

TCNOpen TRDP
1.2

Generated by Doxygen 1.8.4

Tue Nov 11 2014 16:28:51

Contents

1	The TRDP Light Library API Specification	1
1.1	General Information	1
1.1.1	Purpose	1
1.1.2	Scope	1
1.1.3	Related documents	1
1.1.4	Abbreviations and Definitions	1
1.2	Terminology	1
1.3	Conventions of the API	4
2	Data Structure Index	5
2.1	Data Structures	5
3	File Index	7
3.1	File List	7
4	Data Structure Documentation	9
4.1	GNU_PACKED Struct Reference	9
4.1.1	Detailed Description	14
4.1.2	Field Documentation	14
4.1.2.1	confVehCnt	14
4.1.2.2	confVehList	14
4.1.2.3	cstList	14
4.1.2.4	cstUUID	15
4.1.2.5	datasetLength	15
4.1.2.6	deviceName	15
4.1.2.7	etbTopoCnt	15
4.1.2.8	inhibit	15
4.1.2.9	isLead	15
4.1.2.10	leadDir	15
4.1.2.11	lifesign	15
4.1.2.12	msgType	16
4.1.2.13	opCstList	16

4.1.2.14	opTrnDirState	16
4.1.2.15	opTrnTopoCnt	16
4.1.2.16	opVehList	16
4.1.2.17	ownOpCstNo	16
4.1.2.18	protocolVersion	16
4.1.2.19	reserved01	16
4.1.2.20	reserved01	17
4.1.2.21	reserved02	17
4.1.2.22	reserved02	17
4.1.2.23	reserved03	17
4.1.2.24	reserved04	17
4.1.2.25	reserved06	17
4.1.2.26	safetyTrail	17
4.1.2.27	trnCstNo	17
4.1.2.28	trnDirState	18
4.1.2.29	trnId	18
4.1.2.30	trnOperator	18
4.1.2.31	trnTopoCnt	18
4.1.2.32	trnVehNo	18
4.1.2.33	vehId	18
4.1.2.34	vehOrient	18
4.1.2.35	version	18
4.2	PD_ELE Struct Reference	19
4.2.1	Detailed Description	20
4.2.2	Field Documentation	20
4.2.2.1	pFrame	20
4.3	TAU_MARSHALL_INFO_T Struct Reference	20
4.3.1	Detailed Description	21
4.4	TRDP_CLTR_CST_INFO_T Struct Reference	21
4.4.1	Detailed Description	21
4.5	TRDP_COMID_DSID_MAP_T Struct Reference	21
4.5.1	Detailed Description	21
4.6	TRDP_CONSIST_INFO_T Struct Reference	22
4.6.1	Detailed Description	23
4.6.2	Field Documentation	23
4.6.2.1	cstId	23
4.6.2.2	cstOwner	23
4.7	TRDP_DATASET Struct Reference	23
4.7.1	Detailed Description	23
4.8	TRDP_DATASET_ELEMENT_T Struct Reference	23

4.8.1	Detailed Description	24
4.9	TRDP_DBG_CONFIG_T Struct Reference	24
4.9.1	Detailed Description	24
4.10	TRDP_ETB_INFO_T Struct Reference	24
4.10.1	Detailed Description	25
4.10.2	Field Documentation	25
4.10.2.1	cnCnt	25
4.11	TRDP_FUNCTION_INFO_T Struct Reference	25
4.11.1	Detailed Description	25
4.11.2	Field Documentation	26
4.11.2.1	cnId	26
4.11.2.2	cstVehNo	26
4.11.2.3	etbId	26
4.11.2.4	fctId	26
4.12	TRDP_HANDLE Struct Reference	26
4.12.1	Detailed Description	27
4.13	TRDP_LIST_STATISTICS_T Struct Reference	27
4.13.1	Detailed Description	27
4.14	TRDP_MARSHALL_CONFIG_T Struct Reference	27
4.14.1	Detailed Description	28
4.15	TRDP_MD_CONFIG_T Struct Reference	28
4.15.1	Detailed Description	28
4.16	TRDP_MD_INFO_T Struct Reference	28
4.16.1	Detailed Description	30
4.17	TRDP_MD_STATISTICS_T Struct Reference	30
4.17.1	Detailed Description	30
4.18	TRDP_MEM_CONFIG_T Struct Reference	31
4.18.1	Detailed Description	31
4.19	TRDP_MEM_STATISTICS_T Struct Reference	31
4.19.1	Detailed Description	32
4.20	TRDP_PD_CONFIG_T Struct Reference	32
4.20.1	Detailed Description	32
4.21	TRDP_PD_INFO_T Struct Reference	33
4.21.1	Detailed Description	33
4.22	TRDP_PD_STATISTICS_T Struct Reference	33
4.22.1	Detailed Description	34
4.23	TRDP_PROCESS_CONFIG_T Struct Reference	34
4.23.1	Detailed Description	35
4.24	TRDP_PROP_T Struct Reference	35
4.24.1	Detailed Description	35

4.25	TRDP_PUB_STATISTICS_T Struct Reference	35
4.25.1	Detailed Description	36
4.25.2	Field Documentation	36
4.25.2.1	destAddr	36
4.26	TRDP_RED_STATISTICS_T Struct Reference	36
4.26.1	Detailed Description	36
4.27	TRDP_SDT_PAR_T Struct Reference	36
4.27.1	Detailed Description	37
4.28	TRDP_SEND_PARAM_T Struct Reference	37
4.28.1	Detailed Description	37
4.29	TRDP_SEQ_CNT_ENTRY_T Struct Reference	38
4.29.1	Detailed Description	38
4.30	TRDP_SESSION Struct Reference	38
4.30.1	Detailed Description	39
4.31	TRDP_SOCKET_TCP Struct Reference	39
4.31.1	Detailed Description	40
4.32	TRDP_SOCKETS Struct Reference	40
4.32.1	Detailed Description	40
4.32.2	Field Documentation	40
4.32.2.1	usage	40
4.33	TRDP_STATISTICS_T Struct Reference	41
4.33.1	Detailed Description	41
4.34	TRDP_SUBS_STATISTICS_T Struct Reference	42
4.34.1	Detailed Description	42
4.34.2	Field Documentation	42
4.34.2.1	filterAddr	42
4.34.2.2	numRecv	42
4.34.2.3	timeout	42
4.34.2.4	toBehav	43
4.35	TRDP_VEHICLE_INFO_T Struct Reference	43
4.35.1	Detailed Description	43
4.35.2	Field Documentation	43
4.35.2.1	vehId	43
4.36	TRDP_VERSION_T Struct Reference	44
4.36.1	Detailed Description	44
4.37	TRDP_XML_DOC_HANDLE_T Struct Reference	44
4.37.1	Detailed Description	44
4.38	VOS SOCK_OPT_T Struct Reference	44
4.38.1	Detailed Description	45
4.39	VOS_TIME_T Struct Reference	45

4.39.1	Detailed Description	45
4.39.2	Field Documentation	45
4.39.2.1	tv_usec	45
5	File Documentation	47
5.1	tau_ctrl.c File Reference	47
5.1.1	Detailed Description	47
5.1.2	Function Documentation	48
5.1.2.1	tau_getEcspStat	48
5.1.2.2	tau_initEcspCtrl	48
5.1.2.3	tau_requestEcspConfirm	48
5.1.2.4	tau_setEcspCtrl	49
5.1.2.5	tau_terminateEcspCtrl	49
5.2	tau_ctrl.h File Reference	49
5.2.1	Detailed Description	50
5.2.2	Function Documentation	50
5.2.2.1	tau_getEcspStat	50
5.2.2.2	tau_initEcspCtrl	51
5.2.2.3	tau_requestEcspConfirm	51
5.2.2.4	tau_setEcspCtrl	52
5.2.2.5	tau_terminateEcspCtrl	52
5.3	tau_ctrl_types.h File Reference	52
5.3.1	Detailed Description	53
5.4	tau_dnr.c File Reference	53
5.4.1	Detailed Description	54
5.5	tau_dnr.h File Reference	54
5.5.1	Detailed Description	55
5.5.2	Function Documentation	56
5.5.2.1	tau_addr2CstId	56
5.5.2.2	tau_addr2OpCstNo	56
5.5.2.3	tau_addr2OpVehNo	56
5.5.2.4	tau_addr2TcnCstNo	57
5.5.2.5	tau_addr2TcnVehNo	57
5.5.2.6	tau_addr2Uri	57
5.5.2.7	tau_addr2VehId	58
5.5.2.8	tau_getOwnAddr	58
5.5.2.9	tau_getOwnIds	58
5.5.2.10	tau_iecCstNo2CstId	58
5.5.2.11	tau_initDnr	59
5.5.2.12	tau_label2CstId	59

5.5.2.13	tau_label2OpCstNo	59
5.5.2.14	tau_label2OpVehNo	59
5.5.2.15	tau_label2TcnCstNo	60
5.5.2.16	tau_label2TcnVehNo	60
5.5.2.17	tau_label2VehId	60
5.5.2.18	tau_opVehNo2Ids	61
5.5.2.19	tau_tcnCstNo2CstId	61
5.5.2.20	tau_tcnVehNo2Ids	61
5.5.2.21	tau_uri2Addr	62
5.6	tau_marshall.c File Reference	62
5.6.1	Detailed Description	63
5.6.2	Function Documentation	63
5.6.2.1	tau_calcDatasetSize	63
5.6.2.2	tau_calcDatasetSizeByComId	64
5.6.2.3	tau_initMarshall	64
5.6.2.4	tau_marshall	64
5.6.2.5	tau_marshallDs	65
5.6.2.6	tau_unmarshall	65
5.6.2.7	tau_unmarshallDs	66
5.7	tau_marshall.h File Reference	66
5.7.1	Detailed Description	67
5.7.2	Function Documentation	67
5.7.2.1	tau_calcDatasetSize	67
5.7.2.2	tau_calcDatasetSizeByComId	68
5.7.2.3	tau_initMarshall	68
5.7.2.4	tau_marshall	69
5.7.2.5	tau_marshallDs	69
5.7.2.6	tau_unmarshall	70
5.7.2.7	tau_unmarshallDs	70
5.8	tau_tti.c File Reference	70
5.8.1	Detailed Description	71
5.9	tau_tti.h File Reference	71
5.9.1	Detailed Description	72
5.9.2	Function Documentation	73
5.9.2.1	tau_getCarDevCnt	73
5.9.2.2	tau_getCstCarCnt	74
5.9.2.3	tau_getCstFctCnt	74
5.9.2.4	tau_getCstFctInfo	74
5.9.2.5	tau_getCstInfo	75
5.9.2.6	tau_getlecCarOrient	75

5.9.2.7	tau_getOpTrDirectory	75
5.9.2.8	tau_getStaticCstInfo	76
5.9.2.9	tau_getTrDirectory	76
5.9.2.10	tau_getTrnCarCnt	76
5.9.2.11	tau_getTrnCstCnt	76
5.9.2.12	tau_getTTI	77
5.9.2.13	tau_getVehInfo	77
5.9.2.14	tau_getVehOrient	77
5.9.2.15	tau_initTtiAccess	78
5.10	tau_tti_types.h File Reference	78
5.10.1	Detailed Description	79
5.11	tau_xml.c File Reference	79
5.11.1	Detailed Description	80
5.11.2	Macro Definition Documentation	81
5.11.2.1	TRDP_SDT_DEFAULT_CMTHR	81
5.11.3	Function Documentation	81
5.11.3.1	tau_freeTelegrams	81
5.11.3.2	tau_freeXmlDoc	81
5.11.3.3	tau_prepareXmlDoc	81
5.11.3.4	tau_readXmlDatasetConfig	81
5.11.3.5	tau_readXmlDeviceConfig	82
5.11.3.6	tau_readXmlInterfaceConfig	82
5.12	tau_xml.h File Reference	83
5.12.1	Detailed Description	84
5.12.2	Enumeration Type Documentation	84
5.12.2.1	TRDP_DBG_OPTION_T	84
5.12.3	Function Documentation	84
5.12.3.1	tau_freeTelegrams	84
5.12.3.2	tau_freeXmlDoc	85
5.12.3.3	tau_prepareXmlDoc	85
5.12.3.4	tau_readXmlDatasetConfig	85
5.12.3.5	tau_readXmlDeviceConfig	85
5.12.3.6	tau_readXmlInterfaceConfig	86
5.13	trdp_dllmain.c File Reference	86
5.13.1	Detailed Description	86
5.14	trdp_if.c File Reference	87
5.14.1	Detailed Description	88
5.14.2	Function Documentation	89
5.14.2.1	tlc_closeSession	89
5.14.2.2	tlc_getInterval	89

5.14.2.3	tlc_getVersion	90
5.14.2.4	tlc_getVersionString	90
5.14.2.5	tlc_init	90
5.14.2.6	tlc_openSession	91
5.14.2.7	tlc_process	91
5.14.2.8	tlc_reinitSession	91
5.14.2.9	tlc_setETBTopoCount	92
5.14.2.10	tlc_setOpTrainTopoCount	92
5.14.2.11	tlc_terminate	92
5.14.2.12	tlp_get	93
5.14.2.13	tlp_getRedundant	93
5.14.2.14	tlp_publish	93
5.14.2.15	tlp_put	94
5.14.2.16	tlp_republish	94
5.14.2.17	tlp_request	95
5.14.2.18	tlp_resubscribe	96
5.14.2.19	tlp_setRedundant	96
5.14.2.20	tlp_subscribe	96
5.14.2.21	tlp_unpublish	97
5.14.2.22	tlp_unsubscribe	97
5.14.2.23	trdp_isValidSession	98
5.14.2.24	trdp_sessionQueue	98
5.15	trdp_if.h File Reference	98
5.15.1	Detailed Description	98
5.15.2	Function Documentation	99
5.15.2.1	trdp_isValidSession	99
5.15.2.2	trdp_sessionQueue	99
5.16	trdp_if_light.h File Reference	99
5.16.1	Detailed Description	102
5.16.2	Function Documentation	102
5.16.2.1	tlc_closeSession	102
5.16.2.2	tlc_freeBuf	103
5.16.2.3	tlc_getInterval	103
5.16.2.4	tlc_getJoinStatistics	104
5.16.2.5	tlc_getListStatistics	104
5.16.2.6	tlc_getPubStatistics	105
5.16.2.7	tlc_getRedStatistics	105
5.16.2.8	tlc_getStatistics	106
5.16.2.9	tlc_getSubsStatistics	106
5.16.2.10	tlc_getVersion	107

5.16.2.11	tlc_getVersionString	107
5.16.2.12	tlc_init	108
5.16.2.13	tlc_openSession	108
5.16.2.14	tlc_process	108
5.16.2.15	tlc_reinitSession	109
5.16.2.16	tlc_resetStatistics	109
5.16.2.17	tlc_setETBTopoCount	110
5.16.2.18	tlc_setOpTrainTopoCount	110
5.16.2.19	tlc_terminate	110
5.16.2.20	tIm_abortSession	111
5.16.2.21	tIm_addListener	111
5.16.2.22	tIm_confirm	112
5.16.2.23	tIm_delListener	112
5.16.2.24	tIm_notify	112
5.16.2.25	tIm_readdListener	113
5.16.2.26	tIm_reply	113
5.16.2.27	tIm_replyErr	114
5.16.2.28	tIm_replyQuery	114
5.16.2.29	tIm_request	115
5.16.2.30	tIp_get	115
5.16.2.31	tIp_getRedundant	117
5.16.2.32	tIp_publish	118
5.16.2.33	tIp_put	119
5.16.2.34	tIp_republish	120
5.16.2.35	tIp_request	120
5.16.2.36	tIp_resubscribe	121
5.16.2.37	tIp_setRedundant	122
5.16.2.38	tIp_subscribe	123
5.16.2.39	tIp_unpublish	124
5.16.2.40	tIp_unsubscribe	124
5.17	trdp_mdcom.c File Reference	125
5.17.1	Detailed Description	125
5.17.2	Function Documentation	126
5.17.2.1	trdp_mdCheckListenSocks	126
5.17.2.2	trdp_mdCheckPending	126
5.17.2.3	trdp_mdCheckTimeouts	126
5.17.2.4	trdp_mdFreeSession	126
5.17.2.5	trdp_mdGetTCPSocket	127
5.17.2.6	trdp_mdSend	127
5.18	trdp_mdcom.h File Reference	127

5.18.1	Detailed Description	128
5.18.2	Function Documentation	128
5.18.2.1	trdp_mdCheckListenSocks	128
5.18.2.2	trdp_mdCheckPending	128
5.18.2.3	trdp_mdCheckTimeouts	128
5.18.2.4	trdp_mdFreeSession	129
5.18.2.5	trdp_mdGetTCPSocket	129
5.18.2.6	trdp_mdSend	129
5.19	trdp_pdcom.c File Reference	129
5.19.1	Detailed Description	130
5.19.2	Function Documentation	131
5.19.2.1	trdp_pdCheck	131
5.19.2.2	trdp_pdCheckAppTopoCounts	131
5.19.2.3	trdp_pdCheckListenSocks	131
5.19.2.4	trdp_pdCheckPending	132
5.19.2.5	trdp_pdDistribute	133
5.19.2.6	trdp_pdHandleTimeOuts	133
5.19.2.7	trdp_pdInit	133
5.19.2.8	trdp_pdReceive	133
5.19.2.9	trdp_pdSend	134
5.19.2.10	trdp_pdSendQueued	134
5.19.2.11	trdp_pdUpdate	134
5.20	trdp_pdcom.h File Reference	135
5.20.1	Detailed Description	135
5.20.2	Function Documentation	136
5.20.2.1	trdp_pdCheck	136
5.20.2.2	trdp_pdCheckAppTopoCounts	136
5.20.2.3	trdp_pdCheckListenSocks	136
5.20.2.4	trdp_pdCheckPending	137
5.20.2.5	trdp_pdDistribute	137
5.20.2.6	trdp_pdHandleTimeOuts	137
5.20.2.7	trdp_pdInit	137
5.20.2.8	trdp_pdReceive	138
5.20.2.9	trdp_pdSend	138
5.20.2.10	trdp_pdSendQueued	138
5.20.2.11	trdp_pdUpdate	139
5.21	trdp_private.h File Reference	139
5.21.1	Detailed Description	141
5.21.2	Enumeration Type Documentation	141
5.21.2.1	TRDP_MD_ELE_ST_T	141

5.21.2.2	TRDP_PRIV_FLAGS_T	142
5.21.2.3	TRDP SOCK_TYPE_T	142
5.22	trdp_proto.h File Reference	142
5.22.1	Detailed Description	143
5.22.2	Macro Definition Documentation	144
5.22.2.1	TRDP_DEST_URI_SIZE	144
5.22.2.2	TRDP_ETBCTRL_COMID	144
5.22.2.3	TRDP_ETBCTRL_DSID	144
5.22.2.4	TRDP_MAX_FILE_NAME_LEN	144
5.22.2.5	TRDP_MAX_LABEL_LEN	144
5.22.2.6	TRDP_MAX_URI_HOST_LEN	144
5.22.2.7	TRDP_MAX_URI_LEN	145
5.22.2.8	TRDP_MAX_URI_USER_LEN	145
5.22.3	Enumeration Type Documentation	145
5.22.3.1	TRDP_MSG_T	145
5.23	trdp_stats.c File Reference	145
5.23.1	Detailed Description	146
5.23.2	Function Documentation	146
5.23.2.1	tlc_getJoinStatistics	146
5.23.2.2	tlc_getPubStatistics	147
5.23.2.3	tlc_getRedStatistics	147
5.23.2.4	tlc_getStatistics	148
5.23.2.5	tlc_getSubsStatistics	148
5.23.2.6	tlc_resetStatistics	148
5.23.2.7	trdp_initStats	148
5.23.2.8	trdp_pdPrepareStats	150
5.23.2.9	trdp_UpdateStats	150
5.24	trdp_stats.h File Reference	150
5.24.1	Detailed Description	150
5.24.2	Function Documentation	151
5.24.2.1	trdp_initStats	151
5.24.2.2	trdp_pdPrepareStats	151
5.25	trdp_types.h File Reference	151
5.25.1	Detailed Description	155
5.25.2	Typedef Documentation	155
5.25.2.1	TRDP_IP_ADDR_T	155
5.25.2.2	TRDP_MARSHALL_T	156
5.25.2.3	TRDP_MD_CALLBACK_T	156
5.25.2.4	TRDP_PD_CALLBACK_T	156
5.25.2.5	TRDP_PRINT_DBG_T	156

5.25.2.6	TRDP_TIME_T	156
5.25.2.7	TRDP_UNMARSHALL_T	157
5.25.3	Enumeration Type Documentation	157
5.25.3.1	TRDP_DATA_TYPE_T	157
5.25.3.2	TRDP_ERR_T	158
5.25.3.3	TRDP_FLAGS_T	159
5.25.3.4	TRDP_OPTION_T	159
5.25.3.5	TRDP_RED_STATE_T	159
5.25.3.6	TRDP_REPLY_STATUS_T	159
5.25.3.7	TRDP_TO_BEHAVIOR_T	159
5.26	trdp_utils.c File Reference	160
5.26.1	Detailed Description	161
5.26.2	Function Documentation	161
5.26.2.1	printSocketUsage	161
5.26.2.2	trdp_checkSequenceCounter	161
5.26.2.3	trdp_getSeqCnt	162
5.26.2.4	trdp_initSockets	162
5.26.2.5	trdp_isAddressed	162
5.26.2.6	trdp_packetSizeMD	162
5.26.2.7	trdp_packetSizePD	163
5.26.2.8	trdp_queueAppLast	163
5.26.2.9	trdp_queueDelElement	163
5.26.2.10	trdp_queueFindComId	163
5.26.2.11	trdp_queueFindPubAddr	163
5.26.2.12	trdp_queueFindSubAddr	164
5.26.2.13	trdp_queueInsFirst	164
5.26.2.14	trdp_releaseSocket	164
5.26.2.15	trdp_requestSocket	164
5.26.2.16	trdp_resetSequenceCounter	165
5.26.2.17	trdp_SockAddJoin	165
5.26.2.18	trdp_SockDelJoin	166
5.26.2.19	trdp_SockIsJoined	167
5.27	trdp_utils.h File Reference	167
5.27.1	Detailed Description	168
5.27.2	Function Documentation	168
5.27.2.1	trdp_checkSequenceCounter	168
5.27.2.2	trdp_getSeqCnt	169
5.27.2.3	trdp_initSockets	169
5.27.2.4	trdp_initUncompletedTCP	169
5.27.2.5	trdp_isAddressed	169

5.27.2.6	trdp_packetSizeMD	170
5.27.2.7	trdp_packetSizePD	170
5.27.2.8	trdp_queueAppLast	170
5.27.2.9	trdp_queueDelElement	170
5.27.2.10	trdp_queueFindComId	170
5.27.2.11	trdp_queueFindPubAddr	171
5.27.2.12	trdp_queueFindSubAddr	171
5.27.2.13	trdp_queueInsFirst	171
5.27.2.14	trdp_releaseSocket	171
5.27.2.15	trdp_requestSocket	172
5.27.2.16	trdp_resetSequenceCounter	173
5.28	vos_mem.c File Reference	173
5.28.1	Detailed Description	174
5.28.2	Function Documentation	175
5.28.2.1	vos_bsearch	175
5.28.2.2	vos_memAlloc	175
5.28.2.3	vos_memCount	175
5.28.2.4	vos_memDelete	176
5.28.2.5	vos_memFree	176
5.28.2.6	vos_memInit	176
5.28.2.7	vos_mutexLocalCreate	176
5.28.2.8	vos_mutexLocalDelete	177
5.28.2.9	vos_qsort	177
5.28.2.10	vos_queueCreate	177
5.28.2.11	vos_queueDestroy	178
5.28.2.12	vos_queueReceive	178
5.28.2.13	vos_queueSend	178
5.28.2.14	vos_strncpy	179
5.28.2.15	vos_strncmp	179
5.29	vos_mem.h File Reference	179
5.29.1	Detailed Description	181
5.29.2	Macro Definition Documentation	181
5.29.2.1	VOS_MEM_BLOCKSIZE	181
5.29.2.2	VOS_MEM_PREALLOCATE	181
5.29.3	Function Documentation	181
5.29.3.1	vos_bsearch	181
5.29.3.2	vos_memAlloc	182
5.29.3.3	vos_memCount	182
5.29.3.4	vos_memDelete	182
5.29.3.5	vos_memFree	183

5.29.3.6	vos_memInit	183
5.29.3.7	vos_qsort	184
5.29.3.8	vos_queueCreate	184
5.29.3.9	vos_queueDestroy	184
5.29.3.10	vos_queueReceive	185
5.29.3.11	vos_queueSend	185
5.29.3.12	vos_strncpy	185
5.29.3.13	vos_strncmp	186
5.30	vos_private.h File Reference	186
5.30.1	Detailed Description	186
5.30.2	Function Documentation	187
5.30.2.1	vos_mutexLocalCreate	187
5.30.2.2	vos_mutexLocalDelete	187
5.31	vos_private.h File Reference	187
5.31.1	Detailed Description	188
5.31.2	Function Documentation	188
5.31.2.1	vos_mutexLocalCreate	188
5.31.2.2	vos_mutexLocalDelete	188
5.32	vos_shared_mem.c File Reference	188
5.32.1	Detailed Description	189
5.32.2	Function Documentation	189
5.32.2.1	vos_sharedClose	189
5.32.2.2	vos_sharedOpen	190
5.33	vos_shared_mem.c File Reference	190
5.33.1	Detailed Description	191
5.33.2	Function Documentation	191
5.33.2.1	vos_sharedClose	191
5.33.2.2	vos_sharedOpen	191
5.34	vos_shared_mem.h File Reference	192
5.34.1	Detailed Description	192
5.34.2	Function Documentation	193
5.34.2.1	vos_sharedClose	193
5.34.2.2	vos_sharedOpen	193
5.35	vos_sock.c File Reference	194
5.35.1	Detailed Description	196
5.35.2	Function Documentation	196
5.35.2.1	vos_dottedIP	196
5.35.2.2	vos_getInterfaces	196
5.35.2.3	vos_getMacAddress	196
5.35.2.4	vos_htonl	197

5.35.2.5	vos_htons	197
5.35.2.6	vos_ipDotted	197
5.35.2.7	vos_isMulticast	197
5.35.2.8	vos_ntohl	198
5.35.2.9	vos_ntohs	198
5.35.2.10	vos_select	198
5.35.2.11	vos_sockAccept	198
5.35.2.12	vos_sockBind	199
5.35.2.13	vos_sockClose	199
5.35.2.14	vos_sockConnect	199
5.35.2.15	vos_sockGetMAC	200
5.35.2.16	vos_sockInit	200
5.35.2.17	vos_sockJoinMC	200
5.35.2.18	vos_sockLeaveMC	200
5.35.2.19	vos_sockListen	201
5.35.2.20	vos_sockOpenTCP	201
5.35.2.21	vos_sockOpenUDP	201
5.35.2.22	vos_sockReceiveTCP	202
5.35.2.23	vos_sockReceiveUDP	202
5.35.2.24	vos_sockSendTCP	203
5.35.2.25	vos_sockSendUDP	203
5.35.2.26	vos_sockSetBuffer	203
5.35.2.27	vos_sockSetMulticastIf	204
5.35.2.28	vos_sockSetOptions	204
5.35.2.29	vos_sockTerm	204
5.36	vos_sock.c File Reference	204
5.36.1	Detailed Description	206
5.36.2	Function Documentation	207
5.36.2.1	recvmsg	207
5.36.2.2	vos_dottedIP	208
5.36.2.3	vos_getInterfaces	208
5.36.2.4	vos_htonl	208
5.36.2.5	vos_htons	208
5.36.2.6	vos_ipDotted	210
5.36.2.7	vos_isMulticast	210
5.36.2.8	vos_ntohl	210
5.36.2.9	vos_ntohs	210
5.36.2.10	vos_select	211
5.36.2.11	vos_sockAccept	211
5.36.2.12	vos_sockBind	211

5.36.2.13	vos_sockClose	212
5.36.2.14	vos_sockConnect	212
5.36.2.15	vos_sockGetMAC	212
5.36.2.16	vos_sockInit	213
5.36.2.17	vos_sockJoinMC	213
5.36.2.18	vos_sockLeaveMC	213
5.36.2.19	vos_sockListen	213
5.36.2.20	vos_sockOpenTCP	214
5.36.2.21	vos_sockOpenUDP	214
5.36.2.22	vos_sockReceiveTCP	214
5.36.2.23	vos_sockReceiveUDP	215
5.36.2.24	vos_sockSendTCP	215
5.36.2.25	vos_sockSendUDP	216
5.36.2.26	vos_sockSetBuffer	216
5.36.2.27	vos_sockSetMulticastIf	216
5.36.2.28	vos_sockSetOptions	217
5.36.2.29	vos_sockTerm	217
5.37	vos_sock.h File Reference	217
5.37.1	Detailed Description	219
5.37.2	Macro Definition Documentation	219
5.37.2.1	VOS_MAX_SOCKET_CNT	219
5.37.2.2	VOS_TTL_MULTICAST	219
5.37.3	Function Documentation	219
5.37.3.1	vos_dottedIP	219
5.37.3.2	vos_getInterfaces	220
5.37.3.3	vos_htonl	220
5.37.3.4	vos_htons	221
5.37.3.5	vos_ipDotted	222
5.37.3.6	vos_isMulticast	222
5.37.3.7	vos_ntohl	223
5.37.3.8	vos_ntohs	224
5.37.3.9	vos_select	224
5.37.3.10	vos_sockAccept	224
5.37.3.11	vos_sockBind	225
5.37.3.12	vos_sockClose	225
5.37.3.13	vos_sockConnect	225
5.37.3.14	vos_sockGetMAC	226
5.37.3.15	vos_sockInit	226
5.37.3.16	vos_sockJoinMC	227
5.37.3.17	vos_sockLeaveMC	227

5.37.3.18	vos_sockListen	228
5.37.3.19	vos_sockOpenTCP	228
5.37.3.20	vos_sockOpenUDP	229
5.37.3.21	vos_sockReceiveTCP	229
5.37.3.22	vos_sockReceiveUDP	230
5.37.3.23	vos_sockSendTCP	231
5.37.3.24	vos_sockSendUDP	232
5.37.3.25	vos_sockSetMulticastIf	233
5.37.3.26	vos_sockSetOptions	233
5.37.3.27	vos_sockTerm	234
5.38	vos_thread.c File Reference	234
5.38.1	Detailed Description	236
5.38.2	Macro Definition Documentation	236
5.38.2.1	NSECS_PER_USEC	236
5.38.3	Function Documentation	236
5.38.3.1	vos_addTime	236
5.38.3.2	vos_clearTime	236
5.38.3.3	vos_cmpTime	237
5.38.3.4	vos_cyclicThread	237
5.38.3.5	vos_divTime	237
5.38.3.6	vos_getTime	237
5.38.3.7	vos_getTimeStamp	238
5.38.3.8	vos_getUuid	238
5.38.3.9	vos_mulTime	238
5.38.3.10	vos_mutexCreate	238
5.38.3.11	vos_mutexDelete	238
5.38.3.12	vos_mutexLocalCreate	239
5.38.3.13	vos_mutexLocalDelete	239
5.38.3.14	vos_mutexLock	239
5.38.3.15	vos_mutexTryLock	239
5.38.3.16	vos_mutexUnlock	240
5.38.3.17	vos_semaCreate	240
5.38.3.18	vos_semaDelete	240
5.38.3.19	vos_semaGive	240
5.38.3.20	vos_semaTake	241
5.38.3.21	vos_subTime	241
5.38.3.22	vos_threadCreate	241
5.38.3.23	vos_threadDelay	242
5.38.3.24	vos_threadInit	242
5.38.3.25	vos_threadIsActive	242

5.38.3.26	vos_threadTerm	242
5.38.3.27	vos_threadTerminate	242
5.39	vos_thread.c File Reference	243
5.39.1	Detailed Description	244
5.39.2	Macro Definition Documentation	245
5.39.2.1	NSECS_PER_USEC	245
5.39.2.2	NSECS_PER_USEC	245
5.39.3	Function Documentation	245
5.39.3.1	vos_addTime	245
5.39.3.2	vos_clearTime	246
5.39.3.3	vos_cmpTime	246
5.39.3.4	vos_cyclicThread	246
5.39.3.5	vos_divTime	246
5.39.3.6	vos_getFreeThreadHandle	247
5.39.3.7	vos_getTime	248
5.39.3.8	vos_getTimeStamp	248
5.39.3.9	vos_getUuid	248
5.39.3.10	vos_mulTime	248
5.39.3.11	vos_mutexCreate	248
5.39.3.12	vos_mutexDelete	249
5.39.3.13	vos_mutexLocalCreate	249
5.39.3.14	vos_mutexLocalDelete	249
5.39.3.15	vos_mutexLock	249
5.39.3.16	vos_mutexTryLock	250
5.39.3.17	vos_mutexUnlock	250
5.39.3.18	vos_semaCreate	250
5.39.3.19	vos_semaDelete	250
5.39.3.20	vos_semaGive	251
5.39.3.21	vos_semaTake	251
5.39.3.22	vos_subTime	251
5.39.3.23	vos_threadCreate	251
5.39.3.24	vos_threadDelay	252
5.39.3.25	vos_threadInit	252
5.39.3.26	vos_threadIsActive	252
5.39.3.27	vos_threadTerm	253
5.39.3.28	vos_threadTerminate	253
5.40	vos_thread.h File Reference	253
5.40.1	Detailed Description	255
5.40.2	Function Documentation	255
5.40.2.1	vos_addTime	255

5.40.2.2	vos_clearTime	255
5.40.2.3	vos_cmpTime	256
5.40.2.4	vos_cyclicThread	256
5.40.2.5	vos_divTime	256
5.40.2.6	vos_getTime	257
5.40.2.7	vos_getTimeStamp	257
5.40.2.8	vos_getUuid	257
5.40.2.9	vos_mulTime	257
5.40.2.10	vos_mutexCreate	257
5.40.2.11	vos_mutexDelete	258
5.40.2.12	vos_mutexLock	258
5.40.2.13	vos_mutexTryLock	259
5.40.2.14	vos_mutexUnlock	259
5.40.2.15	vos_semaCreate	259
5.40.2.16	vos_semaDelete	260
5.40.2.17	vos_semaGive	260
5.40.2.18	vos_semaTake	260
5.40.2.19	vos_subTime	260
5.40.2.20	vos_threadCreate	261
5.40.2.21	vos_threadDelay	262
5.40.2.22	vos_threadInit	262
5.40.2.23	vos_threadIsActive	262
5.40.2.24	vos_threadTerm	263
5.40.2.25	vos_threadTerminate	263
5.41	vos_types.h File Reference	263
5.41.1	Detailed Description	265
5.41.2	Typedef Documentation	265
5.41.2.1	VOS_PRINT_DBG_T	265
5.41.3	Enumeration Type Documentation	265
5.41.3.1	VOS_ERR_T	265
5.41.3.2	VOS_LOG_T	266
5.42	vos_utils.c File Reference	266
5.42.1	Detailed Description	267
5.42.2	Function Documentation	267
5.42.2.1	vos_crc32	267
5.42.2.2	vos_init	267
5.42.2.3	vos_initRuntimeConsts	268
5.42.2.4	vos_isBigEndian	268
5.42.2.5	vos_terminate	268
5.43	vos_utils.h File Reference	268

5.43.1 Detailed Description	269
5.43.2 Macro Definition Documentation	269
5.43.2.1 INITFCS	269
5.43.2.2 VOS_MAX_ERR_STR_SIZE	269
5.43.2.3 VOS_MAX_FRMT_SIZE	270
5.43.2.4 VOS_MAX_PRNT_STR_SIZE	270
5.43.3 Function Documentation	270
5.43.3.1 vos_crc32	270
5.43.3.2 vos_init	270
5.43.3.3 vos_terminate	271

Index

272

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

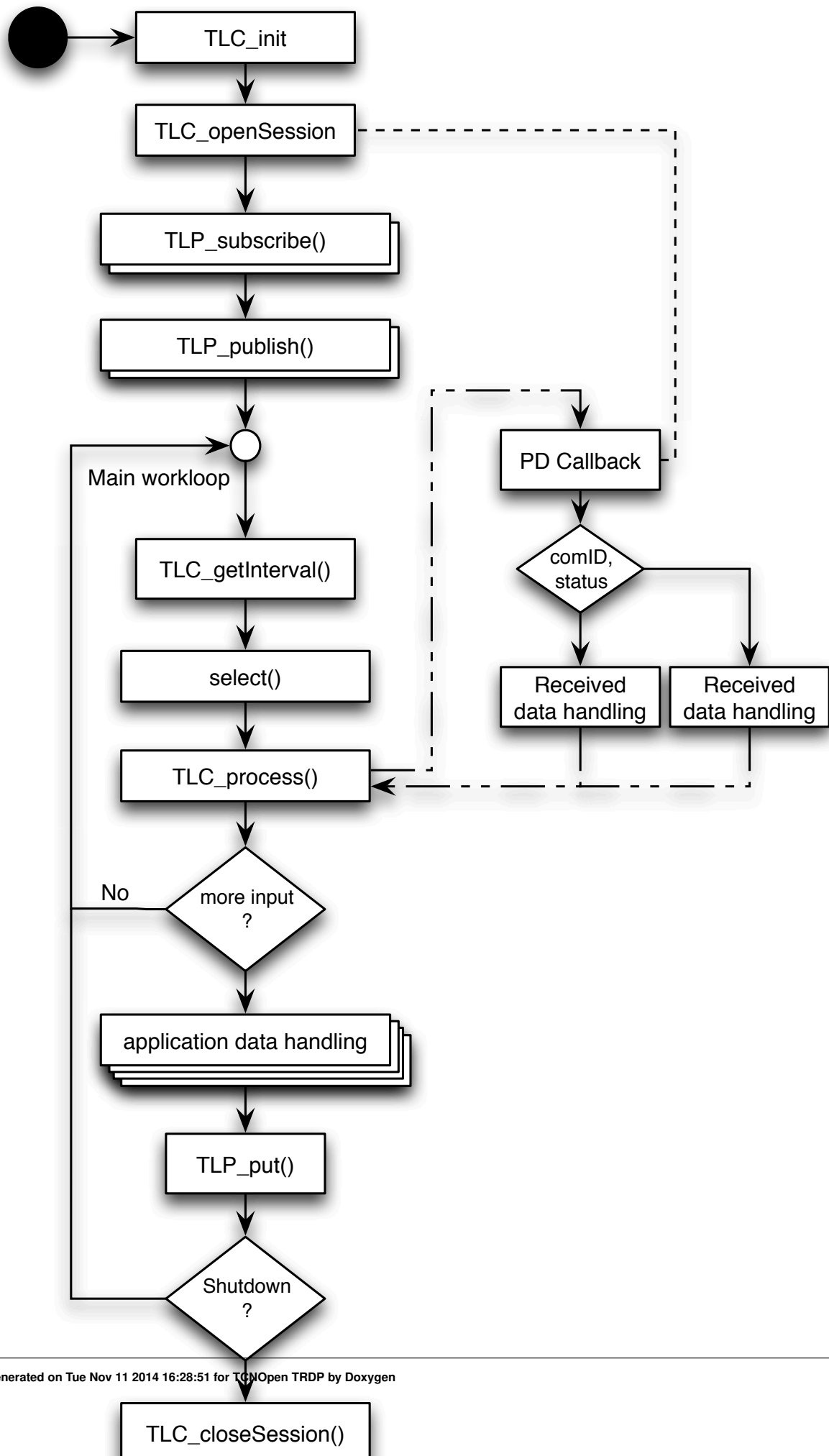
1.1.4 Abbreviations and Definitions

- 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:

