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*srcIpAddr*	own IP address, 0 - srcIP will be set by the stack
in	*destIpAddr*	where to send the packet to
in	*redId*	0 - Non-redundant, > 0 valid redundancy group
in	*pktFlags*	OPTIONS: TTRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	*pSendParam*	optional pointer to send parameter, NULL - default parameters are used
in	*pData*	pointer to packet data / dataset
in	*dataSize*	size of packet data
in	*replyComId*	comId of reply
in	*replyIpAddr*	IP for reply

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	could not insert (out of memory)
TRDP_NOINIT_ERR	handle invalid

5.9.2.39 tlp_resubscribe()

```
EXT_DECL TRDP_ERR_T tlp_resubscribe (
            TRDP_APP_SESSION_T appHandle,
            TRDP_SUB_T subHandle,
            UINT32 etbTopoCnt,
            UINT32 opTrnTopoCnt,
            TRDP_IP_ADDR_T srcIpAddr,
            TRDP_IP_ADDR_T destIpAddr )
```

Reprepare for receiving PD messages.

Resubscribe to a specific PD ComID and source IP

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*subHandle*	handle for this subscription

Parameters

in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*srcIpAddr*	IP for source filtering, set 0 if not used
in	*destIpAddr*	IP address to join

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	could not reserve memory (out of memory)
TRDP_NOINIT_ERR	handle invalid

5.9.2.40 tlp_setRedundant()

```
EXT_DECL TRDP_ERR_T tlp_setRedundant (
             TRDP_APP_SESSION_T appHandle,
             UINT32 redId,
             BOOL8 leader )
```

Do not send redundant PD's when we are follower.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*redId*	will be set for all ComID's with the given redId, 0 to change for all redId
in	*leader*	TRUE if we send

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error / redId not existing
TRDP_NOINIT_ERR	handle invalid

5.9.2.41 tlp_subscribe()

```
EXT_DECL TRDP_ERR_T tlp_subscribe (
             TRDP_APP_SESSION_T appHandle,
             TRDP_SUB_T * pSubHandle,
             const void * pUserRef,
             TRDP_PD_CALLBACK_T pfCbFunction,
             UINT32 comId,
             UINT32 etbTopoCnt,
             UINT32 opTrnTopoCnt,
             TRDP_IP_ADDR_T srcIpAddr,
             TRDP_IP_ADDR_T destIpAddr,
             TRDP_FLAGS_T pktFlags,
```

```
            UINT32 timeout,
            TRDP_TO_BEHAVIOR_T toBehavior )
```

Prepare for receiving PD messages.

Subscribe to a specific PD ComID and source IP

Parameters

in	*appHandle*	the handle returned by tlc_openSession
out	*pSubHandle*	return a handle for this subscription
in	*pUserRef*	user supplied value returned within the info structure
in	*pfCbFunction*	Pointer to subscriber specific callback function, NULL to use default function
in	*comId*	comId of packet to receive
in	*etbTopoCnt*	ETB topocount to use, 0 if consist local communication
in	*opTrnTopoCnt*	operational topocount, != 0 for orientation/direction sensitive communication
in	*srcIpAddr*	IP for source filtering, set 0 if not used Used e.g. for source filtering of redundant devices.
in	*destIpAddr*	IP address to join
in	*pktFlags*	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	*timeout*	timeout (>= 10ms) in usec
in	*toBehavior*	OPTION: TRDP_TO_DEFAULT, TRDP_TO_SET_TO_ZERO, TRDP_TO_KEEP_LAST_VALUE

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_MEM_ERR	could not reserve memory (out of memory)
TRDP_NOINIT_ERR	handle invalid

5.9.2.42 tlp_unpublish()

```
EXT_DECL TRDP_ERR_T tlp_unpublish (
            TRDP_APP_SESSION_T appHandle,
            TRDP_PUB_T pubHandle )
```

Stop sending PD messages.

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*pubHandle*	the handle returned by publish

Return values

TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_NOPUB_ERR	not published

Return values

TRDP_NOINIT_ERR	handle invalid

5.9.2.43 tlp_unsubscribe()

```
EXT_DECL TRDP_ERR_T tlp_unsubscribe (
             TRDP_APP_SESSION_T appHandle,
             TRDP_SUB_T subHandle )
```

Stop receiving PD messages.

Unsubscribe to a specific PD ComID

Parameters

in	*appHandle*	the handle returned by tlc_openSession
in	*subHandle*	the handle for this subscription

Return values

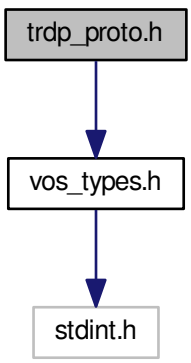
TRDP_NO_ERR	no error
TRDP_PARAM_ERR	parameter error
TRDP_SUB_ERR	not subscribed
TRDP_NOINIT_ERR	handle invalid

5.10 trdp_proto.h File Reference

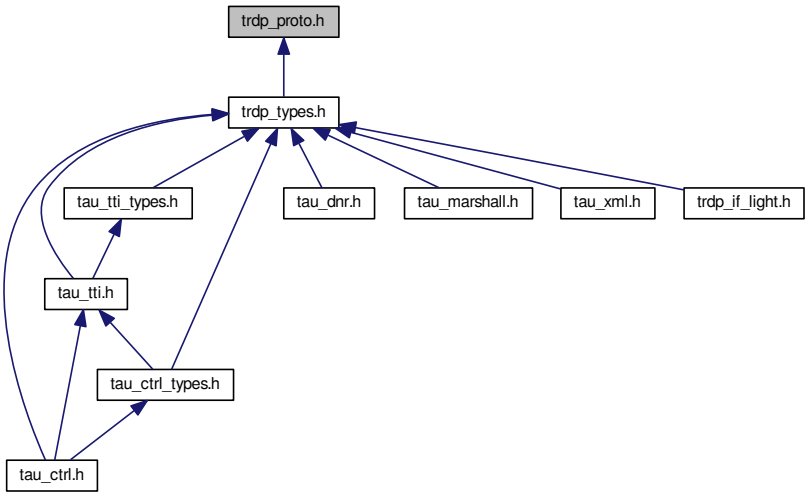
Definitions for the TRDP protocol.

```
#include "vos_types.h"
```

Include dependency graph for trdp_proto.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct GNU_PACKED

  Types for ETB control.
- struct GNU_PACKED

  Types for ETB control.

Macros

- #define TRDP_PD_UDP_PORT 17224

  process data UDP port
- #define TRDP_MD_UDP_PORT 17225

  message data UDP port
- #define TRDP_MD_TCP_PORT 17225

  message data TCP port
- #define TRDP_PROTO_VER 0x0100

  Protocol version.
- #define TRDP_PROTOCOL_VERSION_CHECK_MASK 0xFF00

  Version check, two digits are relevant.
- #define TRDP_SESS_ID_SIZE 16

  Session ID (UUID) size in MD header.
- #define TRDP_DEST_URI_SIZE 32

  max.
- #define TRDP_MIN_PD_HEADER_SIZE sizeof(PD_HEADER_T)

  PD header size with FCS.
- #define TRDP_MAX_PD_DATA_SIZE 1432

  PD data.
- #define TRDP_MAX_LABEL_LEN 16

  Maximum values.

- #define TRDP_MAX_URI_USER_LEN (2 * TRDP_MAX_LABEL_LEN)

  URI user part incl.
- #define TRDP_MAX_URI_HOST_LEN (4 * TRDP_MAX_LABEL_LEN)

  URI host part length incl.
- #define TRDP_MAX_URI_LEN ((6 * TRDP_MAX_LABEL_LEN) + 8)

  URI length incl.
- #define TRDP_MAX_FILE_NAME_LEN 128

  path and file name length incl.
- #define TDRP_VAR_SIZE 0

  Variable size dataset.
- #define TRDP_ETBCTRL_COMID 1

  TRDP reserved COMIDs in the range 1 ...
- #define TRDP_ETBCTRL_DSID 1

  TRDP reserved data set ids in the range 1 ...

Enumerations

- enum TRDP_MSG_T {
  TRDP_MSG_PD = 0x5064,
  TRDP_MSG_PP = 0x5070,
  TRDP_MSG_PR = 0x5072,
  TRDP_MSG_PE = 0x5065,
  TRDP_MSG_MN = 0x4D6E,
  TRDP_MSG_MR = 0x4D72,
  TRDP_MSG_MP = 0x4D70,
  TRDP_MSG_MQ = 0x4D71,
  TRDP_MSG_MC = 0x4D63,
  TRDP_MSG_ME = 0x4D65 }

  Message Types.

5.10.1 Detailed Description

Definitions for the TRDP protocol.

TRDP internal type definitions

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

Id

```
BL 2016-06-08: Ticket #120: ComIds for statistics changed to proposed 61375 errata
BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
```

5.10.2 Macro Definition Documentation

5.10.2.1 TRDP_DEST_URI_SIZE

```
#define TRDP_DEST_URI_SIZE 32
```

max.

Dest URI size in MD header

5.10.2.2 TRDP_ETBCTRL_COMID

```
#define TRDP_ETBCTRL_COMID 1
```

TRDP reserved COMIDs in the range 1 ...

1000

5.10.2.3 TRDP_ETBCTRL_DSID

```
#define TRDP_ETBCTRL_DSID 1
```

TRDP reserved data set ids in the range 1 ...

1000

5.10.2.4 TRDP_MAX_FILE_NAME_LEN

```
#define TRDP_MAX_FILE_NAME_LEN 128
```

path and file name length incl.

terminating '0'

5.10.2.5 TRDP_MAX_LABEL_LEN

```
#define TRDP_MAX_LABEL_LEN 16
```

Maximum values.

A uri is a string of the following form: trdp://[user part]@[host part] trdp://instLabel.funcLabel@devLabel.↩carLabel.cstLabel.trainLabel Hence the exact max. uri length is: 7 + (6 ∗ 15) + 5 ∗ (sizeof (separator)) + 1(terminating 0) to facilitate alignment the size will be increased by 1 byte label length incl. terminating '0'

5.10.2.6 TRDP_MAX_URI_HOST_LEN

```
#define TRDP_MAX_URI_HOST_LEN (4 * TRDP_MAX_LABEL_LEN)
```

URI host part length incl.

terminating '0'

5.10.2.7 TRDP_MAX_URI_LEN

```
#define TRDP_MAX_URI_LEN ((6 * TRDP_MAX_LABEL_LEN) + 8)
```

URI length incl.

terminating '0' and 1 padding byte

5.10.2.8 TRDP_MAX_URI_USER_LEN

```
#define TRDP_MAX_URI_USER_LEN (2 * TRDP_MAX_LABEL_LEN)
```

URI user part incl.

terminating '0'

5.10.3 Enumeration Type Documentation

5.10.3.1 TRDP_MSG_T

```
enum TRDP_MSG_T
```

Message Types.

Enumerator

TRDP_MSG_PD	'Pd' PD Data
TRDP_MSG_PP	'Pp' PD Data (Pull Reply)
TRDP_MSG_PR	'Pr' PD Request
TRDP_MSG_PE	'Pe' PD Error
TRDP_MSG_MN	'Mn' MD Notification (Request without reply)
TRDP_MSG_MR	'Mr' MD Request with reply
TRDP_MSG_MP	'Mp' MD Reply without confirmation
TRDP_MSG_MQ	'Mq' MD Reply with confirmation
TRDP_MSG_MC	'Mc' MD Confirm
TRDP_MSG_ME	'Me' MD Error

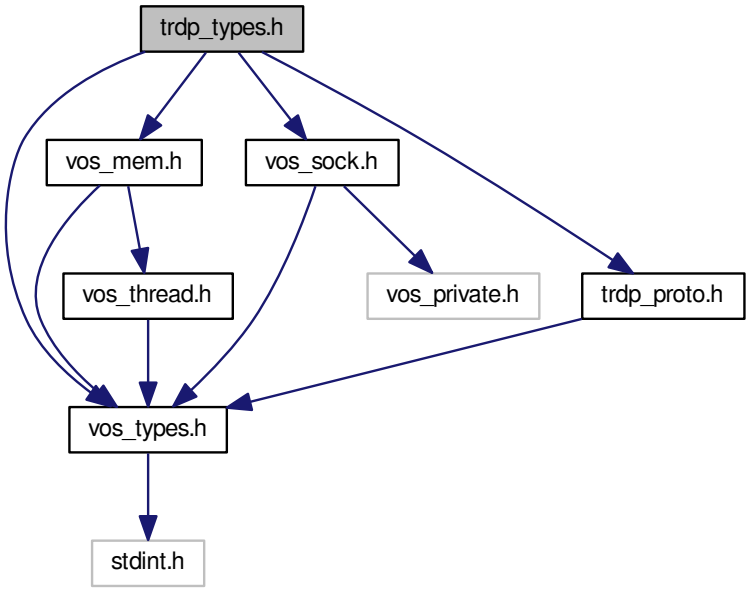
5.11 trdp_types.h File Reference

Typedefs for TRDP communication.

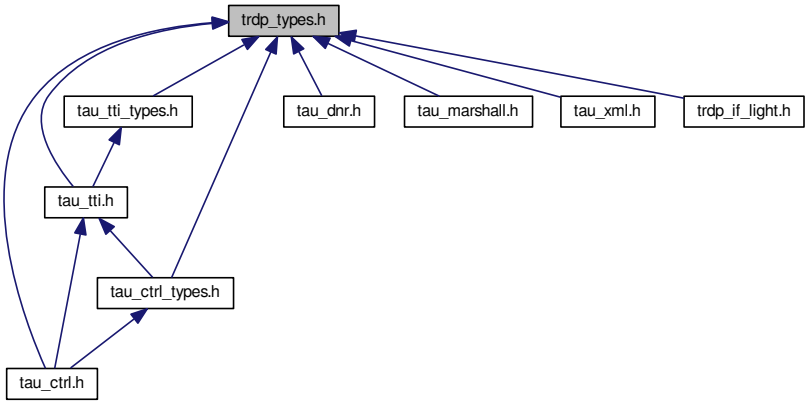
```
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_sock.h"
```

```
#include "trdp_proto.h"
```

Include dependency graph for trdp_types.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct TRDP_PD_INFO_T

  Process data info from received telegram; allows the application to generate responses.
- struct TRDP_MD_INFO_T

  Message data info from received telegram; allows the application to generate responses.