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



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:

GNU_PACKED	
Types for ETB control	9
PD_ELE	
Queue element for PD packets to send or receive	19
TAU_MARSHALL_INFO_T	
Marshalling info, used to and from wire	20
TRDP_CLTR_CST_INFO_T	
Closed train consists information	21
TRDP_COMID_DSID_MAP_T	
ComId - data set mapping element definition	21
TRDP_CONSIST_INFO_T	
Consist information structure	22
TRDP_DATASET	
Dataset definition	23
TRDP_DATASET_ELEMENT_T	
Dataset element definition	23
TRDP_DBG_CONFIG_T	
Control for debug output device/file on application level	24
TRDP_ETB_INFO_T	
Types for train configuration information	24
TRDP_FUNCTION_INFO_T	
Function/device information structure	25
TRDP_HANDLE	
Hidden handle definition, used as unique addressing item	26
TRDP_LIST_STATISTICS_T	
Information about a particular MD listener	27
TRDP_MARSHALL_CONFIG_T	
Marshaling/unmarshalling configuration	27
TRDP_MD_CONFIG_T	
Default MD configuration	28
TRDP_MD_INFO_T	
Message data info from received telegram; allows the application to generate responses	28
TRDP_MD_STATISTICS_T	
Structure containing all general MD statistics information	30
TRDP_MEM_CONFIG_T	
Enumeration type for memory pre-fragmentation, reuse of VOS definition	31
TRDP_MEM_STATISTICS_T	
TRDP statistics type definitions	31

TRDP_PD_CONFIG_T	
Default PD configuration	32
TRDP_PD_INFO_T	
Process data info from received telegram; allows the application to generate responses	33
TRDP_PD_STATISTICS_T	
Structure containing all general PD statistics information	33
TRDP_PROCESS_CONFIG_T	
Various flags/general TRDP options for library initialization	34
TRDP_PROP_T	
Application defined properties	35
TRDP_PUB_STATISTICS_T	
Table containing particular PD publishing information	35
TRDP_RED_STATISTICS_T	
A table containing PD redundant group information	36
TRDP_SDT_PAR_T	
Types to read out the XML configuration	36
TRDP_SEND_PARAM_T	
Quality/type of service and time to live	37
TRDP_SEQ_CNT_ENTRY_T	
Tuples of last received sequence counter per comId	38
TRDP_SESSION	
Session/application variables store	38
TRDP_SOCKET_TCP	
TCP parameters	39
TRDP_SOCKETS	
Socket item	40
TRDP_STATISTICS_T	
Structure containing all general memory, PD and MD statistics information	41
TRDP_SUBS_STATISTICS_T	
Table containing particular PD subscription information	42
TRDP_VEHICLE_INFO_T	
Vehicle information structure	43
TRDP_VERSION_T	
Version information	44
TRDP_XML_DOC_HANDLE_T	
Parsed XML document handle	44
VOS SOCK_OPT_T	
Common socket options	44
VOS_TIME_T	
Timer value compatible with timeval / select	45

Chapter 3

File Index

3.1 File List

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

tau_ctrl.c	
Functions for train switch control	47
tau_ctrl.h	
TRDP utility interface definitions	49
tau_ctrl_types.h	
TRDP utility interface definitions	52
tau_dnr.c	
Functions for domain name resolution	53
tau_dnr.h	
TRDP utility interface definitions	54
tau_marshall.c	
Marshalling functions for TRDP	62
tau_marshall.h	
TRDP utility interface definitions	66
tau_tti.c	
Functions for train topology information access	70
tau_tti.h	
TRDP utility interface definitions	71
tau_tti_types.h	
TRDP utility interface definitions	78
tau_xml.c	
Functions for XML file parsing	79
tau_xml.h	
TRDP utility interface definitions	83
trdp_dllmain.c	
Windows DLL main function	86
trdp_if.c	
Functions for ECN communication	87
trdp_if.h	
Typedefs for TRDP communication	98
trdp_if_light.h	
TRDP Light interface functions (API)	99
trdp_mdcom.c	
Functions for MD communication	125
trdp_mdcom.h	
Functions for MD communication	127
trdp_pdcom.c	
Functions for PD communication	129

trdp_pdcom.h	
Functions for PD communication	135
trdp_private.h	
Typedefs for TRDP communication	139
trdp_proto.h	
Definitions for the TRDP protocol	142
trdp_stats.c	
Statistics functions for TRDP communication	145
trdp_stats.h	
Statistics for TRDP communication	150
trdp_types.h	
Typedefs for TRDP communication	151
trdp_utils.c	
Helper functions for TRDP communication	160
trdp_utils.h	
Common utilities for TRDP communication	167
vos_mem.c	
Memory functions	173
vos_mem.h	
Memory and queue functions for OS abstraction	179
posix/vos_private.h	
Private definitions for the OS abstraction layer	186
windows/vos_private.h	
Private definitions for the OS abstraction layer	187
posix/vos_shared_mem.c	
Shared Memory functions	188
windows/vos_shared_mem.c	
Shared Memory functions	190
vos_shared_mem.h	
Shared Memory functions for OS abstraction	192
posix/vos_sock.c	
Socket functions	194
windows/vos_sock.c	
Socket functions	204
vos_sock.h	
Typedefs for OS abstraction	217
posix/vos_thread.c	
Multitasking functions	234
windows/vos_thread.c	
Multitasking functions	243
vos_thread.h	
Threading functions for OS abstraction	253
vos_types.h	
Typedefs for OS abstraction	263
vos_utils.c	
Common functions for VOS	266
vos_utils.h	
Typedefs for OS abstraction	268

Chapter 4

Data Structure Documentation

4.1 GNU_PACKED Struct Reference

Types for ETB control.

```
#include <trdp_private.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.
- **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 = PrepareForSleepMode
- **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
- **UINT8 reserved01**
reserved (=0)
- **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
- **BITSET8 etbld**
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.
 - **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
 - **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...1436
 - **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.
- **PD_HEADER_T frameHead**
Packet header in network byte order.
- **UINT8 data** [TRDP_MAX_PD_DATA_SIZE]
data ready to be sent or received (with CRCs)

4.1.1 Detailed Description

Types for ETB control.

TRDP PD packet.

TRDP message data header - network order and alignment.

TRDP process data header - network order and alignment.

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 **UINT16 GNU_PACKED::confVehCnt**

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

4.1.2.2 **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 **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 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 UINT32 GNU_PACKED::datasetLength

length of the data to transmit 0...1436

defined by user: length of data to transmit

4.1.2.6 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 UINT32 GNU_PACKED::etbTopoCnt

ETB topography counter.

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

train network directory CRC

4.1.2.8 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.

4.1.2.9 ANTIVALENT8 GNU_PACKED::isLead

vehicle is leading

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

4.1.2.10 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

4.1.2.11 UINT16 GNU_PACKED::lifesign

wrap-around counter, incremented with each produced datagram.

4.1.2.12 UINT16 GNU_PACKED::msgType

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

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

4.1.2.13 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

4.1.2.14 UINT8 GNU_PACKED::opTrnDirState

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

Operatiobal train directory status: '01'B == inalid, '10'B == valid.

4.1.2.15 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

4.1.2.16 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

4.1.2.17 UINT8 GNU_PACKED::ownOpCstNo

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

operational consist number the vehicle belongs to

after Inauguration)

4.1.2.18 UINT16 GNU_PACKED::protocolVersion

fix value for compatibility (set by the API)

fix value for compatibility

4.1.2.19 UINT16 GNU_PACKED::reserved01

reserved (=0)

reserved for future use (= 0)

4.1.2.20 `UINT8 GNU_PACKED::reserved01`

reserved (=0)

reserved for future use (= 0)

4.1.2.21 `UINT8 GNU_PACKED::reserved02`

reserved (=0)

reserved (= 0)

reserved for future use (= 0)

4.1.2.22 `UINT16 GNU_PACKED::reserved02`

reserved (=0)

reserved (= 0)

4.1.2.23 `UINT8 GNU_PACKED::reserved03`

reserved (=0)

reserved for future use (= 0)

4.1.2.24 `UINT8 GNU_PACKED::reserved04`

reserved (=0)

reserved for future use (= 0)

4.1.2.25 `UINT8 GNU_PACKED::reserved06`

reserved (=0)

reserved for future use (= 0)

4.1.2.26 `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 == 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.

4.1.2.27 `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)

4.1.2.28 UINT8 GNU_PACKED::trnDirState

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

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

4.1.2.29 TRDP_LABEL_T GNU_PACKED::trnId

train identifier, application defined (e.g.

ICE75, IC346), informal

4.1.2.30 TRDP_LABEL_T GNU_PACKED::trnOperator

train operator, e.g.

trenitalia.it, informal

4.1.2.31 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)

4.1.2.32 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

4.1.2.33 TRDP_LABEL_T GNU_PACKED::vehId

Unique vehicle identifier, application defined (e.g.

UIC Identifier)

4.1.2.34 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

4.1.2.35 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**
- **trdp_private.h**

4.2 PD_ELE Struct Reference

Queue element for PD packets to send or receive.

```
#include <trdp_private.h>
```

Collaboration diagram for PD_ELE:

Data Fields

- struct **PD_ELE** * **pNext**
pointer to next element or NULL
- UINT32 **magic**
prevent acces through dangeling pointer
- **TRDP_ADDRESSES_T** **addr**
handle of publisher/subscriber
- **TRDP_IP_ADDR_T** **lastSrcIP**
last source IP a subscribed packet was received from
- **TRDP_IP_ADDR_T** **pullIpAddress**
In case of pulling a PD this is the requested Ip.
- UINT32 **redId**
Redundancy group ID or zero.
- UINT32 **curSeqCnt**
the last sent or received sequence counter
- UINT32 **curSeqCnt4Pull**
the last sent sequence counter for PULL
- **TRDP_SEQ_CNT_LIST_T** * **pSeqCntList**
pointer to list of received sequence numbers per comId
- UINT32 **numRxTx**
Counter for received packets (statistics)
- UINT32 **updPkts**
Counter for updated packets (statistics)
- UINT32 **getPkts**
Counter for read packets (statistics)
- **TRDP_ERR_T** **lastErr**
Last error (timeout)
- **TRDP_PRIV_FLAGS_T** **privFlags**
private flags
- **TRDP_FLAGS_T** **pktFlags**
flags
- **TRDP_TIME_T** **interval**
time out value for received packets or interval for packets to send (set from ms)
- **TRDP_TIME_T** **timeToGo**
next time this packet must be sent/rcv

- **TRDP_TO_BEHAVIOR_T toBehavior**
timeout behavior for packets
- **UINT32 dataSize**
net data size
- **UINT32 grossSize**
complete packet size (header, data)
- **UINT32 sendSize**
data size sent out
- **TRDP_DATASET_T * pCachedDS**
Pointer to dataset element if known.
- **INT32 socketIdx**
index into the socket list
- **const void * pUserRef**
from subscribe()
- **TRDP_PD_CALLBACK_T pCbFunction**
Pointer to PD callback function.
- **PD_PACKET_T * pFrame**
header ...

4.2.1 Detailed Description

Queue element for PD packets to send or receive.

4.2.2 Field Documentation

4.2.2.1 PD_PACKET_T* PD_ELE::pFrame

header ...

data + FCS...

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

- **trdp_private.h**

4.3 TAU_MARSHALL_INFO_T Struct Reference

Marshalling info, used to and from wire.

Data Fields

- **INT32 level**
track recursive level
- **UINT8 * pSrc**
source pointer
- **UINT8 * pDst**
destination pointer
- **UINT8 * pDstEnd**
last destination

4.3.1 Detailed Description

Marshalling info, used to and from wire.

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

- **tau_marshall.c**

4.4 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)

4.4.1 Detailed Description

Closed train consists information.

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

- **tau_tti_types.h**

4.5 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

4.5.1 Detailed Description

ComId - data set mapping element definition.

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

- **trdp_types.h**

4.6 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 0 = (single) consist 1 = closed train 2 = 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 * pEtblInfoList**
ETB information list for the consist Ordered list starting with lowest etbld.
- **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: 'FFFFFFF'H

4.6.1 Detailed Description

consist information structure

4.6.2 Field Documentation

4.6.2.1 TRDP_LABEL_T TRDP_CONSIST_INFO_T::cstId

application defined consist identifier, e.g.

UIC identifier

4.6.2.2 TRDP_LABEL_T TRDP_CONSIST_INFO_T::cstOwner

consist owner, e.g.

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

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

- **tau_tti_types.h**

4.7 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.

4.7.1 Detailed Description

Dataset definition.

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

- **trdp_types.h**

4.8 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)
- **struct TRDP_DATASET * pCachedDS**
Used internally for marshalling speed-up.

4.8.1 Detailed Description

Dataset element definition.

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

- **trdp_types.h**

4.9 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.

4.9.1 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**

4.10 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 cnld 0..15 acc.

- **UINT16 reserved01**

reserved for future use (= 0)

4.10.1 Detailed Description

Types for train configuration information.

ETB information

4.10.2 Field Documentation

4.10.2.1 UINT8 TRDP_ETB_INFO_T::cnCnt

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

IEC61375-2-5

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

- **tau_tti_types.h**

4.11 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 cnld**

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

- **UINT8 reserved02**

reserved for future use (= 0)

4.11.1 Detailed Description

function/device information structure

4.11.2 Field Documentation

4.11.2.1 UINT8 TRDP_FUNCTION_INFO_T::cnld

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

Value range: 0..31

4.11.2.2 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

4.11.2.3 UINT8 TRDP_FUNCTION_INFO_T::etbld

number of connected train backbone.

Value range: 0..3

4.11.2.4 UINT16 TRDP_FUNCTION_INFO_T::fctld

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**

4.12 TRDP_HANDLE Struct Reference

Hidden handle definition, used as unique addressing item.

```
#include <trdp_private.h>
```

Data Fields

- **UINT32 comld**
comld for packets to send/receive
- **TRDP_IP_ADDR_T srcIpAddr**
source IP for PD
- **TRDP_IP_ADDR_T destIpAddr**
destination IP for PD
- **TRDP_IP_ADDR_T mcGroup**
multicast group to join for PD
- **UINT32 etbTopoCnt**
etb topocount belongs to addressing item
- **UINT32 opTrnTopoCnt**
opTrn topocount belongs to addressing item

4.12.1 Detailed Description

Hidden handle definition, used as unique addressing item.

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

- **trdp_private.h**

4.13 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.

4.13.1 Detailed Description

Information about a particular MD listener.

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

- **trdp_types.h**

4.14 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.

4.14.1 Detailed Description

Marshaling/unmarshalling configuration.

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

- **trdp_types.h**

4.15 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.

4.15.1 Detailed Description

Default MD configuration.

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

- **trdp_types.h**

4.16 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 destURI**
destination URI user part from MD header
- **TRDP_URI_USER_T srcURI**
source URI user part from MD header
- **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

4.16.1 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**

4.17 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

4.17.1 Detailed Description

Structure containing all general MD statistics information.

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

- **trdp_types.h**

4.18 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

4.18.1 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**

4.19 TRDP_MEM_STATISTICS_T Struct Reference

TRDP statistics type definitions.

```
#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

4.19.1 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?, timeout, 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 all general memory statistics information.

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

- **trdp_types.h**

4.20 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.

4.20.1 Detailed Description

Default PD configuration.

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

- **trdp_types.h**

4.21 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

4.21.1 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**

4.22 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

4.22.1 Detailed Description

Structure containing all general PD statistics information.

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

- **trdp_types.h**

4.23 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.

4.23.1 Detailed Description

Various flags/general TRDP options for library initialization.

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

- **trdp_types.h**

4.24 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

4.24.1 Detailed Description

Application defined properties.

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

- **tau_tti_types.h**

4.25 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.

4.25.1 Detailed Description

Table containing particular PD publishing information.

4.25.2 Field Documentation

4.25.2.1 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**

4.26 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.

4.26.1 Detailed Description

A table containing PD redundant group information.

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

- **trdp_types.h**

4.27 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**
Timeout cycles.
- UINT8 **reserved1**
Reserved for future use.
- UINT16 **reserved2**
Reserved for future use.

4.27.1 Detailed Description

Types to read out the XML configuration.

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

- **tau_xml.h**

4.28 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)

4.28.1 Detailed Description

Quality/type of service and time to live.

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

- **trdp_types.h**

4.29 TRDP_SEQ_CNT_ENTRY_T Struct Reference

Tuples of last received sequence counter per comId.

```
#include <trdp_private.h>
```

Data Fields

- **UINT32 lastSeqCnt**
Sequence counter value for comId.
- **TRDP_IP_ADDR_T srcIpAddr**
Source IP address.
- **TRDP_MSG_T msgType**
message type

4.29.1 Detailed Description

Tuples of last received sequence counter per comId.

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

- **trdp_private.h**

4.30 TRDP_SESSION Struct Reference

Session/application variables store.

```
#include <trdp_private.h>
```

Collaboration diagram for TRDP_SESSION:

Data Fields

- struct **TRDP_SESSION * pNext**
Pointer to next session.
- **VOS_MUTEX_T mutex**
protect this session
- **TRDP_IP_ADDR_T realIP**
Real IP address.
- **TRDP_IP_ADDR_T virtualIP**
Virtual IP address.
- **BOOL8 beQuiet**
if set, only react on ownIP requests
- **UINT32 redID**
redundant comId
- **UINT32 etbTopoCnt**
current valid topocount or zero
- **UINT32 opTrnTopoCnt**
current valid topocount or zero
- **TRDP_TIME_T nextJob**
Store for next select interval.
- **TRDP_PRINT_DBG_T pPrintDebugString**

- Pointer to function to print debug information.*
- **TRDP_MARSHALL_CONFIG_T** **marshall**
Marshalling(unMarshalling) configuration.
- **TRDP_PD_CONFIG_T** **pdDefault**
Default configuration for process data.
- **TRDP_MEM_CONFIG_T** **memConfig**
Internal memory handling configuration.
- **TRDP_OPTION_T** **option**
Stack behavior options.
- **TRDP_SOCKETS_T** **iface** [**VOS_MAX_SOCKET_CNT**]
Collection of sockets to use.
- **PD_ELE_T** * **pSndQueue**
pointer to first element of send queue
- **PD_ELE_T** * **pRcvQueue**
pointer to first element of rcv queue
- **TRDP_TIME_T** **initTime**
initialization time of session
- **TRDP_STATISTICS_T** **stats**
statistics of this session

4.30.1 Detailed Description

Session/application variables store.

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

- **trdp_private.h**

4.31 TRDP_SOCKET_TCP Struct Reference

TCP parameters.

```
#include <trdp_private.h>
```

Collaboration diagram for TRDP_SOCKET_TCP:

Data Fields

- **TRDP_IP_ADDR_T** **cornerIp**
The other TCP corner Ip.
- **BOOL8** **notSend**
If the message has been sent uncompleted.
- **TRDP_TIME_T** **connectionTimeout**
TCP socket connection Timeout.
- **BOOL8** **sendNotOk**
The sending timeout will be start.
- **TRDP_TIME_T** **sendingTimeout**
The timeout sending the message.
- **BOOL8** **addFileDesc**
Ready to add the socket in the fd.
- **BOOL8** **morituri**
about to die

4.31.1 Detailed Description

TCP parameters.

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

- **trdp_private.h**

4.32 TRDP_SOCKETS Struct Reference

Socket item.

```
#include <trdp_private.h>
```

Collaboration diagram for TRDP_SOCKETS:

Data Fields

- **INT32 sock**
vos socket descriptor to use
- **TRDP_IP_ADDR_T bindAddr**
Defines the interface to use.
- **TRDP_SEND_PARAM_T sendParam**
Send parameters.
- **TRDP SOCK_TYPE_T type**
Usage of this socket.
- **BOOL8 rcvMostly**
Used for receiving.
- **INT16 usage**
No.
- **TRDP_SOCKET_TCP_T tcpParams**
Params used for TCP.
- **TRDP_IP_ADDR_T mcGroups [VOS_MAX_MULTICAST_CNT]**
List of multicast addresses for this socket.

4.32.1 Detailed Description

Socket item.

4.32.2 Field Documentation

4.32.2.1 INT16 TRDP_SOCKETS::usage

No.

of current users of this socket

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

- **trdp_private.h**

4.33 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.

4.33.1 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**

4.34 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.

4.34.1 Detailed Description

Table containing particular PD subscription information.

4.34.2 Field Documentation

4.34.2.1 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.

4.34.2.2 UINT32 TRDP_SUBS_STATISTICS_T::numRecv

Number of packets received for this subscription.

4.34.2.3 UINT32 TRDP_SUBS_STATISTICS_T::timeout

Time-out value in us.

0 = No time-out supervision

4.34.2.4 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**

4.35 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

4.35.1 Detailed Description

vehicle information structure

4.35.2 Field Documentation

4.35.2.1 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**

4.36 TRDP_VERSION_T Struct Reference

Version information.

```
#include <trdp_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.

4.36.1 Detailed Description

Version information.

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

- **trdp_types.h**

4.37 TRDP_XML_DOC_HANDLE_T Struct Reference

Parsed XML document handle.

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

Data Fields

- **void * pXmlDocument**
Pointer to parsed XML document.
- **void * pRootElement**
Pointer to the document root element.
- **void * pXPathContext**
Pointer to prepared XPath context.

4.37.1 Detailed Description

Parsed XML document handle.

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

- **tau_xml.h**

4.38 VOS_SOCKET_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

4.38.1 Detailed Description

Common socket options.

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

- **vos_sock.h**

4.39 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.

4.39.1 Detailed Description

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

4.39.2 Field Documentation

4.39.2.1 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`

Chapter 5

File Documentation

5.1 tau_ctrl.c File Reference

Functions for train switch control.

```
#include <string.h>
#include <stdio.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "trdp_if_light.h"
#include "trdp_proto.h"
#include "tau_ctrl.h"
```

Include dependency graph for tau_ctrl.c:

Functions

- EXT_DECL **TRDP_ERR_T** **tau_initEcspCtrl** (**TRDP_APP_SESSION_T** appHandle, **TRDP_IP_ADDR_T** ecscIpAddr, **TRDP_IP_ADDR_T** ecscIpAddr)
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** *pEcspConfRequest)
Function for ECSP confirmation/correction request, reply will be received via call back.

5.1.1 Detailed Description

Functions for train switch 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:

tau_ctrl.c (p. 47) 1354 2014-11-11 15:22:13Z ahweiss

5.1.2 Function Documentation

5.1.2.1 `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	<i>appHandle</i>	Application handle
in, out	<i>pEcspStat</i>	Pointer to the ECSP status structure
in, out	<i>pPdInfo</i>	Pointer to PD status information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

5.1.2.2 `EXT_DECL TRDP_ERR_T tau_initEcspCtrl (TRDP_APP_SESSION_T appHandle, TRDP_IP_ADDR_T ecsplpAddr, TRDP_IP_ADDR_T ecscIpAddr)`

Function to init ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>ecsplpAddr</i>	ECSP address
in	<i>ecscIpAddr</i>	ECSC address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

Here is the call graph for this function:

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

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

Parameters

in	<i>appHandle</i>	Application Handle
in	<i>pUserRef</i>	user reference returned with reply
in	<i>pCbFunction</i>	Pointer to callback function, NULL for default
in	<i>pEcspConf-Request</i>	Pointer to confirmation data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

5.1.2.4 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	<i>appHandle</i>	Application handle
in	<i>pEcspCtrl</i>	Pointer to the ECSP control structure

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

5.1.2.5 EXT_DECL TRDP_ERR_T tau_terminateEcspCtrl (TRDP_APP_SESSION_T *appHandle*)

Function to close ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
----	------------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_UNKNOWN_ERR</i>	undefined error

Here is the call graph for this function:

5.2 tau_ctrl.h File Reference

TRDP utility interface definitions.

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

Include dependency graph for tau_ctrl.h: This graph shows which files directly or indirectly include this file:

Functions

- EXT_DECL **TRDP_ERR_T** **tau_initEcspCtrl** (**TRDP_APP_SESSION_T** appHandle, **TRDP_IP_ADDR_T** ecspIpAddr, **TRDP_IP_ADDR_T** ecscIpAddr)
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** *pEcspConfRequest)
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:

tau_ctrl.h (p. 49) 1349 2014-11-04 11:40:43Z ahweiss

5.2.2 Function Documentation

- 5.2.2.1 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	<i>appHandle</i>	Application Handle
in, out	<i>pEcspStat</i>	Pointer to the ECSP status structure
in, out	<i>pPdInfo</i>	Pointer to PD status information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Parameters

in	<i>appHandle</i>	Application handle
in, out	<i>pEcspStat</i>	Pointer to the ECSP status structure
in, out	<i>pPdInfo</i>	Pointer to PD status information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

5.2.2.2 EXT_DECL TRDP_ERR_T tau_initEcspCtrl (TRDP_APP_SESSION_T *appHandle*, TRDP_IP_ADDR_T *ecsplpAddr*, TRDP_IP_ADDR_T *ecscIpAddr*)

Function to init ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
in	<i>ecsplpAddr</i>	ECSP address
in	<i>ecscIpAddr</i>	ECSC address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

Here is the call graph for this function:

5.2.2.3 EXT_DECL TRDP_ERR_T tau_requestEcspConfirm (TRDP_APP_SESSION_T *appHandle*, const void * *pUserRef*, TRDP_MD_CALLBACK_T *pCbFunction*, TRDP_ECSP_CONF_REQUEST_T * *pEcspConfRequest*)

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

Parameters

in	<i>appHandle</i>	Application Handle
in	<i>pUserRef</i>	user reference returned with reply
in	<i>pCbFunction</i>	Pointer to callback function, NULL for default
in	<i>pEcspConfRequest</i>	Pointer to confirmation data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

5.2.2.4 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	<i>appHandle</i>	Application handle
in	<i>pEcspCtrl</i>	Pointer to the ECSP control structure

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

5.2.2.5 EXT_DECL TRDP_ERR_T tau_terminateEcspCtrl (TRDP_APP_SESSION_T *appHandle*)

Function to close ECSP control interface.

Parameters

in	<i>appHandle</i>	Application handle
----	------------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_UNKNOWN_ERR</i>	undefined error

Parameters

in	<i>appHandle</i>	Application handle
----	------------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	module not initialised
<i>TRDP_UNKNOWN_ERR</i>	undefined error

Here is the call graph for this function:

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

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:

tau_ctrl_types.h (p. 52) 1349 2014-11-04 11:40:43Z ahweiss

5.4 tau_dnr.c File Reference

Functions for domain name resolution.

```
#include <string.h>
#include <stdio.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "tau_dnr.h"
Include dependency graph for tau_dnr.c:
```

5.4.1 Detailed Description

Functions for domain name resolution.

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:

tau_dnr.c (p. 53) 1279 2014-08-07 06:18:07Z ahweiss

5.5 tau_dnr.h File Reference

TRDP utility interface definitions.

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

Include dependency graph for tau_dnr.h: This graph shows which files directly or indirectly include this file:

Functions

- **EXT_DECL TRDP_ERR_T tau_initDnr** (void)
Function to init DNR.
- **EXT_DECL TRDP_ERR_T tau_getOwnIds** (TRDP_LABEL_T devId, TRDP_LABEL_T vehId, TRDP_LABEL_T cstId)
Who am I ?.
- **EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr** (void)
Function to get the own IP address.
- **EXT_DECL TRDP_ERR_T tau_uri2Addr** (TRDP_IP_ADDR_T *pAddr, UINT32 *pTopoCnt, const TRDP_URI_T uri)
Function to convert a URI to an IP address.
- **EXT_DECL TRDP_ERR_T tau_addr2Uri** (TRDP_URI_HOST_T uri, UINT32 *pTopoCnt, **TRDP_IP_ADDR_T** addr)
Function to convert an IP address to a URI.
- **EXT_DECL TRDP_ERR_T tau_label2VehId** (TRDP_LABEL_T vehId, UINT32 *pTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel)
Function to retrieve the vehId of the car with label vehLabel in the consist with cstLabel.
- **EXT_DECL TRDP_ERR_T tau_label2TcnVehNo** (UINT8 *pTcnVehNo, UINT32 *pTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel)
Function The function delivers the TCN vehicle number to the given label.
- **EXT_DECL TRDP_ERR_T tau_label2OpVehNo** (UINT8 *pOpVehNo, UINT32 *pTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel)
Function The function delivers the operational vehicle sequence number to the given label.

- EXT_DECL **TRDP_ERR_T tau_tcnVehNo2Ids** (TRDP_LABEL_T vehId, TRDP_LABEL_T cstId, UINT32 *pTopoCnt, UINT8 tcnVehNo, UINT8 tcnCstNo)
Function to retrieve the car and consist id of the car given with carNo and trnCstNo.
- EXT_DECL **TRDP_ERR_T tau_opVehNo2Ids** (TRDP_LABEL_T vehId, TRDP_LABEL_T cstId, UINT32 *pTopoCnt, UINT8 opVehNo)
Function to retrieve the vehicle and consist id from a given operational vehicle sequence number.
- EXT_DECL **TRDP_ERR_T tau_addr2VehId** (TRDP_LABEL_T vehId, UINT32 *pTopoCnt, **TRDP_IP_ADDR_T** ipAddr)
Function to retrieve the vehId of the car hosting a device with the IPAddress ipAddr.
- EXT_DECL **TRDP_ERR_T tau_addr2TcnVehNo** (UINT8 *pTcnVehNo, UINT8 *pTopoCnt, **TRDP_IP_ADDR_T** ipAddr)
Function to retrieve the TCN vehicle number in consist of the car hosting the device with the given IP address.
- EXT_DECL **TRDP_ERR_T tau_addr2OpVehNo** (UINT8 *pOpVehNo, UINT8 *pTopoCnt, **TRDP_IP_ADDR_T** ipAddr)
Function to retrieve the operational vehicle number of the vehicle hosting the device with the given IP address.
- EXT_DECL **TRDP_ERR_T tau_tcnCstNo2CstId** (TRDP_LABEL_T cstId, UINT32 *pTopoCnt, UINT8 tcnCstNo)
Function to retrieve the consist identifier of the consist with train consist sequence number cstNo.
- EXT_DECL **TRDP_ERR_T tau_iecCstNo2CstId** (TRDP_LABEL_T cstId, UINT32 *pTopoCnt, UINT8 opCstNo)
Function to retrieve the consist identifier of the consist with IEC sequence consist number iecCstNo.
- EXT_DECL **TRDP_ERR_T tau_label2CstId** (TRDP_LABEL_T cstId, UINT32 *pTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel)
Function to retrieve the consist identifier of the consist hosting a car with label vehLabel.
- EXT_DECL **TRDP_ERR_T tau_label2TcnCstNo** (UINT8 *pTcnCstNo, UINT32 *pTopoCnt, const TRDP_LABEL_T vehLabel)
Function to retrieve the TCN consist sequence number of the consist hosting a vehicle with label vehLabel.
- EXT_DECL **TRDP_ERR_T tau_label2OpCstNo** (UINT8 *pOpCstNo, UINT32 *pTopoCnt, const TRDP_LABEL_T vehLabel)
Function to retrieve the operational consist sequence number of the consist hosting a vehicle with label vehLabel.
- EXT_DECL **TRDP_ERR_T tau_addr2CstId** (TRDP_LABEL_T cstId, UINT32 *pTopoCnt, **TRDP_IP_ADDR_T** ipAddr)
Function to retrieve the consist identifier of the consist hosting the device with the IP-Address ipAddr.
- EXT_DECL **TRDP_ERR_T tau_addr2TcnCstNo** (UINT8 *pTcnCstNo, UINT32 *pTopoCnt, **TRDP_IP_ADDR_T** ipAddr)
Function to retrieve the TCN consist number of the consist hosting the device with the IP-Address ipAddr.
- EXT_DECL **TRDP_ERR_T tau_addr2OpCstNo** (UINT8 *pOpCstNo, UINT32 *pTopoCnt, **TRDP_IP_ADDR_T** ipAddr)
Function to retrieve the operational consist number of the consist hosting the device with the IP-Address ipAddr.

5.5.1 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:

tau_dnr.h (p. 54) 1323 2014-08-29 14:09:08Z bloehr

5.5.2 Function Documentation

5.5.2.1 EXT_DECL TRDP_ERR_T tau_addr2CstId (TRDP_LABEL_T *cstId*, UINT32 * *pTopoCnt*, TRDP_IP_ADDR_T *ipAddr*)

Function to retrieve the consist identifier of the consist hosting the device with the IP-Address *ipAddr*.

Parameters

out	<i>cstId</i>	Pointer to the consist id to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>ipAddr</i>	IP address. 0 means own device, so the own consist id is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.2 EXT_DECL TRDP_ERR_T tau_addr2OpCstNo (UINT8 * *pOpCstNo*, UINT32 * *pTopoCnt*, TRDP_IP_ADDR_T *ipAddr*)

Function to retrieve the operational consist number of the consist hosting the device with the IP-Address *ipAddr*.

Parameters

out	<i>pOpCstNo</i>	Pointer to the operational consist number to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>ipAddr</i>	IP address. 0 means own device, so the own IEC consist number is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.3 EXT_DECL TRDP_ERR_T tau_addr2OpVehNo (UINT8 * *pOpVehNo*, UINT8 * *pTopoCnt*, TRDP_IP_ADDR_T *ipAddr*)

Function to retrieve the operational vehicle number of the vehicle hosting the device with the given IP address.

Parameters

out	<i>pOpVehNo</i>	Pointer to the operational vehicle number to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>ipAddr</i>	IP address. 0 means own address, so the own operational vehicle number is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.4 EXT_DECL TRDP_ERR_T tau_addr2TcnCstNo (UINT8 * *pTcnCstNo*, UINT32 * *pTopoCnt*, TRDP_IP_ADDR_T *ipAddr*)

Function to retrieve the TCN consist number of the consist hosting the device with the IP-Address *ipAddr*.

Parameters

out	<i>pTcnCstNo</i>	Pointer to the TCN consist number to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>ipAddr</i>	IP address. 0 means own device, so the own consist number is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.5 EXT_DECL TRDP_ERR_T tau_addr2TcnVehNo (UINT8 * *pTcnVehNo*, UINT8 * *pTopoCnt*, TRDP_IP_ADDR_T *ipAddr*)

Function to retrieve the TCN vehicle number in consist of the car hosting the device with the given IP address.

Parameters

out	<i>pTcnVehNo</i>	Pointer to the TCN vehicle number in consist to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>ipAddr</i>	IP address. 0 means own address, so the own vehicle number is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.6 EXT_DECL TRDP_ERR_T tau_addr2Uri (TRDP_URI_HOST_T *uri*, UINT32 * *pTopoCnt*, 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

out	<i>uri</i>	Pointer to a string to return the URI host part
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>addr</i>	IP address, 0==own address

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.7 EXT_DECL TRDP_ERR_T tau_addr2VehId (TRDP_LABEL_T *vehId*, UINT32 * *pTopoCnt*, TRDP_IP_ADDR_T *ipAddr*)

Function to retrieve the vehId of the car hosting a device with the IPAddress ipAddr.

Parameters

out	<i>vehId</i>	Pointer to the vehicle id to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>ipAddr</i>	IP address. 0 means own address, so the own vehicle id is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.8 EXT_DECL TRDP_IP_ADDR_T tau_getOwnAddr (void)

Function to get the own IP address.

Return values

<i>own</i>	IP address
------------	------------

5.5.2.9 EXT_DECL TRDP_ERR_T tau_getOwnIds (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

out	<i>devId</i>	Returns the device label (host name)
out	<i>vehId</i>	Returns the vehicle label
out	<i>cstId</i>	Returns the consist label

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.10 EXT_DECL TRDP_ERR_T tau_iecCstNo2CstId (TRDP_LABEL_T cstId, UINT32 * pTopoCnt, UINT8 opCstNo)

Function to retrieve the consist identifier of the consist with IEC sequence consist number iecCstNo.

Parameters

out	<i>cstId</i>	Pointer to the consist id to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>opCstNo</i>	Operational consist sequence number based on the leading car. 0 means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.11 EXT_DECL TRDP_ERR_T tau_initDnr (void)

Function to init DNR.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

5.5.2.12 EXT_DECL TRDP_ERR_T tau_label2CstId (TRDP_LABEL_T *cstId*, UINT32 * *pTopoCnt*, const TRDP_LABEL_T *vehLabel*, const TRDP_LABEL_T *cstLabel*)

Function to retrieve the consist identifier of the consist hosting a car with label *vehLabel*.

Parameters

out	<i>cstId</i>	Pointer to the consist id to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to a vehicle label. NULL means any car.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.13 EXT_DECL TRDP_ERR_T tau_label2OpCstNo (UINT8 * *pOpCstNo*, UINT32 * *pTopoCnt*, const TRDP_LABEL_T *vehLabel*)

Function to retrieve the operational consist sequence number of the consist hosting a vehicle with label *vehLabel*.

Parameters

out	<i>pOpCstNo</i>	Pointer to the operational consist number to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to a vehicle label. NULL means own vehicle, so the own IEC consist number is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.14 EXT_DECL TRDP_ERR_T tau_label2OpVehNo (UINT8 * *pOpVehNo*, UINT32 * *pTopoCnt*, const TRDP_LABEL_T *vehLabel*, const TRDP_LABEL_T *cstLabel*)

Function The function delivers the operational vehicle sequence number to the given label.

The first match of the table will be returned in case there is no unique label given.

Parameters

out	<i>pOpVehNo</i>	Pointer to the operational vehicle sequence number to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to a vehicle label. NULL means own vehicle.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.15 EXT_DECL TRDP_ERR_T tau_label2TcnCstNo (UINT8 * *pTcnCstNo*, UINT32 * *pTopoCnt*, const TRDP_LABEL_T *vehLabel*)

Function to retrieve the TCN consist sequence number of the consist hosting a vehicle with label *vehLabel*.

Parameters

out	<i>pTcnCstNo</i>	Pointer to the TCN consist number to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to a vehicle label, NULL means own vehicle, so the own consist number is returned.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.16 EXT_DECL TRDP_ERR_T tau_label2TcnVehNo (UINT8 * *pTcnVehNo*, UINT32 * *pTopoCnt*, const TRDP_LABEL_T *vehLabel*, const TRDP_LABEL_T *cstLabel*)

Function The function delivers the TCN vehicle number to the given label.

The first match of the table will be returned in case there is no unique label given.

Parameters

out	<i>pTcnVehNo</i>	Pointer to the TCN vehicle number to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to the vehicle label. NULL means own vehicle.
in	<i>cstLabel</i>	Pointer to the consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.17 EXT_DECL TRDP_ERR_T tau_label2VehId (TRDP_LABEL_T *vehId*, UINT32 * *pTopoCnt*, const TRDP_LABEL_T *vehLabel*, const TRDP_LABEL_T *cstLabel*)

Function to retrieve the vehId of the car with label vehLabel in the consist with cstLabel.

Parameters

out	<i>vehId</i>	Pointer to a label string to return the vehicle id
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to the vehicle label. NULL means own vehicle if cstLabel == NULL.
in	<i>cstLabel</i>	Pointer to the consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.18 EXT_DECL TRDP_ERR_T tau_opVehNo2Ids (TRDP_LABEL_T *vehId*, TRDP_LABEL_T *cstId*, UINT32 * *pTopoCnt*, UINT8 *opVehNo*)

Function to retrieve the vehicle and consist id from a given operational vehicle sequence number.

Parameters

out	<i>vehId</i>	Pointer to the vehicle id to be returned
out	<i>cstId</i>	Pointer to the consist id to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>opVehNo</i>	Operational vehicle sequence number. 0 means own vehicle.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.19 EXT_DECL TRDP_ERR_T tau_tcnCstNo2CstId (TRDP_LABEL_T *cstId*, UINT32 * *pTopoCnt*, UINT8 *tcnCstNo*)

Function to retrieve the consist identifier of the consist with train consist sequence number *cstNo*.

Parameters

out	<i>cstId</i>	Pointer to the consist id to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>tcnCstNo</i>	Consist sequence number based on IP reference direction. 0 means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.20 EXT_DECL TRDP_ERR_T tau_tcnVehNo2Ids (TRDP_LABEL_T *vehId*, TRDP_LABEL_T *cstId*, UINT32 * *pTopoCnt*, UINT8 *tcnVehNo*, UINT8 *tcnCstNo*)

Function to retrieve the car and consist id of the car given with *carNo* and *trnCstNo*.

Parameters

out	<i>vehId</i>	Pointer to the vehicle id to be returned
out	<i>cstId</i>	Pointer to the consist id to be returned
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>tcnVehNo</i>	Vehicle number in consist. 0 means own vehicle when <i>trnCstNo</i> == 0.
in	<i>tcnCstNo</i>	TCN consist sequence number in train. 0 means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.5.2.21 EXT_DECL TRDP_ERR_T tau_uri2Addr (TRDP_IP_ADDR_T * *pAddr*, UINT32 * *pTopoCnt*, const TRDP_URI_T *uri*)

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

out	<i>pAddr</i>	Pointer to return the IP address
in, out	<i>pTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>uri</i>	Pointer to a URI or an IP Address string, NULL==own URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.6 tau_marshall.c File Reference

Marshalling functions for TRDP.

```
#include <string.h>
#include "trdp_types.h"
#include "trdp_if_light.h"
#include "trdp_utils.h"
#include "vos_mem.h"
#include "tau_marshall.h"
Include dependency graph for tau_marshall.c:
```

Data Structures

- struct **TAU_MARSHALL_INFO_T**
Marshalling info, used to and from wire.

Functions

- EXT_DECL **TRDP_ERR_T tau_initMarshall** (void **ppRefCon, UINT32 numComId, **TRDP_COMID_DSID_MAP_T** *pComIdDsIdMap, UINT32 numDataSet, **TRDP_DATASET_T** *pDataset[])
Function to initialise the marshalling/unmarshalling.
- EXT_DECL **TRDP_ERR_T tau_marshall** (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT8 *pDest, UINT32 *pDestSize, **TRDP_DATASET_T** **ppDSPointer)
marshall function.
- EXT_DECL **TRDP_ERR_T tau_unmarshall** (void *pRefCon, UINT32 comId, UINT8 *pSrc, UINT8 *pDest, UINT32 *pDestSize, **TRDP_DATASET_T** **ppDSPointer)
unmarshall function.
- EXT_DECL **TRDP_ERR_T tau_marshallDs** (void *pRefCon, UINT32 dsId, UINT8 *pSrc, UINT8 *pDest, UINT32 *pDestSize, **TRDP_DATASET_T** **ppDSPointer)
marshall data set function.
- EXT_DECL **TRDP_ERR_T tau_unmarshallDs** (void *pRefCon, UINT32 dsId, UINT8 *pSrc, 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 *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, UINT32 *pDestSize, **TRDP_DATASET_T** **ppDSPointer)
Calculate data set size by given ComId.

5.6.1 Detailed Description

Marshalling functions for TRDP.

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, 2013. All rights reserved.

Id:

tau_marshall.c (p. 62) 1190 2014-03-12 13:15:17Z ahweiss

5.6.2 Function Documentation

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

Calculate data set size by given data set id.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_PARAM_ERR</i>	data set id not existing

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

Calculate data set size by given ComId.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_PARAM_ERR</i>	data set id not existing

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

Function to initialise the marshalling/unmarshalling.

Types for marshalling / unmarshalling.

The supplied array must be sorted by Comlds. The array must exist during the use of the marshalling functions (until **tlc_terminate()** (p. 110)).

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

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

marshall function.

Parameters

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

Return values

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

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

marshall data set function.

Parameters

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

Return values

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

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

unmarshall function.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing

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

unmarshall data set function.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing

5.7 tau_marshall.h File Reference

TRDP utility interface definitions.

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

Include dependency graph for tau_marshall.h: This graph shows which files directly or indirectly include this file:

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, UINT8 *pDest, UINT32 *pDestSize, **TRDP_DATASET_T** **ppDSPointer)
marshall function.
- EXT_DECL **TRDP_ERR_T tau_marshallDs** (void *pRefCon, UINT32 dsId, UINT8 *pSrc, 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, UINT8 *pDest, UINT32 *pDestSize, **TRDP_DATASET_T** **ppDSPointer)
unmarshall function.
- EXT_DECL **TRDP_ERR_T tau_unmarshallDs** (void *pRefCon, UINT32 dsId, UINT8 *pSrc, 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 *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, UINT32 *pDestSize, **TRDP_DATASET_T** **ppDSPointer)
Calculate data set size by given ComId.

5.7.1 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

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:

tau_marshall.h (p. 66) 1279 2014-08-07 06:18:07Z ahweiss

5.7.2 Function Documentation

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

Calculate data set size by given data set id.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_PARAM_ERR</i>	data set id not existing

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

Calculate data set size by given ComId.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_PARAM_ERR</i>	data set id not existing

5.7.2.3 `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	<i>ppRefCon</i>	Returns a pointer to be used for the reference context of marshalling/unmarshalling
in	<i>numComId</i>	Number of datasets found in the configuration
in	<i>pComIdDsidMap</i>	Pointer to an array of structures of type TRDP_DATASET_T
in	<i>numDataSet</i>	Number of datasets found in the configuration
in	<i>pDataset</i>	Pointer to an array of pointers to structures of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error

Types for marshalling / unmarshalling.

The supplied array must be sorted by ComIds. The array must exist during the use of the marshalling functions

(until **tlc_terminate()** (p. 110)).

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	Parameter error

Here is the call graph for this function:

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

marshall function.

Parameters

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

Return values

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

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

marshall data set function.

Parameters

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

Return values

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

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

unmarshall function.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing

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

unmarshall data set function.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_INIT_ERR</i>	marshalling not initialised
<i>TRDP_COMID_ERR</i>	comid not existing

5.8 tau_tti.c File Reference

Functions for train topology information access.

```
#include <string.h>
#include <stdio.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "tau_tti.h"
Include dependency graph for tau_tti.c:
```

5.8.1 Detailed Description

Functions for 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, 2013. All rights reserved.

Id:

tau_tti.c (p. 70) 1279 2014-08-07 06:18:07Z ahweiss

5.9 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** (void)
Function to init TTI access.
- **EXT_DECL TRDP_ERR_T tau_getOpTrDirectory** (TRDP_OP_TRAIN_DIR_STATE_T *pOpTrDirState, TRDP_OP_TRAIN_DIR_T *pOpTrDir, UINT8 const etbld)
Function to retrieve the operational train directory state.
- **EXT_DECL TRDP_ERR_T tau_getTrDirectory** (TRDP_TRAIN_DIR_T *pTrDir, UINT8 const etbld)
Function to retrieve the operational train directory.
- **EXT_DECL TRDP_ERR_T tau_getStaticCstInfo** (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_OP_TRAIN_DIR_STATE_T *pOpTrDirState, TRDP_OP_TRAIN_DIR_T *pOpTrDir, TRDP_TRAIN_DIR_T *pTrDir, TRDP_TRAIN_NET_DIR_T *pTrNetDir, UINT8 const etbld)
Function to retrieve the operational train directory.
- **EXT_DECL TRDP_ERR_T tau_getTrnCstCnt** (UINT16 *pTrnCstCnt, UINT32 *pOpTrTopoCnt)

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

- EXT_DECL **TRDP_ERR_T tau_getTrnCarCnt** (UINT16 *pTrnCarCnt, UINT32 *pOpTrTopoCnt)

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

- EXT_DECL **TRDP_ERR_T tau_getCstCarCnt** (UINT16 *pCstCarCnt, UINT32 *pOpTrTopoCnt, const TRDP_LABEL_T cstLabel)

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

- EXT_DECL **TRDP_ERR_T tau_getCstFctCnt** (UINT16 *pCstFctCnt, UINT32 *pOpTrTopoCnt, const TRDP_LABEL_T cstLabel)

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

- EXT_DECL **TRDP_ERR_T tau_getCarDevCnt** (UINT16 *pDevCnt, UINT32 *pOpTrTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel)

Function to retrieve the total number of devices in a car.

- EXT_DECL **TRDP_ERR_T tau_getCstFctInfo** (**TRDP_FUNCTION_INFO_T** *pFctInfo, UINT32 *pOpTrTopoCnt, const TRDP_LABEL_T cstLabel, UINT16 maxFctCnt)

Function to retrieve the function information of the consist.

- EXT_DECL **TRDP_ERR_T tau_getVehInfo** (**TRDP_VEHICLE_INFO_T** *pVehInfo, UINT32 *pOpTrTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel, UINT32 carPropLen)

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

- EXT_DECL **TRDP_ERR_T tau_getCstInfo** (**TRDP_CONSIST_INFO_T** *pCstInfo, UINT32 *pOpTrTopoCnt, const TRDP_LABEL_T cstLabel)

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

- EXT_DECL **TRDP_ERR_T tau_getVehOrient** (UINT8 *pCarOrient, UINT8 *pCstOrient, UINT32 *pOpTrTopoCnt, TRDP_LABEL_T vehLabel, TRDP_LABEL_T cstLabel)

Function to retrieve the orientation of the given vehicle.

- EXT_DECL **TRDP_ERR_T tau_getlecCarOrient** (UINT8 *plecCarOrient, UINT8 *plecCstOrient, UINT32 *pOpTrTopoCnt, TRDP_LABEL_T vehLabel, TRDP_LABEL_T cstLabel)

Function to retrieve the leading car depending IEC orientation of the given consist.

5.9.1 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:

tau_tti.h (p. 71) 1323 2014-08-29 14:09:08Z bloehr

5.9.2 Function Documentation

5.9.2.1 `EXT_DECL TRDP_ERR_T tau_getCarDevCnt (UINT16 * pDevCnt, UINT32 * pOpTrTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel)`

Function to retrieve the total number of devices in a car.

Parameters

out	<i>pDevCnt</i>	Pointer to the device count to be returned
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to a vehicle label. NULL means own vehicle if cstLabel == NULL.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.2 EXT_DECL TRDP_ERR_T tau_getCstCarCnt (UINT16 * *pCstCarCnt*, UINT32 * *pOpTrTopoCnt*, const TRDP_LABEL_T *cstLabel*)

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

Parameters

out	<i>pCstCarCnt</i>	Pointer to the number of cars to be returned
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.3 EXT_DECL TRDP_ERR_T tau_getCstFctCnt (UINT16 * *pCstFctCnt*, UINT32 * *pOpTrTopoCnt*, const TRDP_LABEL_T *cstLabel*)

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

Parameters

out	<i>pCstFctCnt</i>	Pointer to the number of functions to be returned
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.4 EXT_DECL TRDP_ERR_T tau_getCstFctInfo (TRDP_FUNCTION_INFO_T * *pFctInfo*, UINT32 * *pOpTrTopoCnt*, const TRDP_LABEL_T *cstLabel*, UINT16 *maxFctCnt*)

Function to retrieve the function information of the consist.

Parameters

out	<i>pFctInfo</i>	Pointer to function info list to be returned. Memory needs to be provided by application. Set NULL if not used.
-----	-----------------	---

in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.
in	<i>maxFctCnt</i>	Maximal number of functions to be returned in provided buffer.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.5 EXT_DECL TRDP_ERR_T tau_getCstInfo (TRDP_CONSIST_INFO_T * pCstInfo, UINT32 * pOpTrTopoCnt, const TRDP_LABEL_T cstLabel)

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

Parameters

out	<i>pCstInfo</i>	Pointer to the consist info to be returned.
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.6 EXT_DECL TRDP_ERR_T tau_getIecCarOrient (UINT8 * plecCarOrient, UINT8 * plecCstOrient, UINT32 * pOpTrTopoCnt, TRDP_LABEL_T vehLabel, TRDP_LABEL_T cstLabel)

Function to retrieve the leading car depending IEC orientation of the given consist.

Parameters

out	<i>plecCarOrient</i>	Pointer to the IEC car orientation to be returned
out	<i>plecCstOrient</i>	Pointer to the IEC consist orientation to be returned
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	vehLabel = NULL means own vehicle if cstLabel == NULL
in	<i>cstLabel</i>	cstLabel = NULL means own consist

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.7 EXT_DECL TRDP_ERR_T tau_getOpTrDirectory (TRDP_OP_TRAIN_DIR_STATE_T * pOpTrDirState, TRDP_OP_TRAIN_DIR_T * pOpTrDir, UINT8 const etbld)

Function to retrieve the operational train directory state.

Parameters

out	<i>pOpTrDirState</i>	Pointer to an operational train directory state structure to be returned.
out	<i>pOpTrDir</i>	Pointer to an operational train directory structure to be returned.
in	<i>etbld</i>	Identifier of the ETB the train directory state is asked for.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.8 EXT_DECL TRDP_ERR_T tau_getStaticCstInfo (TRDP_CONSIST_INFO_T * *pCstInfo*, TRDP_UUID_T const *cstUUID*)

Function to retrieve the operational train directory.

Parameters

out	<i>pCstInfo</i>	Pointer to a consist info structure to be returned.
in	<i>cstUUID</i>	UUID of the consist the consist info is requested for.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.9 EXT_DECL TRDP_ERR_T tau_getTrDirectory (TRDP_TRAIN_DIR_T * *pTrDir*, UINT8 const *etbld*)

Function to retrieve the operational train directory.

Parameters

out	<i>pTrDir</i>	Pointer to a train directory structure to be returned.
in	<i>etbld</i>	Identifier of the ETB the train directory is requested for.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.10 EXT_DECL TRDP_ERR_T tau_getTrnCarCnt (UINT16 * *pTrnCarCnt*, UINT32 * *pOpTrTopoCnt*)

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

Parameters

out	<i>pTrnCarCnt</i>	Pointer to the number of cars to be returned
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.11 EXT_DECL TRDP_ERR_T tau_getTrnCstCnt (UINT16 * *pTrnCstCnt*, UINT32 * *pOpTrTopoCnt*)

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

Parameters

out	<i>pTrnCstCnt</i>	Pointer to the number of consists to be returned
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.12 `EXT_DECL TRDP_ERR_T tau_getTTI (TRDP_OP_TRAIN_DIR_STATE_T * pOpTrDirState, TRDP_OP_TRAIN_DIR_T * pOpTrDir, TRDP_TRAIN_DIR_T * pTrDir, TRDP_TRAIN_NET_DIR_T * pTrNetDir, UINT8 const etbld)`

Function to retrieve the operational train directory.

Parameters

out	<i>pOpTrDirState</i>	Pointer to an operational train directory state structure to be returned.
out	<i>pOpTrDir</i>	Pointer to an operational train directory structure to be returned.
out	<i>pTrDir</i>	Pointer to a train directory structure to be returned.
out	<i>pTrNetDir</i>	Pointer to a train network directory structure to be returned.
in	<i>etbld</i>	Identifier of the ETB the train directory state is requested for.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.13 `EXT_DECL TRDP_ERR_T tau_getVehInfo (TRDP_VEHICLE_INFO_T * pVehInfo, UINT32 * pOpTrTopoCnt, const TRDP_LABEL_T vehLabel, const TRDP_LABEL_T cstLabel, UINT32 carPropLen)`

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

Parameters

out	<i>pVehInfo</i>	Pointer to the vehicle info to be returned.
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	Pointer to a vehicle label. NULL means own vehicle if cstLabel refers to own consist.
in	<i>cstLabel</i>	Pointer to a consist label. NULL means own consist.
in	<i>carPropLen</i>	Size of properties

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.14 `EXT_DECL TRDP_ERR_T tau_getVehOrient (UINT8 * pCarOrient, UINT8 * pCstOrient, UINT32 * pOpTrTopoCnt, TRDP_LABEL_T vehLabel, TRDP_LABEL_T cstLabel)`

Function to retrieve the orientation of the given vehicle.

Parameters

out	<i>pCarOrient</i>	Pointer to the vehicle orientation to be returned
-----	-------------------	---

out	<i>pCstOrient</i>	Pointer to the consist orientation to be returned
in, out	<i>pOpTrTopoCnt</i>	Pointer to the actual topo count. If !=0 will be checked. Returns the actual one.
in	<i>vehLabel</i>	vehLabel = NULL means own vehicle if cstLabel == NULL
in	<i>cstLabel</i>	cstLabel = NULL means own consist

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	Parameter error

5.9.2.15 EXT_DECL TRDP_ERR_T tau_initTtiAccess (void)

Function to init TTI access.

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	initialisation error

5.10 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.

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

5.10.1 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

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:

tau_tti_types.h (p. 78) 1335 2014-09-30 07:11:05Z ahweiss

5.11 tau_xml.c File Reference

Functions for XML file parsing.

```
#include <string.h>
#include <stdio.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "tau_xml.h"
#include "libxml/parser.h"
#include "libxml/xpath.h"
Include dependency graph for tau_xml.c:
```


Macros

- **#define TRDP_SDT_DEFAULT_SMI2 0**
Default SDT safe message identifier.
- **#define TRDP_SDT_DEFAULT_NRXSAFE 3**
Default SDT timeout cycles.
- **#define TRDP_SDT_DEFAULT_NGUARD 100**
Default SDT initial timeout cycles.
- **#define TRDP_SDT_DEFAULT_CMTHR 10**
Default SDT chan.

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 **pplfConfig)
Function to read the TRDP device configuration parameters out of the XML 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, TRDP_DATASET_T *pDataset)
Function to read the DataSet configuration out of the XML configuration file.
- **EXT_DECL TRDP_ERR_T tau_readXmlInterfaceConfig** (const TRDP_XML_DOC_HANDLE_T *pDocHnd, const CHAR8 *plfName, 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 void tau_freeTelegrams** (UINT32 numExchgPar, TRDP_EXCHG_PAR_T *pExchgPar)
Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

5.11.1 Detailed Description

Functions for XML file parsing.

Note

Project: TCNOpen TRDP prototype stack

Author

Tomas Svoboda, UniContorls a.s.

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:

tau_xml.c (p. 79) 1265 2014-07-14 16:11:53Z bloehr

5.11.2 Macro Definition Documentation

5.11.2.1 #define TRDP_SDT_DEFAULT_CMTHR 10

Default SDT chan.
monitoring threshold

5.11.3 Function Documentation

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

Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

Parameters

in	<i>numExchgPar</i>	Number of telegram configurations in the array
in	<i>pExchgPar</i>	Pointer to array of telegram configurations

Here is the call graph for this function:

5.11.3.2 EXT_DECL void tau_freeXmlDoc (TRDP_XML_DOC_HANDLE_T * *pDocHnd*)

Free all the memory allocated by tau_prepareXmlDoc.

Parameters

in	<i>pDocHnd</i>	Handle of the parsed XML file
----	----------------	-------------------------------

5.11.3.3 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	<i>pFileName</i>	Path and filename of the xml configuration file
out	<i>pDocHnd</i>	Handle of the parsed XML file

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	File does not exist

5.11.3.4 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	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumComId</i>	Pointer to the number of entries in the ComId DatasetId mapping list

out	<i>ppComIdDsId-Map</i>	Pointer to an array of a structures of type TRDP_COMID_DSID_MAP_T (p. 21)
out	<i>pNumDataset</i>	Pointer to the number of datasets found in the configuration
out	<i>papDataset</i>	Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.11.3.5 **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** ** *pplfConfig*)

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

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.11.3.6 **EXT_DECL TRDP_ERR_T** tau_readXmlInterfaceConfig (const **TRDP_XML_DOC_HANDLE_T** * *pDocHnd*, const **CHAR8** * *plfName*, **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	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
in	<i>plfName</i>	Interface name
out	<i>pProcessConfig</i>	TRDP process (session) configuration for the interface
out	<i>pPdConfig</i>	PD default configuration for the interface
out	<i>pMdConfig</i>	MD default configuration for the interface
out	<i>pNumExchgPar</i>	Number of configured telegrams
out	<i>ppExchgPar</i>	Pointer to array of telegram configurations

Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

Here is the call graph for this function:

5.12 tau_xml.h File Reference

TRDP utility interface definitions.

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

Include dependency graph for tau_xml.h: This graph shows which files directly or indirectly include this file:

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_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 *plfName, **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, **papT-RDP_DATASET_T** papDataset)
Function to read the DataSet configuration out of the XML configuration file.
- EXT_DECL void **tau_freeTelegrams** (UINT32 numExchgPar, **TRDP_EXCHG_PAR_T** *pExchgPar)
Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

5.12.1 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:

tau_xml.h (p. 83) 1298 2014-08-25 14:58:54Z bloehr

5.12.2 Enumeration Type Documentation

5.12.2.1 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)

5.12.3 Function Documentation

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

Free array of telegram configurations allocated by tau_readXmlInterfaceConfig.

Parameters

in	<i>numExchgPar</i>	Number of telegram configurations in the array
in	<i>pExchgPar</i>	Pointer to array of telegram configurations

Here is the call graph for this function:

5.12.3.2 EXT_DECL void tau_freeXmlDoc (TRDP_XML_DOC_HANDLE_T * *pDocHnd*)

Free all the memory allocated by tau_prepareXmlDoc.

Parameters

in	<i>pDocHnd</i>	Handle of the parsed XML file
----	----------------	-------------------------------

5.12.3.3 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	<i>pFileName</i>	Path and filename of the xml configuration file
out	<i>pDocHnd</i>	Handle of the parsed XML file

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	File does not exist

5.12.3.4 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	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
out	<i>pNumComId</i>	Pointer to the number of entries in the ComId DatasetId mapping list
out	<i>ppComIdDsidMap</i>	Pointer to an array of a structures of type TRDP_COMID_DSID_MAP_T (p. 21)
out	<i>pNumDataset</i>	Pointer to the number of datasets found in the configuration
out	<i>papDataset</i>	Pointer to an array of pointers to a structures of type TRDP_DATASET_T

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.12.3.5 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 ** *pplfConfig*)

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

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

5.12.3.6 EXT_DECL TRDP_ERR_T tau_readXmlInterfaceConfig (const TRDP_XML_DOC_HANDLE_T * *pDocHnd*, const CHAR8 * *plfName*, 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	<i>pDocHnd</i>	Handle of the XML document prepared by tau_prepareXmlDoc
in	<i>plfName</i>	Interface name
out	<i>pProcessConfig</i>	TRDP process (session) configuration for the interface
out	<i>pPdConfig</i>	PD default configuration for the interface
out	<i>pMdConfig</i>	MD default configuration for the interface
out	<i>pNumExchgPar</i>	Number of configured telegrams
out	<i>ppExchgPar</i>	Pointer to array of telegram configurations

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_PARAM_ERR</i>	File not existing

Here is the call graph for this function:

5.13 trdp_dllmain.c File Reference

Windows DLL main function.

5.13.1 Detailed Description

Windows DLL main function.

Note

Project: TCNOpen TRDP prototype stack

Author

Armin-H. Weiss, Bombardier

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:

trdp_dllmain.c (p. 86) 1065 2013-09-06 08:12:09Z aweiss

5.14 trdp_if.c File Reference

Functions for ECN communication.

```
#include <string.h>
#include "trdp_if_light.h"
#include "trdp_utils.h"
#include "trdp_pdcom.h"
#include "trdp_stats.h"
#include "vos_sock.h"
#include "vos_mem.h"
```

Include dependency graph for trdp_if.c:

Functions

- **BOOL8 trdp_isValidSession** (TRDP_APP_SESSION_T pSessionHandle)
Check if the session handle is valid.
- **TRDP_APP_SESSION_T * trdp_sessionQueue** (void)
Get the session queue head pointer.
- **EXT_DECL TRDP_ERR_T tlc_init** (const TRDP_PRINT_DBG_T pPrintDebugString, const TRDP_MEM_CONFIG_T *pMemConfig)
Initialize the TRDP stack.
- **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_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_reinitSession** (TRDP_APP_SESSION_T appHandle)
Re-Initialize.
- **const char * tlc_getVersionString** (void)
Return a human readable version representation.
- **EXT_DECL const TRDP_VERSION_T * tlc_getVersion** (void)
Return version.
- **TRDP_ERR_T tlp_setRedundant** (TRDP_APP_SESSION_T appHandle, UINT32 redId, BOOL8 leader)
Do not send non-redundant PDs 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 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** **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, const UINT8 *pData, UINT32 dataSize)

Prepare for sending PD messages.

- **TRDP_ERR_T** **tlp_unpublish** (**TRDP_APP_SESSION_T** appHandle, **TRDP_PUB_T** pubHandle)

Stop sending PD messages.

- **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** **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_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** *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.

- EXT_DECL **TRDP_ERR_T** **tlp_unsubscribe** (**TRDP_APP_SESSION_T** appHandle, **TRDP_SUB_T** subHandle)

Stop receiving PD messages.

- 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.

- 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.

5.14.1 Detailed Description

Functions for ECN communication.

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, 2013. All rights reserved.

Id:

trdp_if.c (p. 87) 1354 2014-11-11 15:22:13Z ahweiss

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

BL 2014-06-03: Do not return error on data-less tlp_request

BL 2014-06-02: Ticket #41: Sequence counter handling fixed
Removing receive queue on session close added

BL 2014-02-27: Ticket #24: trdp_if.c won't compile without MD_SUPPORT

BL 2013-06-24: ID 125: Time-out handling and ready descriptors fixed

BL 2013-02-01: ID 53: Zero dataset size fixed for PD

BL 2013-01-25: ID 20: Redundancy handling fixed

BL 2013-01-08: LADDER: Removed/Changed some ladder specific code in tlp_subscribe()

BL 2012-12-03: ID 1: "using uninitialized PD_ELE_T.pullIpAddress variable"
ID 2: "uninitialized PD_ELE_T.newPD->pNext in tlp_subscribe()"

5.14.2 Function Documentation**5.14.2.1 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	<i>appHandle</i>	The handle returned by tlc_openSession
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	handle NULL

Here is the call graph for this function:

5.14.2.2 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	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of highest used descriptors (for <code>select()</code>)

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.14.2.3 EXT_DECL const TRDP_VERSION_T* tlc_getVersion (void)

Return version.

Return pointer to version structure

Return values

<i>TRDP_VERSION_T</i> (p. 44)	
-------------------------------	--

5.14.2.4 const char* tlc_getVersionString (void)

Return a human readable version representation.

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

Return values

<i>const</i>	string
--------------	--------

5.14.2.5 EXT_DECL TRDP_ERR_T tlc_init (const TRDP_PRINT_DBG_T pPrintDebugString, const TRDP_MEM_CONFIG_T * pMemConfig)

Initialize the TRDP stack.

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

Parameters

in	<i>pPrintDebugString</i>	Pointer to debug print function
in	<i>pMemConfig</i>	Pointer to memory configuration

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	memory allocation failed
<i>TRDP_PARAM_ERR</i>	initialization error

Here is the call graph for this function:

5.14.2.6 **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	<i>pAppHandle</i>	A handle for further calls to the trdp stack
in	<i>ownIpAddr</i>	Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
in	<i>leaderIpAddr</i>	IP address of redundancy leader
in	<i>pMarshall</i>	Pointer to marshalling configuration
in	<i>pPdDefault</i>	Pointer to default PD configuration
in	<i>pMdDefault</i>	Pointer to default MD configuration
in	<i>pProcessConfig</i>	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

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP SOCK_ERR</i>	socket error

Here is the call graph for this function:

5.14.2.7 **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	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.14.2.8 **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 occurred during normal operation. We need to re-join the multicast groups...

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	handle NULL

Here is the call graph for this function:

5.14.2.9 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	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>etbTopoCnt</i>	New <code>etbTopoCnt</code> value

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.14.2.10 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	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
in	<i>opTrnTopoCnt</i>	New operational topocount value

Here is the call graph for this function:

5.14.2.11 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

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	TrafficStore nothing
<i>TRDP_MUTEX_ERR</i>	TrafficStore mutex err

Here is the call graph for this function:

5.14.2.12 **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 callbacks

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_SUB_ERR</i>	not subscribed
<i>TRDP_TIMEOUT_ERR</i>	packet timed out
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

Here is the call graph for this function:

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

Get status of redundant ComIDs.

Only the status of the first redundancy group entry is returned will be returned!

Parameters

in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>redId</i>	will be returned for all ComID's with the given redId
in, out	<i>pLeader</i>	TRUE if we're sending this redundancy group (leader)

Return values

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

Here is the call graph for this function:

5.14.2.14 **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	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pPubHandle</i>	returned handle for related unprepare
in	<i>comId</i>	comId of packet to send
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>interval</i>	frequency of PD packet (≥ 10 ms) in usec, 0 if PD PULL
in	<i>redId</i>	0 - Non-redundant, > 0 valid redundancy group
in	<i>pktFlags</i>	OPTION: <code>TRDP_FLAGS_DEFAULT</code> , <code>TRDP_FLAGS_NONE</code> , <code>TRDP_FLAGS_MARSHALL</code> , <code>TRDP_FLAGS_CALLBACK</code>
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data ≤ 1436 without FCS

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_NOPUB_ERR</i>	Already published

Here is the call graph for this function:

5.14.2.15 `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	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pubHandle</i>	the handle returned by <code>publish</code>
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>dataSize</i>	size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error on uninitialized parameter or changed <code>dataSize</code> compared to published one
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

Here is the call graph for this function:

5.14.2.16 `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, const UINT8 * pData, UINT32 dataSize)`

Prepare for sending PD messages.

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

Parameters

in	<i>appHandle</i>	the handle returned by <i>tlc_init</i>
in	<i>pubHandle</i>	handle for related unpublish
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

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

Here is the call graph for this function:

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_NOSUB_ERR</i>	no matching subscription found

Here is the call graph for this function:

5.14.2.18 **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	<i>appHandle</i>	the handle returned by tlc_init
in	<i>subHandle</i>	handle for this subscription
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	IP for source filtering, set 0 if not used
in	<i>destIpAddr</i>	IP address to join

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not reserve memory (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP SOCK_ERR</i>	Resource (socket) not available, subscription canceled

Here is the call graph for this function:

5.14.2.19 **TRDP_ERR_T** tlp_setRedundant (**TRDP_APP_SESSION_T** *appHandle*, **UINT32** *redId*, **BOOL8** *leader*)

Do not send non-redundant PDs when we are follower.

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

Parameters

in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>redId</i>	will be set for all ComID's with the given redId, 0 to change for all redId
in	<i>leader</i>	TRUE if we send

Return values

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

Here is the call graph for this function:

5.14.2.20 **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	<i>appHandle</i>	the handle returned by tlc_openSession
----	------------------	--

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

Return values

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

Here is the call graph for this function:

5.14.2.21 TRDP_ERR_T tlp_unpublish (TRDP_APP_SESSION_T appHandle, TRDP_PUB_T pubHandle)

Stop sending PD messages.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>pubHandle</i>	the handle returned by prepare

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.14.2.22 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	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>subHandle</i>	the handle for this subscription

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error

<i>TRDP_NOSUB_ERR</i>	not subscribed
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.14.2.23 BOOL8 trdp_isValidSession (TRDP_APP_SESSION_T pSessionHandle)

Check if the session handle is valid.

Parameters

in	<i>pSessionHandle</i>	pointer to packet data (dataset)
----	-----------------------	----------------------------------

Return values

<i>TRUE</i>	is valid
<i>FALSE</i>	is invalid

Here is the call graph for this function:

5.14.2.24 TRDP_APP_SESSION_T* trdp_sessionQueue (void)

Get the session queue head pointer.

Return values

<i>&sSession</i>

5.15 trdp_if.h File Reference

Typedefs for TRDP communication.

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

Include dependency graph for trdp_if.h: This graph shows which files directly or indirectly include this file:

Functions

- **BOOL8 trdp_isValidSession (TRDP_APP_SESSION_T pSessionHandle)**
Check if the session handle is valid.
- **TRDP_APP_SESSION_T * trdp_sessionQueue (void)**
Get the session queue head pointer.

5.15.1 Detailed Description

Typedefs for TRDP communication.

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, 2013. All rights reserved.

Id:

trdp_if.h (p. 98) 1264 2014-07-14 15:54:26Z bloehr

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

5.15.2 Function Documentation**5.15.2.1 BOOL8 trdp_isValidSession (TRDP_APP_SESSION_T pSessionHandle)**

Check if the session handle is valid.

Parameters

in	<i>pSessionHandle</i>	pointer to packet data (dataset)
----	-----------------------	----------------------------------

Return values

<i>TRUE</i>	is valid
<i>FALSE</i>	is invalid

Here is the call graph for this function:

5.15.2.2 TRDP_APP_SESSION_T* trdp_sessionQueue (void)

Get the session queue head pointer.

Return values

<i>&sSession</i>

5.16 trdp_if_light.h File Reference

TRDP Light interface functions (API)

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

Include dependency graph for trdp_if_light.h: This graph shows which files directly or indirectly include this file:

Macros

- **#define MD_SUPPORT 1**

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

Functions

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

Initialize the TRDP stack.

- 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_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_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, const UINT8 *pData, UINT32 dataSize)
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, BOOL 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, BOOL *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 *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.

- **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.

- **EXT_DECL TRDP_ERR_T tlp_unsubscribe** (TRDP_APP_SESSION_T appHandle, TRDP_SUB_T subHandle)

Stop receiving PD messages.

- **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.

- **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.

- **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.

- **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.

- **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 *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.

- **EXT_DECL TRDP_ERR_T tlm_readListener** (TRDP_APP_SESSION_T appHandle, TRDP_LIS_T listenHandle, 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 listenHandle)

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, 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 *pNumSubs, **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_getListStatistics** (**TRDP_APP_SESSION_T** appHandle, UINT16 *pNumList, **TRDP_LIST_STATISTICS_T** *pStatistics)
Return 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.

5.16.1 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

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:

trdp_if_light.h (p. 99) 1341 2014-10-14 16:05:53Z bloehr

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

5.16.2 Function Documentation

5.16.2.1 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	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	handle NULL

Here is the call graph for this function:

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

Frees the buffer reserved by the TRDP layer.

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
in	<i>pBuf</i>	pointer to the buffer to be freed

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	buffer pointer invalid

5.16.2.3 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	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
out	<i>pInterval</i>	pointer to needed interval
in, out	<i>pFileDesc</i>	pointer to file descriptor set
out	<i>pNoDesc</i>	pointer to put no of used descriptors (for select())

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.16.2.4 EXT_DECL TRDP_ERR_T tlc_getJoinStatistics (TRDP_APP_SESSION_T appHandle, UINT16 * pNumJoin, UINT32 * plpAddr)

Return join statistics.

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

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumJoin</i>	Pointer to the number of joined IP Addresses
out	<i>plpAddr</i>	Pointer to a list with the joined IP addresses

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more items than requested

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumJoin</i>	Pointer to the number of joined IP Addresses
out	<i>plpAddr</i>	Pointer to a list with the joined IP addresses

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more items than requested

Here is the call graph for this function:

5.16.2.5 EXT_DECL TRDP_ERR_T tlc_getListStatistics (TRDP_APP_SESSION_T appHandle, UINT16 * pNumList, TRDP_LIST_STATISTICS_T * pStatistics)

Return MD listener statistics.

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

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

5.16.2.6 **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 implicitly.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumPub</i>	Pointer to the number of publishers
out	<i>pStatistics</i>	Pointer to a list with the publish statistics information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Here is the call graph for this function:

5.16.2.7 **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 implicitly.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumRed</i>	Pointer to the number of redundancy groups

out	<i>pStatistics</i>	Pointer to a list with the redundancy group information
-----	--------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumRed</i>	Pointer to the number of redundancy groups
out	<i>pStatistics</i>	Pointer to a list with the redundancy group information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Here is the call graph for this function:

5.16.2.8 `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	<i>appHandle</i>	the handle returned by <code>tlc_init</code>
out	<i>pStatistics</i>	Pointer to statistics for this application session

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pStatistics</i>	Pointer to statistics for this application session

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

Here is the call graph for this function:

5.16.2.9 `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 implicitly.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Memory for statistics information must be provided by the user.

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Here is the call graph for this function:

5.16.2.10 EXT_DECL const TRDP_VERSION_T* tlc_getVersion (void)

Return version.

Return pointer to version structure

Return values

<i>const</i>	TRDP_VERSION_T (p. 44)
--------------	-------------------------------

Return pointer to version structure

Return values

TRDP_VERSION_T (p. 44)	
-------------------------------	--

5.16.2.11 EXT_DECL const CHAR8* tlc_getVersionString (void)

Return a human readable version representation.

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

Return values

<i>const</i>	string
--------------	--------

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

Initialize the TRDP stack.

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

Parameters

in	<i>pPrintDebugString</i>	Pointer to debug print function
in	<i>pMemConfig</i>	Pointer to memory configuration

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	memory allocation failed
<i>TRDP_PARAM_ERR</i>	initialization error

Here is the call graph for this function:

5.16.2.13 **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	<i>pAppHandle</i>	A handle for further calls to the trdp stack
in	<i>ownIpAddr</i>	Own IP address, can be different for each process in multihoming systems, if zero, the default interface / IP will be used.
in	<i>leaderIpAddr</i>	IP address of redundancy leader
in	<i>pMarshall</i>	Pointer to marshalling configuration
in	<i>pPdDefault</i>	Pointer to default PD configuration
in	<i>pMdDefault</i>	Pointer to default MD configuration
in	<i>pProcessConfig</i>	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

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	not yet initied
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP SOCK_ERR</i>	socket error

Here is the call graph for this function:

5.16.2.14 **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	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

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

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.16.2.15 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 occurred during normal operation. We need to re-join the multicast groups...

Parameters

in	<i>appHandle</i>	The handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	handle NULL

Here is the call graph for this function:

5.16.2.16 EXT_DECL TRDP_ERR_T tlc_resetStatistics (TRDP_APP_SESSION_T *appHandle*)

Reset statistics.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_init</code>
----	------------------	--

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error

Here is the call graph for this function:

5.16.2.17 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	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
in	<i>etbTopoCnt</i>	New topocount value

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

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>etbTopoCnt</i>	New <i>etbTopoCnt</i> value

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.16.2.18 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	<i>appHandle</i>	The handle returned by <code>tlc_init</code>
in	<i>opTrnTopoCnt</i>	New operational topocount value

Here is the call graph for this function:

5.16.2.19 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

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_INIT_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	TrafficStore nothing
<i>TRDP_MUTEX_ERR</i>	TrafficStore mutex err

Here is the call graph for this function:

5.16.2.20 **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	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pSessionId</i>	Session ID returned by request

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory

<i>TRDP_NOINIT_ERR</i>	handle invalid
------------------------	----------------

5.16.2.22 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	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pSessionId</i>	Session ID returned by request
in	<i>userStatus</i>	Info for requester about application errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.16.2.23 EXT_DECL TRDP_ERR_T tlm_delListener (TRDP_APP_SESSION_T *appHandle*, TRDP_LIS_T *listenHandle*)

Remove Listener.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_init
out	<i>listenHandle</i>	Handle for this listener

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.16.2.24 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	<i>appHandle</i>	the handle returned by tlc_init
----	------------------	---------------------------------

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	Out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.16.2.27 `TRDP_ERR_T tlm_replyErr (TRDP_APP_SESSION_T appHandle, const TRDP_UUID_T * pSessionId, 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	<i>appHandle</i>	the handle returned by <code>tlc_init</code>
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>replyStatus</i>	Info for requester about stack errors
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.16.2.28 `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	<i>appHandle</i>	the handle returned by <code>tlc_init</code>
in	<i>pSessionId</i>	Session ID returned by indication
in	<i>comId</i>	comId of packet to be sent
in	<i>userStatus</i>	Info for requester about application errors

in	<i>confirmTimeout</i>	timeout for confirmation
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NO_SESSION_ERR</i>	no such session
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.16.2.29 **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	<i>appHandle</i>	the handle returned by <i>tlc_init</i>
in	<i>pUserRef</i>	user supplied value returned with reply
in	<i>pfCbFunction</i>	Pointer to listener specific callback function, NULL to use default function
out	<i>pSessionId</i>	return session ID
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>pktFlags</i>	OPTIONS: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_TCP
in	<i>numReplies</i>	number of expected replies, 0 if unknown
in	<i>replyTimeout</i>	timeout for reply
in	<i>maxNumRetries</i>	maximum number of retries (0 2)
in	<i>pSendParam</i>	Pointer to send parameters, NULL to use default send parameters
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>sourceURI</i>	only functional group of source URI
in	<i>destURI</i>	only functional group of destination URI

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	out of memory
<i>TRDP_NOINIT_ERR</i>	handle invalid

5.16.2.30 **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	<i>appHandle</i>	the handle returned by <i>tlc_init</i>
in	<i>subHandle</i>	the handle returned by subscription
in, out	<i>pPdInfo</i>	pointer to application's info buffer
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>pDataSize</i>	in: size of buffer, out: size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_SUB_ERR</i>	not subscribed
<i>TRDP_TIMEOUT_ERR</i>	packet timed out
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

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

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_SUB_ERR</i>	not subscribed
<i>TRDP_TIMEOUT_ERR</i>	packet timed out
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

Here is the call graph for this function:

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

Get status of redundant ComIDs.

Parameters

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

Return values

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

Only the status of the first redundancy group entry is returned will be returned!

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_init</code>
in	<i>redId</i>	will be returned for all ComID's with the given <code>redId</code>
in,out	<i>pLeader</i>	TRUE if we're sending this redundancy group (leader)

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error / <code>redId</code> not existing
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

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

Return values

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

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

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
out	<i>pPubHandle</i>	returned handle for related unprepare
in	<i>comId</i>	comId of packet to send
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - <code>srcIP</code> will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to

in	<i>interval</i>	frequency of PD packet (≥ 10 ms) in usec, 0 if PD PULL
in	<i>redId</i>	0 - Non-redundant, > 0 valid redundancy group
in	<i>pktFlags</i>	OPTION: TRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK
in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data ≤ 1436 without FCS

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_NOPUB_ERR</i>	Already published

Here is the call graph for this function:

5.16.2.33 `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	<i>appHandle</i>	the handle returned by <code>tlc_init</code>
in	<i>pubHandle</i>	the handle returned by <code>publish</code>
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>dataSize</i>	size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error on uninitialized parameter or changed <code>dataSize</code> compared to published one
<i>TRDP_PUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_COMID_ERR</i>	ComID not found when marshalling

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

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>pubHandle</i>	the handle returned by <code>publish</code>
in, out	<i>pData</i>	pointer to application's data buffer
in, out	<i>dataSize</i>	size of data

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error on uninitialized parameter or changed <code>dataSize</code> compared to published one
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid

<i>TRDP_COMID_ERR</i>	ComID not found when marshalling
-----------------------	----------------------------------

Here is the call graph for this function:

5.16.2.34 **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*, **const** **UINT8** * *pData*, **UINT32** *dataSize*)

Prepare for sending PD messages.

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

Parameters

in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pubHandle</i>	handle for related unpublish
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data

Return values

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

Here is the call graph for this function:

5.16.2.35 **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	<i>appHandle</i>	the handle returned by tlc_init
in	<i>subHandle</i>	handle from related subscribe
in	<i>comId</i>	comId of packet to be sent
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	own IP address, 0 - srcIP will be set by the stack
in	<i>destIpAddr</i>	where to send the packet to
in	<i>redId</i>	0 - Non-redundant, > 0 valid redundancy group
in	<i>pktFlags</i>	OPTIONS: TTRDP_FLAGS_DEFAULT, TRDP_FLAGS_NONE, TRDP_FLAGS_MARSHALL, TRDP_FLAGS_CALLBACK

in	<i>pSendParam</i>	optional pointer to send parameter, NULL - default parameters are used
in	<i>pData</i>	pointer to packet data / dataset
in	<i>dataSize</i>	size of packet data
in	<i>replyComId</i>	comId of reply
in	<i>replyIpAddr</i>	IP for reply

Return values

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

Send a PD request message

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not insert (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_NOSUB_ERR</i>	no matching subscription found

Here is the call graph for this function:

5.16.2.36 `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	<i>appHandle</i>	the handle returned by <i>tlc_init</i>
in	<i>subHandle</i>	handle for this subscription
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication

in	<i>srcIpAddr</i>	IP for source filtering, set 0 if not used
in	<i>destIpAddr</i>	IP address to join

Return values

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

Resubscribe to a specific PD ComID and source IP

Parameters

in	<i>appHandle</i>	the handle returned by <i>tlc_init</i>
in	<i>subHandle</i>	handle for this subscription
in	<i>etbTopoCnt</i>	ETB topocount to use, 0 if consist local communication
in	<i>opTrnTopoCnt</i>	operational topocount, != 0 for orientation/direction sensitive communication
in	<i>srcIpAddr</i>	IP for source filtering, set 0 if not used
in	<i>destIpAddr</i>	IP address to join

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	could not reserve memory (out of memory)
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP SOCK_ERR</i>	Resource (socket) not available, subscription canceled

Here is the call graph for this function:

5.16.2.37 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	<i>appHandle</i>	the handle returned by <i>tlc_init</i>
in	<i>redId</i>	will be set for all ComID's with the given <i>redId</i> , 0 to change for all <i>redId</i>
in	<i>leader</i>	TRUE if we send

Return values

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

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

Parameters

in	<i>appHandle</i>	the handle returned by <i>tlc_init</i>
in	<i>redId</i>	will be set for all ComID's with the given <i>redId</i> , 0 to change for all <i>redId</i>
in	<i>leader</i>	TRUE if we send

Return values

<i>TRDP_NO_ERR</i>	no error
--------------------	----------

<i>TRDP_PARAM_ERR</i>	parameter error / redId not existing
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

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

Return values

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

Subscribe to a specific PD ComID and source IP.

Parameters

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

in	<i>toBehavior</i>	timeout behavior
----	-------------------	------------------

Return values

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

Here is the call graph for this function:

5.16.2.39 EXT_DECL TRDP_ERR_T tlp_unpublish (TRDP_APP_SESSION_T *appHandle*, TRDP_PUB_T *pubHandle*)

Stop sending PD messages.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_init
in	<i>pubHandle</i>	the handle returned by publish

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in	<i>pubHandle</i>	the handle returned by prepare

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOPUB_ERR</i>	not published
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.16.2.40 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	<i>appHandle</i>	the handle returned by tlc_init
in	<i>subHandle</i>	the handle for this subscription

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error

<i>TRDP_SUB_ERR</i>	not subscribed
<i>TRDP_NOINIT_ERR</i>	handle invalid

Unsubscribe to a specific PD ComID

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in	<i>subHandle</i>	the handle for this subscription

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_NOSUB_ERR</i>	not subscribed
<i>TRDP_NOINIT_ERR</i>	handle invalid

Here is the call graph for this function:

5.17 trdp_mdcom.c File Reference

Functions for MD communication.