- struct TRDP_SEND_PARAM_T

  Quality/type of service and time to live.
- struct TRDP_DATASET_ELEMENT_T

  Dataset element definition.
- struct TRDP_DATASET

  Dataset definition.
- struct TRDP_COMID_DSID_MAP_T

  ComId - data set mapping element definition.
- struct TRDP_STATISTICS_REQUEST_T

  TRDP statistics type definitions.
- struct TRDP_MEM_STATISTICS_T

  Structure containing all general memory statistics information.
- struct TRDP_PD_STATISTICS_T

  Structure containing all general PD statistics information.
- struct TRDP_MD_STATISTICS_T

  Structure containing all general MD statistics information.
- struct TRDP_STATISTICS_T

  Structure containing all general memory, PD and MD statistics information.
- struct TRDP_SUBS_STATISTICS_T

  Table containing particular PD subscription information.
- struct TRDP_PUB_STATISTICS_T

  Table containing particular PD publishing information.
- struct TRDP_LIST_STATISTICS_T

  Information about a particular MD listener.
- struct TRDP_RED_STATISTICS_T

  A table containing PD redundant group information.
- struct TRDP_MARSHALL_CONFIG_T

  Marshaling/unmarshalling configuration.
- struct TRDP_PD_CONFIG_T

  Default PD configuration.
- struct TRDP_MD_CONFIG_T

  Default MD configuration.
- struct TRDP_MEM_CONFIG_T

  Enumeration type for memory pre-fragmentation, reuse of VOS definition.
- struct TRDP_PROCESS_CONFIG_T

  Various flags/general TRDP options for library initialization.

Macros

- #define USE_HEAP 0

  If this is set, we can allocate dynamically memory.
- #define TRDP_BOOL8 TRDP_BITSET8

  1 bit relevant (equal to zero = false, not equal to zero = true)
- #define TRDP_ANTIVALENT8 TRDP_BITSET8

  2 bit relevant (0x0 = errror, 0x01 = false, 0x02 = true, 0x03 undefined)

Typedefs

- typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T

  TRDP general type definitions.
- typedef VOS_VERSION_T TRDP_VERSION_T

  Version information.
- typedef VOS_TIME_T TRDP_TIME_T

  Timer value compatible with timeval / select.
- typedef VOS_FDS_T TRDP_FDS_T

  File descriptor set compatible with fd_set / select.
- typedef VOS_UUID_T TRDP_UUID_T

  UUID definition reuses the VOS definition.
- typedef struct TRDP_DATASET TRDP_DATASET_T

  Dataset definition.
- typedef TRDP_DATASET_T * pTRDP_DATASET_T

  Array of pointers to dataset.
- typedef VOS_PRINT_DBG_T TRDP_PRINT_DBG_T

  TRDP configuration type definitions.
- typedef VOS_LOG_T TRDP_LOG_T

  Categories for logging, reuse of the VOS definition.
- typedef TRDP_ERR_T(* TRDP_MARSHALL_T) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)

  Function type for marshalling .
- typedef TRDP_ERR_T(* TRDP_UNMARSHALL_T) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 srcSize, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)

  Function type for unmarshalling.
- typedef void(* TRDP_PD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_PD_INFO_T *pMsg, UINT8 *pData, UINT32 dataSize)

  Callback for receiving indications, timeouts, releases, responses.
- typedef void(* TRDP_MD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_MD_INFO_T *pMsg, UINT8 *pData, UINT32 dataSize)

  Callback for receiving indications, timeouts, releases, responses.

Enumerations

- enum TRDP_ERR_T {
  TRDP_NO_ERR = 0,
  TRDP_PARAM_ERR = -1,
  TRDP_INIT_ERR = -2,
  TRDP_NOINIT_ERR = -3,
  TRDP_TIMEOUT_ERR = -4,
  TRDP_NODATA_ERR = -5,
  TRDP_SOCK_ERR = -6,
  TRDP_IO_ERR = -7,
  TRDP_MEM_ERR = -8,
  TRDP_SEMA_ERR = -9,
  TRDP_QUEUE_ERR = -10,
  TRDP_QUEUE_FULL_ERR = -11,
  TRDP_MUTEX_ERR = -12,
  TRDP_THREAD_ERR = -13,
  TRDP_BLOCK_ERR = -14,
  TRDP_INTEGRATION_ERR = -15,
  TRDP_NOCONN_ERR = -16,

TRDP_NOSESSION_ERR = -30,
TRDP_SESSION_ABORT_ERR = -31,
TRDP_NOSUB_ERR = -32,
TRDP_NOPUB_ERR = -33,
TRDP_NOLIST_ERR = -34,
TRDP_CRC_ERR = -35,
TRDP_WIRE_ERR = -36,
TRDP_TOPO_ERR = -37,
TRDP_COMID_ERR = -38,
TRDP_STATE_ERR = -39,
TRDP_APP_TIMEOUT_ERR = -40,
TRDP_APP_REPLYTO_ERR = -41,
TRDP_APP_CONFIRMTO_ERR = -42,
TRDP_REPLYTO_ERR = -43,
TRDP_CONFIRMTO_ERR = -44,
TRDP_REQCONFIRMTO_ERR = -45,
TRDP_PACKET_ERR = -46,
TRDP_UNRESOLVED_ERR = -47,
TRDP_XML_PARSER_ERR = -48,
TRDP_INUSE_ERR = -49,
TRDP_MARSHALLING_ERR = -50,
TRDP_UNKNOWN_ERR = -99 }

Return codes for all API functions, -1..-29 taken over from vos.

- enum TRDP_REPLY_STATUS_T

  TRDP data transfer type definitions.

- enum TRDP_FLAGS_T {
  TRDP_FLAGS_DEFAULT = 0,
  TRDP_FLAGS_NONE = 0x01,
  TRDP_FLAGS_MARSHALL = 0x02,
  TRDP_FLAGS_CALLBACK = 0x04,
  TRDP_FLAGS_TCP = 0x08,
  TRDP_FLAGS_FORCE_CB = 0x10 }

  Various flags for PD and MD packets.

- enum TRDP_RED_STATE_T {
  TRDP_RED_FOLLOWER = 0,
  TRDP_RED_LEADER = 1 }

  Redundancy states.

- enum TRDP_TO_BEHAVIOR_T {
  TRDP_TO_DEFAULT = 0,
  TRDP_TO_SET_TO_ZERO = 1,
  TRDP_TO_KEEP_LAST_VALUE = 2 }

  How invalid PD shall be handled.

- enum TRDP_DATA_TYPE_T {
  TRDP_INVALID = 0,
  TRDP_BITSET8 = 1,
  TRDP_CHAR8 = 2,
  TRDP_UTF16 = 3,
  TRDP_INT8 = 4,
  TRDP_INT16 = 5,
  TRDP_INT32 = 6,
  TRDP_INT64 = 7,
  TRDP_UINT8 = 8,
  TRDP_UINT16 = 9,
  TRDP_UINT32 = 10,
  TRDP_UINT64 = 11,
  TRDP_REAL32 = 12,
  TRDP_REAL64 = 13,

TRDP_TIMEDATE32 = 14,
TRDP_TIMEDATE48 = 15,
TRDP_TIMEDATE64 = 16,
TRDP_TYPE_MAX = 30 }

TRDP dataset description definitions.

- enum TRDP_OPTION_T { ,
TRDP_OPTION_BLOCK = 0x01,
TRDP_OPTION_TRAFFIC_SHAPING = 0x02,
TRDP_OPTION_NO_REUSE_ADDR = 0x04,
TRDP_OPTION_NO_MC_LOOP_BACK = 0x08,
TRDP_OPTION_NO_UDP_CHK = 0x10 }

Various flags/general TRDP options for library initialization.

5.11.1 Detailed Description

Typedefs for TRDP communication.

F

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks


```
BL 2016-06-08: Ticket #120: ComIds for statistics changed to proposed 61375 errata
BL 2016-02-11: Ticket #111: 'unit', 'scale', 'offset' attributes added to TRDP_DATASET_ELEMENT
BL 2016-01-25: Ticket #106: User needs to be informed on every received PD packet
BL 2015-12-14: Ticket #33: source size check for marshalling
BL 2015-08-05: Ticket #81: Counts for packet loss
BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
BL 2014-02-27: Ticket #17: tlp_subscribe() returns wrong *pSubHandle
```

5.11.2 Typedef Documentation

5.11.2.1 TRDP_IP_ADDR_T

```
typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T
```

TRDP general type definitions.

5.11.2.2 TRDP_MARSHALL_T

```
typedef TRDP_ERR_T(* TRDP_MARSHALL_T) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 src↩
Size, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)
```

Function type for marshalling .

The function must know about the dataset's alignment etc.

Parameters

in	*pRefCon	pointer to user context
in	*comId*	ComId to identify the structure out of a configuration
in	*pSrc	pointer to received original message
in	*srcSize*	size of the source buffer
in	*pDst	pointer to a buffer for the treated message
in,out	*pDstSize	size of the provide buffer / size of the treated message
in,out	*ppCachedDS	pointer to pointer of cached dataset

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provided buffer to small
TRDP_COMID_ERR	comid not existing

5.11.2.3 TRDP_MD_CALLBACK_T

```
typedef void(* TRDP_MD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_M↩
D_INFO_T *pMsg, UINT8 *pData, UINT32 dataSize)
```

Callback for receiving indications, timeouts, releases, responses.

Parameters

in	*appHandle*	handle returned also by tlc_init
in	*pRefCon	pointer to user context
in	*pMsg	pointer to received message information
in	*pData	pointer to received data
in	*dataSize*	size of received data pointer to received data

5.11.2.4 TRDP_PD_CALLBACK_T

```
typedef void(* TRDP_PD_CALLBACK_T) (void *pRefCon, TRDP_APP_SESSION_T appHandle, const TRDP_P↩
D_INFO_T *pMsg, UINT8 *pData, UINT32 dataSize)
```

Callback for receiving indications, timeouts, releases, responses.

Parameters

in	*pRefCon	pointer to user context
in	*appHandle*	application handle returned by tlc_openSession
in	*pMsg	pointer to received message information
in	*pData	pointer to received data
in	*dataSize*	size of received data pointer to received data

5.11.2.5 TRDP_PRINT_DBG_T

```
typedef VOS_PRINT_DBG_T TRDP_PRINT_DBG_T
```

TRDP configuration type definitions.

Callback function definition for error/debug output, reuse of the VOS defined function.

5.11.2.6 TRDP_TIME_T

```
typedef VOS_TIME_T TRDP_TIME_T
```

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

5.11.2.7 TRDP_UNMARSHALL_T

```
typedef TRDP_ERR_T(* TRDP_UNMARSHALL_T) (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT32 src↩
Size, UINT8 *pDst, UINT32 *pDstSize, TRDP_DATASET_T **ppCachedDS)
```

Function type for unmarshalling.

The function must know about the dataset's alignment etc.

Parameters

in	*pRefCon	pointer to user context
in	*comId*	ComId to identify the structure out of a configuration
in	*pSrc	pointer to received original message
in	*srcSize*	data length from TRDP packet header
in	*pDst	pointer to a buffer for the treated message
in,out	*pDstSize	size of the provide buffer / size of the treated message
in,out	*ppCachedDS	pointer to pointer of cached dataset

Return values

TRDP_NO_ERR	no error
TRDP_MEM_ERR	provide buffer to small
TRDP_COMID_ERR	comid not existing

5.11.3 Enumeration Type Documentation

5.11.3.1 TRDP_DATA_TYPE_T

```
enum TRDP_DATA_TYPE_T
```

TRDP dataset description definitions.

Dataset element definition

Enumerator

TRDP_INVALID	Invalid/unknown.
TRDP_BITSET8	=UINT8
TRDP_CHAR8	char, can be used also as UTF8
TRDP_UTF16	Unicode UTF-16 character.
TRDP_INT8	Signed integer, 8 bit.
TRDP_INT16	Signed integer, 16 bit.
TRDP_INT32	Signed integer, 32 bit.
TRDP_INT64	Signed integer, 64 bit.
TRDP_UINT8	Unsigned integer, 8 bit.
TRDP_UINT16	Unsigned integer, 16 bit.
TRDP_UINT32	Unsigned integer, 32 bit.
TRDP_UINT64	Unsigned integer, 64 bit.
TRDP_REAL32	Floating point real, 32 bit.
TRDP_REAL64	Floating point real, 64 bit.
TRDP_TIMEDATE32	32 bit UNIX time
TRDP_TIMEDATE48	48 bit TCN time (32 bit UNIX time and 16 bit ticks)
TRDP_TIMEDATE64	32 bit UNIX time + 32 bit microseconds (== struct timeval)
TRDP_TYPE_MAX	Values greater are considered nested datasets.

5.11.3.2 TRDP_ERR_T

```
enum TRDP_ERR_T
```

Return codes for all API functions, -1..-29 taken over from vos.

Enumerator

TRDP_NO_ERR	No error.
TRDP_PARAM_ERR	Parameter missing or out of range.
TRDP_INIT_ERR	Call without valid initialization.
TRDP_NOINIT_ERR	Call with invalid handle.
TRDP_TIMEOUT_ERR	Timout.
TRDP_NODATA_ERR	Non blocking mode: no data received.
TRDP_SOCK_ERR	Socket error / option not supported.
TRDP_IO_ERR	Socket IO error, data can't be received/sent.
TRDP_MEM_ERR	No more memory available.
TRDP_SEMA_ERR	Semaphore not available.
TRDP_QUEUE_ERR	Queue empty.
TRDP_QUEUE_FULL_ERR	Queue full.
TRDP_MUTEX_ERR	Mutex not available.
TRDP_THREAD_ERR	Thread error.
TRDP_BLOCK_ERR	System call would have blocked in blocking mode.
TRDP_INTEGRATION_ERR	Alignment or endianess for selected target wrong.
TRDP_NOCONN_ERR	No TCP connection.
TRDP_NOSESSION_ERR	No such session.
TRDP_SESSION_ABORT_ERR	Session aborted.
TRDP_NOSUB_ERR	No subscriber.