```
#include <string.h>
#include "trdp_if_light.h"
#include "trdp_if.h"
#include "trdp_utils.h"
#include "trdp_mdcom.h"
```

Include dependency graph for `trdp_mdcom.c`:

Functions

- **TRDP_ERR_T trdp_mdGetTCPSocket (TRDP_SESSION_PT pSession)**
Initialize the specific parameters for message data Open a listening socket.
- void **trdp_mdFreeSession** (MD_ELE_T *pMDSession)
Free memory of session.
- **TRDP_ERR_T trdp_mdSend** (TRDP_SESSION_PT appHandle)
Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.
- void **trdp_mdCheckPending** (TRDP_APP_SESSION_T appHandle, TRDP_FDS_T *pFileDesc, INT32 *pNoDesc)
Check for pending packets, set FD if non blocking.
- void **trdp_mdCheckListenSocks** (const TRDP_SESSION_PT appHandle, TRDP_FDS_T *pRfds, INT32 *pCount)
Checking receive connection requests and data Call user's callback if needed.
- void **trdp_mdCheckTimeouts** (TRDP_SESSION_PT appHandle)
Checking message data timeouts Call user's callback if needed.

5.17.1 Detailed Description

Functions for MD communication.

Note

Project: TCNOpen TRDP prototype stack

Author

Simone Pachera, FARsystems Gari Oiarbide, CAF Bernd Loehr, NewTec

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:

trdp_mdcom.c (p. 125) 1354 2014-11-11 15:22:13Z ahweiss

BL 2014-08-28: Ticket #62: Failing TCP communication fixed,
Do not read if there's nothing to read ('Mc' has no data!)

BL 2014-08-25: Ticket #57+58: Padding / zero bytes trailing MD & PD packets fixed

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
Ticket #47: Protocol change: no FCS for data part of telegrams

BL 2014-02-28: Ticket #25: CRC32 calculation is not according to IEEE802.3

5.17.2 Function Documentation
5.17.2.1 void trdp_mdCheckListenSocks (const TRDP_SESSION_PT *appHandle*, TRDP_FDS_T * *pRfds*, INT32 * *pCount*)

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Here is the call graph for this function:

5.17.2.2 void trdp_mdCheckPending (TRDP_APP_SESSION_T *appHandle*, TRDP_FDS_T * *pFileDesc*, INT32 * *pNoDesc*)

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors

5.17.2.3 void trdp_mdCheckTimeouts (TRDP_SESSION_PT *appHandle*)

Checking message data timeouts Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

Here is the call graph for this function:

5.17.2.4 void trdp_mdFreeSession (MD_ELE_T * *pMDSession*)

Free memory of session.

Parameters

in	<i>pMDSession</i>	session pointer
----	-------------------	-----------------

Here is the call graph for this function:

5.17.2.5 TRDP_ERR_T trdp_mdGetTCPSocket (TRDP_SESSION_PT pSession)

Initialize the specific parameters for message data Open a listening socket.

Parameters

in	<i>pSession</i>	session parameters
----	-----------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	initialization error

Here is the call graph for this function:

5.17.2.6 TRDP_ERR_T trdp_mdSend (TRDP_SESSION_PT appHandle)

Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

Here is the call graph for this function:

5.18 trdp_mdcom.h File Reference

Functions for MD communication.

```
#include "trdp_private.h"
```

Include dependency graph for trdp_mdcom.h: This graph shows which files directly or indirectly include this file:

Functions

- **TRDP_ERR_T trdp_mdGetTCPSocket (TRDP_SESSION_PT pSession)**
Initialize the specific parameters for message data Open a listening socket.
- **void trdp_mdFreeSession (MD_ELE_T *pMDSession)**
Free memory of session.
- **TRDP_ERR_T trdp_mdSend (TRDP_SESSION_PT appHandle)**
Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.
- **void trdp_mdCheckPending (TRDP_APP_SESSION_T appHandle, TRDP_FDS_T *pFileDesc, INT32 *pNoDesc)**
Check for pending packets, set FD if non blocking.
- **void trdp_mdCheckListenSocks (const TRDP_SESSION_PT appHandle, TRDP_FDS_T *pRfds, INT32 *pCount)**
Checking receive connection requests and data Call user's callback if needed.
- **void trdp_mdCheckTimeouts (TRDP_SESSION_PT appHandle)**
Checking message data timeouts Call user's callback if needed.

5.18.1 Detailed Description

Functions for MD communication.

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, 2013. All rights reserved.

Id:

trdp_mdcom.h (p. 127) 1345 2014-10-16 13:43:05Z railroad-mike

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
Ticket #47: Protocol change: no FCS for data part of telegrams

5.18.2 Function Documentation

5.18.2.1 void trdp_mdCheckListenSocks (const TRDP_SESSION_PT *appHandle*, TRDP_FDS_T * *pRfds*, INT32 * *pCount*)

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Here is the call graph for this function:

5.18.2.2 void trdp_mdCheckPending (TRDP_APP_SESSION_T *appHandle*, TRDP_FDS_T * *pFileDesc*, INT32 * *pNoDesc*)

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors

5.18.2.3 void trdp_mdCheckTimeouts (TRDP_SESSION_PT *appHandle*)

Checking message data timeouts Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

Here is the call graph for this function:

5.18.2.4 void trdp_mdFreeSession (MD_ELE_T * *pMDSession*)

Free memory of session.

Parameters

in	<i>pMDSession</i>	session pointer
----	-------------------	-----------------

Here is the call graph for this function:

5.18.2.5 TRDP_ERR_T trdp_mdGetTCPSocket (TRDP_SESSION_PT *pSession*)

Initialize the specific parameters for message data Open a listening socket.

Parameters

in	<i>pSession</i>	session parameters
----	-----------------	--------------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	initialization error

Here is the call graph for this function:

5.18.2.6 TRDP_ERR_T trdp_mdSend (TRDP_SESSION_PT *appHandle*)

Sending MD messages Send the messages stored in the sendQueue Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

Here is the call graph for this function:

5.19 trdp_pdcom.c File Reference

Functions for PD communication.

```
#include <string.h>
#include "trdp_types.h"
#include "trdp_utils.h"
#include "trdp_pdcom.h"
#include "trdp_if.h"
#include "trdp_stats.h"
#include "vos_sock.h"
#include "vos_mem.h"
```

Include dependency graph for trdp_pdcom.c:

Functions

- void **trdp_pdInit** (PD_ELE_T *pPacket, TRDP_MSG_T type, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, UINT32 replyComId, UINT32 replyIpAddress)

Initialize/construct the packet Set the header infos.

- **TRDP_ERR_T trdp_pdPut** (**PD_ELE_T** *pPacket, **TRDP_MARSHALL_T** marshall, void *refCon, const **UINT8** *pData, **UINT32** dataSize)

Copy data Set the header infos.

- **TRDP_ERR_T trdp_pdGet** (**PD_ELE_T** *pPacket, **TRDP_UNMARSHALL_T** unmarshall, void *refCon, const **UINT8** *pData, **UINT32** *pDataSize)

Copy data Set the header infos.

- **TRDP_ERR_T trdp_pdSendQueued** (**TRDP_SESSION_PT** appHandle)

Send all due PD messages.

- **TRDP_ERR_T trdp_pdReceive** (**TRDP_SESSION_PT** appHandle, **INT32** sock)

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.

- void **trdp_pdCheckPending** (**TRDP_APP_SESSION_T** appHandle, **TRDP_FDS_T** *pFileDesc, **INT32** *pNoDesc)

Check for pending packets, set FD if non blocking.

- void **trdp_pdHandleTimeOuts** (**TRDP_SESSION_PT** appHandle)

Check for time outs.

- **TRDP_ERR_T trdp_pdCheckListenSocks** (**TRDP_SESSION_PT** appHandle, **TRDP_FDS_T** *pRfds, **INT32** *pCount)

Checking receive connection requests and data Call user's callback if needed.

- void **trdp_pdUpdate** (**PD_ELE_T** *pPacket)

Update the header values.

- **TRDP_ERR_T trdp_pdCheckAppTopoCounts** (**TRDP_SESSION_PT** appHandle, **PD_HEADER_T** *pFrame)

Check if the PD header topocounts are correct compared to the session values.

- **TRDP_ERR_T trdp_pdCheck** (**PD_HEADER_T** *pPacket, **UINT32** packetSize)

Check if the PD header values and the CRCs are sane.

- **TRDP_ERR_T trdp_pdSend** (**INT32** pdSock, **PD_ELE_T** *pPacket, **UINT16** port)

Send one PD packet.

- **TRDP_ERR_T trdp_pdDistribute** (**PD_ELE_T** *pSndQueue)

Distribute send time of PD packets over time.

5.19.1 Detailed Description

Functions for PD communication.

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, 2013. All rights reserved.

Id:

trdp_pdcom.c (p. 129) 1354 2014-11-11 15:22:13Z ahweiss

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
 Ticket #47: Protocol change: no FCS for data part of telegrams
 Ticket #43: Usage of memset() in the trdp_pdReceive() function
 BL 2014-06-02: Ticket #41: Sequence counter handling fixed
 Ticket #42: memcmp only if callback enabled
 BL 2014-02-28: Ticket #25: CRC32 calculation is not according IEEE802.3
 BL 2014-02-27: Ticket #23: tlc_getInterval() always returning 10ms
 BL 2014-01-09: Ticket #14: Wrong error return in trdp_pdDistribute()
 BL 2013-06-24: ID 125: Time-out handling and ready descriptors fixed
 BL 2013-04-09: ID 92: Pull request led to reset of push message type
 BL 2013-01-25: ID 20: Redundancy handling fixed

5.19.2 Function Documentation

5.19.2.1 TRDP_ERR_T trdp_pdCheck (PD_HEADER_T * *pPacket*, UINT32 *packetSize*)

Check if the PD header values and the CRCs are sane.

Parameters

in	<i>pPacket</i>	pointer to the packet to check
in	<i>packetSize</i>	max size to check

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_CRC_ERR</i>	

Here is the call graph for this function:

5.19.2.2 TRDP_ERR_T trdp_pdCheckAppTopoCounts (TRDP_SESSION_PT *appHandle*, PD_HEADER_T * *pFrame*)

Check if the PD header topocounts are correct compared to the session values.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pFrame</i>	pointer to the packet to check

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_TOPO_ERR</i>	

Here is the call graph for this function:

5.19.2.3 TRDP_ERR_T trdp_pdCheckListenSocks (TRDP_SESSION_PT *appHandle*, TRDP_FDS_T * *pRfds*, INT32 * *pCount*)

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Here is the call graph for this function:

5.19.2.4 void trdp_pdCheckPending (TRDP_APP_SESSION_T *appHandle*, TRDP_FDS_T * *pFileDesc*, INT32 * *pNoDesc*)

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors

5.19.2.5 TRDP_ERR_T trdp_pdDistribute (PD_ELE_T * pSndQueue)

Distribute send time of PD packets over time.

The duration of PD packets on a 100MBit/s network ranges from 3us to 150us max. Because a cyclic thread scheduling below 5ms would put a too heavy load on the system, and PD packets cannot get larger than 1436 (+ UDP header), we will not account for differences in packet size. Another factor is the differences in intervals for different packets: We should only change the starting times of the packets within 1/2 the interval time. Otherwise a late addition of packets could lead to timeouts of already queued packets. Scheduling will be computed based on the smallest interval time.

Parameters

in	<i>pSndQueue</i>	pointer to send queue
----	------------------	-----------------------

Return values

<i>TRDP_NO_ERR</i>

Here is the call graph for this function:

5.19.2.6 void trdp_pdHandleTimeOuts (TRDP_SESSION_PT appHandle)

Check for time outs.

Parameters

in	<i>appHandle</i>	application handle
----	------------------	--------------------

Here is the call graph for this function:

5.19.2.7 void trdp_pdlNit (PD_ELE_T * pPacket, TRDP_MSG_T type, UINT32 etbTopoCnt, UINT32 opTrnTopoCnt, UINT32 replyComId, UINT32 replyIpAddress)

Initialize/construct the packet Set the header infos.

Parameters

in	<i>pPacket</i>	pointer to the packet element to init
in	<i>type</i>	type the packet
in	<i>etbTopoCnt</i>	topocount to use for PD frame
in	<i>opTrnTopoCnt</i>	topocount to use for PD frame
in	<i>replyComId</i>	Pull request comId
in	<i>replyIpAddress</i>	Pull request Ip

Here is the call graph for this function:

5.19.2.8 TRDP_ERR_T trdp_pdReceive (TRDP_SESSION_PT appHandle, INT32 sock)

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.

If it is a new packet, check if it is a PD Request (PULL). If it is an update, exchange the existing entry with the new one Call user's callback if needed

Parameters

in	<i>appHandle</i>	session pointer
in	<i>sock</i>	the socket to read from

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_WIRE_ERR</i>	protocol error (late packet, version mismatch)
<i>TRDP_QUEUE_ERR</i>	not in queue
<i>TRDP_CRC_ERR</i>	header checksum
<i>TRDP_TOPOCOUNT_ERR</i>	invalid topocount

Here is the call graph for this function:

5.19.2.9 TRDP_ERR_T trdp_pdSend (INT32 *pdSock*, PD_ELE_T * *pPacket*, UINT16 *port*)

Send one PD packet.

Parameters

in	<i>pdSock</i>	socket descriptor
in	<i>pPacket</i>	pointer to packet to be sent
in	<i>port</i>	port on which to send

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_IO_ERR</i>	

Here is the call graph for this function:

5.19.2.10 TRDP_ERR_T trdp_pdSendQueued (TRDP_SESSION_PT *appHandle*)

Send all due PD messages.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_IO_ERR</i>	socket I/O error

Here is the call graph for this function:

5.19.2.11 void trdp_pdUpdate (PD_ELE_T * *pPacket*)

Update the header values.

Parameters

in	<i>pPacket</i>	pointer to the packet to update
----	----------------	---------------------------------

Here is the call graph for this function:

5.20 trdp_pdcom.h File Reference

Functions for PD communication.

```
#include "trdp_private.h"
```

Include dependency graph for trdp_pdcom.h: This graph shows which files directly or indirectly include this file:

Functions

- void **trdp_pdInit** (**PD_ELE_T** *, **TRDP_MSG_T**, UINT32 topoCount, UINT32 optopoCount, UINT32 reply-ComId, UINT32 replyIpAddress)
Initialize/construct the packet Set the header infos.
- void **trdp_pdUpdate** (**PD_ELE_T** *)
Update the header values.
- **TRDP_ERR_T** **trdp_pdPut** (**PD_ELE_T** *, **TRDP_MARSHALL_T** func, void *refCon, const UINT8 *pData, UINT32 dataSize)
Copy data Set the header infos.
- **TRDP_ERR_T** **trdp_pdCheck** (**PD_HEADER_T** *pPacket, UINT32 packetSize)
Check if the PD header values and the CRCs are sane.
- **TRDP_ERR_T** **trdp_pdCheckAppTopoCounts** (**TRDP_SESSION_PT** appHandle, **PD_HEADER_T** *pFrame)
Check if the PD header topocounts are correct compared to the session values.
- **TRDP_ERR_T** **trdp_pdSend** (INT32 pdSock, **PD_ELE_T** *pPacket, UINT16 port)
Send one PD packet.
- **TRDP_ERR_T** **trdp_pdGet** (**PD_ELE_T** *pPacket, **TRDP_UNMARSHALL_T** unmarshall, void *refCon, const UINT8 *pData, UINT32 *pDataSize)
Copy data Set the header infos.
- **TRDP_ERR_T** **trdp_pdSendQueued** (**TRDP_SESSION_PT** appHandle)
Send all due PD messages.
- **TRDP_ERR_T** **trdp_pdReceive** (**TRDP_SESSION_PT** pSessionHandle, INT32 sock)
Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.
- void **trdp_pdCheckPending** (**TRDP_APP_SESSION_T** appHandle, **TRDP_FDS_T** *pFileDesc, INT32 *pNoDesc)
Check for pending packets, set FD if non blocking.
- void **trdp_pdHandleTimeOuts** (**TRDP_SESSION_PT** appHandle)
Check for time outs.
- **TRDP_ERR_T** **trdp_pdCheckListenSocks** (**TRDP_SESSION_PT** appHandle, **TRDP_FDS_T** *pRfds, INT32 *pCount)
Checking receive connection requests and data Call user's callback if needed.
- **TRDP_ERR_T** **trdp_pdDistribute** (**PD_ELE_T** *pSndQueue)
Distribute send time of PD packets over time.

5.20.1 Detailed Description

Functions for PD communication.

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, 2013. All rights reserved.

Id:

trdp_pdcom.h (p. 135) 1327 2014-09-04 09:21:51Z bloehr

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
Ticket #47: Protocol change: no FCS for data part of telegrams

5.20.2 Function Documentation**5.20.2.1 TRDP_ERR_T trdp_pdCheck (PD_HEADER_T * pPacket, UINT32 packetSize)**

Check if the PD header values and the CRCs are sane.

Parameters

in	<i>pPacket</i>	pointer to the packet to check
in	<i>packetSize</i>	max size to check

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_CRC_ERR</i>	

Here is the call graph for this function:

5.20.2.2 TRDP_ERR_T trdp_pdCheckAppTopoCounts (TRDP_SESSION_PT appHandle, PD_HEADER_T * pFrame)

Check if the PD header topocounts are correct compared to the session values.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pFrame</i>	pointer to the packet to check

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_TOPO_ERR</i>	

Here is the call graph for this function:

5.20.2.3 TRDP_ERR_T trdp_pdCheckListenSocks (TRDP_SESSION_PT appHandle, TRDP_FDS_T * pRfds, INT32 * pCount)

Checking receive connection requests and data Call user's callback if needed.

Parameters

in	<i>appHandle</i>	session pointer
in	<i>pRfds</i>	pointer to set of ready descriptors
in, out	<i>pCount</i>	pointer to number of ready descriptors

Here is the call graph for this function:

5.20.2.4 void trdp_pdCheckPending (TRDP_APP_SESSION_T *appHandle*, TRDP_FDS_T * *pFileDesc*, INT32 * *pNoDesc*)

Check for pending packets, set FD if non blocking.

Parameters

in	<i>appHandle</i>	session pointer
in, out	<i>pFileDesc</i>	pointer to set of ready descriptors
in, out	<i>pNoDesc</i>	pointer to number of ready descriptors

5.20.2.5 TRDP_ERR_T trdp_pdDistribute (PD_ELE_T * *pSndQueue*)

Distribute send time of PD packets over time.

The duration of PD packets on a 100MBit/s network ranges from 3us to 150us max. Because a cyclic thread scheduling below 5ms would put a too heavy load on the system, and PD packets cannot get larger than 1436 (+ UDP header), we will not account for differences in packet size. Another factor is the differences in intervals for different packets: We should only change the starting times of the packets within 1/2 the interval time. Otherwise a late addition of packets could lead to timeouts of already queued packets. Scheduling will be computed based on the smallest interval time.

Parameters

in	<i>pSndQueue</i>	pointer to send queue
----	------------------	-----------------------

Return values

<i>TRDP_NO_ERR</i>

Here is the call graph for this function:

5.20.2.6 void trdp_pdHandleTimeOuts (TRDP_SESSION_PT *appHandle*)

Check for time outs.

Parameters

in	<i>appHandle</i>	application handle
----	------------------	--------------------

Here is the call graph for this function:

5.20.2.7 void trdp_pdInit (PD_ELE_T * *pPacket*, TRDP_MSG_T *type*, UINT32 *etbTopoCnt*, UINT32 *opTrnTopoCnt*, UINT32 *replyComId*, UINT32 *replyIpAddress*)

Initialize/construct the packet Set the header infos.

Parameters

in	<i>pPacket</i>	pointer to the packet element to init
in	<i>type</i>	type the packet
in	<i>etbTopoCnt</i>	topocount to use for PD frame
in	<i>opTrnTopoCnt</i>	topocount to use for PD frame
in	<i>replyComId</i>	Pull request comId
in	<i>replyIpAddress</i>	Pull request Ip

Here is the call graph for this function:

5.20.2.8 TRDP_ERR_T trdp_pdReceive (TRDP_SESSION_PT *appHandle*, INT32 *sock*)

Receiving PD messages Read the receive socket for arriving PDs, copy the packet to a new PD_ELE_T Check for protocol errors and compare the received data to the data in our receive queue.

If it is a new packet, check if it is a PD Request (PULL). If it is an update, exchange the existing entry with the new one Call user's callback if needed

Parameters

in	<i>appHandle</i>	session pointer
in	<i>sock</i>	the socket to read from

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_WIRE_ERR</i>	protocol error (late packet, version mismatch)
<i>TRDP_QUEUE_ERR</i>	not in queue
<i>TRDP_CRC_ERR</i>	header checksum
<i>TRDP_TOPOCOUNT_ERR</i>	invalid topocount

Here is the call graph for this function:

5.20.2.9 TRDP_ERR_T trdp_pdSend (INT32 *pdSock*, PD_ELE_T * *pPacket*, UINT16 *port*)

Send one PD packet.

Parameters

in	<i>pdSock</i>	socket descriptor
in	<i>pPacket</i>	pointer to packet to be sent
in	<i>port</i>	port on which to send

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_IO_ERR</i>	

Here is the call graph for this function:

5.20.2.10 TRDP_ERR_T trdp_pdSendQueued (TRDP_SESSION_PT *appHandle*)

Send all due PD messages.

Parameters

in	<i>appHandle</i>	session pointer
----	------------------	-----------------

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_IO_ERR</i>	socket I/O error

Here is the call graph for this function:

5.20.2.11 void trdp_pdUpdate (PD_ELE_T * pPacket)

Update the header values.

Parameters

in	<i>pPacket</i>	pointer to the packet to update
----	----------------	---------------------------------

Here is the call graph for this function:

5.21 trdp_private.h File Reference

Typedefs for TRDP communication.

```
#include "trdp_types.h"
#include "trdp_proto.h"
#include "vos_thread.h"
#include "vos_sock.h"
```

Include dependency graph for trdp_private.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct **TRDP_HANDLE**
Hidden handle definition, used as unique addressing item.
- struct **TRDP_SEQ_CNT_ENTRY_T**
Tuples of last received sequence counter per comId.
- struct **TRDP_SOCKET_TCP**
TCP parameters.
- struct **TRDP_SOCKETS**
Socket item.
- struct **GNU_PACKED**
Types for ETB control.
- struct **PD_ELE**
Queue element for PD packets to send or receive.
- struct **TRDP_SESSION**
Session/application variables store.

Macros

- #define **TRDP_TIMER_GRANULARITY** 10000
granularity in us
- #define **TRDP_TIMER_FOREVER** 0xffffffff
granularity in us
- #define **TRDP_MD_DEFAULT_REPLY_TIMEOUT** 5000000

- default reply time out 5s*
- #define **TRDP_MD_DEFAULT_CONFIRM_TIMEOUT** 1000000
 - default confirm time out 1s*
- #define **TRDP_MD_DEFAULT_CONNECTION_TIMEOUT** 60000000
 - Socket connection time out 1 minute.*
- #define **TRDP_MD_DEFAULT_SENDING_TIMEOUT** 5000000
 - Socket sending time out 5s.*
- #define **TRDP_PROCESS_DEFAULT_CYCLE_TIME** 10000
 - Default cycle time for TRDP process.*
- #define **TRDP_PROCESS_DEFAULT_PRIORITY** 64
 - Default priority of TRDP process.*
- #define **TRDP_PROCESS_DEFAULT_OPTIONS** TRDP_OPTION_TRAFFIC_SHAPING
 - Default options for TRDP process.*
- #define **TRDP_DEBUG_DEFAULT_FILE_SIZE** 65536
 - Default maximum size of log file.*
- #define **TRDP_SEQ_CNT_START_ARRAY_SIZE** 64
 - This should be enough for the start.*

Typedefs

- typedef struct **TRDP_HANDLE** TRDP_ADDRESSES_T
 - Hidden handle definition, used as unique addressing item.*
- typedef struct **TRDP_SOCKET_TCP** TRDP_SOCKET_TCP_T
 - TCP parameters.*
- typedef struct **TRDP_SOCKETS** TRDP_SOCKETS_T
 - Socket item.*
- typedef struct **PD_ELE** PD_ELE_T
 - Queue element for PD packets to send or receive.*
- typedef struct **TRDP_SESSION** TRDP_SESSION_T
 - Session/application variables store.*

Enumerations

- enum **TRDP_MD_ELE_ST_T** {
 - TRDP_ST_NONE = 0,
 - TRDP_ST_TX_NOTIFY_ARM = 1,
 - TRDP_ST_TX_REQUEST_ARM = 2,
 - TRDP_ST_TX_REPLY_ARM = 3,
 - TRDP_ST_TX_REPLYQUERY_ARM = 4,
 - TRDP_ST_TX_CONFIRM_ARM = 5,
 - TRDP_ST_RX_READY = 6,
 - TRDP_ST_TX_REQUEST_W4REPLY = 7,
 - TRDP_ST_RX_REPLYQUERY_W4C = 8,
 - TRDP_ST_RX_REQ_W4AP_REPLY = 9,
 - TRDP_ST_TX_REQ_W4AP_CONFIRM = 10,
 - TRDP_ST_RX_REPLY_SENT = 11,
 - TRDP_ST_RX_NOTIFY_RECEIVED = 12,
 - TRDP_ST_TX_REPLY_RECEIVED = 13,
 - TRDP_ST_RX_CONF_RECEIVED = 14 }
 - Internal MD state.*

- enum **TRDP_PRIV_FLAGS_T** { ,
TRDP_TIMED_OUT = 0x2,
TRDP_INVALID_DATA = 0x4,
TRDP_REQ_2B_SENT = 0x8,
TRDP_PULL_SUB = 0x10,
TRDP_REDUNDANT = 0x20 }

Internal flags for packets.

- enum **TRDP SOCK_TYPE_T** {
TRDP SOCK_PD = 0,
TRDP SOCK_MD_UDP = 1,
TRDP SOCK_MD_TCP = 2 }

Socket usage.

5.21.1 Detailed Description

Typedefs for TRDP communication. TRDP internal type definitions

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, 2013. All rights reserved.

Id:

trdp_private.h (p. 139) 1329 2014-09-04 16:03:50Z bloehr

BL 2014-06-02: Ticket #41: Sequence counter handling fixed

5.21.2 Enumeration Type Documentation

5.21.2.1 enum TRDP_MD_ELE_ST_T

Internal MD state.

Enumerator

TRDP_ST_NONE neutral value
TRDP_ST_TX_NOTIFY_ARM ready to send notify MD
TRDP_ST_TX_REQUEST_ARM ready to send request MD
TRDP_ST_TX_REPLY_ARM ready to send reply MD
TRDP_ST_TX_REPLYQUERY_ARM ready to send reply with confirm request MD
TRDP_ST_TX_CONFIRM_ARM ready to send confirm MD
TRDP_ST_RX_READY armed listener
TRDP_ST_TX_REQUEST_W4REPLY request sent, wait for reply
TRDP_ST_RX_REPLYQUERY_W4C reply send, with confirm request MD

TRDP_ST_RX_REQ_W4AP_REPLY request received, wait for application reply send
TRDP_ST_TX_REQ_W4AP_CONFIRM reply conf. rq. tx, wait for application conf send
TRDP_ST_RX_REPLY_SENT reply sent
TRDP_ST_RX_NOTIFY_RECEIVED notification received, wait for application to accept
TRDP_ST_TX_REPLY_RECEIVED reply received
TRDP_ST_RX_CONF_RECEIVED confirmation received

5.21.2.2 enum TRDP_PRIV_FLAGS_T

Internal flags for packets.

Enumerator

TRDP_TIMED_OUT if set, inform the user
TRDP_INVALID_DATA if set, inform the user
TRDP_REQ_2B_SENT if set, the request needs to be sent
TRDP_PULL_SUB if set, its a PULL subscription
TRDP_REDUNDANT if set, packet should not be sent (redundant)

5.21.2.3 enum TRDP SOCK_TYPE_T

Socket usage.

Enumerator

TRDP SOCK_PD Socket is used for UDP process data.
TRDP SOCK_MD_UDP Socket is used for UDP message data.
TRDP SOCK_MD_TCP Socket is used for TCP message data.

5.22 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** 20548
process data UDP port
- #define **TRDP_MD_UDP_PORT** 20550
message data UDP port

- **#define TRDP_MD_TCP_PORT** 20550
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.

5.22.1 Detailed Description

Definitions for the TRDP protocol. TRDP internal type definitions

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, 2013. All rights reserved.

Id:

trdp_proto.h (p. 142) 1316 2014-08-27 15:05:54Z bloehr

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

5.22.2 Macro Definition Documentation**5.22.2.1 #define TRDP_DEST_URI_SIZE 32**

max.

Dest URI size in MD header

5.22.2.2 #define TRDP_ETBCTRL_COMID 1

TRDP reserved COMIDs in the range 1 ...

1000

5.22.2.3 #define TRDP_ETBCTRL_DSID 1

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

1000

5.22.2.4 #define TRDP_MAX_FILE_NAME_LEN 128

path and file name length incl.

terminating '0'

5.22.2.5 #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'

5.22.2.6 #define TRDP_MAX_URI_HOST_LEN (4 * TRDP_MAX_LABEL_LEN)

URI host part length incl.

terminating '0'

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

URI length incl.

terminating '0' and 1 padding byte

5.22.2.8 `#define TRDP_MAX_URI_USER_LEN (2 * TRDP_MAX_LABEL_LEN)`

URI user part incl.

terminating '0'

5.22.3 Enumeration Type Documentation

5.22.3.1 `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

5.23 trdp_stats.c File Reference

Statistics functions for TRDP communication.

```
#include <stdio.h>
#include <string.h>
#include "trdp_stats.h"
#include "trdp_if_light.h"
#include "trdp_if.h"
#include "trdp_private.h"
#include "trdp_pdcom.h"
#include "vos_mem.h"
#include "vos_thread.h"
Include dependency graph for trdp_stats.c:
```

Functions

- void **`trdp_UpdateStats`** (**`TRDP_APP_SESSION_T`** appHandle)
Update the statistics.
- void **`trdp_initStats`** (**`TRDP_APP_SESSION_T`** appHandle)
Init statistics.
- EXT_DECL **`TRDP_ERR_T tlc_resetStatistics`** (**`TRDP_APP_SESSION_T`** appHandle)

Reset statistics.

- 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** *pNumSubs, **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_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** *plpAddr)

Return join statistics.

- void **trdp_pdPrepareStats** (**TRDP_APP_SESSION_T** appHandle, **PD_ELE_T** *pPacket)

Fill the statistics packet.

5.23.1 Detailed Description

Statistics functions for TRDP communication.

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, 2013. All rights reserved.

Id:

trdp_stats.c (p. 145) 1336 2014-09-30 07:25:02Z ahweiss

5.23.2 Function Documentation

5.23.2.1 EXT_DECL TRDP_ERR_T tlc_getJoinStatistics (TRDP_APP_SESSION_T appHandle, UINT16 * pNumJoin, UINT32 * plpAddr)

Return join statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumJoin</i>	Pointer to the number of joined IP Addresses
out	<i>plpAddr</i>	Pointer to a list with the joined IP addresses

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more items than requested

Here is the call graph for this function:

5.23.2.2 **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.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumPub</i>	Pointer to the number of publishers
out	<i>pStatistics</i>	Pointer to a list with the publish statistics information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Here is the call graph for this function:

5.23.2.3 **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.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pNumRed</i>	Pointer to the number of redundancy groups
out	<i>pStatistics</i>	Pointer to a list with the redundancy group information

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_NOINIT_ERR</i>	handle invalid
<i>TRDP_PARAM_ERR</i>	parameter error
<i>TRDP_MEM_ERR</i>	there are more subscriptions than requested

Here is the call graph for this function:

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

Return statistics.

Memory for statistics information must be provided by the user.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
out	<i>pStatistics</i>	Pointer to statistics for this application session

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error

Here is the call graph for this function:

5.23.2.5 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.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
in, out	<i>pNumSubs</i>	In: The number of subscriptions requested Out: Number of subscriptions returned
in, out	<i>pStatistics</i>	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

Here is the call graph for this function:

5.23.2.6 EXT_DECL TRDP_ERR_T tlc_resetStatistics (TRDP_APP_SESSION_T *appHandle*)

Reset statistics.

Parameters

in	<i>appHandle</i>	the handle returned by tlc_openSession
----	------------------	--

Return values

TRDP_NO_ERR	no error
TRDP_NOINIT_ERR	handle invalid
TRDP_PARAM_ERR	parameter error

Here is the call graph for this function:

5.23.2.7 void trdp_initStats (TRDP_APP_SESSION_T *appHandle*)

Init statistics.

Clear the stats structure for a session.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

< host name

< leader host name

Here is the call graph for this function:

5.23.2.8 void trdp_pdPrepareStats (TRDP_APP_SESSION_T appHandle, PD_ELE_T * pPacket)

Fill the statistics packet.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pPacket</i>	pointer to the packet to fill

Here is the call graph for this function:

5.23.2.9 void trdp_UpdateStats (TRDP_APP_SESSION_T appHandle)

Update the statistics.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

Here is the call graph for this function:

5.24 trdp_stats.h File Reference

Statistics for TRDP communication.

```
#include "trdp_if_light.h"
#include "trdp_private.h"
#include "vos_utils.h"
```

Include dependency graph for `trdp_stats.h`: This graph shows which files directly or indirectly include this file:**Functions**

- void **trdp_initStats** (TRDP_APP_SESSION_T appHandle)
Init statistics.
- void **trdp_pdPrepareStats** (TRDP_APP_SESSION_T appHandle, PD_ELE_T *pPacket)
Fill the statistics packet.

5.24.1 Detailed Description

Statistics for TRDP communication.

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, 2013. All rights reserved.

Id:

trdp_stats.h (p. 150) 1065 2013-09-06 08:12:09Z aweiss

5.24.2 Function Documentation

5.24.2.1 void trdp_initStats (TRDP_APP_SESSION_T appHandle)

Init statistics.

Clear the stats structure for a session.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
----	------------------	---

< host name

< leader host name

Here is the call graph for this function:

5.24.2.2 void trdp_pdPrepareStats (TRDP_APP_SESSION_T appHandle, PD_ELE_T * pPacket)

Fill the statistics packet.

Parameters

in	<i>appHandle</i>	the handle returned by <code>tlc_openSession</code>
in, out	<i>pPacket</i>	pointer to the packet to fill

Here is the call graph for this function:

5.25 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_VERSION_T**
Version information.
- 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_MEM_STATISTICS_T**
TRDP statistics type definitions.
- 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 = error, 0x01 = false, 0x02 = true, 0x03 undefined)

Typedefs

- typedef VOS_IP4_ADDR_T **TRDP_IP_ADDR_T**
TRDP general type definitions.
- 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, UINT8 *pDst, UINT32 *pDstSize, **TRDP_DATASET_T** **ppCachedDS)

Function type for marshallng .

- typedef **TRDP_ERR_T**(* **TRDP_UNMARSHALL_T**)(void *pRefCon, UINT32 comId, UINT8 *pSrc, 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_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 }
- enum **TRDP_RED_STATE_T** {
 - TRDP_RED_FOLLOWER = 0,
 - TRDP_RED_LEADER = 1 }
- enum **TRDP_TO_BEHAVIOR_T** {
 - TRDP_TO_DEFAULT = 0,
 - TRDP_TO_SET_TO_ZERO = 1,
 - TRDP_TO_KEEP_LAST_VALUE = 2 }

Redundancy states.

How invalid PD shall be handled.

```

• enum TRDP_DATA_TYPE_T {
    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.

5.25.1 Detailed Description

Typedefs for TRDP communication. F

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, 2013. All rights reserved.

Id:

trdp_types.h (p. 151) 1350 2014-11-06 12:41:20Z ahweiss

BL 2014-07-14: Ticket #46: Protocol change: operational topocount needed
 BL 2014-02-27: Ticket #17: tlp_subscribe() returns wrong *pSubHandle

5.25.2 Typedef Documentation

5.25.2.1 typedef VOS_IP4_ADDR_T TRDP_IP_ADDR_T

TRDP general type definitions.

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

Function type for marshalling .

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

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provided buffer to small
<i>TRDP_COMID_ERR</i>	comid not existing

5.25.2.3 `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.

Parameters

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

5.25.2.4 `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.

Parameters

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

5.25.2.5 `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.

5.25.2.6 `typedef VOS_TIME_T TRDP_TIME_T`

Timer value compatible with timeval / select.

Relative or absolute date, depending on usage

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

Function type for unmarshalling.

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

Parameters

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

Return values

<i>TRDP_NO_ERR</i>	no error
<i>TRDP_MEM_ERR</i>	provide buffer to small
<i>TRDP_COMID_ERR</i>	comid not existing

5.25.3 Enumeration Type Documentation

5.25.3.1 enum TRDP_DATA_TYPE_T

TRDP dataset description definitions.

Dataset element definition

Enumerator

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.

5.25.3.2 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** Timeout.
- 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 endianness 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.
- 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_UNKNOWN_ERR** Unspecified error.

5.25.3.3 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.

5.25.3.4 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()

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.

5.25.3.5 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.

5.25.3.6 enum TRDP_REPLY_STATUS_T

TRDP data transfer type definitions.

Reply status messages

5.25.3.7 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.

5.26 trdp_utils.c File Reference

Helper functions for TRDP communication.

```
#include <string.h>
#include "trdp_if.h"
#include "trdp_utils.h"
Include dependency graph for trdp_utils.c:
```

Functions

- void **printSocketUsage** (TRDP_SOCKETS_T iface[])
Debug socket usage output.
- BOOL8 **trdp_SockIsJoined** (const TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)
Check if a mc group is in the list.
- BOOL8 **trdp_SockAddJoin** (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)
Add mc group to the list.
- BOOL8 **trdp_SockDelJoin** (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)
remove mc group from the list
- UINT32 **trdp_packetSizePD** (UINT32 dataSize)
Get the packet size from the raw data size.
- UINT32 **trdp_packetSizeMD** (UINT32 dataSize)
Get the packet size from the raw data size.
- PD_ELE_T * **trdp_queueFindComId** (PD_ELE_T *pHead, UINT32 comId)
Return the element with same comId.
- PD_ELE_T * **trdp_queueFindPubAddr** (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)
Return the element with same comId and IP addresses.
- PD_ELE_T * **trdp_queueFindSubAddr** (PD_ELE_T *pHead, TRDP_ADDRESSES_T *addr)
Return the element with same comId and IP addresses.
- void **trdp_queueDelElement** (PD_ELE_T **ppHead, PD_ELE_T *pDelete)
Delete an element.
- void **trdp_queueAppLast** (PD_ELE_T **ppHead, PD_ELE_T *pNew)
Append an element at end of queue.
- void **trdp_queueInsFirst** (PD_ELE_T **ppHead, PD_ELE_T *pNew)
Insert an element at front of queue.
- void **trdp_initSockets** (TRDP_SOCKETS_T iface[])
Handle the socket pool: Initialize it.
- TRDP_ERR_T **trdp_requestSocket** (TRDP_SOCKETS_T iface[], UINT32 port, const TRDP_SEND_PARAM_T *params, TRDP_IP_ADDR_T srcIp, TRDP_IP_ADDR_T mcGroup, TRDP SOCK_TYPE_T usage, TRDP_OPTION_T options, BOOL8 rcvMostly, INT32 useSocket, INT32 *pIndex, TRDP_IP_ADDR_T cornerIp)
Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.
- void **trdp_releaseSocket** (TRDP_SOCKETS_T iface[], INT32 lIndex, UINT32 connectTimeout, BOOL8 checkAll)
Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.
- UINT32 **trdp_getSeqCnt** (UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIpAddr)
Get the initial sequence counter for the comId/message type and subnet (source IP).
- void **trdp_resetSequenceCounter** (PD_ELE_T *pElement, TRDP_IP_ADDR_T srcIp, TRDP_MSG_T msgType)

remove the sequence counter for the comID/source IP.

- int **trdp_checkSequenceCounter** (PD_ELE_T *pElement, UINT32 sequenceCounter, TRDP_IP_ADDR_T srcIP, TRDP_MSG_T msgType)

check and update the sequence counter for the comID/source IP.

- BOOL8 **trdp_isAddressed** (const TRDP_URI_USER_T listUri, const TRDP_URI_USER_T destUri)

Check if listener URI is in addressing range of destination URI.

5.26.1 Detailed Description

Helper functions for TRDP communication.

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, 2013. All rights reserved.

Id:

trdp_utils.c (p. 160) 1331 2014-09-22 11:12:05Z railroad-mike

BL 2014-08-25: Ticket #57+58: Padding / zero bytes trailing MD & PD packets fixed
BL 2014-06-02: Ticket #41: Sequence counter handling fixed

5.26.2 Function Documentation

5.26.2.1 void printSocketUsage (TRDP_SOCKETS_T iface[])

Debug socket usage output.

Parameters

in	iface[]	List of sockets
----	---------	-----------------

5.26.2.2 int trdp_checkSequenceCounter (PD_ELE_T * pElement, UINT32 sequenceCounter, TRDP_IP_ADDR_T srcIP, TRDP_MSG_T msgType)

check and update the sequence counter for the comID/source IP.

If the comID/srcIP is not found, update it and return 0 - else if already received, return 1 On memory error, return -1

Parameters

in	pElement	subscription element
----	----------	----------------------

in	<i>sequence-Counter</i>	sequence counter to check
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	type of the message

Return values

0	- no duplicate 1 - duplicate sequence counter -1 - memory error
---	---

Here is the call graph for this function:

5.26.2.3 UINT32 trdp_getSeqCnt (UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIpAddr)

Get the initial sequence counter for the comID/message type and subnet (source IP).

If the comID/srcIP is not found elsewhere, return 0 - else return its current sequence number (the redundant packet needs the same seqNo)

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

Parameters

in	<i>comId</i>	comID to look for
in	<i>msgType</i>	PD/MD type
in	<i>srcIpAddr</i>	Source IP address

Return values

<i>return</i>	the sequence number
---------------	---------------------

Here is the call graph for this function:

5.26.2.4 void trdp_initSockets (TRDP_SOCKETS_T iface[])

Handle the socket pool: Initialize it.

Parameters

in	<i>iface</i>	pointer to the socket pool
----	--------------	----------------------------

5.26.2.5 BOOL8 trdp_isAddressed (const TRDP_URI_USER_T listUri, const TRDP_URI_USER_T destUri)

Check if listener URI is in addressing range of destination URI.

Parameters

in	<i>listUri</i>	Null terminated listener URI string to compare
in	<i>destUri</i>	Null terminated destination URI string to compare

Return values

<i>FALSE</i>	- not in addressing range
<i>TRUE</i>	- listener URI is in addressing range of destination URI

Here is the call graph for this function:

5.26.2.6 UINT32 trdp_packetSizeMD (UINT32 dataSize)

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size
----	-----------------	---------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.26.2.7 `UINT32 trdp_packetSizePD (UINT32 dataSize)`

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size
----	-----------------	---------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.26.2.8 `void trdp_queueAppLast (PD_ELE_T ** ppHead, PD_ELE_T * pNew)`

Append an element at end of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to append

5.26.2.9 `void trdp_queueDelElement (PD_ELE_T ** ppHead, PD_ELE_T * pDelete)`

Delete an element.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pDelete</i>	pointer to element to delete

5.26.2.10 `PD_ELE_T* trdp_queueFindComId (PD_ELE_T * pHead, UINT32 comId)`

Return the element with same comId.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>comId</i>	ComID to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.26.2.11 `PD_ELE_T* trdp_queueFindPubAddr (PD_ELE_T * pHead, TRDP_ADDRESSES_T * addr)`

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.26.2.12 PD_ELE_T* trdp_queueFindSubAddr (PD_ELE_T * *pHead*, TRDP_ADDRESSES_T * *addr*)

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.26.2.13 void trdp_queueInsFirst (PD_ELE_T ** *ppHead*, PD_ELE_T * *pNew*)

Insert an element at front of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to insert

5.26.2.14 void trdp_releaseSocket (TRDP_SOCKETS_T *iface*[], INT32 *lIndex*, UINT32 *connectTimeout*, BOOL8 *checkAll*)

Handle the socket pool: if a received TCP socket is unused, the socket connection timeout is started.

Handle the socket pool: Release a socket from our socket pool.

In Udp, Release a socket from our socket pool

Parameters

in, out	<i>iface</i>	socket pool
in	<i>lIndex</i>	index of socket to release
in	<i>connectTimeout</i>	time out
in	<i>checkAll</i>	release all TCP pending sockets

Here is the call graph for this function:

5.26.2.15 TRDP_ERR_T trdp_requestSocket (TRDP_SOCKETS_T *iface*[], UINT32 *port*, const TRDP_SEND_PARAM_T * *params*, TRDP_IP_ADDR_T *srcIP*, TRDP_IP_ADDR_T *mcGroup*, TRDP SOCK_TYPE_T *usage*, TRDP_OPTION_T *options*, BOOL8 *rcvMostly*, INT32 *useSocket*, INT32 * *pIndex*, TRDP_IP_ADDR_T *cornerIp*)

Handle the socket pool: Request a socket from our socket pool First we loop through the socket pool and check if there is already a socket which would suit us.

Handle the socket pool: Request a socket from our socket pool.

If a multicast group should be joined, we do that on an otherwise suitable socket - up to 20 multicast groups can be joined per socket. If a socket for multicast publishing is requested, we also use the source IP to determine the interface for outgoing multicast traffic.

Parameters

in, out	<i>iface</i>	socket pool
in	<i>port</i>	port to use
in	<i>params</i>	parameters to use
in	<i>srcIP</i>	IP to bind to (0 = any address)
in	<i>mcGroup</i>	MC group to join (0 = do not join)
in	<i>usage</i>	type and port to bind to (PD, MD/UDP, MD/TCP)
in	<i>options</i>	blocking/nonblocking
in	<i>rcvMostly</i>	primarily used for receiving (tbd: bind on sender, too?)
out	<i>useSocket</i>	socket to use, do not open a new one
out	<i>pIndex</i>	returned index of socket pool
in	<i>cornerIp</i>	only used for receiving

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_PARAM_ERR</i>	

Here is the call graph for this function:

5.26.2.16 void trdp_resetSequenceCounter (PD_ELE_T * *pElement*, TRDP_IP_ADDR_T *srcIP*, TRDP_MSG_T *msgType*)

remove the sequence counter for the comID/source IP.

The sequence counter should be reset if there was a packet time out.

Parameters

in	<i>pElement</i>	subscription element
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	message type

Return values

<i>none</i>	
-------------	--

5.26.2.17 BOOL8 trdp_SockAddJoin (TRDP_IP_ADDR_T *mcList[VOS_MAX_MULTICAST_CNT]*, TRDP_IP_ADDR_T *mcGroup*)

Add mc group to the list.

Parameters

in	<i>mcList[]</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

<i>1</i>	if added 0 if list is full
----------	----------------------------

5.26.2.18 `BOOL8 trdp_SockDelJoin (TRDP_IP_ADDR_T mcList[VOS_MAX_MULTICAST_CNT], TRDP_IP_ADDR_T mcGroup)`

remove mc group from the list

Parameters

in	<i>mcList[]</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if deleted 0 was not in list
---	------------------------------

5.26.2.19 **BOOL8** trdp_SockIsJoined (const **TRDP_IP_ADDR_T** *mcList[VOS_MAX_MULTICAST_CNT]*,
TRDP_IP_ADDR_T *mcGroup*)

Check if a mc group is in the list.

Parameters

in	<i>mcList[]</i>	List of multicast groups
in	<i>mcGroup</i>	multicast group

Return values

1	if found 0 if not found
---	-------------------------

5.27 trdp_utils.h File Reference

Common utilities for TRDP communication.

```
#include <stdio.h>
#include "trdp_private.h"
#include "vos_utils.h"
#include "vos_sock.h"
```

Include dependency graph for trdp_utils.h: This graph shows which files directly or indirectly include this file:

Functions

- **PD_ELE_T * trdp_queueFindComId** (**PD_ELE_T** *pHead, **UINT32** comId)
Return the element with same comId.
- **PD_ELE_T * trdp_queueFindSubAddr** (**PD_ELE_T** *pHead, **TRDP_ADDRESSES_T** *pAddr)
Return the element with same comId and IP addresses.
- **PD_ELE_T * trdp_queueFindPubAddr** (**PD_ELE_T** *pHead, **TRDP_ADDRESSES_T** *addr)
Return the element with same comId and IP addresses.
- **void trdp_queueDelElement** (**PD_ELE_T** **pHead, **PD_ELE_T** *pDelete)
Delete an element.
- **void trdp_queueAppLast** (**PD_ELE_T** **pHead, **PD_ELE_T** *pNew)
Append an element at end of queue.
- **void trdp_queueInsFirst** (**PD_ELE_T** **pHead, **PD_ELE_T** *pNew)
Insert an element at front of queue.
- **void trdp_initSockets** (**TRDP_SOCKETS_T** iface[])
Handle the socket pool: Initialize it.
- **void trdp_initUncompletedTCP** (**TRDP_APP_SESSION_T** appHandle)
???
- **void trdp_resetSequenceCounter** (**PD_ELE_T** *pElement, **TRDP_IP_ADDR_T** srcIP, **TRDP_MSG_T** msg-
Type)
remove the sequence counter for the comID/source IP.

- **TRDP_ERR_T trdp_requestSocket** (**TRDP_SOCKETS_T** iface[], **UINT32** port, const **TRDP_SEND_PARAM_T** *params, **TRDP_IP_ADDR_T** srcIP, **TRDP_IP_ADDR_T** mcGroup, **TRDP SOCK_TYPE_T** usage, **TRDP_OPTION_T** options, **BOOL8** rcvMostly, **INT32** useSocket, **INT32** *pIndex, **TRDP_IP_ADDR_T** cornerIp)
Handle the socket pool: Request a socket from our socket pool.
- void **trdp_releaseSocket** (**TRDP_SOCKETS_T** iface[], **INT32** lIndex, **UINT32** connectTimeout, **BOOL8** checkAll)
Handle the socket pool: Release a socket from our socket pool.
- **UINT32 trdp_packetSizePD** (**UINT32** dataSize)
Get the packet size from the raw data size.
- **UINT32 trdp_packetSizeMD** (**UINT32** dataSize)
Get the packet size from the raw data size.
- **UINT32 trdp_getSeqCnt** (**UINT32** comID, **TRDP_MSG_T** msgType, **TRDP_IP_ADDR_T** srcIP)
Get the initial sequence counter for the comID/message type and subnet (source IP).
- int **trdp_checkSequenceCounter** (**PD_ELE_T** *pElement, **UINT32** sequenceCounter, **TRDP_IP_ADDR_T** srcIP, **TRDP_MSG_T** msgType)
check and update the sequence counter for the comID/source IP.
- **BOOL8 trdp_isAddressed** (const **TRDP_URI_USER_T** listUri, const **TRDP_URI_USER_T** destUri)
Check if listener URI is in addressing range of destination URI.

5.27.1 Detailed Description

Common utilities for TRDP communication.

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, 2013. All rights reserved.

Id:

trdp_utils.h (p. 167) 1353 2014-11-11 15:11:13Z ahweiss

5.27.2 Function Documentation

5.27.2.1 int **trdp_checkSequenceCounter** (**PD_ELE_T** * pElement, **UINT32** sequenceCounter, **TRDP_IP_ADDR_T** srcIP, **TRDP_MSG_T** msgType)

check and update the sequence counter for the comID/source IP.

If the comID/srcIP is not found, update it and return 0 - else if already received, return 1 On memory error, return -1

Parameters

in	<i>pElement</i>	subscription element
in	<i>sequence-Counter</i>	sequence counter to check
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	type of the message

Return values

<i>0</i>	- no duplicate 1 - duplicate sequence counter -1 - memory error
----------	---

Here is the call graph for this function:

5.27.2.2 `UINT32 trdp_getSeqCnt (UINT32 comId, TRDP_MSG_T msgType, TRDP_IP_ADDR_T srcIpAddr)`

Get the initial sequence counter for the comID/message type and subnet (source IP).

If the comID/srcIP is not found elsewhere, return 0 - else return its current sequence number (the redundant packet needs the same seqNo)

Note: The standard demands that sequenceCounter is managed per comID/msgType at each publisher, but shall be the same for redundant telegrams (subnet/srcIP).

Parameters

in	<i>comId</i>	comID to look for
in	<i>msgType</i>	PD/MD type
in	<i>srcIpAddr</i>	Source IP address

Return values

<i>return</i>	the sequence number
---------------	---------------------

Here is the call graph for this function:

5.27.2.3 `void trdp_initSockets (TRDP_SOCKETS_T iface[])`

Handle the socket pool: Initialize it.

Parameters

in	<i>iface</i>	pointer to the socket pool
----	--------------	----------------------------

5.27.2.4 `void trdp_initUncompletedTCP (TRDP_APP_SESSION_T appHandle)`

???

Parameters

in	<i>appHandle</i>	session handle
----	------------------	----------------

5.27.2.5 `BOOL8 trdp_isAddressed (const TRDP_URI_USER_T listUri, const TRDP_URI_USER_T destUri)`

Check if listener URI is in addressing range of destination URI.

Parameters

in	<i>listUri</i>	Null terminated listener URI string to compare
in	<i>destUri</i>	Null terminated destination URI string to compare

Return values

<i>FALSE</i>	- not in addressing range
<i>TRUE</i>	- listener URI is in addressing range of destination URI

Here is the call graph for this function:

5.27.2.6 UINT32 trdp_packetSizeMD (UINT32 dataSize)

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size
----	-----------------	---------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.27.2.7 UINT32 trdp_packetSizePD (UINT32 dataSize)

Get the packet size from the raw data size.

Parameters

in	<i>dataSize</i>	net data size
----	-----------------	---------------

Return values

<i>packet</i>	size the size of the complete packet to be sent or received
---------------	---

5.27.2.8 void trdp_queueAppLast (PD_ELE_T ** ppHead, PD_ELE_T * pNew)

Append an element at end of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to append

5.27.2.9 void trdp_queueDelElement (PD_ELE_T ** ppHead, PD_ELE_T * pDelete)

Delete an element.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pDelete</i>	pointer to element to delete

5.27.2.10 PD_ELE_T* trdp_queueFindComId (PD_ELE_T * pHead, UINT32 comId)

Return the element with same comId.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>comId</i>	ComID to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.27.2.11 PD_ELE_T* trdp_queueFindPubAddr (PD_ELE_T * pHead, TRDP_ADDRESSES_T * addr)

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.27.2.12 PD_ELE_T* trdp_queueFindSubAddr (PD_ELE_T * pHead, TRDP_ADDRESSES_T * addr)

Return the element with same comId and IP addresses.

Parameters

in	<i>pHead</i>	pointer to head of queue
in	<i>addr</i>	Pub/Sub handle (Address, ComID, srcIP & dest IP) to search for

Return values

<i>!=</i>	NULL pointer to PD element
<i>NULL</i>	No PD element found

5.27.2.13 void trdp_queueInsFirst (PD_ELE_T ** ppHead, PD_ELE_T * pNew)

Insert an element at front of queue.

Parameters

in	<i>ppHead</i>	pointer to pointer to head of queue
in	<i>pNew</i>	pointer to element to insert

5.27.2.14 void trdp_releaseSocket (TRDP_SOCKETS_T iface[], INT32 lIndex, UINT32 connectTimeout, BOOL8 checkAll)

Handle the socket pool: Release a socket from our socket pool.

Parameters

in, out	<i>iface</i>	socket pool
---------	--------------	-------------

in	<i>lIndex</i>	index of socket to release
in	<i>connectTimeout</i>	timeout value
in	<i>checkAll</i>	release all TCP pending sockets

Handle the socket pool: Release a socket from our socket pool.

In Udp, Release a socket from our socket pool

Parameters

in, out	<i>iface</i>	socket pool
in	<i>lIndex</i>	index of socket to release
in	<i>connectTimeout</i>	time out
in	<i>checkAll</i>	release all TCP pending sockets

Here is the call graph for this function:

5.27.2.15 `TRDP_ERR_T trdp_requestSocket (TRDP_SOCKETS_T iface[], UINT32 port, const TRDP_SEND_PARAM_T * params, TRDP_IP_ADDR_T srcIP, TRDP_IP_ADDR_T mcGroup, TRDP SOCK_TYPE_T usage, TRDP_OPTION_T options, BOOL8 rcvMostly, INT32 useSocket, INT32 * pIndex, TRDP_IP_ADDR_T cornerIp)`

Handle the socket pool: Request a socket from our socket pool.

Parameters

in, out	<i>iface</i>	socket pool
in	<i>port</i>	port to use
in	<i>params</i>	parameters to use
in	<i>srcIP</i>	IP to bind to (0 = any address)
in	<i>mcGroup</i>	MC group to join (0 = do not join)
in	<i>usage</i>	type and port to bind to
in	<i>options</i>	blocking/nonblocking
in	<i>rcvMostly</i>	only used for receiving
out	<i>useSocket</i>	socket to use, do not open a new one
out	<i>pIndex</i>	returned index of socket pool
in	<i>cornerIp</i>	only used for receiving

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_PARAM_ERR</i>	Handle the socket pool: Request a socket from our socket pool.

If a multicast group should be joined, we do that on an otherwise suitable socket - up to 20 multicast groups can be joined per socket. If a socket for multicast publishing is requested, we also use the source IP to determine the interface for outgoing multicast traffic.

Parameters

in, out	<i>iface</i>	socket pool
in	<i>port</i>	port to use
in	<i>params</i>	parameters to use
in	<i>srcIP</i>	IP to bind to (0 = any address)
in	<i>mcGroup</i>	MC group to join (0 = do not join)
in	<i>usage</i>	type and port to bind to (PD, MD/UDP, MD/TCP)
in	<i>options</i>	blocking/nonblocking
in	<i>rcvMostly</i>	primarily used for receiving (tbd: bind on sender, too?)

out	<i>useSocket</i>	socket to use, do not open a new one
out	<i>pIndex</i>	returned index of socket pool
in	<i>cornerIp</i>	only used for receiving

Return values

<i>TRDP_NO_ERR</i>	
<i>TRDP_PARAM_ERR</i>	

Here is the call graph for this function:

5.27.2.16 void trdp_resetSequenceCounter (PD_ELE_T * *pElement*, TRDP_IP_ADDR_T *srcIP*, TRDP_MSG_T *msgType*)

remove the sequence counter for the comID/source IP.

The sequence counter should be reset if there was a packet time out.

Parameters

in	<i>pElement</i>	subscription element
in	<i>srcIP</i>	Source IP address
in	<i>msgType</i>	message type

Return values

<i>none</i>	
-------------	--

5.28 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

- **VOS_ERR_T vos_mutexLocalCreate** (struct VOS_MUTEX *pMutex)
Create a recursive mutex.
- void **vos_mutexLocalDelete** (struct VOS_MUTEX *pMutex)
Delete a mutex.
- 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 **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.

5.28.1 Detailed Description

Memory functions. OS abstraction of memory access and control

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, 2013. All rights reserved.

Id:

vos_mem.c (p. 173) 1323 2014-08-29 14:09:08Z bloehr

Changes: BL 2012-12-03: ID 1: "using uninitialized PD_ELE_T.pullIpAddress variable" ID 2: "uninitialized PD_ELE_T newPD->pNext in tlp_subscribe()"

5.28.2 Function Documentation

5.28.2.1 `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	<i>pKey</i>	Key to search for
in	<i>pBuf</i>	Pointer to the array to sort
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	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

<i>Pointer</i>	to found element or NULL
----------------	--------------------------

5.28.2.2 `EXT_DECL UINT8* vos_memAlloc (UINT32 size)`

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

Parameters

in	<i>size</i>	Size of requested block
----	-------------	-------------------------

Return values

<i>Pointer</i>	to memory area
<i>NULL</i>	if no memory available

Here is the call graph for this function:

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

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised

5.28.2.4 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	<i>pMemoryArea</i>	Pointer to memory area used
----	--------------------	-----------------------------

Here is the call graph for this function:

5.28.2.5 EXT_DECL void vos_memFree (void * *pMemBlock*)

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

Parameters

in	<i>pMemBlock</i>	Pointer to memory block to be freed
----	------------------	-------------------------------------

Here is the call graph for this function:

5.28.2.6 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	<i>pMemoryArea</i>	Pointer to memory area to use
in	<i>size</i>	Size of provided memory area
in	<i>fragMem</i>	Pointer to list of preallocated block sizes, used to fragment memory for large blocks

Return values

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

Here is the call graph for this function:

5.28.2.7 VOS_ERR_T vos_mutexLocalCreate (struct VOS_MUTEX * *pMutex*)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

Parameters

out	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

5.28.2.8 void vos_mutexLocalDelete (struct VOS_MUTEX * *pMutex*)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	Pointer to mutex struct
----	---------------	-------------------------

5.28.2.9 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	<i>pBuf</i>	Pointer to the array to sort
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	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

<i>none</i>

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

Return values

<i>VOS_NO_ERR</i>	no error
-------------------	----------

<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_INIT_ERR</i>	not supported
<i>VOS_QUEUE_ERR</i>	error creating queue

Here is the call graph for this function:

5.28.2.11 EXT_DECL VOS_ERR_T vos_queueDestroy (VOS_QUEUE_T queueHandle)

Destroy a message queue.

Free all resources used by this queue

Parameters

in	<i>queueHandle</i>	Queue handle
----	--------------------	--------------

Return values

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

Here is the call graph for this function:

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

Get a message.

Parameters

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

Return values

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

Here is the call graph for this function:

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

Send a message.

Parameters

in	<i>queueHandle</i>	Queue handle
----	--------------------	--------------

in	<i>pData</i>	Pointer to data to be sent
in	<i>size</i>	Size of data to be sent

Return values

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

Here is the call graph for this function:

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

String copy with length limitation.

Parameters

in	<i>pStrDst</i>	Destination string
in	<i>pStrSrc</i>	Null terminated string to copy
in	<i>count</i>	Maximum number of characters to copy

Return values

<i>none</i>

5.28.2.15 EXT_DECL INT32 vos_strncmp (const CHAR8 * *pStr1*, const CHAR8 * *pStr2*, UINT32 *count*)

Case insensitive string compare.

Parameters

in	<i>pStr1</i>	Null terminated string to compare
in	<i>pStr2</i>	Null terminated string to compare
in	<i>count</i>	Maximum number of characters to compare

Return values

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

5.29 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, 0, 8, 0, 0, 1, 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 **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.

5.29.1 Detailed Description

Memory and queue functions for OS abstraction. This module provides memory control supervisor

Note

Project: TCNOpen TRDP prototype stack

Author

Bernd Loehr, NewTec GmbH Peter Brander (Memory scheme)

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:

vos_mem.h (p. 179) 1323 2014-08-29 14:09:08Z bloehr

5.29.2 Macro Definition Documentation

5.29.2.1 #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 overall size of the used memory allocation area is larger.

5.29.2.2 #define VOS_MEM_PREALLOCATE {0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 1, 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.

5.29.3 Function Documentation

5.29.3.1 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	<i>pKey</i>	Key to search for
----	-------------	-------------------

in	<i>pBuf</i>	Pointer to the array to sort
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	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

<i>Pointer</i>	to found element or NULL
----------------	--------------------------

5.29.3.2 EXT_DECL UINT8* vos_memAlloc (UINT32 size)

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

Parameters

in	<i>size</i>	Size of requested block
----	-------------	-------------------------

Return values

<i>Pointer</i>	to memory area
<i>NULL</i>	if no memory available

Here is the call graph for this function:

5.29.3.3 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	<i>pAllocated-Memory</i>	Pointer to allocated memory size
out	<i>pFreeMemory</i>	Pointer to free memory size
out	<i>pMinFree</i>	Pointer to minimal free memory size in statistics interval
out	<i>pNumAlloc-Blocks</i>	Pointer to number of allocated memory blocks
out	<i>pNumAllocErr</i>	Pointer to number of allocation errors
out	<i>pNumFreeErr</i>	Pointer to number of free errors
out	<i>blockSize</i>	Pointer to list of memory block sizes
out	<i>usedBlockSize</i>	Pointer to list of used memoryblocks

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised

5.29.3.4 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

<i>in</i>	<i>pMemoryArea</i>	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

<i>in</i>	<i>pMemoryArea</i>	Pointer to memory area used
-----------	--------------------	-----------------------------

Here is the call graph for this function:

5.29.3.5 EXT_DECL void vos_memFree (void * *pMemBlock*)

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

Parameters

<i>in</i>	<i>pMemBlock</i>	Pointer to memory block to be freed
-----------	------------------	-------------------------------------

Here is the call graph for this function:

5.29.3.6 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

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

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_MEM_ERR</i>	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

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

Return values

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

<i>VOS_MUTEX_ERR</i>	no mutex available
----------------------	--------------------

Here is the call graph for this function:

5.29.3.7 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	<i>pBuf</i>	Pointer to the array to sort
in	<i>num</i>	number of elements
in	<i>size</i>	size of one element
in	<i>compare</i>	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

<i>none</i>	
-------------	--

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

Return values

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

Here is the call graph for this function:

5.29.3.9 EXT_DECL VOS_ERR_T vos_queueDestroy (VOS_QUEUE_T *queueHandle*)

Destroy a message queue.

Free all resources used by this queue

Parameters

in	<i>queueHandle</i>	Queue handle
----	--------------------	--------------

Return values

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

Here is the call graph for this function:

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

Get a message.

Parameters

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

Return values

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

Here is the call graph for this function:

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

Send a message.

Parameters

in	<i>queueHandle</i>	Queue handle
in	<i>pData</i>	Pointer to data to be sent
in	<i>size</i>	Size of data to be sent

Return values

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

Here is the call graph for this function:

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

String copy with length limitation.

Parameters

in	<i>pStrDst</i>	Destination string
in	<i>pStrSrc</i>	Null terminated string to copy
in	<i>count</i>	Maximum number of characters to copy

Return values

<i>none</i>

5.29.3.13 EXT_DECL INT32 vos_strncmp (const CHAR8 * *pStr1*, const CHAR8 * *pStr2*, UINT32 *count*)

Case insensitive string compare.

Parameters

in	<i>pStr1</i>	Null terminated string to compare
in	<i>pStr2</i>	Null terminated string to compare
in	<i>count</i>	Maximum number of characters to compare

Return values

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

5.30 vos_private.h File Reference

Private definitions for the OS abstraction layer.

```
#include <pthread.h>
#include <sys/types.h>
#include "vos_types.h"
#include "vos_thread.h"
```

Include dependency graph for posix/vos_private.h: This graph shows which files directly or indirectly include this file:

Functions

- **VOS_ERR_T vos_mutexLocalCreate** (struct VOS_MUTEX *pMutex)
Create a recursive mutex.
- void **vos_mutexLocalDelete** (struct VOS_MUTEX *pMutex)
Delete a mutex.

5.30.1 Detailed Description

Private definitions for the OS abstraction layer.

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, 2013. All rights reserved.

Id:

vos_private.h 1133 2013-12-18 08:00:43Z ahweiss

5.30.2 Function Documentation**5.30.2.1 VOS_ERR_T vos_mutexLocalCreate (struct VOS_MUTEX * pMutex)**

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

Parameters

out	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

5.30.2.2 void vos_mutexLocalDelete (struct VOS_MUTEX * pMutex)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	Pointer to mutex struct
----	---------------	-------------------------

5.31 vos_private.h File Reference

Private definitions for the OS abstraction layer.

```
#include <pthread.h>
#include "vos_types.h"
#include "vos_thread.h"
```

Include dependency graph for windows/vos_private.h: This graph shows which files directly or indirectly include this file:

Functions

- **VOS_ERR_T vos_mutexLocalCreate** (struct VOS_MUTEX *pMutex)
Create a recursive mutex.
- void **vos_mutexLocalDelete** (struct VOS_MUTEX *pMutex)
Delete a mutex.

5.31.1 Detailed Description

Private definitions for the OS abstraction layer.

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, 2013. All rights reserved.

Id:

vos_private.h 1065 2013-09-06 08:12:09Z aweiss

*

5.31.2 Function Documentation

5.31.2.1 VOS_ERR_T vos_mutexLocalCreate (struct VOS_MUTEX * *pMutex*)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

Parameters

out	<i>pMutex</i>	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

5.31.2.2 void vos_mutexLocalDelete (struct VOS_MUTEX * *pMutex*)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	Pointer to mutex struct
----	---------------	-------------------------

5.32 vos_shared_mem.c File Reference

Shared Memory functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include "vos_types.h"
#include "vos_mem.h"
#include "vos_utils.h"
#include "vos_private.h"
#include "vos_shared_mem.h"
Include dependency graph for posix/vos_shared_mem.c:
```

Functions

- 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.
- EXT_DECL **VOS_ERR_T vos_sharedClose** (VOS_SHRD_T handle, const UINT8 *pMemoryArea)
Close connection to the shared memory area.

5.32.1 Detailed Description

Shared Memory functions. OS abstraction of Shared memory access and control

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 (p. 179) 282 2013-01-11 07:08:44Z 97029

5.32.2 Function Documentation

5.32.2.1 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	<i>handle</i>	Returned handle
in	<i>pMemoryArea</i>	Pointer to memory area

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

5.32.2.2 **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 attached. This function is not available in each target implementation.

Parameters

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

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

Here is the call graph for this function:

5.33 vos_shared_mem.c File Reference

Shared Memory functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include "vos_shared_mem.h"
#include "vos_utils.h"
#include <windows.h>
#include <conio.h>
#include <tchar.h>
```

Include dependency graph for windows/vos_shared_mem.c:

Functions

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

Close connection to the shared memory area.

5.33.1 Detailed Description

Shared Memory functions. OS abstraction of Shared memory access and control

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 Bombardier Transportation Inc. or its subsidiaries and others, 2013. All rights reserved.

Id:

vos_sock.c 253 2013-01-07 13:48:40Z aweiss

*

5.33.2 Function Documentation

5.33.2.1 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	<i>handle</i>	Returned handle
in	<i>pMemoryArea</i>	Pointer to memory area

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

Here is the call graph for this function:

5.33.2.2 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	<i>pKey</i>	Unique identifier (file name)
out	<i>pHandle</i>	Pointer to returned handle
out	<i>ppMemoryArea</i>	Pointer to pointer to memory area
in, out	<i>pSize</i>	Pointer to size of area to allocate, on return actual size after attach. Independent from actual value, always multiples of page size (4k) are allocated

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

Here is the call graph for this function:

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

Functions

- 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.
- EXT_DECL **VOS_ERR_T vos_sharedClose** (VOS_SHRD_T handle, const UINT8 *pMemoryArea)
Close connection to the shared memory area.

5.34.1 Detailed Description

Shared Memory functions for OS abstraction. This module provides shared memory control supervision

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 (p. 179) 282 2013-01-11 07:08:44Z 97029

5.34.2 Function Documentation

5.34.2.1 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	<i>handle</i>	Returned handle
in	<i>pMemoryArea</i>	Pointer to memory area

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

Here is the call graph for this function:

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

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

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 attached. This function is not available in each target implementation.