Enumerator

TRDP_NOPUB_ERR	No publisher.
TRDP_NOLIST_ERR	No listener.
TRDP_CRC_ERR	Wrong CRC.
TRDP_WIRE_ERR	Wire.
TRDP_TOPO_ERR	Invalid topo count.
TRDP_COMID_ERR	Unknown ComId.
TRDP_STATE_ERR	Call in wrong state.
TRDP_APP_TIMEOUT_ERR	Application Timeout.
TRDP_APP_REPLYTO_ERR	Application Reply Sent Timeout.
TRDP_APP_CONFIRMTO_ERR	Application Confirm Sent Timeout.
TRDP_REPLYTO_ERR	Protocol Reply Timeout.
TRDP_CONFIRMTO_ERR	Protocol Confirm Timeout.
TRDP_REQCONFIRMTO_ERR	Protocol Confirm Timeout (Request sender)
TRDP_PACKET_ERR	Incomplete message data packet.
TRDP_UNRESOLVED_ERR	DNR: address could not be resolved.
TRDP_XML_PARSER_ERR	Returned by the tau_xml subsystem.
TRDP_INUSE_ERR	Resource is still in use.
TRDP_MARSHALLING_ERR	Source size exceeded, dataset mismatch.
TRDP_UNKNOWN_ERR	Unspecified error.

5.11.3.3 TRDP_FLAGS_T

`enum TRDP_FLAGS_T`

Various flags for PD and MD packets.

Enumerator

TRDP_FLAGS_DEFAULT	Default value defined in tlc_openDession will be taken.
TRDP_FLAGS_NONE	No flags set.
TRDP_FLAGS_MARSHALL	Optional marshalling/unmarshalling in TRDP stack.
TRDP_FLAGS_CALLBACK	Use of callback function.
TRDP_FLAGS_TCP	Use TCP for message data.
TRDP_FLAGS_FORCE_CB	Force a callback for every received packet.

5.11.3.4 TRDP_OPTION_T

`enum TRDP_OPTION_T`

Various flags/general TRDP options for library initialization.

Enumerator

TRDP_OPTION_BLOCK	Default: Use nonblocking I/O calls, polling necessary Set: Read calls will block, use select()

Enumerator

TRDP_OPTION_TRAFFIC_SHAPING	Use traffic shaping - distribute packet sending Default: OFF.
TRDP_OPTION_NO_REUSE_ADDR	Do not allow re-use of address/port (-> no multihoming) Default: Allow.
TRDP_OPTION_NO_MC_LOOP_BACK	Do not allow loop back of multicast traffic Default: Allow.
TRDP_OPTION_NO_UDP_CHK	Suppress UDP CRC generation Default: Compute UDP CRC.

5.11.3.5 TRDP_RED_STATE_T

`enum TRDP_RED_STATE_T`

Redundancy states.

Enumerator

TRDP_RED_FOLLOWER	Redundancy follower - redundant PD will be not sent out.
TRDP_RED_LEADER	Redundancy leader - redundant PD will be sent out.

5.11.3.6 TRDP_REPLY_STATUS_T

`enum TRDP_REPLY_STATUS_T`

TRDP data transfer type definitions.

Reply status messages

5.11.3.7 TRDP_TO_BEHAVIOR_T

`enum TRDP_TO_BEHAVIOR_T`

How invalid PD shall be handled.

Enumerator

TRDP_TO_DEFAULT	Default value defined in tlc_openDession will be taken.
TRDP_TO_SET_TO_ZERO	If set, data will be reset to zero on time out.
TRDP_TO_KEEP_LAST_VALUE	If set, last received values will be returned.

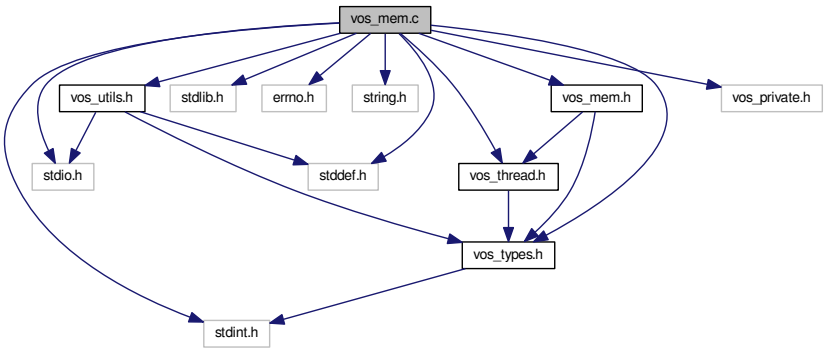
5.12 vos_mem.c File Reference

Memory functions.

```
#include <stdio.h>
#include <stddef.h>
```

```
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include "vos_types.h"
#include "vos_utils.h"
#include "vos_mem.h"
#include "vos_thread.h"
#include "vos_private.h"
```

Include dependency graph for vos_mem.c:



Functions

- EXT_DECL VOS_ERR_T vos_memInit (UINT8 *pMemoryArea, UINT32 size, const UINT32 fragMem[VOS_MEM_NBLOCKSIZES])

  Initialize the memory unit.
- EXT_DECL void vos_memDelete (UINT8 *pMemoryArea)

  Delete the memory area.
- EXT_DECL UINT8 * vos_memAlloc (UINT32 size)

  Allocate a block of memory (from memory area above).
- EXT_DECL void vos_memFree (void *pMemBlock)

  Deallocate a block of memory (from memory area above).
- EXT_DECL VOS_ERR_T vos_memCount (UINT32 *pAllocatedMemory, UINT32 *pFreeMemory, UINT32 *pMinFree, UINT32 *pNumAllocBlocks, UINT32 *pNumAllocErr, UINT32 *pNumFreeErr, UINT32 blockSize[VOS_MEM_NBLOCKSIZES], UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES])

  Return used and available memory (of memory area above).
- EXT_DECL void vos_qsort (void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))

  Sort an array.
- EXT_DECL void * vos_bsearch (const void *pKey, const void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))

  Binary search in a sorted array.
- EXT_DECL INT32 vos_strnicmp (const CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)

  Case insensitive string compare.
- EXT_DECL void vos_strncpy (CHAR8 *pStrDst, const CHAR8 *pStrSrc, UINT32 count)

  String copy with length limitation.
- EXT_DECL void vos_strncat (CHAR8 *pStrDst, UINT32 count, const CHAR8 *pStrSrc)

  String concatenation with length limitation.

- EXT_DECL VOS_ERR_T vos_queueCreate (VOS_QUEUE_POLICY_T queueType, UINT32 maxNoOfMsg, VOS_QUEUE_T *pQueueHandle)

  Initialize a message queue.
- EXT_DECL VOS_ERR_T vos_queueSend (VOS_QUEUE_T queueHandle, UINT8 *pData, UINT32 size)

  Send a message.
- EXT_DECL VOS_ERR_T vos_queueReceive (VOS_QUEUE_T queueHandle, UINT8 **ppData, UINT32 *pSize, UINT32 usTimeout)

  Get a message.
- EXT_DECL VOS_ERR_T vos_queueDestroy (VOS_QUEUE_T queueHandle)

  Destroy a message queue.

5.12.1 Detailed Description

Memory functions.

OS abstraction of memory access and control

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

Id

Changes: BL 2016-07-06: Ticket #122 64Bit compatibility (+ compiler warnings) BL 2016-02-10: Debug print: tabs before size output BL 2012-12-03: ID 1: "using uninitialized PD_ELE_T.pullpAddress variable" ID 2: "uninitialized PD_ELE_T newPD->pNext in tlp_subscribe()"

5.12.2 Function Documentation

5.12.2.1 vos_bsearch()

```
EXT_DECL void* vos_bsearch (
            const void * pKey,
            const void * pBuf,
            UINT32 num,
            UINT32 size,
            int(*)(const void *, const void *) compare )
```

Binary search in a sorted array.

This is just a wrapper for the standard bsearch function.

Parameters

in	*pKey*	Key to search for
in	*pBuf*	Pointer to the array to search
in	*num*	number of elements
in	*size*	size of one element
in	*compare*	Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values

Pointer	to found element or NULL

5.12.2.2 vos_memAlloc()

```
EXT_DECL UINT8* vos_memAlloc (
            UINT32 size )
```

Allocate a block of memory (from memory area above).

Parameters

in	*size*	Size of requested block

Return values

Pointer	to memory area
NULL	if no memory available

5.12.2.3 vos_memCount()

```
EXT_DECL VOS_ERR_T vos_memCount (
            UINT32 * pAllocatedMemory,
            UINT32 * pFreeMemory,
            UINT32 * pMinFree,
            UINT32 * pNumAllocBlocks,
            UINT32 * pNumAllocErr,
            UINT32 * pNumFreeErr,
            UINT32 blockSize[VOS_MEM_NBLOCKSIZES],
            UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES] )
```

Return used and available memory (of memory area above).

Parameters

out	*pAllocatedMemory*	Pointer to allocated memory size
out	*pFreeMemory*	Pointer to free memory size
out	*pMinFree*	Pointer to minimal free memory size in statistics interval

Parameters

out	*pNumAllocBlocks*	Pointer to number of allocated memory blocks
out	*pNumAllocErr*	Pointer to number of allocation errors
out	*pNumFreeErr*	Pointer to number of free errors
out	*blockSize*	Pointer to list of memory block sizes
out	*usedBlockSize*	Pointer to list of used memoryblocks

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised

5.12.2.4 vos_memDelete()

```
EXT_DECL void vos_memDelete (
            UINT8 * pMemoryArea )
```

Delete the memory area.

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters

in	*pMemoryArea*	Pointer to memory area used

5.12.2.5 vos_memFree()

```
EXT_DECL void vos_memFree (
            void * pMemBlock )
```

Deallocate a block of memory (from memory area above).

Parameters

in	*pMemBlock*	Pointer to memory block to be freed

5.12.2.6 vos_memInit()

```
EXT_DECL VOS_ERR_T vos_memInit (
            UINT8 * pMemoryArea,
            UINT32 size,
            const UINT32 fragMem[VOS_MEM_NBLOCKSIZES] )
```

Initialize the memory unit.

Init a supplied block of memory and prepare it for use with vos_memAlloc and vos_memFree. The used block sizes can be supplied and will be preallocated. If half of the overall size of the requested memory area would be pre-allocated, either by the default pre-allocation table or a provided one, no pre-allocation takes place.

Parameters

in	*pMemoryArea*	Pointer to memory area to use
in	*size*	Size of provided memory area
in	*fragMem*	Pointer to list of preallocated block sizes, used to fragment memory for large blocks

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_MEM_ERR	no memory available
VOS_MUTEX_ERR	no mutex available

5.12.2.7 vos_qsort()

```
EXT_DECL void vos_qsort (
            void * pBuf,
            UINT32 num,
            UINT32 size,
            int(*)(const void *, const void *) compare )
```

Sort an array.

This is just a wrapper for the standard qsort function.

Parameters

in,out	*pBuf*	Pointer to the array to sort
in	*num*	number of elements
in	*size*	size of one element
in	*compare*	Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values

none	

5.12.2.8 vos_queueCreate()

```
EXT_DECL VOS_ERR_T vos_queueCreate (
            VOS_QUEUE_POLICY_T queueType,
            UINT32 maxNoOfMsg,
            VOS_QUEUE_T * pQueueHandle )
```

Initialize a message queue.

Returns a handle for further calls

Parameters

in	*queueType*	Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)
in	*maxNoOfMsg*	Maximum number of messages
out	*pQueueHandle*	Handle of created queue

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_INIT_ERR	not supported
VOS_QUEUE_ERR	error creating queue

5.12.2.9 vos_queueDestroy()

```
EXT_DECL VOS_ERR_T vos_queueDestroy (
            VOS_QUEUE_T queueHandle )
```

Destroy a message queue.

Free all resources used by this queue

Parameters

in	*queueHandle*	Queue handle

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid

5.12.2.10 vos_queueReceive()

```
EXT_DECL VOS_ERR_T vos_queueReceive (
            VOS_QUEUE_T queueHandle,
            UINT8 ** ppData,
            UINT32 * pSize,
            UINT32 usTimeout )
```

Get a message.

Parameters

in	*queueHandle*	Queue handle

Parameters

out	*ppData*	Pointer to data pointer to be received
out	*pSize*	Size of receive data
in	*usTimeout*	Maximum time to wait for a message (in usec)

Return values

VOSNO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_QUEUE_ERR	queue is empty

5.12.2.11 vos_queueSend()

```
EXT_DECL VOS_ERR_T vos_queueSend (
            VOS_QUEUE_T queueHandle,
            UINT8 * pData,
            UINT32 size )
```

Send a message.

Parameters

in	*queueHandle*	Queue handle
in	*pData*	Pointer to data to be sent
in	*size*	Size of data to be sent

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_INIT_ERR	not supported
VOS_QUEUE_ERR	error creating queue

5.12.2.12 vos_strncat()

```
EXT_DECL void vos_strncat (
            CHAR8 * pStrDst,
            UINT32 count,
            const CHAR8 * pStrSrc )
```

String concatenation with length limitation.

Parameters

in	*pStrDst*	Destination string
in	*count*	Size of destination buffer
in	*pStrSrc*	Null terminated string to append

Return values

none	

5.12.2.13 vos_strncpy()

```
EXT_DECL void vos_strncpy (
            CHAR8 * pStrDst,
            const CHAR8 * pStrSrc,
            UINT32 count )
```

String copy with length limitation.

Parameters

in	*pStrDst*	Destination string
in	*pStrSrc*	Null terminated string to copy
in	*count*	Maximum number of characters to copy

Return values

none	

5.12.2.14 vos_strnicmp()

```
EXT_DECL INT32 vos_strnicmp (
            const CHAR8 * pStr1,
            const CHAR8 * pStr2,
            UINT32 count )
```

Case insensitive string compare.