Parameters

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

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

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	<i>pKey</i>	Unique identifier (file name)
out	<i>pHandle</i>	Pointer to returned handle
out	<i>ppMemoryArea</i>	Pointer to pointer to memory area
in, out	<i>pSize</i>	Pointer to size of area to allocate, on return actual size after attach. Independent from actual value, always multiples of page size (4k) are allocated

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_MEM_ERR</i>	no memory available

Here is the call graph for this function:

5.35 vos_sock.c File Reference

Socket functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <ifaddrs.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "vos_private.h"
```

Include dependency graph for posix/vos_sock.c:

Functions

- **BOOL8 vos_getMacAddress** (UINT8 *pMacAddr, const char *pIfName)
Get the MAC address for a named interface.
- **VOS_ERR_T vos_sockSetBuffer** (INT32 sock)
Enlarge send and receive buffers to TRDP_SOCKETBUF_SIZE if necessary.
- **EXT_DECL UINT16 vos_htons** (UINT16 val)
Byte swapping.
- **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 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_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 **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_SOCKET_OPT_T** *pOptions)
- Create an UDP socket.*

 - EXT_DECL **VOS_ERR_T vos_sockOpenTCP** (INT32 *pSock, const **VOS_SOCKET_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_SOCKET_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 *pSrcIPAddr, 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 connections.*

 - EXT_DECL **VOS_ERR_T vos_sockAccept** (INT32 sock, INT32 *pSock, UINT32 *pIPAddr, UINT16 *pPort)
- 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 mclAddress)
- Set Using Multicast I/F.*

5.35.1 Detailed Description

Socket functions. OS abstraction of IP socket functions for UDP and TCP

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, 2013. All rights reserved.

Id:

vos_sock.c 1332 2014-09-22 13:10:58Z railroad-mike

BL 2014-08-25: Ticket #51: change underlying function for vos_dottedIP

5.35.2 Function Documentation

5.35.2.1 EXT_DECL UINT32 vos_dottedIP (const CHAR8 * *pDottedIP*)

Convert IP address from dotted dec.

to !host! endianness

Parameters

in	<i>pDottedIP</i>	IP address as dotted decimal.
----	------------------	-------------------------------

Return values

<i>address</i>	in UINT32 in host endianness 0 (Zero) if error
----------------	--

Here is the call graph for this function:

5.35.2.2 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	<i>pAddrCnt</i>	in: pointer to array size of interface record out: pointer to number of interface records read
in, out	<i>ifAddrs</i>	array of interface records

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pMAC == NULL

Here is the call graph for this function:

5.35.2.3 BOOL8 vos_getMacAddress (UINT8 * *pMacAddr*, const char * *plfName*)

Get the MAC address for a named interface.

Parameters

out	<i>pMacAddr</i>	pointer to array of MAC address to return
in	<i>plfName</i>	pointer to interface name

Return values

<i>TRUE</i>	if successfull
-------------	----------------

5.35.2.4 EXT_DECL UINT32 vos_htonl (UINT32 val)

Byte swapping 4 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.35.2.5 EXT_DECL UINT16 vos_htons (UINT16 val)

Byte swapping.

Byte swapping 2 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.35.2.6 EXT_DECL const CHAR8* vos_ipDotted (UINT32 ipAddress)

Convert IP address to dotted dec.

from !host! endianness.

Parameters

in	<i>ipAddress</i>	address in UINT32 in host endianness
----	------------------	--------------------------------------

Return values

<i>IP</i>	address as dotted decimal.
-----------	----------------------------

5.35.2.7 EXT_DECL BOOL8 vos_isMulticast (UINT32 ipAddress)

Check if the supplied address is a multicast group address.

Parameters

<i>in</i>	<i>ipAddress</i>	IP address to check.
-----------	------------------	----------------------

Return values

<i>TRUE</i>	address is multicast
<i>FALSE</i>	address is not a multicast address

5.35.2.8 EXT_DECL UINT32 vos_ntohl (UINT32 *val*)

Byte swapping 4 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.35.2.9 EXT_DECL UINT16 vos_ntohs (UINT16 *val*)

Byte swapping 2 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.35.2.10 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

<i>in</i>	<i>highDesc</i>	max. socket descriptor + 1
<i>in, out</i>	<i>pReadableFD</i>	pointer to readable socket set
<i>in, out</i>	<i>pWriteableFD</i>	pointer to writeable socket set
<i>in, out</i>	<i>pErrorFD</i>	pointer to error socket set
<i>in</i>	<i>pTimeout</i>	pointer to time out value

Return values

<i>number</i>	of ready file descriptors
---------------	---------------------------

5.35.2.11 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	<i>sock</i>	Socket descriptor
out	<i>pSock</i>	Pointer to socket descriptor, on exit new socket
out	<i>pIPAddress</i>	source IP to receive on, 0 for any
out	<i>pPort</i>	port to receive on, 20548 for PD

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	NULL parameter, parameter error
<i>VOS_UNKNOWN_ERR</i>	sock descriptor unknown error

Here is the call graph for this function:

5.35.2.12 EXT_DECL VOS_ERR_T vos_sockBind (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Bind a socket to an address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>ipAddress</i>	source IP to receive on, 0 for any
in	<i>port</i>	port to receive on, 20548 for PD

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

Here is the call graph for this function:

5.35.2.13 EXT_DECL VOS_ERR_T vos_sockClose (INT32 *sock*)

Close a socket.

Release any resources acquired by this socket

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown

5.35.2.14 EXT_DECL VOS_ERR_T vos_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error

Here is the call graph for this function:

5.35.2.15 EXT_DECL VOS_ERR_T vos_sockGetMAC (UINT8 *pMAC*[VOS_MAC_SIZE])

Return the MAC address of the default adapter.

Parameters

out	<i>pMAC</i>	return MAC address.
-----	-------------	---------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pMAC</i> == NULL
<i>VOS SOCK_ERR</i>	socket not available or option not supported

Here is the call graph for this function:

5.35.2.16 EXT_DECL VOS_ERR_T vos_sockInit (void)

Initialize the socket library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS SOCK_ERR</i>	sockets not supported

5.35.2.17 EXT_DECL VOS_ERR_T vos_sockJoinMC (INT32 *sock*, UINT32 *mcAddress*, UINT32 *ipAddress*)

Join a multicast group.

Note: Some targeted systems might not support this option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to join, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS SOCK_ERR</i>	option not supported

Here is the call graph for this function:

5.35.2.18 EXT_DECL VOS_ERR_T vos_sockLeaveMC (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

Leave a multicast group.

Note: Some targeted 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_PARAM_ERR	sock descriptor unknown, parameter error
VOS_SOCK_ERR	option not supported

Here is the call graph for this function:

5.35.2.19 EXT_DECL VOS_ERR_T vos_sockListen (INT32 sock, UINT32 backlog)

Listen for incoming connections.

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	sock descriptor unknown, parameter error
VOS_IO_ERR	Input/Output error
VOS_MEM_ERR	resource error

5.35.2.20 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

Here is the call graph for this function:

5.35.2.21 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 targeted systems might not support every option.

Parameters

out	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pSock</i> == NULL
<i>VOS SOCK_ERR</i>	socket not available or option not supported

Here is the call graph for this function:

5.35.2.22 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 occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned.

Parameters

in	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

5.35.2.23 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 occurred. 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	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size
out	<i>pSrcIPAddr</i>	pointer to source IP
out	<i>pSrcIPPort</i>	pointer to source port

out	<i>pDstIPAddr</i>	pointer to dest IP
in	<i>peek</i>	if true, leave data in queue

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

Here is the call graph for this function:

5.35.2.24 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	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	In: size of the data to send, Out: no of bytes sent

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_NOCONN_ERR</i>	no TCP connection
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

5.35.2.25 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 supplied address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	In: size of the data to send, Out: no of bytes sent
in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

Here is the call graph for this function:

5.35.2.26 VOS_ERR_T vos_sockSetBuffer (INT32 sock)

Enlarge send and receive buffers to TRDP_SOCKETBUF_SIZE if necessary.

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_SOCK_ERR</i>	buffer size can't be set

5.35.2.27 EXT_DECL VOS_ERR_T vos_sockSetMulticastIf (INT32 *sock*, UINT32 *mclfAddress*)

Set Using Multicast I/F.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mclfAddress</i>	using Multicast I/F Address

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_SOCK_ERR</i>	option not supported

Here is the call graph for this function:

5.35.2.28 EXT_DECL VOS_ERR_T vos_sockSetOptions (INT32 *sock*, const VOS_SOCK_OPT_T * *pOptions*)

Set socket options.

Note: Some targeted systems might not support every option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown

5.35.2.29 EXT_DECL void vos_sockTerm (void)

De-Initialize the socket library.

Must be called after last socket call

5.36 vos_sock.c File Reference

Socket functions.

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <winsock2.h>
#include <Ws2tcpip.h>
#include <MSWSock.h>
#include <lm.h>
#include <iphlpapi.h>
#include "vos_utils.h"
#include "vos_sock.h"
#include "vos_thread.h"
#include "vos_mem.h"
```

Include dependency graph for windows/vos_sock.c:

Functions

- INT32 **recvmsg** (int sock, struct msghdr *pMessage, int flags)
Receive a message including sender address information.
- **VOS_ERR_T vos_sockSetBuffer** (INT32 sock)
Enlarge send and receive buffers to TRDP_SOCKETBUF_SIZE if necessary.
- EXT_DECL UINT16 **vos_htons** (UINT16 val)
Byte swapping.
- 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 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_SOCKET_OPT_T** *pOptions)
Create an UDP socket.
- EXT_DECL **VOS_ERR_T vos_sockOpenTCP** (INT32 *pSock, const **VOS_SOCKET_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 *pSrcIPAddr, 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 connections.

- EXT_DECL **VOS_ERR_T vos_sockAccept** (INT32 sock, INT32 *pSock, UINT32 *pIPAddress, UINT16 *pPort)

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_sockSetMulticastIff** (INT32 sock, UINT32 mclAddress)

Set Using Multicast I/F.

5.36.1 Detailed Description

Socket functions. OS abstraction of IP socket functions for UDP and TCP

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, 2013. All rights reserved.

Id:

vos_sock.c 1352 2014-11-11 15:06:54Z ahweiss

*

5.36.2 Function Documentation

5.36.2.1 INT32 recvmsg (int *sock*, struct msghdr * *pMessage*, int *flags*)

Receive a message including sender address information.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pMessage</i>	Pointer to message header
in	<i>flags</i>	Receive flags

Return values

<i>number</i>	of received bytes, -1 for error
---------------	---------------------------------

5.36.2.2 EXT_DECL UINT32 vos_dottedIP (const CHAR8 * *pDottedIP*)

Convert IP address from dotted dec.

to !host! endianness

Parameters

in	<i>pDottedIP</i>	IP address as dotted decimal.
----	------------------	-------------------------------

Return values

<i>address</i>	in UINT32 in host endianness
----------------	------------------------------

Here is the call graph for this function:

5.36.2.3 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	<i>pAddrCnt</i>	in: pointer to array size of interface record out: pointer to number of interface records read
in, out	<i>ifAddrs</i>	array of interface records

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pAddrCnt</i> and/or <i>ifAddrs</i> == NULL
<i>VOS_MEM_ERR</i>	memory allocation error
<i>VOS SOCK_ERR</i>	GetAdaptersInfo() error

Here is the call graph for this function:

5.36.2.4 EXT_DECL UINT32 vos_htonl (UINT32 *val*)

Byte swapping 4 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.36.2.5 EXT_DECL UINT16 vos_htons (UINT16 *val*)

Byte swapping.

Byte swapping 2 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.36.2.6 EXT_DECL const CHAR8* vos_ipDotted (UINT32 *ipAddress*)

Convert IP address to dotted dec.

from !host! endianness.

Parameters

<i>in</i>	<i>ipAddress</i>	address in UINT32 in host endianness
-----------	------------------	--------------------------------------

Return values

<i>IP</i>	address as dotted decimal.
-----------	----------------------------

5.36.2.7 EXT_DECL BOOL8 vos_isMulticast (UINT32 *ipAddress*)

Check if the supplied address is a multicast group address.

Parameters

<i>in</i>	<i>ipAddress</i>	IP address to check.
-----------	------------------	----------------------

Return values

<i>TRUE</i>	address is multicast
<i>FALSE</i>	address is not a multicast address

5.36.2.8 EXT_DECL UINT32 vos_ntohl (UINT32 *val*)

Byte swapping 4 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.36.2.9 EXT_DECL UINT16 vos_ntohs (UINT16 *val*)

Byte swapping 2 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.36.2.10 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	<i>highDesc</i>	max. socket descriptor + 1
in, out	<i>pReadableFD</i>	pointer to readable socket set
in, out	<i>pWriteableFD</i>	pointer to writeable socket set
in, out	<i>pErrorFD</i>	pointer to error socket set
in	<i>pTimeout</i>	pointer to time out value

Return values

<i>number</i>	of ready file descriptors
---------------	---------------------------

5.36.2.11 EXT_DECL VOS_ERR_T vos_sockAccept (INT32 *sock*, INT32 * *pSock*, UINT32 * *plPAddress*, 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	<i>sock</i>	Socket descriptor
out	<i>pSock</i>	Pointer to socket descriptor, on exit new socket
out	<i>plPAddress</i>	source IP to receive on, 0 for any
out	<i>pPort</i>	port to receive on, 20548 for PD

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	NULL parameter, parameter error
<i>VOS_UNKNOWN_ERR</i>	sock descriptor unknown error

Here is the call graph for this function:

5.36.2.12 EXT_DECL VOS_ERR_T vos_sockBind (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Bind a socket to an address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>ipAddress</i>	source IP to receive on, 0 for any
in	<i>port</i>	port to receive on, 20548 for PD

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

Here is the call graph for this function:

5.36.2.13 EXT_DECL VOS_ERR_T vos_sockClose (INT32 *sock*)

Close a socket.

Release any resources aquired by this socket

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown

5.36.2.14 EXT_DECL VOS_ERR_T vos_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

Here is the call graph for this function:

5.36.2.15 EXT_DECL VOS_ERR_T vos_sockGetMAC (UINT8 *pMAC*[VOS_MAC_SIZE])

Return the MAC address of the default adapter.

Parameters

out	<i>pMAC</i>	return MAC address.
-----	-------------	---------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pMAC</i> == NULL
<i>VOS_SOCKET_ERR</i>	socket not available or option not supported

5.36.2.16 EXT_DECL VOS_ERR_T vos_sockInit (void)

Initialize the socket library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_SOCK_ERR</i>	sockets not supported

5.36.2.17 EXT_DECL VOS_ERR_T vos_sockJoinMC (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

Join a multicast group.

Note: Some targeted systems might not support this option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to join, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_SOCK_ERR</i>	option not supported

Here is the call graph for this function:

5.36.2.18 EXT_DECL VOS_ERR_T vos_sockLeaveMC (INT32 sock, UINT32 mcAddress, UINT32 ipAddress)

Leave a multicast group.

Note: Some targeted systems might not support this option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to leave, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_SOCK_ERR</i>	option not supported

Here is the call graph for this function:

5.36.2.19 EXT_DECL VOS_ERR_T vos_sockListen (INT32 sock, UINT32 backlog)

Listen for incoming connections.

Listen for incoming TCP connections.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>backlog</i>	maximum connection attempts if system is busy

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

5.36.2.20 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	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pSock == NULL
<i>VOS_SOCK_ERR</i>	socket not available or option not supported

Here is the call graph for this function:

5.36.2.21 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 targeted systems might not support every option.

Parameters

out	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pSock == NULL
<i>VOS_SOCK_ERR</i>	socket not available or option not supported

Here is the call graph for this function:

5.36.2.22 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 occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned.

Parameters

in	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data
<i>VOS_BLOCK_ERR</i>	call would have blocked in blocking mode

5.36.2.23 `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 occurred. 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	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size
out	<i>pSrcIPAddr</i>	pointer to source IP
out	<i>pSrcIPPort</i>	pointer to source port
out	<i>pDstIPAddr</i>	pointer to dest IP
in	<i>peek</i>	if true, leave data in queue

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

Here is the call graph for this function:

5.36.2.24 `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	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	IN: bytes to send, OUT: bytes sent

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_NOCONN_ERR</i>	no TCP connection
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

5.36.2.25 **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 supplied address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	IN: bytes to send, OUT: bytes sent
in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

Here is the call graph for this function:

5.36.2.26 **VOS_ERR_T vos_sockSetBuffer** (INT32 *sock*)

Enlarge send and receive buffers to TRDP_SOCKETBUF_SIZE if necessary.

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_SOCKET_ERR</i>	buffer size can't be set

5.36.2.27 **EXT_DECL VOS_ERR_T vos_sockSetMulticastIf** (INT32 *sock*, UINT32 *mclfAddress*)

Set Using Multicast I/F.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mclfAddress</i>	using Multicast I/F Address

Return values

<code>VOS_NO_ERR</code>	no error
<code>VOS_PARAM_ERR</code>	sock descriptor unknown, parameter error

Here is the call graph for this function:

5.36.2.28 EXT_DECL VOS_ERR_T vos_sockSetOptions (INT32 sock, const VOS_SOCK_OPT_T * pOptions)

Set socket options.

Note: Some targeted systems might not support every option.

Parameters

in	<code>sock</code>	socket descriptor
in	<code>pOptions</code>	pointer to socket options (optional)

Return values

<code>VOS_NO_ERR</code>	no error
<code>VOS_PARAM_ERR</code>	sock descriptor unknown

5.36.2.29 EXT_DECL void vos_sockTerm (void)

De-Initialize the socket library.

Must be called after last socket call

5.37 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 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_SOCKET_OPT_T *pOptions)
Create an UDP socket.
- **EXT_DECL VOS_ERR_T vos_sockOpenTCP** (INT32 *pSock, const VOS_SOCKET_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_SOCKET_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 *pSrcIPAddr, 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 *pPort)
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_sockSetMulticastIff** (INT32 sock, UINT32 mclAddress)
Set Using Multicast I/F.

5.37.1 Detailed Description

Typedefs for OS abstraction. This is the declaration for the OS independent socket interface

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, 2013. All rights reserved.

Id:

vos_sock.h (p. 217) 1226 2014-06-04 15:00:44Z bloehr

5.37.2 Macro Definition Documentation

5.37.2.1 #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

5.37.2.2 #define VOS_TTL_MULTICAST 64

The maximum number of hops a multicast packet can take.

The maximum size for the interface name

5.37.3 Function Documentation

5.37.3.1 EXT_DECL UINT32 vos_dottedIP (const CHAR8 * pDottedIP)

Convert IP address from dotted dec.

to !host! endianness

Parameters

in	<i>pDottedIP</i>	IP address as dotted decimal.
----	------------------	-------------------------------

Return values

<i>address</i>	in UINT32 in host endianness
----------------	------------------------------

to !host! endianness

Parameters

in	<i>pDottedIP</i>	IP address as dotted decimal.
----	------------------	-------------------------------

Return values

<i>address</i>	in UINT32 in host endianness 0 (Zero) if error
----------------	--

Here is the call graph for this function:

5.37.3.2 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	<i>pAddrCnt</i>	in: pointer to array size of interface record out: pointer to number of interface records read
in, out	<i>ifAddrs</i>	array of interface records

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pAddrCnt</i> and/or <i>ifAddrs</i> == NULL
<i>VOS_MEM_ERR</i>	memory allocation error
<i>VOS SOCK_ERR</i>	GetAdaptersInfo() error

Parameters

in, out	<i>pAddrCnt</i>	in: pointer to array size of interface record out: pointer to number of interface records read
in, out	<i>ifAddrs</i>	array of interface records

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pMAC</i> == NULL

Here is the call graph for this function:

5.37.3.3 EXT_DECL UINT32 vos_htonl (UINT32 *val*)

Byte swapping 4 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.37.3.4 EXT_DECL UINT16 vos_htons (UINT16 *val*)

Byte swapping 2 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

Byte swapping 2 Bytes.

Parameters

<i>in</i>	<i>val</i>	Initial value.
-----------	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.37.3.5 EXT_DECL const CHAR8* vos_ipDotted (UINT32 *ipAddress*)

Convert IP address to dotted dec.

from !host! endianness

Parameters

<i>in</i>	<i>ipAddress</i>	address in UINT32 in host endianness
-----------	------------------	--------------------------------------

Return values

<i>IP</i>	address as dotted decimal.
-----------	----------------------------

from !host! endianness.

Parameters

<i>in</i>	<i>ipAddress</i>	address in UINT32 in host endianness
-----------	------------------	--------------------------------------

Return values

<i>IP</i>	address as dotted decimal.
-----------	----------------------------

5.37.3.6 EXT_DECL BOOL8 vos_isMulticast (UINT32 *ipAddress*)

Check if the supplied address is a multicast group address.

Parameters

<i>in</i>	<i>ipAddress</i>	IP address to check.
-----------	------------------	----------------------

Return values

<i>TRUE</i>	address is a multicast address
<i>FALSE</i>	address is not a multicast address

Parameters

in	<i>ipAddress</i>	IP address to check.
----	------------------	----------------------

Return values

<i>TRUE</i>	address is multicast
<i>FALSE</i>	address is not a multicast address

5.37.3.7 EXT_DECL UINT32 vos_ntohl (UINT32 *val*)

Byte swapping 4 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.37.3.8 EXT_DECL UINT16 vos_ntohs (UINT16 *val*)

Byte swapping 2 Bytes.

Parameters

in	<i>val</i>	Initial value.
----	------------	----------------

Return values

<i>swapped</i>	value
----------------	-------

5.37.3.9 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	<i>highDesc</i>	max. socket descriptor + 1
in, out	<i>pReadableFD</i>	pointer to readable socket set
in, out	<i>pWriteableFD</i>	pointer to writeable socket set
in, out	<i>pErrorFD</i>	pointer to error socket set
in	<i>pTimeOut</i>	pointer to time out value

Return values

<i>number</i>	of ready file descriptors
---------------	---------------------------

5.37.3.10 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	<i>sock</i>	Socket descriptor
out	<i>pSock</i>	Pointer to socket descriptor, on exit new socket
out	<i>pIPAddress</i>	source IP to receive on, 0 for any
out	<i>pPort</i>	port to receive on, 20548 for PD

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	NULL parameter, parameter error
<i>VOS_UNKNOWN_ERR</i>	sock descriptor unknown error

Here is the call graph for this function:

5.37.3.11 EXT_DECL VOS_ERR_T vos_sockBind (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Bind a socket to an address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>ipAddress</i>	source IP to receive from, 0 for any
in	<i>port</i>	port to receive from

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

Parameters

in	<i>sock</i>	socket descriptor
in	<i>ipAddress</i>	source IP to receive on, 0 for any
in	<i>port</i>	port to receive on, 20548 for PD

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

Here is the call graph for this function:

5.37.3.12 EXT_DECL VOS_ERR_T vos_sockClose (INT32 *sock*)

Close a socket.

Release any resources acquired by this socket

Parameters

in	<i>sock</i>	socket descriptor
----	-------------	-------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pSock == NULL

Release any resources aquired by this socket

Parameters

<i>in</i>	<i>sock</i>	socket descriptor
-----------	-------------	-------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown

5.37.3.13 EXT_DECL VOS_ERR_T vos_sockConnect (INT32 *sock*, UINT32 *ipAddress*, UINT16 *port*)

Open a TCP connection.

Parameters

<i>in</i>	<i>sock</i>	socket descriptor
<i>in</i>	<i>ipAddress</i>	destination IP
<i>in</i>	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	Input/Output error

Parameters

<i>in</i>	<i>sock</i>	socket descriptor
<i>in</i>	<i>ipAddress</i>	destination IP
<i>in</i>	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error

Parameters

<i>in</i>	<i>sock</i>	socket descriptor
<i>in</i>	<i>ipAddress</i>	destination IP
<i>in</i>	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

Here is the call graph for this function:

5.37.3.14 EXT_DECL VOS_ERR_T vos_sockGetMAC (UINT8 *pMAC*[VOS_MAC_SIZE])

Return the MAC address of the default adapter.

Parameters

out	<i>pMAC</i>	return MAC address.
-----	-------------	---------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pMAC == NULL
<i>VOS SOCK_ERR</i>	socket not available or option not supported

Here is the call graph for this function:

5.37.3.15 EXT_DECL VOS_ERR_T vos_sockInit (void)

Initialize the socket library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS SOCK_ERR</i>	sockets not supported

5.37.3.16 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	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to join, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS SOCK_ERR</i>	option not supported

Note: Some targeted systems might not support this option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to join, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS SOCK_ERR</i>	option not supported

Here is the call graph for this function:

5.37.3.17 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	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to leave, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SOCK_ERR</i>	option not supported

Note: Some targeted systems might not support this option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>mcAddress</i>	multicast group to join
in	<i>ipAddress</i>	depicts interface on which to leave, default 0 for any

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_SOCK_ERR</i>	option not supported

Here is the call graph for this function:

5.37.3.18 EXT_DECL VOS_ERR_T vos_sockListen (INT32 *sock*, UINT32 *backlog*)

Listen for incoming TCP connections.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>backlog</i>	maximum connection attempts if system is busy

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

Listen for incoming TCP connections.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>backlog</i>	maximum connection attempts if system is busy

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	Input/Output error
<i>VOS_MEM_ERR</i>	resource error

5.37.3.19 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	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	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

Here is the call graph for this function:

5.37.3.20 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	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	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

Return a socket descriptor for further calls. The socket options are optional and can be applied later. Note: Some targeted systems might not support every option.

Parameters

out	<i>pSock</i>	pointer to socket descriptor returned
in	<i>pOptions</i>	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

Here is the call graph for this function:

5.37.3.21 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 occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned.

Parameters

in	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data in non-blocking
<i>VOS_BLOCK_ERR</i>	call would have blocked in blocking mode

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 occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned.

Parameters

in	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

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 occurred. If called in non-blocking mode, and no data is available, VOS_NODATA_ERR will be returned.

Parameters

in	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be read
<i>VOS_NODATA_ERR</i>	no data
<i>VOS_BLOCK_ERR</i>	call would have blocked in blocking mode

5.37.3.22 `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 occurred. 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	<i>sock</i>	socket descriptor
out	<i>pBuffer</i>	pointer to applications data buffer
in, out	<i>pSize</i>	pointer to the received data size
out	<i>pSrcIPAddr</i>	pointer to source IP
out	<i>pSrcIPPort</i>	pointer to source port
out	<i>pDstIPAddr</i>	pointer to dest IP
in	<i>peek</i>	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

Here is the call graph for this function:

5.37.3.23 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	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	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

Send data to the supplied address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	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

Send data to the supplied address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	IN: bytes to send, OUT: bytes sent

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_NOCONN_ERR</i>	no TCP connection
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

5.37.3.24 **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	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	In: size of the data to send, Out: no of bytes sent
in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

Send data to the supplied address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send
in, out	<i>pSize</i>	In: size of the data to send, Out: no of bytes sent
in	<i>ipAddress</i>	destination IP
in	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

Send data to the supplied address and port.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pBuffer</i>	pointer to data to send

<i>in, out</i>	<i>pSize</i>	IN: bytes to send, OUT: bytes sent
<i>in</i>	<i>ipAddress</i>	destination IP
<i>in</i>	<i>port</i>	destination port

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS_IO_ERR</i>	data could not be sent
<i>VOS_BLOCK_ERR</i>	Call would have blocked in blocking mode

Here is the call graph for this function:

5.37.3.25 EXT_DECL VOS_ERR_T vos_sockSetMulticastIf (INT32 *sock*, UINT32 *mclfAddress*)

Set Using Multicast I/F.

Parameters

<i>in</i>	<i>sock</i>	socket descriptor
<i>in</i>	<i>mclfAddress</i>	using Multicast I/F Address

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error

Parameters

<i>in</i>	<i>sock</i>	socket descriptor
<i>in</i>	<i>mclfAddress</i>	using Multicast I/F Address

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown, parameter error
<i>VOS SOCK_ERR</i>	option not supported

Here is the call graph for this function:

5.37.3.26 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

<i>in</i>	<i>sock</i>	socket descriptor
<i>in</i>	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

Note: Some targeted systems might not support every option.

Parameters

in	<i>sock</i>	socket descriptor
in	<i>pOptions</i>	pointer to socket options (optional)

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	sock descriptor unknown

5.37.3.27 EXT_DECL void vos_sockTerm (void)

De-Initialize the socket library.

Must be called after last socket call

5.38 vos_thread.c File Reference

Multitasking functions.

```
#include <stdint.h>
#include <unistd.h>
#include <errno.h>
#include <sys/time.h>
#include <pthread.h>
#include <semaphore.h>
#include <sched.h>
#include "vos_sock.h"
#include "vos_types.h"
#include "vos_thread.h"
#include "vos_mem.h"
#include "vos_utils.h"
#include "vos_private.h"
```

Include dependency graph for posix/vos_thread.c:

Macros

- #define **NSECS_PER_USEC** 1000
Cyclic thread functions.

Functions

- EXT_DECL void **vos_cyclicThread** (UINT32 interval, **VOS_THREAD_FUNC_T** pFunction, void *pArguments)
Cyclic thread 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 **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 void **vos_divTime** (**VOS_TIME_T** *pTime, UINT32 divisor)
Divide the first time value 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 INT32 **vos_cmpTime** (const **VOS_TIME_T** *pTime, const **VOS_TIME_T** *pCmp)
Compare the second to the first time stamp.
- 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 recursive mutex.
- EXT_DECL **VOS_ERR_T vos_mutexLocalCreate** (struct **VOS_MUTEX** *pMutex)
Create a recursive mutex.
- EXT_DECL void **vos_mutexDelete** (**VOS_MUTEX_T** pMutex)
Delete a mutex.
- EXT_DECL void **vos_mutexLocalDelete** (struct **VOS_MUTEX** *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.

5.38.1 Detailed Description

Multitasking functions. OS abstraction of thread-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, 2013. All rights reserved.

Id:

vos_thread.c 1334 2014-09-23 09:27:40Z railroad-mike

5.38.2 Macro Definition Documentation

5.38.2.1 #define NSECS_PER_USEC 1000

Cyclic thread functions.

Wrapper for cyclic threads. The thread function will be called cyclically with interval.

Parameters

in	<i>interval</i>	Interval for cyclic threads in us (incl. runtime)
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>void</i>

5.38.3 Function Documentation

5.38.3.1 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	<i>pTime</i>	Pointer to time value
in	<i>pAdd</i>	Pointer to time value

5.38.3.2 EXT_DECL void vos_clearTime (VOS_TIME_T * *pTime*)

Clear the time stamp.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.38.3.3 EXT_DECL INT32 vos_cmpTime (const VOS_TIME_T * *pTime*, const VOS_TIME_T * *pCmp*)

Compare the second to the first time stamp.

Compare the second from the first time stamp, return diff in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>pCmp</i>	Pointer to time value to compare

Return values

0	<i>pTime</i> == <i>pCmp</i>
-1	<i>pTime</i> < <i>pCmp</i>
1	<i>pTime</i> > <i>pCmp</i>

5.38.3.4 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	<i>interval</i>	Interval for cyclic threads in us (incl. runtime)
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

void

Here is the call graph for this function:

5.38.3.5 EXT_DECL void vos_divTime (VOS_TIME_T * *pTime*, UINT32 *divisor*)

Divide the first time value by the second, return quotient in first.

Divide the first time by the second, return quotient in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>divisor</i>	Divisor

5.38.3.6 EXT_DECL void vos_getTime (VOS_TIME_T * *pTime*)

Return the current time in sec and us.

Parameters

--

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.38.3.7 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

<i>timestamp</i>	"yyyymmdd-hh:mm:ss.ms"
------------------	------------------------

5.38.3.8 EXT_DECL void vos_getUuid (VOS_UUID_T pUuid)

Get a universal unique identifier according to RFC 4122 time based version.

Parameters

out	<i>pUuid</i>	Pointer to a universal unique identifier
-----	--------------	--

Here is the call graph for this function:

5.38.3.9 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	<i>pTime</i>	Pointer to time value
in	<i>mul</i>	Factor

5.38.3.10 EXT_DECL VOS_ERR_T vos_mutexCreate (VOS_MUTEX_T * pMutex)

Create a recursive mutex.

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

Parameters

out	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

Here is the call graph for this function:

5.38.3.11 EXT_DECL void vos_mutexDelete (VOS_MUTEX_T pMutex)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Here is the call graph for this function:

5.38.3.12 EXT_DECL VOS_ERR_T vos_mutexLocalCreate (struct VOS_MUTEX * *pMutex*)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

Parameters

out	<i>pMutex</i>	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

5.38.3.13 EXT_DECL void vos_mutexLocalDelete (struct VOS_MUTEX * *pMutex*)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	Pointer to mutex struct
----	---------------	-------------------------

5.38.3.14 EXT_DECL VOS_ERR_T vos_mutexLock (VOS_MUTEX_T *pMutex*)

Take a mutex.

Wait for the mutex to become available (lock).

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

VOS_NO_ERR	no error
VOS_PARAM_ERR	pMutex == NULL or wrong type
VOS_MUTEX_ERR	no such mutex

5.38.3.15 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	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pMutex == NULL or wrong type
<i>VOS_MUTEX_ERR</i>	mutex not locked

5.38.3.16 EXT_DECL VOS_ERR_T vos_mutexUnlock (VOS_MUTEX_T *pMutex*)

Release a mutex.

Unlock the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

5.38.3.17 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	<i>pSema</i>	Pointer to semaphore handle
in	<i>initialState</i>	The initial state of the semaphore

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SEMA_ERR</i>	no semaphore available

Here is the call graph for this function:

5.38.3.18 EXT_DECL void vos_semaDelete (VOS_SEMA_T *sema*)

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

Here is the call graph for this function:

5.38.3.19 EXT_DECL void vos_semaGive (VOS_SEMA_T *sema*)

Give a semaphore.

Release (increase) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

5.38.3.20 EXT_DECL VOS_ERR_T vos_semaTake (VOS_SEMA_T *sema*, UINT32 *timeout*)

Take a semaphore.

Try to get (decrease) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
in	<i>timeout</i>	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

Here is the call graph for this function:

5.38.3.21 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	<i>pTime</i>	Pointer to time value
in	<i>pSub</i>	Pointer to time value

5.38.3.22 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	<i>pThread</i>	Pointer to returned thread handle
in	<i>pName</i>	Pointer to name of the thread (optional)
in	<i>policy</i>	Scheduling policy (FIFO, Round Robin or other)
in	<i>priority</i>	Scheduling priority (1...255 (highest), default 0)
in	<i>interval</i>	Interval for cyclic threads in us (optional)
in	<i>stackSize</i>	Minimum stacksize, default 0: 16kB
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_THREAD_ERR</i>	thread creation error

5.38.3.23 EXT_DECL VOS_ERR_T vos_threadDelay (UINT32 *delay*)

Delay the execution of the current thread by the given delay in us.

Parameters

<i>in</i>	<i>delay</i>	Delay in us
-----------	--------------	-------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.38.3.24 EXT_DECL VOS_ERR_T vos_threadInit (void)

Initialize the thread library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	threading not supported

5.38.3.25 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

<i>in</i>	<i>thread</i>	Thread handle
-----------	---------------	---------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.38.3.26 EXT_DECL void vos_threadTerm (void)

De-Initialize the thread library.

Must be called after last thread/timer call

5.38.3.27 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

<code>in</code>	<code>thread</code>	Thread handle (or NULL if current thread)
-----------------	---------------------	---

Return values

<code>VOS_NO_ERR</code>	no error
<code>VOS_THREAD_ERR</code>	cancel failed

5.39 vos_thread.c File Reference

Multitasking functions.

```
#include <errno.h>
#include <sys/timeb.h>
#include <time.h>
#include <pthread.h>
#include <semaphore.h>
#include <string.h>
#include "vos_thread.h"
#include "vos_sock.h"
#include "vos_mem.h"
#include "vos_utils.h"
#include "vos_private.h"
```

Include dependency graph for windows/vos_thread.c:

Macros

- `#define NSECS_PER_USEC 1000`
Cyclic thread functions.
- `#define NSECS_PER_USEC 1000`
Cyclic thread functions.

Functions

- void **vos_cyclicThread** (UINT32 interval, **VOS_THREAD_FUNC_T** pFunction, void *pArguments)
Cyclic thread functions.
- EXT_DECL **VOS_ERR_T vos_threadInit** (void)
Initialize the thread library.
- EXT_DECL void **vos_threadTerm** (void)
De-Initialize the thread library.
- pthread_t * **vos_getFreeThreadHandle** (void)
Search a free Handle place in the thread handle list.
- 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 **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 void **vos_divTime** (VOS_TIME_T *pTime, UINT32 divisor)
Divide the first time value 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 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_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 recursive mutex.
- **VOS_ERR_T vos_mutexLocalCreate** (struct VOS_MUTEX *pMutex)
Create a recursive mutex.
- EXT_DECL void **vos_mutexDelete** (VOS_MUTEX_T pMutex)
Delete a mutex.
- void **vos_mutexLocalDelete** (struct VOS_MUTEX *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.

5.39.1 Detailed Description

Multitasking functions. OS abstraction of thread-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, 2013. All rights reserved.

Id:

vos_thread.c 1334 2014-09-23 09:27:40Z railroad-mike

5.39.2 Macro Definition Documentation**5.39.2.1 #define NSECS_PER_USEC 1000**

Cyclic thread functions.

Wrapper for cyclic threads. The thread function will be called cyclically with interval.

Parameters

in	<i>interval</i>	Interval for cyclic threads in us (incl. runtime)
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>void</i>

5.39.2.2 #define NSECS_PER_USEC 1000

Cyclic thread functions.

Wrapper for cyclic threads. The thread function will be called cyclically with interval.

Parameters

in	<i>interval</i>	Interval for cyclic threads in us (incl. runtime)
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>void</i>

5.39.3 Function Documentation**5.39.3.1 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	<i>pTime</i>	Pointer to time value
in	<i>pAdd</i>	Pointer to time value

5.39.3.2 EXT_DECL void vos_clearTime (VOS_TIME_T * *pTime*)

Clear the time stamp.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.39.3.3 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	<i>pTime</i>	Pointer to time value
in	<i>pCmp</i>	Pointer to time value to compare

Return values

0	<i>pTime</i> == <i>pCmp</i>
-1	<i>pTime</i> < <i>pCmp</i>
1	<i>pTime</i> > <i>pCmp</i>

5.39.3.4 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	<i>interval</i>	Interval for cyclic threads in us (incl. runtime)
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>void</i>

Here is the call graph for this function:

5.39.3.5 EXT_DECL void vos_divTime (VOS_TIME_T * *pTime*, UINT32 *divisor*)

Divide the first time value by the second, return quotient in first.

Divide the first time by the second, return quotient in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>divisor</i>	Divisor

5.39.3.6 pthread_t* vos_getFreeThreadHandle (void)

Search a free Handle place in the thread handle list.

Return values

<i>pointer</i>	to a free thread handle or NULL if not available
----------------	--

5.39.3.7 EXT_DECL void vos_getTime (VOS_TIME_T * *pTime*)

Return the current time in sec and us.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.39.3.8 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

<i>timestamp</i>	"yyyymmdd-hh:mm:ss.ms"
------------------	------------------------

5.39.3.9 EXT_DECL void vos_getUuid (VOS_UUID_T *pUuid*)

Get a universal unique identifier according to RFC 4122 time based version.

Parameters

out	<i>pUuid</i>	Pointer to a universal unique identifier
-----	--------------	--

Here is the call graph for this function:

5.39.3.10 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	<i>pTime</i>	Pointer to time value
in	<i>mul</i>	Factor

5.39.3.11 EXT_DECL VOS_ERR_T vos_mutexCreate (VOS_MUTEX_T * *pMutex*)

Create a recursive mutex.

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

Parameters

out	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

Here is the call graph for this function:

5.39.3.12 EXT_DECL void vos_mutexDelete (VOS_MUTEX_T pMutex)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Here is the call graph for this function:

5.39.3.13 VOS_ERR_T vos_mutexLocalCreate (struct VOS_MUTEX * pMutex)

Create a recursive mutex.

Fill in a mutex handle. The mutex storage must be already allocated.

Parameters

out	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

5.39.3.14 void vos_mutexLocalDelete (struct VOS_MUTEX * pMutex)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	Pointer to mutex struct
----	---------------	-------------------------

5.39.3.15 EXT_DECL VOS_ERR_T vos_mutexLock (VOS_MUTEX_T pMutex)

Take a mutex.

Wait for the mutex to become available (lock).

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pMutex == NULL or wrong type
<i>VOS_MUTEX_ERR</i>	no such mutex

5.39.3.16 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	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	pMutex == NULL or wrong type
<i>VOS_MUTEX_ERR</i>	mutex not locked

5.39.3.17 EXT_DECL VOS_ERR_T vos_mutexUnlock (VOS_MUTEX_T pMutex)

Release a mutex.