Parameters

in	*pStr1*	Null terminated string to compare
in	*pStr2*	Null terminated string to compare
in	*count*	Maximum number of characters to compare

Return values

0	- equal

Return values

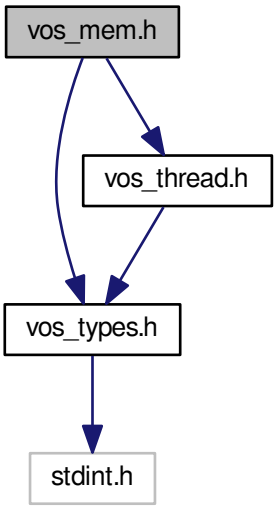
<0	- string1 less than string 2
>0	- string 1 greater than string 2

5.13 vos_mem.h File Reference

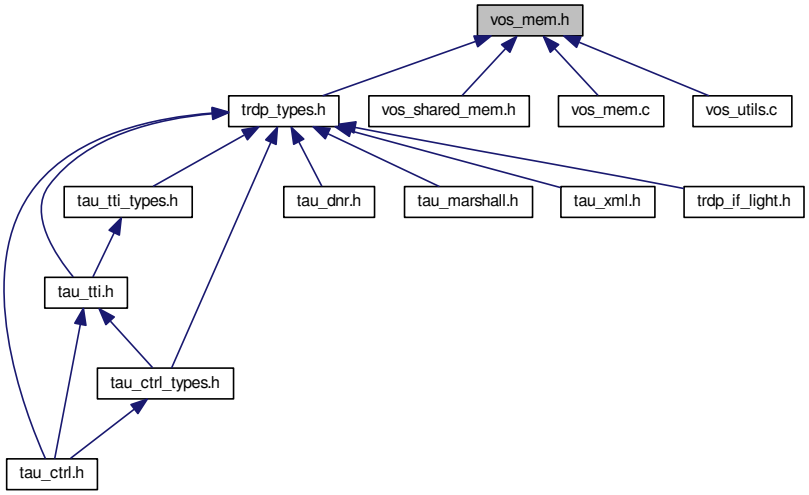
Memory and queue functions for OS abstraction.

```
#include "vos_types.h"
#include "vos_thread.h"
```

Include dependency graph for vos_mem.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define VOS_MEM_BLOCKSIZES

  We internally allocate memory always by these block sizes.
- #define VOS_MEM_PREALLOCATE {0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0}

  Default pre-allocation of free memory blocks.

Typedefs

- typedef struct VOS_QUEUE * VOS_QUEUE_T

  Opaque queue define.

Enumerations

- enum VOS_QUEUE_POLICY_T

  Queue policy matching pthread/Posix defines.

Functions

- EXT_DECL VOS_ERR_T vos_memInit (UINT8 *pMemoryArea, UINT32 size, const UINT32 fragMem[VOS_MEM_NBLOCKSIZES])

  Initialize the memory unit.
- EXT_DECL void vos_memDelete (UINT8 *pMemoryArea)

  Delete the memory area.
- EXT_DECL UINT8 * vos_memAlloc (UINT32 size)

  Allocate a block of memory (from memory area above).
- EXT_DECL void vos_memFree (void *pMemBlock)

Deallocate a block of memory (from memory area above).

- EXT_DECL VOS_ERR_T vos_memCount (UINT32 *pAllocatedMemory, UINT32 *pFreeMemory, UINT32 *pMinFree, UINT32 *pNumAllocBlocks, UINT32 *pNumAllocErr, UINT32 *pNumFreeErr, UINT32 blockSize[VOS_MEM_NBLOCKSIZES], UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES])

  Return used and available memory (of memory area above).
- EXT_DECL void vos_qsort (void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))

  Sort an array.
- EXT_DECL void * vos_bsearch (const void *pKey, const void *pBuf, UINT32 num, UINT32 size, int(*compare)(const void *, const void *))

  Binary search in a sorted array.
- EXT_DECL INT32 vos_strnicmp (const CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)

  Case insensitive string compare.
- EXT_DECL void vos_strncpy (CHAR8 *pStr1, const CHAR8 *pStr2, UINT32 count)

  String copy with length limitation.
- EXT_DECL void vos_strncat (CHAR8 *pStrDst, UINT32 count, const CHAR8 *pStrSrc)

  String concatenation with length limitation.
- EXT_DECL VOS_ERR_T vos_queueCreate (VOS_QUEUE_POLICY_T queueType, UINT32 maxNoOfMsg, VOS_QUEUE_T *pQueueHandle)

  Initialize a message queue.
- EXT_DECL VOS_ERR_T vos_queueSend (VOS_QUEUE_T queueHandle, UINT8 *pData, UINT32 size)

  Send a message.
- EXT_DECL VOS_ERR_T vos_queueReceive (VOS_QUEUE_T queueHandle, UINT8 **ppData, UINT32 *pSize, UINT32 usTimeout)

  Get a message.
- EXT_DECL VOS_ERR_T vos_queueDestroy (VOS_QUEUE_T queueHandle)

  Destroy a message queue.

5.13.1 Detailed Description

Memory and queue functions for OS abstraction.

This module provides memory control supervison

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH Peter Brander (Memory scheme)

Remarks

Id

5.13.2 Macro Definition Documentation

5.13.2.1 VOS_MEM_BLOCKSIZES

```
#define VOS_MEM_BLOCKSIZES
```

Value:

```
{32, 48, 128, 180, 256, 512, 1024, 1480, 2048, \
                             4096, 11520, 16384, 32768, 65536, 131072}
```

We internally allocate memory always by these block sizes.

The largest available block is 524288 Bytes, provided the overal size of the used memory allocation area is larger.

5.13.2.2 VOS_MEM_PREALLOCATE

```
#define VOS_MEM_PREALLOCATE {0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0}
```

Default pre-allocation of free memory blocks.

To avoid problems with too many small blocks and no large one. Specify how many of each block size that should be pre-allocated (and freed!) to pre-segment the memory area.

5.13.3 Function Documentation

5.13.3.1 vos_bsearch()

```
EXT_DECL void* vos_bsearch (
            const void * pKey,
            const void * pBuf,
            UINT32 num,
            UINT32 size,
            int(*)(const void *, const void *) compare )
```

Binary search in a sorted array.

This is just a wrapper for the standard bsearch function.

Parameters

in	*pKey*	Key to search for
in	*pBuf*	Pointer to the array to search
in	*num*	number of elements
in	*size*	size of one element
in	*compare*	Pointer to compare function return -n if arg1 < arg2, return 0 if arg1 == arg2, return +n if arg1 > arg2 where n is an integer != 0

Return values

Pointer	to found element or NULL

5.13.3.2 vos_memAlloc()

```
EXT_DECL UINT8* vos_memAlloc (
            UINT32 size )
```

Allocate a block of memory (from memory area above).

Parameters

in	*size*	Size of requested block

Return values

Pointer	to memory area
NULL	if no memory available

5.13.3.3 vos_memCount()

```
EXT_DECL VOS_ERR_T vos_memCount (
            UINT32 * pAllocatedMemory,
            UINT32 * pFreeMemory,
            UINT32 * pMinFree,
            UINT32 * pNumAllocBlocks,
            UINT32 * pNumAllocErr,
            UINT32 * pNumFreeErr,
            UINT32 blockSize[VOS_MEM_NBLOCKSIZES],
            UINT32 usedBlockSize[VOS_MEM_NBLOCKSIZES] )
```

Return used and available memory (of memory area above).

Parameters

out	*pAllocatedMemory*	Pointer to allocated memory size
out	*pFreeMemory*	Pointer to free memory size
out	*pMinFree*	Pointer to minimal free memory size in statistics interval
out	*pNumAllocBlocks*	Pointer to number of allocated memory blocks
out	*pNumAllocErr*	Pointer to number of allocation errors
out	*pNumFreeErr*	Pointer to number of free errors
out	*blockSize*	Pointer to list of memory block sizes
out	*usedBlockSize*	Pointer to list of used memoryblocks

Return values

VOS_NO_ERR	no error

Return values

VOS_INIT_ERR	module not initialised

5.13.3.4 vos_memDelete()

```
EXT_DECL void vos_memDelete (
            UINT8 * pMemoryArea )
```

Delete the memory area.

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters

in	*pMemoryArea*	Pointer to memory area to use

This will eventually invalidate any previously allocated memory blocks! It should be called last before the application quits. No further access to the memory blocks is allowed after this call.

Parameters

in	*pMemoryArea*	Pointer to memory area used

5.13.3.5 vos_memFree()

```
EXT_DECL void vos_memFree (
            void * pMemBlock )
```

Deallocate a block of memory (from memory area above).

Parameters

in	*pMemBlock*	Pointer to memory block to be freed

5.13.3.6 vos_memInit()

```
EXT_DECL VOS_ERR_T vos_memInit (
            UINT8 * pMemoryArea,
            UINT32 size,
            const UINT32 fragMem[VOS_MEM_NBLOCKSIZES] )
```

Initialize the memory unit.

Init a supplied block of memory and prepare it for use with vos_alloc and vos_dealloc. The used block sizes can be supplied and will be preallocated.

Parameters

in	*pMemoryArea*	Pointer to memory area to use
in	*size*	Size of provided memory area
in	*fragMem*	Pointer to list of preallocate block sizes, used to fragment memory for large blocks

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_MEM_ERR	no memory available

Init a supplied block of memory and prepare it for use with vos_memAlloc and vos_memFree. The used block sizes can be supplied and will be preallocated. If half of the overall size of the requested memory area would be pre-allocated, either by the default pre-allocation table or a provided one, no pre-allocation takes place.

Parameters

in	*pMemoryArea*	Pointer to memory area to use
in	*size*	Size of provided memory area
in	*fragMem*	Pointer to list of preallocated block sizes, used to fragment memory for large blocks

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_MEM_ERR	no memory available
VOS_MUTEX_ERR	no mutex available

5.13.3.7 vos_qsort()

```
EXT_DECL void vos_qsort (
            void * pBuf,
            UINT32 num,
            UINT32 size,
            int(*)(const void *, const void *) compare )
```

Sort an array.

This is just a wrapper for the standard qsort function.

Parameters

in,out	*pBuf*	Pointer to the array to sort
in	*num*	number of elements
in	*size*	size of one element
in	*compare*	Pointer to compare function return -n if arg1 $<$ arg2, return 0 if arg1 == arg2, return +n if arg1 $>$ arg2 where n is an integer != 0

Return values

none	

5.13.3.8 vos_queueCreate()

```
EXT_DECL VOS_ERR_T vos_queueCreate (
            VOS_QUEUE_POLICY_T queueType,
            UINT32 maxNoOfMsg,
            VOS_QUEUE_T * pQueueHandle )
```

Initialize a message queue.

Returns a handle for further calls

Parameters

in	*queueType*	Define queue type (1 = FIFO, 2 = LIFO, 3 = PRIO)
in	*maxNoOfMsg*	Maximum number of messages
out	*pQueueHandle*	Handle of created queue

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_INIT_ERR	not supported
VOS_QUEUE_ERR	error creating queue

5.13.3.9 vos_queueDestroy()

```
EXT_DECL VOS_ERR_T vos_queueDestroy (
            VOS_QUEUE_T queueHandle )
```

Destroy a message queue.

Free all resources used by this queue

Parameters

in	*queueHandle*	Queue handle

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle

Return values

VOS_PARAM_ERR	parameter out of range/invalid

5.13.3.10 vos_queueReceive()

```
EXT_DECL VOS_ERR_T vos_queueReceive (
            VOS_QUEUE_T queueHandle,
            UINT8 ** ppData,
            UINT32 * pSize,
            UINT32 usTimeout )
```

Get a message.

Parameters

in	*queueHandle*	Queue handle
out	*ppData*	Pointer to data pointer to be received
out	*pSize*	Size of receive data
in	*usTimeout*	Maximum time to wait for a message (in usec)

Return values

VOSNO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_QUEUE_ERR	queue is empty

5.13.3.11 vos_queueSend()

```
EXT_DECL VOS_ERR_T vos_queueSend (
            VOS_QUEUE_T queueHandle,
            UINT8 * pData,
            UINT32 size )
```

Send a message.

Parameters

in	*queueHandle*	Queue handle
in	*pData*	Pointer to data to be sent
in	*size*	Size of data to be sent

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised

Return values

VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_INIT_ERR	not supported
VOS_QUEUE_ERR	error creating queue

5.13.3.12 vos_strncat()

```
EXT_DECL void vos_strncat (
            CHAR8 * pStrDst,
            UINT32 count,
            const CHAR8 * pStrSrc )
```

String concatenation with length limitation.

Parameters

in	*pStrDst*	Destination string
in	*count*	Size of destination buffer
in	*pStrSrc*	Null terminated string to append

Return values

none	

5.13.3.13 vos_strncpy()

```
EXT_DECL void vos_strncpy (
            CHAR8 * pStrDst,
            const CHAR8 * pStrSrc,
            UINT32 count )
```

String copy with length limitation.

Parameters

in	*pStrDst*	Destination string
in	*pStrSrc*	Null terminated string to copy
in	*count*	Maximum number of characters to copy

Return values

none	

5.13.3.14 vos_strnicmp()

```
EXT_DECL INT32 vos_strnicmp (
            const CHAR8 * pStr1,
            const CHAR8 * pStr2,
            UINT32 count )
```

Case insensitive string compare.

Parameters

in	*pStr1*	Null terminated string to compare
in	*pStr2*	Null terminated string to compare
in	*count*	Maximum number of characters to compare

Return values

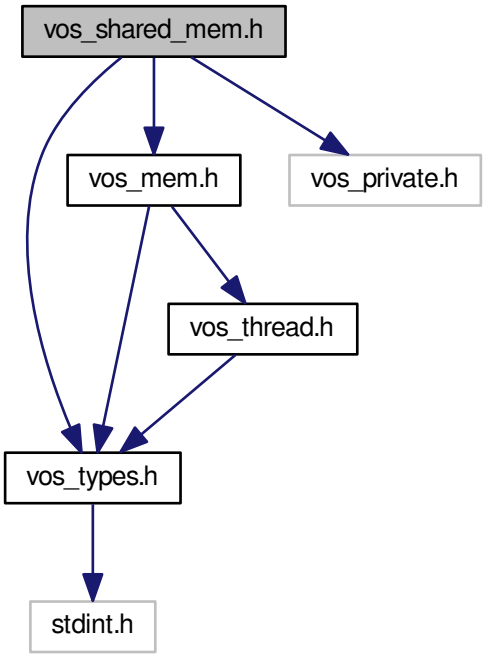
0	- equal
<0	- string1 less than string 2
>0	- string 1 greater than string 2

5.14 vos_shared_mem.h File Reference

Shared Memory functions for OS abstraction.

```
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_private.h"
```

Include dependency graph for vos_shared_mem.h:



Functions

- EXT_DECL VOS_ERR_T vos_sharedOpen (const CHAR8 *pKey, VOS_SHRD_T *pHandle, UINT8 **pp↩ MemoryArea, UINT32 *pSize)

  Create a shared memory area or attach to existing one.
- EXT_DECL VOS_ERR_T vos_sharedClose (VOS_SHRD_T handle, const UINT8 *pMemoryArea)

  Close connection to the shared memory area.

5.14.1 Detailed Description

Shared Memory functions for OS abstraction.

This module provides shared memory control supervison

Note

Project: TCNOpen TRDP prototype stack

Author

Kazumasa Aiba, TOSHIBA

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/. Copyright TOSHIBA, Japan, 2013.

Id

vos_mem.h 282 2013-01-11 07:08:44Z 97029

5.14.2 Function Documentation

5.14.2.1 vos_sharedClose()

```
EXT_DECL VOS_ERR_T vos_sharedClose (
            VOS_SHRD_T handle,
            const UINT8 * pMemoryArea )
```

Close connection to the shared memory area.

If the area was created by the calling process, the area will be closed (freed). If the area was attached, it will be detached. This function is not available in each target implementation.

Parameters

in	*handle*	Returned handle
in	*pMemoryArea*	Pointer to memory area

Return values

VOS_NO_ERR	no error
VOS_MEM_ERR	no memory available

5.14.2.2 vos_sharedOpen()

```
EXT_DECL VOS_ERR_T vos_sharedOpen (
            const CHAR8 * pKey,
            VOS_SHRD_T * pHandle,
            UINT8 ** ppMemoryArea,
            UINT32 * pSize )
```

Create a shared memory area or attach to existing one.

The first call with the a specified key will create a shared memory area with the supplied size and will return a handle and a pointer to that area. If the area already exists, the area will be opened. This function is not available in each target implementation.

Parameters

in	*pKey*	Unique identifier (file name)
out	*pHandle*	Pointer to returned handle
out	*ppMemoryArea*	Pointer to pointer to memory area
in,out	*pSize*	Pointer to size of area to allocate, on return actual size after attach

Return values

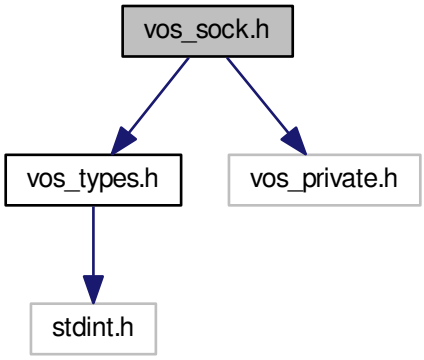
VOS_NO_ERR	no error
VOS_MEM_ERR	no memory available

5.15 vos_sock.h File Reference

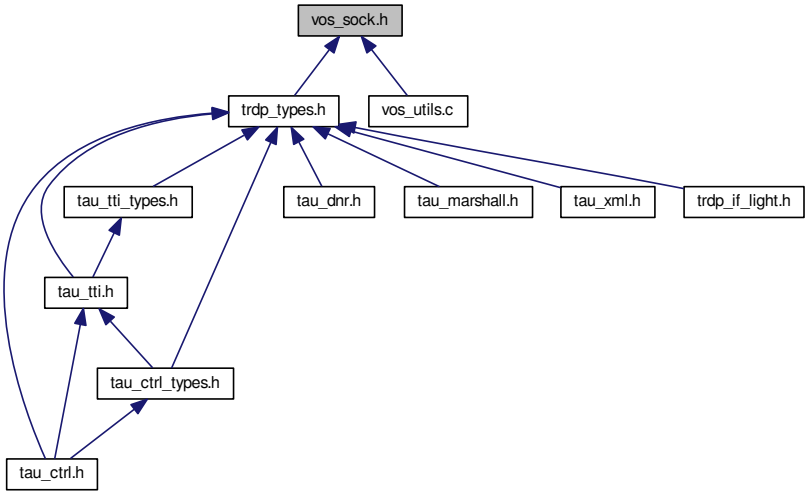
Typedefs for OS abstraction.

```
#include "vos_types.h"
#include "vos_private.h"
```

Include dependency graph for vos_sock.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct VOS_SOCK_OPT_T

  Common socket options.

Macros

- #define VOS_MAX_SOCKET_CNT 4

  The maximum number of sockets influences memory usage; for small systems we should define a smaller set.
- #define VOS_MAX_MULTICAST_CNT 5

  The maximum number of multicast groups one socket can join.
- #define VOS_TTL_MULTICAST 64

  The maximum number of hops a multicast packet can take.
- #define VOS_MAX_IF_NAME_SIZE 16

  The maximum number of IP interface adapters that can be handled by VOS.
- #define VOS_MAX_NUM_IF 8

  The maximum number of unicast addresses that can be handled by VOS.
- #define VOS_MAX_NUM_UNICAST 10

  The MAC size supported by VOS.
- #define VOS_MAC_SIZE 6

  Size of socket send and receive buffer.
- #define VOS_INVALID_SOCKET -1

  Invalid socket number.

Functions

- EXT_DECL UINT16 vos_htons (UINT16 val)

  Byte swapping 2 Bytes.
- EXT_DECL UINT16 vos_ntohs (UINT16 val)

  Byte swapping 2 Bytes.
- EXT_DECL UINT32 vos_htonl (UINT32 val)

  Byte swapping 4 Bytes.
- EXT_DECL UINT32 vos_ntohl (UINT32 val)

  Byte swapping 4 Bytes.
- EXT_DECL UINT32 vos_dottedIP (const CHAR8 *pDottedIP)

  Convert IP address from dotted dec.
- EXT_DECL const CHAR8 * vos_ipDotted (UINT32 ipAddress)

  Convert IP address to dotted dec.
- EXT_DECL BOOL8 vos_isMulticast (UINT32 ipAddress)

  Check if the supplied address is a multicast group address.
- EXT_DECL VOS_ERR_T vos_getInterfaces (UINT32 *pAddrCnt, VOS_IF_REC_T ifAddrs[ ])

  Get a list of interface addresses The caller has to provide an array of interface records to be filled.
- EXT_DECL BOOL8 vos_netIfUp (VOS_IP4_ADDR_T ifAddress)

  Get the state of an interface.
- EXT_DECL INT32 vos_select (INT32 highDesc, VOS_FDS_T *pReadableFD, VOS_FDS_T *pWriteableFD, VOS_FDS_T *pErrorFD, VOS_TIME_T *pTimeOut)

  select function.
- EXT_DECL VOS_ERR_T vos_sockInit (void)

  Initialize the socket library.
- EXT_DECL void vos_sockTerm (void)

De-Initialize the socket library.

- EXT_DECL VOS_ERR_T vos_sockGetMAC (UINT8 pMAC[VOS_MAC_SIZE])

  Return the MAC address of the default adapter.
- EXT_DECL VOS_ERR_T vos_sockOpenUDP (INT32 *pSock, const VOS_SOCK_OPT_T *pOptions)

  Create an UDP socket.
- EXT_DECL VOS_ERR_T vos_sockOpenTCP (INT32 *pSock, const VOS_SOCK_OPT_T *pOptions)

  Create a TCP socket.
- EXT_DECL VOS_ERR_T vos_sockClose (INT32 sock)

  Close a socket.
- EXT_DECL VOS_ERR_T vos_sockSetOptions (INT32 sock, const VOS_SOCK_OPT_T *pOptions)

  Set socket options.
- EXT_DECL VOS_ERR_T vos_sockJoinMC (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

  Join a multicast group.
- EXT_DECL VOS_ERR_T vos_sockLeaveMC (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

  Leave a multicast group.
- EXT_DECL VOS_ERR_T vos_sockSendUDP (INT32 sock, const UINT8 *pBuffer, UINT32 *pSize, UINT32 ipAddress, UINT16 port)

  Send UDP data.
- EXT_DECL VOS_ERR_T vos_sockReceiveUDP (INT32 sock, UINT8 *pBuffer, UINT32 *pSize, UINT32 *p↩ SrcIPAddr, UINT16 *pSrcIPPort, UINT32 *pDstIPAddr, BOOL8 peek)

  Receive UDP data.
- EXT_DECL VOS_ERR_T vos_sockBind (INT32 sock, UINT32 ipAddress, UINT16 port)

  Bind a socket to an address and port.
- EXT_DECL VOS_ERR_T vos_sockListen (INT32 sock, UINT32 backlog)

  Listen for incoming TCP connections.
- EXT_DECL VOS_ERR_T vos_sockAccept (INT32 sock, INT32 *pSock, UINT32 *pIPAddress, UINT16 *p↩ Port)

  Accept an incoming TCP connection.
- EXT_DECL VOS_ERR_T vos_sockConnect (INT32 sock, UINT32 ipAddress, UINT16 port)

  Open a TCP connection.
- EXT_DECL VOS_ERR_T vos_sockSendTCP (INT32 sock, const UINT8 *pBuffer, UINT32 *pSize)

  Send TCP data.
- EXT_DECL VOS_ERR_T vos_sockReceiveTCP (INT32 sock, UINT8 *pBuffer, UINT32 *pSize)

  Receive TCP data.
- EXT_DECL VOS_ERR_T vos_sockSetMulticastIf (INT32 sock, UINT32 mcIfAddress)

  Set Using Multicast I/F.
- EXT_DECL VOS_IP4_ADDR_T vos_determineBindAddr (VOS_IP4_ADDR_T srcIP, VOS_IP4_ADDR_↩ T mcGroup, VOS_IP4_ADDR_T rcvMostly)

  Determines the address to bind to since the behaviour in the different OS is different.

5.15.1 Detailed Description

Typedefs for OS abstraction.

This is the declaration for the OS independend socket interface

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

Id

5.15.2 Macro Definition Documentation

5.15.2.1 VOS_MAX_SOCKET_CNT

```
#define VOS_MAX_SOCKET_CNT 4
```

The maximum number of sockets influences memory usage; for small systems we should define a smaller set.

The maximum number of concurrent usable sockets per application session

5.15.2.2 VOS_TTL_MULTICAST

```
#define VOS_TTL_MULTICAST 64
```

The maximum number of hops a multicast packet can take.

The maximum size for the interface name

5.15.3 Function Documentation

5.15.3.1 vos_determineBindAddr()

```
EXT_DECL VOS_IP4_ADDR_T vos_determineBindAddr (
            VOS_IP4_ADDR_T srcIP,
            VOS_IP4_ADDR_T mcGroup,
            VOS_IP4_ADDR_T rcvMostly )
```

Determines the address to bind to since the behaviour in the different OS is different.

Parameters

in	*srcIP*	IP to bind to (0 = any address)
in	*mcGroup*	MC group to join (0 = do not join)
in	*rcvMostly*	primarily used for receiving (tbd: bind on sender, too?)

Return values

Address	to bind to

5.15.3.2 vos_dottedIP()

```
EXT_DECL UINT32 vos_dottedIP (
            const CHAR8 * pDottedIP )
```

Convert IP address from dotted dec.

to !host! endianess

Parameters

in	*p↩ DottedIP*	IP address as dotted decimal.

Return values

address	in UINT32 in host endianess

5.15.3.3 vos_getInterfaces()

```
EXT_DECL VOS_ERR_T vos_getInterfaces (
            UINT32 * pAddrCnt,
            VOS_IF_REC_T ifAddrs[] )
```

Get a list of interface addresses The caller has to provide an array of interface records to be filled.

Parameters

in,out	*pAddrCnt*	in: pointer to array size of interface record out: pointer to number of interface records read
in,out	*ifAddrs*	array of interface records

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	pAddrCnt and/or ifAddrs == NULL
VOS_MEM_ERR	memory allocation error
VOS_SOCK_ERR	GetAdaptersInfo() error

5.15.3.4 vos_htonl()

```
EXT_DECL UINT32 vos_htonl (
            UINT32 val )
```

Byte swapping 4 Bytes.

Parameters

in	*val*	Initial value.

Return values

swapped	value

5.15.3.5 vos_htons()

```
EXT_DECL UINT16 vos_htons (
            UINT16 val )
```

Byte swapping 2 Bytes.

Parameters

in	*val*	Initial value.

Return values

swapped	value

5.15.3.6 vos_ipDotted()

```
EXT_DECL const CHAR8* vos_ipDotted (
            UINT32 ipAddress )
```

Convert IP address to dotted dec.

from !host! endianess

Parameters

in	*ipAddress*	address in UINT32 in host endianess

Return values

IP	address as dotted decimal.

5.15.3.7 vos_isMulticast()

```
EXT_DECL BOOL8 vos_isMulticast (
            UINT32 ipAddress )
```

Check if the supplied address is a multicast group address.

Parameters

in	*ipAddress*	IP address to check.

Return values

TRUE	address is a multicast address
FALSE	address is not a multicast address

5.15.3.8 vos_netIfUp()

```
EXT_DECL BOOL8 vos_netIfUp (
            VOS_IP4_ADDR_T ifAddress )
```

Get the state of an interface.

Parameters

in	*ifAddress*	address of interface to check

Return values

TRUE	interface is up and ready FALSE interface is down / not ready

5.15.3.9 vos_ntohl()

```
EXT_DECL UINT32 vos_ntohl (
            UINT32 val )
```

Byte swapping 4 Bytes.

Parameters

in	*val*	Initial value.

Return values

swapped	value

5.15.3.10 vos_ntohs()

```
EXT_DECL UINT16 vos_ntohs (
            UINT16 val )
```

Byte swapping 2 Bytes.

Parameters

in	*val*	Initial value.

Return values

swapped	value

5.15.3.11 vos_select()

```
EXT_DECL INT32 vos_select (
            INT32 highDesc,
            VOS_FDS_T * pReadableFD,
            VOS_FDS_T * pWriteableFD,
            VOS_FDS_T * pErrorFD,
            VOS_TIME_T * pTimeOut )
```

select function.