Unlock the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

5.39.3.18 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	<i>pSema</i>	Pointer to semaphore handle
in	<i>initialState</i>	The initial state of the semaphore

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SEMA_ERR</i>	no semaphore available

Here is the call graph for this function:

5.39.3.19 EXT_DECL void vos_semaDelete (VOS_SEMA_T sema)

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

Here is the call graph for this function:

5.39.3.20 EXT_DECL void vos_semaGive (VOS_SEMA_T *sema*)

Give a semaphore.

Release (increase) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

5.39.3.21 EXT_DECL VOS_ERR_T vos_semaTake (VOS_SEMA_T *sema*, UINT32 *timeout*)

Take a semaphore.

Try to get (decrease) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
in	<i>timeout</i>	Max. time in us to wait, 0 means no wait

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SEMA_ERR</i>	could not get semaphore in time

Here is the call graph for this function:

5.39.3.22 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	<i>pTime</i>	Pointer to time value
in	<i>pSub</i>	Pointer to time value

5.39.3.23 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	<i>pThread</i>	Pointer to returned thread handle
in	<i>pName</i>	Pointer to name of the thread (optional)
in	<i>policy</i>	Scheduling policy (FIFO, Round Robin or other)
in	<i>priority</i>	Scheduling priority (1...255 (highest), default 0)
in	<i>interval</i>	Interval for cyclic threads in us (optional)
in	<i>stackSize</i>	Minimum stacksize, default 0: 16kB
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_THREAD_ERR</i>	thread creation error
<i>VOS_INIT_ERR</i>	no threads available

Here is the call graph for this function:

5.39.3.24 EXT_DECL VOS_ERR_T vos_threadDelay (UINT32 delay)

Delay the execution of the current thread by the given delay in us.

Parameters

in	<i>delay</i>	Delay in us
----	--------------	-------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.39.3.25 EXT_DECL VOS_ERR_T vos_threadInit (void)

Initialize the thread library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	threading not supported

5.39.3.26 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	<i>thread</i>	Thread handle
----	---------------	---------------

Return values

<code>VOS_NO_ERR</code>	no error
<code>VOS_PARAM_ERR</code>	parameter out of range/invalid

5.39.3.27 EXT_DECL void vos_threadTerm (void)

De-Initialize the thread library.

Must be called after last thread/timer call

5.39.3.28 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

<code>in</code>	<code>thread</code>	Thread handle (or NULL if current thread)
-----------------	---------------------	---

Return values

<code>VOS_NO_ERR</code>	no error
<code>VOS_THREAD_ERR</code>	cancel failed

5.40 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_t 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.

5.40.1 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:

vos_thread.h (p. 253) 1334 2014-09-23 09:27:40Z railroad-mike

5.40.2 Function Documentation

5.40.2.1 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	<i>pTime</i>	Pointer to time value
in	<i>pAdd</i>	Pointer to time value

5.40.2.2 EXT_DECL void vos_clearTime (VOS_TIME_T * pTime)

Clear the time stamp.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.40.2.3 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	<i>pTime</i>	Pointer to time value
in	<i>pCmp</i>	Pointer to time value to compare

Return values

0	<i>pTime</i> == <i>pCmp</i>
-1	<i>pTime</i> < <i>pCmp</i>
1	<i>pTime</i> > <i>pCmp</i>

Compare the second from the first time stamp, return diff in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>pCmp</i>	Pointer to time value to compare

Return values

0	<i>pTime</i> == <i>pCmp</i>
-1	<i>pTime</i> < <i>pCmp</i>
1	<i>pTime</i> > <i>pCmp</i>

5.40.2.4 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	<i>interval</i>	Interval for cyclic threads in us (incl. runtime)
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

void

Here is the call graph for this function:

5.40.2.5 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	<i>pTime</i>	Pointer to time value
---------	--------------	-----------------------

in	<i>divisor</i>	Divisor
----	----------------	---------

Divide the first time by the second, return quotient in first.

Parameters

in, out	<i>pTime</i>	Pointer to time value
in	<i>divisor</i>	Divisor

5.40.2.6 EXT_DECL void vos_getTime (VOS_TIME_T * *pTime*)

Return the current time in sec and us.

Parameters

out	<i>pTime</i>	Pointer to time value
-----	--------------	-----------------------

5.40.2.7 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

<i>timestamp</i>	"yyyymmdd-hh:mm:ss.ms"
------------------	------------------------

5.40.2.8 EXT_DECL void vos_getUuid (VOS_UUID_T *pUuid*)

Get a universal unique identifier according to RFC 4122 time based version.

Parameters

out	<i>pUuid</i>	Pointer to a universal unique identifier
-----	--------------	--

Here is the call graph for this function:

5.40.2.9 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	<i>pTime</i>	Pointer to time value
in	<i>mul</i>	Factor

5.40.2.10 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	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

Create a mutex.

Return a mutex handle. The mutex will be available at creation.

Parameters

out	<i>pMutex</i>	Pointer to mutex handle
-----	---------------	-------------------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	pMutex == NULL
<i>VOS_MUTEX_ERR</i>	no mutex available

Here is the call graph for this function:

5.40.2.11 EXT_DECL void vos_mutexDelete (VOS_MUTEX_T *pMutex*)

Delete a mutex.

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
-------------------	----------

Release the resources taken by the mutex.

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Here is the call graph for this function:

5.40.2.12 EXT_DECL VOS_ERR_T vos_mutexLock (VOS_MUTEX_T *pMutex*)

Take a mutex.

Wait for the mutex to become available (lock).

Parameters

in	<i>pMutex</i>	mutex handle
----	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
-------------------	----------

<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle

Wait for the mutex to become available (lock).

Parameters

<i>in</i>	<i>pMutex</i>	mutex handle
-----------	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pMutex</i> == NULL or wrong type
<i>VOS_MUTEX_ERR</i>	no such mutex

5.40.2.13 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

<i>in</i>	<i>pMutex</i>	mutex handle
-----------	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_MUTEX_ERR</i>	no mutex available

If mutex is can't be taken VOS_MUTEX_ERR is returned.

Parameters

<i>in</i>	<i>pMutex</i>	mutex handle
-----------	---------------	--------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	<i>pMutex</i> == NULL or wrong type
<i>VOS_MUTEX_ERR</i>	mutex not locked

5.40.2.14 EXT_DECL VOS_ERR_T vos_mutexUnlock (VOS_MUTEX_T *pMutex*)

Release a mutex.

Unlock the mutex.

Parameters

<i>in</i>	<i>pMutex</i>	mutex handle
-----------	---------------	--------------

5.40.2.15 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	<i>pSema</i>	Pointer to semaphore handle
in	<i>initialState</i>	The initial state of the semaphore

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SEMA_ERR</i>	no semaphore available

Here is the call graph for this function:

5.40.2.16 EXT_DECL void vos_semaDelete (VOS_SEMA_T *sema*)

Delete a semaphore.

This will eventually release any processes waiting for the semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

Here is the call graph for this function:

5.40.2.17 EXT_DECL void vos_semaGive (VOS_SEMA_T *sema*)

Give a semaphore.

Release (increase) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
----	-------------	------------------

5.40.2.18 EXT_DECL VOS_ERR_T vos_semaTake (VOS_SEMA_T *sema*, UINT32 *timeout*)

Take a semaphore.

Try to get (decrease) a semaphore.

Parameters

in	<i>sema</i>	semaphore handle
in	<i>timeout</i>	Max. time in us to wait, 0 means no wait

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_SEMA_ERR</i>	could not get semaphore in time

Here is the call graph for this function:

5.40.2.19 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	<i>pTime</i>	Pointer to time value
in	<i>pSub</i>	Pointer to time value

5.40.2.20 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	<i>pThread</i>	Pointer to returned thread handle
in	<i>pName</i>	Pointer to name of the thread (optional)
in	<i>policy</i>	Scheduling policy (FIFO, Round Robin or other)
in	<i>priority</i>	Scheduling priority (1...255 (highest), default 0)
in	<i>interval</i>	Interval for cyclic threads in us (optional)
in	<i>stackSize</i>	Minimum stacksize, default 0: 16kB
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	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

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

Parameters

out	<i>pThread</i>	Pointer to returned thread handle
in	<i>pName</i>	Pointer to name of the thread (optional)
in	<i>policy</i>	Scheduling policy (FIFO, Round Robin or other)
in	<i>priority</i>	Scheduling priority (1...255 (highest), default 0)
in	<i>interval</i>	Interval for cyclic threads in us (optional)
in	<i>stackSize</i>	Minimum stacksize, default 0: 16kB
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	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
VOS_THREAD_ERR	thread creation error

Create a thread and return a thread handle for further requests. Not each parameter may be supported by all target systems!

Parameters

out	<i>pThread</i>	Pointer to returned thread handle
in	<i>pName</i>	Pointer to name of the thread (optional)
in	<i>policy</i>	Scheduling policy (FIFO, Round Robin or other)
in	<i>priority</i>	Scheduling priority (1...255 (highest), default 0)
in	<i>interval</i>	Interval for cyclic threads in us (optional)
in	<i>stackSize</i>	Minimum stacksize, default 0: 16kB
in	<i>pFunction</i>	Pointer to the thread function
in	<i>pArguments</i>	Pointer to the thread function parameters

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised
<i>VOS_NOINIT_ERR</i>	invalid handle
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid
<i>VOS_THREAD_ERR</i>	thread creation error
<i>VOS_INIT_ERR</i>	no threads available

Here is the call graph for this function:

5.40.2.21 EXT_DECL VOS_ERR_T vos_threadDelay (UINT32 delay)

Delay the execution of the current thread by the given delay in us.

Parameters

in	<i>delay</i>	Delay in us
----	--------------	-------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	module not initialised

Parameters

in	<i>delay</i>	Delay in us
----	--------------	-------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.40.2.22 EXT_DECL VOS_ERR_T vos_threadInit (void)

Initialize the thread library.

Must be called once before any other call

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INIT_ERR</i>	threading not supported

5.40.2.23 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

<i>in</i>	<i>thread</i>	Thread handle
-----------	---------------	---------------

Return values

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

Parameters

<i>in</i>	<i>thread</i>	Thread handle
-----------	---------------	---------------

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_PARAM_ERR</i>	parameter out of range/invalid

5.40.2.24 EXT_DECL void vos_threadTerm (void)

De-Initialize the thread library.

Must be called after last thread/timer call

5.40.2.25 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

<i>in</i>	<i>thread</i>	Thread handle (or NULL if current thread)
-----------	---------------	---

Return values

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

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

<i>in</i>	<i>thread</i>	Thread handle (or NULL if current thread)
-----------	---------------	---

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_THREAD_ERR</i>	cancel failed

5.41 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_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_SOCKET_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.

5.41.1 Detailed Description

Typedefs for OS abstraction.

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, 2013. All rights reserved.

Id:

vos_types.h (p. 263) 1262 2014-07-14 13:03:58Z bloehr

5.41.2 Typedef Documentation

5.41.2.1 `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	<i>*pRefCon</i>	pointer to user context
in	<i>category</i>	Log category (Error, Warning, Info etc.)
in	<i>pTime</i>	pointer to NULL-terminated string of time stamp
in	<i>pFile</i>	pointer to NULL-terminated string of source module
in	<i>LineNumber</i>	Line number
in	<i>pMsgStr</i>	pointer to NULL-terminated string

Return values

<i>none</i>

5.41.3 Enumeration Type Documentation

5.41.3.1 `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 Timeout.

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 endianness for selected target wrong.

VOS_NOCONN_ERR No TCP connection.

VOS_UNKNOWN_ERR Unknown error.

5.41.3.2 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.

5.42 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 endianness.
- **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.
- **INLINE BOOL8 vos_isBigEndian** (void)
Return endianness.

5.42.1 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

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:

vos_utils.c (p. 266) 1353 2014-11-11 15:11:13Z ahweiss

BL 2014-02-28: Ticket #25: CRC32 calculation is not according IEEE802.3

5.42.2 Function Documentation

5.42.2.1 `UINT32 vos_crc32 (UINT32 crc, const UINT8 * pData, UINT32 dataLen)`

Compute crc32 according to IEEE802.3.

Calculate CRC for the given buffer and length.

Note: Returned CRC is inverted

Parameters

<code>in</code>	<code>crc</code>	Initial value.
<code>in, out</code>	<code>pData</code>	Pointer to data.
<code>in</code>	<code>dataLen</code>	length in bytes of data.

Return values

<code>crc32</code>	according to IEEE802.3
--------------------	------------------------

5.42.2.2 `VOS_ERR_T vos_init (void * pRefCon, VOS_PRINT_DBG_T pDebugOutput)`

Initialize the virtual operating system.

Initialize the vos library.

Parameters

<code>in</code>	<code>pRefCon</code>	context for debug output function
<code>in</code>	<code>pDebugOutput</code>	Pointer to debug output function.

Return values

<code>VOS_NO_ERR</code>	no error <code>VOS_INTEGRATION_ERR</code> if endianness/alignment mismatch <code>VOS_SOCKET_ERR</code> sockets not supported <code>VOS_UNKNOWN_ERR</code> initialisation error
-------------------------	--

Here is the call graph for this function:

5.42.2.3 `VOS_ERR_T vos_initRuntimeConsts (void)`

Pre-compute alignment and endianness.

Return values

<code>VOS_INTEGRATION_ERR</code>	or <code>VOS_NO_ERR</code>
----------------------------------	----------------------------

5.42.2.4 `INLINE BOOL8 vos_isBigEndian (void)`

Return endianness.

Return values

<code>TRUE</code>	if big endian
-------------------	---------------

5.42.2.5 `EXT_DECL void vos_terminate ()`

Deinitialize the vos library.

Should be called last after TRDP stack/application does not use any VOS function anymore.

Here is the call graph for this function:

5.43 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

Functions

- EXT_DECL UINT32 **vos_crc32** (UINT32 crc, const UINT8 *pData, UINT32 dataLen)
Calculate CRC for the given buffer and length.
- 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.

5.43.1 Detailed Description

Typedefs for OS abstraction.

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, 2013. All rights reserved.

Id:

vos_utils.h (p. 268) 1181 2014-02-28 15:55:27Z bloehr

BL 2014-02-28: Ticket #25: CRC32 calculation is not according IEEE802.3

5.43.2 Macro Definition Documentation

5.43.2.1 #define INITFCS 0xffffffff

CRC/FCS constants.

Initial FCS value

5.43.2.2 #define VOS_MAX_ERR_STR_SIZE (VOS_MAX_PRNT_STR_SIZE - VOS_MAX_FRMT_SIZE)

Max.

size of the error part

5.43.2.3 #define VOS_MAX_FRMT_SIZE 64

Max.

size of the 'format' part

5.43.2.4 #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

5.43.3 Function Documentation

5.43.3.1 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	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEEE802.3
--------------	------------------------

Calculate CRC for the given buffer and length.

Note: Returned CRC is inverted

Parameters

in	<i>crc</i>	Initial value.
in, out	<i>pData</i>	Pointer to data.
in	<i>dataLen</i>	length in bytes of data.

Return values

<i>crc32</i>	according to IEEE802.3
--------------	------------------------

5.43.3.2 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	* <i>pRefCon</i>	user context
in	* <i>pDebugOutput</i>	pointer to debug output function

Return values

<i>VOS_NO_ERR</i>	no error
-------------------	----------

<i>VOS_INIT_ERR</i>	unsupported
---------------------	-------------

Initialize the vos library.

Parameters

in	<i>pRefCon</i>	context for debug output function
in	<i>pDebugOutput</i>	Pointer to debug output function.

Return values

<i>VOS_NO_ERR</i>	no error
<i>VOS_INTEGRATION_ERR</i>	if endianness/alignment mismatch
<i>VOS_SOCKET_ERR</i>	sockets not supported
<i>VOS_UNKNOWN_ERR</i>	initialisation error

Here is the call graph for this function:

5.43.3.3 EXT_DECL void vos_terminate ()

Deinitialize the vos library.

Should be called last after TRDP stack/application does not use any VOS function anymore.

Here is the call graph for this function:

Index

cnCnt
 TRDP_ETB_INFO_T, 25
cnId
 TRDP_FUNCTION_INFO_T, 26
confVehCnt
 GNU_PACKED, 14
confVehList
 GNU_PACKED, 14
cstId
 TRDP_CONSIST_INFO_T, 23
cstList
 GNU_PACKED, 14
cstOwner
 TRDP_CONSIST_INFO_T, 23
cstUUID
 GNU_PACKED, 15
cstVehNo
 TRDP_FUNCTION_INFO_T, 26

datasetLength
 GNU_PACKED, 15
destAddr
 TRDP_PUB_STATISTICS_T, 36
deviceName
 GNU_PACKED, 15

etbId
 TRDP_FUNCTION_INFO_T, 26
etbTopoCnt
 GNU_PACKED, 15

fctId
 TRDP_FUNCTION_INFO_T, 26
filterAddr
 TRDP_SUBS_STATISTICS_T, 42

GNU_PACKED, 9
 confVehCnt, 14
 confVehList, 14
 cstList, 14
 cstUUID, 15
 datasetLength, 15
 deviceName, 15
 etbTopoCnt, 15
 inhibit, 15
 isLead, 15
 leadDir, 15
 lifesign, 15
 msgType, 15
 opCstList, 16
 opTrnDirState, 16
 opTrnTopoCnt, 16
 opVehList, 16
 ownOpCstNo, 16
 protocolVersion, 16
 reserved01, 16
 reserved02, 17
 reserved03, 17
 reserved04, 17
 reserved06, 17
 safetyTrail, 17
 trnCstNo, 17
 trnDirState, 17
 trnId, 18
 trnOperator, 18
 trnTopoCnt, 18
 trnVehNo, 18
 vehId, 18
 vehOrient, 18
 version, 18

INITFCS
 vos_utils.h, 269
inhibit
 GNU_PACKED, 15
isLead
 GNU_PACKED, 15

leadDir
 GNU_PACKED, 15
lifesign
 GNU_PACKED, 15

msgType
 GNU_PACKED, 15

NSECS_PER_USEC
 posix/vos_thread.c, 236
 windows/vos_thread.c, 245
numRecv
 TRDP_SUBS_STATISTICS_T, 42

opCstList
 GNU_PACKED, 16
opTrnDirState
 GNU_PACKED, 16
opTrnTopoCnt
 GNU_PACKED, 16
opVehList
 GNU_PACKED, 16
ownOpCstNo

- GNU_PACKED, 16
- PD_ELE, 19
 - pFrame, 20
- pFrame
 - PD_ELE, 20
- posix/vos_private.h
 - vos_mutexLocalCreate, 187
 - vos_mutexLocalDelete, 187
- posix/vos_shared_mem.c
 - vos_sharedClose, 189
 - vos_sharedOpen, 190
- posix/vos_sock.c
 - vos_dottedIP, 196
 - vos_getInterfaces, 196
 - vos_getMacAddress, 196
 - vos_htonl, 197
 - vos_htons, 197
 - vos_ipDotted, 197
 - vos_isMulticast, 197
 - vos_ntohl, 198
 - vos_ntohs, 198
 - vos_select, 198
 - vos_sockAccept, 198
 - vos_sockBind, 199
 - vos_sockClose, 199
 - vos_sockConnect, 199
 - vos_sockGetMAC, 200
 - vos_sockInit, 200
 - vos_sockJoinMC, 200
 - vos_sockLeaveMC, 200
 - vos_sockListen, 201
 - vos_sockOpenTCP, 201
 - vos_sockOpenUDP, 201
 - vos_sockReceiveTCP, 202
 - vos_sockReceiveUDP, 202
 - vos_sockSendTCP, 203
 - vos_sockSendUDP, 203
 - vos_sockSetBuffer, 203
 - vos_sockSetMulticastIff, 204
 - vos_sockSetOptions, 204
 - vos_sockTerm, 204
- posix/vos_thread.c
 - NSECS_PER_USEC, 236
 - vos_addTime, 236
 - vos_clearTime, 236
 - vos_cmpTime, 237
 - vos_cyclicThread, 237
 - vos_divTime, 237
 - vos_getTime, 237
 - vos_getTimeStamp, 238
 - vos_getUuid, 238
 - vos_mulTime, 238
 - vos_mutexCreate, 238
 - vos_mutexDelete, 238
 - vos_mutexLocalCreate, 239
 - vos_mutexLocalDelete, 239
 - vos_mutexLock, 239
 - vos_mutexTryLock, 239
 - vos_mutexUnlock, 240
 - vos_semaCreate, 240
 - vos_semaDelete, 240
 - vos_semaGive, 240
 - vos_semaTake, 241
 - vos_subTime, 241
 - vos_threadCreate, 241
 - vos_threadDelay, 242
 - vos_threadInit, 242
 - vos_threadIsActive, 242
 - vos_threadTerm, 242
 - vos_threadTerminate, 242
- printSocketUsage
 - trdp_utils.c, 161
- protocolVersion
 - GNU_PACKED, 16
- recvmsg
 - windows/vos_sock.c, 207
- reserved01
 - GNU_PACKED, 16
- reserved02
 - GNU_PACKED, 17
- reserved03
 - GNU_PACKED, 17
- reserved04
 - GNU_PACKED, 17
- reserved06
 - GNU_PACKED, 17
- safetyTrail
 - GNU_PACKED, 17
- TAU_MARSHALL_INFO_T, 20
- TRDP_APP_CONFIRMTO_ERR
 - trdp_types.h, 158
- TRDP_APP_REPLYTO_ERR
 - trdp_types.h, 158
- TRDP_APP_TIMEOUT_ERR
 - trdp_types.h, 158
- TRDP_BITSET8
 - trdp_types.h, 157
- TRDP_BLOCK_ERR
 - trdp_types.h, 158
- TRDP_CHAR8
 - trdp_types.h, 157
- TRDP_CLTR_CST_INFO_T, 21
- TRDP_COMID_DSID_MAP_T, 21
- TRDP_COMID_ERR
 - trdp_types.h, 158
- TRDP_CONFIRMTO_ERR
 - trdp_types.h, 158
- TRDP_CONSIST_INFO_T, 22
 - cstId, 23
 - cstOwner, 23
- TRDP_CRC_ERR
 - trdp_types.h, 158
- TRDP_DATA_TYPE_T
 - trdp_types.h, 157

TRDP_DATASET, 23
TRDP_DATASET_ELEMENT_T, 23
TRDP_DBG_CAT
 tau_xml.h, 84
TRDP_DBG_CONFIG_T, 24
TRDP_DBG_DBG
 tau_xml.h, 84
TRDP_DBG_DEFAULT
 tau_xml.h, 84
TRDP_DBG_ERR
 tau_xml.h, 84
TRDP_DBG_INFO
 tau_xml.h, 84
TRDP_DBG_LOC
 tau_xml.h, 84
TRDP_DBG_OFF
 tau_xml.h, 84
TRDP_DBG_OPTION_T
 tau_xml.h, 84
TRDP_DBG_TIME
 tau_xml.h, 84
TRDP_DBG_WARN
 tau_xml.h, 84
TRDP_DEST_URI_SIZE
 trdp_proto.h, 144
TRDP_ERR_T
 trdp_types.h, 157
TRDP_ETB_INFO_T, 24
 cnCnt, 25
TRDP_ETBCTRL_COMID
 trdp_proto.h, 144
TRDP_ETBCTRL_DSID
 trdp_proto.h, 144
TRDP_FLAGS_CALLBACK
 trdp_types.h, 159
TRDP_FLAGS_DEFAULT
 trdp_types.h, 159
TRDP_FLAGS_MARSHALL
 trdp_types.h, 159
TRDP_FLAGS_NONE
 trdp_types.h, 159
TRDP_FLAGS_T
 trdp_types.h, 158
TRDP_FLAGS_TCP
 trdp_types.h, 159
TRDP_FUNCTION_INFO_T, 25
 cnId, 26
 cstVehNo, 26
 etbId, 26
 fctId, 26
TRDP_HANDLE, 26
TRDP_INIT_ERR
 trdp_types.h, 158
TRDP_INT16
 trdp_types.h, 157
TRDP_INT32
 trdp_types.h, 157
TRDP_INT64
 trdp_types.h, 157
TRDP_INT8
 trdp_types.h, 157
TRDP_INTEGRATION_ERR
 trdp_types.h, 158
TRDP_INVALID_DATA
 trdp_private.h, 142
TRDP_IO_ERR
 trdp_types.h, 158
TRDP_IP_ADDR_T
 trdp_types.h, 155
TRDP_LIST_STATISTICS_T, 27
TRDP_MARSHALL_CONFIG_T, 27
TRDP_MARSHALL_T
 trdp_types.h, 155
TRDP_MAX_FILE_NAME_LEN
 trdp_proto.h, 144
TRDP_MAX_LABEL_LEN
 trdp_proto.h, 144
TRDP_MAX_URI_HOST_LEN
 trdp_proto.h, 144
TRDP_MAX_URI_LEN
 trdp_proto.h, 144
TRDP_MAX_URI_USER_LEN
 trdp_proto.h, 145
TRDP_MD_CALLBACK_T
 trdp_types.h, 156
TRDP_MD_CONFIG_T, 28
TRDP_MD_ELE_ST_T
 trdp_private.h, 141
TRDP_MD_INFO_T, 28
TRDP_MD_STATISTICS_T, 30
TRDP_MEM_CONFIG_T, 31
TRDP_MEM_ERR
 trdp_types.h, 158
TRDP_MEM_STATISTICS_T, 31
TRDP_MSG_MC
 trdp_proto.h, 145
TRDP_MSG_ME
 trdp_proto.h, 145
TRDP_MSG_MN
 trdp_proto.h, 145
TRDP_MSG_MP
 trdp_proto.h, 145
TRDP_MSG_MQ
 trdp_proto.h, 145
TRDP_MSG_MR
 trdp_proto.h, 145
TRDP_MSG_PD
 trdp_proto.h, 145
TRDP_MSG_PE
 trdp_proto.h, 145
TRDP_MSG_PP
 trdp_proto.h, 145
TRDP_MSG_PR
 trdp_proto.h, 145
TRDP_MSG_T
 trdp_proto.h, 145

TRDP_MUTEX_ERR
 trdp_types.h, 158
 TRDP_NO_ERR
 trdp_types.h, 158
 TRDP_NOCONN_ERR
 trdp_types.h, 158
 TRDP_NODATA_ERR
 trdp_types.h, 158
 TRDP_NOINIT_ERR
 trdp_types.h, 158
 TRDP_NOLIST_ERR
 trdp_types.h, 158
 TRDP_NOPUB_ERR
 trdp_types.h, 158
 TRDP_NOSESSION_ERR
 trdp_types.h, 158
 TRDP_NOSUB_ERR
 trdp_types.h, 158
 TRDP_OPTION_BLOCK
 trdp_types.h, 159
 TRDP_OPTION_NO_MC_LOOP_BACK
 trdp_types.h, 159
 TRDP_OPTION_NO_REUSE_ADDR
 trdp_types.h, 159
 TRDP_OPTION_NO_UDP_CHK
 trdp_types.h, 159
 TRDP_OPTION_T
 trdp_types.h, 159
 TRDP_OPTION_TRAFFIC_SHAPING
 trdp_types.h, 159
 TRDP_PACKET_ERR
 trdp_types.h, 158
 TRDP_PARAM_ERR
 trdp_types.h, 158
 TRDP_PD_CALLBACK_T
 trdp_types.h, 156
 TRDP_PD_CONFIG_T, 32
 TRDP_PD_INFO_T, 33
 TRDP_PD_STATISTICS_T, 33
 TRDP_PRINT_DBG_T
 trdp_types.h, 156
 TRDP_PRIV_FLAGS_T
 trdp_private.h, 142
 TRDP_PROCESS_CONFIG_T, 34
 TRDP_PROP_T, 35
 TRDP_PUB_STATISTICS_T, 35
 destAddr, 36
 TRDP_PULL_SUB
 trdp_private.h, 142
 TRDP_QUEUE_ERR
 trdp_types.h, 158
 TRDP_QUEUE_FULL_ERR
 trdp_types.h, 158
 TRDP_REAL32
 trdp_types.h, 157
 TRDP_REAL64
 trdp_types.h, 157
 TRDP_RED_FOLLOWER
 trdp_types.h, 159
 TRDP_RED_LEADER
 trdp_types.h, 159
 TRDP_RED_STATE_T
 trdp_types.h, 159
 TRDP_RED_STATISTICS_T, 36
 TRDP_REDUNDANT
 trdp_private.h, 142
 TRDP_REPLY_STATUS_T
 trdp_types.h, 159
 TRDP_REPLYTO_ERR
 trdp_types.h, 158
 TRDP_REQ_2B_SENT
 trdp_private.h, 142
 TRDP_REQCONFIRMTO_ERR
 trdp_types.h, 158
 TRDP_SDT_DEFAULT_CMTHR
 tau_xml.c, 81
 TRDP_SDT_PAR_T, 36
 TRDP_SEMA_ERR
 trdp_types.h, 158
 TRDP_SEND_PARAM_T, 37
 TRDP_SEQ_CNT_ENTRY_T, 38
 TRDP_SESSION, 38
 TRDP_SESSION_ABORT_ERR
 trdp_types.h, 158
 TRDP SOCK_ERR
 trdp_types.h, 158
 TRDP SOCK_MD_TCP
 trdp_private.h, 142
 TRDP SOCK_MD_UDP
 trdp_private.h, 142
 TRDP SOCK_PD
 trdp_private.h, 142
 TRDP SOCK_TYPE_T
 trdp_private.h, 142
 TRDP SOCKET_TCP, 39
 TRDP_SOCKETS, 40
 usage, 40
 TRDP_ST_NONE
 trdp_private.h, 141
 TRDP_ST_RX_CONF_RECEIVED
 trdp_private.h, 142
 TRDP_ST_RX_NOTIFY_RECEIVED
 trdp_private.h, 142
 TRDP_ST_RX_READY
 trdp_private.h, 141
 TRDP_ST_RX_REPLY_SENT
 trdp_private.h, 142
 TRDP_ST_RX_REPLYQUERY_W4C
 trdp_private.h, 141
 TRDP_ST_RX_REQ_W4AP_REPLY
 trdp_private.h, 141
 TRDP_ST_TX_CONFIRM_ARM
 trdp_private.h, 141
 TRDP_ST_TX_NOTIFY_ARM
 trdp_private.h, 141
 TRDP_ST_TX_REPLY_ARM

- trdp_private.h, 141
- TRDP_ST_TX_REPLY_RECEIVED
 - trdp_private.h, 142
- TRDP_ST_TX_REPLYQUERY_ARM
 - trdp_private.h, 141
- TRDP_ST_TX_REQ_W4AP_CONFIRM
 - trdp_private.h, 142
- TRDP_ST_TX_REQUEST_ARM
 - trdp_private.h, 141
- TRDP_ST_TX_REQUEST_W4REPLY
 - trdp_private.h, 141
- TRDP_STATE_ERR
 - trdp_types.h, 158
- TRDP_STATISTICS_T, 41
- TRDP_SUBS_STATISTICS_T, 42
 - filterAddr, 42
 - numRecv, 42
 - timeout, 42
 - toBehav, 42
- TRDP_THREAD_ERR
 - trdp_types.h, 158
- TRDP_TIME_T
 - trdp_types.h, 156
- TRDP_TIMED_OUT
 - trdp_private.h, 142
- TRDP_TIMEDATE32
 - trdp_types.h, 157
- TRDP_TIMEDATE48
 - trdp_types.h, 157
- TRDP_TIMEDATE64
 - trdp_types.h, 157
- TRDP_TIMEOUT_ERR
 - trdp_types.h, 158
- TRDP_TO_BEHAVIOR_T
 - trdp_types.h, 159
- TRDP_TO_DEFAULT
 - trdp_types.h, 159
- TRDP_TO_KEEP_LAST_VALUE
 - trdp_types.h, 159
- TRDP_TO_SET_TO_ZERO
 - trdp_types.h, 159
- TRDP_TOPO_ERR
 - trdp_types.h, 158
- TRDP_TYPE_MAX
 - trdp_types.h, 157
- TRDP_UINT16
 - trdp_types.h, 157
- TRDP_UINT32
 - trdp_types.h, 157
- TRDP_UINT64
 - trdp_types.h, 157
- TRDP_UINT8
 - trdp_types.h, 157
- TRDP_UNKNOWN_ERR
 - trdp_types.h, 158
- TRDP_UNMARSHALL_T
 - trdp_types.h, 157
- TRDP_UTF16
 - trdp_types.h, 157
- TRDP_VEHICLE_INFO_T, 43
 - vehId, 43
- TRDP_VERSION_T, 44
- TRDP_WIRE_ERR
 - trdp_types.h, 158
- TRDP_XML_DOC_HANDLE_T, 44
- tau_addr2CstId
 - tau_dnr.h, 56
- tau_addr2OpCstNo
 - tau_dnr.h, 56
- tau_addr2OpVehNo
 - tau_dnr.h, 56
- tau_addr2TcnCstNo
 - tau_dnr.h, 57
- tau_addr2TcnVehNo
 - tau_dnr.h, 57
- tau_addr2Uri
 - tau_dnr.h, 57
- tau_addr2VehId
 - tau_dnr.h, 58
- tau_calcDatasetSize
 - tau_marshall.c, 63
 - tau_marshall.h, 67
- tau_calcDatasetSizeByComId
 - tau_marshall.c, 64
 - tau_marshall.h, 68
- tau_ctrl.c, 47
 - tau_getEcspStat, 48
 - tau_initEcspCtrl, 48
 - tau_requestEcspConfirm, 48
 - tau_setEcspCtrl, 49
 - tau_terminateEcspCtrl, 49
- tau_ctrl.h, 49
 - tau_getEcspStat, 50
 - tau_initEcspCtrl, 51
 - tau_requestEcspConfirm, 51
 - tau_setEcspCtrl, 52
 - tau_terminateEcspCtrl, 52
- tau_ctrl_types.h, 52
- tau_dnr.c, 53
- tau_dnr.h, 54
 - tau_addr2CstId, 56
 - tau_addr2OpCstNo, 56
 - tau_addr2OpVehNo, 56
 - tau_addr2TcnCstNo, 57
 - tau_addr2TcnVehNo, 57
 - tau_addr2Uri, 57
 - tau_addr2VehId, 58
 - tau_getOwnAddr, 58
 - tau_getOwnIds, 58
 - tau_iecCstNo2CstId, 58
 - tau_initDnr, 59
 - tau_label2CstId, 59
 - tau_label2OpCstNo, 59
 - tau_label2OpVehNo, 59
 - tau_label2TcnCstNo, 60
 - tau_label2TcnVehNo, 60

- tau_label2VehId, 60
- tau_opVehNo2Ids, 61
- tau_tcnCstNo2CstId, 61
- tau_tcnVehNo2Ids, 61
- tau_uri2Addr, 62
- tau_freeTelegrams
 - tau_xml.c, 81
 - tau_xml.h, 84
- tau_freeXmlDoc
 - tau_xml.c, 81
 - tau_xml.h, 85
- tau_getCarDevCnt
 - tau_tti.h, 73
- tau_getCstCarCnt
 - tau_tti.h, 74
- tau_getCstFctCnt
 - tau_tti.h, 74
- tau_getCstFctInfo
 - tau_tti.h, 74
- tau_getCstInfo
 - tau_tti.h, 75
- tau_getEcspStat
 - tau_ctrl.c, 48
 - tau_ctrl.h, 50
- tau_getlecCarOrient
 - tau_tti.h, 75
- tau_getOpTrDirectory
 - tau_tti.h, 75
- tau_getOwnAddr
 - tau_dnr.h, 58
- tau_getOwnIds
 - tau_dnr.h, 58
- tau_getStaticCstInfo
 - tau_tti.h, 76
- tau_getTTI
 - tau_tti.h, 77
- tau_getTrDirectory
 - tau_tti.h, 76
- tau_getTrnCarCnt
 - tau_tti.h, 76
- tau_getTrnCstCnt
 - tau_tti.h, 76
- tau_getVehInfo
 - tau_tti.h, 77
- tau_getVehOrient
 - tau_tti.h, 77
- tau_iecCstNo2CstId
 - tau_dnr.h, 58
- tau_initDnr
 - tau_dnr.h, 59
- tau_initEcspCtrl
 - tau_ctrl.c, 48
 - tau_ctrl.h, 51
- tau_initMarshall
 - tau_marshall.c, 64
 - tau_marshall.h, 68
- tau_initTtiAccess
 - tau_tti.h, 78
- tau_label2CstId
 - tau_dnr.h, 59
- tau_label2OpCstNo
 - tau_dnr.h, 59
- tau_label2OpVehNo
 - tau_dnr.h, 59
- tau_label2TcnCstNo
 - tau_dnr.h, 60
- tau_label2TcnVehNo
 - tau_dnr.h, 60
- tau_label2VehId
 - tau_dnr.h, 60
- tau_marshall
 - tau_marshall.c, 64
 - tau_marshall.h, 69
- tau_marshall.c, 62
 - tau_calcDatasetSize, 63
 - tau_calcDatasetSizeByComId, 64
 - tau_initMarshall, 64
 - tau_marshall, 64
 - tau_marshallIds, 65
 - tau_unmarshall, 65
 - tau_unmarshallIds, 66
- tau_marshall.h, 66
 - tau_calcDatasetSize, 67
 - tau_calcDatasetSizeByComId, 68
 - tau_initMarshall, 68
 - tau_marshall, 69
 - tau_marshallIds, 69
 - tau_unmarshall, 70
 - tau_unmarshallIds, 70
- tau_marshallIds
 - tau_marshall.c, 65
 - tau_marshall.h, 69
- tau_opVehNo2Ids
 - tau_dnr.h, 61
- tau_prepareXmlDoc
 - tau_xml.c, 81
 - tau_xml.h, 85
- tau_readXmlDatasetConfig
 - tau_xml.c, 81
 - tau_xml.h, 85
- tau_readXmlDeviceConfig
 - tau_xml.c, 82
 - tau_xml.h, 85
- tau_readXmlInterfaceConfig
 - tau_xml.c, 82
 - tau_xml.h, 86
- tau_requestEcspConfirm
 - tau_ctrl.c, 48
 - tau_ctrl.h, 51
- tau_setEcspCtrl
 - tau_ctrl.c, 49
 - tau_ctrl.h, 52
- tau_tcnCstNo2CstId
 - tau_dnr.h, 61
- tau_tcnVehNo2Ids
 - tau_dnr.h, 61