Set the ready sockets in the supplied sets. Note: Some target systems might define this function as NOP.

Parameters

in	*highDesc*	max. socket descriptor + 1
in,out	*pReadableFD*	pointer to readable socket set
in,out	*pWriteableFD*	pointer to writeable socket set
in,out	*pErrorFD*	pointer to error socket set
in	*pTimeOut*	pointer to time out value

Return values

number	of ready file descriptors

5.15.3.12 vos_sockAccept()

```
EXT_DECL VOS_ERR_T vos_sockAccept (
            INT32 sock,
            INT32 * pSock,
            UINT32 * pIPAddress,
            UINT16 * pPort )
```

Accept an incoming TCP connection.

Accept incoming connections on the provided socket. May block and will return a new socket descriptor when accepting a connection. The original socket ∗pSock, remains open.

Parameters

in	*sock*	Socket descriptor

Parameters

out	*pSock*	Pointer to socket descriptor, on exit new socket
out	*pIPAddress*	source IP to receive on, 0 for any
out	*pPort*	port to receive on, 17224 for PD

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	NULL parameter, parameter error
VOS_UNKNOWN_ERR	sock descriptor unknown error

5.15.3.13 vos_sockBind()

```
EXT_DECL VOS_ERR_T vos_sockBind (
            INT32 sock,
            UINT32 ipAddress,
            UINT16 port )
```

Bind a socket to an address and port.

Parameters

in	*sock*	socket descriptor
in	*ipAddress*	source IP to receive from, 0 for any
in	*port*	port to receive from

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_IO_ERR	Input/Output error
VOS_MEM_ERR	resource error

5.15.3.14 vos_sockClose()

```
EXT_DECL VOS_ERR_T vos_sockClose (
            INT32 sock )
```

Close a socket.

Release any resources aquired by this socket

Parameters

in	*sock*	socket descriptor

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	pSock == NULL

5.15.3.15 vos_sockConnect()

```
EXT_DECL VOS_ERR_T vos_sockConnect (
            INT32 sock,
            UINT32 ipAddress,
            UINT16 port )
```

Open a TCP connection.

Parameters

in	*sock*	socket descriptor
in	*ipAddress*	destination IP
in	*port*	destination port

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_IO_ERR	Input/Output error

5.15.3.16 vos_sockGetMAC()

```
EXT_DECL VOS_ERR_T vos_sockGetMAC (
            UINT8 pMAC[VOS_MAC_SIZE] )
```

Return the MAC address of the default adapter.

Parameters

out	*pMAC*	return MAC address.

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	pMAC == NULL
VOS_SOCK_ERR	socket not available or option not supported

5.15.3.17 vos_sockInit()

```
EXT_DECL VOS_ERR_T vos_sockInit (
```

```
            void  )
```

Initialize the socket library.

Must be called once before any other call

Return values

VOS_NO_ERR	no error
VOS_SOCK_ERR	sockets not supported

5.15.3.18 vos_sockJoinMC()

```
EXT_DECL VOS_ERR_T vos_sockJoinMC (
            INT32 sock,
            UINT32 mcAddress,
            UINT32 ipAddress )
```

Join a multicast group.

Note: Some target systems might not support this option.

Parameters

in	*sock*	socket descriptor
in	*mcAddress*	multicast group to join
in	*ipAddress*	depicts interface on which to join, default 0 for any

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_SOCK_ERR	option not supported

5.15.3.19 vos_sockLeaveMC()

```
EXT_DECL VOS_ERR_T vos_sockLeaveMC (
            INT32 sock,
            UINT32 mcAddress,
            UINT32 ipAddress )
```

Leave a multicast group.

Note: Some target systems might not support this option.

Parameters

in	*sock*	socket descriptor
in	*mcAddress*	multicast group to join
in	*ipAddress*	depicts interface on which to leave, default 0 for any

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_SOCK_ERR	option not supported

5.15.3.20 vos_sockListen()

```
EXT_DECL VOS_ERR_T vos_sockListen (
            INT32 sock,
            UINT32 backlog )
```

Listen for incoming TCP connections.

Parameters

in	*sock*	socket descriptor
in	*backlog*	maximum connection attempts if system is busy

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_IO_ERR	Input/Output error
VOS_MEM_ERR	resource error

5.15.3.21 vos_sockOpenTCP()

```
EXT_DECL VOS_ERR_T vos_sockOpenTCP (
            INT32 * pSock,
            const VOS_SOCK_OPT_T * pOptions )
```

Create a TCP socket.

Return a socket descriptor for further calls. The socket options are optional and can be applied later.

Parameters

out	*pSock*	pointer to socket descriptor returned
in	*pOptions*	pointer to socket options (optional)

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	pSock == NULL
VOS_SOCK_ERR	socket not available or option not supported

5.15.3.22 vos_sockOpenUDP()

```
EXT_DECL VOS_ERR_T vos_sockOpenUDP (
            INT32 * pSock,
            const VOS_SOCK_OPT_T * pOptions )
```

Create an UDP socket.

Return a socket descriptor for further calls. The socket options are optional and can be applied later. Note: Some target systems might not support every option.

Parameters

out	*pSock*	pointer to socket descriptor returned
in	*pOptions*	pointer to socket options (optional)

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	pSock == NULL
VOS_SOCK_ERR	socket not available or option not supported

5.15.3.23 vos_sockReceiveTCP()

```
EXT_DECL VOS_ERR_T vos_sockReceiveTCP (
            INT32 sock,
            UINT8 * pBuffer,
            UINT32 * pSize )
```

Receive TCP data.

The caller must provide a sufficient sized buffer. If the supplied buffer is smaller than the bytes received, ∗pSize will reflect the number of copied bytes and the call should be repeated until ∗pSize is 0 (zero). If the socket was created in blocking-mode (default), then this call will block and will only return if data has been received or the socket was closed or an error occured. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned.

Parameters

in	*sock*	socket descriptor
out	*pBuffer*	pointer to applications data buffer
in,out	*pSize*	pointer to the received data size

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	sock descriptor unknown, parameter error
VOS_IO_ERR	data could not be read
VOS_NODATA_ERR	no data in non-blocking
VOS_BLOCK_ERR	call would have blocked in blocking mode

5.15.3.24 vos_sockReceiveUDP()

```
EXT_DECL VOS_ERR_T vos_sockReceiveUDP (
            INT32 sock,
            UINT8 * pBuffer,
            UINT32 * pSize,
            UINT32 * pSrcIPAddr,
            UINT16 * pSrcIPPort,
            UINT32 * pDstIPAddr,
            BOOL8 peek )
```

Receive UDP data.

The caller must provide a sufficient sized buffer. If the supplied buffer is smaller than the bytes received, ∗pSize will reflect the number of copied bytes and the call should be repeated until ∗pSize is 0 (zero). If the socket was created in blocking-mode (default), then this call will block and will only return if data has been received or the socket was closed or an error occured. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned. If pointers are provided, source IP, source port and destination IP will be reported on return.

Parameters

in	*sock*	socket descriptor
out	*pBuffer*	pointer to applications data buffer
in,out	*pSize*	pointer to the received data size
out	*pSrcIPAddr*	pointer to source IP
out	*pSrcIPPort*	pointer to source port
out	*pDstIPAddr*	pointer to dest IP
in	*peek*	if true, leave data in queue

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	sock descriptor unknown, parameter error
VOS_IO_ERR	data could not be read
VOS_NODATA_ERR	no data
VOS_BLOCK_ERR	Call would have blocked in blocking mode

5.15.3.25 vos_sockSendTCP()

```
EXT_DECL VOS_ERR_T vos_sockSendTCP (
            INT32 sock,
            const UINT8 * pBuffer,
            UINT32 * pSize )
```

Send TCP data.

Send data to the supplied address and port.

Parameters

in	*sock*	socket descriptor
in	*pBuffer*	pointer to data to send
in,out	*pSize*	In: size of the data to send, Out: no of bytes sent

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	sock descriptor unknown, parameter error
VOS_IO_ERR	data could not be sent
VOS_NOCONN_ERR	no TCP connection
VOS_BLOCK_ERR	call would have blocked in blocking mode, data partially sent

5.15.3.26 vos_sockSendUDP()

```
EXT_DECL VOS_ERR_T vos_sockSendUDP (
            INT32 sock,
            const UINT8 * pBuffer,
            UINT32 * pSize,
            UINT32 ipAddress,
            UINT16 port )
```

Send UDP data.

Send data to the given address and port.

Parameters

in	*sock*	socket descriptor
in	*pBuffer*	pointer to data to send
in,out	*pSize*	In: size of the data to send, Out: no of bytes sent
in	*ipAddress*	destination IP
in	*port*	destination port

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid
VOS_IO_ERR	data could not be sent
VOS_BLOCK_ERR	Call would have blocked in blocking mode

5.15.3.27 vos_sockSetMulticastIf()

```
EXT_DECL VOS_ERR_T vos_sockSetMulticastIf (
            INT32 sock,
            UINT32 mcIfAddress )
```

Set Using Multicast I/F.

Parameters

in	*sock*	socket descriptor
in	*mcIfAddress*	using Multicast I/F Address

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	sock descriptor unknown, parameter error

5.15.3.28 vos_sockSetOptions()

```
EXT_DECL VOS_ERR_T vos_sockSetOptions (
            INT32 sock,
            const VOS_SOCK_OPT_T * pOptions )
```

Set socket options.

Note: Some target systems might not support each option.

Parameters

in	*sock*	socket descriptor
in	*pOptions*	pointer to socket options (optional)

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	parameter out of range/invalid

5.15.3.29 vos_sockTerm()

```
EXT_DECL void vos_sockTerm (
            void  )
```

De-Initialize the socket library.

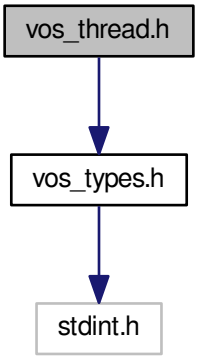
Must be called after last socket call

5.16 vos_thread.h File Reference

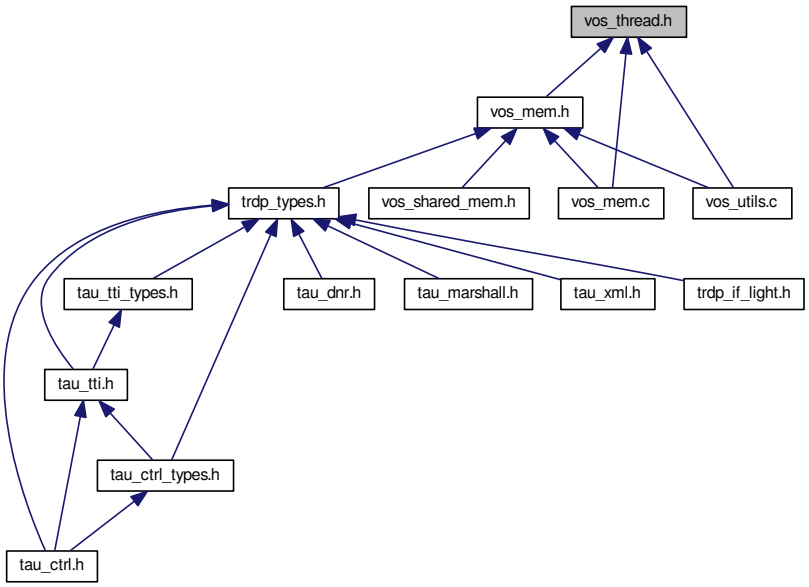
Threading functions for OS abstraction.

```
#include "vos_types.h"
```

Include dependency graph for vos_thread.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define VOS_MAX_THREAD_CNT 100

  The maximum number of concurrent usable threads.

- #define VOS_SEMA_WAIT_FOREVER 0xFFFFFFFFU

  Timeout value to wait forever for a semaphore.

Typedefs

- typedef UINT8 VOS_THREAD_PRIORITY_T

  Thread priority range from 1 (highest) to 255 (lowest), 0 default of the target system.
- typedef void(__cdecl * VOS_THREAD_FUNC_T) (void *pArg)

  Thread function definition.
- typedef struct VOS_MUTEX * VOS_MUTEX_T

  Hidden mutex handle definition.
- typedef struct VOS_SEMA * VOS_SEMA_T

  Hidden semaphore handle definition.
- typedef void * VOS_THREAD_T

  Hidden thread handle definition.

Enumerations

- enum VOS_THREAD_POLICY_T

  Thread policy matching pthread/Posix defines.
- enum VOS_SEMA_STATE_T

  State of the semaphore.

Functions

- EXT_DECL VOS_ERR_T vos_threadInit (void)

  Initialize the thread library.
- EXT_DECL void vos_threadTerm (void)

  De-Initialize the thread library.
- EXT_DECL VOS_ERR_T vos_threadCreate (VOS_THREAD_T *pThread, const CHAR8 *pName, VOS_THREAD_POLICY_T policy, VOS_THREAD_PRIORITY_T priority, UINT32 interval, UINT32 stackSize, VOS_THREAD_FUNC_T pFunction, void *pArguments)

  Create a thread.
- EXT_DECL void vos_cyclicThread (UINT32 interval, VOS_THREAD_FUNC_T pFunction, void *pArguments)

  Cyclic thread functions.
- EXT_DECL VOS_ERR_T vos_threadTerminate (VOS_THREAD_T thread)

  Terminate a thread.
- EXT_DECL VOS_ERR_T vos_threadIsActive (VOS_THREAD_T thread)

  Is the thread still active? This call will return VOS_NO_ERR if the thread is still active, VOS_PARAM_ERR in case it ran out.
- EXT_DECL VOS_ERR_T vos_threadDelay (UINT32 delay)

  Delay the execution of the current thread by the given delay in us.
- EXT_DECL void vos_getTime (VOS_TIME_T *pTime)

  Return the current time in sec and us.
- EXT_DECL const CHAR8 * vos_getTimeStamp (void)

  Get a time-stamp string.
- EXT_DECL void vos_clearTime (VOS_TIME_T *pTime)

  Clear the time stamp.
- EXT_DECL void vos_addTime (VOS_TIME_T *pTime, const VOS_TIME_T *pAdd)

  Add the second to the first time stamp, return sum in first.
- EXT_DECL void vos_subTime (VOS_TIME_T *pTime, const VOS_TIME_T *pSub)

  Subtract the second from the first time stamp, return diff in first.
- EXT_DECL INT32 vos_cmpTime (const VOS_TIME_T *pTime, const VOS_TIME_T *pCmp)

Compare the second from the first time stamp, return diff in first.