- tau_terminateEcspCtrl
 - tau_ctrl.c, 49
 - tau_ctrl.h, 52
- tau_tti.c, 70
- tau_tti.h, 71
 - tau_getCarDevCnt, 73
 - tau_getCstCarCnt, 74
 - tau_getCstFctCnt, 74
 - tau_getCstFctInfo, 74
 - tau_getCstInfo, 75
 - tau_getlecCarOrient, 75
 - tau_getOpTrDirectory, 75
 - tau_getStaticCstInfo, 76
 - tau_getTTI, 77
 - tau_getTrDirectory, 76
 - tau_getTrnCarCnt, 76
 - tau_getTrnCstCnt, 76
 - tau_getVehInfo, 77
 - tau_getVehOrient, 77
 - tau_initTtiAccess, 78
- tau_tti_types.h, 78
- tau_unmarshall
 - tau_marshall.c, 65
 - tau_marshall.h, 70
- tau_unmarshallDs
 - tau_marshall.c, 66
 - tau_marshall.h, 70
- tau_uri2Addr
 - tau_dnr.h, 62
- tau_xml.c, 79
 - TRDP_SDT_DEFAULT_CMTHR, 81
 - tau_freeTelegrams, 81
 - tau_freeXmlDoc, 81
 - tau_prepareXmlDoc, 81
 - tau_readXmlDatasetConfig, 81
 - tau_readXmlDeviceConfig, 82
 - tau_readXmlInterfaceConfig, 82
- tau_xml.h, 83
 - TRDP_DBG_CAT, 84
 - TRDP_DBG_DBG, 84
 - TRDP_DBG_DEFAULT, 84
 - TRDP_DBG_ERR, 84
 - TRDP_DBG_INFO, 84
 - TRDP_DBG_LOC, 84
 - TRDP_DBG_OFF, 84
 - TRDP_DBG_OPTION_T, 84
 - TRDP_DBG_TIME, 84
 - TRDP_DBG_WARN, 84
 - tau_freeTelegrams, 84
 - tau_freeXmlDoc, 85
 - tau_prepareXmlDoc, 85
 - tau_readXmlDatasetConfig, 85
 - tau_readXmlDeviceConfig, 85
 - tau_readXmlInterfaceConfig, 86
- timeout
 - TRDP_SUBS_STATISTICS_T, 42
- tlc_closeSession
 - trdp_if.c, 89
 - trdp_if_light.h, 102
- tlc_freeBuf
 - trdp_if_light.h, 103
- tlc_getInterval
 - trdp_if.c, 89
 - trdp_if_light.h, 103
- tlc_getJoinStatistics
 - trdp_if_light.h, 104
 - trdp_stats.c, 146
- tlc_getListStatistics
 - trdp_if_light.h, 104
- tlc_getPubStatistics
 - trdp_if_light.h, 105
 - trdp_stats.c, 147
- tlc_getRedStatistics
 - trdp_if_light.h, 105
 - trdp_stats.c, 147
- tlc_getStatistics
 - trdp_if_light.h, 106
 - trdp_stats.c, 147
- tlc_getSubsStatistics
 - trdp_if_light.h, 106
 - trdp_stats.c, 148
- tlc_getVersion
 - trdp_if.c, 90
 - trdp_if_light.h, 107
- tlc_getVersionString
 - trdp_if.c, 90
 - trdp_if_light.h, 107
- tlc_init
 - trdp_if.c, 90
 - trdp_if_light.h, 107
- tlc_openSession
 - trdp_if.c, 90
 - trdp_if_light.h, 108
- tlc_process
 - trdp_if.c, 91
 - trdp_if_light.h, 108
- tlc_reinitSession
 - trdp_if.c, 91
 - trdp_if_light.h, 109
- tlc_resetStatistics
 - trdp_if_light.h, 109
 - trdp_stats.c, 148
- tlc_setETBTopoCount
 - trdp_if.c, 92
 - trdp_if_light.h, 110
- tlc_setOpTrainTopoCount
 - trdp_if.c, 92
 - trdp_if_light.h, 110
- tlc_terminate
 - trdp_if.c, 92
 - trdp_if_light.h, 110
- tlim_abortSession
 - trdp_if_light.h, 111
- tlim_addListener
 - trdp_if_light.h, 111
- tlim_confirm

- trdp_if_light.h, 112
- tlm_delListener
 - trdp_if_light.h, 112
- tlm_notify
 - trdp_if_light.h, 112
- tlm_readdListener
 - trdp_if_light.h, 113
- tlm_reply
 - trdp_if_light.h, 113
- tlm_replyErr
 - trdp_if_light.h, 114
- tlm_replyQuery
 - trdp_if_light.h, 114
- tlm_request
 - trdp_if_light.h, 115
- tlp_get
 - trdp_if.c, 92
 - trdp_if_light.h, 115
- tlp_getRedundant
 - trdp_if.c, 93
 - trdp_if_light.h, 117
- tlp_publish
 - trdp_if.c, 93
 - trdp_if_light.h, 118
- tlp_put
 - trdp_if.c, 94
 - trdp_if_light.h, 119
- tlp_republish
 - trdp_if.c, 94
 - trdp_if_light.h, 120
- tlp_request
 - trdp_if.c, 95
 - trdp_if_light.h, 120
- tlp_resubscribe
 - trdp_if.c, 95
 - trdp_if_light.h, 121
- tlp_setRedundant
 - trdp_if.c, 96
 - trdp_if_light.h, 122
- tlp_subscribe
 - trdp_if.c, 96
 - trdp_if_light.h, 123
- tlp_unpublish
 - trdp_if.c, 97
 - trdp_if_light.h, 124
- tlp_unsubscribe
 - trdp_if.c, 97
 - trdp_if_light.h, 124
- toBehav
 - TRDP_SUBS_STATISTICS_T, 42
- trdp_SockAddJoin
 - trdp_utils.c, 165
- trdp_SockDelJoin
 - trdp_utils.c, 165
- trdp_SockIsJoined
 - trdp_utils.c, 167
- trdp_UpdateStats
 - trdp_stats.c, 150
- trdp_checkSequenceCounter
 - trdp_utils.c, 161
 - trdp_utils.h, 168
- trdp_dllmain.c, 86
- trdp_getSeqCnt
 - trdp_utils.c, 162
 - trdp_utils.h, 169
- trdp_if.c, 87
 - tlc_closeSession, 89
 - tlc_getInterval, 89
 - tlc_getVersion, 90
 - tlc_getVersionString, 90
 - tlc_init, 90
 - tlc_openSession, 90
 - tlc_process, 91
 - tlc_reinitSession, 91
 - tlc_setETBTopoCount, 92
 - tlc_setOpTrainTopoCount, 92
 - tlc_terminate, 92
 - tlp_get, 92
 - tlp_getRedundant, 93
 - tlp_publish, 93
 - tlp_put, 94
 - tlp_republish, 94
 - tlp_request, 95
 - tlp_resubscribe, 95
 - tlp_setRedundant, 96
 - tlp_subscribe, 96
 - tlp_unpublish, 97
 - tlp_unsubscribe, 97
 - trdp_isValidSession, 98
 - trdp_sessionQueue, 98
- trdp_if.h, 98
 - trdp_isValidSession, 99
 - trdp_sessionQueue, 99
- trdp_if_light.h, 99
 - tlc_closeSession, 102
 - tlc_freeBuf, 103
 - tlc_getInterval, 103
 - tlc_getJoinStatistics, 104
 - tlc_getListStatistics, 104
 - tlc_getPubStatistics, 105
 - tlc_getRedStatistics, 105
 - tlc_getStatistics, 106
 - tlc_getSubsStatistics, 106
 - tlc_getVersion, 107
 - tlc_getVersionString, 107
 - tlc_init, 107
 - tlc_openSession, 108
 - tlc_process, 108
 - tlc_reinitSession, 109
 - tlc_resetStatistics, 109
 - tlc_setETBTopoCount, 110
 - tlc_setOpTrainTopoCount, 110
 - tlc_terminate, 110
 - tlm_abortSession, 111
 - tlm_addListener, 111
 - tlm_confirm, 112

- tlm_delListener, 112
- tlm_notify, 112
- tlm_readdListener, 113
- tlm_reply, 113
- tlm_replyErr, 114
- tlm_replyQuery, 114
- tlm_request, 115
- tlp_get, 115
- tlp_getRedundant, 117
- tlp_publish, 118
- tlp_put, 119
- tlp_republish, 120
- tlp_request, 120
- tlp_resubscribe, 121
- tlp_setRedundant, 122
- tlp_subscribe, 123
- tlp_unpublish, 124
- tlp_unsubscribe, 124
- trdp_initSockets
 - trdp_utils.c, 162
 - trdp_utils.h, 169
- trdp_initStats
 - trdp_stats.c, 148
 - trdp_stats.h, 151
- trdp_initUncompletedTCP
 - trdp_utils.h, 169
- trdp_isAddressed
 - trdp_utils.c, 162
 - trdp_utils.h, 169
- trdp_isValidSession
 - trdp_if.c, 98
 - trdp_if.h, 99
- trdp_mdCheckListenSocks
 - trdp_mdcom.c, 126
 - trdp_mdcom.h, 128
- trdp_mdCheckPending
 - trdp_mdcom.c, 126
 - trdp_mdcom.h, 128
- trdp_mdCheckTimeouts
 - trdp_mdcom.c, 126
 - trdp_mdcom.h, 128
- trdp_mdFreeSession
 - trdp_mdcom.c, 126
 - trdp_mdcom.h, 129
- trdp_mdGetTCPSocket
 - trdp_mdcom.c, 127
 - trdp_mdcom.h, 129
- trdp_mdSend
 - trdp_mdcom.c, 127
 - trdp_mdcom.h, 129
- trdp_mdcom.c, 125
 - trdp_mdCheckListenSocks, 126
 - trdp_mdCheckPending, 126
 - trdp_mdCheckTimeouts, 126
 - trdp_mdFreeSession, 126
 - trdp_mdGetTCPSocket, 127
 - trdp_mdSend, 127
- trdp_mdcom.h, 127
 - trdp_mdCheckListenSocks, 128
 - trdp_mdCheckPending, 128
 - trdp_mdCheckTimeouts, 128
 - trdp_mdFreeSession, 129
 - trdp_mdGetTCPSocket, 129
 - trdp_mdSend, 129
- trdp_packetSizeMD
 - trdp_utils.c, 162
 - trdp_utils.h, 170
- trdp_packetSizePD
 - trdp_utils.c, 163
 - trdp_utils.h, 170
- trdp_pdCheck
 - trdp_pdcom.c, 131
 - trdp_pdcom.h, 136
- trdp_pdCheckAppTopoCounts
 - trdp_pdcom.c, 131
 - trdp_pdcom.h, 136
- trdp_pdCheckListenSocks
 - trdp_pdcom.c, 131
 - trdp_pdcom.h, 136
- trdp_pdCheckPending
 - trdp_pdcom.c, 131
 - trdp_pdcom.h, 137
- trdp_pdDistribute
 - trdp_pdcom.c, 133
 - trdp_pdcom.h, 137
- trdp_pdHandleTimeOuts
 - trdp_pdcom.c, 133
 - trdp_pdcom.h, 137
- trdp_pdInit
 - trdp_pdcom.c, 133
 - trdp_pdcom.h, 137
- trdp_pdPrepareStats
 - trdp_stats.c, 150
 - trdp_stats.h, 151
- trdp_pdReceive
 - trdp_pdcom.c, 133
 - trdp_pdcom.h, 138
- trdp_pdSend
 - trdp_pdcom.c, 134
 - trdp_pdcom.h, 138
- trdp_pdSendQueued
 - trdp_pdcom.c, 134
 - trdp_pdcom.h, 138
- trdp_pdUpdate
 - trdp_pdcom.c, 134
 - trdp_pdcom.h, 139
- trdp_pdcom.c, 129
 - trdp_pdCheck, 131
 - trdp_pdCheckAppTopoCounts, 131
 - trdp_pdCheckListenSocks, 131
 - trdp_pdCheckPending, 131
 - trdp_pdDistribute, 133
 - trdp_pdHandleTimeOuts, 133
 - trdp_pdInit, 133
 - trdp_pdReceive, 133
 - trdp_pdSend, 134

- trdp_pdSendQueued, 134
- trdp_pdUpdate, 134
- trdp_pdcom.h, 135
 - trdp_pdCheck, 136
 - trdp_pdCheckAppTopoCounts, 136
 - trdp_pdCheckListenSocks, 136
 - trdp_pdCheckPending, 137
 - trdp_pdDistribute, 137
 - trdp_pdHandleTimeOuts, 137
 - trdp_pdlInit, 137
 - trdp_pdReceive, 138
 - trdp_pdSend, 138
 - trdp_pdSendQueued, 138
 - trdp_pdUpdate, 139
- trdp_private.h, 139
 - TRDP_INVALID_DATA, 142
 - TRDP_MD_ELE_ST_T, 141
 - TRDP_PRIV_FLAGS_T, 142
 - TRDP_PULL_SUB, 142
 - TRDP_REDUNDANT, 142
 - TRDP_REQ_2B_SENT, 142
 - TRDP SOCK_MD_TCP, 142
 - TRDP SOCK_MD_UDP, 142
 - TRDP SOCK_PD, 142
 - TRDP SOCK_TYPE_T, 142
 - TRDP_ST_NONE, 141
 - TRDP_ST_RX_CONF_RECEIVED, 142
 - TRDP_ST_RX_NOTIFY_RECEIVED, 142
 - TRDP_ST_RX_READY, 141
 - TRDP_ST_RX_REPLY_SENT, 142
 - TRDP_ST_RX_REPLYQUERY_W4C, 141
 - TRDP_ST_RX_REQ_W4AP_REPLY, 141
 - TRDP_ST_TX_CONFIRM_ARM, 141
 - TRDP_ST_TX_NOTIFY_ARM, 141
 - TRDP_ST_TX_REPLY_ARM, 141
 - TRDP_ST_TX_REPLY_RECEIVED, 142
 - TRDP_ST_TX_REPLYQUERY_ARM, 141
 - TRDP_ST_TX_REQ_W4AP_CONFIRM, 142
 - TRDP_ST_TX_REQUEST_ARM, 141
 - TRDP_ST_TX_REQUEST_W4REPLY, 141
 - TRDP_TIMED_OUT, 142
- trdp_proto.h, 142
 - TRDP_DEST_URI_SIZE, 144
 - TRDP_ETBCTRL_COMID, 144
 - TRDP_ETBCTRL_DSID, 144
 - TRDP_MAX_FILE_NAME_LEN, 144
 - TRDP_MAX_LABEL_LEN, 144
 - TRDP_MAX_URI_HOST_LEN, 144
 - TRDP_MAX_URI_LEN, 144
 - TRDP_MAX_URI_USER_LEN, 145
 - TRDP_MSG_MC, 145
 - TRDP_MSG_ME, 145
 - TRDP_MSG_MN, 145
 - TRDP_MSG_MP, 145
 - TRDP_MSG_MQ, 145
 - TRDP_MSG_MR, 145
 - TRDP_MSG_PD, 145
 - TRDP_MSG_PE, 145
 - TRDP_MSG_PP, 145
 - TRDP_MSG_PR, 145
 - TRDP_MSG_T, 145
- trdp_queueAppLast
 - trdp_utils.c, 163
 - trdp_utils.h, 170
- trdp_queueDelElement
 - trdp_utils.c, 163
 - trdp_utils.h, 170
- trdp_queueFindComId
 - trdp_utils.c, 163
 - trdp_utils.h, 170
- trdp_queueFindPubAddr
 - trdp_utils.c, 163
 - trdp_utils.h, 171
- trdp_queueFindSubAddr
 - trdp_utils.c, 164
 - trdp_utils.h, 171
- trdp_queueInsFirst
 - trdp_utils.c, 164
 - trdp_utils.h, 171
- trdp_releaseSocket
 - trdp_utils.c, 164
 - trdp_utils.h, 171
- trdp_requestSocket
 - trdp_utils.c, 164
 - trdp_utils.h, 172
- trdp_resetSequenceCounter
 - trdp_utils.c, 165
 - trdp_utils.h, 173
- trdp_sessionQueue
 - trdp_if.c, 98
 - trdp_if.h, 99
- trdp_stats.c, 145
 - tlc_getJoinStatistics, 146
 - tlc_getPubStatistics, 147
 - tlc_getRedStatistics, 147
 - tlc_getStatistics, 147
 - tlc_getSubsStatistics, 148
 - tlc_resetStatistics, 148
 - trdp_UpdateStats, 150
 - trdp_initStats, 148
 - trdp_pdPrepareStats, 150
- trdp_stats.h, 150
 - trdp_initStats, 151
 - trdp_pdPrepareStats, 151
- trdp_types.h, 151
 - TRDP_APP_CONFIRMTO_ERR, 158
 - TRDP_APP_REPLYTO_ERR, 158
 - TRDP_APP_TIMEOUT_ERR, 158
 - TRDP_BITSET8, 157
 - TRDP_BLOCK_ERR, 158
 - TRDP_CHAR8, 157
 - TRDP_COMID_ERR, 158
 - TRDP_CONFIRMTO_ERR, 158
 - TRDP_CRC_ERR, 158
 - TRDP_DATA_TYPE_T, 157
 - TRDP_ERR_T, 157

TRDP_FLAGS_CALLBACK, 159
 TRDP_FLAGS_DEFAULT, 159
 TRDP_FLAGS_MARSHALL, 159
 TRDP_FLAGS_NONE, 159
 TRDP_FLAGS_T, 158
 TRDP_FLAGS_TCP, 159
 TRDP_INIT_ERR, 158
 TRDP_INT16, 157
 TRDP_INT32, 157
 TRDP_INT64, 157
 TRDP_INT8, 157
 TRDP_INTEGRATION_ERR, 158
 TRDP_IO_ERR, 158
 TRDP_IP_ADDR_T, 155
 TRDP_MARSHALL_T, 155
 TRDP_MD_CALLBACK_T, 156
 TRDP_MEM_ERR, 158
 TRDP_MUTEX_ERR, 158
 TRDP_NO_ERR, 158
 TRDP_NOCONN_ERR, 158
 TRDP_NODATA_ERR, 158
 TRDP_NOINIT_ERR, 158
 TRDP_NOLIST_ERR, 158
 TRDP_NOPUB_ERR, 158
 TRDP_NOSESSION_ERR, 158
 TRDP_NOSUB_ERR, 158
 TRDP_OPTION_BLOCK, 159
 TRDP_OPTION_NO_MC_LOOP_BACK, 159
 TRDP_OPTION_NO_REUSE_ADDR, 159
 TRDP_OPTION_NO_UDP_CHK, 159
 TRDP_OPTION_T, 159
 TRDP_OPTION_TRAFFIC_SHAPING, 159
 TRDP_PACKET_ERR, 158
 TRDP_PARAM_ERR, 158
 TRDP_PD_CALLBACK_T, 156
 TRDP_PRINT_DBG_T, 156
 TRDP_QUEUE_ERR, 158
 TRDP_QUEUE_FULL_ERR, 158
 TRDP_REAL32, 157
 TRDP_REAL64, 157
 TRDP_RED_FOLLOWER, 159
 TRDP_RED_LEADER, 159
 TRDP_RED_STATE_T, 159
 TRDP_REPLY_STATUS_T, 159
 TRDP_REPLYTO_ERR, 158
 TRDP_REQCONFIRMTO_ERR, 158
 TRDP_SEMA_ERR, 158
 TRDP_SESSION_ABORT_ERR, 158
 TRDP SOCK_ERR, 158
 TRDP_STATE_ERR, 158
 TRDP_THREAD_ERR, 158
 TRDP_TIME_T, 156
 TRDP_TIMESTAMP32, 157
 TRDP_TIMESTAMP48, 157
 TRDP_TIMESTAMP64, 157
 TRDP_TIMEOUT_ERR, 158
 TRDP_TO_BEHAVIOR_T, 159
 TRDP_TO_DEFAULT, 159

TRDP_TO_KEEP_LAST_VALUE, 159
 TRDP_TO_SET_TO_ZERO, 159
 TRDP_TOPO_ERR, 158
 TRDP_TYPE_MAX, 157
 TRDP_UINT16, 157
 TRDP_UINT32, 157
 TRDP_UINT64, 157
 TRDP_UINT8, 157
 TRDP_UNKNOWN_ERR, 158
 TRDP_UNMARSHALL_T, 157
 TRDP_UTF16, 157
 TRDP_WIRE_ERR, 158
 trdp_utils.c, 160
 printSocketUsage, 161
 trdp_SockAddJoin, 165
 trdp_SockDelJoin, 165
 trdp_SockIsJoined, 167
 trdp_checkSequenceCounter, 161
 trdp_getSeqCnt, 162
 trdp_initSockets, 162
 trdp_isAddressed, 162
 trdp_packetSizeMD, 162
 trdp_packetSizePD, 163
 trdp_queueAppLast, 163
 trdp_queueDelElement, 163
 trdp_queueFindComId, 163
 trdp_queueFindPubAddr, 163
 trdp_queueFindSubAddr, 164
 trdp_queueInsFirst, 164
 trdp_releaseSocket, 164
 trdp_requestSocket, 164
 trdp_resetSequenceCounter, 165
 trdp_utils.h, 167
 trdp_checkSequenceCounter, 168
 trdp_getSeqCnt, 169
 trdp_initSockets, 169
 trdp_initUncompletedTCP, 169
 trdp_isAddressed, 169
 trdp_packetSizeMD, 170
 trdp_packetSizePD, 170
 trdp_queueAppLast, 170
 trdp_queueDelElement, 170
 trdp_queueFindComId, 170
 trdp_queueFindPubAddr, 171
 trdp_queueFindSubAddr, 171
 trdp_queueInsFirst, 171
 trdp_releaseSocket, 171
 trdp_requestSocket, 172
 trdp_resetSequenceCounter, 173
 trnCstNo
 GNU_PACKED, 17
 trnDirState
 GNU_PACKED, 17
 trnId
 GNU_PACKED, 18
 trnOperator
 GNU_PACKED, 18
 trnTopoCnt

- GNU_PACKED, 18
- trnVehNo
 - GNU_PACKED, 18
- tv_usec
 - VOS_TIME_T, 45
- usage
 - TRDP_SOCKETS, 40
- VOS_BLOCK_ERR
 - vos_types.h, 266
- VOS_ERR_T
 - vos_types.h, 265
- VOS_INIT_ERR
 - vos_types.h, 265
- VOS_INTEGRATION_ERR
 - vos_types.h, 266
- VOS_IO_ERR
 - vos_types.h, 266
- VOS_LOG_DBG
 - vos_types.h, 266
- VOS_LOG_ERROR
 - vos_types.h, 266
- VOS_LOG_INFO
 - vos_types.h, 266
- VOS_LOG_T
 - vos_types.h, 266
- VOS_LOG_WARNING
 - vos_types.h, 266
- VOS_MAX_ERR_STR_SIZE
 - vos_utils.h, 269
- VOS_MAX_FRMT_SIZE
 - vos_utils.h, 269
- VOS_MAX_PRNT_STR_SIZE
 - vos_utils.h, 270
- VOS_MAX_SOCKET_CNT
 - vos_sock.h, 219
- VOS_MEM_BLOCKSIZE
 - vos_mem.h, 181
- VOS_MEM_ERR
 - vos_types.h, 266
- VOS_MEM_PREALLOCATE
 - vos_mem.h, 181
- VOS_MUTEX_ERR
 - vos_types.h, 266
- VOS_NO_ERR
 - vos_types.h, 265
- VOS_NOCONN_ERR
 - vos_types.h, 266
- VOS_NODATA_ERR
 - vos_types.h, 265
- VOS_NOINIT_ERR
 - vos_types.h, 265
- VOS_PARAM_ERR
 - vos_types.h, 265
- VOS_PRINT_DBG_T
 - vos_types.h, 265
- VOS_QUEUE_ERR
 - vos_types.h, 266
- VOS_QUEUE_FULL_ERR
 - vos_types.h, 266
- VOS_SEMA_ERR
 - vos_types.h, 266
- VOS SOCK_ERR
 - vos_types.h, 266
- VOS SOCK_OPT_T, 44
- VOS_THREAD_ERR
 - vos_types.h, 266
- VOS_TIME_T, 45
 - tv_usec, 45
- VOS_TIMEOUT_ERR
 - vos_types.h, 265
- VOS_TTL_MULTICAST
 - vos_sock.h, 219
- VOS_UNKNOWN_ERR
 - vos_types.h, 266
- vehId
 - GNU_PACKED, 18
 - TRDP_VEHICLE_INFO_T, 43
- vehOrient
 - GNU_PACKED, 18
- version
 - GNU_PACKED, 18
- vos_addTime
 - posix/vos_thread.c, 236
 - vos_thread.h, 255
 - windows/vos_thread.c, 245
- vos_bsearch
 - vos_mem.c, 175
 - vos_mem.h, 181
- vos_clearTime
 - posix/vos_thread.c, 236
 - vos_thread.h, 255
 - windows/vos_thread.c, 246
- vos_cmpTime
 - posix/vos_thread.c, 237
 - vos_thread.h, 256
 - windows/vos_thread.c, 246
- vos_crc32
 - vos_utils.c, 267
 - vos_utils.h, 270
- vos_cyclicThread
 - posix/vos_thread.c, 237
 - vos_thread.h, 256
 - windows/vos_thread.c, 246
- vos_divTime
 - posix/vos_thread.c, 237
 - vos_thread.h, 256
 - windows/vos_thread.c, 246
- vos_dottedIP
 - posix/vos_sock.c, 196
 - vos_sock.h, 219
 - windows/vos_sock.c, 208
- vos_getFreeThreadHandle
 - windows/vos_thread.c, 246
- vos_getInterfaces
 - posix/vos_sock.c, 196

- vos_sock.h, 220
- windows/vos_sock.c, 208
- vos_getMacAddress
 - posix/vos_sock.c, 196
- vos_getTime
 - posix/vos_thread.c, 237
 - vos_thread.h, 257
 - windows/vos_thread.c, 248
- vos_getTimeStamp
 - posix/vos_thread.c, 238
 - vos_thread.h, 257
 - windows/vos_thread.c, 248
- vos_getUuid
 - posix/vos_thread.c, 238
 - vos_thread.h, 257
 - windows/vos_thread.c, 248
- vos_htonl
 - posix/vos_sock.c, 197
 - vos_sock.h, 220
 - windows/vos_sock.c, 208
- vos_htons
 - posix/vos_sock.c, 197
 - vos_sock.h, 220
 - windows/vos_sock.c, 208
- vos_init
 - vos_utils.c, 267
 - vos_utils.h, 270
- vos_initRuntimeConsts
 - vos_utils.c, 268
- vos_ipDotted
 - posix/vos_sock.c, 197
 - vos_sock.h, 222
 - windows/vos_sock.c, 210
- vos_isBigEndian
 - vos_utils.c, 268
- vos_isMulticast
 - posix/vos_sock.c, 197
 - vos_sock.h, 222
 - windows/vos_sock.c, 210
- vos_mem.c, 173
 - vos_bsearch, 175
 - vos_memAlloc, 175
 - vos_memCount, 175
 - vos_memDelete, 176
 - vos_memFree, 176
 - vos_memInit, 176
 - vos_mutexLocalCreate, 176
 - vos_mutexLocalDelete, 177
 - vos_qsort, 177
 - vos_queueCreate, 177
 - vos_queueDestroy, 178
 - vos_queueReceive, 178
 - vos_queueSend, 178
 - vos_strncpy, 179
 - vos_strncmp, 179
- vos_mem.h, 179
 - VOS_MEM_BLOCKSIZE, 181
 - VOS_MEM_PREALLOCATE, 181
- vos_bsearch, 181
- vos_memAlloc, 182
- vos_memCount, 182
- vos_memDelete, 182
- vos_memFree, 183
- vos_memInit, 183
- vos_qsort, 184
- vos_queueCreate, 184
- vos_queueDestroy, 184
- vos_queueReceive, 185
- vos_queueSend, 185
- vos_strncpy, 185
- vos_strncmp, 186
- vos_memAlloc
 - vos_mem.c, 175
 - vos_mem.h, 182
- vos_memCount
 - vos_mem.c, 175
 - vos_mem.h, 182
- vos_memDelete
 - vos_mem.c, 176
 - vos_mem.h, 182
- vos_memFree
 - vos_mem.c, 176
 - vos_mem.h, 183
- vos_memInit
 - vos_mem.c, 176
 - vos_mem.h, 183
- vos_mulTime
 - posix/vos_thread.c, 238
 - vos_thread.h, 257
 - windows/vos_thread.c, 248
- vos_mutexCreate
 - posix/vos_thread.c, 238
 - vos_thread.h, 257
 - windows/vos_thread.c, 248
- vos_mutexDelete
 - posix/vos_thread.c, 238
 - vos_thread.h, 258
 - windows/vos_thread.c, 249
- vos_mutexLocalCreate
 - posix/vos_private.h, 187
 - posix/vos_thread.c, 239
 - vos_mem.c, 176
 - windows/vos_private.h, 188
 - windows/vos_thread.c, 249
- vos_mutexLocalDelete
 - posix/vos_private.h, 187
 - posix/vos_thread.c, 239
 - vos_mem.c, 177
 - windows/vos_private.h, 188
 - windows/vos_thread.c, 249
- vos_mutexLock
 - posix/vos_thread.c, 239
 - vos_thread.h, 258
 - windows/vos_thread.c, 249
- vos_mutexTryLock
 - posix/vos_thread.c, 239

- vos_thread.h, 259
 - windows/vos_thread.c, 250
- vos_mutexUnlock
 - posix/vos_thread.c, 240
 - vos_thread.h, 259
 - windows/vos_thread.c, 250
- vos_ntohl
 - posix/vos_sock.c, 198
 - vos_sock.h, 222
 - windows/vos_sock.c, 210
- vos_ntohs
 - posix/vos_sock.c, 198
 - vos_sock.h, 224
 - windows/vos_sock.c, 210
- vos_private.h, 186, 187
- vos_qsort
 - vos_mem.c, 177
 - vos_mem.h, 184
- vos_queueCreate
 - vos_mem.c, 177
 - vos_mem.h, 184
- vos_queueDestroy
 - vos_mem.c, 178
 - vos_mem.h, 184
- vos_queueReceive
 - vos_mem.c, 178
 - vos_mem.h, 185
- vos_queueSend
 - vos_mem.c, 178
 - vos_mem.h, 185
- vos_select
 - posix/vos_sock.c, 198
 - vos_sock.h, 224
 - windows/vos_sock.c, 211
- vos_semaCreate
 - posix/vos_thread.c, 240
 - vos_thread.h, 259
 - windows/vos_thread.c, 250
- vos_semaDelete
 - posix/vos_thread.c, 240
 - vos_thread.h, 260
 - windows/vos_thread.c, 250
- vos_semaGive
 - posix/vos_thread.c, 240
 - vos_thread.h, 260
 - windows/vos_thread.c, 251
- vos_semaTake
 - posix/vos_thread.c, 241
 - vos_thread.h, 260
 - windows/vos_thread.c, 251
- vos_shared_mem.c, 188, 190
- vos_shared_mem.h, 192
 - vos_sharedClose, 193
 - vos_sharedOpen, 193
- vos_sharedClose
 - posix/vos_shared_mem.c, 189
 - vos_shared_mem.h, 193
 - windows/vos_shared_mem.c, 191
- vos_sharedOpen
 - posix/vos_shared_mem.c, 190
 - vos_shared_mem.h, 193
 - windows/vos_shared_mem.c, 191
- vos_sock.c, 194, 204
- vos_sock.h, 217
 - VOS_MAX_SOCKET_CNT, 219
 - VOS_TTL_MULTICAST, 219
 - vos_dottedIP, 219
 - vos_getInterfaces, 220
 - vos_htonl, 220
 - vos_htons, 220
 - vos_ipDotted, 222
 - vos_isMulticast, 222
 - vos_ntohl, 222
 - vos_ntohs, 224
 - vos_select, 224
 - vos_sockAccept, 224
 - vos_sockBind, 224
 - vos_sockClose, 225
 - vos_sockConnect, 225
 - vos_sockGetMAC, 226
 - vos_sockInit, 226
 - vos_sockJoinMC, 227
 - vos_sockLeaveMC, 227
 - vos_sockListen, 228
 - vos_sockOpenTCP, 228
 - vos_sockOpenUDP, 229
 - vos_sockReceiveTCP, 229
 - vos_sockReceiveUDP, 230
 - vos_sockSendTCP, 231
 - vos_sockSendUDP, 232
 - vos_sockSetMulticastIf, 233
 - vos_sockSetOptions, 233
 - vos_sockTerm, 234
- vos_sockAccept
 - posix/vos_sock.c, 198
 - vos_sock.h, 224
 - windows/vos_sock.c, 211
- vos_sockBind
 - posix/vos_sock.c, 199
 - vos_sock.h, 224
 - windows/vos_sock.c, 211
- vos_sockClose
 - posix/vos_sock.c, 199
 - vos_sock.h, 225
 - windows/vos_sock.c, 212
- vos_sockConnect
 - posix/vos_sock.c, 199
 - vos_sock.h, 225
 - windows/vos_sock.c, 212
- vos_sockGetMAC
 - posix/vos_sock.c, 200
 - vos_sock.h, 226
 - windows/vos_sock.c, 212
- vos_sockInit
 - posix/vos_sock.c, 200
 - vos_sock.h, 226

- windows/vos_sock.c, 212
- vos_sockJoinMC
 - posix/vos_sock.c, 200
 - vos_sock.h, 227
 - windows/vos_sock.c, 213
- vos_sockLeaveMC
 - posix/vos_sock.c, 200
 - vos_sock.h, 227
 - windows/vos_sock.c, 213
- vos_sockListen
 - posix/vos_sock.c, 201
 - vos_sock.h, 228
 - windows/vos_sock.c, 213
- vos_sockOpenTCP
 - posix/vos_sock.c, 201
 - vos_sock.h, 228
 - windows/vos_sock.c, 214
- vos_sockOpenUDP
 - posix/vos_sock.c, 201
 - vos_sock.h, 229
 - windows/vos_sock.c, 214
- vos_sockReceiveTCP
 - posix/vos_sock.c, 202
 - vos_sock.h, 229
 - windows/vos_sock.c, 214
- vos_sockReceiveUDP
 - posix/vos_sock.c, 202
 - vos_sock.h, 230
 - windows/vos_sock.c, 215
- vos_sockSendTCP
 - posix/vos_sock.c, 203
 - vos_sock.h, 231
 - windows/vos_sock.c, 215
- vos_sockSendUDP
 - posix/vos_sock.c, 203
 - vos_sock.h, 232
 - windows/vos_sock.c, 216
- vos_sockSetBuffer
 - posix/vos_sock.c, 203
 - windows/vos_sock.c, 216
- vos_sockSetMulticastIf
 - posix/vos_sock.c, 204
 - vos_sock.h, 233
 - windows/vos_sock.c, 216
- vos_sockSetOptions
 - posix/vos_sock.c, 204
 - vos_sock.h, 233
 - windows/vos_sock.c, 216
- vos_sockTerm
 - posix/vos_sock.c, 204
 - vos_sock.h, 234
 - windows/vos_sock.c, 217
- vos_strncpy
 - vos_mem.c, 179
 - vos_mem.h, 185
- vos_strncmp
 - vos_mem.c, 179
 - vos_mem.h, 186
- vos_subTime
 - posix/vos_thread.c, 241
 - vos_thread.h, 260
 - windows/vos_thread.c, 251
- vos_terminate
 - vos_utils.c, 268
 - vos_utils.h, 271
- vos_thread.c, 234, 243
- vos_thread.h, 253
 - vos_addTime, 255
 - vos_clearTime, 255
 - vos_cmpTime, 256
 - vos_cyclicThread, 256
 - vos_divTime, 256
 - vos_getTime, 257
 - vos_getTimeStamp, 257
 - vos_getUuid, 257
 - vos_mulTime, 257
 - vos_mutexCreate, 257
 - vos_mutexDelete, 258
 - vos_mutexLock, 258
 - vos_mutexTryLock, 259
 - vos_mutexUnlock, 259
 - vos_semaCreate, 259
 - vos_semaDelete, 260
 - vos_semaGive, 260
 - vos_semaTake, 260
 - vos_subTime, 260
 - vos_threadCreate, 261
 - vos_threadDelay, 262
 - vos_threadInit, 262
 - vos_threadIsActive, 262
 - vos_threadTerm, 263
 - vos_threadTerminate, 263
- vos_threadCreate
 - posix/vos_thread.c, 241
 - vos_thread.h, 261
 - windows/vos_thread.c, 251
- vos_threadDelay
 - posix/vos_thread.c, 242
 - vos_thread.h, 262
 - windows/vos_thread.c, 252
- vos_threadInit
 - posix/vos_thread.c, 242
 - vos_thread.h, 262
 - windows/vos_thread.c, 252
- vos_threadIsActive
 - posix/vos_thread.c, 242
 - vos_thread.h, 262
 - windows/vos_thread.c, 252
- vos_threadTerm
 - posix/vos_thread.c, 242
 - vos_thread.h, 263
 - windows/vos_thread.c, 253
- vos_threadTerminate
 - posix/vos_thread.c, 242
 - vos_thread.h, 263
 - windows/vos_thread.c, 253

- vos_types.h, 263
 - VOS_BLOCK_ERR, 266
 - VOS_ERR_T, 265
 - VOS_INIT_ERR, 265
 - VOS_INTEGRATION_ERR, 266
 - VOS_IO_ERR, 266
 - VOS_LOG_DBG, 266
 - VOS_LOG_ERROR, 266
 - VOS_LOG_INFO, 266
 - VOS_LOG_T, 266
 - VOS_LOG_WARNING, 266
 - VOS_MEM_ERR, 266
 - VOS_MUTEX_ERR, 266
 - VOS_NO_ERR, 265
 - VOS_NOCONN_ERR, 266
 - VOS_NODATA_ERR, 265
 - VOS_NOINIT_ERR, 265
 - VOS_PARAM_ERR, 265
 - VOS_PRINT_DBG_T, 265
 - VOS_QUEUE_ERR, 266
 - VOS_QUEUE_FULL_ERR, 266
 - VOS_SEMA_ERR, 266
 - VOS_SOCKET_ERR, 266
 - VOS_THREAD_ERR, 266
 - VOS_TIMEOUT_ERR, 265
 - VOS_UNKNOWN_ERR, 266
- vos_utils.c, 266
 - vos_crc32, 267
 - vos_init, 267
 - vos_initRuntimeConsts, 268
 - vos_isBigEndian, 268
 - vos_terminate, 268
- vos_utils.h, 268
 - INITFCS, 269
 - VOS_MAX_ERR_STR_SIZE, 269
 - VOS_MAX_FRMT_SIZE, 269
 - VOS_MAX_PRNT_STR_SIZE, 270
 - vos_crc32, 270
 - vos_init, 270
 - vos_terminate, 271
- windows/vos_private.h
 - vos_mutexLocalCreate, 188
 - vos_mutexLocalDelete, 188
- windows/vos_shared_mem.c
 - vos_sharedClose, 191
 - vos_sharedOpen, 191
- windows/vos_sock.c
 - recvmsg, 207
 - vos_dottedIP, 208
 - vos_getInterfaces, 208
 - vos_htonl, 208
 - vos_htons, 208
 - vos_ipDotted, 210
 - vos_isMulticast, 210
 - vos_ntohl, 210
 - vos_ntohs, 210
 - vos_select, 211
 - vos_sockAccept, 211
 - vos_sockBind, 211
 - vos_sockClose, 212
 - vos_sockConnect, 212
 - vos_sockGetMAC, 212
 - vos_sockInit, 212
 - vos_sockJoinMC, 213
 - vos_sockLeaveMC, 213
 - vos_sockListen, 213
 - vos_sockOpenTCP, 214
 - vos_sockOpenUDP, 214
 - vos_sockReceiveTCP, 214
 - vos_sockReceiveUDP, 215
 - vos_sockSendTCP, 215
 - vos_sockSendUDP, 216
 - vos_sockSetBuffer, 216
 - vos_sockSetMulticastIf, 216
 - vos_sockSetOptions, 216
 - vos_sockTerm, 217
- windows/vos_thread.c
 - NSECS_PER_USEC, 245
 - vos_addTime, 245
 - vos_clearTime, 246
 - vos_cmpTime, 246
 - vos_cyclicThread, 246
 - vos_divTime, 246
 - vos_getFreeThreadHandle, 246
 - vos_getTime, 248
 - vos_getTimeStamp, 248
 - vos_getUuid, 248
 - vos_mulTime, 248
 - vos_mutexCreate, 248
 - vos_mutexDelete, 249
 - vos_mutexLocalCreate, 249
 - vos_mutexLocalDelete, 249
 - vos_mutexLock, 249
 - vos_mutexTryLock, 250
 - vos_mutexUnlock, 250
 - vos_semaCreate, 250
 - vos_semaDelete, 250
 - vos_semaGive, 251
 - vos_semaTake, 251
 - vos_subTime, 251
 - vos_threadCreate, 251
 - vos_threadDelay, 252
 - vos_threadInit, 252
 - vos_threadIsActive, 252
 - vos_threadTerm, 253
 - vos_threadTerminate, 253