- EXT_DECL void vos_divTime (VOS_TIME_T *pTime, UINT32 divisor)

  Divide the first time by the second, return quotient in first.
- EXT_DECL void vos_mulTime (VOS_TIME_T *pTime, UINT32 mul)

  Multiply the first time by the second, return product in first.
- EXT_DECL void vos_getUuid (VOS_UUID_T pUuID)

  Get a universal unique identifier according to RFC 4122 time based version.
- EXT_DECL VOS_ERR_T vos_mutexCreate (VOS_MUTEX_T *pMutex)

  Create a mutex.
- EXT_DECL void vos_mutexDelete (VOS_MUTEX_T pMutex)

  Delete a mutex.
- EXT_DECL VOS_ERR_T vos_mutexLock (VOS_MUTEX_T pMutex)

  Take a mutex.
- EXT_DECL VOS_ERR_T vos_mutexTryLock (VOS_MUTEX_T pMutex)

  Try to take a mutex.
- EXT_DECL VOS_ERR_T vos_mutexUnlock (VOS_MUTEX_T pMutex)

  Release a mutex.
- EXT_DECL VOS_ERR_T vos_semaCreate (VOS_SEMA_T *pSema, VOS_SEMA_STATE_T initialState)

  Create a semaphore.
- EXT_DECL void vos_semaDelete (VOS_SEMA_T sema)

  Delete a semaphore.
- EXT_DECL VOS_ERR_T vos_semaTake (VOS_SEMA_T sema, UINT32 timeout)

  Take a semaphore.
- EXT_DECL void vos_semaGive (VOS_SEMA_T sema)

  Give a semaphore.

5.16.1 Detailed Description

Threading functions for OS abstraction.

Thread-, semaphore- and time-handling functions

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/. Copyright Bombardier Transportation Inc. or its subsidiaries and others, 2014. All rights reserved.

Id

5.16.2 Function Documentation

5.16.2.1 vos_addTime()

```
EXT_DECL void vos_addTime (
            VOS_TIME_T * pTime,
            const VOS_TIME_T * pAdd )
```

Add the second to the first time stamp, return sum in first.

Parameters

in,out	*pTime*	Pointer to time value
in	*pAdd*	Pointer to time value

5.16.2.2 vos_clearTime()

```
EXT_DECL void vos_clearTime (
            VOS_TIME_T * pTime )
```

Clear the time stamp.

Parameters

out	*pTime*	Pointer to time value

5.16.2.3 vos_cmpTime()

```
EXT_DECL INT32 vos_cmpTime (
            const VOS_TIME_T * pTime,
            const VOS_TIME_T * pCmp )
```

Compare the second from the first time stamp, return diff in first.

Parameters

in,out	*pTime*	Pointer to time value
in	*pCmp*	Pointer to time value to compare

Return values

0	pTime == pCmp
-1	pTime < pCmp
1	pTime > pCmp

5.16.2.4 vos_cyclicThread()

```
EXT_DECL void vos_cyclicThread (
            UINT32 interval,
            VOS_THREAD_FUNC_T pFunction,
            void * pArguments )
```

Cyclic thread functions.

Wrapper for cyclic threads. The thread function will be called cyclically with interval.

Parameters

in	*interval*	Interval for cyclic threads in us (incl. runtime)
in	*pFunction*	Pointer to the thread function
in	*pArguments*	Pointer to the thread function parameters

Return values

void	

5.16.2.5 vos_divTime()

```
EXT_DECL void vos_divTime (
            VOS_TIME_T * pTime,
            UINT32 divisor )
```

Divide the first time by the second, return quotient in first.

Parameters

in,out	*pTime*	Pointer to time value
in	*divisor*	Divisor

5.16.2.6 vos_getTime()

```
EXT_DECL void vos_getTime (
            VOS_TIME_T * pTime )
```

Return the current time in sec and us.

Parameters

out	*pTime*	Pointer to time value

5.16.2.7 vos_getTimeStamp()

```
EXT_DECL const CHAR8* vos_getTimeStamp (
            void  )
```

Get a time-stamp string.

Get a time-stamp string for debugging in the form "yyyymmdd-hh:mm:ss.ms" Depending on the used OS / hardware the time might not be a real-time stamp but relative from start of system.

Return values

timestamp	"yyyymmdd-hh:mm:ss.ms"

5.16.2.8 vos_getUuid()

```
EXT_DECL void vos_getUuid (
            VOS_UUID_T pUuID )
```

Get a universal unique identifier according to RFC 4122 time based version.

Parameters

out	*pUuID*	Pointer to a universal unique identifier

5.16.2.9 vos_mulTime()

```
EXT_DECL void vos_mulTime (
            VOS_TIME_T * pTime,
            UINT32 mul )
```

Multiply the first time by the second, return product in first.

Parameters

in,out	*pTime*	Pointer to time value
in	*mul*	Factor

5.16.2.10 vos_mutexCreate()

```
EXT_DECL VOS_ERR_T vos_mutexCreate (
            VOS_MUTEX_T * pMutex )
```

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

Parameters

out	*pMutex*	Pointer to mutex handle

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_PARAM_ERR	pMutex == NULL
VOS_MUTEX_ERR	no mutex available

5.16.2.11 vos_mutexDelete()

```
EXT_DECL void vos_mutexDelete (
```

```
            VOS_MUTEX_T pMutex )
```

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	*pMutex*	mutex handle

Return values

VOS_NO_ERR	no error

5.16.2.12 vos_mutexLock()

```
EXT_DECL VOS_ERR_T vos_mutexLock (
            VOS_MUTEX_T pMutex )
```

Take a mutex.

Wait for the mutex to become available (lock).

Parameters

in	*pMutex*	mutex handle

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle

5.16.2.13 vos_mutexTryLock()

```
EXT_DECL VOS_ERR_T vos_mutexTryLock (
            VOS_MUTEX_T pMutex )
```

Try to take a mutex.

If mutex is can't be taken VOS_MUTEX_ERR is returned.

Parameters

in	*pMutex*	mutex handle

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_MUTEX_ERR	no mutex available

5.16.2.14 vos_mutexUnlock()

```
EXT_DECL VOS_ERR_T vos_mutexUnlock (
            VOS_MUTEX_T pMutex )
```

Release a mutex.

Unlock the mutex.

Parameters

in	*pMutex*	mutex handle

5.16.2.15 vos_semaCreate()

```
EXT_DECL VOS_ERR_T vos_semaCreate (
            VOS_SEMA_T * pSema,
            VOS_SEMA_STATE_T initialState )
```

Create a semaphore.

Return a semaphore handle. Depending on the initial state the semaphore will be available on creation or not.

Parameters

out	*pSema*	Pointer to semaphore handle
in	*initialState*	The initial state of the sempahore

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_PARAM_ERR	parameter out of range/invalid
VOS_SEMA_ERR	no semaphore available

5.16.2.16 vos_semaDelete()

```
EXT_DECL void vos_semaDelete (
            VOS_SEMA_T sema )
```

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

Parameters

in	*sema*	semaphore handle

5.16.2.17 vos_semaGive()

```
EXT_DECL void vos_semaGive (
            VOS_SEMA_T sema )
```

Give a semaphore.

Release (increase) a semaphore.

Parameters

in	*sema*	semaphore handle

5.16.2.18 vos_semaTake()

```
EXT_DECL VOS_ERR_T vos_semaTake (
            VOS_SEMA_T sema,
            UINT32 timeout )
```

Take a semaphore.

Try to get (decrease) a semaphore.

Parameters

in	*sema*	semaphore handle
in	*timeout*	Max. time in us to wait, 0 means no wait

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid
VOS_SEMA_ERR	could not get semaphore in time

5.16.2.19 vos_subTime()

```
EXT_DECL void vos_subTime (
```

```
            VOS_TIME_T * pTime,
            const VOS_TIME_T * pSub )
```

Subtract the second from the first time stamp, return diff in first.

Parameters

in,out	*pTime*	Pointer to time value
in	*pSub*	Pointer to time value

5.16.2.20 vos_threadCreate()

```
EXT_DECL VOS_ERR_T vos_threadCreate (
            VOS_THREAD_T * pThread,
            const CHAR8 * pName,
            VOS_THREAD_POLICY_T policy,
            VOS_THREAD_PRIORITY_T priority,
            UINT32 interval,
            UINT32 stackSize,
            VOS_THREAD_FUNC_T pFunction,
            void * pArguments )
```

Create a thread.

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

Parameters

out	*pThread*	Pointer to returned thread handle
in	*pName*	Pointer to name of the thread (optional)
in	*policy*	Scheduling policy (FIFO, Round Robin or other)
in	*priority*	Scheduling priority (1...255 (highest), default 0)
in	*interval*	Interval for cyclic threads in us (optional)
in	*stackSize*	Minimum stacksize, default 0: 16kB
in	*pFunction*	Pointer to the thread function
in	*pArguments*	Pointer to the thread function parameters

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid

5.16.2.21 vos_threadDelay()

```
EXT_DECL VOS_ERR_T vos_threadDelay (
            UINT32 delay )
```

Delay the execution of the current thread by the given delay in us.

Parameters

in	*delay*	Delay in us

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised

5.16.2.22 vos_threadInit()

```
EXT_DECL VOS_ERR_T vos_threadInit (
            void  )
```

Initialize the thread library.

Must be called once before any other call

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	threading not supported

5.16.2.23 vos_threadIsActive()

```
EXT_DECL VOS_ERR_T vos_threadIsActive (
            VOS_THREAD_T thread )
```

Is the thread still active? This call will return VOS_NO_ERR if the thread is still active, VOS_PARAM_ERR in case it ran out.

Parameters

in	*thread*	Thread handle

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid

5.16.2.24 vos_threadTerm()

```
EXT_DECL void vos_threadTerm (
```

```
            void  )
```

De-Initialize the thread library.

Must be called after last thread/timer call

5.16.2.25 vos_threadTerminate()

```
EXT_DECL VOS_ERR_T vos_threadTerminate (
            VOS_THREAD_T thread )
```

Terminate a thread.

This call will terminate the thread with the given threadId and release all resources. Depending on the underlying architectures, it may just block until the thread ran out.

Parameters

in	*thread*	Thread handle (or NULL if current thread)

Return values

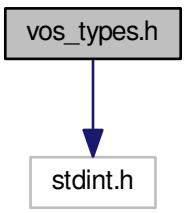
VOS_NO_ERR	no error
VOS_INIT_ERR	module not initialised
VOS_NOINIT_ERR	invalid handle
VOS_PARAM_ERR	parameter out of range/invalid

5.17 vos_types.h File Reference

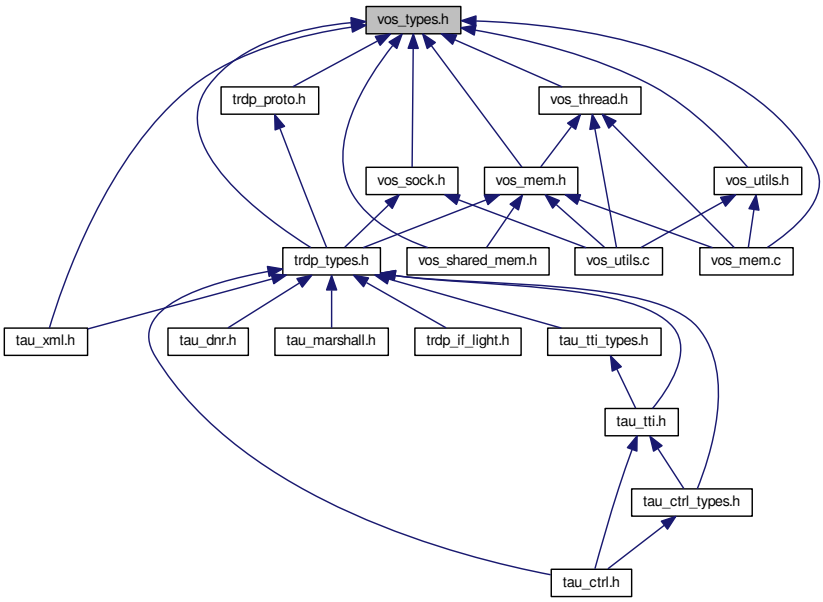
Typedefs for OS abstraction.

```
#include <stdint.h>
```

Include dependency graph for vos_types.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct VOS_VERSION_T

  Version information.
- struct VOS_TIME_T

  Timer value compatible with timeval / select.

Macros

- #define INLINE inline

  inline macros
- #define AV_ERROR 0x00

  ANTIVALENT8 values.
- #define TR_DIR1 0x01

  Directions/Orientations.

Typedefs

- typedef UINT8 VOS_UUID_T[16]

  universal unique identifier according to RFC 4122, time based version
- typedef void(∗ VOS_PRINT_DBG_T) (void ∗pRefCon, VOS_LOG_T category, const CHAR8 ∗pTime, const CHAR8 ∗pFile, UINT16 LineNumber, const CHAR8 ∗pMsgStr)

  Function definition for error/debug output.

Enumerations

- enum VOS_ERR_T {
  VOS_NO_ERR = 0,
  VOS_PARAM_ERR = -1,
  VOS_INIT_ERR = -2,
  VOS_NOINIT_ERR = -3,
  VOS_TIMEOUT_ERR = -4,
  VOS_NODATA_ERR = -5,
  VOS_SOCK_ERR = -6,
  VOS_IO_ERR = -7,
  VOS_MEM_ERR = -8,
  VOS_SEMA_ERR = -9,
  VOS_QUEUE_ERR = -10,
  VOS_QUEUE_FULL_ERR = -11,
  VOS_MUTEX_ERR = -12,
  VOS_THREAD_ERR = -13,
  VOS_BLOCK_ERR = -14,
  VOS_INTEGRATION_ERR = -15,
  VOS_NOCONN_ERR = -16,
  VOS_UNKNOWN_ERR = -99 }

  Return codes for all VOS API functions.
- enum VOS_LOG_T {
  VOS_LOG_ERROR = 0,
  VOS_LOG_WARNING = 1,
  VOS_LOG_INFO = 2,
  VOS_LOG_DBG = 3 }

  Categories for logging.

5.17.1 Detailed Description

Typedefs for OS abstraction.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

Id

```
BL 2016-07-06: Ticket #122 64Bit compatibility (+ compiler warnings)
```

5.17.2 Typedef Documentation

5.17.2.1 VOS_PRINT_DBG_T

```
typedef void(* VOS_PRINT_DBG_T) (void *pRefCon, VOS_LOG_T category, const CHAR8 *pTime, const
CHAR8 *pFile, UINT16 LineNumber, const CHAR8 *pMsgStr)
```

Function definition for error/debug output.

The function will be called for logging and error message output. The user can decide, what kind of info will be logged by filtering the category.

Parameters

in	*pRefCon	pointer to user context
in	*category*	Log category (Error, Warning, Info etc.)
in	*pTime*	pointer to NULL-terminated string of time stamp
in	*pFile*	pointer to NULL-terminated string of source module
in	*LineNumber*	Line number
in	*pMsgStr*	pointer to NULL-terminated string

Return values

none	

5.17.3 Enumeration Type Documentation

5.17.3.1 VOS_ERR_T

`enum VOS_ERR_T`

Return codes for all VOS API functions.

Enumerator

VOS_NO_ERR	No error.
VOS_PARAM_ERR	Necessary parameter missing or out of range.
VOS_INIT_ERR	Call without valid initialization.
VOS_NOINIT_ERR	The supplied handle/reference is not valid.
VOS_TIMEOUT_ERR	Timout.
VOS_NODATA_ERR	Non blocking mode: no data received.
VOS_SOCK_ERR	Socket option not supported.
VOS_IO_ERR	Socket IO error, data can't be received/sent.
VOS_MEM_ERR	No more memory available.
VOS_SEMA_ERR	Semaphore not available.
VOS_QUEUE_ERR	Queue empty.
VOS_QUEUE_FULL_ERR	Queue full.
VOS_MUTEX_ERR	Mutex not available.
VOS_THREAD_ERR	Thread creation error.
VOS_BLOCK_ERR	System call would have blocked in blocking mode.
VOS_INTEGRATION_ERR	Alignment or endianess for selected target wrong.
VOS_NOCONN_ERR	No TCP connection.
VOS_UNKNOWN_ERR	Unknown error.

5.17.3.2 VOS_LOG_T

`enum VOS_LOG_T`

Categories for logging.

Enumerator

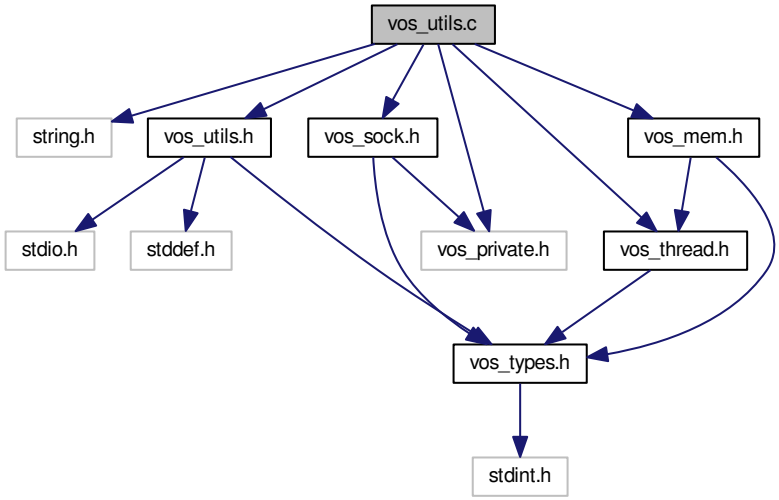
VOS_LOG_ERROR	This is a critical error.
VOS_LOG_WARNING	This is a warning.
VOS_LOG_INFO	This is an info.
VOS_LOG_DBG	This is a debug info.

5.18 vos_utils.c File Reference

Common functions for VOS.

```
#include <string.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "vos_mem.h"
#include "vos_private.h"
```

Include dependency graph for vos_utils.c:



Functions

- VOS_ERR_T vos_initRuntimeConsts (void)

  Pre-compute alignment and endianess.
- VOS_ERR_T vos_init (void *pRefCon, VOS_PRINT_DBG_T pDebugOutput)

  Initialize the virtual operating system.
- EXT_DECL void vos_terminate ()

  DeInitialize the vos library.
- UINT32 vos_crc32 (UINT32 crc, const UINT8 *pData, UINT32 dataLen)

  Compute crc32 according to IEEE802.3.

- UINT32 vos_sc32 (UINT32 crc, const UINT8 *pData, UINT32 dataLen)

  Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.
- const char * vos_getVersionString (void)

  Return a human readable version representation.
- EXT_DECL const VOS_VERSION_T * vos_getVersion (void)

  Return version.

5.18.1 Detailed Description

Common functions for VOS.

Common functions of the abstraction layer. Mainly debugging support.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

Id

```
BL 2016-08-17: parentheses added (compiler warning)
BL 2016-07-06: Ticket #122 64Bit compatibility (+ compiler warnings)
BL 2016-03-10: Ticket #114 SC-32
BL 2016-02-10: ifdef DEBUG for some functions
BL 2014-02-28: Ticket #25: CRC32 calculation is not according IEEE802.3
```

5.18.2 Function Documentation

5.18.2.1 vos_crc32()

```
UINT32 vos_crc32 (
            UINT32 crc,
            const UINT8 * pData,
            UINT32 dataLen )
```

Compute crc32 according to IEEE802.3.

Calculate CRC for the given buffer and length.

/ to IEC 61375-2-3 A.3 Note: Returned CRC is inverted

Parameters

in	*crc*	Initial value.
in,out	*pData*	Pointer to data.
in	*dataLen*	length in bytes of data.

Return values

crc32	according to IEEE802.3

5.18.2.2 vos_getVersion()

```
EXT_DECL const VOS_VERSION_T* vos_getVersion (
            void  )
```

Return version.

Return pointer to version structure

Return values

VOS_VERSION_T	

5.18.2.3 vos_getVersionString()

```
const char* vos_getVersionString (
            void  )
```

Return a human readable version representation.

Return string in the form 'v.r.u.b'

Return values

const	string

5.18.2.4 vos_init()

```
VOS_ERR_T vos_init (
            void * pRefCon,
            VOS_PRINT_DBG_T pDebugOutput )
```

Initialize the virtual operating system.

Initialize the vos library.

Parameters

in	*pRefCon*	context for debug output function
in	*pDebugOutput*	Pointer to debug output function.

Return values

VOS_NO_ERR	no error VOS_INTEGRATION_ERR if endianess/alignment mismatch VOS_SOCK_ERR sockets not supported VOS_UNKNOWN_ERR initialisation error

5.18.2.5 vos_initRuntimeConsts()

```
VOS_ERR_T vos_initRuntimeConsts (
            void  )
```

Pre-compute alignment and endianess.

Return values

VOS_INTEGRATION_ERR	or VOS_NO_ERR

5.18.2.6 vos_sc32()

```
UINT32 vos_sc32 (
            UINT32 crc,
            const UINT8 * pData,
            UINT32 dataLen )
```

Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.

Parameters

in	*crc*	Initial value.
in,out	*pData*	Pointer to data.
in	*dataLen*	length in bytes of data.

Return values

crc32	according to IEC 61375-2-3

5.18.2.7 vos_terminate()

```
EXT_DECL void vos_terminate ( )
```

DeInitialize the vos library.

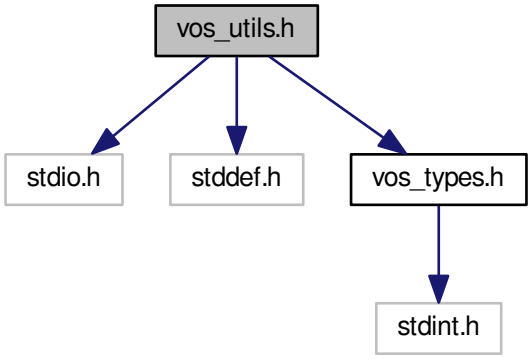
Should be called last after TRDP stack/application does not use any VOS function anymore.

5.19 vos_utils.h File Reference

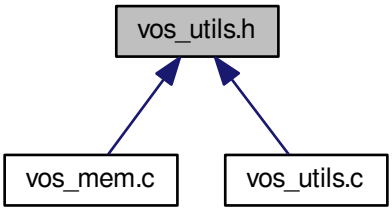
Typedefs for OS abstraction.

```
#include <stdio.h>
#include <stddef.h>
#include "vos_types.h"
```

Include dependency graph for vos_utils.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define VOS_MAX_PRNT_STR_SIZE 256

  String size definitions for the debug output functions.
- #define VOS_MAX_FRMT_SIZE 64

  Max.
- #define VOS_MAX_ERR_STR_SIZE (VOS_MAX_PRNT_STR_SIZE - VOS_MAX_FRMT_SIZE)

  Max.

- #define vos_snprintf(str, size, format, args ...) snprintf(str, size, format, ## args)

  Safe printf function.
- #define vos_printLogStr(level, string)

  Debug output macro without formatting options.
- #define vos_printLog(level, format, args ...)

  Debug output macro with formatting options.
- #define ALIGNOF(type) ((UINT32)offsetof(struct { char c; type member; }, member))

  Alignment macros.
- #define INITFCS 0xffffffff

  CRC/FCS constants.
- #define SIZE_OF_FCS 4

  for better understanding of address calculations
- #define L_ENDIAN

  Define endianess if not already done by compiler.

Functions

- EXT_DECL UINT32 vos_crc32 (UINT32 crc, const UINT8 *pData, UINT32 dataLen)

  Calculate CRC for the given buffer and length.
- EXT_DECL UINT32 vos_sc32 (UINT32 crc, const UINT8 *pData, UINT32 dataLen)

  Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.
- EXT_DECL VOS_ERR_T vos_init (void *pRefCon, VOS_PRINT_DBG_T pDebugOutput)

  Initialize the vos library.
- EXT_DECL void vos_terminate ()

  DeInitialize the vos library.
- EXT_DECL const CHAR8 * vos_getVersionString (void)

  Return a human readable version representation.
- EXT_DECL const VOS_VERSION_T * vos_getVersion (void)

  Return version.

5.19.1 Detailed Description

Typedefs for OS abstraction.

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH

Remarks

Id

```
BL 2016-03-10: Ticket #114 SC-32
BL 2014-02-28: Ticket #25: CRC32 calculation is not according IEEE802.3
```

5.19.2 Macro Definition Documentation

5.19.2.1 INITFCS

```
#define INITFCS 0xffffffff
```

CRC/FCS constants.

Initial FCS value

5.19.2.2 VOS_MAX_ERR_STR_SIZE

```
#define VOS_MAX_ERR_STR_SIZE (VOS_MAX_PRNT_STR_SIZE - VOS_MAX_FRMT_SIZE)
```

Max.

size of the error part

5.19.2.3 VOS_MAX_FRMT_SIZE

```
#define VOS_MAX_FRMT_SIZE 64
```

Max.

size of the 'format' part

5.19.2.4 VOS_MAX_PRNT_STR_SIZE

```
#define VOS_MAX_PRNT_STR_SIZE 256
```

String size definitions for the debug output functions.

Max. size of the debug/error string of debug function

5.19.3 Function Documentation

5.19.3.1 vos_crc32()

```
EXT_DECL UINT32 vos_crc32 (
            UINT32 crc,
            const UINT8 * pData,
            UINT32 dataLen )
```

Calculate CRC for the given buffer and length.

For TRDP FCS CRC calculation the CRC32 according to IEEE802.3 with start value 0xffffffff is used.

Parameters

in	*crc*	Initial value.
in,out	*pData*	Pointer to data.
in	*dataLen*	length in bytes of data.

Return values

crc32	according to IEEE802.3

Calculate CRC for the given buffer and length.

/ to IEC 61375-2-3 A.3 Note: Returned CRC is inverted

Parameters

in	*crc*	Initial value.
in,out	*pData*	Pointer to data.
in	*dataLen*	length in bytes of data.

Return values

crc32	according to IEEE802.3

5.19.3.2 vos_getVersion()

```
EXT_DECL const VOS_VERSION_T* vos_getVersion (
            void  )
```

Return version.

Return pointer to version structure

Return values

const	VOS_VERSION↩ _T

Return pointer to version structure

Return values

VOS_VERSION↩ _T	

5.19.3.3 vos_getVersionString()

```
EXT_DECL const CHAR8* vos_getVersionString (
            void  )
```

Return a human readable version representation.

Return string in the form 'v.r.u.b'

Return values

const	string

5.19.3.4 vos_init()

```
EXT_DECL VOS_ERR_T vos_init (
            void * pRefCon,
            VOS_PRINT_DBG_T pDebugOutput )
```

Initialize the vos library.

This is used to set the output function for all VOS error and debug output.

Parameters

in	*pRefCon	user context
in	*pDebugOutput	pointer to debug output function

Return values

VOS_NO_ERR	no error
VOS_INIT_ERR	unsupported

Initialize the vos library.

Parameters

in	*pRefCon*	context for debug output function
in	*pDebugOutput*	Pointer to debug output function.

Return values

VOS_NO_ERR	no error VOS_INTEGRATION_ERR if endianess/alignment mismatch VOS_SOCK_ERR sockets not supported VOS_UNKNOWN_ERR initialisation error

5.19.3.5 vos_sc32()

```
EXT_DECL UINT32 vos_sc32 (
```

```
            UINT32 crc,
            const UINT8 * pData,
            UINT32 dataLen )
```

Compute crc32 according to IEC 61375-2-3 B.7 Note: Returned CRC is inverted.

Parameters

in	*crc*	Initial value.
in,out	*pData*	Pointer to data.
in	*dataLen*	length in bytes of data.

Return values

crc32	according to IEC 61375-2-3

5.19.3.6 vos_terminate()

```
EXT_DECL void vos_terminate ( )
```

DeInitialize the vos library.

Should be called last after TRDP stack/application does not use any VOS function anymore.

Index